

**Milena (Olena)**  
User documentation 2.0 Id

Generated by Doxygen 1.7.1

Thu May 3 2012 19:52:38



# Contents

|          |  |          |
|----------|--|----------|
| <b>1</b> | <b>Documentation of milena</b>                       | <b>1</b> |
| 1.1      | Introduction . . . . .                               | 1        |
| 1.2      | Overview of Milena. . . . .                          | 1        |
| 1.3      | Copyright and License. . . . .                       | 2        |
| <b>2</b> | <b>Quick Reference Guide</b>                         | <b>3</b> |
| 2.1      | Installation . . . . .                               | 4        |
| 2.1.1    | Requirements . . . . .                               | 4        |
| 2.1.1.1  | To compile the user examples . . . . .               | 4        |
| 2.1.1.2  | To compile the documentation (Optional) . . . . .    | 4        |
| 2.1.1.3  | To develop in Olena . . . . .                        | 4        |
| 2.1.2    | Getting Olena . . . . .                              | 4        |
| 2.1.3    | Building Olena . . . . .                             | 4        |
| 2.2      | Foreword . . . . .                                   | 4        |
| 2.2.1    | Generality . . . . .                                 | 4        |
| 2.2.2    | Directory hierarchy . . . . .                        | 4        |
| 2.2.3    | Writing and compiling a program with Olena . . . . . | 4        |
| 2.3      | Site . . . . .                                       | 4        |
| 2.4      | Site set . . . . .                                   | 4        |
| 2.4.1    | Basic interface . . . . .                            | 4        |
| 2.4.2    | Optional interface . . . . .                         | 4        |
| 2.5      | Image . . . . .                                      | 5        |
| 2.5.1    | Definition . . . . .                                 | 5        |
| 2.5.2    | Possible image types . . . . .                       | 5        |
| 2.5.3    | Possible value types . . . . .                       | 5        |
| 2.5.4    | Domain . . . . .                                     | 5        |
| 2.5.5    | Border and extension . . . . .                       | 6        |
| 2.5.5.1  | Image border . . . . .                               | 6        |

|          |  |    |
|----------|--|----|
| 2.5.5.2  | Generality on image extension                | 6  |
| 2.5.5.3  | Different extensions                         | 6  |
| 2.5.6    | Interface                                    | 8  |
| 2.5.7    | Load and save images                         | 8  |
| 2.5.8    | Create an image                              | 8  |
| 2.5.9    | Access and modify values                     | 9  |
| 2.5.10   | Image size                                   | 9  |
| 2.6      | Structural elements: Window and neighborhood | 10 |
| 2.6.1    | Define an element                            | 10 |
| 2.6.1.1  | Window                                       | 10 |
| 2.6.1.2  | Neighborhood                                 | 10 |
| 2.6.1.3  | Custom structural elements                   | 10 |
| 2.6.1.4  | Conversion between Neighborhoods and Windows | 10 |
| 2.7      | Sites, psites and dpoints                    | 10 |
| 2.7.1    | Need for site                                | 10 |
| 2.7.2    | Need for psite                               | 10 |
| 2.7.3    | From psite to site                           | 11 |
| 2.7.4    | Dpoint                                       | 11 |
| 2.8      | Iterators                                    | 11 |
| 2.9      | Memory management                            | 12 |
| 2.10     | Basic routines                               | 12 |
| 2.10.1   | Fill   | 12 |
| 2.10.2   | Paste  | 12 |
| 2.10.3   | Blobs  | 13 |
| 2.10.4   | Logical not                                  | 13 |
| 2.10.5   | Compute                                      | 13 |
| 2.10.5.1 | Accumulators                                 | 13 |
| 2.10.5.2 | Example with labeling::compute()             | 13 |
| 2.10.6   | Working with parts of an image               | 14 |
| 2.10.6.1 | Restrict an image with a site set            | 14 |
| 2.10.6.2 | Restrict an image with a predicate           | 15 |
| 2.10.6.3 | Restrict an image with a C function          | 15 |
| 2.11     | Input / Output                               | 16 |
| 2.11.1   | ImageMagick                                  | 16 |
| 2.11.2   | GDCM   | 16 |
| 2.12     | Graphs and images                            | 16 |

---

|          |                                  |           |
|----------|----------------------------------|-----------|
| 2.12.1   | Description                      | 16        |
| 2.12.2   | Example                          | 16        |
| 2.13     | Useful global variables          | 19        |
| 2.14     | Useful macros                    | 19        |
| 2.14.1   | Variable declaration macros      | 19        |
| 2.14.2   | Iterator type macros             | 19        |
| 2.14.2.1 | Default iterator types           | 19        |
| 2.14.2.2 | Forward iterator types           | 19        |
| 2.14.2.3 | Backward iterators               | 19        |
| 2.14.2.4 | Graph iterators                  | 19        |
| 2.15     | Common Compilation Errors        | 19        |
| <b>3</b> | <b>Tutorial</b>                  | <b>21</b> |
| 3.1      | Welcome                          | 21        |
| 3.1.1    | How to learn Milena              | 21        |
| 3.1.2    | Obtaining the library            | 21        |
| 3.1.3    | Downloading the library          | 21        |
| 3.1.3.1  | Downloading from SVN             | 21        |
| 3.1.3.2  | Downloading packaged releases    | 21        |
| 3.1.4    | Join the mailing lists           | 22        |
| 3.1.5    | Directory structure              | 22        |
| 3.1.6    | Documentation                    | 22        |
| 3.1.7    | Community and Support            | 22        |
| 3.1.8    | Project status                   | 22        |
| 3.1.9    | A brief history of Milena        | 22        |
| 3.1.10   | Contacts                         | 22        |
| 3.2      | Installation                     | 22        |
| 3.2.1    | Bootstrap (SVN Sources)          | 22        |
| 3.2.2    | Configure                        | 22        |
| 3.2.3    | Install                          | 22        |
| 3.2.4    | Optional compilation             | 22        |
| 3.2.4.1  | Examples                         | 22        |
| 3.2.4.2  | Tools                            | 22        |
| 3.2.4.3  | Tests                            | 22        |
| 3.2.5    | Installation content             | 22        |
| 3.3      | Getting started with Milena      | 23        |
| 3.3.1    | Getting familiar with genericity | 23        |

|          |  |           |
|----------|--|-----------|
| 3.3.2    | First generic algorithm                | 23        |
| 3.3.3    | Compilation                            | 25        |
| 3.3.3.1  | Include path                           | 25        |
| 3.3.3.2  | Library linking                        | 25        |
| 3.3.3.3  | Disable Debug                          | 25        |
| 3.3.3.4  | Compiler optimization flags            | 25        |
| 3.3.4    | Debug hints                            | 25        |
| 3.3.4.1  | Using assertions and GDB               | 25        |
| 3.3.4.2  | Traces                                 | 25        |
| 3.3.4.3  | Debug routines                         | 25        |
| 3.4      | Data representation                    | 26        |
| 3.4.1    | Sites                                  | 26        |
| 3.4.2    | Site sets                              | 26        |
| 3.4.2.1  | Creating a site set                    | 26        |
| 3.4.2.2  | Getting access to sites                | 27        |
| 3.4.3    | Images                                 | 27        |
| 3.4.3.1  | Creating an image                      | 27        |
| 3.4.3.2  | Reading an image from a file           | 27        |
| 3.4.3.3  | Accessing data                         | 27        |
| 3.5      | Load and save images                   | 27        |
| 3.6      | Create your first image                | 27        |
| 3.7      | Read and write images                  | 28        |
| 3.8      | Regions of interest                    | 28        |
| 3.8.1    | Image domain restricted by a site set  | 29        |
| 3.8.2    | Image domain restricted by a function  | 29        |
| 3.8.3    | Image domain restricted by a mask      | 29        |
| 3.8.4    | Image domain restricted by a predicate | 29        |
| <b>4</b> | <b>Module Index</b>                    | <b>31</b> |
| 4.1      | Modules                                | 31        |
| <b>5</b> | <b>Namespace Index</b>                 | <b>33</b> |
| 5.1      | Namespace List                         | 33        |
| <b>6</b> | <b>Class Index</b>                     | <b>37</b> |
| 6.1      | Class Hierarchy                        | 37        |
| <b>7</b> | <b>Class Index</b>                     | <b>81</b> |

---

|          |                             |           |
|----------|-----------------------------|-----------|
| 7.1      | Class List                  | 81        |
| <b>8</b> | <b>Module Documentation</b> | <b>91</b> |
| 8.1      | On site sets                | 91        |
| 8.1.1    | Detailed Description        | 91        |
| 8.2      | On images                   | 91        |
| 8.2.1    | Detailed Description        | 92        |
| 8.3      | On values                   | 92        |
| 8.3.1    | Detailed Description        | 94        |
| 8.4      | Multiple accumulators       | 94        |
| 8.4.1    | Detailed Description        | 94        |
| 8.5      | Graphes                     | 94        |
| 8.5.1    | Detailed Description        | 95        |
| 8.6      | Images                      | 95        |
| 8.6.1    | Detailed Description        | 95        |
| 8.7      | Basic types                 | 95        |
| 8.7.1    | Detailed Description        | 96        |
| 8.8      | Image morphers              | 96        |
| 8.9      | Values morphers             | 96        |
| 8.9.1    | Detailed Description        | 97        |
| 8.10     | Domain morphers             | 97        |
| 8.10.1   | Detailed Description        | 97        |
| 8.11     | Identity morphers           | 98        |
| 8.11.1   | Detailed Description        | 98        |
| 8.12     | Types                       | 98        |
| 8.12.1   | Detailed Description        | 99        |
| 8.13     | Accumulators                | 99        |
| 8.13.1   | Detailed Description        | 99        |
| 8.14     | Routines                    | 99        |
| 8.15     | Canvas                      | 99        |
| 8.16     | Functions                   | 100       |
| 8.16.1   | Detailed Description        | 101       |
| 8.17     | Neighborhoods               | 101       |
| 8.17.1   | Detailed Description        | 101       |
| 8.18     | 1D neighborhoods            | 101       |
| 8.18.1   | Detailed Description        | 102       |
| 8.18.2   | Typedef Documentation       | 102       |

|          |                        |     |
|----------|------------------------|-----|
| 8.18.2.1 | neighb1d               | 102 |
| 8.18.3   | Function Documentation | 102 |
| 8.18.3.1 | c2                     | 102 |
| 8.19     | 2D neighborhoods       | 102 |
| 8.19.1   | Detailed Description   | 103 |
| 8.19.2   | Typedef Documentation  | 103 |
| 8.19.2.1 | neighb2d               | 103 |
| 8.19.3   | Function Documentation | 103 |
| 8.19.3.1 | c2_col                 | 103 |
| 8.19.3.2 | c2_row                 | 103 |
| 8.19.3.3 | c4                     | 103 |
| 8.19.3.4 | c8                     | 104 |
| 8.20     | 3D neighborhoods       | 104 |
| 8.20.1   | Detailed Description   | 105 |
| 8.20.2   | Typedef Documentation  | 105 |
| 8.20.2.1 | neighb3d               | 105 |
| 8.20.3   | Function Documentation | 105 |
| 8.20.3.1 | c18                    | 105 |
| 8.20.3.2 | c26                    | 105 |
| 8.20.3.3 | c2_3d_sli              | 106 |
| 8.20.3.4 | c4_3d                  | 106 |
| 8.20.3.5 | c6                     | 107 |
| 8.20.3.6 | c8_3d                  | 107 |
| 8.21     | Site sets              | 108 |
| 8.21.1   | Detailed Description   | 108 |
| 8.22     | Basic types            | 108 |
| 8.22.1   | Detailed Description   | 108 |
| 8.23     | Graph based            | 109 |
| 8.23.1   | Detailed Description   | 109 |
| 8.24     | Complex based          | 109 |
| 8.24.1   | Detailed Description   | 109 |
| 8.25     | Sparse types           | 109 |
| 8.25.1   | Detailed Description   | 110 |
| 8.26     | Queue based            | 110 |
| 8.26.1   | Detailed Description   | 110 |
| 8.27     | Utilities              | 110 |



|  |            |
|--|------------|
| 8.27.1 Detailed Description . . . . .      | 111        |
| 8.28 Windows . . . . .                     | 111        |
| 8.28.1 Detailed Description . . . . .      | 112        |
| 8.29 1D windows . . . . .                  | 112        |
| 8.29.1 Detailed Description . . . . .      | 112        |
| 8.29.2 Typedef Documentation . . . . .     | 112        |
| 8.29.2.1 segment1d . . . . .               | 112        |
| 8.29.2.2 window1d . . . . .                | 113        |
| 8.30 2D windows . . . . .                  | 113        |
| 8.30.1 Detailed Description . . . . .      | 114        |
| 8.30.2 Typedef Documentation . . . . .     | 114        |
| 8.30.2.1 disk2d . . . . .                  | 114        |
| 8.30.2.2 hline2d . . . . .                 | 114        |
| 8.30.2.3 vline2d . . . . .                 | 114        |
| 8.30.2.4 window2d . . . . .                | 115        |
| 8.30.3 Function Documentation . . . . .    | 115        |
| 8.30.3.1 win_c4p . . . . .                 | 115        |
| 8.30.3.2 win_c8p . . . . .                 | 115        |
| 8.31 3D windows . . . . .                  | 115        |
| 8.31.1 Detailed Description . . . . .      | 116        |
| 8.31.2 Typedef Documentation . . . . .     | 116        |
| 8.31.2.1 sline3d . . . . .                 | 116        |
| 8.31.2.2 sphere3d . . . . .                | 117        |
| 8.31.2.3 window3d . . . . .                | 117        |
| 8.31.3 Function Documentation . . . . .    | 117        |
| 8.31.3.1 win_c4p_3d . . . . .              | 117        |
| 8.31.3.2 win_c8p_3d . . . . .              | 117        |
| 8.32 N-D windows . . . . .                 | 118        |
| 8.32.1 Detailed Description . . . . .      | 118        |
| 8.33 Multiple windows . . . . .            | 118        |
| 8.33.1 Detailed Description . . . . .      | 118        |
| 8.34 v2w2v functions . . . . .             | 118        |
| 8.35 v2w_w2v functions . . . . .           | 119        |
| 8.36 vv2b functions . . . . .              | 119        |
| <b>9 Namespace Documentation</b> . . . . . | <b>121</b> |
| 9.1 mln Namespace Reference . . . . .      | 121        |

---

|          |                                  |     |
|----------|----------------------------------|-----|
| 9.1.1    | Detailed Description             | 142 |
| 9.1.2    | Typedef Documentation            | 145 |
| 9.1.2.1  | bin_1complex_image2d             | 145 |
| 9.1.2.2  | bin_2complex_image3df            | 145 |
| 9.1.2.3  | box1d                            | 145 |
| 9.1.2.4  | box2d                            | 145 |
| 9.1.2.5  | box2d_h                          | 146 |
| 9.1.2.6  | box3d                            | 146 |
| 9.1.2.7  | discrete_plane_1complex_geometry | 146 |
| 9.1.2.8  | discrete_plane_2complex_geometry | 146 |
| 9.1.2.9  | dpoint1d                         | 146 |
| 9.1.2.10 | dpoint2d                         | 146 |
| 9.1.2.11 | dpoint2d_h                       | 146 |
| 9.1.2.12 | dpoint3d                         | 147 |
| 9.1.2.13 | float_2complex_image3df          | 147 |
| 9.1.2.14 | int_u8_1complex_image2d          | 147 |
| 9.1.2.15 | int_u8_2complex_image2d          | 147 |
| 9.1.2.16 | int_u8_2complex_image3df         | 147 |
| 9.1.2.17 | p_run2d                          | 147 |
| 9.1.2.18 | p_runs2d                         | 147 |
| 9.1.2.19 | point1d                          | 148 |
| 9.1.2.20 | point1df                         | 148 |
| 9.1.2.21 | point2d                          | 148 |
| 9.1.2.22 | point2d_h                        | 148 |
| 9.1.2.23 | point2df                         | 148 |
| 9.1.2.24 | point3d                          | 148 |
| 9.1.2.25 | point3df                         | 148 |
| 9.1.2.26 | rgb8_2complex_image3df           | 148 |
| 9.1.2.27 | space_2complex_geometry          | 149 |
| 9.1.2.28 | unsigned_2complex_image3df       | 149 |
| 9.1.2.29 | vec2d_d                          | 149 |
| 9.1.2.30 | vec2d_f                          | 149 |
| 9.1.2.31 | vec3d_d                          | 149 |
| 9.1.2.32 | vec3d_f                          | 149 |
| 9.1.2.33 | w_window1d_float                 | 149 |
| 9.1.2.34 | w_window1d_int                   | 149 |

---

|          |                              |     |
|----------|------------------------------|-----|
| 9.1.2.35 | w_window2d_float             | 150 |
| 9.1.2.36 | w_window2d_int               | 150 |
| 9.1.2.37 | w_window3d_float             | 150 |
| 9.1.2.38 | w_window3d_int               | 150 |
| 9.1.3    | Function Documentation       | 150 |
| 9.1.3.1  | a_point_of                   | 150 |
| 9.1.3.2  | apply_p2p                    | 150 |
| 9.1.3.3  | apply_p2p                    | 150 |
| 9.1.3.4  | compose                      | 151 |
| 9.1.3.5  | duplicate                    | 151 |
| 9.1.3.6  | extend                       | 151 |
| 9.1.3.7  | extend                       | 151 |
| 9.1.3.8  | extend                       | 152 |
| 9.1.3.9  | implies                      | 152 |
| 9.1.3.10 | initialize                   | 152 |
| 9.1.3.11 | larger_than                  | 152 |
| 9.1.3.12 | make_debug_graph_image       | 152 |
| 9.1.3.13 | mln_exact                    | 153 |
| 9.1.3.14 | mln_gen_complex_neighborhood | 153 |
| 9.1.3.15 | mln_gen_complex_neighborhood | 153 |
| 9.1.3.16 | mln_gen_complex_neighborhood | 153 |
| 9.1.3.17 | mln_gen_complex_neighborhood | 153 |
| 9.1.3.18 | mln_gen_complex_neighborhood | 153 |
| 9.1.3.19 | mln_gen_complex_neighborhood | 154 |
| 9.1.3.20 | mln_gen_complex_window       | 154 |
| 9.1.3.21 | mln_gen_complex_window       | 154 |
| 9.1.3.22 | mln_gen_complex_window       | 154 |
| 9.1.3.23 | mln_gen_complex_window       | 154 |
| 9.1.3.24 | mln_gen_complex_window       | 154 |
| 9.1.3.25 | mln_gen_complex_window       | 154 |
| 9.1.3.26 | mln_gen_complex_window_p     | 154 |
| 9.1.3.27 | mln_gen_complex_window_p     | 154 |
| 9.1.3.28 | mln_gen_complex_window_p     | 155 |
| 9.1.3.29 | mln_gen_complex_window_p     | 155 |
| 9.1.3.30 | mln_gen_complex_window_p     | 155 |
| 9.1.3.31 | mln_gen_complex_window_p     | 155 |

---

|          |                                   |     |
|----------|-----------------------------------|-----|
| 9.1.3.32 | <code>mln_regular</code>          | 155 |
| 9.1.3.33 | <code>mln_trait_op_geq</code>     | 155 |
| 9.1.3.34 | <code>mln_trait_op_greater</code> | 155 |
| 9.1.3.35 | <code>mln_trait_op_leq</code>     | 156 |
| 9.1.3.36 | <code>mln_trait_op_neq</code>     | 156 |
| 9.1.3.37 | <code>operator!=</code>           | 156 |
| 9.1.3.38 | <code>operator!=</code>           | 157 |
| 9.1.3.39 | <code>operator*</code>            | 157 |
| 9.1.3.40 | <code>operator++</code>           | 157 |
| 9.1.3.41 | <code>operator-</code>            | 157 |
| 9.1.3.42 | <code>operator-</code>            | 157 |
| 9.1.3.43 | <code>operator--</code>           | 158 |
| 9.1.3.44 | <code>operator&lt;</code>         | 158 |
| 9.1.3.45 | <code>operator&lt;</code>         | 158 |
| 9.1.3.46 | <code>operator&lt;</code>         | 158 |
| 9.1.3.47 | <code>operator&lt;&lt;</code>     | 159 |
| 9.1.3.48 | <code>operator&lt;&lt;</code>     | 159 |
| 9.1.3.49 | <code>operator&lt;&lt;</code>     | 159 |
| 9.1.3.50 | <code>operator&lt;&lt;</code>     | 159 |
| 9.1.3.51 | <code>operator&lt;=</code>        | 159 |
| 9.1.3.52 | <code>operator&lt;=</code>        | 159 |
| 9.1.3.53 | <code>operator&lt;=</code>        | 159 |
| 9.1.3.54 | <code>operator&lt;=</code>        | 160 |
| 9.1.3.55 | <code>operator&lt;=</code>        | 160 |
| 9.1.3.56 | <code>operator==</code>           | 160 |
| 9.1.3.57 | <code>operator==</code>           | 160 |
| 9.1.3.58 | <code>operator==</code>           | 161 |
| 9.1.3.59 | <code>operator==</code>           | 161 |
| 9.1.3.60 | <code>operator==</code>           | 161 |
| 9.1.3.61 | <code>operator==</code>           | 161 |
| 9.1.3.62 | <code>operator==</code>           | 162 |
| 9.1.3.63 | <code>operator </code>            | 162 |
| 9.1.3.64 | <code>operator </code>            | 162 |
| 9.1.3.65 | <code>operator </code>            | 162 |
| 9.1.3.66 | <code>operator </code>            | 162 |
| 9.1.3.67 | <code>operator </code>            | 163 |

---

|          |  |     |
|----------|--|-----|
| 9.1.3.68 | operator                                   | 163 |
| 9.1.3.69 | primary                                    | 163 |
| 9.1.3.70 | ptransform                                 | 163 |
| 9.1.4    | Variable Documentation                     | 163 |
| 9.1.4.1  | before                                     | 163 |
| 9.1.4.2  | sagittal_dec                               | 163 |
| 9.1.4.3  | up   | 164 |
| 9.2      | mln::accu Namespace Reference              | 164 |
| 9.2.1    | Detailed Description                       | 166 |
| 9.2.2    | Function Documentation                     | 166 |
| 9.2.2.1  | compute                                    | 166 |
| 9.2.2.2  | line                                       | 166 |
| 9.2.2.3  | mln_meta_accu_result                       | 167 |
| 9.2.2.4  | take                                       | 167 |
| 9.3      | mln::accu::image Namespace Reference       | 167 |
| 9.3.1    | Detailed Description                       | 167 |
| 9.4      | mln::accu::impl Namespace Reference        | 167 |
| 9.4.1    | Detailed Description                       | 168 |
| 9.5      | mln::accu::logic Namespace Reference       | 168 |
| 9.5.1    | Detailed Description                       | 168 |
| 9.6      | mln::accu::math Namespace Reference        | 168 |
| 9.6.1    | Detailed Description                       | 169 |
| 9.7      | mln::accu::meta::logic Namespace Reference | 169 |
| 9.7.1    | Detailed Description                       | 169 |
| 9.8      | mln::accu::meta::math Namespace Reference  | 169 |
| 9.8.1    | Detailed Description                       | 170 |
| 9.9      | mln::accu::meta::shape Namespace Reference | 170 |
| 9.9.1    | Detailed Description                       | 170 |
| 9.10     | mln::accu::meta::stat Namespace Reference  | 170 |
| 9.10.1   | Detailed Description                       | 171 |
| 9.11     | mln::accu::shape Namespace Reference       | 171 |
| 9.11.1   | Detailed Description                       | 171 |
| 9.12     | mln::accu::stat Namespace Reference        | 171 |
| 9.12.1   | Detailed Description                       | 173 |
| 9.12.2   | Function Documentation                     | 173 |
| 9.12.2.1 | operator==                                 | 173 |

|           |   |     |
|-----------|---|-----|
| 9.13      | <code>mln::algebra</code> Namespace Reference | 173 |
| 9.13.1    | Detailed Description                          | 174 |
| 9.13.2    | Function Documentation                        | 174 |
| 9.13.2.1  | <code>ldlt_decomp</code>                      | 174 |
| 9.13.2.2  | <code>ldlt_solve</code>                       | 174 |
| 9.13.2.3  | <code>operator*</code>                        | 174 |
| 9.13.2.4  | <code>vprod</code>                            | 174 |
| 9.14      | <code>mln::arith</code> Namespace Reference   | 175 |
| 9.14.1    | Detailed Description                          | 177 |
| 9.14.2    | Function Documentation                        | 177 |
| 9.14.2.1  | <code>diff_abs</code>                         | 177 |
| 9.14.2.2  | <code>div</code>                              | 178 |
| 9.14.2.3  | <code>div_cst</code>                          | 178 |
| 9.14.2.4  | <code>div_inplace</code>                      | 178 |
| 9.14.2.5  | <code>min</code>                              | 179 |
| 9.14.2.6  | <code>min_inplace</code>                      | 179 |
| 9.14.2.7  | <code>minus</code>                            | 179 |
| 9.14.2.8  | <code>minus</code>                            | 180 |
| 9.14.2.9  | <code>minus</code>                            | 180 |
| 9.14.2.10 | <code>minus_cst</code>                        | 181 |
| 9.14.2.11 | <code>minus_cst</code>                        | 181 |
| 9.14.2.12 | <code>minus_cst_inplace</code>                | 181 |
| 9.14.2.13 | <code>minus_inplace</code>                    | 182 |
| 9.14.2.14 | <code>plus</code>                             | 182 |
| 9.14.2.15 | <code>plus</code>                             | 183 |
| 9.14.2.16 | <code>plus</code>                             | 183 |
| 9.14.2.17 | <code>plus_cst</code>                         | 183 |
| 9.14.2.18 | <code>plus_cst</code>                         | 184 |
| 9.14.2.19 | <code>plus_cst</code>                         | 184 |
| 9.14.2.20 | <code>plus_cst_inplace</code>                 | 185 |
| 9.14.2.21 | <code>plus_inplace</code>                     | 185 |
| 9.14.2.22 | <code>revert</code>                           | 185 |
| 9.14.2.23 | <code>revert_inplace</code>                   | 186 |
| 9.14.2.24 | <code>times</code>                            | 186 |
| 9.14.2.25 | <code>times_cst</code>                        | 186 |
| 9.14.2.26 | <code>times_inplace</code>                    | 187 |

---

|          |  |     |
|----------|--|-----|
| 9.15     | <a href="#">mln::arith::impl Namespace Reference</a>           | 187 |
| 9.15.1   | <a href="#">Detailed Description</a>                           | 187 |
| 9.16     | <a href="#">mln::arith::impl::generic Namespace Reference</a>  | 188 |
| 9.16.1   | <a href="#">Detailed Description</a>                           | 188 |
| 9.17     | <a href="#">mln::binarization Namespace Reference</a>          | 188 |
| 9.17.1   | <a href="#">Detailed Description</a>                           | 188 |
| 9.17.2   | <a href="#">Function Documentation</a>                         | 188 |
| 9.17.2.1 | <a href="#">binarization</a>                                   | 188 |
| 9.17.2.2 | <a href="#">threshold</a>                                      | 189 |
| 9.18     | <a href="#">mln::border Namespace Reference</a>                | 189 |
| 9.18.1   | <a href="#">Detailed Description</a>                           | 189 |
| 9.18.2   | <a href="#">Function Documentation</a>                         | 190 |
| 9.18.2.1 | <a href="#">adjust</a>   | 190 |
| 9.18.2.2 | <a href="#">duplicate</a>                                      | 190 |
| 9.18.2.3 | <a href="#">equalize</a>                                       | 190 |
| 9.18.2.4 | <a href="#">fill</a>   | 191 |
| 9.18.2.5 | <a href="#">find</a>   | 191 |
| 9.18.2.6 | <a href="#">get</a>  | 191 |
| 9.18.2.7 | <a href="#">mirror</a>   | 192 |
| 9.18.2.8 | <a href="#">resize</a>   | 192 |
| 9.19     | <a href="#">mln::border::impl Namespace Reference</a>          | 192 |
| 9.19.1   | <a href="#">Detailed Description</a>                           | 193 |
| 9.20     | <a href="#">mln::border::impl::generic Namespace Reference</a> | 193 |
| 9.20.1   | <a href="#">Detailed Description</a>                           | 193 |
| 9.21     | <a href="#">mln::canvas Namespace Reference</a>                | 193 |
| 9.21.1   | <a href="#">Detailed Description</a>                           | 194 |
| 9.21.2   | <a href="#">Function Documentation</a>                         | 194 |
| 9.21.2.1 | <a href="#">distance_front</a>                                 | 194 |
| 9.21.2.2 | <a href="#">distance_geodesic</a>                              | 194 |
| 9.22     | <a href="#">mln::canvas::browsing Namespace Reference</a>      | 194 |
| 9.22.1   | <a href="#">Detailed Description</a>                           | 195 |
| 9.23     | <a href="#">mln::canvas::impl Namespace Reference</a>          | 195 |
| 9.23.1   | <a href="#">Detailed Description</a>                           | 195 |
| 9.24     | <a href="#">mln::canvas::labeling Namespace Reference</a>      | 195 |
| 9.24.1   | <a href="#">Detailed Description</a>                           | 196 |
| 9.24.2   | <a href="#">Function Documentation</a>                         | 196 |

|           |   |     |
|-----------|---|-----|
| 9.24.2.1  | blobs   | 196 |
| 9.25      | mln::canvas::labeling::impl Namespace Reference | 196 |
| 9.25.1    | Detailed Description                            | 197 |
| 9.26      | mln::canvas::morpho Namespace Reference         | 197 |
| 9.26.1    | Detailed Description                            | 197 |
| 9.27      | mln::convert Namespace Reference                | 197 |
| 9.27.1    | Detailed Description                            | 199 |
| 9.27.2    | Function Documentation                          | 199 |
| 9.27.2.1  | from_to   | 199 |
| 9.27.2.2  | from_to   | 200 |
| 9.27.2.3  | from_to   | 200 |
| 9.27.2.4  | from_to   | 200 |
| 9.27.2.5  | mln_image_from_grid                             | 200 |
| 9.27.2.6  | mln_image_from_grid                             | 200 |
| 9.27.2.7  | mln_image_from_grid                             | 200 |
| 9.27.2.8  | mln_image_from_grid                             | 200 |
| 9.27.2.9  | mln_window                                      | 200 |
| 9.27.2.10 | to  | 200 |
| 9.27.2.11 | to_dpoint                                       | 201 |
| 9.27.2.12 | to_fun  | 201 |
| 9.27.2.13 | to_image  | 201 |
| 9.27.2.14 | to_p_array                                      | 201 |
| 9.27.2.15 | to_p_array                                      | 201 |
| 9.27.2.16 | to_p_array                                      | 201 |
| 9.27.2.17 | to_p_set  | 201 |
| 9.27.2.18 | to_p_set  | 202 |
| 9.27.2.19 | to_p_set  | 202 |
| 9.27.2.20 | to_p_set  | 202 |
| 9.27.2.21 | to_p_set  | 202 |
| 9.27.2.22 | to_qimage                                       | 202 |
| 9.27.2.23 | to_upper_window                                 | 202 |
| 9.27.2.24 | to_upper_window                                 | 203 |
| 9.27.2.25 | to_window                                       | 203 |
| 9.27.2.26 | to_window                                       | 203 |
| 9.27.2.27 | to_window                                       | 203 |
| 9.27.3    | Variable Documentation                          | 203 |



---

|           |                                       |     |
|-----------|---------------------------------------|-----|
| 9.27.3.1  | to_fun                                | 203 |
| 9.28      | mln::data Namespace Reference         | 203 |
| 9.28.1    | Detailed Description                  | 205 |
| 9.28.2    | Function Documentation                | 206 |
| 9.28.2.1  | abs                                   | 206 |
| 9.28.2.2  | abs_inplace                           | 206 |
| 9.28.2.3  | apply                                 | 206 |
| 9.28.2.4  | compute                               | 206 |
| 9.28.2.5  | compute                               | 207 |
| 9.28.2.6  | convert                               | 207 |
| 9.28.2.7  | fast_median                           | 207 |
| 9.28.2.8  | fill                                  | 208 |
| 9.28.2.9  | fill_with_image                       | 208 |
| 9.28.2.10 | fill_with_value                       | 209 |
| 9.28.2.11 | median                                | 209 |
| 9.28.2.12 | mln_meta_accu_result                  | 209 |
| 9.28.2.13 | paste                                 | 209 |
| 9.28.2.14 | paste_without_localization            | 210 |
| 9.28.2.15 | replace                               | 210 |
| 9.28.2.16 | saturate                              | 210 |
| 9.28.2.17 | saturate                              | 211 |
| 9.28.2.18 | saturate_inplace                      | 211 |
| 9.28.2.19 | sort_offsets_increasing               | 211 |
| 9.28.2.20 | sort_psites_decreasing                | 212 |
| 9.28.2.21 | sort_psites_increasing                | 212 |
| 9.28.2.22 | stretch                               | 212 |
| 9.28.2.23 | to_enc                                | 213 |
| 9.28.2.24 | transform                             | 213 |
| 9.28.2.25 | transform                             | 213 |
| 9.28.2.26 | transform_inplace                     | 214 |
| 9.28.2.27 | transform_inplace                     | 214 |
| 9.28.2.28 | update                                | 214 |
| 9.28.2.29 | wrap                                  | 215 |
| 9.29      | mln::data::approx Namespace Reference | 215 |
| 9.29.1    | Detailed Description                  | 215 |
| 9.29.2    | Function Documentation                | 215 |

|          |  |     |
|----------|--|-----|
| 9.29.2.1 | median                                       | 215 |
| 9.29.2.2 | median                                       | 216 |
| 9.29.2.3 | median                                       | 216 |
| 9.30     | mln::data::approx::impl Namespace Reference  | 217 |
| 9.30.1   | Detailed Description                         | 217 |
| 9.31     | mln::data::impl Namespace Reference          | 217 |
| 9.31.1   | Detailed Description                         | 217 |
| 9.31.2   | Function Documentation                       | 218 |
| 9.31.2.1 | paste_without_localization_fast              | 218 |
| 9.31.2.2 | paste_without_localization_fastest           | 218 |
| 9.31.2.3 | paste_without_localization_lines             | 219 |
| 9.31.2.4 | stretch                                      | 219 |
| 9.31.2.5 | transform_inplace_lowq                       | 219 |
| 9.31.2.6 | update_fastest                               | 220 |
| 9.32     | mln::data::impl::generic Namespace Reference | 220 |
| 9.32.1   | Detailed Description                         | 221 |
| 9.32.2   | Function Documentation                       | 221 |
| 9.32.2.1 | fill_with_image                              | 221 |
| 9.32.2.2 | fill_with_value                              | 221 |
| 9.32.2.3 | paste  | 221 |
| 9.32.2.4 | transform                                    | 222 |
| 9.32.2.5 | transform                                    | 222 |
| 9.32.2.6 | transform_inplace                            | 222 |
| 9.32.2.7 | transform_inplace                            | 222 |
| 9.32.2.8 | update                                       | 223 |
| 9.33     | mln::data::naive Namespace Reference         | 223 |
| 9.33.1   | Detailed Description                         | 223 |
| 9.33.2   | Function Documentation                       | 223 |
| 9.33.2.1 | median                                       | 223 |
| 9.34     | mln::data::naive::impl Namespace Reference   | 224 |
| 9.34.1   | Detailed Description                         | 224 |
| 9.35     | mln::debug Namespace Reference               | 224 |
| 9.35.1   | Detailed Description                         | 226 |
| 9.35.2   | Function Documentation                       | 226 |
| 9.35.2.1 | draw_graph                                   | 226 |
| 9.35.2.2 | draw_graph                                   | 226 |

|           |   |     |
|-----------|---|-----|
| 9.35.2.3  | draw_graph                                      | 227 |
| 9.35.2.4  | filename  | 227 |
| 9.35.2.5  | format  | 227 |
| 9.35.2.6  | format  | 227 |
| 9.35.2.7  | format  | 227 |
| 9.35.2.8  | format  | 228 |
| 9.35.2.9  | iota  | 228 |
| 9.35.2.10 | mosaic  | 228 |
| 9.35.2.11 | println   | 228 |
| 9.35.2.12 | println   | 228 |
| 9.35.2.13 | println_with_border                             | 229 |
| 9.35.2.14 | put_word  | 229 |
| 9.35.2.15 | slices_2d                                       | 229 |
| 9.35.2.16 | slices_2d                                       | 229 |
| 9.35.2.17 | superpose                                       | 229 |
| 9.35.2.18 | superpose                                       | 229 |
| 9.35.2.19 | z_order   | 230 |
| 9.36      | mln::debug::impl Namespace Reference            | 230 |
| 9.36.1    | Detailed Description                            | 230 |
| 9.37      | mln::def Namespace Reference                    | 230 |
| 9.37.1    | Detailed Description                            | 231 |
| 9.37.2    | Typedef Documentation                           | 231 |
| 9.37.2.1  | coord   | 231 |
| 9.37.2.2  | coordf  | 231 |
| 9.37.3    | Enumeration Type Documentation                  | 231 |
| 9.37.3.1  | "@21  | 231 |
| 9.38      | mln::display Namespace Reference                | 231 |
| 9.38.1    | Detailed Description                            | 232 |
| 9.39      | mln::display::impl Namespace Reference          | 232 |
| 9.39.1    | Detailed Description                            | 232 |
| 9.40      | mln::display::impl::generic Namespace Reference | 232 |
| 9.40.1    | Detailed Description                            | 232 |
| 9.41      | mln::doc Namespace Reference                    | 232 |
| 9.41.1    | Detailed Description                            | 233 |
| 9.42      | mln::draw Namespace Reference                   | 234 |
| 9.42.1    | Detailed Description                            | 234 |

|          |                                      |     |
|----------|--------------------------------------|-----|
| 9.42.2   | Function Documentation               | 234 |
| 9.42.2.1 | box                                  | 234 |
| 9.42.2.2 | box_plain                            | 235 |
| 9.42.2.3 | dashed_line                          | 235 |
| 9.42.2.4 | line                                 | 235 |
| 9.42.2.5 | plot                                 | 236 |
| 9.42.2.6 | polygon                              | 236 |
| 9.42.2.7 | site_set                             | 237 |
| 9.43     | mln::estim Namespace Reference       | 237 |
| 9.43.1   | Detailed Description                 | 237 |
| 9.43.2   | Function Documentation               | 238 |
| 9.43.2.1 | mean                                 | 238 |
| 9.43.2.2 | mean                                 | 238 |
| 9.43.2.3 | min_max                              | 238 |
| 9.43.2.4 | sum                                  | 239 |
| 9.43.2.5 | sum                                  | 239 |
| 9.44     | mln::extension Namespace Reference   | 239 |
| 9.44.1   | Detailed Description                 | 240 |
| 9.44.2   | Function Documentation               | 240 |
| 9.44.2.1 | adjust                               | 240 |
| 9.44.2.2 | adjust                               | 240 |
| 9.44.2.3 | adjust                               | 240 |
| 9.44.2.4 | adjust                               | 241 |
| 9.44.2.5 | adjust_duplicate                     | 241 |
| 9.44.2.6 | adjust_fill                          | 241 |
| 9.44.2.7 | duplicate                            | 241 |
| 9.44.2.8 | fill                                 | 241 |
| 9.45     | mln::fun Namespace Reference         | 242 |
| 9.45.1   | Detailed Description                 | 243 |
| 9.46     | mln::fun::access Namespace Reference | 243 |
| 9.46.1   | Detailed Description                 | 243 |
| 9.47     | mln::fun::i2v Namespace Reference    | 243 |
| 9.47.1   | Detailed Description                 | 243 |
| 9.47.2   | Function Documentation               | 244 |
| 9.47.2.1 | operator<<                           | 244 |
| 9.48     | mln::fun::n2v Namespace Reference    | 244 |

---

|  |     |
|--|-----|
| 9.48.1 Detailed Description . . . . .                | 244 |
| 9.49 mln::fun::p2b Namespace Reference . . . . .     | 244 |
| 9.49.1 Detailed Description . . . . .                | 244 |
| 9.50 mln::fun::p2p Namespace Reference . . . . .     | 244 |
| 9.50.1 Detailed Description . . . . .                | 245 |
| 9.51 mln::fun::p2v Namespace Reference . . . . .     | 245 |
| 9.51.1 Detailed Description . . . . .                | 245 |
| 9.52 mln::fun::stat Namespace Reference . . . . .    | 245 |
| 9.52.1 Detailed Description . . . . .                | 245 |
| 9.53 mln::fun::v2b Namespace Reference . . . . .     | 245 |
| 9.53.1 Detailed Description . . . . .                | 245 |
| 9.54 mln::fun::v2i Namespace Reference . . . . .     | 245 |
| 9.54.1 Detailed Description . . . . .                | 246 |
| 9.55 mln::fun::v2v Namespace Reference . . . . .     | 246 |
| 9.55.1 Detailed Description . . . . .                | 247 |
| 9.55.2 Variable Documentation . . . . .              | 247 |
| 9.55.2.1 f_hsi_to_rgb_3x8 . . . . .                  | 247 |
| 9.55.2.2 f_hsl_to_rgb_3x8 . . . . .                  | 247 |
| 9.55.2.3 f_rgb_to_hsi_f . . . . .                    | 247 |
| 9.55.2.4 f_rgb_to_hsl_f . . . . .                    | 247 |
| 9.56 mln::fun::v2w2v Namespace Reference . . . . .   | 247 |
| 9.56.1 Detailed Description . . . . .                | 247 |
| 9.57 mln::fun::v2w_w2v Namespace Reference . . . . . | 247 |
| 9.57.1 Detailed Description . . . . .                | 248 |
| 9.58 mln::fun::vv2b Namespace Reference . . . . .    | 248 |
| 9.58.1 Detailed Description . . . . .                | 248 |
| 9.59 mln::fun::vv2v Namespace Reference . . . . .    | 249 |
| 9.59.1 Detailed Description . . . . .                | 249 |
| 9.60 mln::fun::x2p Namespace Reference . . . . .     | 249 |
| 9.60.1 Detailed Description . . . . .                | 249 |
| 9.61 mln::fun::x2v Namespace Reference . . . . .     | 250 |
| 9.61.1 Detailed Description . . . . .                | 250 |
| 9.62 mln::fun::x2x Namespace Reference . . . . .     | 250 |
| 9.62.1 Detailed Description . . . . .                | 250 |
| 9.63 mln::geom Namespace Reference . . . . .         | 250 |
| 9.63.1 Detailed Description . . . . .                | 255 |

---

|           |                        |     |
|-----------|------------------------|-----|
| 9.63.2    | Function Documentation | 255 |
| 9.63.2.1  | bbox                   | 255 |
| 9.63.2.2  | bbox                   | 255 |
| 9.63.2.3  | bbox                   | 255 |
| 9.63.2.4  | bbox                   | 255 |
| 9.63.2.5  | chamfer                | 255 |
| 9.63.2.6  | delta                  | 256 |
| 9.63.2.7  | delta                  | 256 |
| 9.63.2.8  | delta                  | 256 |
| 9.63.2.9  | horizontal_symmetry    | 256 |
| 9.63.2.10 | max_col                | 256 |
| 9.63.2.11 | max_col                | 256 |
| 9.63.2.12 | max_ind                | 257 |
| 9.63.2.13 | max_row                | 257 |
| 9.63.2.14 | max_row                | 257 |
| 9.63.2.15 | max_sli                | 257 |
| 9.63.2.16 | mesh_corner_point_area | 257 |
| 9.63.2.17 | mesh_curvature         | 258 |
| 9.63.2.18 | mesh_normal            | 258 |
| 9.63.2.19 | min_col                | 258 |
| 9.63.2.20 | min_col                | 259 |
| 9.63.2.21 | min_ind                | 259 |
| 9.63.2.22 | min_row                | 259 |
| 9.63.2.23 | min_row                | 259 |
| 9.63.2.24 | min_sli                | 259 |
| 9.63.2.25 | ncols                  | 259 |
| 9.63.2.26 | ncols                  | 260 |
| 9.63.2.27 | ninds                  | 260 |
| 9.63.2.28 | nrows                  | 260 |
| 9.63.2.29 | nrows                  | 260 |
| 9.63.2.30 | nsites                 | 260 |
| 9.63.2.31 | nslis                  | 260 |
| 9.63.2.32 | pmin_pmax              | 261 |
| 9.63.2.33 | pmin_pmax              | 261 |
| 9.63.2.34 | pmin_pmax              | 261 |
| 9.63.2.35 | pmin_pmax              | 261 |

---

|  |     |
|--|-----|
| 9.63.2.36 rotate . . . . .                                   | 261 |
| 9.63.2.37 rotate . . . . .                                   | 261 |
| 9.63.2.38 rotate . . . . .                                   | 262 |
| 9.63.2.39 rotate . . . . .                                   | 262 |
| 9.63.2.40 rotate . . . . .                                   | 262 |
| 9.63.2.41 seeds2tiling . . . . .                             | 262 |
| 9.63.2.42 seeds2tiling_roundness . . . . .                   | 263 |
| 9.63.2.43 translate . . . . .                                | 263 |
| 9.63.2.44 translate . . . . .                                | 263 |
| 9.63.2.45 translate . . . . .                                | 264 |
| 9.63.2.46 vertical_symmetry . . . . .                        | 264 |
| 9.64 mln::geom::impl Namespace Reference . . . . .           | 264 |
| 9.64.1 Detailed Description . . . . .                        | 264 |
| 9.64.2 Function Documentation . . . . .                      | 265 |
| 9.64.2.1 seeds2tiling . . . . .                              | 265 |
| 9.65 mln::graph Namespace Reference . . . . .                | 265 |
| 9.65.1 Detailed Description . . . . .                        | 265 |
| 9.65.2 Function Documentation . . . . .                      | 266 |
| 9.65.2.1 compute . . . . .                                   | 266 |
| 9.65.2.2 labeling . . . . .                                  | 266 |
| 9.65.2.3 to_neighb . . . . .                                 | 266 |
| 9.65.2.4 to_win . . . . .                                    | 267 |
| 9.66 mln::grid Namespace Reference . . . . .                 | 267 |
| 9.66.1 Detailed Description . . . . .                        | 267 |
| 9.67 mln::histo Namespace Reference . . . . .                | 267 |
| 9.67.1 Detailed Description . . . . .                        | 268 |
| 9.67.2 Function Documentation . . . . .                      | 268 |
| 9.67.2.1 compute . . . . .                                   | 268 |
| 9.67.2.2 equalize . . . . .                                  | 268 |
| 9.68 mln::histo::impl Namespace Reference . . . . .          | 269 |
| 9.68.1 Detailed Description . . . . .                        | 269 |
| 9.69 mln::histo::impl::generic Namespace Reference . . . . . | 269 |
| 9.69.1 Detailed Description . . . . .                        | 269 |
| 9.70 mln::impl Namespace Reference . . . . .                 | 269 |
| 9.70.1 Detailed Description . . . . .                        | 269 |
| 9.71 mln::io Namespace Reference . . . . .                   | 269 |

---

|  |     |
|--|-----|
| 9.71.1 Detailed Description . . . . .              | 271 |
| 9.72 mln::io::cloud Namespace Reference . . . . .  | 271 |
| 9.72.1 Detailed Description . . . . .              | 271 |
| 9.72.2 Function Documentation . . . . .            | 271 |
| 9.72.2.1 load . . . . .                            | 271 |
| 9.72.2.2 save . . . . .                            | 271 |
| 9.73 mln::io::dicom Namespace Reference . . . . .  | 272 |
| 9.73.1 Detailed Description . . . . .              | 272 |
| 9.73.2 Function Documentation . . . . .            | 272 |
| 9.73.2.1 get_header . . . . .                      | 272 |
| 9.73.2.2 load . . . . .                            | 272 |
| 9.74 mln::io::dump Namespace Reference . . . . .   | 273 |
| 9.74.1 Detailed Description . . . . .              | 273 |
| 9.74.2 Function Documentation . . . . .            | 273 |
| 9.74.2.1 get_header . . . . .                      | 273 |
| 9.74.2.2 load . . . . .                            | 274 |
| 9.74.2.3 save . . . . .                            | 274 |
| 9.75 mln::io::fits Namespace Reference . . . . .   | 274 |
| 9.75.1 Detailed Description . . . . .              | 274 |
| 9.75.2 Function Documentation . . . . .            | 274 |
| 9.75.2.1 load . . . . .                            | 274 |
| 9.75.2.2 load . . . . .                            | 275 |
| 9.76 mln::io::fld Namespace Reference . . . . .    | 275 |
| 9.76.1 Detailed Description . . . . .              | 275 |
| 9.76.2 Function Documentation . . . . .            | 276 |
| 9.76.2.1 load . . . . .                            | 276 |
| 9.76.2.2 read_header . . . . .                     | 276 |
| 9.76.2.3 write_header . . . . .                    | 276 |
| 9.77 mln::io::magick Namespace Reference . . . . . | 276 |
| 9.77.1 Detailed Description . . . . .              | 277 |
| 9.77.2 Function Documentation . . . . .            | 277 |
| 9.77.2.1 load . . . . .                            | 277 |
| 9.77.2.2 save . . . . .                            | 277 |
| 9.77.2.3 save . . . . .                            | 278 |
| 9.78 mln::io::off Namespace Reference . . . . .    | 278 |
| 9.78.1 Detailed Description . . . . .              | 278 |



|          |   |     |
|----------|---|-----|
| 9.78.2   | Function Documentation                  | 278 |
| 9.78.2.1 | load                                    | 278 |
| 9.78.2.2 | save                                    | 279 |
| 9.78.2.3 | save_bin_alt                            | 279 |
| 9.79     | mln::io::pbm Namespace Reference        | 279 |
| 9.79.1   | Detailed Description                    | 280 |
| 9.79.2   | Function Documentation                  | 280 |
| 9.79.2.1 | load                                    | 280 |
| 9.79.2.2 | load                                    | 280 |
| 9.79.2.3 | save                                    | 281 |
| 9.80     | mln::io::pbm::impl Namespace Reference  | 281 |
| 9.80.1   | Detailed Description                    | 281 |
| 9.81     | mln::io::pbms Namespace Reference       | 281 |
| 9.81.1   | Detailed Description                    | 281 |
| 9.81.2   | Function Documentation                  | 281 |
| 9.81.2.1 | load                                    | 281 |
| 9.82     | mln::io::pbms::impl Namespace Reference | 282 |
| 9.82.1   | Detailed Description                    | 282 |
| 9.83     | mln::io::pfm Namespace Reference        | 282 |
| 9.83.1   | Detailed Description                    | 282 |
| 9.83.2   | Function Documentation                  | 283 |
| 9.83.2.1 | load                                    | 283 |
| 9.83.2.2 | load                                    | 283 |
| 9.83.2.3 | save                                    | 283 |
| 9.84     | mln::io::pfm::impl Namespace Reference  | 283 |
| 9.84.1   | Detailed Description                    | 283 |
| 9.85     | mln::io::pgm Namespace Reference        | 284 |
| 9.85.1   | Detailed Description                    | 284 |
| 9.85.2   | Function Documentation                  | 284 |
| 9.85.2.1 | load                                    | 284 |
| 9.85.2.2 | load                                    | 284 |
| 9.85.2.3 | save                                    | 285 |
| 9.86     | mln::io::pgms Namespace Reference       | 285 |
| 9.86.1   | Detailed Description                    | 285 |
| 9.86.2   | Function Documentation                  | 285 |
| 9.86.2.1 | load                                    | 285 |

---

|          |  |     |
|----------|--|-----|
| 9.87     | mln::io::plot Namespace Reference      | 285 |
| 9.87.1   | Detailed Description                   | 286 |
| 9.87.2   | Function Documentation                 | 286 |
| 9.87.2.1 | load                                   | 286 |
| 9.87.2.2 | save                                   | 286 |
| 9.87.2.3 | save                                   | 287 |
| 9.87.2.4 | save                                   | 287 |
| 9.88     | mln::io::pnm Namespace Reference       | 287 |
| 9.88.1   | Detailed Description                   | 288 |
| 9.88.2   | Function Documentation                 | 288 |
| 9.88.2.1 | load                                   | 288 |
| 9.88.2.2 | load                                   | 288 |
| 9.88.2.3 | load_ascii_builtin                     | 288 |
| 9.88.2.4 | load_ascii_value                       | 288 |
| 9.88.2.5 | load_raw_2d                            | 289 |
| 9.88.2.6 | max_component                          | 289 |
| 9.88.2.7 | save                                   | 289 |
| 9.89     | mln::io::pnm::impl Namespace Reference | 289 |
| 9.89.1   | Detailed Description                   | 289 |
| 9.90     | mln::io::pnms Namespace Reference      | 289 |
| 9.90.1   | Detailed Description                   | 290 |
| 9.90.2   | Function Documentation                 | 290 |
| 9.90.2.1 | load                                   | 290 |
| 9.90.2.2 | load                                   | 290 |
| 9.91     | mln::io::ppm Namespace Reference       | 290 |
| 9.91.1   | Detailed Description                   | 291 |
| 9.91.2   | Function Documentation                 | 291 |
| 9.91.2.1 | load                                   | 291 |
| 9.91.2.2 | load                                   | 291 |
| 9.91.2.3 | save                                   | 291 |
| 9.92     | mln::io::ppms Namespace Reference      | 292 |
| 9.92.1   | Detailed Description                   | 292 |
| 9.92.2   | Function Documentation                 | 292 |
| 9.92.2.1 | load                                   | 292 |
| 9.93     | mln::io::raw Namespace Reference       | 292 |
| 9.93.1   | Detailed Description                   | 293 |

---

|           |                                   |     |
|-----------|-----------------------------------|-----|
| 9.93.2    | Function Documentation            | 293 |
| 9.93.2.1  | get_header                        | 293 |
| 9.93.2.2  | load                              | 293 |
| 9.93.2.3  | save                              | 293 |
| 9.94      | mln::io::tiff Namespace Reference | 294 |
| 9.94.1    | Detailed Description              | 294 |
| 9.94.2    | Function Documentation            | 294 |
| 9.94.2.1  | load                              | 294 |
| 9.95      | mln::io::txt Namespace Reference  | 294 |
| 9.95.1    | Detailed Description              | 294 |
| 9.95.2    | Function Documentation            | 295 |
| 9.95.2.1  | save                              | 295 |
| 9.96      | mln::labeling Namespace Reference | 295 |
| 9.96.1    | Detailed Description              | 298 |
| 9.96.2    | Function Documentation            | 298 |
| 9.96.2.1  | background                        | 298 |
| 9.96.2.2  | blobs                             | 299 |
| 9.96.2.3  | blobs_and_compute                 | 299 |
| 9.96.2.4  | colorize                          | 300 |
| 9.96.2.5  | colorize                          | 300 |
| 9.96.2.6  | colorize                          | 300 |
| 9.96.2.7  | compute                           | 300 |
| 9.96.2.8  | compute                           | 301 |
| 9.96.2.9  | compute                           | 301 |
| 9.96.2.10 | compute                           | 302 |
| 9.96.2.11 | compute                           | 302 |
| 9.96.2.12 | compute_image                     | 302 |
| 9.96.2.13 | compute_image                     | 303 |
| 9.96.2.14 | compute_image                     | 303 |
| 9.96.2.15 | fill_holes                        | 304 |
| 9.96.2.16 | flat_zones                        | 304 |
| 9.96.2.17 | foreground                        | 305 |
| 9.96.2.18 | pack                              | 305 |
| 9.96.2.19 | pack                              | 306 |
| 9.96.2.20 | pack_inplace                      | 306 |
| 9.96.2.21 | pack_inplace                      | 306 |

|           |  |     |
|-----------|--|-----|
| 9.96.2.22 | regional_maxima                                  | 306 |
| 9.96.2.23 | regional_minima                                  | 307 |
| 9.96.2.24 | relabel  | 307 |
| 9.96.2.25 | relabel  | 307 |
| 9.96.2.26 | relabel_inplace                                  | 308 |
| 9.96.2.27 | relabel_inplace                                  | 308 |
| 9.96.2.28 | superpose  | 308 |
| 9.96.2.29 | value  | 309 |
| 9.96.2.30 | value_and_compute                                | 309 |
| 9.96.2.31 | wrap   | 310 |
| 9.96.2.32 | wrap   | 310 |
| 9.97      | mln::labeling::impl Namespace Reference          | 310 |
| 9.97.1    | Detailed Description                             | 311 |
| 9.97.2    | Function Documentation                           | 311 |
| 9.97.2.1  | compute_fastest                                  | 311 |
| 9.97.2.2  | compute_fastest                                  | 312 |
| 9.98      | mln::labeling::impl::generic Namespace Reference | 312 |
| 9.98.1    | Detailed Description                             | 312 |
| 9.98.2    | Function Documentation                           | 313 |
| 9.98.2.1  | compute  | 313 |
| 9.98.2.2  | compute  | 313 |
| 9.98.2.3  | compute  | 313 |
| 9.98.2.4  | compute  | 314 |
| 9.99      | mln::linear Namespace Reference                  | 314 |
| 9.99.1    | Detailed Description                             | 315 |
| 9.99.2    | Function Documentation                           | 315 |
| 9.99.2.1  | gaussian   | 315 |
| 9.99.2.2  | gaussian   | 316 |
| 9.99.2.3  | gaussian_1st_derivative                          | 316 |
| 9.99.2.4  | gaussian_1st_derivative                          | 316 |
| 9.99.2.5  | gaussian_2nd_derivative                          | 316 |
| 9.99.2.6  | mln_ch_convolve                                  | 317 |
| 9.99.2.7  | mln_ch_convolve                                  | 317 |
| 9.99.2.8  | mln_ch_convolve                                  | 317 |
| 9.99.2.9  | mln_ch_convolve                                  | 317 |
| 9.99.2.10 | mln_ch_convolve_grad                             | 318 |

|   |     |
|---|-----|
| 9.100mln::linear::impl Namespace Reference . . . . .        | 318 |
| 9.100.1 Detailed Description . . . . .                      | 318 |
| 9.101mln::linear::local Namespace Reference . . . . .       | 318 |
| 9.101.1 Detailed Description . . . . .                      | 318 |
| 9.101.2 Function Documentation . . . . .                    | 319 |
| 9.101.2.1 convolve . . . . .                                | 319 |
| 9.101.2.2 convolve . . . . .                                | 319 |
| 9.102mln::linear::local::impl Namespace Reference . . . . . | 319 |
| 9.102.1 Detailed Description . . . . .                      | 319 |
| 9.103mln::literal Namespace Reference . . . . .             | 319 |
| 9.103.1 Detailed Description . . . . .                      | 322 |
| 9.103.2 Variable Documentation . . . . .                    | 323 |
| 9.103.2.1 black . . . . .                                   | 323 |
| 9.103.2.2 blue . . . . .                                    | 323 |
| 9.103.2.3 brown . . . . .                                   | 323 |
| 9.103.2.4 cyan . . . . .                                    | 323 |
| 9.103.2.5 dark_gray . . . . .                               | 323 |
| 9.103.2.6 green . . . . .                                   | 323 |
| 9.103.2.7 identity . . . . .                                | 323 |
| 9.103.2.8 light_gray . . . . .                              | 323 |
| 9.103.2.9 lime . . . . .                                    | 324 |
| 9.103.2.10magenta . . . . .                                 | 324 |
| 9.103.2.11lmax . . . . .                                    | 324 |
| 9.103.2.12medium_gray . . . . .                             | 324 |
| 9.103.2.13min . . . . .                                     | 324 |
| 9.103.2.14olive . . . . .                                   | 324 |
| 9.103.2.15one . . . . .                                     | 324 |
| 9.103.2.16orange . . . . .                                  | 324 |
| 9.103.2.17origin . . . . .                                  | 325 |
| 9.103.2.18pink . . . . .                                    | 325 |
| 9.103.2.19purple . . . . .                                  | 325 |
| 9.103.2.20red . . . . .                                     | 325 |
| 9.103.2.21teal . . . . .                                    | 325 |
| 9.103.2.22violet . . . . .                                  | 325 |
| 9.103.2.23white . . . . .                                   | 325 |
| 9.103.2.24yellow . . . . .                                  | 325 |

|  |     |
|--|-----|
| 9.103.2.25zero                                       | 326 |
| 9.104mln::logical Namespace Reference                | 326 |
| 9.104.1 Detailed Description                         | 326 |
| 9.104.2 Function Documentation                       | 327 |
| 9.104.2.1 and_inplace                                | 327 |
| 9.104.2.2 and_not                                    | 327 |
| 9.104.2.3 and_not_inplace                            | 327 |
| 9.104.2.4 not_inplace                                | 328 |
| 9.104.2.5 or_inplace                                 | 328 |
| 9.104.2.6 xor_inplace                                | 329 |
| 9.105mln::logical::impl Namespace Reference          | 329 |
| 9.105.1 Detailed Description                         | 329 |
| 9.106mln::logical::impl::generic Namespace Reference | 329 |
| 9.106.1 Detailed Description                         | 330 |
| 9.107mln::make Namespace Reference                   | 330 |
| 9.107.1 Detailed Description                         | 335 |
| 9.107.2 Function Documentation                       | 335 |
| 9.107.2.1 attachment                                 | 335 |
| 9.107.2.2 box1d                                      | 335 |
| 9.107.2.3 box1d                                      | 336 |
| 9.107.2.4 box2d                                      | 336 |
| 9.107.2.5 box2d                                      | 336 |
| 9.107.2.6 box2d_h                                    | 337 |
| 9.107.2.7 box2d_h                                    | 337 |
| 9.107.2.8 box3d                                      | 338 |
| 9.107.2.9 box3d                                      | 338 |
| 9.107.2.10cell                                       | 339 |
| 9.107.2.11couple                                     | 339 |
| 9.107.2.12detachment                                 | 339 |
| 9.107.2.13dpoint2d_h                                 | 339 |
| 9.107.2.14dummy_p_edges                              | 340 |
| 9.107.2.15dummy_p_edges                              | 340 |
| 9.107.2.16dummy_p_vertices                           | 340 |
| 9.107.2.17dummy_p_vertices                           | 341 |
| 9.107.2.18edge_image                                 | 341 |
| 9.107.2.19edge_image                                 | 341 |

---

|            |                                |     |
|------------|--------------------------------|-----|
| 9.107.2.20 | edge_image                     | 342 |
| 9.107.2.21 | edge_image                     | 342 |
| 9.107.2.22 | edge_image                     | 342 |
| 9.107.2.23 | edge_image                     | 343 |
| 9.107.2.24 | h_mat                          | 343 |
| 9.107.2.25 | image                          | 343 |
| 9.107.2.26 | image                          | 343 |
| 9.107.2.27 | image                          | 344 |
| 9.107.2.28 | image2d                        | 344 |
| 9.107.2.29 | image3d                        | 344 |
| 9.107.2.30 | image3d                        | 344 |
| 9.107.2.31 | influence_zone_adjacency_graph | 345 |
| 9.107.2.32 | mat                            | 345 |
| 9.107.2.33 | ord_pair                       | 345 |
| 9.107.2.34 | p_edges_with_mass_centers      | 345 |
| 9.107.2.35 | p_vertices_with_mass_centers   | 346 |
| 9.107.2.36 | pix                            | 346 |
| 9.107.2.37 | pixel                          | 346 |
| 9.107.2.38 | pixel                          | 347 |
| 9.107.2.39 | point2d_h                      | 347 |
| 9.107.2.40 | rag_and_labeled_wsl            | 347 |
| 9.107.2.41 | region_adjacency_graph         | 348 |
| 9.107.2.42 | relabelfun                     | 348 |
| 9.107.2.43 | relabelfun                     | 349 |
| 9.107.2.44 | vec                            | 349 |
| 9.107.2.45 | vec                            | 349 |
| 9.107.2.46 | vec                            | 350 |
| 9.107.2.47 | vec                            | 350 |
| 9.107.2.48 | vertex_image                   | 350 |
| 9.107.2.49 | vertex_image                   | 351 |
| 9.107.2.50 | voronoi                        | 351 |
| 9.107.2.51 | w_window                       | 351 |
| 9.107.2.52 | w_window1d                     | 352 |
| 9.107.2.53 | w_window1d_int                 | 352 |
| 9.107.2.54 | w_window2d                     | 352 |
| 9.107.2.55 | w_window2d_int                 | 353 |

|   |     |
|---|-----|
| 9.107.2.56w_window3d                            | 353 |
| 9.107.2.57w_window3d_int                        | 353 |
| 9.107.2.58w_window_directional                  | 354 |
| 9.108mln::math Namespace Reference              | 354 |
| 9.108.1 Detailed Description                    | 355 |
| 9.108.2 Function Documentation                  | 355 |
| 9.108.2.1 abs                                   | 355 |
| 9.108.2.2 abs                                   | 355 |
| 9.108.2.3 abs                                   | 355 |
| 9.109mln::metal Namespace Reference             | 355 |
| 9.109.1 Detailed Description                    | 356 |
| 9.110mln::metal::impl Namespace Reference       | 356 |
| 9.110.1 Detailed Description                    | 356 |
| 9.111mln::metal::math Namespace Reference       | 356 |
| 9.111.1 Detailed Description                    | 356 |
| 9.112mln::metal::math::impl Namespace Reference | 357 |
| 9.112.1 Detailed Description                    | 357 |
| 9.113mln::morpho Namespace Reference            | 357 |
| 9.113.1 Detailed Description                    | 360 |
| 9.113.2 Function Documentation                  | 360 |
| 9.113.2.1 complementation                       | 360 |
| 9.113.2.2 complementation_inplace               | 360 |
| 9.113.2.3 contrast                              | 360 |
| 9.113.2.4 dilation                              | 360 |
| 9.113.2.5 erosion                               | 361 |
| 9.113.2.6 general                               | 361 |
| 9.113.2.7 gradient                              | 361 |
| 9.113.2.8 gradient_external                     | 361 |
| 9.113.2.9 gradient_internal                     | 361 |
| 9.113.2.10hit_or_miss                           | 362 |
| 9.113.2.11hit_or_miss_background_closing        | 362 |
| 9.113.2.12hit_or_miss_background_opening        | 362 |
| 9.113.2.13hit_or_miss_closing                   | 362 |
| 9.113.2.14hit_or_miss_opening                   | 363 |
| 9.113.2.15laplacian                             | 363 |
| 9.113.2.16line_gradient                         | 363 |



---

|            |  |     |
|------------|--|-----|
| 9.113.2.17 | meyer_wst  | 363 |
| 9.113.2.18 | meyer_wst  | 364 |
| 9.113.2.19 | min  | 364 |
| 9.113.2.20 | min_inplace                                      | 364 |
| 9.113.2.21 | minus  | 364 |
| 9.113.2.22 | plus   | 365 |
| 9.113.2.23 | rank_filter                                      | 365 |
| 9.113.2.24 | thick_miss                                       | 365 |
| 9.113.2.25 | thickening                                       | 365 |
| 9.113.2.26 | thin_fit   | 365 |
| 9.113.2.27 | thinning   | 366 |
| 9.113.2.28 | top_hat_black                                    | 366 |
| 9.113.2.29 | top_hat_self_complementary                       | 366 |
| 9.113.2.30 | top_hat_white                                    | 366 |
| 9.114      | mln::morpho::approx Namespace Reference          | 367 |
| 9.114.1    | Detailed Description                             | 367 |
| 9.115      | mln::morpho::attribute Namespace Reference       | 367 |
| 9.115.1    | Detailed Description                             | 367 |
| 9.116      | mln::morpho::closing::approx Namespace Reference | 367 |
| 9.116.1    | Detailed Description                             | 368 |
| 9.116.2    | Function Documentation                           | 368 |
| 9.116.2.1  | structural                                       | 368 |
| 9.117      | mln::morpho::elementary Namespace Reference      | 368 |
| 9.117.1    | Detailed Description                             | 369 |
| 9.117.2    | Function Documentation                           | 369 |
| 9.117.2.1  | closing  | 369 |
| 9.117.2.2  | mln_trait_op_minus_twice                         | 369 |
| 9.117.2.3  | opening  | 369 |
| 9.117.2.4  | top_hat_black                                    | 369 |
| 9.117.2.5  | top_hat_white                                    | 369 |
| 9.118      | mln::morpho::impl Namespace Reference            | 370 |
| 9.118.1    | Detailed Description                             | 370 |
| 9.119      | mln::morpho::impl::generic Namespace Reference   | 370 |
| 9.119.1    | Detailed Description                             | 370 |
| 9.120      | mln::morpho::opening::approx Namespace Reference | 370 |
| 9.120.1    | Detailed Description                             | 370 |

|  |     |
|--|-----|
| 9.120.2 Function Documentation . . . . .                                     | 371 |
| 9.120.2.1 structural . . . . .   | 371 |
| 9.121 mln::morpho::reconstruction Namespace Reference . . . . .              | 371 |
| 9.121.1 Detailed Description . . . . .                                       | 371 |
| 9.122 mln::morpho::reconstruction::by_dilation Namespace Reference . . . . . | 371 |
| 9.122.1 Detailed Description . . . . .                                       | 371 |
| 9.123 mln::morpho::reconstruction::by_erosion Namespace Reference . . . . .  | 371 |
| 9.123.1 Detailed Description . . . . .                                       | 372 |
| 9.124 mln::morpho::tree Namespace Reference . . . . .                        | 372 |
| 9.124.1 Detailed Description . . . . .                                       | 373 |
| 9.124.2 Function Documentation . . . . .                                     | 373 |
| 9.124.2.1 compute_attribute_image . . . . .                                  | 373 |
| 9.124.2.2 compute_attribute_image_from . . . . .                             | 374 |
| 9.124.2.3 compute_parent . . . . .   | 374 |
| 9.124.2.4 dual_input_max_tree . . . . .                                      | 375 |
| 9.124.2.5 max_tree . . . . .   | 376 |
| 9.124.2.6 min_tree . . . . .   | 376 |
| 9.124.2.7 propagate_if . . . . .   | 377 |
| 9.124.2.8 propagate_if . . . . .   | 377 |
| 9.124.2.9 propagate_if_value . . . . .                                       | 377 |
| 9.124.2.10 propagate_if_value . . . . .                                      | 377 |
| 9.124.2.11 propagate_node_to_ancestors . . . . .                             | 378 |
| 9.124.2.12 propagate_node_to_ancestors . . . . .                             | 378 |
| 9.124.2.13 propagate_node_to_descendants . . . . .                           | 378 |
| 9.124.2.14 propagate_node_to_descendants . . . . .                           | 379 |
| 9.124.2.15 propagate_representative . . . . .                                | 379 |
| 9.125 mln::morpho::tree::filter Namespace Reference . . . . .                | 379 |
| 9.125.1 Detailed Description . . . . .                                       | 380 |
| 9.125.2 Function Documentation . . . . .                                     | 380 |
| 9.125.2.1 direct . . . . .   | 380 |
| 9.125.2.2 filter . . . . .   | 380 |
| 9.125.2.3 max . . . . .  | 381 |
| 9.125.2.4 min . . . . .  | 381 |
| 9.125.2.5 subtractive . . . . .  | 381 |
| 9.126 mln::morpho::watershed Namespace Reference . . . . .                   | 382 |
| 9.126.1 Detailed Description . . . . .                                       | 382 |

|   |     |
|---|-----|
| 9.126.2 Function Documentation . . . . .                                      | 382 |
| 9.126.2.1 flooding . . . . .  | 382 |
| 9.126.2.2 flooding . . . . .  | 383 |
| 9.126.2.3 superpose . . . . .   | 383 |
| 9.126.2.4 superpose . . . . .   | 383 |
| 9.126.2.5 topological . . . . .   | 384 |
| 9.127mln::morpho::watershed::watershed Namespace Reference . . . . .          | 384 |
| 9.127.1 Detailed Description . . . . .  | 384 |
| 9.128mln::morpho::watershed::watershed::generic Namespace Reference . . . . . | 384 |
| 9.128.1 Detailed Description . . . . .  | 384 |
| 9.129mln::norm Namespace Reference . . . . .                                  | 384 |
| 9.129.1 Detailed Description . . . . .  | 385 |
| 9.129.2 Function Documentation . . . . .                                      | 385 |
| 9.129.2.1 l1 . . . . .  | 385 |
| 9.129.2.2 l1_distance . . . . .   | 386 |
| 9.129.2.3 l2 . . . . .  | 386 |
| 9.129.2.4 l2_distance . . . . .   | 386 |
| 9.129.2.5 linfty . . . . .  | 386 |
| 9.129.2.6 linfty_distance . . . . .   | 386 |
| 9.129.2.7 sqr_l2 . . . . .  | 386 |
| 9.130mln::norm::impl Namespace Reference . . . . .                            | 386 |
| 9.130.1 Detailed Description . . . . .  | 387 |
| 9.131mln::opt Namespace Reference . . . . .                                   | 387 |
| 9.131.1 Detailed Description . . . . .  | 387 |
| 9.131.2 Function Documentation . . . . .                                      | 388 |
| 9.131.2.1 at . . . . .  | 388 |
| 9.131.2.2 at . . . . .  | 388 |
| 9.131.2.3 at . . . . .  | 388 |
| 9.131.2.4 at . . . . .  | 388 |
| 9.131.2.5 at . . . . .  | 388 |
| 9.131.2.6 at . . . . .  | 388 |
| 9.132mln::opt::impl Namespace Reference . . . . .                             | 388 |
| 9.132.1 Detailed Description . . . . .  | 389 |
| 9.133mln::pw Namespace Reference . . . . .                                    | 389 |
| 9.133.1 Detailed Description . . . . .  | 389 |
| 9.134mln::registration Namespace Reference . . . . .                          | 389 |

|   |     |
|---|-----|
| 9.134.1 Detailed Description . . . . .              | 390 |
| 9.134.2 Function Documentation . . . . .            | 390 |
| 9.134.2.1 get_rot . . . . .                         | 390 |
| 9.134.2.2 icp . . . . .                             | 390 |
| 9.134.2.3 icp . . . . .                             | 391 |
| 9.134.2.4 registration1 . . . . .                   | 391 |
| 9.134.2.5 registration2 . . . . .                   | 391 |
| 9.134.2.6 registration3 . . . . .                   | 392 |
| 9.135mln::select Namespace Reference . . . . .      | 392 |
| 9.135.1 Detailed Description . . . . .              | 392 |
| 9.136mln::set Namespace Reference . . . . .         | 392 |
| 9.136.1 Detailed Description . . . . .              | 393 |
| 9.136.2 Function Documentation . . . . .            | 393 |
| 9.136.2.1 card . . . . .                            | 393 |
| 9.136.2.2 compute . . . . .                         | 393 |
| 9.136.2.3 compute_with_weights . . . . .            | 393 |
| 9.136.2.4 get . . . . .                             | 394 |
| 9.136.2.5 has . . . . .                             | 394 |
| 9.136.2.6 mln_meta_accu_result . . . . .            | 394 |
| 9.136.2.7 mln_meta_accu_result . . . . .            | 394 |
| 9.137mln::subsampling Namespace Reference . . . . . | 394 |
| 9.137.1 Detailed Description . . . . .              | 395 |
| 9.137.2 Function Documentation . . . . .            | 395 |
| 9.137.2.1 antialiased . . . . .                     | 395 |
| 9.137.2.2 antialiased . . . . .                     | 395 |
| 9.137.2.3 gaussian_subsampling . . . . .            | 396 |
| 9.137.2.4 subsampling . . . . .                     | 396 |
| 9.138mln::tag Namespace Reference . . . . .         | 396 |
| 9.138.1 Detailed Description . . . . .              | 396 |
| 9.139mln::test Namespace Reference . . . . .        | 396 |
| 9.139.1 Detailed Description . . . . .              | 397 |
| 9.139.2 Function Documentation . . . . .            | 397 |
| 9.139.2.1 positive . . . . .                        | 397 |
| 9.139.2.2 predicate . . . . .                       | 397 |
| 9.139.2.3 predicate . . . . .                       | 397 |
| 9.139.2.4 predicate . . . . .                       | 398 |

|  |     |
|--|-----|
| 9.140mln::test::impl Namespace Reference | 398 |
| 9.140.1 Detailed Description             | 398 |
| 9.141mln::topo Namespace Reference       | 398 |
| 9.141.1 Detailed Description             | 402 |
| 9.141.2 Function Documentation           | 403 |
| 9.141.2.1 detach                         | 403 |
| 9.141.2.2 edge                           | 403 |
| 9.141.2.3 is_facet                       | 403 |
| 9.141.2.4 make_algebraic_face            | 403 |
| 9.141.2.5 operator!=                     | 404 |
| 9.141.2.6 operator!=                     | 404 |
| 9.141.2.7 operator!=                     | 404 |
| 9.141.2.8 operator!=                     | 404 |
| 9.141.2.9 operator+                      | 405 |
| 9.141.2.10operator-                      | 405 |
| 9.141.2.11operator-                      | 405 |
| 9.141.2.12operator-                      | 405 |
| 9.141.2.13operator<                      | 405 |
| 9.141.2.14operator<                      | 405 |
| 9.141.2.15operator<                      | 406 |
| 9.141.2.16operator<                      | 406 |
| 9.141.2.17operator<<                     | 406 |
| 9.141.2.18operator<<                     | 406 |
| 9.141.2.19operator<<                     | 406 |
| 9.141.2.20operator<<                     | 407 |
| 9.141.2.21operator<<                     | 407 |
| 9.141.2.22operator==                     | 407 |
| 9.141.2.23operator==                     | 407 |
| 9.141.2.24operator==                     | 407 |
| 9.141.2.25operator==                     | 408 |
| 9.141.2.26operator==                     | 408 |
| 9.142mln::trace Namespace Reference      | 408 |
| 9.142.1 Detailed Description             | 408 |
| 9.143mln::trait Namespace Reference      | 408 |
| 9.143.1 Detailed Description             | 408 |
| 9.144mln::transform Namespace Reference  | 408 |

|  |     |
|--|-----|
| 9.144.1 Detailed Description                   | 410 |
| 9.144.2 Function Documentation                 | 410 |
| 9.144.2.1 distance_and_closest_point_geodesic  | 410 |
| 9.144.2.2 distance_and_closest_point_geodesic  | 410 |
| 9.144.2.3 distance_and_influence_zone_geodesic | 411 |
| 9.144.2.4 distance_front                       | 411 |
| 9.144.2.5 distance_geodesic                    | 411 |
| 9.144.2.6 hough                                | 412 |
| 9.144.2.7 influence_zone_front                 | 412 |
| 9.144.2.8 influence_zone_front                 | 412 |
| 9.144.2.9 influence_zone_geodesic              | 412 |
| 9.144.2.10influence_zone_geodesic_saturated    | 413 |
| 9.144.2.11influence_zone_geodesic_saturated    | 413 |
| 9.145mln::util Namespace Reference             | 413 |
| 9.145.1 Detailed Description                   | 417 |
| 9.145.2 Typedef Documentation                  | 417 |
| 9.145.2.1 vertex_id_t                          | 417 |
| 9.145.3 Function Documentation                 | 417 |
| 9.145.3.1 display_branch                       | 417 |
| 9.145.3.2 display_tree                         | 417 |
| 9.145.3.3 lemmings                             | 418 |
| 9.145.3.4 make_greater_point                   | 418 |
| 9.145.3.5 make_greater_psite                   | 418 |
| 9.145.3.6 operator<                            | 418 |
| 9.145.3.7 operator<<                           | 418 |
| 9.145.3.8 operator<<                           | 418 |
| 9.145.3.9 operator==                           | 419 |
| 9.145.3.10operator==                           | 419 |
| 9.145.3.11ord_strict                           | 419 |
| 9.145.3.12ord_weak                             | 419 |
| 9.145.3.13tree_fast_to_image                   | 419 |
| 9.145.3.14tree_to_fast                         | 419 |
| 9.145.3.15tree_to_image                        | 420 |
| 9.146mln::util::impl Namespace Reference       | 420 |
| 9.146.1 Detailed Description                   | 420 |
| 9.147mln::value Namespace Reference            | 420 |

|                                |     |
|--------------------------------|-----|
| 9.147.1 Detailed Description   | 425 |
| 9.147.2 Typedef Documentation  | 425 |
| 9.147.2.1 float01_16           | 425 |
| 9.147.2.2 float01_8            | 425 |
| 9.147.2.3 gl16                 | 425 |
| 9.147.2.4 gl8                  | 425 |
| 9.147.2.5 glf                  | 425 |
| 9.147.2.6 int_s16              | 426 |
| 9.147.2.7 int_s32              | 426 |
| 9.147.2.8 int_s8               | 426 |
| 9.147.2.9 int_u12              | 426 |
| 9.147.2.10 int_u16             | 426 |
| 9.147.2.11 int_u32             | 426 |
| 9.147.2.12 int_u8              | 426 |
| 9.147.2.13 label_16            | 426 |
| 9.147.2.14 label_32            | 426 |
| 9.147.2.15 label_8             | 427 |
| 9.147.2.16 rgb16               | 427 |
| 9.147.2.17 rgb8                | 427 |
| 9.147.3 Function Documentation | 427 |
| 9.147.3.1 cast                 | 427 |
| 9.147.3.2 equiv                | 427 |
| 9.147.3.3 operator*            | 427 |
| 9.147.3.4 operator*            | 427 |
| 9.147.3.5 operator+            | 428 |
| 9.147.3.6 operator+            | 428 |
| 9.147.3.7 operator-            | 428 |
| 9.147.3.8 operator-            | 428 |
| 9.147.3.9 operator/            | 428 |
| 9.147.3.10 operator/           | 428 |
| 9.147.3.11 operator<<          | 428 |
| 9.147.3.12 operator<<          | 429 |
| 9.147.3.13 operator<<          | 429 |
| 9.147.3.14 operator<<          | 429 |
| 9.147.3.15 operator<<          | 429 |
| 9.147.3.16 operator<<          | 430 |

|            |  |            |
|------------|--|------------|
| 9.147.3.17 | operator<<   | 430        |
| 9.147.3.18 | operator<<   | 430        |
| 9.147.3.19 | operator<<   | 431        |
| 9.147.3.20 | operator<<   | 431        |
| 9.147.3.21 | operator<<   | 431        |
| 9.147.3.22 | operator==   | 431        |
| 9.147.3.23 | operator==   | 432        |
| 9.147.3.24 | other  | 432        |
| 9.147.3.25 | stack  | 432        |
| 9.148      | mln::value::impl Namespace Reference                       | 432        |
| 9.148.1    | Detailed Description                                       | 432        |
| 9.149      | mln::win Namespace Reference                               | 432        |
| 9.149.1    | Detailed Description                                       | 434        |
| 9.149.2    | Function Documentation                                     | 434        |
| 9.149.2.1  | diff   | 434        |
| 9.149.2.2  | mln_regular  | 434        |
| 9.149.2.3  | mln_regular  | 434        |
| 9.149.2.4  | sym  | 434        |
| 9.149.2.5  | sym  | 434        |
| <b>10</b>  | <b>Class Documentation</b>                                 | <b>437</b> |
| 10.1       | mln::accu::center< P, V > Struct Template Reference        | 437        |
| 10.1.1     | Detailed Description                                       | 437        |
| 10.1.2     | Member Function Documentation                              | 438        |
| 10.1.2.1   | init   | 438        |
| 10.1.2.2   | is_valid   | 438        |
| 10.1.2.3   | nsites   | 438        |
| 10.1.2.4   | take_as_init   | 438        |
| 10.1.2.5   | take_n_times   | 438        |
| 10.1.2.6   | to_result  | 438        |
| 10.2       | mln::accu::convolve< T1, T2, R > Struct Template Reference | 439        |
| 10.2.1     | Detailed Description                                       | 439        |
| 10.2.2     | Member Function Documentation                              | 439        |
| 10.2.2.1   | init   | 439        |
| 10.2.2.2   | is_valid   | 440        |
| 10.2.2.3   | take_as_init   | 440        |
| 10.2.2.4   | take_n_times   | 440        |



---

|          |  |     |
|----------|--|-----|
| 10.2.2.5 | to_result  | 440 |
| 10.3     | mln::accu::count_adjacent_vertices< F, S > Struct Template Reference | 440 |
| 10.3.1   | Detailed Description   | 441 |
| 10.3.2   | Member Function Documentation  | 441 |
| 10.3.2.1 | init   | 441 |
| 10.3.2.2 | is_valid   | 441 |
| 10.3.2.3 | set_value  | 441 |
| 10.3.2.4 | take_as_init   | 441 |
| 10.3.2.5 | take_n_times   | 442 |
| 10.3.2.6 | to_result  | 442 |
| 10.4     | mln::accu::count_labels< L > Struct Template Reference               | 442 |
| 10.4.1   | Detailed Description   | 442 |
| 10.4.2   | Member Function Documentation  | 443 |
| 10.4.2.1 | init   | 443 |
| 10.4.2.2 | is_valid   | 443 |
| 10.4.2.3 | set_value  | 443 |
| 10.4.2.4 | take_as_init   | 443 |
| 10.4.2.5 | take_n_times   | 443 |
| 10.4.2.6 | to_result  | 443 |
| 10.5     | mln::accu::count_value< V > Struct Template Reference                | 443 |
| 10.5.1   | Detailed Description   | 444 |
| 10.5.2   | Member Function Documentation  | 444 |
| 10.5.2.1 | init   | 444 |
| 10.5.2.2 | is_valid   | 444 |
| 10.5.2.3 | set_value  | 444 |
| 10.5.2.4 | take_as_init   | 445 |
| 10.5.2.5 | take_n_times   | 445 |
| 10.5.2.6 | to_result  | 445 |
| 10.6     | mln::accu::histo< V > Struct Template Reference                      | 445 |
| 10.6.1   | Detailed Description   | 446 |
| 10.6.2   | Member Function Documentation  | 446 |
| 10.6.2.1 | is_valid   | 446 |
| 10.6.2.2 | take   | 446 |
| 10.6.2.3 | take_as_init   | 446 |
| 10.6.2.4 | take_n_times   | 446 |
| 10.6.2.5 | vect   | 446 |

---

|           |   |     |
|-----------|---|-----|
| 10.7      | <code>mln::accu::label_used&lt; L &gt;</code> Struct Template Reference | 446 |
| 10.7.1    | Detailed Description  | 447 |
| 10.7.2    | Member Function Documentation   | 447 |
| 10.7.2.1  | <code>init</code>   | 447 |
| 10.7.2.2  | <code>is_valid</code>   | 447 |
| 10.7.2.3  | <code>take</code>   | 447 |
| 10.7.2.4  | <code>take_as_init</code>   | 448 |
| 10.7.2.5  | <code>take_n_times</code>   | 448 |
| 10.7.2.6  | <code>to_result</code>  | 448 |
| 10.8      | <code>mln::accu::logic::land</code> Struct Reference                    | 448 |
| 10.8.1    | Detailed Description  | 449 |
| 10.8.2    | Member Function Documentation   | 449 |
| 10.8.2.1  | <code>init</code>   | 449 |
| 10.8.2.2  | <code>is_valid</code>   | 449 |
| 10.8.2.3  | <code>take_as_init</code>   | 449 |
| 10.8.2.4  | <code>take_n_times</code>   | 449 |
| 10.8.2.5  | <code>to_result</code>  | 449 |
| 10.9      | <code>mln::accu::logic::land_basic</code> Struct Reference              | 449 |
| 10.9.1    | Detailed Description  | 450 |
| 10.9.2    | Member Function Documentation   | 450 |
| 10.9.2.1  | <code>can_stop</code>   | 450 |
| 10.9.2.2  | <code>init</code>   | 450 |
| 10.9.2.3  | <code>is_valid</code>   | 450 |
| 10.9.2.4  | <code>take_as_init</code>   | 451 |
| 10.9.2.5  | <code>take_n_times</code>   | 451 |
| 10.9.2.6  | <code>to_result</code>  | 451 |
| 10.10     | <code>mln::accu::logic::lor</code> Struct Reference                     | 451 |
| 10.10.1   | Detailed Description  | 451 |
| 10.10.2   | Member Function Documentation   | 452 |
| 10.10.2.1 | <code>init</code>   | 452 |
| 10.10.2.2 | <code>is_valid</code>   | 452 |
| 10.10.2.3 | <code>take_as_init</code>   | 452 |
| 10.10.2.4 | <code>take_n_times</code>   | 452 |
| 10.10.2.5 | <code>to_result</code>  | 452 |
| 10.11     | <code>mln::accu::logic::lor_basic</code> Struct Reference               | 452 |
| 10.11.1   | Detailed Description  | 453 |

|   |     |
|---|-----|
| 10.11.2 Member Function Documentation                       | 453 |
| 10.11.2.1 can_stop  | 453 |
| 10.11.2.2 init  | 453 |
| 10.11.2.3 is_valid  | 453 |
| 10.11.2.4 take_as_init                                      | 453 |
| 10.11.2.5 take_n_times                                      | 454 |
| 10.11.2.6 to_result   | 454 |
| 10.12mln::accu::maj_h< T > Struct Template Reference        | 454 |
| 10.12.1 Detailed Description                                | 454 |
| 10.12.2 Member Function Documentation                       | 455 |
| 10.12.2.1 init  | 455 |
| 10.12.2.2 is_valid  | 455 |
| 10.12.2.3 take_as_init                                      | 455 |
| 10.12.2.4 take_n_times                                      | 455 |
| 10.12.2.5 to_result   | 455 |
| 10.13mln::accu::math::count< T > Struct Template Reference  | 455 |
| 10.13.1 Detailed Description                                | 456 |
| 10.13.2 Member Function Documentation                       | 456 |
| 10.13.2.1 init  | 456 |
| 10.13.2.2 is_valid  | 456 |
| 10.13.2.3 set_value   | 456 |
| 10.13.2.4 take_as_init                                      | 456 |
| 10.13.2.5 take_n_times                                      | 457 |
| 10.13.2.6 to_result   | 457 |
| 10.14mln::accu::math::inf< T > Struct Template Reference    | 457 |
| 10.14.1 Detailed Description                                | 457 |
| 10.14.2 Member Function Documentation                       | 458 |
| 10.14.2.1 init  | 458 |
| 10.14.2.2 is_valid  | 458 |
| 10.14.2.3 take_as_init                                      | 458 |
| 10.14.2.4 take_n_times                                      | 458 |
| 10.14.2.5 to_result   | 458 |
| 10.15mln::accu::math::sum< T, S > Struct Template Reference | 458 |
| 10.15.1 Detailed Description                                | 459 |
| 10.15.2 Member Function Documentation                       | 459 |
| 10.15.2.1 init  | 459 |

|           |   |     |
|-----------|---|-----|
| 10.15.2.2 | is_valid  | 459 |
| 10.15.2.3 | take_as_init  | 459 |
| 10.15.2.4 | take_n_times  | 459 |
| 10.15.2.5 | to_result   | 460 |
| 10.16     | mln::accu::math::sup< T > Struct Template Reference       | 460 |
| 10.16.1   | Detailed Description                                      | 460 |
| 10.16.2   | Member Function Documentation                             | 460 |
| 10.16.2.1 | init  | 460 |
| 10.16.2.2 | is_valid  | 461 |
| 10.16.2.3 | take_as_init  | 461 |
| 10.16.2.4 | take_n_times  | 461 |
| 10.16.2.5 | to_result   | 461 |
| 10.17     | mln::accu::max_site< I > Struct Template Reference        | 461 |
| 10.17.1   | Detailed Description                                      | 462 |
| 10.17.2   | Member Function Documentation                             | 462 |
| 10.17.2.1 | init  | 462 |
| 10.17.2.2 | is_valid  | 462 |
| 10.17.2.3 | take_as_init  | 462 |
| 10.17.2.4 | take_n_times  | 462 |
| 10.17.2.5 | to_result   | 462 |
| 10.18     | mln::accu::meta::center Struct Reference                  | 463 |
| 10.18.1   | Detailed Description                                      | 463 |
| 10.19     | mln::accu::meta::count_adjacent_vertices Struct Reference | 463 |
| 10.19.1   | Detailed Description                                      | 464 |
| 10.20     | mln::accu::meta::count_labels Struct Reference            | 464 |
| 10.20.1   | Detailed Description                                      | 465 |
| 10.21     | mln::accu::meta::count_value Struct Reference             | 465 |
| 10.21.1   | Detailed Description                                      | 466 |
| 10.22     | mln::accu::meta::histo Struct Reference                   | 466 |
| 10.22.1   | Detailed Description                                      | 467 |
| 10.23     | mln::accu::meta::label_used Struct Reference              | 467 |
| 10.23.1   | Detailed Description                                      | 468 |
| 10.24     | mln::accu::meta::logic::land Struct Reference             | 468 |
| 10.24.1   | Detailed Description                                      | 469 |
| 10.25     | mln::accu::meta::logic::land_basic Struct Reference       | 469 |
| 10.25.1   | Detailed Description                                      | 470 |

|  |     |
|--|-----|
| 10.26mln::accu::meta::logic::lor Struct Reference . . . . .              | 470 |
| 10.26.1 Detailed Description . . . . .                                   | 471 |
| 10.27mln::accu::meta::logic::lor_basic Struct Reference . . . . .        | 471 |
| 10.27.1 Detailed Description . . . . .                                   | 472 |
| 10.28mln::accu::meta::maj_h Struct Reference . . . . .                   | 472 |
| 10.28.1 Detailed Description . . . . .                                   | 473 |
| 10.29mln::accu::meta::math::count Struct Reference . . . . .             | 473 |
| 10.29.1 Detailed Description . . . . .                                   | 474 |
| 10.30mln::accu::meta::math::inf Struct Reference . . . . .               | 474 |
| 10.30.1 Detailed Description . . . . .                                   | 475 |
| 10.31mln::accu::meta::math::sum Struct Reference . . . . .               | 475 |
| 10.31.1 Detailed Description . . . . .                                   | 476 |
| 10.32mln::accu::meta::math::sup Struct Reference . . . . .               | 476 |
| 10.32.1 Detailed Description . . . . .                                   | 477 |
| 10.33mln::accu::meta::max_site Struct Reference . . . . .                | 477 |
| 10.33.1 Detailed Description . . . . .                                   | 478 |
| 10.34mln::accu::meta::nil Struct Reference . . . . .                     | 478 |
| 10.34.1 Detailed Description . . . . .                                   | 479 |
| 10.35mln::accu::meta::p< mA > Struct Template Reference . . . . .        | 479 |
| 10.35.1 Detailed Description . . . . .                                   | 480 |
| 10.36mln::accu::meta::pair< A1, A2 > Struct Template Reference . . . . . | 480 |
| 10.36.1 Detailed Description . . . . .                                   | 481 |
| 10.37mln::accu::meta::rms Struct Reference . . . . .                     | 481 |
| 10.37.1 Detailed Description . . . . .                                   | 482 |
| 10.38mln::accu::meta::shape::bbox Struct Reference . . . . .             | 482 |
| 10.38.1 Detailed Description . . . . .                                   | 483 |
| 10.39mln::accu::meta::shape::height Struct Reference . . . . .           | 483 |
| 10.39.1 Detailed Description . . . . .                                   | 484 |
| 10.40mln::accu::meta::shape::volume Struct Reference . . . . .           | 484 |
| 10.40.1 Detailed Description . . . . .                                   | 485 |
| 10.41mln::accu::meta::stat::max Struct Reference . . . . .               | 485 |
| 10.41.1 Detailed Description . . . . .                                   | 486 |
| 10.42mln::accu::meta::stat::max_h Struct Reference . . . . .             | 486 |
| 10.42.1 Detailed Description . . . . .                                   | 487 |
| 10.43mln::accu::meta::stat::mean Struct Reference . . . . .              | 487 |
| 10.43.1 Detailed Description . . . . .                                   | 488 |

|           |   |     |
|-----------|---|-----|
| 10.44     | <code>mln::accu::meta::stat::median_alt&lt; T &gt;</code> Struct Template Reference | 488 |
| 10.44.1   | Detailed Description  | 489 |
| 10.45     | <code>mln::accu::meta::stat::median_h</code> Struct Reference                       | 489 |
| 10.45.1   | Detailed Description  | 490 |
| 10.46     | <code>mln::accu::meta::stat::min</code> Struct Reference                            | 490 |
| 10.46.1   | Detailed Description  | 491 |
| 10.47     | <code>mln::accu::meta::stat::min_h</code> Struct Reference                          | 491 |
| 10.47.1   | Detailed Description  | 492 |
| 10.48     | <code>mln::accu::meta::stat::rank</code> Struct Reference                           | 492 |
| 10.48.1   | Detailed Description  | 493 |
| 10.49     | <code>mln::accu::meta::stat::rank_high_quant</code> Struct Reference                | 493 |
| 10.49.1   | Detailed Description  | 494 |
| 10.50     | <code>mln::accu::meta::tuple&lt; n, &gt;</code> Struct Template Reference           | 494 |
| 10.50.1   | Detailed Description  | 495 |
| 10.51     | <code>mln::accu::meta::val&lt; mA &gt;</code> Struct Template Reference             | 495 |
| 10.51.1   | Detailed Description  | 496 |
| 10.52     | <code>mln::accu::nil&lt; T &gt;</code> Struct Template Reference                    | 496 |
| 10.52.1   | Detailed Description  | 497 |
| 10.52.2   | Member Function Documentation   | 497 |
| 10.52.2.1 | <code>init</code>   | 497 |
| 10.52.2.2 | <code>is_valid</code>   | 497 |
| 10.52.2.3 | <code>take_as_init</code>   | 497 |
| 10.52.2.4 | <code>take_n_times</code>   | 497 |
| 10.52.2.5 | <code>to_result</code>  | 498 |
| 10.53     | <code>mln::accu::p&lt; A &gt;</code> Struct Template Reference                      | 498 |
| 10.53.1   | Detailed Description  | 498 |
| 10.53.2   | Member Function Documentation   | 498 |
| 10.53.2.1 | <code>init</code>   | 498 |
| 10.53.2.2 | <code>is_valid</code>   | 499 |
| 10.53.2.3 | <code>take_as_init</code>   | 499 |
| 10.53.2.4 | <code>take_n_times</code>   | 499 |
| 10.53.2.5 | <code>to_result</code>  | 499 |
| 10.54     | <code>mln::accu::pair&lt; A1, A2, T &gt;</code> Struct Template Reference           | 499 |
| 10.54.1   | Detailed Description  | 501 |
| 10.54.2   | Member Function Documentation   | 501 |
| 10.54.2.1 | <code>first</code>  | 501 |

---

|           |   |     |
|-----------|---|-----|
| 10.54.2.2 | first_accu . . . . .  | 501 |
| 10.54.2.3 | init . . . . .  | 501 |
| 10.54.2.4 | is_valid . . . . .  | 501 |
| 10.54.2.5 | second . . . . .  | 501 |
| 10.54.2.6 | second_accu . . . . .   | 502 |
| 10.54.2.7 | take_as_init . . . . .  | 502 |
| 10.54.2.8 | take_n_times . . . . .  | 502 |
| 10.54.2.9 | to_result . . . . .   | 502 |
| 10.55     | mln::accu::rms< T, V > Struct Template Reference . . . . .        | 502 |
| 10.55.1   | Detailed Description . . . . .                                    | 503 |
| 10.55.2   | Member Function Documentation . . . . .                           | 503 |
| 10.55.2.1 | init . . . . .  | 503 |
| 10.55.2.2 | is_valid . . . . .  | 503 |
| 10.55.2.3 | take_as_init . . . . .  | 503 |
| 10.55.2.4 | take_n_times . . . . .  | 503 |
| 10.55.2.5 | to_result . . . . .   | 503 |
| 10.56     | mln::accu::shape::bbox< P > Struct Template Reference . . . . .   | 504 |
| 10.56.1   | Detailed Description . . . . .                                    | 504 |
| 10.56.2   | Member Function Documentation . . . . .                           | 504 |
| 10.56.2.1 | init . . . . .  | 504 |
| 10.56.2.2 | is_valid . . . . .  | 504 |
| 10.56.2.3 | take_as_init . . . . .  | 505 |
| 10.56.2.4 | take_n_times . . . . .  | 505 |
| 10.56.2.5 | to_result . . . . .   | 505 |
| 10.57     | mln::accu::shape::height< I > Struct Template Reference . . . . . | 505 |
| 10.57.1   | Detailed Description . . . . .                                    | 506 |
| 10.57.2   | Member Typedef Documentation . . . . .                            | 506 |
| 10.57.2.1 | argument . . . . .  | 506 |
| 10.57.2.2 | value . . . . .   | 506 |
| 10.57.3   | Member Function Documentation . . . . .                           | 506 |
| 10.57.3.1 | init . . . . .  | 506 |
| 10.57.3.2 | is_valid . . . . .  | 506 |
| 10.57.3.3 | set_value . . . . .   | 507 |
| 10.57.3.4 | take_as_init . . . . .  | 507 |
| 10.57.3.5 | take_n_times . . . . .  | 507 |
| 10.57.3.6 | to_result . . . . .   | 507 |

---

|  |     |
|--|-----|
| 10.58mln::accu::shape::volume< I > Struct Template Reference . . . . .           | 507 |
| 10.58.1 Detailed Description . . . . .   | 508 |
| 10.58.2 Member Typedef Documentation . . . . .                                   | 508 |
| 10.58.2.1 argument . . . . .   | 508 |
| 10.58.2.2 value . . . . .  | 508 |
| 10.58.3 Member Function Documentation . . . . .                                  | 508 |
| 10.58.3.1 init . . . . .   | 508 |
| 10.58.3.2 is_valid . . . . .   | 509 |
| 10.58.3.3 set_value . . . . .  | 509 |
| 10.58.3.4 take_as_init . . . . .   | 509 |
| 10.58.3.5 take_n_times . . . . .   | 509 |
| 10.58.3.6 to_result . . . . .  | 509 |
| 10.59mln::accu::site_set::rectangularity< P > Class Template Reference . . . . . | 509 |
| 10.59.1 Detailed Description . . . . .   | 510 |
| 10.59.2 Constructor & Destructor Documentation . . . . .                         | 510 |
| 10.59.2.1 rectangularity . . . . .   | 510 |
| 10.59.3 Member Function Documentation . . . . .                                  | 510 |
| 10.59.3.1 area . . . . .   | 510 |
| 10.59.3.2 bbox . . . . .   | 510 |
| 10.59.3.3 take_as_init . . . . .   | 511 |
| 10.59.3.4 take_n_times . . . . .   | 511 |
| 10.59.3.5 to_result . . . . .  | 511 |
| 10.60mln::accu::stat::deviation< T, S, M > Struct Template Reference . . . . .   | 511 |
| 10.60.1 Detailed Description . . . . .   | 512 |
| 10.60.2 Member Function Documentation . . . . .                                  | 512 |
| 10.60.2.1 init . . . . .   | 512 |
| 10.60.2.2 is_valid . . . . .   | 512 |
| 10.60.2.3 take_as_init . . . . .   | 512 |
| 10.60.2.4 take_n_times . . . . .   | 512 |
| 10.60.2.5 to_result . . . . .  | 512 |
| 10.61mln::accu::stat::histo3d_rgb< V > Struct Template Reference . . . . .       | 513 |
| 10.61.1 Detailed Description . . . . .   | 513 |
| 10.61.2 Constructor & Destructor Documentation . . . . .                         | 514 |
| 10.61.2.1 histo3d_rgb . . . . .  | 514 |
| 10.61.3 Member Function Documentation . . . . .                                  | 514 |
| 10.61.3.1 init . . . . .   | 514 |



|           |  |     |
|-----------|--|-----|
| 10.61.3.2 | is_valid   | 514 |
| 10.61.3.3 | take   | 514 |
| 10.61.3.4 | take   | 514 |
| 10.61.3.5 | take_as_init   | 515 |
| 10.61.3.6 | take_n_times   | 515 |
| 10.61.3.7 | to_result  | 515 |
| 10.62     | mln::accu::stat::max< T > Struct Template Reference        | 515 |
| 10.62.1   | Detailed Description                                       | 516 |
| 10.62.2   | Member Function Documentation                              | 516 |
| 10.62.2.1 | init   | 516 |
| 10.62.2.2 | is_valid   | 516 |
| 10.62.2.3 | set_value  | 516 |
| 10.62.2.4 | take_as_init   | 516 |
| 10.62.2.5 | take_n_times   | 516 |
| 10.62.2.6 | to_result  | 517 |
| 10.63     | mln::accu::stat::max_h< V > Struct Template Reference      | 517 |
| 10.63.1   | Detailed Description                                       | 517 |
| 10.63.2   | Member Function Documentation                              | 517 |
| 10.63.2.1 | init   | 517 |
| 10.63.2.2 | is_valid   | 518 |
| 10.63.2.3 | take_as_init   | 518 |
| 10.63.2.4 | take_n_times   | 518 |
| 10.63.2.5 | to_result  | 518 |
| 10.64     | mln::accu::stat::mean< T, S, M > Struct Template Reference | 518 |
| 10.64.1   | Detailed Description                                       | 519 |
| 10.64.2   | Member Function Documentation                              | 519 |
| 10.64.2.1 | count  | 519 |
| 10.64.2.2 | init   | 519 |
| 10.64.2.3 | is_valid   | 519 |
| 10.64.2.4 | sum  | 519 |
| 10.64.2.5 | take_as_init   | 520 |
| 10.64.2.6 | take_n_times   | 520 |
| 10.64.2.7 | to_result  | 520 |
| 10.65     | mln::accu::stat::median_alt< S > Struct Template Reference | 520 |
| 10.65.1   | Detailed Description                                       | 521 |
| 10.65.2   | Member Function Documentation                              | 521 |

|           |  |     |
|-----------|--|-----|
| 10.65.2.1 | is_valid . . . . .   | 521 |
| 10.65.2.2 | take . . . . .   | 521 |
| 10.65.2.3 | take_as_init . . . . .   | 521 |
| 10.65.2.4 | take_n_times . . . . .   | 521 |
| 10.65.2.5 | to_result . . . . .  | 522 |
| 10.66     | mln::accu::stat::median_h< V > Struct Template Reference . . . . . | 522 |
| 10.66.1   | Detailed Description . . . . .                                     | 523 |
| 10.66.2   | Member Function Documentation . . . . .                            | 523 |
| 10.66.2.1 | init . . . . .   | 523 |
| 10.66.2.2 | is_valid . . . . .   | 523 |
| 10.66.2.3 | take_as_init . . . . .   | 523 |
| 10.66.2.4 | take_n_times . . . . .   | 523 |
| 10.66.2.5 | to_result . . . . .  | 523 |
| 10.67     | mln::accu::stat::meta::deviation Struct Reference . . . . .        | 523 |
| 10.67.1   | Detailed Description . . . . .                                     | 524 |
| 10.68     | mln::accu::stat::min< T > Struct Template Reference . . . . .      | 524 |
| 10.68.1   | Detailed Description . . . . .                                     | 525 |
| 10.68.2   | Member Function Documentation . . . . .                            | 525 |
| 10.68.2.1 | init . . . . .   | 525 |
| 10.68.2.2 | is_valid . . . . .   | 525 |
| 10.68.2.3 | set_value . . . . .  | 525 |
| 10.68.2.4 | take_as_init . . . . .   | 525 |
| 10.68.2.5 | take_n_times . . . . .   | 526 |
| 10.68.2.6 | to_result . . . . .  | 526 |
| 10.69     | mln::accu::stat::min_h< V > Struct Template Reference . . . . .    | 526 |
| 10.69.1   | Detailed Description . . . . .                                     | 526 |
| 10.69.2   | Member Function Documentation . . . . .                            | 527 |
| 10.69.2.1 | init . . . . .   | 527 |
| 10.69.2.2 | is_valid . . . . .   | 527 |
| 10.69.2.3 | take_as_init . . . . .   | 527 |
| 10.69.2.4 | take_n_times . . . . .   | 527 |
| 10.69.2.5 | to_result . . . . .  | 527 |
| 10.70     | mln::accu::stat::min_max< V > Struct Template Reference . . . . .  | 527 |
| 10.70.1   | Detailed Description . . . . .                                     | 529 |
| 10.70.2   | Member Function Documentation . . . . .                            | 529 |
| 10.70.2.1 | first . . . . .  | 529 |

|           |  |     |
|-----------|--|-----|
| 10.70.2.2 | <code>first_accu</code>  | 529 |
| 10.70.2.3 | <code>init</code>  | 529 |
| 10.70.2.4 | <code>is_valid</code>  | 529 |
| 10.70.2.5 | <code>second</code>  | 529 |
| 10.70.2.6 | <code>second_accu</code>   | 529 |
| 10.70.2.7 | <code>take_as_init</code>  | 530 |
| 10.70.2.8 | <code>take_n_times</code>  | 530 |
| 10.70.2.9 | <code>to_result</code>   | 530 |
| 10.71     | <code>mln::accu::stat::rank&lt; T &gt;</code> Struct Template Reference            | 530 |
| 10.71.1   | Detailed Description   | 531 |
| 10.71.2   | Member Function Documentation  | 531 |
| 10.71.2.1 | <code>init</code>  | 531 |
| 10.71.2.2 | <code>is_valid</code>  | 531 |
| 10.71.2.3 | <code>k</code>   | 531 |
| 10.71.2.4 | <code>take_as_init</code>  | 531 |
| 10.71.2.5 | <code>take_n_times</code>  | 531 |
| 10.71.2.6 | <code>to_result</code>   | 532 |
| 10.72     | <code>mln::accu::stat::rank&lt; bool &gt;</code> Struct Template Reference         | 532 |
| 10.72.1   | Detailed Description   | 532 |
| 10.72.2   | Member Function Documentation  | 532 |
| 10.72.2.1 | <code>init</code>  | 532 |
| 10.72.2.2 | <code>is_valid</code>  | 533 |
| 10.72.2.3 | <code>take_as_init</code>  | 533 |
| 10.72.2.4 | <code>take_n_times</code>  | 533 |
| 10.72.2.5 | <code>to_result</code>   | 533 |
| 10.73     | <code>mln::accu::stat::rank_high_quant&lt; T &gt;</code> Struct Template Reference | 533 |
| 10.73.1   | Detailed Description   | 534 |
| 10.73.2   | Member Function Documentation  | 534 |
| 10.73.2.1 | <code>init</code>  | 534 |
| 10.73.2.2 | <code>is_valid</code>  | 534 |
| 10.73.2.3 | <code>take_as_init</code>  | 534 |
| 10.73.2.4 | <code>take_n_times</code>  | 534 |
| 10.73.2.5 | <code>to_result</code>   | 534 |
| 10.74     | <code>mln::accu::stat::var&lt; T &gt;</code> Struct Template Reference             | 535 |
| 10.74.1   | Detailed Description   | 535 |
| 10.74.2   | Member Typedef Documentation   | 536 |

|            |  |     |
|------------|--|-----|
| 10.74.2.1  | mean_t   | 536 |
| 10.74.3    | Member Function Documentation                                  | 536 |
| 10.74.3.1  | init   | 536 |
| 10.74.3.2  | is_valid   | 536 |
| 10.74.3.3  | mean   | 536 |
| 10.74.3.4  | n_items  | 536 |
| 10.74.3.5  | take_as_init   | 536 |
| 10.74.3.6  | take_n_times   | 536 |
| 10.74.3.7  | to_result  | 537 |
| 10.74.3.8  | variance   | 537 |
| 10.75      | mln::accu::stat::variance< T, S, R > Struct Template Reference | 537 |
| 10.75.1    | Detailed Description   | 538 |
| 10.75.2    | Member Function Documentation                                  | 538 |
| 10.75.2.1  | init   | 538 |
| 10.75.2.2  | is_valid   | 538 |
| 10.75.2.3  | mean   | 538 |
| 10.75.2.4  | n_items  | 538 |
| 10.75.2.5  | standard_deviation   | 539 |
| 10.75.2.6  | sum  | 539 |
| 10.75.2.7  | take_as_init   | 539 |
| 10.75.2.8  | take_n_times   | 539 |
| 10.75.2.9  | to_result  | 539 |
| 10.75.2.10 | var  | 539 |
| 10.76      | mln::accu::tuple< A, n, > Struct Template Reference            | 539 |
| 10.76.1    | Detailed Description   | 540 |
| 10.76.2    | Member Function Documentation                                  | 540 |
| 10.76.2.1  | init   | 540 |
| 10.76.2.2  | is_valid   | 540 |
| 10.76.2.3  | take_as_init   | 541 |
| 10.76.2.4  | take_n_times   | 541 |
| 10.76.2.5  | to_result  | 541 |
| 10.77      | mln::accu::val< A > Struct Template Reference                  | 541 |
| 10.77.1    | Detailed Description   | 542 |
| 10.77.2    | Member Function Documentation                                  | 542 |
| 10.77.2.1  | init   | 542 |
| 10.77.2.2  | is_valid   | 542 |

---

|            |  |     |
|------------|--|-----|
| 10.77.2.3  | take_as_init                                     | 542 |
| 10.77.2.4  | take_n_times                                     | 542 |
| 10.77.2.5  | to_result  | 542 |
| 10.78mln:: | Accumulator< E > Struct Template Reference       | 542 |
| 10.78.1    | Detailed Description                             | 544 |
| 10.78.2    | Member Function Documentation                    | 544 |
| 10.78.2.1  | take_as_init                                     | 544 |
| 10.78.2.2  | take_n_times                                     | 544 |
| 10.79mln:: | algebra::h_mat< d, T > Struct Template Reference | 544 |
| 10.79.1    | Detailed Description                             | 545 |
| 10.79.2    | Member Enumeration Documentation                 | 545 |
| 10.79.2.1  | "@7  | 545 |
| 10.79.3    | Constructor & Destructor Documentation           | 545 |
| 10.79.3.1  | h_mat  | 545 |
| 10.79.3.2  | h_mat  | 545 |
| 10.79.4    | Member Function Documentation                    | 546 |
| 10.79.4.1  | _1   | 546 |
| 10.79.4.2  | t  | 546 |
| 10.80mln:: | algebra::h_vec< d, C > Class Template Reference  | 546 |
| 10.80.1    | Detailed Description                             | 547 |
| 10.80.2    | Member Enumeration Documentation                 | 547 |
| 10.80.2.1  | "@8  | 547 |
| 10.80.3    | Constructor & Destructor Documentation           | 547 |
| 10.80.3.1  | h_vec  | 547 |
| 10.80.3.2  | h_vec  | 547 |
| 10.80.4    | Member Function Documentation                    | 547 |
| 10.80.4.1  | operator mat< n, 1, U >                          | 547 |
| 10.80.4.2  | t  | 548 |
| 10.80.4.3  | to_vec   | 548 |
| 10.80.5    | Member Data Documentation                        | 548 |
| 10.80.5.1  | origin   | 548 |
| 10.80.5.2  | zero   | 548 |
| 10.81mln:: | bkd_pixterId< I > Class Template Reference       | 548 |
| 10.81.1    | Detailed Description                             | 549 |
| 10.81.2    | Member Typedef Documentation                     | 549 |
| 10.81.2.1  | image  | 549 |

---

|  |     |
|--|-----|
| 10.81.3 Constructor & Destructor Documentation . . . . .       | 549 |
| 10.81.3.1 bkd_pixter1d . . . . .                               | 549 |
| 10.81.4 Member Function Documentation . . . . .                | 549 |
| 10.81.4.1 next . . . . .                                       | 549 |
| 10.82mln::bkd_pixter2d< I > Class Template Reference . . . . . | 549 |
| 10.82.1 Detailed Description . . . . .                         | 550 |
| 10.82.2 Member Typedef Documentation . . . . .                 | 550 |
| 10.82.2.1 image . . . . .                                      | 550 |
| 10.82.3 Constructor & Destructor Documentation . . . . .       | 550 |
| 10.82.3.1 bkd_pixter2d . . . . .                               | 550 |
| 10.82.4 Member Function Documentation . . . . .                | 550 |
| 10.82.4.1 next . . . . .                                       | 550 |
| 10.83mln::bkd_pixter3d< I > Class Template Reference . . . . . | 551 |
| 10.83.1 Detailed Description . . . . .                         | 551 |
| 10.83.2 Member Typedef Documentation . . . . .                 | 551 |
| 10.83.2.1 image . . . . .                                      | 551 |
| 10.83.3 Constructor & Destructor Documentation . . . . .       | 552 |
| 10.83.3.1 bkd_pixter3d . . . . .                               | 552 |
| 10.83.4 Member Function Documentation . . . . .                | 552 |
| 10.83.4.1 next . . . . .                                       | 552 |
| 10.84mln::box< P > Class Template Reference . . . . .          | 552 |
| 10.84.1 Detailed Description . . . . .                         | 555 |
| 10.84.2 Member Typedef Documentation . . . . .                 | 555 |
| 10.84.2.1 bkd_piter . . . . .                                  | 555 |
| 10.84.2.2 element . . . . .                                    | 555 |
| 10.84.2.3 fwd_piter . . . . .                                  | 556 |
| 10.84.2.4 piter . . . . .                                      | 556 |
| 10.84.2.5 psite . . . . .                                      | 556 |
| 10.84.2.6 site . . . . .                                       | 556 |
| 10.84.3 Member Enumeration Documentation . . . . .             | 556 |
| 10.84.3.1 "@31 . . . . .                                       | 556 |
| 10.84.4 Constructor & Destructor Documentation . . . . .       | 556 |
| 10.84.4.1 box . . . . .  | 556 |
| 10.84.4.2 box . . . . .  | 556 |
| 10.84.4.3 box . . . . .  | 556 |
| 10.84.5 Member Function Documentation . . . . .                | 557 |

---

|            |  |     |
|------------|--|-----|
| 10.84.5.1  | bbox                                       | 557 |
| 10.84.5.2  | crop_wrt                                   | 557 |
| 10.84.5.3  | enlarge                                    | 557 |
| 10.84.5.4  | enlarge                                    | 557 |
| 10.84.5.5  | has  | 557 |
| 10.84.5.6  | is_empty                                   | 558 |
| 10.84.5.7  | is_valid                                   | 558 |
| 10.84.5.8  | len  | 558 |
| 10.84.5.9  | memory_size                                | 558 |
| 10.84.5.10 | merge                                      | 558 |
| 10.84.5.11 | nsites                                     | 558 |
| 10.84.5.12 | pcenter                                    | 559 |
| 10.84.5.13 | pmax                                       | 559 |
| 10.84.5.14 | pmax                                       | 559 |
| 10.84.5.15 | pmin                                       | 559 |
| 10.84.5.16 | pmin                                       | 559 |
| 10.84.5.17 | to_larger                                  | 559 |
| 10.84.6    | Friends And Related Function Documentation | 560 |
| 10.84.6.1  | operator<<                                 | 560 |
| 10.85      | mln::Box< E > Struct Template Reference    | 560 |
| 10.85.1    | Detailed Description                       | 562 |
| 10.85.2    | Member Function Documentation              | 563 |
| 10.85.2.1  | bbox                                       | 563 |
| 10.85.2.2  | is_empty                                   | 563 |
| 10.85.2.3  | len  | 563 |
| 10.85.2.4  | nsites                                     | 563 |
| 10.85.3    | Friends And Related Function Documentation | 564 |
| 10.85.3.1  | diff                                       | 564 |
| 10.85.3.2  | inter                                      | 564 |
| 10.85.3.3  | operator<                                  | 564 |
| 10.85.3.4  | operator<                                  | 564 |
| 10.85.3.5  | operator<<                                 | 564 |
| 10.85.3.6  | operator<=                                 | 565 |
| 10.85.3.7  | operator<=                                 | 565 |
| 10.85.3.8  | operator==                                 | 565 |
| 10.85.3.9  | sym_diff                                   | 565 |

|   |     |
|---|-----|
| 10.85.3.10uni . . . . .   | 565 |
| 10.85.3.1lunique . . . . .  | 566 |
| 10.86mln::box_runend_piter< P > Class Template Reference . . . . .                  | 566 |
| 10.86.1 Detailed Description . . . . .  | 566 |
| 10.86.2 Constructor & Destructor Documentation . . . . .                            | 566 |
| 10.86.2.1 box_runend_piter . . . . .  | 566 |
| 10.86.3 Member Function Documentation . . . . .                                     | 567 |
| 10.86.3.1 next . . . . .  | 567 |
| 10.86.3.2 run_length . . . . .  | 567 |
| 10.87mln::box_runstart_piter< P > Class Template Reference . . . . .                | 567 |
| 10.87.1 Detailed Description . . . . .  | 567 |
| 10.87.2 Constructor & Destructor Documentation . . . . .                            | 568 |
| 10.87.2.1 box_runstart_piter . . . . .  | 568 |
| 10.87.3 Member Function Documentation . . . . .                                     | 568 |
| 10.87.3.1 next . . . . .  | 568 |
| 10.87.3.2 run_length . . . . .  | 568 |
| 10.88mln::Browsing< E > Struct Template Reference . . . . .                         | 568 |
| 10.88.1 Detailed Description . . . . .  | 569 |
| 10.89mln::canvas::browsing::backdiagonal2d_t Struct Reference . . . . .             | 569 |
| 10.89.1 Detailed Description . . . . .  | 570 |
| 10.90mln::canvas::browsing::breadth_first_search_t Struct Reference . . . . .       | 571 |
| 10.90.1 Detailed Description . . . . .  | 571 |
| 10.91mln::canvas::browsing::depth_first_search_t Struct Reference . . . . .         | 571 |
| 10.91.1 Detailed Description . . . . .  | 571 |
| 10.92mln::canvas::browsing::diagonal2d_t Struct Reference . . . . .                 | 571 |
| 10.92.1 Detailed Description . . . . .  | 572 |
| 10.93mln::canvas::browsing::dir_struct_elt_incr_update_t Struct Reference . . . . . | 573 |
| 10.93.1 Detailed Description . . . . .  | 573 |
| 10.94mln::canvas::browsing::directional_t Struct Reference . . . . .                | 574 |
| 10.94.1 Detailed Description . . . . .  | 575 |
| 10.95mln::canvas::browsing::fwd_t Struct Reference . . . . .                        | 576 |
| 10.95.1 Detailed Description . . . . .  | 577 |
| 10.96mln::canvas::browsing::hyper_directional_t Struct Reference . . . . .          | 577 |
| 10.96.1 Detailed Description . . . . .  | 578 |
| 10.97mln::canvas::browsing::snake_fwd_t Struct Reference . . . . .                  | 579 |
| 10.97.1 Detailed Description . . . . .  | 579 |



|           |  |     |
|-----------|--|-----|
| 10.98     | <code>mln::canvas::browsing::snake_generic_t</code> Struct Reference                       | 580 |
| 10.98.1   | Detailed Description   | 581 |
| 10.99     | <code>mln::canvas::browsing::snake_vert_t</code> Struct Reference                          | 582 |
| 10.99.1   | Detailed Description   | 582 |
| 10.100    | <code>mln::canvas::chamfer&lt; F &gt;</code> Struct Template Reference                     | 583 |
| 10.100.   | Detailed Description   | 583 |
| 10.101    | <code>mln::category&lt; R(*) (A) &gt;</code> Struct Template Reference                     | 583 |
| 10.101.   | Detailed Description   | 583 |
| 10.102    | <code>mln::complex_image&lt; D, G, V &gt;</code> Class Template Reference                  | 584 |
| 10.102.   | Detailed Description   | 585 |
| 10.102.   | Member Typedef Documentation   | 585 |
| 10.102.2. | <code>geom</code>  | 585 |
| 10.102.2. | <code>value</code>   | 585 |
| 10.102.2. | <code>value</code>   | 585 |
| 10.102.2. | <code>skeleton</code>  | 585 |
| 10.102.2. | <code>value</code>   | 585 |
| 10.102.   | Constructor & Destructor Documentation   | 586 |
| 10.102.3. | <code>lcomplex_image</code>  | 586 |
| 10.102.   | Member Function Documentation  | 586 |
| 10.102.4. | <code>ldomain</code>   | 586 |
| 10.102.4. | <code>operator()</code>  | 586 |
| 10.102.4. | <code>operator()</code>  | 586 |
| 10.102.4. | <code>values</code>  | 586 |
| 10.102.   | Member Data Documentation  | 586 |
| 10.102.5. | <code>ldim</code>  | 586 |
| 10.103    | <code>mln::complex_neighborhood_bkd_piter&lt; I, G, N &gt;</code> Class Template Reference | 587 |
| 10.103.   | Detailed Description   | 587 |
| 10.103.   | Member Typedef Documentation   | 587 |
| 10.103.2. | <code>liter_type</code>  | 587 |
| 10.103.2. | <code>psite</code>   | 588 |
| 10.103.   | Constructor & Destructor Documentation   | 588 |
| 10.103.3. | <code>lcomplex_neighborhood_bkd_piter</code>   | 588 |
| 10.103.   | Member Function Documentation  | 588 |
| 10.103.4. | <code>liter</code>   | 588 |
| 10.103.4. | <code>next</code>  | 588 |
| 10.104    | <code>mln::complex_neighborhood_fwd_piter&lt; I, G, N &gt;</code> Class Template Reference | 588 |

|           |   |     |
|-----------|---|-----|
| 10.104.   | Detailed Description  | 589 |
| 10.104.   | Member Typedef Documentation                                      | 589 |
| 10.104.2. | liter_type  | 589 |
| 10.104.2. | psite   | 589 |
| 10.104.   | Constructor & Destructor Documentation                            | 589 |
| 10.104.3. | lcomplex_neighborhood_fwd_piter                                   | 589 |
| 10.104.   | Member Function Documentation                                     | 590 |
| 10.104.4. | liter   | 590 |
| 10.104.4. | next  | 590 |
| 10.105.   | mln::complex_psite< D, G > Class Template Reference               | 590 |
| 10.105.   | Detailed Description  | 591 |
| 10.105.   | Constructor & Destructor Documentation                            | 591 |
| 10.105.2. | lcomplex_psite  | 591 |
| 10.105.2. | complex_psite   | 591 |
| 10.105.   | Member Function Documentation                                     | 591 |
| 10.105.3. | lchange_target  | 591 |
| 10.105.3. | face  | 592 |
| 10.105.3. | face_id   | 592 |
| 10.105.3. | invalidate  | 592 |
| 10.105.3. | is_valid  | 592 |
| 10.105.3. | mln   | 592 |
| 10.105.3. | site_set  | 593 |
| 10.106.   | mln::complex_window_bkd_piter< I, G, W > Class Template Reference | 593 |
| 10.106.   | Detailed Description  | 593 |
| 10.106.   | Member Typedef Documentation                                      | 594 |
| 10.106.2. | liter_type  | 594 |
| 10.106.2. | psite   | 594 |
| 10.106.   | Constructor & Destructor Documentation                            | 594 |
| 10.106.3. | lcomplex_window_bkd_piter   | 594 |
| 10.106.   | Member Function Documentation                                     | 594 |
| 10.106.4. | liter   | 594 |
| 10.106.4. | next  | 594 |
| 10.107.   | mln::complex_window_fwd_piter< I, G, W > Class Template Reference | 595 |
| 10.107.   | Detailed Description  | 595 |
| 10.107.   | Member Typedef Documentation                                      | 595 |
| 10.107.2. | liter_type  | 595 |

|  |     |
|--|-----|
| 10.107.2.2psite . . . . .  | 596 |
| 10.107.3.Constructor & Destructor Documentation . . . . .                | 596 |
| 10.107.3.1complex_window_fwd_piter . . . . .                             | 596 |
| 10.107.4.Member Function Documentation . . . . .                         | 596 |
| 10.107.4.1iter . . . . .   | 596 |
| 10.107.4.2next . . . . .   | 596 |
| 10.108.mln::decorated_image< I, D > Struct Template Reference . . . . .  | 596 |
| 10.108.1.Detailed Description . . . . .                                  | 597 |
| 10.108.2.Member Typedef Documentation . . . . .                          | 597 |
| 10.108.2.1lvalue . . . . .   | 597 |
| 10.108.2.2psite . . . . .  | 598 |
| 10.108.2.3rvalue . . . . .   | 598 |
| 10.108.2.4skeleton . . . . .   | 598 |
| 10.108.3.Constructor & Destructor Documentation . . . . .                | 598 |
| 10.108.3.1decorated_image . . . . .                                      | 598 |
| 10.108.3.2~decorated_image . . . . .                                     | 598 |
| 10.108.4.Member Function Documentation . . . . .                         | 598 |
| 10.108.4.1decoration . . . . .   | 598 |
| 10.108.4.2decoration . . . . .   | 598 |
| 10.108.4.3operator decorated_image< const I, D > . . . . .               | 599 |
| 10.108.4.4operator() . . . . .   | 599 |
| 10.108.4.5operator() . . . . .   | 599 |
| 10.109.mln::Delta_Point_Site< void > Struct Template Reference . . . . . | 599 |
| 10.109.1.Detailed Description . . . . .                                  | 599 |
| 10.110.mln::doc::Accumulator< E > Struct Template Reference . . . . .    | 599 |
| 10.110.1.Detailed Description . . . . .                                  | 600 |
| 10.110.2.Member Typedef Documentation . . . . .                          | 600 |
| 10.110.2.1argument . . . . .   | 600 |
| 10.110.3.Member Function Documentation . . . . .                         | 600 |
| 10.110.3.1init . . . . .   | 600 |
| 10.110.3.2take . . . . .   | 600 |
| 10.110.3.3take . . . . .   | 600 |
| 10.111.mln::doc::Box< E > Struct Template Reference . . . . .            | 601 |
| 10.111.1.Detailed Description . . . . .                                  | 602 |
| 10.111.2.Member Typedef Documentation . . . . .                          | 602 |
| 10.111.2.1bkd_piter . . . . .  | 602 |

|             |   |     |
|-------------|---|-----|
| 10.111.2.2  | 2d_fwd_piter  | 602 |
| 10.111.2.3  | psite   | 602 |
| 10.111.2.4  | site  | 602 |
| 10.111.3    | Member Function Documentation                         | 603 |
| 10.111.3.1  | bbox  | 603 |
| 10.111.3.2  | has   | 603 |
| 10.111.3.3  | nsites  | 603 |
| 10.111.3.4  | pmax  | 603 |
| 10.111.3.5  | pmin  | 603 |
| 10.112      | nl::doc::Dpoint< E > Struct Template Reference        | 604 |
| 10.112.1    | Detailed Description                                  | 604 |
| 10.112.2    | Member Typedef Documentation                          | 605 |
| 10.112.2.1  | coord   | 605 |
| 10.112.2.2  | dpoint  | 605 |
| 10.112.2.3  | point   | 605 |
| 10.112.3    | Member Enumeration Documentation                      | 605 |
| 10.112.3.1  | "@19  | 605 |
| 10.112.4    | Member Function Documentation                         | 605 |
| 10.112.4.1  | operator[]  | 605 |
| 10.113      | nl::doc::Fastest_Image< E > Struct Template Reference | 606 |
| 10.113.1    | Detailed Description                                  | 608 |
| 10.113.2    | Member Typedef Documentation                          | 608 |
| 10.113.2.1  | bkd_piter   | 608 |
| 10.113.2.2  | coord   | 608 |
| 10.113.2.3  | dpoint  | 609 |
| 10.113.2.4  | fwd_piter   | 609 |
| 10.113.2.5  | value   | 609 |
| 10.113.2.6  | point   | 609 |
| 10.113.2.7  | pset  | 609 |
| 10.113.2.8  | psite   | 609 |
| 10.113.2.9  | rvalue  | 610 |
| 10.113.2.10 | skeleton  | 610 |
| 10.113.2.11 | value   | 610 |
| 10.113.2.12 | set   | 610 |
| 10.113.3    | Member Function Documentation                         | 610 |
| 10.113.3.1  | bbox  | 610 |

|   |     |
|---|-----|
| 10.113.3.2border  | 611 |
| 10.113.3.3buffer  | 611 |
| 10.113.3.4delta_index   | 611 |
| 10.113.3.5domain  | 611 |
| 10.113.3.6has   | 611 |
| 10.113.3.7has   | 612 |
| 10.113.3.8s_valid   | 612 |
| 10.113.3.9elements  | 612 |
| 10.113.3.10sites  | 612 |
| 10.113.3.1operator()  | 612 |
| 10.113.3.12operator()   | 613 |
| 10.113.3.13operator[]   | 613 |
| 10.113.3.14operator[]   | 613 |
| 10.113.3.15point_at_index   | 614 |
| 10.113.3.16values   | 614 |
| 10.114.1In::doc::Generalized_Pixel< E > Struct Template Reference | 614 |
| 10.114.1.Detailed Description                                     | 615 |
| 10.114.2Member Typedef Documentation                              | 615 |
| 10.114.2.1image   | 615 |
| 10.114.2.2value   | 615 |
| 10.114.2.3value   | 615 |
| 10.114.3Member Function Documentation                             | 616 |
| 10.114.3.1ima   | 616 |
| 10.114.3.2val   | 616 |
| 10.115.1In::doc::Image< E > Struct Template Reference             | 616 |
| 10.115.1.Detailed Description                                     | 618 |
| 10.115.2Member Typedef Documentation                              | 618 |
| 10.115.2.1bkd_piter   | 618 |
| 10.115.2.2coord   | 618 |
| 10.115.2.3dpoint  | 618 |
| 10.115.2.4fwd_piter   | 619 |
| 10.115.2.5value   | 619 |
| 10.115.2.6point   | 619 |
| 10.115.2.7pset  | 619 |
| 10.115.2.8psite   | 619 |
| 10.115.2.9value   | 619 |

|  |     |
|--|-----|
| 10.115.2.1 <b>keleton</b> . . . . .  | 620 |
| 10.115.2.1 <b>value</b> . . . . .  | 620 |
| 10.115.2.1 <b>set</b> . . . . .  | 620 |
| 10.115.3 <b>Member Function Documentation</b> . . . . .                              | 620 |
| 10.115.3.1 <b>bbox</b> . . . . .   | 620 |
| 10.115.3.2 <b>domain</b> . . . . .   | 620 |
| 10.115.3.3 <b>has</b> . . . . .  | 620 |
| 10.115.3.4 <b>has</b> . . . . .  | 621 |
| 10.115.3.5 <b>is_valid</b> . . . . .   | 621 |
| 10.115.3.6 <b>nsites</b> . . . . .   | 621 |
| 10.115.3.7 <b>operator()</b> . . . . .   | 621 |
| 10.115.3.8 <b>operator()</b> . . . . .   | 621 |
| 10.115.3.9 <b>values</b> . . . . .   | 622 |
| 10.116 <b>ln::doc::Iterator&lt; E &gt; Struct Template Reference</b> . . . . .       | 622 |
| 10.116.1 <b>Detailed Description</b> . . . . .                                       | 622 |
| 10.116.2 <b>Member Function Documentation</b> . . . . .                              | 623 |
| 10.116.2.1 <b>invalidate</b> . . . . .   | 623 |
| 10.116.2.2 <b>is_valid</b> . . . . .   | 623 |
| 10.116.2.3 <b>start</b> . . . . .  | 623 |
| 10.117 <b>ln::doc::Neighborhood&lt; E &gt; Struct Template Reference</b> . . . . .   | 623 |
| 10.117.1 <b>Detailed Description</b> . . . . .                                       | 624 |
| 10.117.2 <b>Member Typedef Documentation</b> . . . . .                               | 624 |
| 10.117.2.1 <b>bkd_niter</b> . . . . .  | 624 |
| 10.117.2.2 <b>dpoint</b> . . . . .   | 624 |
| 10.117.2.3 <b>fwd_niter</b> . . . . .  | 624 |
| 10.117.2.4 <b>niter</b> . . . . .  | 624 |
| 10.117.2.5 <b>point</b> . . . . .  | 624 |
| 10.118 <b>ln::doc::Object&lt; E &gt; Struct Template Reference</b> . . . . .         | 625 |
| 10.118.1 <b>Detailed Description</b> . . . . .                                       | 625 |
| 10.119 <b>ln::doc::Pixel_Iterator&lt; E &gt; Struct Template Reference</b> . . . . . | 625 |
| 10.119.1 <b>Detailed Description</b> . . . . .                                       | 627 |
| 10.119.2 <b>Member Typedef Documentation</b> . . . . .                               | 627 |
| 10.119.2.1 <b>image</b> . . . . .  | 627 |
| 10.119.2.2 <b>value</b> . . . . .  | 627 |
| 10.119.2.3 <b>value</b> . . . . .  | 627 |
| 10.119.2.4 <b>value</b> . . . . .  | 627 |

|            |   |     |
|------------|---|-----|
| 10.119.3   | Member Function Documentation                         | 627 |
| 10.119.3.1 | lima  | 627 |
| 10.119.3.2 | invalidate  | 628 |
| 10.119.3.3 | is_valid  | 628 |
| 10.119.3.4 | start   | 628 |
| 10.119.3.5 | val   | 628 |
| 10.120     | ln::doc::Point_Site< E > Struct Template Reference    | 628 |
| 10.120.1   | Detailed Description                                  | 629 |
| 10.120.2   | Member Typedef Documentation                          | 629 |
| 10.120.2.1 | coord   | 629 |
| 10.120.2.2 | dpoint  | 629 |
| 10.120.2.3 | mesh  | 629 |
| 10.120.2.4 | point   | 630 |
| 10.120.3   | Member Enumeration Documentation                      | 630 |
| 10.120.3.1 | "@20  | 630 |
| 10.120.4   | Member Function Documentation                         | 630 |
| 10.120.4.1 | operator[]  | 630 |
| 10.120.4.2 | to_point  | 630 |
| 10.121     | ln::doc::Site_Iterator< E > Struct Template Reference | 631 |
| 10.121.1   | Detailed Description                                  | 632 |
| 10.121.2   | Member Typedef Documentation                          | 632 |
| 10.121.2.1 | psite   | 632 |
| 10.121.3   | Member Function Documentation                         | 632 |
| 10.121.3.1 | invalidate  | 632 |
| 10.121.3.2 | is_valid  | 632 |
| 10.121.3.3 | operator psite  | 632 |
| 10.121.3.4 | start   | 632 |
| 10.122     | ln::doc::Site_Set< E > Struct Template Reference      | 633 |
| 10.122.1   | Detailed Description                                  | 634 |
| 10.122.2   | Member Typedef Documentation                          | 634 |
| 10.122.2.1 | l_bkd_piter   | 634 |
| 10.122.2.2 | l_fwd_piter   | 634 |
| 10.122.2.3 | psite   | 634 |
| 10.122.2.4 | site  | 634 |
| 10.122.3   | Member Function Documentation                         | 634 |
| 10.122.3.1 | l_has   | 634 |

|            |   |     |
|------------|---|-----|
| 10.123     | <a href="#">std::Value_Iterator&lt; E &gt; Struct Template Reference</a>  | 635 |
| 10.123.1   | Detailed Description  | 636 |
| 10.123.2   | Member Typedef Documentation  | 636 |
| 10.123.2.1 | lvalue  | 636 |
| 10.123.3   | Member Function Documentation   | 636 |
| 10.123.3.1 | invalidate  | 636 |
| 10.123.3.2 | is_valid  | 636 |
| 10.123.3.3 | operator value  | 636 |
| 10.123.3.4 | start   | 636 |
| 10.124     | <a href="#">std::Value_Set&lt; E &gt; Struct Template Reference</a>       | 636 |
| 10.124.1   | Detailed Description  | 637 |
| 10.124.2   | Member Typedef Documentation  | 638 |
| 10.124.2.1 | l_bkd_viter   | 638 |
| 10.124.2.2 | fwd_viter   | 638 |
| 10.124.2.3 | value   | 638 |
| 10.124.3   | Member Function Documentation   | 638 |
| 10.124.3.1 | lhas  | 638 |
| 10.124.3.2 | index_of  | 638 |
| 10.124.3.3 | nvalues   | 638 |
| 10.124.3.4 | operator[]  | 639 |
| 10.125     | <a href="#">std::Weighted_Window&lt; E &gt; Struct Template Reference</a> | 639 |
| 10.125.1   | Detailed Description  | 640 |
| 10.125.2   | Member Typedef Documentation  | 640 |
| 10.125.2.1 | l_bkd_qiter   | 640 |
| 10.125.2.2 | dpoint  | 640 |
| 10.125.2.3 | fwd_qiter   | 640 |
| 10.125.2.4 | point   | 641 |
| 10.125.2.5 | weight  | 641 |
| 10.125.2.6 | window  | 641 |
| 10.125.3   | Member Function Documentation   | 641 |
| 10.125.3.1 | ldelta  | 641 |
| 10.125.3.2 | is_centered   | 641 |
| 10.125.3.3 | is_empty  | 641 |
| 10.125.3.4 | sym   | 641 |
| 10.125.3.5 | win   | 641 |
| 10.126     | <a href="#">std::Window&lt; E &gt; Struct Template Reference</a>          | 641 |



|  |     |
|--|-----|
| 10.126. Detailed Description   | 642 |
| 10.126. Member Typedef Documentation   | 642 |
| 10.126.2.1 bkd_qiter   | 642 |
| 10.126.2.2 fwd_qiter   | 643 |
| 10.126.2.3 qiter   | 643 |
| 10.127. <code>mln::Dpoint&lt; E &gt;</code> Struct Template Reference                    | 643 |
| 10.127. Detailed Description   | 643 |
| 10.127. Member Function Documentation  | 643 |
| 10.127.2. <code>lto_dpoint</code>  | 643 |
| 10.128. <code>mln::dpoint&lt; G, C &gt;</code> Struct Template Reference                 | 644 |
| 10.128. Detailed Description   | 645 |
| 10.128. Member Typedef Documentation   | 645 |
| 10.128.2.1 <code>lcoord</code>   | 645 |
| 10.128.2.2 <code>grid</code>   | 646 |
| 10.128.2.3 <code>psite</code>  | 646 |
| 10.128.2.4 <code>site</code>   | 646 |
| 10.128.2.5 <code>vec</code>  | 646 |
| 10.128. Member Enumeration Documentation   | 646 |
| 10.128.3.1 <code>"@22</code>   | 646 |
| 10.128. Constructor & Destructor Documentation   | 646 |
| 10.128.4.1 <code>ldpoint</code>  | 646 |
| 10.128.4.2 <code>2dpoint</code>  | 646 |
| 10.128.4.3 <code>3dpoint</code>  | 647 |
| 10.128.4.4 <code>4dpoint</code>  | 647 |
| 10.128.4.5 <code>5dpoint</code>  | 647 |
| 10.128. Member Function Documentation  | 647 |
| 10.128.5.1 <code>operator mln::algebra::vec&lt; dpoint&lt; G, C &gt;::dim, Q &gt;</code> | 647 |
| 10.128.5.2 <code>operator[]</code>   | 647 |
| 10.128.5.3 <code>operator[]</code>   | 647 |
| 10.128.5.4 <code>set_all</code>  | 648 |
| 10.128.5.5 <code>to_vec</code>   | 648 |
| 10.129. <code>mln::dpoints_bkd_pixter&lt; I &gt;</code> Class Template Reference         | 648 |
| 10.129. Detailed Description   | 650 |
| 10.129. Constructor & Destructor Documentation   | 650 |
| 10.129.2.1 <code>ldpoints_bkd_pixter</code>  | 650 |
| 10.129.2.2 <code>2dpoints_bkd_pixter</code>  | 650 |

|            |   |     |
|------------|---|-----|
| 10.129.3   | Member Function Documentation                         | 651 |
| 10.129.3.1 | center_val  | 651 |
| 10.129.3.2 | invalidate  | 651 |
| 10.129.3.3 | is_valid  | 651 |
| 10.129.3.4 | next  | 651 |
| 10.129.3.5 | start   | 651 |
| 10.129.3.6 | update  | 651 |
| 10.130     | mln::dpoints_fwd_pixter< I > Class Template Reference | 652 |
| 10.130.1   | Detailed Description                                  | 653 |
| 10.130.2   | Constructor & Destructor Documentation                | 653 |
| 10.130.2.1 | dpoints_fwd_pixter                                    | 653 |
| 10.130.2.2 | dpoints_fwd_pixter                                    | 653 |
| 10.130.3   | Member Function Documentation                         | 654 |
| 10.130.3.1 | center_val  | 654 |
| 10.130.3.2 | invalidate  | 654 |
| 10.130.3.3 | is_valid  | 654 |
| 10.130.3.4 | next  | 654 |
| 10.130.3.5 | start   | 654 |
| 10.130.3.6 | update  | 654 |
| 10.131     | mln::Edge< E > Struct Template Reference              | 655 |
| 10.131.1   | Detailed Description                                  | 655 |
| 10.132     | mln::edge_image< P, V, G > Class Template Reference   | 655 |
| 10.132.1   | Detailed Description                                  | 656 |
| 10.132.2   | Member Typedef Documentation                          | 656 |
| 10.132.2.1 | ledge_nbh_t   | 656 |
| 10.132.2.2 | edge_win_t  | 656 |
| 10.132.2.3 | graph_t   | 656 |
| 10.132.2.4 | nbh_t   | 656 |
| 10.132.2.5 | site_function_t                                       | 657 |
| 10.132.2.6 | skeleton  | 657 |
| 10.132.2.7 | win_t   | 657 |
| 10.132.3   | Constructor & Destructor Documentation                | 657 |
| 10.132.3.1 | ledge_image   | 657 |
| 10.132.4   | Member Function Documentation                         | 657 |
| 10.132.4.1 | operator()  | 657 |
| 10.133     | mln::extended< I > Struct Template Reference          | 657 |

---

|   |     |
|---|-----|
| 10.133. Detailed Description  | 658 |
| 10.133. Member Typedef Documentation  | 658 |
| 10.133.2.1 skeleton   | 658 |
| 10.133.2.2 value  | 658 |
| 10.133. Constructor & Destructor Documentation                                | 658 |
| 10.133.3.1 extended   | 658 |
| 10.133.3.2 extended   | 659 |
| 10.133. Member Function Documentation   | 659 |
| 10.133.4.1 domain   | 659 |
| 10.134. <code>ln::extension_fun&lt; I, F &gt;</code> Class Template Reference | 659 |
| 10.134. Detailed Description  | 660 |
| 10.134. Member Typedef Documentation  | 660 |
| 10.134.2.1 rvalue   | 660 |
| 10.134.2.2 skeleton   | 660 |
| 10.134.2.3 value  | 660 |
| 10.134. Constructor & Destructor Documentation                                | 660 |
| 10.134.3.1 extension_fun  | 660 |
| 10.134.3.2 extension_fun  | 661 |
| 10.134. Member Function Documentation   | 661 |
| 10.134.4.1 extension  | 661 |
| 10.134.4.2 has  | 661 |
| 10.134.4.3 operator()   | 661 |
| 10.134.4.4 operator()   | 661 |
| 10.135. <code>ln::extension_ima&lt; I, J &gt;</code> Class Template Reference | 661 |
| 10.135. Detailed Description  | 662 |
| 10.135. Member Typedef Documentation  | 662 |
| 10.135.2.1 rvalue   | 662 |
| 10.135.2.2 skeleton   | 663 |
| 10.135.2.3 value  | 663 |
| 10.135. Constructor & Destructor Documentation                                | 663 |
| 10.135.3.1 extension_ima  | 663 |
| 10.135.3.2 extension_ima  | 663 |
| 10.135. Member Function Documentation   | 663 |
| 10.135.4.1 extension  | 663 |
| 10.135.4.2 has  | 663 |
| 10.135.4.3 operator()   | 663 |



|   |     |
|---|-----|
| 10.138.3.Constructor & Destructor Documentation . . . . .                         | 670 |
| 10.138.3.1flat_image . . . . .  | 670 |
| 10.138.3.2flat_image . . . . .  | 671 |
| 10.138.4.Member Function Documentation . . . . .                                  | 671 |
| 10.138.4.1domain . . . . .  | 671 |
| 10.138.4.2has . . . . .   | 671 |
| 10.138.4.3operator() . . . . .  | 671 |
| 10.138.4.4operator() . . . . .  | 671 |
| 10.139.mn::fun::from_accu< A > Struct Template Reference . . . . .                | 671 |
| 10.139.Detailed Description . . . . .   | 671 |
| 10.140.mn::fun::n2v::white_gaussian< V > Struct Template Reference . . . . .      | 672 |
| 10.140.Detailed Description . . . . .   | 672 |
| 10.141.mn::fun::p2b::antilogy Struct Reference . . . . .                          | 672 |
| 10.141.Detailed Description . . . . .   | 673 |
| 10.142.mn::fun::p2b::tautology Struct Reference . . . . .                         | 673 |
| 10.142.Detailed Description . . . . .   | 674 |
| 10.143.mn::fun::v2b::Inot< V > Struct Template Reference . . . . .                | 675 |
| 10.143.Detailed Description . . . . .   | 675 |
| 10.144.mn::fun::v2b::threshold< V > Struct Template Reference . . . . .           | 676 |
| 10.144.Detailed Description . . . . .   | 676 |
| 10.145.mn::fun::v2v::ch_function_value< F, V > Class Template Reference . . . . . | 677 |
| 10.145.Detailed Description . . . . .   | 677 |
| 10.146.mn::fun::v2v::component< T, i > Struct Template Reference . . . . .        | 678 |
| 10.146.Detailed Description . . . . .   | 678 |
| 10.147.mn::fun::v2v::l1_norm< V, R > Struct Template Reference . . . . .          | 678 |
| 10.147.Detailed Description . . . . .   | 679 |
| 10.148.mn::fun::v2v::l2_norm< V, R > Struct Template Reference . . . . .          | 679 |
| 10.148.Detailed Description . . . . .   | 680 |
| 10.149.mn::fun::v2v::linear< V, T, R > Struct Template Reference . . . . .        | 680 |
| 10.149.Detailed Description . . . . .   | 681 |
| 10.150.mn::fun::v2v::linfty_norm< V, R > Struct Template Reference . . . . .      | 681 |
| 10.150.Detailed Description . . . . .   | 682 |
| 10.151.mn::fun::v2v::rgb8_to_rgbn< n > Struct Template Reference . . . . .        | 682 |
| 10.151.Detailed Description . . . . .   | 683 |
| 10.151.2.Member Function Documentation . . . . .                                  | 684 |
| 10.151.2.1operator() . . . . .  | 684 |

|        |  |     |
|--------|--|-----|
| 10.152 | <code>ln::fun::v2w2v::cos&lt; V &gt;</code> Struct Template Reference              | 684 |
|        | 10.152. Detailed Description   | 685 |
| 10.153 | <code>ln::fun::v2w_w2v::l1_norm&lt; V, R &gt;</code> Struct Template Reference     | 685 |
|        | 10.153. Detailed Description   | 686 |
| 10.154 | <code>ln::fun::v2w_w2v::l2_norm&lt; V, R &gt;</code> Struct Template Reference     | 687 |
|        | 10.154. Detailed Description   | 687 |
| 10.155 | <code>ln::fun::v2w_w2v::linfty_norm&lt; V, R &gt;</code> Struct Template Reference | 688 |
|        | 10.155. Detailed Description   | 688 |
| 10.156 | <code>ln::fun::vv2b::eq&lt; L, R &gt;</code> Struct Template Reference             | 689 |
|        | 10.156. Detailed Description   | 689 |
| 10.157 | <code>ln::fun::vv2b::ge&lt; L, R &gt;</code> Struct Template Reference             | 689 |
|        | 10.157. Detailed Description   | 690 |
| 10.158 | <code>ln::fun::vv2b::gt&lt; L, R &gt;</code> Struct Template Reference             | 690 |
|        | 10.158. Detailed Description   | 691 |
| 10.159 | <code>ln::fun::vv2b::implies&lt; L, R &gt;</code> Struct Template Reference        | 691 |
|        | 10.159. Detailed Description   | 692 |
| 10.160 | <code>ln::fun::vv2b::le&lt; L, R &gt;</code> Struct Template Reference             | 692 |
|        | 10.160. Detailed Description   | 693 |
| 10.161 | <code>ln::fun::vv2b::lt&lt; L, R &gt;</code> Struct Template Reference             | 693 |
|        | 10.161. Detailed Description   | 694 |
| 10.162 | <code>ln::fun::vv2v::diff_abs&lt; V &gt;</code> Struct Template Reference          | 694 |
|        | 10.162. Detailed Description   | 695 |
| 10.163 | <code>ln::fun::vv2v::land&lt; L, R &gt;</code> Struct Template Reference           | 695 |
|        | 10.163. Detailed Description   | 696 |
| 10.164 | <code>ln::fun::vv2v::land_not&lt; L, R &gt;</code> Struct Template Reference       | 696 |
|        | 10.164. Detailed Description   | 697 |
| 10.165 | <code>ln::fun::vv2v::lor&lt; L, R &gt;</code> Struct Template Reference            | 697 |
|        | 10.165. Detailed Description   | 698 |
| 10.166 | <code>ln::fun::vv2v::lxor&lt; L, R &gt;</code> Struct Template Reference           | 698 |
|        | 10.166. Detailed Description   | 699 |
| 10.167 | <code>ln::fun::vv2v::max&lt; V &gt;</code> Struct Template Reference               | 699 |
|        | 10.167. Detailed Description   | 700 |
| 10.168 | <code>ln::fun::vv2v::min&lt; L, R &gt;</code> Struct Template Reference            | 700 |
|        | 10.168. Detailed Description   | 701 |
| 10.169 | <code>ln::fun::vv2v::vec&lt; V &gt;</code> Struct Template Reference               | 701 |
|        | 10.169. Detailed Description   | 702 |

|          |  |     |
|----------|--|-----|
| 10.170   | <a href="#">ln::fun::x2p::closest_point&lt; P &gt; Struct Template Reference</a> | 702 |
| 10.170   | <a href="#">Detailed Description</a>   | 703 |
| 10.171   | <a href="#">ln::fun::x2v::bilinear&lt; I &gt; Struct Template Reference</a>      | 703 |
| 10.171   | <a href="#">Detailed Description</a>   | 703 |
| 10.171   | <a href="#">Member Function Documentation</a>                                    | 703 |
| 10.171.2 | <a href="#">operator()</a>   | 703 |
| 10.171.2 | <a href="#">operator()</a>   | 704 |
| 10.172   | <a href="#">ln::fun::x2v::trilinear&lt; I &gt; Struct Template Reference</a>     | 704 |
| 10.172   | <a href="#">Detailed Description</a>   | 704 |
| 10.173   | <a href="#">ln::fun::x2x::composed&lt; T2, T1 &gt; Struct Template Reference</a> | 704 |
| 10.173   | <a href="#">Detailed Description</a>   | 704 |
| 10.173   | <a href="#">Constructor &amp; Destructor Documentation</a>                       | 705 |
| 10.173.2 | <a href="#">lcomposed</a>  | 705 |
| 10.173.2 | <a href="#">ocomposed</a>  | 705 |
| 10.174   | <a href="#">ln::fun::x2x::linear&lt; I &gt; Struct Template Reference</a>        | 705 |
| 10.174   | <a href="#">Detailed Description</a>   | 706 |
| 10.174   | <a href="#">Constructor &amp; Destructor Documentation</a>                       | 706 |
| 10.174.2 | <a href="#">linear</a>   | 706 |
| 10.174   | <a href="#">Member Function Documentation</a>                                    | 706 |
| 10.174.3 | <a href="#">operator()</a>   | 706 |
| 10.174   | <a href="#">Member Data Documentation</a>  | 706 |
| 10.174.4 | <a href="#">lima</a>   | 706 |
| 10.175   | <a href="#">ln::fun::x2x::rotation&lt; n, C &gt; Struct Template Reference</a>   | 706 |
| 10.175   | <a href="#">Detailed Description</a>   | 708 |
| 10.175   | <a href="#">Member Typedef Documentation</a>                                     | 708 |
| 10.175.2 | <a href="#">ldata_t</a>  | 708 |
| 10.175.2 | <a href="#">invert</a>   | 708 |
| 10.175   | <a href="#">Constructor &amp; Destructor Documentation</a>                       | 708 |
| 10.175.3 | <a href="#">lrotation</a>  | 708 |
| 10.175.3 | <a href="#">rotation</a>   | 709 |
| 10.175.3 | <a href="#">rotation</a>   | 709 |
| 10.175.3 | <a href="#">rotation</a>   | 709 |
| 10.175   | <a href="#">Member Function Documentation</a>                                    | 709 |
| 10.175.4 | <a href="#">linv</a>   | 709 |
| 10.175.4 | <a href="#">operator()</a>   | 709 |
| 10.175.4 | <a href="#">set_alpha</a>  | 709 |





|          |   |     |
|----------|---|-----|
| 10.181   | <a href="#">Inl::Function_v2b&lt; E &gt; Struct Template Reference</a>  | 716 |
| 10.181   | <a href="#">Detailed Description</a>                                    | 717 |
| 10.182   | <a href="#">Inl::Function_v2v&lt; E &gt; Struct Template Reference</a>  | 717 |
| 10.182   | <a href="#">Detailed Description</a>                                    | 718 |
| 10.183   | <a href="#">Inl::Function_vv2b&lt; E &gt; Struct Template Reference</a> | 718 |
| 10.183   | <a href="#">Detailed Description</a>                                    | 718 |
| 10.184   | <a href="#">Inl::Function_vv2v&lt; E &gt; Struct Template Reference</a> | 718 |
| 10.184   | <a href="#">Detailed Description</a>                                    | 719 |
| 10.185   | <a href="#">Inl::fwd_pixter1d&lt; I &gt; Class Template Reference</a>   | 719 |
| 10.185   | <a href="#">Detailed Description</a>                                    | 720 |
| 10.185   | <a href="#">Member Typedef Documentation</a>                            | 720 |
| 10.185.2 | <a href="#">limage</a>  | 720 |
| 10.185   | <a href="#">Constructor &amp; Destructor Documentation</a>              | 720 |
| 10.185.3 | <a href="#">Ifwd_pixter1d</a>   | 720 |
| 10.185   | <a href="#">Member Function Documentation</a>                           | 720 |
| 10.185.4 | <a href="#">Inext</a>   | 720 |
| 10.186   | <a href="#">Inl::fwd_pixter2d&lt; I &gt; Class Template Reference</a>   | 720 |
| 10.186   | <a href="#">Detailed Description</a>                                    | 721 |
| 10.186   | <a href="#">Member Typedef Documentation</a>                            | 721 |
| 10.186.2 | <a href="#">limage</a>  | 721 |
| 10.186   | <a href="#">Constructor &amp; Destructor Documentation</a>              | 721 |
| 10.186.3 | <a href="#">Ifwd_pixter2d</a>   | 721 |
| 10.186   | <a href="#">Member Function Documentation</a>                           | 722 |
| 10.186.4 | <a href="#">Inext</a>   | 722 |
| 10.187   | <a href="#">Inl::fwd_pixter3d&lt; I &gt; Class Template Reference</a>   | 722 |
| 10.187   | <a href="#">Detailed Description</a>                                    | 722 |
| 10.187   | <a href="#">Member Typedef Documentation</a>                            | 722 |
| 10.187.2 | <a href="#">limage</a>  | 722 |
| 10.187   | <a href="#">Constructor &amp; Destructor Documentation</a>              | 723 |
| 10.187.3 | <a href="#">Ifwd_pixter3d</a>   | 723 |
| 10.187   | <a href="#">Member Function Documentation</a>                           | 723 |
| 10.187.4 | <a href="#">Inext</a>   | 723 |
| 10.188   | <a href="#">Inl::Gdpoint&lt; E &gt; Struct Template Reference</a>       | 723 |
| 10.188   | <a href="#">Detailed Description</a>                                    | 724 |
| 10.189   | <a href="#">Inl::Gdpoint&lt; void &gt; Struct Template Reference</a>    | 724 |
| 10.189   | <a href="#">Detailed Description</a>                                    | 724 |

|  |     |
|--|-----|
| 10.190. <a href="#">In::geom::complex_geometry&lt; D, P &gt; Class Template Reference</a> . . . . .            | 725 |
| 10.190.1. Detailed Description . . . . .   | 725 |
| 10.190.2. Constructor & Destructor Documentation . . . . .   | 725 |
| 10.190.2.1. <a href="#">complex_geometry</a> . . . . .   | 725 |
| 10.190.3. Member Function Documentation . . . . .  | 726 |
| 10.190.3.1. <a href="#">add_location</a> . . . . .   | 726 |
| 10.190.3.2. <a href="#">operator()</a> . . . . .   | 726 |
| 10.191. <a href="#">In::Gpoint&lt; E &gt; Struct Template Reference</a> . . . . .                              | 726 |
| 10.191.1. Detailed Description . . . . .   | 728 |
| 10.191.2. Friends And Related Function Documentation . . . . .   | 728 |
| 10.191.2.1. <a href="#">operator+</a> . . . . .  | 728 |
| 10.191.2.2. <a href="#">operator+=</a> . . . . .   | 729 |
| 10.191.2.3. <a href="#">operator-</a> . . . . .  | 729 |
| 10.191.2.4. <a href="#">operator-=</a> . . . . .   | 730 |
| 10.191.2.5. <a href="#">operator/</a> . . . . .  | 730 |
| 10.191.2.6. <a href="#">operator&lt;&lt;</a> . . . . .   | 730 |
| 10.191.2.7. <a href="#">operator==</a> . . . . .   | 731 |
| 10.192. <a href="#">In::Graph&lt; E &gt; Struct Template Reference</a> . . . . .                               | 731 |
| 10.192.1. Detailed Description . . . . .   | 731 |
| 10.193. <a href="#">In::graph::attribute::card_t Struct Reference</a> . . . . .                                | 732 |
| 10.193.1. Detailed Description . . . . .   | 732 |
| 10.193.2. Member Typedef Documentation . . . . .   | 732 |
| 10.193.2.1. <a href="#">lresult</a> . . . . .  | 732 |
| 10.194. <a href="#">In::graph::attribute::representative_t Struct Reference</a> . . . . .                      | 732 |
| 10.194.1. Detailed Description . . . . .   | 733 |
| 10.194.2. Member Typedef Documentation . . . . .   | 733 |
| 10.194.2.1. <a href="#">lresult</a> . . . . .  | 733 |
| 10.195. <a href="#">In::graph_elt_mixed_neighborhood&lt; G, S, S2 &gt; Struct Template Reference</a> . . . . . | 733 |
| 10.195.1. Detailed Description . . . . .   | 734 |
| 10.195.2. Member Typedef Documentation . . . . .   | 734 |
| 10.195.2.1. <a href="#">lbkd_niter</a> . . . . .   | 734 |
| 10.195.2.2. <a href="#">fwd_niter</a> . . . . .  | 734 |
| 10.195.2.3. <a href="#">niter</a> . . . . .  | 734 |
| 10.196. <a href="#">In::graph_elt_mixed_window&lt; G, S, S2 &gt; Class Template Reference</a> . . . . .        | 734 |
| 10.196.1. Detailed Description . . . . .   | 736 |
| 10.196.2. Member Typedef Documentation . . . . .   | 736 |

|            |   |     |
|------------|---|-----|
| 10.196.2.1 | <code>l_bkd_qiter</code>  | 736 |
| 10.196.2.2 | <code>center_t</code>   | 736 |
| 10.196.2.3 | <code>fwd_qiter</code>  | 737 |
| 10.196.2.4 | <code>graph_element</code>  | 737 |
| 10.196.2.5 | <code>psite</code>  | 737 |
| 10.196.2.6 | <code>qiter</code>  | 737 |
| 10.196.2.7 | <code>site</code>   | 737 |
| 10.196.2.8 | <code>target</code>   | 737 |
| 10.196.3   | Member Function Documentation   | 737 |
| 10.196.3.1 | <code>delta</code>  | 737 |
| 10.196.3.2 | <code>is_centered</code>  | 738 |
| 10.196.3.3 | <code>is_empty</code>   | 738 |
| 10.196.3.4 | <code>is_symmetric</code>   | 738 |
| 10.196.3.5 | <code>is_valid</code>   | 738 |
| 10.196.3.6 | <code>sym</code>  | 738 |
| 10.197     | <code>ln::graph_elt_neighborhood&lt; G, S &gt;</code> Struct Template Reference       | 738 |
| 10.197.1   | Detailed Description  | 739 |
| 10.197.2   | Member Typedef Documentation  | 740 |
| 10.197.2.1 | <code>l_bkd_niter</code>  | 740 |
| 10.197.2.2 | <code>fwd_niter</code>  | 740 |
| 10.197.2.3 | <code>niter</code>  | 740 |
| 10.198     | <code>ln::graph_elt_neighborhood_if&lt; G, S, I &gt;</code> Struct Template Reference | 740 |
| 10.198.1   | Detailed Description  | 741 |
| 10.198.2   | Member Typedef Documentation  | 741 |
| 10.198.2.1 | <code>l_bkd_niter</code>  | 741 |
| 10.198.2.2 | <code>fwd_niter</code>  | 741 |
| 10.198.2.3 | <code>niter</code>  | 742 |
| 10.198.3   | Constructor & Destructor Documentation  | 742 |
| 10.198.3.1 | <code>l_graph_elt_neighborhood_if</code>  | 742 |
| 10.198.3.2 | <code>graph_elt_neighborhood_if</code>  | 742 |
| 10.198.4   | Member Function Documentation   | 742 |
| 10.198.4.1 | <code>l_mask</code>   | 742 |
| 10.199     | <code>ln::graph_elt_window&lt; G, S &gt;</code> Class Template Reference              | 742 |
| 10.199.1   | Detailed Description  | 744 |
| 10.199.2   | Member Typedef Documentation  | 744 |
| 10.199.2.1 | <code>l_bkd_qiter</code>  | 744 |

|   |     |
|---|-----|
| 10.199.2.2center_t . . . . .  | 744 |
| 10.199.2.3fwd_qiter . . . . .   | 745 |
| 10.199.2.4graph_element . . . . .   | 745 |
| 10.199.2.5psite . . . . .   | 745 |
| 10.199.2.6qiter . . . . .   | 745 |
| 10.199.2.7site . . . . .  | 745 |
| 10.199.2.8target . . . . .  | 745 |
| 10.199.3Member Function Documentation . . . . .                             | 745 |
| 10.199.3.1delta . . . . .   | 745 |
| 10.199.3.2is_centered . . . . .   | 746 |
| 10.199.3.3is_empty . . . . .  | 746 |
| 10.199.3.4is_symmetric . . . . .  | 746 |
| 10.199.3.5is_valid . . . . .  | 746 |
| 10.199.3.6sym . . . . .   | 746 |
| 10.200ln::graph_elt_window_if< G, S, I > Class Template Reference . . . . . | 746 |
| 10.200.Detailed Description . . . . .                                       | 748 |
| 10.200.2Member Typedef Documentation . . . . .                              | 748 |
| 10.200.2.1bkd_qiter . . . . .   | 748 |
| 10.200.2.2fwd_qiter . . . . .   | 749 |
| 10.200.2.3mask_t . . . . .  | 749 |
| 10.200.2.4psite . . . . .   | 749 |
| 10.200.2.5qiter . . . . .   | 749 |
| 10.200.2.6site . . . . .  | 749 |
| 10.200.2.7target . . . . .  | 749 |
| 10.200.3Constructor & Destructor Documentation . . . . .                    | 750 |
| 10.200.3.1graph_elt_window_if . . . . .                                     | 750 |
| 10.200.3.2graph_elt_window_if . . . . .                                     | 750 |
| 10.200.4Member Function Documentation . . . . .                             | 750 |
| 10.200.4.1change_mask . . . . .   | 750 |
| 10.200.4.2delta . . . . .   | 750 |
| 10.200.4.3is_centered . . . . .   | 750 |
| 10.200.4.4is_empty . . . . .  | 750 |
| 10.200.4.5is_symmetric . . . . .  | 751 |
| 10.200.4.6is_valid . . . . .  | 751 |
| 10.200.4.7mask . . . . .  | 751 |
| 10.200.4.8sym . . . . .   | 751 |

---

|            |  |     |
|------------|--|-----|
| 10.201     | <code>ln::graph_window_base&lt; P, E &gt;</code> Class Template Reference . . . . .        | 751 |
| 10.201.1   | Detailed Description . . . . .   | 752 |
| 10.201.2   | Member Typedef Documentation . . . . .   | 752 |
| 10.201.2.1 | <code>site</code> . . . . .  | 752 |
| 10.201.3   | Member Function Documentation . . . . .  | 752 |
| 10.201.3.1 | <code>delta</code> . . . . .   | 752 |
| 10.201.3.2 | <code>is_centered</code> . . . . .   | 752 |
| 10.201.3.3 | <code>is_empty</code> . . . . .  | 753 |
| 10.201.3.4 | <code>is_symmetric</code> . . . . .  | 753 |
| 10.201.3.5 | <code>is_valid</code> . . . . .  | 753 |
| 10.201.3.6 | <code>sym</code> . . . . .   | 753 |
| 10.202     | <code>ln::graph_window_if_piter&lt; S, W, I &gt;</code> Class Template Reference . . . . . | 753 |
| 10.202.1   | Detailed Description . . . . .   | 754 |
| 10.202.2   | Member Typedef Documentation . . . . .   | 754 |
| 10.202.2.1 | <code>IP</code> . . . . .  | 754 |
| 10.202.3   | Constructor & Destructor Documentation . . . . .   | 754 |
| 10.202.3.1 | <code>lgraph_window_if_piter</code> . . . . .  | 754 |
| 10.202.4   | Member Function Documentation . . . . .  | 754 |
| 10.202.4.1 | <code>element</code> . . . . .   | 754 |
| 10.202.4.2 | <code>id</code> . . . . .  | 754 |
| 10.202.4.3 | <code>next</code> . . . . .  | 755 |
| 10.203     | <code>ln::graph_window_piter&lt; S, W, I &gt;</code> Class Template Reference . . . . .    | 755 |
| 10.203.1   | Detailed Description . . . . .   | 756 |
| 10.203.2   | Member Typedef Documentation . . . . .   | 756 |
| 10.203.2.1 | <code>lcenter_t</code> . . . . .   | 756 |
| 10.203.2.2 | <code>lgraph_element</code> . . . . .  | 756 |
| 10.203.2.3 | <code>IP</code> . . . . .  | 757 |
| 10.203.3   | Constructor & Destructor Documentation . . . . .   | 757 |
| 10.203.3.1 | <code>lgraph_window_piter</code> . . . . .   | 757 |
| 10.203.3.2 | <code>lgraph_window_piter</code> . . . . .   | 757 |
| 10.203.3.3 | <code>lgraph_window_piter</code> . . . . .   | 757 |
| 10.203.4   | Member Function Documentation . . . . .  | 757 |
| 10.203.4.1 | <code>lchange_target_site_set</code> . . . . .   | 757 |
| 10.203.4.2 | <code>element</code> . . . . .   | 758 |
| 10.203.4.3 | <code>id</code> . . . . .  | 758 |
| 10.203.4.4 | <code>next</code> . . . . .  | 758 |

|   |     |
|---|-----|
| 10.203.4.5target_site_set . . . . .                             | 758 |
| 10.204ln::hexa< I > Struct Template Reference . . . . .         | 758 |
| 10.204.1.Detailed Description . . . . .                         | 760 |
| 10.204.2.Member Typedef Documentation . . . . .                 | 760 |
| 10.204.2.1bkd_piter . . . . .                                   | 760 |
| 10.204.2.2fwd_piter . . . . .                                   | 760 |
| 10.204.2.3value . . . . .                                       | 760 |
| 10.204.2.4psite . . . . .                                       | 761 |
| 10.204.2.5value . . . . .                                       | 761 |
| 10.204.2.6skeleton . . . . .                                    | 761 |
| 10.204.2.7value . . . . .                                       | 761 |
| 10.204.3.Constructor & Destructor Documentation . . . . .       | 761 |
| 10.204.3.1hexa . . . . .  | 761 |
| 10.204.3.2hexa . . . . .  | 761 |
| 10.204.4.Member Function Documentation . . . . .                | 761 |
| 10.204.4.1domain . . . . .                                      | 761 |
| 10.204.4.2has . . . . .   | 761 |
| 10.204.4.3operator() . . . . .                                  | 762 |
| 10.204.4.4operator() . . . . .                                  | 762 |
| 10.205ln::histo::array< T > Struct Template Reference . . . . . | 762 |
| 10.205.1.Detailed Description . . . . .                         | 762 |
| 10.206ln::Image< E > Struct Template Reference . . . . .        | 762 |
| 10.206.1.Detailed Description . . . . .                         | 764 |
| 10.207ln::image1d< T > Struct Template Reference . . . . .      | 764 |
| 10.207.1.Detailed Description . . . . .                         | 765 |
| 10.207.2.Member Typedef Documentation . . . . .                 | 766 |
| 10.207.2.1lvalue . . . . .                                      | 766 |
| 10.207.2.2rvalue . . . . .                                      | 766 |
| 10.207.2.3skeleton . . . . .                                    | 766 |
| 10.207.2.4value . . . . .                                       | 766 |
| 10.207.3.Constructor & Destructor Documentation . . . . .       | 766 |
| 10.207.3.1image1d . . . . .                                     | 766 |
| 10.207.3.2image1d . . . . .                                     | 766 |
| 10.207.3.3image1d . . . . .                                     | 766 |
| 10.207.4.Member Function Documentation . . . . .                | 767 |
| 10.207.4.1bbox . . . . .  | 767 |

|   |     |
|---|-----|
| 10.207.4.2border                                | 767 |
| 10.207.4.3buffer                                | 767 |
| 10.207.4.4buffer                                | 767 |
| 10.207.4.5delta_index                           | 767 |
| 10.207.4.6domain                                | 767 |
| 10.207.4.7element                               | 767 |
| 10.207.4.8element                               | 768 |
| 10.207.4.9has                                   | 768 |
| 10.207.4.10elements                             | 768 |
| 10.207.4.11finds                                | 768 |
| 10.207.4.12operator()                           | 768 |
| 10.207.4.13operator()                           | 768 |
| 10.207.4.14point_at_index                       | 768 |
| 10.207.4.15bbox                                 | 769 |
| 10.208In::image2d< T > Class Template Reference | 769 |
| 10.208.1Detailed Description                    | 770 |
| 10.208.2Member Typedef Documentation            | 771 |
| 10.208.2.1lvalue                                | 771 |
| 10.208.2.2rvalue                                | 771 |
| 10.208.2.3skeleton                              | 771 |
| 10.208.2.4value                                 | 771 |
| 10.208.3Constructor & Destructor Documentation  | 771 |
| 10.208.3.1image2d                               | 771 |
| 10.208.3.2image2d                               | 771 |
| 10.208.3.3image2d                               | 771 |
| 10.208.4Member Function Documentation           | 772 |
| 10.208.4.1bbox                                  | 772 |
| 10.208.4.2border                                | 772 |
| 10.208.4.3buffer                                | 772 |
| 10.208.4.4buffer                                | 772 |
| 10.208.4.5delta_index                           | 772 |
| 10.208.4.6domain                                | 772 |
| 10.208.4.7element                               | 772 |
| 10.208.4.8element                               | 773 |
| 10.208.4.9has                                   | 773 |
| 10.208.4.10cols                                 | 773 |

|  |     |
|--|-----|
| 10.208.4.1elements                                   | 773 |
| 10.208.4.2rows                                       | 773 |
| 10.208.4.3operator()                                 | 773 |
| 10.208.4.4operator()                                 | 773 |
| 10.208.4.5point_at_index                             | 774 |
| 10.209.1In::image2d_h< V > Struct Template Reference | 774 |
| 10.209.1Detailed Description                         | 775 |
| 10.209.2Member Typedef Documentation                 | 775 |
| 10.209.2.1bkd_piter                                  | 775 |
| 10.209.2.2fwd_piter                                  | 775 |
| 10.209.2.3value                                      | 776 |
| 10.209.2.4psite                                      | 776 |
| 10.209.2.5value                                      | 776 |
| 10.209.2.6skeleton                                   | 776 |
| 10.209.2.7value                                      | 776 |
| 10.209.3Constructor & Destructor Documentation       | 776 |
| 10.209.3.1image2d_h                                  | 776 |
| 10.209.4Member Function Documentation                | 776 |
| 10.209.4.1domain                                     | 776 |
| 10.209.4.2has  | 777 |
| 10.209.4.3operator()                                 | 777 |
| 10.209.4.4operator()                                 | 777 |
| 10.210.1In::image3d< T > Struct Template Reference   | 777 |
| 10.210.1Detailed Description                         | 779 |
| 10.210.2Member Typedef Documentation                 | 779 |
| 10.210.2.1lvalue                                     | 779 |
| 10.210.2.2rvalue                                     | 779 |
| 10.210.2.3skeleton                                   | 779 |
| 10.210.2.4value                                      | 779 |
| 10.210.3Constructor & Destructor Documentation       | 780 |
| 10.210.3.1image3d                                    | 780 |
| 10.210.3.2image3d                                    | 780 |
| 10.210.3.3image3d                                    | 780 |
| 10.210.4Member Function Documentation                | 780 |
| 10.210.4.1bbox                                       | 780 |
| 10.210.4.2border                                     | 780 |



|             |  |     |
|-------------|--|-----|
| 10.210.4.3  | buffer   | 780 |
| 10.210.4.4  | buffer   | 780 |
| 10.210.4.5  | delta_index  | 781 |
| 10.210.4.6  | domain   | 781 |
| 10.210.4.7  | element  | 781 |
| 10.210.4.8  | element  | 781 |
| 10.210.4.9  | has  | 781 |
| 10.210.4.10 | cols   | 781 |
| 10.210.4.11 | elements   | 781 |
| 10.210.4.12 | rows   | 782 |
| 10.210.4.13 | slis   | 782 |
| 10.210.4.14 | operator()   | 782 |
| 10.210.4.15 | operator()   | 782 |
| 10.210.4.16 | point_at_index                                     | 782 |
| 10.210.4.17 | bbox   | 782 |
| 10.211      | In::image_if< I, F > Struct Template Reference     | 782 |
| 10.211.1    | Detailed Description                               | 783 |
| 10.211.2    | Member Typedef Documentation                       | 783 |
| 10.211.2.1  | Iskeleton  | 783 |
| 10.211.3    | Constructor & Destructor Documentation             | 783 |
| 10.211.3.1  | image_if   | 783 |
| 10.211.3.2  | image_if   | 784 |
| 10.211.4    | Member Function Documentation                      | 784 |
| 10.211.4.1  | domain   | 784 |
| 10.211.4.2  | operator image_if< const I, F >                    | 784 |
| 10.212      | In::interpolated< I, F > Struct Template Reference | 784 |
| 10.212.1    | Detailed Description                               | 785 |
| 10.212.2    | Member Typedef Documentation                       | 785 |
| 10.212.2.1  | lvalue   | 785 |
| 10.212.2.2  | psite  | 785 |
| 10.212.2.3  | rvalue   | 785 |
| 10.212.2.4  | skeleton   | 785 |
| 10.212.2.5  | value  | 786 |
| 10.212.3    | Constructor & Destructor Documentation             | 786 |
| 10.212.3.1  | interpolated                                       | 786 |
| 10.212.4    | Member Function Documentation                      | 786 |

|   |     |
|---|-----|
| 10.212.4.1has   | 786 |
| 10.212.4.2is_valid  | 786 |
| 10.213.1nl::io::dicom::dicom_header Struct Reference            | 786 |
| 10.213.1.Detailed Description                                   | 786 |
| 10.214.1nl::io::dump::dump_header Struct Reference              | 786 |
| 10.214.1.Detailed Description                                   | 787 |
| 10.215.1nl::io::fld::fld_header Struct Reference                | 787 |
| 10.215.1.Detailed Description                                   | 787 |
| 10.216.1nl::io::raw::raw_header Struct Reference                | 787 |
| 10.216.1.Detailed Description                                   | 787 |
| 10.217.1nl::Iterator< E > Struct Template Reference             | 787 |
| 10.217.1.Detailed Description                                   | 788 |
| 10.217.2.Member Function Documentation                          | 789 |
| 10.217.2.1next  | 789 |
| 10.218.1nl::labeled_image< I > Class Template Reference         | 789 |
| 10.218.1.Detailed Description                                   | 791 |
| 10.218.2.Member Typedef Documentation                           | 791 |
| 10.218.2.1bbox_t  | 791 |
| 10.218.2.2skeleton  | 791 |
| 10.218.3.Constructor & Destructor Documentation                 | 791 |
| 10.218.3.1labeled_image   | 791 |
| 10.218.3.2labeled_image   | 791 |
| 10.218.3.3labeled_image   | 791 |
| 10.218.4.Member Function Documentation                          | 792 |
| 10.218.4.1bbox  | 792 |
| 10.218.4.2bboxes  | 792 |
| 10.218.4.3labels  | 792 |
| 10.218.4.4relabel   | 792 |
| 10.218.4.5relabel   | 792 |
| 10.218.4.6subdomain   | 792 |
| 10.218.4.7update_data   | 792 |
| 10.219.1nl::labeled_image_base< I, E > Class Template Reference | 793 |
| 10.219.1.Detailed Description                                   | 794 |
| 10.219.2.Member Typedef Documentation                           | 794 |
| 10.219.2.1bbox_t  | 794 |
| 10.219.3.Constructor & Destructor Documentation                 | 794 |

|   |     |
|---|-----|
| 10.219.3. <code>llabeled_image_base</code> . . . . .                                      | 794 |
| 10.219.4. Member Function Documentation . . . . .   | 795 |
| 10.219.4.1. <code>lbbox</code> . . . . .  | 795 |
| 10.219.4.2. <code>lbboxes</code> . . . . .  | 795 |
| 10.219.4.3. <code>llabels</code> . . . . .  | 795 |
| 10.219.4.4. <code>lrelabel</code> . . . . .   | 795 |
| 10.219.4.5. <code>lrelabel</code> . . . . .   | 795 |
| 10.219.4.6. <code>lsubdomain</code> . . . . .   | 796 |
| 10.219.4.7. <code>lupdate_data</code> . . . . .   | 796 |
| 10.220. <code>mln::lazy_image&lt; I, F, B &gt;</code> Struct Template Reference . . . . . | 796 |
| 10.220.1. Detailed Description . . . . .  | 797 |
| 10.220.2. Member Typedef Documentation . . . . .  | 797 |
| 10.220.2.1. <code>lvalue</code> . . . . .   | 797 |
| 10.220.2.2. <code>lvalue</code> . . . . .   | 797 |
| 10.220.2.3. <code>lskeleton</code> . . . . .  | 797 |
| 10.220.3. Constructor & Destructor Documentation . . . . .                                | 798 |
| 10.220.3.1. <code>llazy_image</code> . . . . .  | 798 |
| 10.220.3.2. <code>llazy_image</code> . . . . .  | 798 |
| 10.220.4. Member Function Documentation . . . . .   | 798 |
| 10.220.4.1. <code>ldomain</code> . . . . .  | 798 |
| 10.220.4.2. <code>lhas</code> . . . . .   | 798 |
| 10.220.4.3. <code>operator()</code> . . . . .   | 798 |
| 10.220.4.4. <code>operator()</code> . . . . .   | 798 |
| 10.220.4.5. <code>operator()</code> . . . . .   | 798 |
| 10.220.4.6. <code>operator()</code> . . . . .   | 799 |
| 10.221. <code>mln::Literal&lt; E &gt;</code> Struct Template Reference . . . . .          | 799 |
| 10.221.1. Detailed Description . . . . .  | 801 |
| 10.222. <code>mln::literal::black_t</code> Struct Reference . . . . .                     | 801 |
| 10.222.1. Detailed Description . . . . .  | 801 |
| 10.223. <code>mln::literal::blue_t</code> Struct Reference . . . . .                      | 802 |
| 10.223.1. Detailed Description . . . . .  | 802 |
| 10.224. <code>mln::literal::brown_t</code> Struct Reference . . . . .                     | 802 |
| 10.224.1. Detailed Description . . . . .  | 803 |
| 10.225. <code>mln::literal::cyan_t</code> Struct Reference . . . . .                      | 803 |
| 10.225.1. Detailed Description . . . . .  | 804 |
| 10.226. <code>mln::literal::green_t</code> Struct Reference . . . . .                     | 804 |

|  |     |
|--|-----|
| 10.226. Detailed Description . . . . .                                       | 805 |
| 10.227. <a href="#">ln::literal::identity_t Struct Reference</a> . . . . .   | 805 |
| 10.227. Detailed Description . . . . .                                       | 806 |
| 10.228. <a href="#">ln::literal::light_gray_t Struct Reference</a> . . . . . | 806 |
| 10.228. Detailed Description . . . . .                                       | 807 |
| 10.229. <a href="#">ln::literal::lime_t Struct Reference</a> . . . . .       | 807 |
| 10.229. Detailed Description . . . . .                                       | 808 |
| 10.230. <a href="#">ln::literal::magenta_t Struct Reference</a> . . . . .    | 808 |
| 10.230. Detailed Description . . . . .                                       | 809 |
| 10.231. <a href="#">ln::literal::max_t Struct Reference</a> . . . . .        | 809 |
| 10.231. Detailed Description . . . . .                                       | 810 |
| 10.232. <a href="#">ln::literal::min_t Struct Reference</a> . . . . .        | 810 |
| 10.232. Detailed Description . . . . .                                       | 811 |
| 10.233. <a href="#">ln::literal::olive_t Struct Reference</a> . . . . .      | 811 |
| 10.233. Detailed Description . . . . .                                       | 812 |
| 10.234. <a href="#">ln::literal::one_t Struct Reference</a> . . . . .        | 812 |
| 10.234. Detailed Description . . . . .                                       | 813 |
| 10.235. <a href="#">ln::literal::orange_t Struct Reference</a> . . . . .     | 813 |
| 10.235. Detailed Description . . . . .                                       | 814 |
| 10.236. <a href="#">ln::literal::origin_t Struct Reference</a> . . . . .     | 814 |
| 10.236. Detailed Description . . . . .                                       | 815 |
| 10.237. <a href="#">ln::literal::pink_t Struct Reference</a> . . . . .       | 815 |
| 10.237. Detailed Description . . . . .                                       | 816 |
| 10.238. <a href="#">ln::literal::purple_t Struct Reference</a> . . . . .     | 816 |
| 10.238. Detailed Description . . . . .                                       | 817 |
| 10.239. <a href="#">ln::literal::red_t Struct Reference</a> . . . . .        | 817 |
| 10.239. Detailed Description . . . . .                                       | 818 |
| 10.240. <a href="#">ln::literal::teal_t Struct Reference</a> . . . . .       | 818 |
| 10.240. Detailed Description . . . . .                                       | 819 |
| 10.241. <a href="#">ln::literal::violet_t Struct Reference</a> . . . . .     | 819 |
| 10.241. Detailed Description . . . . .                                       | 820 |
| 10.242. <a href="#">ln::literal::white_t Struct Reference</a> . . . . .      | 820 |
| 10.242. Detailed Description . . . . .                                       | 821 |
| 10.243. <a href="#">ln::literal::yellow_t Struct Reference</a> . . . . .     | 821 |
| 10.243. Detailed Description . . . . .                                       | 822 |
| 10.244. <a href="#">ln::literal::zero_t Struct Reference</a> . . . . .       | 822 |

|   |     |
|---|-----|
| 10.244. Detailed Description  | 823 |
| 10.245. <code>std::Mesh&lt; E &gt;</code> Struct Template Reference                                     | 823 |
| 10.245. Detailed Description  | 824 |
| 10.246. <code>std::Meta_Accumulator&lt; E &gt;</code> Struct Template Reference                         | 824 |
| 10.246. Detailed Description  | 825 |
| 10.247. <code>std::Meta_Function&lt; E &gt;</code> Struct Template Reference                            | 826 |
| 10.247. Detailed Description  | 826 |
| 10.248. <code>std::Meta_Function_v2v&lt; E &gt;</code> Struct Template Reference                        | 827 |
| 10.248. Detailed Description  | 827 |
| 10.249. <code>std::Meta_Function_vv2v&lt; E &gt;</code> Struct Template Reference                       | 827 |
| 10.249. Detailed Description  | 828 |
| 10.250. <code>std::metal::ands&lt; E1, E2, E3, E4, E5, E6, E7, E8 &gt;</code> Struct Template Reference | 828 |
| 10.250. Detailed Description  | 828 |
| 10.251. <code>std::metal::converts_to&lt; T, U &gt;</code> Struct Template Reference                    | 828 |
| 10.251. Detailed Description  | 828 |
| 10.252. <code>std::metal::equal&lt; T1, T2 &gt;</code> Struct Template Reference                        | 828 |
| 10.252. Detailed Description  | 829 |
| 10.253. <code>std::metal::goes_to&lt; T, U &gt;</code> Struct Template Reference                        | 829 |
| 10.253. Detailed Description  | 829 |
| 10.254. <code>std::metal::is&lt; T, U &gt;</code> Struct Template Reference                             | 829 |
| 10.254. Detailed Description  | 829 |
| 10.255. <code>std::metal::is_a&lt; T, M &gt;</code> Struct Template Reference                           | 830 |
| 10.255. Detailed Description  | 830 |
| 10.256. <code>std::metal::is_not&lt; T, U &gt;</code> Struct Template Reference                         | 830 |
| 10.256. Detailed Description  | 830 |
| 10.257. <code>std::metal::is_not_a&lt; T, M &gt;</code> Struct Template Reference                       | 830 |
| 10.257. Detailed Description  | 830 |
| 10.258. <code>std::mixed_neighb&lt; W &gt;</code> Class Template Reference                              | 830 |
| 10.258. Detailed Description  | 831 |
| 10.258. Member Typedef Documentation  | 831 |
| 10.258.2. <code>l_bkd_niter</code>  | 831 |
| 10.258.2. <code>fwd_niter</code>  | 831 |
| 10.258.2. <code>niter</code>  | 831 |
| 10.258. Constructor & Destructor Documentation  | 832 |
| 10.258.3. <code>lmixed_neighb</code>  | 832 |
| 10.258.3. <code>mixed_neighb</code>   | 832 |

|          |  |     |
|----------|--|-----|
| 10.259   | <code>ln::morpho::attribute::card&lt; I &gt; Class Template Reference</code>                     | 832 |
| 10.259   | Detailed Description   | 832 |
| 10.259   | Member Function Documentation  | 833 |
| 10.259.2 | <code>limit</code>   | 833 |
| 10.259.2 | <code>is_valid</code>  | 833 |
| 10.259.2 | <code>take_as_init</code>  | 833 |
| 10.259.2 | <code>take_n_times</code>  | 833 |
| 10.259.2 | <code>to_result</code>   | 833 |
| 10.260   | <code>ln::morpho::attribute::count_adjacent_vertices&lt; I &gt; Struct Template Reference</code> | 833 |
| 10.260   | Detailed Description   | 834 |
| 10.260   | Member Function Documentation  | 834 |
| 10.260.2 | <code>limit</code>   | 834 |
| 10.260.2 | <code>is_valid</code>  | 834 |
| 10.260.2 | <code>take_as_init</code>  | 834 |
| 10.260.2 | <code>take_n_times</code>  | 834 |
| 10.260.2 | <code>to_result</code>   | 835 |
| 10.261   | <code>ln::morpho::attribute::height&lt; I &gt; Struct Template Reference</code>                  | 835 |
| 10.261   | Detailed Description   | 835 |
| 10.261   | Member Function Documentation  | 836 |
| 10.261.2 | <code>lbase_level</code>   | 836 |
| 10.261.2 | <code>init</code>  | 836 |
| 10.261.2 | <code>is_valid</code>  | 836 |
| 10.261.2 | <code>take_as_init</code>  | 836 |
| 10.261.2 | <code>take_n_times</code>  | 836 |
| 10.261.2 | <code>to_result</code>   | 836 |
| 10.262   | <code>ln::morpho::attribute::sharpness&lt; I &gt; Struct Template Reference</code>               | 836 |
| 10.262   | Detailed Description   | 837 |
| 10.262   | Member Function Documentation  | 837 |
| 10.262.2 | <code>larea</code>   | 837 |
| 10.262.2 | <code>height</code>  | 838 |
| 10.262.2 | <code>init</code>  | 838 |
| 10.262.2 | <code>is_valid</code>  | 838 |
| 10.262.2 | <code>take_as_init</code>  | 838 |
| 10.262.2 | <code>take_n_times</code>  | 838 |
| 10.262.2 | <code>to_result</code>   | 838 |
| 10.262.2 | <code>volume</code>  | 838 |

|           |  |     |
|-----------|--|-----|
| 10.263    | <code>mln::morpho::attribute::sum&lt; I, S &gt;</code> Class Template Reference . . . . .  | 839 |
| 10.263.   | Detailed Description . . . . .   | 839 |
| 10.263.   | Member Function Documentation . . . . .  | 839 |
| 10.263.2. | <code>limit</code> . . . . .   | 839 |
| 10.263.2. | <code>is_valid</code> . . . . .  | 840 |
| 10.263.2. | <code>set_value</code> . . . . .   | 840 |
| 10.263.2. | <code>take_as_init</code> . . . . .  | 840 |
| 10.263.2. | <code>take_n_times</code> . . . . .  | 840 |
| 10.263.2. | <code>to_result</code> . . . . .   | 840 |
| 10.263.2. | <code>untake</code> . . . . .  | 840 |
| 10.264    | <code>mln::morpho::attribute::volume&lt; I &gt;</code> Struct Template Reference . . . . . | 840 |
| 10.264.   | Detailed Description . . . . .   | 841 |
| 10.264.   | Member Function Documentation . . . . .  | 841 |
| 10.264.2. | <code>larea</code> . . . . .   | 841 |
| 10.264.2. | <code>init</code> . . . . .  | 841 |
| 10.264.2. | <code>is_valid</code> . . . . .  | 841 |
| 10.264.2. | <code>take_as_init</code> . . . . .  | 842 |
| 10.264.2. | <code>take_n_times</code> . . . . .  | 842 |
| 10.264.2. | <code>to_result</code> . . . . .   | 842 |
| 10.265    | <code>mln::neighb&lt; W &gt;</code> Class Template Reference . . . . .                     | 842 |
| 10.265.   | Detailed Description . . . . .   | 843 |
| 10.265.   | Member Typedef Documentation . . . . .   | 843 |
| 10.265.2. | <code>lbkd_niter</code> . . . . .  | 843 |
| 10.265.2. | <code>2fwd_niter</code> . . . . .  | 843 |
| 10.265.2. | <code>3niter</code> . . . . .  | 843 |
| 10.265.   | Constructor & Destructor Documentation . . . . .   | 843 |
| 10.265.3. | <code>lneighb</code> . . . . .   | 843 |
| 10.265.3. | <code>2neighb</code> . . . . .   | 843 |
| 10.266    | <code>mln::Neighborhood&lt; E &gt;</code> Struct Template Reference . . . . .              | 844 |
| 10.266.   | Detailed Description . . . . .   | 844 |
| 10.267    | <code>mln::Neighborhood&lt; void &gt;</code> Struct Template Reference . . . . .           | 844 |
| 10.267.   | Detailed Description . . . . .   | 844 |
| 10.268    | <code>mln::Object&lt; E &gt;</code> Struct Template Reference . . . . .                    | 844 |
| 10.268.   | Detailed Description . . . . .   | 845 |
| 10.269    | <code>mln::p2p_image&lt; I, F &gt;</code> Struct Template Reference . . . . .              | 845 |
| 10.269.   | Detailed Description . . . . .   | 846 |

|             |  |     |
|-------------|--|-----|
| 10.269.2    | Member Typedef Documentation               | 846 |
| 10.269.2.1  | lskeleton                                  | 846 |
| 10.269.3    | Constructor & Destructor Documentation     | 846 |
| 10.269.3.1  | lp2p_image                                 | 846 |
| 10.269.3.2  | p2p_image                                  | 846 |
| 10.269.4    | Member Function Documentation              | 846 |
| 10.269.4.1  | ldomain                                    | 846 |
| 10.269.4.2  | fun  | 846 |
| 10.269.4.3  | operator()                                 | 847 |
| 10.269.4.4  | operator()                                 | 847 |
| 10.270      | std::p_array< P > Class Template Reference | 847 |
| 10.270.1    | Detailed Description                       | 849 |
| 10.270.2    | Member Typedef Documentation               | 849 |
| 10.270.2.1  | l_bkd_piter                                | 849 |
| 10.270.2.2  | element                                    | 849 |
| 10.270.2.3  | l_fwd_piter                                | 849 |
| 10.270.2.4  | l_element                                  | 849 |
| 10.270.2.5  | l_piter                                    | 849 |
| 10.270.2.6  | l_site                                     | 849 |
| 10.270.3    | Constructor & Destructor Documentation     | 850 |
| 10.270.3.1  | lp_array                                   | 850 |
| 10.270.3.2  | p_array                                    | 850 |
| 10.270.4    | Member Function Documentation              | 850 |
| 10.270.4.1  | lappend                                    | 850 |
| 10.270.4.2  | append                                     | 850 |
| 10.270.4.3  | change                                     | 850 |
| 10.270.4.4  | clear                                      | 850 |
| 10.270.4.5  | has  | 850 |
| 10.270.4.6  | has  | 851 |
| 10.270.4.7  | insert                                     | 851 |
| 10.270.4.8  | is_valid                                   | 851 |
| 10.270.4.9  | memory_size                                | 851 |
| 10.270.4.10 | l_sites                                    | 851 |
| 10.270.4.11 | loperator[]                                | 851 |
| 10.270.4.12 | operator[]                                 | 852 |
| 10.270.4.13 | operator[]                                 | 852 |



|  |     |
|--|-----|
| 10.270.4.14reserve                                   | 852 |
| 10.270.4.15size                                      | 852 |
| 10.270.4.16d_vector                                  | 852 |
| 10.271ln::p_centered< W > Class Template Reference   | 852 |
| 10.271.1Detailed Description                         | 853 |
| 10.271.2Member Typedef Documentation                 | 854 |
| 10.271.2.1bkd_piter                                  | 854 |
| 10.271.2.2element                                    | 854 |
| 10.271.2.3fwd_piter                                  | 854 |
| 10.271.2.4piter                                      | 854 |
| 10.271.2.5psite                                      | 854 |
| 10.271.2.6site                                       | 854 |
| 10.271.3Constructor & Destructor Documentation       | 854 |
| 10.271.3.1p_centered                                 | 854 |
| 10.271.3.2p_centered                                 | 854 |
| 10.271.4Member Function Documentation                | 855 |
| 10.271.4.1center                                     | 855 |
| 10.271.4.2has  | 855 |
| 10.271.4.3is_valid                                   | 855 |
| 10.271.4.4memory_size                                | 855 |
| 10.271.4.5window                                     | 855 |
| 10.272ln::p_complex< D, G > Class Template Reference | 855 |
| 10.272.1Detailed Description                         | 857 |
| 10.272.2Member Typedef Documentation                 | 857 |
| 10.272.2.1bkd_piter                                  | 857 |
| 10.272.2.2element                                    | 857 |
| 10.272.2.3fwd_piter                                  | 857 |
| 10.272.2.4piter                                      | 857 |
| 10.272.2.5psite                                      | 857 |
| 10.272.3Constructor & Destructor Documentation       | 858 |
| 10.272.3.1p_complex                                  | 858 |
| 10.272.4Member Function Documentation                | 858 |
| 10.272.4.1cplx                                       | 858 |
| 10.272.4.2cplx                                       | 858 |
| 10.272.4.3geom                                       | 858 |
| 10.272.4.4has  | 858 |

|            |   |     |
|------------|---|-----|
| 10.272.4.5 | is_valid  | 859 |
| 10.272.4.6 | faces   | 859 |
| 10.272.4.7 | faces_of_dim                                      | 859 |
| 10.272.4.8 | sites   | 859 |
| 10.273     | mln::p_edges< G, F > Class Template Reference     | 859 |
| 10.273.1   | Detailed Description                              | 861 |
| 10.273.2   | Member Typedef Documentation                      | 861 |
| 10.273.2.1 | bkd_piter   | 861 |
| 10.273.2.2 | edge  | 861 |
| 10.273.2.3 | element   | 861 |
| 10.273.2.4 | fun_t   | 862 |
| 10.273.2.5 | wd_piter  | 862 |
| 10.273.2.6 | graph_element                                     | 862 |
| 10.273.2.7 | graph_t   | 862 |
| 10.273.2.8 | piter   | 862 |
| 10.273.2.9 | psite   | 862 |
| 10.273.3   | Constructor & Destructor Documentation            | 863 |
| 10.273.3.1 | p_edges   | 863 |
| 10.273.3.2 | p_edges   | 863 |
| 10.273.3.3 | p_edges   | 863 |
| 10.273.3.4 | p_edges   | 863 |
| 10.273.4   | Member Function Documentation                     | 864 |
| 10.273.4.1 | function  | 864 |
| 10.273.4.2 | graph   | 864 |
| 10.273.4.3 | has   | 864 |
| 10.273.4.4 | has   | 864 |
| 10.273.4.5 | invalidate  | 864 |
| 10.273.4.6 | is_valid  | 864 |
| 10.273.4.7 | memory_size                                       | 865 |
| 10.273.4.8 | nedges  | 865 |
| 10.273.4.9 | sites   | 865 |
| 10.274     | mln::p_faces< N, D, P > Struct Template Reference | 865 |
| 10.274.1   | Detailed Description                              | 866 |
| 10.274.2   | Member Typedef Documentation                      | 866 |
| 10.274.2.1 | bkd_piter   | 866 |
| 10.274.2.2 | element   | 866 |

|            |  |     |
|------------|--|-----|
| 10.274.2.3 | <code>3fwd_piter</code>  | 867 |
| 10.274.2.4 | <code>4piter</code>  | 867 |
| 10.274.2.5 | <code>5psite</code>  | 867 |
| 10.274.3   | Constructor & Destructor Documentation                                 | 867 |
| 10.274.3.1 | <code>1p_faces</code>  | 867 |
| 10.274.3.2 | <code>2p_faces</code>  | 867 |
| 10.274.4   | Member Function Documentation  | 867 |
| 10.274.4.1 | <code>1cplx</code>   | 867 |
| 10.274.4.2 | <code>2cplx</code>   | 868 |
| 10.274.4.3 | <code>3is_valid</code>   | 868 |
| 10.274.4.4 | <code>4nfaces</code>   | 868 |
| 10.274.4.5 | <code>5nsites</code>   | 868 |
| 10.275     | <code>mln::p_graph_piter&lt; S, I &gt;</code> Class Template Reference | 868 |
| 10.275.1   | Detailed Description   | 869 |
| 10.275.2   | Constructor & Destructor Documentation                                 | 869 |
| 10.275.2.1 | <code>1p_graph_piter</code>  | 869 |
| 10.275.3   | Member Function Documentation  | 869 |
| 10.275.3.1 | <code>1graph</code>  | 869 |
| 10.275.3.2 | <code>2id</code>   | 869 |
| 10.275.3.3 | <code>3mln_q_subject</code>  | 870 |
| 10.275.3.4 | <code>4next</code>   | 870 |
| 10.276     | <code>mln::p_if&lt; S, F &gt;</code> Class Template Reference          | 870 |
| 10.276.1   | Detailed Description   | 871 |
| 10.276.2   | Member Typedef Documentation   | 871 |
| 10.276.2.1 | <code>1bkd_piter</code>  | 871 |
| 10.276.2.2 | <code>2element</code>  | 871 |
| 10.276.2.3 | <code>3fwd_piter</code>  | 871 |
| 10.276.2.4 | <code>4piter</code>  | 872 |
| 10.276.2.5 | <code>5psite</code>  | 872 |
| 10.276.3   | Constructor & Destructor Documentation                                 | 872 |
| 10.276.3.1 | <code>1p_if</code>   | 872 |
| 10.276.3.2 | <code>2p_if</code>   | 872 |
| 10.276.4   | Member Function Documentation  | 872 |
| 10.276.4.1 | <code>1has</code>  | 872 |
| 10.276.4.2 | <code>2is_valid</code>   | 872 |
| 10.276.4.3 | <code>3memory_size</code>  | 872 |

|  |     |
|--|-----|
| 10.276.4.4overset . . . . .  | 873 |
| 10.276.4.5pred . . . . .   | 873 |
| 10.276.4.6predicate . . . . .  | 873 |
| 10.277\nln::p_image< I > Class Template Reference . . . . .                | 873 |
| 10.277.1.Detailed Description . . . . .                                    | 874 |
| 10.277.2.Member Typedef Documentation . . . . .                            | 875 |
| 10.277.2.1bkd_piter . . . . .  | 875 |
| 10.277.2.2element . . . . .  | 875 |
| 10.277.2.3fwd_piter . . . . .  | 875 |
| 10.277.2.4i_element . . . . .  | 875 |
| 10.277.2.5piter . . . . .  | 875 |
| 10.277.2.6psite . . . . .  | 875 |
| 10.277.2.7r_element . . . . .  | 875 |
| 10.277.2.8S . . . . .  | 875 |
| 10.277.3.Constructor & Destructor Documentation . . . . .                  | 876 |
| 10.277.3.1p_image . . . . .  | 876 |
| 10.277.3.2p_image . . . . .  | 876 |
| 10.277.4.Member Function Documentation . . . . .                           | 876 |
| 10.277.4.1clear . . . . .  | 876 |
| 10.277.4.2has . . . . .  | 876 |
| 10.277.4.3insert . . . . .   | 876 |
| 10.277.4.4is_valid . . . . .   | 876 |
| 10.277.4.5memory_size . . . . .  | 877 |
| 10.277.4.6nsites . . . . .   | 877 |
| 10.277.4.7operator typename internal::p_image_site_set< I >::ret . . . . . | 877 |
| 10.277.4.8remove . . . . .   | 877 |
| 10.277.4.9toggle . . . . .   | 877 |
| 10.278\nln::p_indexed_bkd_piter< S > Class Template Reference . . . . .    | 877 |
| 10.278.1.Detailed Description . . . . .                                    | 878 |
| 10.278.2.Constructor & Destructor Documentation . . . . .                  | 878 |
| 10.278.2.1p_indexed_bkd_piter . . . . .                                    | 878 |
| 10.278.2.2p_indexed_bkd_piter . . . . .                                    | 878 |
| 10.278.3.Member Function Documentation . . . . .                           | 878 |
| 10.278.3.1index . . . . .  | 878 |
| 10.278.3.2next . . . . .   | 878 |
| 10.279\nln::p_indexed_fwd_piter< S > Class Template Reference . . . . .    | 879 |

---

|   |     |
|---|-----|
| 10.279. Detailed Description  | 879 |
| 10.279. Constructor & Destructor Documentation                                | 879 |
| 10.279.2. <code>lp_indexed_fwd_piter</code>                                   | 879 |
| 10.279.2. <code>rp_indexed_fwd_piter</code>                                   | 879 |
| 10.279. Member Function Documentation   | 880 |
| 10.279.3. <code>lindex</code>   | 880 |
| 10.279.3. <code>2next</code>  | 880 |
| 10.280. <code>mln::p_indexed_psite&lt; S &gt;</code> Class Template Reference | 880 |
| 10.280. Detailed Description  | 880 |
| 10.281. <code>mln::p_key&lt; K, P &gt;</code> Class Template Reference        | 880 |
| 10.281. Detailed Description  | 882 |
| 10.281. Member Typedef Documentation  | 882 |
| 10.281.2. <code>l_bkd_piter</code>  | 882 |
| 10.281.2. <code>2element</code>   | 882 |
| 10.281.2. <code>3fwd_piter</code>   | 883 |
| 10.281.2. <code>4_element</code>  | 883 |
| 10.281.2. <code>5piter</code>   | 883 |
| 10.281.2. <code>6psite</code>   | 883 |
| 10.281.2. <code>7_element</code>  | 883 |
| 10.281. Constructor & Destructor Documentation                                | 883 |
| 10.281.3. <code>lp_key</code>   | 883 |
| 10.281. Member Function Documentation   | 883 |
| 10.281.4. <code>lchange_key</code>  | 883 |
| 10.281.4. <code>2change_keys</code>   | 884 |
| 10.281.4. <code>3clear</code>   | 884 |
| 10.281.4. <code>4exists_key</code>  | 884 |
| 10.281.4. <code>5has</code>   | 884 |
| 10.281.4. <code>6has</code>   | 884 |
| 10.281.4. <code>7insert</code>  | 884 |
| 10.281.4. <code>8insert</code>  | 884 |
| 10.281.4. <code>9s_valid</code>   | 885 |
| 10.281.4. <code>lkey</code>   | 885 |
| 10.281.4. <code>lkeys</code>  | 885 |
| 10.281.4. <code>lmemory_size</code>   | 885 |
| 10.281.4. <code>lpsites</code>  | 885 |
| 10.281.4. <code>loperator()</code>  | 885 |

|             |  |     |
|-------------|--|-----|
| 10.281.4.15 | move   | 885 |
| 10.281.4.16 | move_key   | 886 |
| 10.282      | nl::p_line2d Class Reference                         | 886 |
| 10.282.1    | Detailed Description                                 | 887 |
| 10.282.2    | Member Typedef Documentation                         | 887 |
| 10.282.2.1  | l_bkd_piter  | 887 |
| 10.282.2.2  | element  | 887 |
| 10.282.2.3  | l_fwd_piter  | 887 |
| 10.282.2.4  | piter  | 888 |
| 10.282.2.5  | psite  | 888 |
| 10.282.2.6  | q_box  | 888 |
| 10.282.3    | Constructor & Destructor Documentation               | 888 |
| 10.282.3.1  | l_p_line2d   | 888 |
| 10.282.3.2  | p_line2d   | 888 |
| 10.282.4    | Member Function Documentation                        | 888 |
| 10.282.4.1  | l_bbox   | 888 |
| 10.282.4.2  | begin  | 888 |
| 10.282.4.3  | end  | 889 |
| 10.282.4.4  | has  | 889 |
| 10.282.4.5  | has  | 889 |
| 10.282.4.6  | is_valid   | 889 |
| 10.282.4.7  | memory_size  | 889 |
| 10.282.4.8  | nsites   | 889 |
| 10.282.4.9  | operator[]   | 889 |
| 10.282.4.10 | l_fd_vector  | 890 |
| 10.283      | nl::p_mutable_array_of< S > Class Template Reference | 890 |
| 10.283.1    | Detailed Description                                 | 891 |
| 10.283.2    | Member Typedef Documentation                         | 891 |
| 10.283.2.1  | l_bkd_piter  | 891 |
| 10.283.2.2  | element  | 891 |
| 10.283.2.3  | l_fwd_piter  | 891 |
| 10.283.2.4  | element  | 892 |
| 10.283.2.5  | piter  | 892 |
| 10.283.2.6  | psite  | 892 |
| 10.283.3    | Constructor & Destructor Documentation               | 892 |
| 10.283.3.1  | l_p_mutable_array_of                                 | 892 |

|            |  |     |
|------------|--|-----|
| 10.283.4   | Member Function Documentation                            | 892 |
| 10.283.4.1 | clear  | 892 |
| 10.283.4.2 | has  | 892 |
| 10.283.4.3 | insert   | 892 |
| 10.283.4.4 | is_valid   | 893 |
| 10.283.4.5 | memory_size  | 893 |
| 10.283.4.6 | elements   | 893 |
| 10.283.4.7 | operator[]   | 893 |
| 10.283.4.8 | operator[]   | 893 |
| 10.283.4.9 | reserve  | 893 |
| 10.284     | ln::p_n_faces_bkd_piter< D, G > Class Template Reference | 893 |
| 10.284.1   | Detailed Description                                     | 894 |
| 10.284.2   | Constructor & Destructor Documentation                   | 894 |
| 10.284.2.1 | p_n_faces_bkd_piter                                      | 894 |
| 10.284.3   | Member Function Documentation                            | 894 |
| 10.284.3.1 | ln   | 894 |
| 10.284.3.2 | next   | 894 |
| 10.285     | ln::p_n_faces_fwd_piter< D, G > Class Template Reference | 895 |
| 10.285.1   | Detailed Description                                     | 895 |
| 10.285.2   | Constructor & Destructor Documentation                   | 895 |
| 10.285.2.1 | p_n_faces_fwd_piter                                      | 895 |
| 10.285.3   | Member Function Documentation                            | 895 |
| 10.285.3.1 | ln   | 895 |
| 10.285.3.2 | next   | 896 |
| 10.286     | ln::p_priority< P, Q > Class Template Reference          | 896 |
| 10.286.1   | Detailed Description                                     | 898 |
| 10.286.2   | Member Typedef Documentation                             | 898 |
| 10.286.2.1 | bkd_piter  | 898 |
| 10.286.2.2 | element  | 898 |
| 10.286.2.3 | fwd_piter  | 898 |
| 10.286.2.4 | _element   | 898 |
| 10.286.2.5 | piter  | 898 |
| 10.286.2.6 | psite  | 898 |
| 10.286.3   | Constructor & Destructor Documentation                   | 899 |
| 10.286.3.1 | p_priority   | 899 |
| 10.286.4   | Member Function Documentation                            | 899 |

|  |     |
|--|-----|
| 10.286.4.1clear                                      | 899 |
| 10.286.4.2exists_priority                            | 899 |
| 10.286.4.3front                                      | 899 |
| 10.286.4.4has  | 899 |
| 10.286.4.5highest_priority                           | 899 |
| 10.286.4.6insert                                     | 900 |
| 10.286.4.7insert                                     | 900 |
| 10.286.4.8s_valid                                    | 900 |
| 10.286.4.9lowest_priority                            | 900 |
| 10.286.4.10memory_size                               | 900 |
| 10.286.4.11sites                                     | 900 |
| 10.286.4.12operator()                                | 901 |
| 10.286.4.13pop                                       | 901 |
| 10.286.4.14pop_front                                 | 901 |
| 10.286.4.15priorities                                | 901 |
| 10.286.4.16push                                      | 901 |
| 10.287.1. In: p_queue < P > Class Template Reference | 902 |
| 10.287.1.1 Detailed Description                      | 903 |
| 10.287.1.2 Member Typedef Documentation              | 903 |
| 10.287.1.2.1 bkd_piter                               | 903 |
| 10.287.1.2.2 element                                 | 903 |
| 10.287.1.2.3 fwd_piter                               | 904 |
| 10.287.1.2.4 element                                 | 904 |
| 10.287.1.2.5 piter                                   | 904 |
| 10.287.1.2.6 psite                                   | 904 |
| 10.287.1.3 Constructor & Destructor Documentation    | 904 |
| 10.287.1.3.1 p_queue                                 | 904 |
| 10.287.1.4 Member Function Documentation             | 904 |
| 10.287.1.4.1 clear                                   | 904 |
| 10.287.1.4.2 front                                   | 904 |
| 10.287.1.4.3 has                                     | 904 |
| 10.287.1.4.4 has                                     | 905 |
| 10.287.1.4.5 insert                                  | 905 |
| 10.287.1.4.6 s_valid                                 | 905 |
| 10.287.1.4.7 memory_size                             | 905 |
| 10.287.1.4.8 sites                                   | 905 |





|             |  |     |
|-------------|--|-----|
| 10.289.2    | Member Typedef Documentation             | 913 |
| 10.289.2.1  | l_bkd_piter                              | 913 |
| 10.289.2.2  | element                                  | 913 |
| 10.289.2.3  | l_fwd_piter                              | 914 |
| 10.289.2.4  | piter                                    | 914 |
| 10.289.2.5  | psite                                    | 914 |
| 10.289.2.6  | q_box                                    | 914 |
| 10.289.3    | Constructor & Destructor Documentation   | 914 |
| 10.289.3.1  | lp_run                                   | 914 |
| 10.289.3.2  | pp_run                                   | 914 |
| 10.289.3.3  | pp_run                                   | 914 |
| 10.289.4    | Member Function Documentation            | 915 |
| 10.289.4.1  | l_bbox                                   | 915 |
| 10.289.4.2  | end                                      | 915 |
| 10.289.4.3  | has                                      | 915 |
| 10.289.4.4  | has                                      | 915 |
| 10.289.4.5  | has_index                                | 915 |
| 10.289.4.6  | init                                     | 915 |
| 10.289.4.7  | is_valid                                 | 916 |
| 10.289.4.8  | length                                   | 916 |
| 10.289.4.9  | memory_size                              | 916 |
| 10.289.4.10 | nsites                                   | 916 |
| 10.289.4.11 | operator[]                               | 916 |
| 10.289.4.12 | start                                    | 916 |
| 10.290      | l_n::p_set< P > Class Template Reference | 916 |
| 10.290.1    | Detailed Description                     | 918 |
| 10.290.2    | Member Typedef Documentation             | 918 |
| 10.290.2.1  | l_bkd_piter                              | 918 |
| 10.290.2.2  | element                                  | 918 |
| 10.290.2.3  | l_fwd_piter                              | 918 |
| 10.290.2.4  | l_element                                | 918 |
| 10.290.2.5  | piter                                    | 919 |
| 10.290.2.6  | psite                                    | 919 |
| 10.290.2.7  | r_element                                | 919 |
| 10.290.3    | Constructor & Destructor Documentation   | 919 |
| 10.290.3.1  | lp_set                                   | 919 |

|             |   |     |
|-------------|---|-----|
| 10.290.4    | Member Function Documentation                       | 919 |
| 10.290.4.1  | clear   | 919 |
| 10.290.4.2  | has   | 919 |
| 10.290.4.3  | has   | 919 |
| 10.290.4.4  | has   | 919 |
| 10.290.4.5  | insert  | 920 |
| 10.290.4.6  | is_valid  | 920 |
| 10.290.4.7  | memory_size   | 920 |
| 10.290.4.8  | nsites  | 920 |
| 10.290.4.9  | operator[]  | 920 |
| 10.290.4.10 | move  | 920 |
| 10.290.4.11 | std_vector  | 920 |
| 10.290.4.12 | til_set   | 921 |
| 10.291      | mln::p_set_of< S > Class Template Reference         | 921 |
| 10.291.1    | Detailed Description                                | 922 |
| 10.291.2    | Member Typedef Documentation                        | 922 |
| 10.291.2.1  | l_bkd_piter   | 922 |
| 10.291.2.2  | element   | 922 |
| 10.291.2.3  | fwd_piter   | 922 |
| 10.291.2.4  | _element  | 922 |
| 10.291.2.5  | piter   | 923 |
| 10.291.2.6  | psite   | 923 |
| 10.291.3    | Constructor & Destructor Documentation              | 923 |
| 10.291.3.1  | lp_set_of   | 923 |
| 10.291.4    | Member Function Documentation                       | 923 |
| 10.291.4.1  | clear   | 923 |
| 10.291.4.2  | has   | 923 |
| 10.291.4.3  | insert  | 923 |
| 10.291.4.4  | is_valid  | 923 |
| 10.291.4.5  | memory_size   | 923 |
| 10.291.4.6  | elements  | 924 |
| 10.291.4.7  | operator[]  | 924 |
| 10.292      | mln::p_transformed< S, F > Class Template Reference | 924 |
| 10.292.1    | Detailed Description                                | 925 |
| 10.292.2    | Member Typedef Documentation                        | 925 |
| 10.292.2.1  | l_bkd_piter   | 925 |

|  |     |
|--|-----|
| 10.292.2.2element  | 925 |
| 10.292.2.3fwd_piter  | 925 |
| 10.292.2.4piter  | 926 |
| 10.292.2.5psite  | 926 |
| 10.292.3Constructor & Destructor Documentation                       | 926 |
| 10.292.3.1p_transformed  | 926 |
| 10.292.3.2p_transformed  | 926 |
| 10.292.4Member Function Documentation                                | 926 |
| 10.292.4.1function   | 926 |
| 10.292.4.2has  | 926 |
| 10.292.4.3is_valid   | 926 |
| 10.292.4.4memory_size  | 927 |
| 10.292.4.5primary_set  | 927 |
| 10.293Inl::p_transformed_piter< Pi, S, F > Struct Template Reference | 927 |
| 10.293.1Detailed Description   | 927 |
| 10.293.2Constructor & Destructor Documentation                       | 928 |
| 10.293.2.1p_transformed_piter  | 928 |
| 10.293.2.2p_transformed_piter  | 928 |
| 10.293.3Member Function Documentation                                | 928 |
| 10.293.3.1change_target  | 928 |
| 10.293.3.2next   | 928 |
| 10.294Inl::p_vaccess< V, S > Class Template Reference                | 928 |
| 10.294.1Detailed Description   | 930 |
| 10.294.2Member Typedef Documentation                                 | 930 |
| 10.294.2.1bkd_piter  | 930 |
| 10.294.2.2element  | 930 |
| 10.294.2.3fwd_piter  | 930 |
| 10.294.2.4i_element  | 930 |
| 10.294.2.5piter  | 930 |
| 10.294.2.6pset   | 931 |
| 10.294.2.7psite  | 931 |
| 10.294.2.8value  | 931 |
| 10.294.2.9vset   | 931 |
| 10.294.3Constructor & Destructor Documentation                       | 931 |
| 10.294.3.1p_vaccess  | 931 |
| 10.294.4Member Function Documentation                                | 931 |



|            |  |     |
|------------|--|-----|
| 10.295.4.1 | operator()   | 938 |
| 10.296     | mln::pixel< I > Struct Template Reference          | 939 |
| 10.296.1   | Detailed Description                               | 939 |
| 10.296.2   | Constructor & Destructor Documentation             | 940 |
| 10.296.2.1 | pixel  | 940 |
| 10.296.2.2 | pixel  | 940 |
| 10.296.3   | Member Function Documentation                      | 940 |
| 10.296.3.1 | lchange_to   | 940 |
| 10.296.3.2 | is_valid   | 940 |
| 10.297     | mln::Pixel_Iterator< E > Struct Template Reference | 940 |
| 10.297.1   | Detailed Description                               | 941 |
| 10.297.2   | Member Function Documentation                      | 941 |
| 10.297.2.1 | next   | 941 |
| 10.298     | mln::plain< I > Class Template Reference           | 942 |
| 10.298.1   | Detailed Description                               | 942 |
| 10.298.2   | Member Typedef Documentation                       | 942 |
| 10.298.2.1 | Iskeleton  | 942 |
| 10.298.3   | Constructor & Destructor Documentation             | 943 |
| 10.298.3.1 | lplain   | 943 |
| 10.298.3.2 | plain  | 943 |
| 10.298.3.3 | plain  | 943 |
| 10.298.4   | Member Function Documentation                      | 943 |
| 10.298.4.1 | operator I   | 943 |
| 10.298.4.2 | operator=  | 943 |
| 10.298.4.3 | operator=  | 943 |
| 10.299     | mln::Point< P > Struct Template Reference          | 943 |
| 10.299.1   | Detailed Description                               | 945 |
| 10.299.2   | Member Typedef Documentation                       | 945 |
| 10.299.2.1 | lpoint   | 945 |
| 10.299.3   | Member Function Documentation                      | 945 |
| 10.299.3.1 | lto_point  | 945 |
| 10.299.4   | Friends And Related Function Documentation         | 945 |
| 10.299.4.1 | operator+=   | 945 |
| 10.299.4.2 | operator-=   | 946 |
| 10.299.4.3 | operator/  | 946 |
| 10.300     | mln::point< G, C > Struct Template Reference       | 946 |

---

|  |     |
|--|-----|
| 10.300.Detailed Description . . . . .                          | 949 |
| 10.300.2Member Typedef Documentation . . . . .                 | 949 |
| 10.300.2.1coord . . . . .                                      | 949 |
| 10.300.2.2delta . . . . .                                      | 949 |
| 10.300.2.3dpsite . . . . .                                     | 949 |
| 10.300.2.4grid . . . . .                                       | 949 |
| 10.300.2.5h_vec . . . . .                                      | 950 |
| 10.300.2.6vec . . . . .  | 950 |
| 10.300.3Member Enumeration Documentation . . . . .             | 950 |
| 10.300.3.1"@30 . . . . .                                       | 950 |
| 10.300.4Constructor & Destructor Documentation . . . . .       | 950 |
| 10.300.4.1point . . . . .                                      | 950 |
| 10.300.4.2point . . . . .                                      | 950 |
| 10.300.4.3point . . . . .                                      | 950 |
| 10.300.4.4point . . . . .                                      | 950 |
| 10.300.4.5point . . . . .                                      | 951 |
| 10.300.5Member Function Documentation . . . . .                | 951 |
| 10.300.5.1last_coord . . . . .                                 | 951 |
| 10.300.5.2last_coord . . . . .                                 | 951 |
| 10.300.5.3minus_infty . . . . .                                | 951 |
| 10.300.5.4operator+= . . . . .                                 | 951 |
| 10.300.5.5operator-= . . . . .                                 | 951 |
| 10.300.5.6operator[] . . . . .                                 | 951 |
| 10.300.5.7operator[] . . . . .                                 | 952 |
| 10.300.5.8plus_infty . . . . .                                 | 952 |
| 10.300.5.9set_all . . . . .                                    | 952 |
| 10.300.5.10h_vec . . . . .                                     | 952 |
| 10.300.5.11b_vec . . . . .                                     | 952 |
| 10.300.6Member Data Documentation . . . . .                    | 953 |
| 10.300.6.1origin . . . . .                                     | 953 |
| 10.301mln::Point_Site< E > Struct Template Reference . . . . . | 953 |
| 10.301.Detailed Description . . . . .                          | 954 |
| 10.301.2Friends And Related Function Documentation . . . . .   | 954 |
| 10.301.2.1operator+ . . . . .                                  | 954 |
| 10.301.2.2operator- . . . . .                                  | 955 |
| 10.301.2.3operator- . . . . .                                  | 955 |

|            |   |     |
|------------|---|-----|
| 10.301.2.4 | operator<<  | 956 |
| 10.301.2.5 | operator==  | 956 |
| 10.302     | mln::Point_Site< void > Struct Template Reference                       | 956 |
| 10.302     | Detailed Description  | 957 |
| 10.303     | mln::Proxy< E > Struct Template Reference                               | 957 |
| 10.303     | Detailed Description  | 957 |
| 10.304     | mln::Proxy< void > Struct Template Reference                            | 957 |
| 10.304     | Detailed Description  | 957 |
| 10.305     | mln::Pseudo_Site< E > Struct Template Reference                         | 957 |
| 10.305     | Detailed Description  | 958 |
| 10.306     | mln::Pseudo_Site< void > Struct Template Reference                      | 958 |
| 10.306     | Detailed Description  | 958 |
| 10.307     | mln::pw::image< F, S > Class Template Reference                         | 958 |
| 10.307     | Detailed Description  | 959 |
| 10.307     | Member Typedef Documentation  | 959 |
| 10.307.2   | Iskeleton   | 959 |
| 10.307     | Constructor & Destructor Documentation                                  | 959 |
| 10.307.3   | limage  | 959 |
| 10.307.3   | 2image  | 959 |
| 10.308     | mln::registration::closest_point_basic< P > Class Template Reference    | 959 |
| 10.308     | Detailed Description  | 960 |
| 10.309     | mln::registration::closest_point_with_map< P > Class Template Reference | 960 |
| 10.309     | Detailed Description  | 960 |
| 10.310     | mln::Regular_Grid< E > Struct Template Reference                        | 960 |
| 10.310     | Detailed Description  | 960 |
| 10.311     | mln::safe_image< I > Class Template Reference                           | 961 |
| 10.311     | Detailed Description  | 961 |
| 10.311     | Member Typedef Documentation  | 961 |
| 10.311.2   | Iskeleton   | 961 |
| 10.311     | Member Function Documentation   | 961 |
| 10.311.3   | loperator safe_image< const I >   | 961 |
| 10.312     | mln::select::p_of< P > Struct Template Reference                        | 961 |
| 10.312     | Detailed Description  | 962 |
| 10.313     | mln::Site< E > Struct Template Reference                                | 962 |
| 10.313     | Detailed Description  | 962 |
| 10.314     | mln::Site< void > Struct Template Reference                             | 963 |



|   |     |
|---|-----|
| 10.314. Detailed Description  | 963 |
| 10.315. <code>ln::Site_Iterator&lt; E &gt;</code> Struct Template Reference | 963 |
| 10.315. Detailed Description  | 965 |
| 10.315. Member Function Documentation                                       | 965 |
| 10.315.2. <code>lnext</code>  | 965 |
| 10.316. <code>ln::Site_Proxy&lt; E &gt;</code> Struct Template Reference    | 965 |
| 10.316. Detailed Description  | 965 |
| 10.317. <code>ln::Site_Proxy&lt; void &gt;</code> Struct Template Reference | 966 |
| 10.317. Detailed Description  | 966 |
| 10.318. <code>ln::Site_Set&lt; E &gt;</code> Struct Template Reference      | 966 |
| 10.318. Detailed Description  | 967 |
| 10.318. Friends And Related Function Documentation                          | 968 |
| 10.318.2. <code>ldiff</code>  | 968 |
| 10.318.2. <code>linter</code>   | 968 |
| 10.318.2.3. <code>operator&lt;</code>                                       | 968 |
| 10.318.2.4. <code>operator&lt;&lt;</code>                                   | 968 |
| 10.318.2.5. <code>operator&lt;=</code>                                      | 968 |
| 10.318.2.6. <code>operator==</code>   | 969 |
| 10.318.2.7. <code>sym_diff</code>   | 969 |
| 10.318.2.8. <code>uni</code>  | 969 |
| 10.318.2.9. <code>unique</code>   | 969 |
| 10.319. <code>ln::Site_Set&lt; void &gt;</code> Struct Template Reference   | 969 |
| 10.319. Detailed Description  | 969 |
| 10.320. <code>ln::slice_image&lt; I &gt;</code> Struct Template Reference   | 969 |
| 10.320. Detailed Description  | 970 |
| 10.320. Member Typedef Documentation  | 970 |
| 10.320.2. <code>lskeleton</code>  | 970 |
| 10.320. Constructor & Destructor Documentation                              | 971 |
| 10.320.3. <code>lslice_image</code>   | 971 |
| 10.320.3.2. <code>slice_image</code>  | 971 |
| 10.320. Member Function Documentation                                       | 971 |
| 10.320.4. <code>ldomain</code>  | 971 |
| 10.320.4.2. <code>operator slice_image&lt; const I &gt;</code>              | 971 |
| 10.320.4.3. <code>operator()</code>   | 971 |
| 10.320.4.4. <code>operator()</code>   | 971 |
| 10.320.4.5. <code>li</code>   | 971 |

---

|            |  |     |
|------------|--|-----|
| 10.321     | nl::sub_image< I, S > Class Template Reference                                   | 972 |
| 10.321.1   | Detailed Description   | 972 |
| 10.321.2   | Member Typedef Documentation   | 972 |
| 10.321.2.1 | Iskeleton  | 972 |
| 10.321.3   | Constructor & Destructor Documentation   | 973 |
| 10.321.3.1 | Isub_image   | 973 |
| 10.321.3.2 | Sub_image  | 973 |
| 10.321.4   | Member Function Documentation  | 973 |
| 10.321.4.1 | Idomain  | 973 |
| 10.321.4.2 | operator sub_image< const I, S >   | 973 |
| 10.322     | nl::sub_image_if< I, S > Struct Template Reference                               | 973 |
| 10.322.1   | Detailed Description   | 974 |
| 10.322.2   | Member Typedef Documentation   | 974 |
| 10.322.2.1 | Iskeleton  | 974 |
| 10.322.3   | Constructor & Destructor Documentation   | 974 |
| 10.322.3.1 | Isub_image_if  | 974 |
| 10.322.3.2 | Sub_image_if   | 974 |
| 10.322.4   | Member Function Documentation  | 974 |
| 10.322.4.1 | Idomain  | 974 |
| 10.323     | nl::thru_image< I, F > Class Template Reference                                  | 975 |
| 10.323.1   | Detailed Description   | 975 |
| 10.323.2   | Member Function Documentation  | 975 |
| 10.323.2.1 | operator thru_image< const I, F >  | 975 |
| 10.324     | nl::thrubin_image< I1, I2, F > Class Template Reference                          | 975 |
| 10.324.1   | Detailed Description   | 976 |
| 10.324.2   | Member Typedef Documentation   | 976 |
| 10.324.2.1 | Ipsite   | 976 |
| 10.324.2.2 | Ivalue   | 976 |
| 10.324.2.3 | Iskeleton  | 976 |
| 10.324.2.4 | Ivalue   | 976 |
| 10.324.3   | Member Function Documentation  | 976 |
| 10.324.3.1 | operator thrubin_image< const I1, const I2, F >                                  | 976 |
| 10.325     | nl::topo::adj_higher_dim_connected_n_face_bkd_iter< D > Class Template Reference | 977 |
| 10.325.1   | Detailed Description   | 977 |
| 10.325.2   | Constructor & Destructor Documentation   | 977 |
| 10.325.2.1 | Iadj_higher_dim_connected_n_face_bkd_iter  | 977 |

---

|           |   |     |
|-----------|---|-----|
| 10.325.   | Member Function Documentation   | 978 |
| 10.325.3. | lnext   | 978 |
| 10.326.   | mln::topo::adj_higher_dim_connected_n_face_fwd_iter< D > Class Template Reference | 978 |
| 10.326.   | Detailed Description  | 978 |
| 10.326.   | Constructor & Destructor Documentation  | 979 |
| 10.326.2. | ladj_higher_dim_connected_n_face_fwd_iter   | 979 |
| 10.326.   | Member Function Documentation   | 979 |
| 10.326.3. | lnext   | 979 |
| 10.327.   | mln::topo::adj_higher_face_bkd_iter< D > Class Template Reference                 | 979 |
| 10.327.   | Detailed Description  | 979 |
| 10.327.   | Constructor & Destructor Documentation  | 980 |
| 10.327.2. | ladj_higher_face_bkd_iter   | 980 |
| 10.327.   | Member Function Documentation   | 980 |
| 10.327.3. | lnext   | 980 |
| 10.328.   | mln::topo::adj_higher_face_fwd_iter< D > Class Template Reference                 | 980 |
| 10.328.   | Detailed Description  | 981 |
| 10.328.   | Constructor & Destructor Documentation  | 981 |
| 10.328.2. | ladj_higher_face_fwd_iter   | 981 |
| 10.328.   | Member Function Documentation   | 981 |
| 10.328.3. | lnext   | 981 |
| 10.329.   | mln::topo::adj_lower_dim_connected_n_face_bkd_iter< D > Class Template Reference  | 981 |
| 10.329.   | Detailed Description  | 982 |
| 10.329.   | Constructor & Destructor Documentation  | 982 |
| 10.329.2. | ladj_lower_dim_connected_n_face_bkd_iter  | 982 |
| 10.329.   | Member Function Documentation   | 982 |
| 10.329.3. | lnext   | 982 |
| 10.330.   | mln::topo::adj_lower_dim_connected_n_face_fwd_iter< D > Class Template Reference  | 982 |
| 10.330.   | Detailed Description  | 983 |
| 10.330.   | Constructor & Destructor Documentation  | 983 |
| 10.330.2. | ladj_lower_dim_connected_n_face_fwd_iter  | 983 |
| 10.330.   | Member Function Documentation   | 983 |
| 10.330.3. | lnext   | 983 |
| 10.331.   | mln::topo::adj_lower_face_bkd_iter< D > Class Template Reference                  | 984 |
| 10.331.   | Detailed Description  | 984 |
| 10.331.   | Constructor & Destructor Documentation  | 984 |
| 10.331.2. | ladj_lower_face_bkd_iter  | 984 |

|           |  |     |
|-----------|--|-----|
| 10.331.   | Member Function Documentation  | 984 |
| 10.331.3. | lnext  | 984 |
| 10.332.   | nl::topo::adj_lower_face_fwd_iter< D > Class Template Reference        | 985 |
| 10.332.   | Detailed Description   | 985 |
| 10.332.   | Constructor & Destructor Documentation                                 | 985 |
| 10.332.2. | ladj_lower_face_fwd_iter   | 985 |
| 10.332.   | Member Function Documentation  | 985 |
| 10.332.3. | lnext  | 985 |
| 10.333.   | nl::topo::adj_lower_higher_face_bkd_iter< D > Class Template Reference | 986 |
| 10.333.   | Detailed Description   | 986 |
| 10.333.   | Constructor & Destructor Documentation                                 | 986 |
| 10.333.2. | ladj_lower_higher_face_bkd_iter  | 986 |
| 10.333.   | Member Function Documentation  | 987 |
| 10.333.3. | lnext  | 987 |
| 10.334.   | nl::topo::adj_lower_higher_face_fwd_iter< D > Class Template Reference | 987 |
| 10.334.   | Detailed Description   | 987 |
| 10.334.   | Constructor & Destructor Documentation                                 | 988 |
| 10.334.2. | ladj_lower_higher_face_fwd_iter  | 988 |
| 10.334.   | Member Function Documentation  | 988 |
| 10.334.3. | lnext  | 988 |
| 10.335.   | nl::topo::adj_m_face_bkd_iter< D > Class Template Reference            | 988 |
| 10.335.   | Detailed Description   | 989 |
| 10.335.   | Constructor & Destructor Documentation                                 | 989 |
| 10.335.2. | ladj_m_face_bkd_iter   | 989 |
| 10.335.2. | 2adj_m_face_bkd_iter   | 989 |
| 10.335.   | Member Function Documentation  | 989 |
| 10.335.3. | lnext  | 989 |
| 10.336.   | nl::topo::adj_m_face_fwd_iter< D > Class Template Reference            | 989 |
| 10.336.   | Detailed Description   | 990 |
| 10.336.   | Constructor & Destructor Documentation                                 | 990 |
| 10.336.2. | ladj_m_face_fwd_iter   | 990 |
| 10.336.2. | 2adj_m_face_fwd_iter   | 990 |
| 10.336.   | Member Function Documentation  | 991 |
| 10.336.3. | lnext  | 991 |
| 10.337.   | nl::topo::algebraic_face< D > Class Template Reference                 | 991 |
| 10.337.   | Detailed Description   | 993 |

|             |   |     |
|-------------|---|-----|
| 10.337.1    | Constructor & Destructor Documentation                      | 993 |
| 10.337.2.1  | algebraic_face  | 993 |
| 10.337.2.2  | algebraic_face  | 993 |
| 10.337.2.3  | algebraic_face  | 993 |
| 10.337.2.4  | algebraic_face  | 993 |
| 10.337.3    | Member Function Documentation                               | 993 |
| 10.337.3.1  | cplx  | 993 |
| 10.337.3.2  | data  | 994 |
| 10.337.3.3  | dec_face_id   | 994 |
| 10.337.3.4  | dec_n   | 994 |
| 10.337.3.5  | face_id   | 994 |
| 10.337.3.6  | higher_dim_adj_faces  | 994 |
| 10.337.3.7  | inc_face_id   | 994 |
| 10.337.3.8  | inc_n   | 994 |
| 10.337.3.9  | invalidate  | 995 |
| 10.337.3.10 | is_valid  | 995 |
| 10.337.3.11 | lower_dim_adj_faces   | 995 |
| 10.337.3.12 |   | 995 |
| 10.337.3.13 | set_cplx  | 995 |
| 10.337.3.14 | set_face_id   | 995 |
| 10.337.3.15 | set_n   | 995 |
| 10.337.3.16 | set_sign  | 996 |
| 10.337.3.17 | sign  | 996 |
| 10.338      | In::topo::algebraic_n_face< N, D > Class Template Reference | 996 |
| 10.338.1    | Detailed Description  | 998 |
| 10.338.2    | Constructor & Destructor Documentation                      | 998 |
| 10.338.2.1  | algebraic_n_face  | 998 |
| 10.338.2.2  | algebraic_n_face  | 998 |
| 10.338.2.3  | algebraic_n_face  | 998 |
| 10.338.3    | Member Function Documentation                               | 998 |
| 10.338.3.1  | cplx  | 998 |
| 10.338.3.2  | data  | 998 |
| 10.338.3.3  | dec_face_id   | 999 |
| 10.338.3.4  | face_id   | 999 |
| 10.338.3.5  | higher_dim_adj_faces  | 999 |
| 10.338.3.6  | inc_face_id   | 999 |

|             |  |      |
|-------------|--|------|
| 10.338.3.7  | invalidate   | 999  |
| 10.338.3.8  | is_valid   | 999  |
| 10.338.3.9  | lower_dim_adj_faces  | 1000 |
| 10.338.3.10 |  | 1000 |
| 10.338.3.11 | set_cplx   | 1000 |
| 10.338.3.12 | set_face_id  | 1000 |
| 10.338.3.13 | set_sign   | 1000 |
| 10.338.3.14 | sign   | 1000 |
| 10.339      | nl::topo::center_only_iter< D > Class Template Reference             | 1001 |
| 10.339.1    | Detailed Description   | 1001 |
| 10.339.2    | Constructor & Destructor Documentation                               | 1001 |
| 10.339.2.1  | center_only_iter   | 1001 |
| 10.339.3    | Member Function Documentation  | 1002 |
| 10.339.3.1  | next   | 1002 |
| 10.340      | nl::topo::centered_bkd_iter_adapter< D, I > Class Template Reference | 1002 |
| 10.340.1    | Detailed Description   | 1002 |
| 10.340.2    | Constructor & Destructor Documentation                               | 1003 |
| 10.340.2.1  | centered_bkd_iter_adapter  | 1003 |
| 10.340.3    | Member Function Documentation  | 1003 |
| 10.340.3.1  | next   | 1003 |
| 10.341      | nl::topo::centered_fwd_iter_adapter< D, I > Class Template Reference | 1003 |
| 10.341.1    | Detailed Description   | 1003 |
| 10.341.2    | Constructor & Destructor Documentation                               | 1004 |
| 10.341.2.1  | centered_fwd_iter_adapter  | 1004 |
| 10.341.3    | Member Function Documentation  | 1004 |
| 10.341.3.1  | next   | 1004 |
| 10.342      | nl::topo::complex< D > Class Template Reference                      | 1004 |
| 10.342.1    | Detailed Description   | 1005 |
| 10.342.2    | Member Typedef Documentation   | 1005 |
| 10.342.2.1  | bkd_citer  | 1005 |
| 10.342.2.2  | fwd_citer  | 1006 |
| 10.342.3    | Constructor & Destructor Documentation                               | 1006 |
| 10.342.3.1  | complex  | 1006 |
| 10.342.4    | Member Function Documentation  | 1006 |
| 10.342.4.1  | add_face   | 1006 |
| 10.342.4.2  | add_face   | 1006 |

|             |  |      |
|-------------|--|------|
| 10.342.4.3  | addr                                   | 1006 |
| 10.342.4.4  | faces                                  | 1006 |
| 10.342.4.5  | faces_of_dim                           | 1007 |
| 10.342.4.6  | faces_of_static_dim                    | 1007 |
| 10.342.4.7  | print                                  | 1007 |
| 10.342.4.8  | print_faces                            | 1007 |
| 10.343      | Class Template Reference               | 1007 |
| 10.343.1    | Detailed Description                   | 1009 |
| 10.343.2    | Constructor & Destructor Documentation | 1009 |
| 10.343.2.1  | iface                                  | 1009 |
| 10.343.2.2  | face                                   | 1009 |
| 10.343.2.3  | face                                   | 1009 |
| 10.343.3    | Member Function Documentation          | 1010 |
| 10.343.3.1  | cplx                                   | 1010 |
| 10.343.3.2  | data                                   | 1010 |
| 10.343.3.3  | dec_face_id                            | 1010 |
| 10.343.3.4  | dec_n                                  | 1010 |
| 10.343.3.5  | face_id                                | 1010 |
| 10.343.3.6  | higher_dim_adj_faces                   | 1010 |
| 10.343.3.7  | inc_face_id                            | 1011 |
| 10.343.3.8  | inc_n                                  | 1011 |
| 10.343.3.9  | invalidate                             | 1011 |
| 10.343.3.10 | is_valid                               | 1011 |
| 10.343.3.11 | lower_dim_adj_faces                    | 1011 |
| 10.343.3.12 | l2                                     | 1011 |
| 10.343.3.13 | set_cplx                               | 1011 |
| 10.343.3.14 | set_face_id                            | 1012 |
| 10.343.3.15 | set_n                                  | 1012 |
| 10.344      | Class Template Reference               | 1012 |
| 10.344.1    | Detailed Description                   | 1012 |
| 10.344.2    | Constructor & Destructor Documentation | 1013 |
| 10.344.2.1  | iface_bkd_iter                         | 1013 |
| 10.344.3    | Member Function Documentation          | 1013 |
| 10.344.3.1  | next                                   | 1013 |
| 10.344.3.2  | start                                  | 1013 |
| 10.345      | Class Template Reference               | 1013 |

|  |      |
|--|------|
| 10.345. Detailed Description . . . . .                                     | 1014 |
| 10.345. Constructor & Destructor Documentation . . . . .                   | 1014 |
| 10.345.2. lface_fwd_iter . . . . .   | 1014 |
| 10.345. Member Function Documentation . . . . .                            | 1014 |
| 10.345.3. lnext . . . . .  | 1014 |
| 10.345.3.2start . . . . .  | 1014 |
| 10.346. mln::topo::is_n_face< N > Struct Template Reference . . . . .      | 1014 |
| 10.346. Detailed Description . . . . .                                     | 1015 |
| 10.347. mln::topo::is_simple_2d_t< N > Struct Template Reference . . . . . | 1015 |
| 10.347. Detailed Description . . . . .                                     | 1016 |
| 10.348. mln::topo::is_simple_cell< I > Class Template Reference . . . . .  | 1016 |
| 10.348. Detailed Description . . . . .                                     | 1018 |
| 10.348. Member Typedef Documentation . . . . .                             | 1018 |
| 10.348.2. lpsite . . . . .   | 1018 |
| 10.348.2.2result . . . . .   | 1018 |
| 10.348. Member Function Documentation . . . . .                            | 1018 |
| 10.348.3. lmln_geom . . . . .  | 1018 |
| 10.348.3.2operator() . . . . .   | 1018 |
| 10.348.3.3set_image . . . . .  | 1019 |
| 10.348. Member Data Documentation . . . . .                                | 1019 |
| 10.348.4. ID . . . . .   | 1019 |
| 10.349. mln::topo::n_face< N, D > Class Template Reference . . . . .       | 1019 |
| 10.349. Detailed Description . . . . .                                     | 1020 |
| 10.349. Constructor & Destructor Documentation . . . . .                   | 1020 |
| 10.349.2. ln_face . . . . .  | 1020 |
| 10.349.2.2n_face . . . . .   | 1021 |
| 10.349. Member Function Documentation . . . . .                            | 1021 |
| 10.349.3. lcplx . . . . .  | 1021 |
| 10.349.3.2data . . . . .   | 1021 |
| 10.349.3.3dec_face_id . . . . .  | 1021 |
| 10.349.3.4face_id . . . . .  | 1021 |
| 10.349.3.5higher_dim_adj_faces . . . . .                                   | 1021 |
| 10.349.3.6nc_face_id . . . . .   | 1022 |
| 10.349.3.7invalidate . . . . .   | 1022 |
| 10.349.3.8s_valid . . . . .  | 1022 |
| 10.349.3.9lower_dim_adj_faces . . . . .                                    | 1022 |



|  |      |
|--|------|
| 10.349.3.10  | 1022 |
| 10.349.3.1set_cplx   | 1022 |
| 10.349.3.1set_face_id  | 1022 |
| 10.350. In::topo::n_face_bkd_iter< D > Class Template Reference            | 1023 |
| 10.350. Detailed Description   | 1023 |
| 10.350. Constructor & Destructor Documentation                             | 1023 |
| 10.350.2. In_face_bkd_iter   | 1023 |
| 10.350. Member Function Documentation                                      | 1024 |
| 10.350.3. In   | 1024 |
| 10.350.3.2next   | 1024 |
| 10.350.3.3start  | 1024 |
| 10.351. In::topo::n_face_fwd_iter< D > Class Template Reference            | 1024 |
| 10.351. Detailed Description   | 1025 |
| 10.351. Constructor & Destructor Documentation                             | 1025 |
| 10.351.2. In_face_fwd_iter   | 1025 |
| 10.351. Member Function Documentation                                      | 1025 |
| 10.351.3. In   | 1025 |
| 10.351.3.2next   | 1025 |
| 10.351.3.3start  | 1026 |
| 10.352. In::topo::n_faces_set< N, D > Class Template Reference             | 1026 |
| 10.352. Detailed Description   | 1026 |
| 10.352. Member Typedef Documentation                                       | 1026 |
| 10.352.2. Ifaces_type  | 1026 |
| 10.352. Member Function Documentation                                      | 1027 |
| 10.352.3. ladd   | 1027 |
| 10.352.3.2faces  | 1027 |
| 10.352.3.3reserve  | 1027 |
| 10.353. In::topo::skeleton::is_simple_point< N > Struct Template Reference | 1027 |
| 10.353. Detailed Description   | 1027 |
| 10.354. In::topo::static_n_face_bkd_iter< N, D > Class Template Reference  | 1028 |
| 10.354. Detailed Description   | 1028 |
| 10.354. Constructor & Destructor Documentation                             | 1028 |
| 10.354.2. Istatic_n_face_bkd_iter  | 1028 |
| 10.354. Member Function Documentation                                      | 1029 |
| 10.354.3. Inext  | 1029 |
| 10.354.3.2start  | 1029 |

|            |  |      |
|------------|--|------|
| 10.355     | <code>ln::topo::static_n_face_fwd_iter&lt; N, D &gt;</code> Class Template Reference . . . . . | 1029 |
| 10.355.1   | Detailed Description . . . . .   | 1029 |
| 10.355.2   | Constructor & Destructor Documentation . . . . .   | 1030 |
| 10.355.2.1 | <code>static_n_face_fwd_iter</code> . . . . .  | 1030 |
| 10.355.3   | Member Function Documentation . . . . .  | 1030 |
| 10.355.3.1 | <code>next</code> . . . . .  | 1030 |
| 10.355.3.2 | <code>start</code> . . . . .   | 1030 |
| 10.356     | <code>ln::tr_image&lt; S, I, T &gt;</code> Struct Template Reference . . . . .                 | 1030 |
| 10.356.1   | Detailed Description . . . . .   | 1031 |
| 10.356.2   | Member Typedef Documentation . . . . .   | 1032 |
| 10.356.2.1 | <code>lvalue</code> . . . . .  | 1032 |
| 10.356.2.2 | <code>psite</code> . . . . .   | 1032 |
| 10.356.2.3 | <code>rvalue</code> . . . . .  | 1032 |
| 10.356.2.4 | <code>site</code> . . . . .  | 1032 |
| 10.356.2.5 | <code>skeleton</code> . . . . .  | 1032 |
| 10.356.2.6 | <code>value</code> . . . . .   | 1032 |
| 10.356.3   | Constructor & Destructor Documentation . . . . .   | 1032 |
| 10.356.3.1 | <code>tr_image</code> . . . . .  | 1032 |
| 10.356.4   | Member Function Documentation . . . . .  | 1033 |
| 10.356.4.1 | <code>domain</code> . . . . .  | 1033 |
| 10.356.4.2 | <code>has</code> . . . . .   | 1033 |
| 10.356.4.3 | <code>is_valid</code> . . . . .  | 1033 |
| 10.356.4.4 | <code>operator()</code> . . . . .  | 1033 |
| 10.356.4.5 | <code>set_tr</code> . . . . .  | 1033 |
| 10.356.4.6 | <code>tr</code> . . . . .  | 1033 |
| 10.357     | <code>ln::unproject_image&lt; I, D, F &gt;</code> Struct Template Reference . . . . .          | 1033 |
| 10.357.1   | Detailed Description . . . . .   | 1034 |
| 10.357.2   | Constructor & Destructor Documentation . . . . .   | 1034 |
| 10.357.2.1 | <code>unproject_image</code> . . . . .   | 1034 |
| 10.357.2.2 | <code>unproject_image</code> . . . . .   | 1034 |
| 10.357.3   | Member Function Documentation . . . . .  | 1034 |
| 10.357.3.1 | <code>domain</code> . . . . .  | 1034 |
| 10.357.3.2 | <code>operator()</code> . . . . .  | 1035 |
| 10.357.3.3 | <code>operator()</code> . . . . .  | 1035 |
| 10.358     | <code>ln::util::adjacency_matrix&lt; V &gt;</code> Class Template Reference . . . . .          | 1035 |
| 10.358.1   | Detailed Description . . . . .   | 1035 |

|  |      |
|--|------|
| 10.358. Constructor & Destructor Documentation . . . . .         | 1035 |
| 10.358.2.1adjacency_matrix . . . . .                             | 1035 |
| 10.358.2.2adjacency_matrix . . . . .                             | 1036 |
| 10.359. In::util::array< T > Class Template Reference . . . . .  | 1036 |
| 10.359. Detailed Description . . . . .                           | 1038 |
| 10.359. Member Typedef Documentation . . . . .                   | 1038 |
| 10.359.2.1bkd_eiter . . . . .                                    | 1038 |
| 10.359.2.2eiter . . . . .  | 1038 |
| 10.359.2.3element . . . . .                                      | 1039 |
| 10.359.2.4fwd_eiter . . . . .                                    | 1039 |
| 10.359.2.5result . . . . .                                       | 1039 |
| 10.359. Constructor & Destructor Documentation . . . . .         | 1039 |
| 10.359.3.1array . . . . .  | 1039 |
| 10.359.3.2array . . . . .  | 1039 |
| 10.359.3.3array . . . . .  | 1039 |
| 10.359. Member Function Documentation . . . . .                  | 1039 |
| 10.359.4.1append . . . . .                                       | 1039 |
| 10.359.4.2append . . . . .                                       | 1040 |
| 10.359.4.3clear . . . . .  | 1040 |
| 10.359.4.4fill . . . . .   | 1040 |
| 10.359.4.5is_empty . . . . .                                     | 1040 |
| 10.359.4.6last . . . . .   | 1040 |
| 10.359.4.7last . . . . .   | 1041 |
| 10.359.4.8memory_size . . . . .                                  | 1041 |
| 10.359.4.9elements . . . . .                                     | 1041 |
| 10.359.4.10operator() . . . . .                                  | 1041 |
| 10.359.4.11operator() . . . . .                                  | 1041 |
| 10.359.4.12operator[] . . . . .                                  | 1042 |
| 10.359.4.13operator[] . . . . .                                  | 1042 |
| 10.359.4.14reserve . . . . .                                     | 1042 |
| 10.359.4.15size . . . . .  | 1042 |
| 10.359.4.16size . . . . .  | 1042 |
| 10.359.4.17size . . . . .  | 1042 |
| 10.359.4.18d_vector . . . . .                                    | 1043 |
| 10.360. In::util::branch< T > Class Template Reference . . . . . | 1043 |
| 10.360. Detailed Description . . . . .                           | 1043 |

|            |   |      |
|------------|---|------|
| 10.360.2   | Constructor & Destructor Documentation              | 1043 |
| 10.360.2.1 | branch  | 1043 |
| 10.360.3   | Member Function Documentation                       | 1044 |
| 10.360.3.1 | lapex   | 1044 |
| 10.360.3.2 | util_tree   | 1044 |
| 10.361     | util::branch_iter< T > Class Template Reference     | 1044 |
| 10.361.1   | Detailed Description                                | 1045 |
| 10.361.2   | Member Function Documentation                       | 1045 |
| 10.361.2.1 | deepness  | 1045 |
| 10.361.2.2 | invalidate  | 1045 |
| 10.361.2.3 | is_valid  | 1045 |
| 10.361.2.4 | next  | 1045 |
| 10.361.2.5 | operator util::tree_node< T > &                     | 1046 |
| 10.361.2.6 | start   | 1046 |
| 10.362     | util::branch_iter_ind< T > Class Template Reference | 1046 |
| 10.362.1   | Detailed Description                                | 1046 |
| 10.362.2   | Member Function Documentation                       | 1047 |
| 10.362.2.1 | deepness  | 1047 |
| 10.362.2.2 | invalidate  | 1047 |
| 10.362.2.3 | is_valid  | 1047 |
| 10.362.2.4 | next  | 1047 |
| 10.362.2.5 | operator util::tree_node< T > &                     | 1047 |
| 10.362.2.6 | start   | 1047 |
| 10.363     | util::couple< T, U > Class Template Reference       | 1047 |
| 10.363.1   | Detailed Description                                | 1048 |
| 10.363.2   | Member Function Documentation                       | 1049 |
| 10.363.2.1 | change_both   | 1049 |
| 10.363.2.2 | change_first  | 1049 |
| 10.363.2.3 | change_second                                       | 1049 |
| 10.363.2.4 | first   | 1049 |
| 10.363.2.5 | second  | 1049 |
| 10.364     | util::eat Struct Reference                          | 1049 |
| 10.364.1   | Detailed Description                                | 1050 |
| 10.365     | util::edge< G > Class Template Reference            | 1050 |
| 10.365.1   | Detailed Description                                | 1051 |
| 10.365.2   | Member Typedef Documentation                        | 1052 |

|   |      |
|---|------|
| 10.365.2.1category  | 1052 |
| 10.365.2.2graph_t   | 1052 |
| 10.365.2.3id_t  | 1052 |
| 10.365.2.4id_value_t  | 1052 |
| 10.365.3Constructor & Destructor Documentation                  | 1052 |
| 10.365.3.1edge  | 1052 |
| 10.365.4Member Function Documentation                           | 1052 |
| 10.365.4.1change_graph  | 1052 |
| 10.365.4.2graph   | 1052 |
| 10.365.4.3id  | 1053 |
| 10.365.4.4invalidate  | 1053 |
| 10.365.4.5is_valid  | 1053 |
| 10.365.4.6th_nbh_edge   | 1053 |
| 10.365.4.7nmax_nbh_edges  | 1053 |
| 10.365.4.8operator edge_id_t                                    | 1053 |
| 10.365.4.9update_id   | 1053 |
| 10.365.4.10l  | 1054 |
| 10.365.4.112  | 1054 |
| 10.365.4.12_other   | 1054 |
| 10.366In::util::fibonacci_heap< P, T > Class Template Reference | 1054 |
| 10.366.1Detailed Description                                    | 1056 |
| 10.366.2Constructor & Destructor Documentation                  | 1056 |
| 10.366.2.1fibonacci_heap  | 1056 |
| 10.366.2.2fibonacci_heap  | 1056 |
| 10.366.3Member Function Documentation                           | 1056 |
| 10.366.3.1clear   | 1056 |
| 10.366.3.2front   | 1056 |
| 10.366.3.3is_empty  | 1057 |
| 10.366.3.4is_valid  | 1057 |
| 10.366.3.5elements  | 1057 |
| 10.366.3.6operator=   | 1057 |
| 10.366.3.7pop_front   | 1057 |
| 10.366.3.8push  | 1057 |
| 10.366.3.9push  | 1058 |
| 10.367In::util::graph Class Reference                           | 1058 |
| 10.367.1Detailed Description                                    | 1060 |

|             |   |      |
|-------------|---|------|
| 10.367.2    | Member Typedef Documentation                          | 1060 |
| 10.367.2.1  | edge_fwd_iter   | 1060 |
| 10.367.2.2  | edge_nbh_edge_fwd_iter                                | 1060 |
| 10.367.2.3  | edges_set_t   | 1060 |
| 10.367.2.4  | edges_t   | 1061 |
| 10.367.2.5  | vertex_fwd_iter                                       | 1061 |
| 10.367.2.6  | vertex_nbh_edge_fwd_iter                              | 1061 |
| 10.367.2.7  | vertex_nbh_vertex_fwd_iter                            | 1061 |
| 10.367.2.8  | vertices_t  | 1061 |
| 10.367.3    | Constructor & Destructor Documentation                | 1061 |
| 10.367.3.1  | lgraph  | 1061 |
| 10.367.3.2  | graph   | 1061 |
| 10.367.4    | Member Function Documentation                         | 1062 |
| 10.367.4.1  | add_edge  | 1062 |
| 10.367.4.2  | add_vertex  | 1062 |
| 10.367.4.3  | add_vertices  | 1062 |
| 10.367.4.4  | e_ith_nbh_edge  | 1062 |
| 10.367.4.5  | e_nmax  | 1063 |
| 10.367.4.6  | e_nmax_nbh_edges                                      | 1063 |
| 10.367.4.7  | edge  | 1063 |
| 10.367.4.8  | edge  | 1063 |
| 10.367.4.9  | edges   | 1063 |
| 10.367.4.10 | has_e   | 1063 |
| 10.367.4.11 | has_v   | 1064 |
| 10.367.4.12 | is_subgraph_of  | 1064 |
| 10.367.4.13 | l1  | 1064 |
| 10.367.4.14 | l2  | 1064 |
| 10.367.4.15 | l5_ith_nbh_edge                                       | 1064 |
| 10.367.4.16 | l6_ith_nbh_vertex                                     | 1064 |
| 10.367.4.17 | l7_nmax   | 1065 |
| 10.367.4.18 | l8_nmax_nbh_edges                                     | 1065 |
| 10.367.4.19 | l9_nmax_nbh_vertices                                  | 1065 |
| 10.367.4.20 | lvertex   | 1065 |
| 10.368      | ln::util::greater_point< I > Class Template Reference | 1065 |
| 10.368.1    | Detailed Description                                  | 1065 |
| 10.368.2    | Member Function Documentation                         | 1066 |

|  |      |
|--|------|
| 10.368.2.operator()  | 1066 |
| 10.369. <a href="#">util::greater_site&lt; I &gt; Class Template Reference</a> | 1066 |
| 10.369. Detailed Description   | 1066 |
| 10.369. Member Function Documentation  | 1066 |
| 10.369.2.operator()  | 1066 |
| 10.370. <a href="#">util::head&lt; T, R &gt; Class Template Reference</a>      | 1066 |
| 10.370. Detailed Description   | 1067 |
| 10.371. <a href="#">util::ignore Struct Reference</a>                          | 1067 |
| 10.371. Detailed Description   | 1067 |
| 10.372. <a href="#">util::ilcell&lt; T &gt; Struct Template Reference</a>      | 1067 |
| 10.372. Detailed Description   | 1068 |
| 10.373. <a href="#">util::line_graph&lt; G &gt; Class Template Reference</a>   | 1068 |
| 10.373. Detailed Description   | 1070 |
| 10.373. Member Typedef Documentation   | 1070 |
| 10.373.2. <a href="#">ledge_fwd_iter</a>                                       | 1070 |
| 10.373.2.2. <a href="#">edge_nbh_edge_fwd_iter</a>                             | 1070 |
| 10.373.2.3. <a href="#">edges_t</a>  | 1070 |
| 10.373.2.4. <a href="#">vertex_fwd_iter</a>                                    | 1070 |
| 10.373.2.5. <a href="#">vertex_nbh_edge_fwd_iter</a>                           | 1071 |
| 10.373.2.6. <a href="#">vertex_nbh_vertex_fwd_iter</a>                         | 1071 |
| 10.373.2.7. <a href="#">vertices_t</a>   | 1071 |
| 10.373. Member Function Documentation  | 1071 |
| 10.373.3.1. <a href="#">le_ith_nbh_edge</a>                                    | 1071 |
| 10.373.3.2. <a href="#">e_nmax</a>   | 1071 |
| 10.373.3.3. <a href="#">e_nmax_nbh_edges</a>                                   | 1071 |
| 10.373.3.4. <a href="#">edge</a>   | 1072 |
| 10.373.3.5. <a href="#">graph</a>  | 1072 |
| 10.373.3.6. <a href="#">has</a>  | 1072 |
| 10.373.3.7. <a href="#">has</a>  | 1072 |
| 10.373.3.8. <a href="#">has_e</a>  | 1072 |
| 10.373.3.9. <a href="#">has_v</a>  | 1072 |
| 10.373.3.10. <a href="#">is_subgraph_of</a>                                    | 1073 |
| 10.373.3.11. <a href="#">is</a>  | 1073 |
| 10.373.3.12. <a href="#">is</a>  | 1073 |
| 10.373.3.13. <a href="#">is_ith_nbh_edge</a>                                   | 1073 |
| 10.373.3.14. <a href="#">is_ith_nbh_vertex</a>                                 | 1073 |

|   |      |
|---|------|
| 10.373.3.15_nmax  | 1074 |
| 10.373.3.16_nmax_nbh_edges  | 1074 |
| 10.373.3.17_nmax_nbh_vertices   | 1074 |
| 10.373.3.18_vertex  | 1074 |
| 10.374. <code>util::nil</code> Struct Reference                               | 1074 |
| 10.374. Detailed Description  | 1075 |
| 10.375. <code>util::node&lt; T, R &gt;</code> Class Template Reference        | 1075 |
| 10.375. Detailed Description  | 1075 |
| 10.376. <code>util::object_id&lt; Tag, V &gt;</code> Class Template Reference | 1075 |
| 10.376. Detailed Description  | 1076 |
| 10.376. Member Typedef Documentation  | 1077 |
| 10.376.2. <code>lvalue_t</code>   | 1077 |
| 10.376. Constructor & Destructor Documentation                                | 1077 |
| 10.376.3. <code>lobject_id</code>   | 1077 |
| 10.377. <code>util::ord&lt; T &gt;</code> Struct Template Reference           | 1077 |
| 10.377. Detailed Description  | 1077 |
| 10.378. <code>util::ord_pair&lt; T &gt;</code> Struct Template Reference      | 1077 |
| 10.378. Detailed Description  | 1078 |
| 10.378. Member Function Documentation   | 1079 |
| 10.378.2. <code>lchange_both</code>   | 1079 |
| 10.378.2.2. <code>change_first</code>   | 1079 |
| 10.378.2.3. <code>change_second</code>  | 1079 |
| 10.378.2.4. <code>first</code>  | 1079 |
| 10.378.2.5. <code>second</code>   | 1079 |
| 10.379. <code>util::pix&lt; I &gt;</code> Struct Template Reference           | 1080 |
| 10.379. Detailed Description  | 1080 |
| 10.379. Member Typedef Documentation  | 1080 |
| 10.379.2. <code>lpsite</code>   | 1080 |
| 10.379.2.2. <code>value</code>  | 1080 |
| 10.379. Constructor & Destructor Documentation                                | 1081 |
| 10.379.3. <code>lpix</code>   | 1081 |
| 10.379. Member Function Documentation   | 1081 |
| 10.379.4. <code>lima</code>   | 1081 |
| 10.379.4.2. <code>p</code>  | 1081 |
| 10.379.4.3. <code>v</code>  | 1081 |
| 10.380. <code>util::set&lt; T &gt;</code> Class Template Reference            | 1081 |



|  |      |
|--|------|
| 10.380. Detailed Description                                 | 1083 |
| 10.380. Member Typedef Documentation                         | 1084 |
| 10.380.2. lbkd_eiter   | 1084 |
| 10.380.2. 2eiter   | 1084 |
| 10.380.2. 3element   | 1084 |
| 10.380.2. 4fwd_eiter   | 1084 |
| 10.380. Constructor & Destructor Documentation               | 1084 |
| 10.380.3. lset   | 1084 |
| 10.380. Member Function Documentation                        | 1084 |
| 10.380.4. lclear   | 1084 |
| 10.380.4. 2first_element                                     | 1084 |
| 10.380.4. 3has   | 1085 |
| 10.380.4. 4insert  | 1085 |
| 10.380.4. 5insert  | 1085 |
| 10.380.4. 6is_empty  | 1086 |
| 10.380.4. 7last_element                                      | 1086 |
| 10.380.4. 8memory_size                                       | 1086 |
| 10.380.4. 9elements  | 1086 |
| 10.380.4. 10operator[]                                       | 1086 |
| 10.380.4. 11remove   | 1087 |
| 10.380.4. 12std_vector                                       | 1087 |
| 10.381. In::util::site_pair< P > Class Template Reference    | 1087 |
| 10.381. Detailed Description                                 | 1088 |
| 10.381. Member Function Documentation                        | 1088 |
| 10.381.2. lfirst   | 1088 |
| 10.381.2. 2pair  | 1089 |
| 10.381.2. 3second  | 1089 |
| 10.382. In::util::soft_heap< T, R > Class Template Reference | 1089 |
| 10.382. Detailed Description                                 | 1090 |
| 10.382. Member Typedef Documentation                         | 1090 |
| 10.382.2. lelement   | 1090 |
| 10.382. Constructor & Destructor Documentation               | 1090 |
| 10.382.3. lsoft_heap   | 1090 |
| 10.382.3. 2~soft_heap  | 1091 |
| 10.382. Member Function Documentation                        | 1091 |
| 10.382.4. lclear   | 1091 |

|            |  |      |
|------------|--|------|
| 10.382.4.2 | is_empty   | 1091 |
| 10.382.4.3 | is_valid   | 1091 |
| 10.382.4.4 | elements   | 1091 |
| 10.382.4.5 | pop_front  | 1091 |
| 10.382.4.6 | push   | 1092 |
| 10.382.4.7 | push   | 1092 |
| 10.383     | util::timer Class Reference                      | 1092 |
| 10.383     | Detailed Description                             | 1093 |
| 10.384     | util::tracked_ptr< T > Struct Template Reference | 1093 |
| 10.384     | Detailed Description                             | 1093 |
| 10.384     | Constructor & Destructor Documentation           | 1094 |
| 10.384.2   | tracked_ptr                                      | 1094 |
| 10.384.2   | tracked_ptr                                      | 1094 |
| 10.384.2.3 | ~tracked_ptr                                     | 1094 |
| 10.384     | Member Function Documentation                    | 1094 |
| 10.384.3   | operator bool                                    | 1094 |
| 10.384.3.2 | operator!  | 1094 |
| 10.384.3.3 | operator->                                       | 1094 |
| 10.384.3.4 | operator->                                       | 1094 |
| 10.384.3.5 | operator=  | 1095 |
| 10.384.3.6 | operator=  | 1095 |
| 10.385     | util::tree< T > Class Template Reference         | 1095 |
| 10.385     | Detailed Description                             | 1096 |
| 10.385     | Constructor & Destructor Documentation           | 1096 |
| 10.385.2   | tree   | 1096 |
| 10.385.2   | tree   | 1096 |
| 10.385     | Member Function Documentation                    | 1096 |
| 10.385.3   | ladd_tree_down                                   | 1096 |
| 10.385.3.2 | add_tree_up                                      | 1096 |
| 10.385.3.3 | check_consistency                                | 1097 |
| 10.385.3.4 | main_branch                                      | 1097 |
| 10.385.3.5 | root   | 1097 |
| 10.386     | util::tree_node< T > Class Template Reference    | 1097 |
| 10.386     | Detailed Description                             | 1098 |
| 10.386     | Constructor & Destructor Documentation           | 1099 |
| 10.386.2   | tree_node  | 1099 |

|             |  |      |
|-------------|--|------|
| 10.386.2    | tree_node                                      | 1099 |
| 10.386.3    | Member Function Documentation                  | 1099 |
| 10.386.3.1  | add_child                                      | 1099 |
| 10.386.3.2  | add_child                                      | 1099 |
| 10.386.3.3  | check_consistency                              | 1100 |
| 10.386.3.4  | children                                       | 1100 |
| 10.386.3.5  | children                                       | 1100 |
| 10.386.3.6  | delete_tree_node                               | 1100 |
| 10.386.3.7  | elt  | 1100 |
| 10.386.3.8  | elt  | 1101 |
| 10.386.3.9  | parent   | 1101 |
| 10.386.3.10 | print  | 1101 |
| 10.386.3.11 | search   | 1101 |
| 10.386.3.12 | search_rec                                     | 1102 |
| 10.386.3.13 | set_parent                                     | 1102 |
| 10.387      | In::util::vertex< G > Class Template Reference | 1102 |
| 10.387.1    | Detailed Description                           | 1104 |
| 10.387.2    | Member Typedef Documentation                   | 1104 |
| 10.387.2.1  | Category                                       | 1104 |
| 10.387.2.2  | graph_t  | 1104 |
| 10.387.2.3  | id_t   | 1104 |
| 10.387.2.4  | id_value_t                                     | 1104 |
| 10.387.3    | Constructor & Destructor Documentation         | 1104 |
| 10.387.3.1  | vertex   | 1104 |
| 10.387.4    | Member Function Documentation                  | 1105 |
| 10.387.4.1  | change_graph                                   | 1105 |
| 10.387.4.2  | edge_with                                      | 1105 |
| 10.387.4.3  | graph  | 1105 |
| 10.387.4.4  | id   | 1105 |
| 10.387.4.5  | invalidate                                     | 1105 |
| 10.387.4.6  | is_valid                                       | 1105 |
| 10.387.4.7  | nth_nbh_edge                                   | 1106 |
| 10.387.4.8  | nth_nbh_vertex                                 | 1106 |
| 10.387.4.9  | nmax_nbh_edges                                 | 1106 |
| 10.387.4.10 | nmax_nbh_vertices                              | 1106 |
| 10.387.4.11 | operator vertex_id_t                           | 1106 |

|  |      |
|--|------|
| 10.387.4.1other  | 1106 |
| 10.387.4.1update_id  | 1106 |
| 10.388.1.1.1.1util::yes Struct Reference                       | 1107 |
| 10.388.1.1.1.1.1.1Detailed Description                         | 1107 |
| 10.389.1.1.1.1.Value< E > Struct Template Reference            | 1107 |
| 10.389.1.1.1.1.1.1Detailed Description                         | 1108 |
| 10.390.1.1.1.1.value::float01 Class Reference                  | 1109 |
| 10.390.1.1.1.1.1.1Detailed Description                         | 1110 |
| 10.390.1.1.1.1.1.2Member Typedef Documentation                 | 1110 |
| 10.390.1.1.1.1.1.1.2.1lenc                                     | 1110 |
| 10.390.1.1.1.1.1.1.2.2equiv                                    | 1110 |
| 10.390.1.1.1.1.1.3Constructor & Destructor Documentation       | 1110 |
| 10.390.1.1.1.1.1.1.3.1float01                                  | 1110 |
| 10.390.1.1.1.1.1.1.3.2float01                                  | 1110 |
| 10.390.1.1.1.1.1.1.3.3float01                                  | 1110 |
| 10.390.1.1.1.1.1.4Member Function Documentation                | 1110 |
| 10.390.1.1.1.1.1.1.4.1Inbits                                   | 1110 |
| 10.390.1.1.1.1.1.1.4.2operator float                           | 1110 |
| 10.390.1.1.1.1.1.1.4.3set_nbits                                | 1111 |
| 10.390.1.1.1.1.1.1.4.4to_nbits                                 | 1111 |
| 10.390.1.1.1.1.1.1.4.5value                                    | 1111 |
| 10.390.1.1.1.1.1.1.4.6value_ind                                | 1111 |
| 10.391.1.1.1.1.value::float01_f Struct Reference               | 1111 |
| 10.391.1.1.1.1.1.1.1Detailed Description                       | 1112 |
| 10.391.1.1.1.1.1.1.2Constructor & Destructor Documentation     | 1112 |
| 10.391.1.1.1.1.1.1.1.2.1float01_f                              | 1112 |
| 10.391.1.1.1.1.1.1.1.2.2float01_f                              | 1112 |
| 10.391.1.1.1.1.1.1.3Member Function Documentation              | 1112 |
| 10.391.1.1.1.1.1.1.1.3.1operator float                         | 1112 |
| 10.391.1.1.1.1.1.1.1.3.2operator=                              | 1112 |
| 10.391.1.1.1.1.1.1.1.3.3value                                  | 1112 |
| 10.392.1.1.1.1.value::graylevel< n > Struct Template Reference | 1112 |
| 10.392.1.1.1.1.1.1.1Detailed Description                       | 1114 |
| 10.392.1.1.1.1.1.1.2Constructor & Destructor Documentation     | 1114 |
| 10.392.1.1.1.1.1.1.1.2.1graylevel                              | 1114 |
| 10.392.1.1.1.1.1.1.1.2.2graylevel                              | 1114 |

|   |      |
|---|------|
| 10.392.2.3graylevel . . . . .                                     | 1114 |
| 10.392.2.4graylevel . . . . .                                     | 1114 |
| 10.392.2.5graylevel . . . . .                                     | 1115 |
| 10.392.3Member Function Documentation . . . . .                   | 1115 |
| 10.392.3.1operator= . . . . .                                     | 1115 |
| 10.392.3.2operator= . . . . .                                     | 1115 |
| 10.392.3.3operator= . . . . .                                     | 1115 |
| 10.392.3.4operator= . . . . .                                     | 1115 |
| 10.392.3.5to_float . . . . .                                      | 1115 |
| 10.392.3.6value . . . . .   | 1115 |
| 10.392.4In::value::graylevel_f Struct Reference . . . . .         | 1116 |
| 10.393.Detailed Description . . . . .                             | 1117 |
| 10.393.Constructor & Destructor Documentation . . . . .           | 1117 |
| 10.393.2.1graylevel_f . . . . .                                   | 1117 |
| 10.393.2.2graylevel_f . . . . .                                   | 1117 |
| 10.393.2.3graylevel_f . . . . .                                   | 1117 |
| 10.393.2.4graylevel_f . . . . .                                   | 1117 |
| 10.393.2.5graylevel_f . . . . .                                   | 1117 |
| 10.393.3Member Function Documentation . . . . .                   | 1117 |
| 10.393.3.1operator graylevel< n > . . . . .                       | 1117 |
| 10.393.3.2operator= . . . . .                                     | 1117 |
| 10.393.3.3operator= . . . . .                                     | 1118 |
| 10.393.3.4operator= . . . . .                                     | 1118 |
| 10.393.3.5operator= . . . . .                                     | 1118 |
| 10.393.3.6value . . . . .   | 1118 |
| 10.392.5In::value::int_s< n > Struct Template Reference . . . . . | 1118 |
| 10.394.Detailed Description . . . . .                             | 1120 |
| 10.394.Constructor & Destructor Documentation . . . . .           | 1120 |
| 10.394.2.1int_s . . . . .   | 1120 |
| 10.394.2.2int_s . . . . .   | 1120 |
| 10.394.2.3int_s . . . . .   | 1120 |
| 10.394.3Member Function Documentation . . . . .                   | 1120 |
| 10.394.3.1operator int . . . . .                                  | 1120 |
| 10.394.3.2operator= . . . . .                                     | 1120 |
| 10.394.4Member Data Documentation . . . . .                       | 1120 |
| 10.394.4.1one . . . . .   | 1120 |

|  |      |
|--|------|
| 10.394.4.2zero   | 1121 |
| 10.395.1In::value::int_u< n > Struct Template Reference      | 1121 |
| 10.395.1.Detailed Description                                | 1122 |
| 10.395.1.Constructor & Destructor Documentation              | 1122 |
| 10.395.2.1int_u  | 1122 |
| 10.395.2.2int_u  | 1122 |
| 10.395.2.3int_u  | 1122 |
| 10.395.3Member Function Documentation                        | 1122 |
| 10.395.3.1next   | 1122 |
| 10.395.3.2operator unsigned                                  | 1122 |
| 10.395.3.3operator-  | 1123 |
| 10.395.3.4operator=  | 1123 |
| 10.396.1In::value::int_u_sat< n > Struct Template Reference  | 1123 |
| 10.396.1.Detailed Description                                | 1124 |
| 10.396.1.Constructor & Destructor Documentation              | 1124 |
| 10.396.2.1int_u_sat  | 1124 |
| 10.396.2.2int_u_sat  | 1124 |
| 10.396.3Member Function Documentation                        | 1124 |
| 10.396.3.1operator int                                       | 1124 |
| 10.396.3.2operator+=   | 1125 |
| 10.396.3.3operator-=   | 1125 |
| 10.396.3.4operator=  | 1125 |
| 10.396.4Member Data Documentation                            | 1125 |
| 10.396.4.1one  | 1125 |
| 10.396.4.2zero   | 1125 |
| 10.397.1In::value::Integer< E > Struct Template Reference    | 1125 |
| 10.397.1.Detailed Description                                | 1126 |
| 10.398.1In::value::Integer< void > Struct Template Reference | 1126 |
| 10.398.1.Detailed Description                                | 1126 |
| 10.399.1In::value::label< n > Struct Template Reference      | 1126 |
| 10.399.1.Detailed Description                                | 1127 |
| 10.399.2Member Typedef Documentation                         | 1127 |
| 10.399.2.1enc  | 1127 |
| 10.399.3Constructor & Destructor Documentation               | 1128 |
| 10.399.3.1label  | 1128 |
| 10.399.3.2label  | 1128 |

|  |      |
|--|------|
| 10.399.3.3label  | 1128 |
| 10.399.4Member Function Documentation                      | 1128 |
| 10.399.4.1next   | 1128 |
| 10.399.4.2operator unsigned                                | 1128 |
| 10.399.4.3operator++                                       | 1128 |
| 10.399.4.4operator--                                       | 1128 |
| 10.399.4.5operator=  | 1129 |
| 10.399.4.6operator=  | 1129 |
| 10.399.4.7prev   | 1129 |
| 10.400nl::value::lut_vec< S, T > Struct Template Reference | 1129 |
| 10.400.1Detailed Description                               | 1131 |
| 10.400.2Member Typedef Documentation                       | 1131 |
| 10.400.2.1bkd_viter  | 1131 |
| 10.400.2.2fwd_viter  | 1131 |
| 10.400.2.3value  | 1131 |
| 10.400.3Constructor & Destructor Documentation             | 1132 |
| 10.400.3.1lut_vec  | 1132 |
| 10.400.3.2lut_vec  | 1132 |
| 10.400.3.3lut_vec  | 1132 |
| 10.400.4Member Function Documentation                      | 1132 |
| 10.400.4.1has  | 1132 |
| 10.400.4.2index_of   | 1132 |
| 10.400.4.3nvalues  | 1132 |
| 10.400.4.4operator[]                                       | 1133 |
| 10.401nl::value::proxy< I > Class Template Reference       | 1133 |
| 10.401.1Detailed Description                               | 1134 |
| 10.401.2Member Typedef Documentation                       | 1134 |
| 10.401.2.1enc  | 1134 |
| 10.401.2.2equiv  | 1134 |
| 10.401.3Constructor & Destructor Documentation             | 1134 |
| 10.401.3.1proxy  | 1134 |
| 10.401.3.2proxy  | 1135 |
| 10.401.3.3~proxy   | 1135 |
| 10.401.4Member Function Documentation                      | 1135 |
| 10.401.4.1operator=  | 1135 |
| 10.401.4.2operator=  | 1135 |

|  |      |
|--|------|
| 10.401.4.3to_value                                       | 1135 |
| 10.401.4.4Inn::value::qt::rgb32 Struct Reference         | 1135 |
| 10.402. Detailed Description                             | 1136 |
| 10.402. Constructor & Destructor Documentation           | 1136 |
| 10.402.2.lrgb32  | 1136 |
| 10.402.2.2rgb32  | 1136 |
| 10.402.2.3rgb32  | 1136 |
| 10.402.2.4rgb32  | 1137 |
| 10.402.3Member Function Documentation                    | 1137 |
| 10.402.3.loperator=                                      | 1137 |
| 10.402.3.2red  | 1137 |
| 10.402.4Member Data Documentation                        | 1137 |
| 10.402.4.lzero   | 1137 |
| 10.401.4.5Inn::value::rgb< n > Struct Template Reference | 1137 |
| 10.403. Detailed Description                             | 1138 |
| 10.403. Constructor & Destructor Documentation           | 1138 |
| 10.403.2.lrgb  | 1138 |
| 10.403.2.2rgb  | 1138 |
| 10.403.2.3rgb  | 1138 |
| 10.403.2.4rgb  | 1138 |
| 10.403.3Member Function Documentation                    | 1139 |
| 10.403.3.loperator=                                      | 1139 |
| 10.403.3.2red  | 1139 |
| 10.403.4Member Data Documentation                        | 1139 |
| 10.403.4.lzero   | 1139 |
| 10.401.4.6Inn::value::set< T > Struct Template Reference | 1139 |
| 10.404. Detailed Description                             | 1139 |
| 10.404.2Member Function Documentation                    | 1140 |
| 10.404.2.lthe  | 1140 |
| 10.401.4.7Inn::value::sign Class Reference               | 1140 |
| 10.405. Detailed Description                             | 1141 |
| 10.405.2Member Typedef Documentation                     | 1141 |
| 10.405.2.lenc  | 1141 |
| 10.405.2.2equiv  | 1141 |
| 10.405.3Constructor & Destructor Documentation           | 1141 |
| 10.405.3.lsign   | 1141 |



|  |      |
|--|------|
| 10.405.3.2sign   | 1141 |
| 10.405.3.3sign   | 1141 |
| 10.405.4Member Function Documentation                          | 1142 |
| 10.405.4.1operator int   | 1142 |
| 10.405.4.2operator=  | 1142 |
| 10.405.5Member Data Documentation                              | 1142 |
| 10.405.5.1one  | 1142 |
| 10.405.5.2zero   | 1142 |
| 10.406In::value::stack_image< n, I > Struct Template Reference | 1142 |
| 10.406.1Detailed Description                                   | 1143 |
| 10.406.2Member Typedef Documentation                           | 1143 |
| 10.406.2.1domain_t   | 1143 |
| 10.406.2.2value  | 1143 |
| 10.406.2.3site   | 1144 |
| 10.406.2.4rvalue   | 1144 |
| 10.406.2.5skeleton   | 1144 |
| 10.406.2.6value  | 1144 |
| 10.406.3Constructor & Destructor Documentation                 | 1144 |
| 10.406.3.1stack_image  | 1144 |
| 10.406.4Member Function Documentation                          | 1144 |
| 10.406.4.1is_valid   | 1144 |
| 10.406.4.2operator()   | 1144 |
| 10.406.4.3operator()   | 1145 |
| 10.407In::value::super_value< sign > Struct Template Reference | 1145 |
| 10.407.1Detailed Description                                   | 1145 |
| 10.408In::value::value_array< T, V > Struct Template Reference | 1145 |
| 10.408.1Detailed Description                                   | 1145 |
| 10.408.2Constructor & Destructor Documentation                 | 1146 |
| 10.408.2.1value_array  | 1146 |
| 10.408.3Member Function Documentation                          | 1146 |
| 10.408.3.1operator()   | 1146 |
| 10.408.3.2operator[]   | 1146 |
| 10.408.3.3vset   | 1146 |
| 10.409In::Value_Iterator< E > Struct Template Reference        | 1146 |
| 10.409.1Detailed Description                                   | 1147 |
| 10.409.2Member Function Documentation                          | 1148 |

|  |      |
|--|------|
| 10.409.2. Inext  | 1148 |
| 10.409. Friends And Related Function Documentation               | 1148 |
| 10.409.3. operator<<   | 1148 |
| 10.410. In::Value_Set< E > Struct Template Reference             | 1148 |
| 10.410. Detailed Description                                     | 1149 |
| 10.411. In::Vertex< E > Struct Template Reference                | 1149 |
| 10.411. Detailed Description                                     | 1149 |
| 10.412. In::vertex_image< P, V, G > Class Template Reference     | 1150 |
| 10.412. Detailed Description                                     | 1151 |
| 10.412. Member Typedef Documentation                             | 1151 |
| 10.412.2. lgraph_t   | 1151 |
| 10.412.2.2. nbh_t  | 1151 |
| 10.412.2.3. site_function_t                                      | 1151 |
| 10.412.2.4. skeleton   | 1151 |
| 10.412.2.5. vertex_nbh_t   | 1151 |
| 10.412.2.6. vertex_win_t   | 1151 |
| 10.412.2.7. win_t  | 1152 |
| 10.412.3. Constructor & Destructor Documentation                 | 1152 |
| 10.412.3. lvertex_image  | 1152 |
| 10.412.4. Member Function Documentation                          | 1152 |
| 10.412.4. operator()   | 1152 |
| 10.413. In::violent_cast_image< T, I > Struct Template Reference | 1152 |
| 10.413. Detailed Description                                     | 1153 |
| 10.413. Member Typedef Documentation                             | 1153 |
| 10.413.2. lvalue   | 1153 |
| 10.413.2.2. rvalue   | 1153 |
| 10.413.2.3. skeleton   | 1153 |
| 10.413.2.4. value  | 1153 |
| 10.413.3. Constructor & Destructor Documentation                 | 1154 |
| 10.413.3. lviolent_cast_image                                    | 1154 |
| 10.413.4. Member Function Documentation                          | 1154 |
| 10.413.4. operator()   | 1154 |
| 10.413.4.2. operator()   | 1154 |
| 10.414. In::w_window< D, W > Struct Template Reference           | 1154 |
| 10.414. Detailed Description                                     | 1155 |
| 10.414. Member Typedef Documentation                             | 1156 |

|   |      |
|---|------|
| 10.414.2.1bkd_qiter . . . . .                                       | 1156 |
| 10.414.2.2dpsite . . . . .  | 1156 |
| 10.414.2.3fwd_qiter . . . . .                                       | 1156 |
| 10.414.2.4weight . . . . .  | 1156 |
| 10.414.3Constructor & Destructor Documentation . . . . .            | 1156 |
| 10.414.3.1w_window . . . . .  | 1156 |
| 10.414.4Member Function Documentation . . . . .                     | 1156 |
| 10.414.4.1clear . . . . .   | 1156 |
| 10.414.4.2insert . . . . .  | 1156 |
| 10.414.4.3is_symmetric . . . . .                                    | 1157 |
| 10.414.4.4std_vector . . . . .                                      | 1157 |
| 10.414.4.5sym . . . . .   | 1157 |
| 10.414.4.6w . . . . .   | 1157 |
| 10.414.4.7weights . . . . .   | 1157 |
| 10.414.4.8win . . . . .   | 1157 |
| 10.414.5Friends And Related Function Documentation . . . . .        | 1158 |
| 10.414.5.1operator<< . . . . .                                      | 1158 |
| 10.414.5.2operator== . . . . .                                      | 1158 |
| 10.415mln::Weighted_Window< E > Struct Template Reference . . . . . | 1158 |
| 10.415.1Detailed Description . . . . .                              | 1159 |
| 10.415.2Friends And Related Function Documentation . . . . .        | 1159 |
| 10.415.2.1operator- . . . . .                                       | 1159 |
| 10.416mln::win::backdiag2d Struct Reference . . . . .               | 1159 |
| 10.416.1Detailed Description . . . . .                              | 1159 |
| 10.416.2Constructor & Destructor Documentation . . . . .            | 1160 |
| 10.416.2.1backdiag2d . . . . .                                      | 1160 |
| 10.416.3Member Function Documentation . . . . .                     | 1160 |
| 10.416.3.1length . . . . .  | 1160 |
| 10.417mln::win::ball< G, C > Struct Template Reference . . . . .    | 1160 |
| 10.417.1Detailed Description . . . . .                              | 1160 |
| 10.417.2Constructor & Destructor Documentation . . . . .            | 1161 |
| 10.417.2.1ball . . . . .  | 1161 |
| 10.417.3Member Function Documentation . . . . .                     | 1161 |
| 10.417.3.1diameter . . . . .  | 1161 |
| 10.418mln::win::cube3d Struct Reference . . . . .                   | 1161 |
| 10.418.1Detailed Description . . . . .                              | 1161 |

|          |   |      |
|----------|---|------|
| 10.418.1 | Constructor & Destructor Documentation                      | 1162 |
| 10.418.2 | 1cube3d   | 1162 |
| 10.418.3 | Member Function Documentation                               | 1162 |
| 10.418.3 | length  | 1162 |
| 10.419   | mln::win::cuboid3d Struct Reference                         | 1162 |
| 10.419   | Detailed Description  | 1163 |
| 10.419   | Constructor & Destructor Documentation                      | 1163 |
| 10.419.2 | 1cuboid3d   | 1163 |
| 10.419.3 | Member Function Documentation                               | 1164 |
| 10.419.3 | depth   | 1164 |
| 10.419.3 | height  | 1164 |
| 10.419.3 | volume  | 1164 |
| 10.419.3 | width   | 1164 |
| 10.420   | mln::win::diag2d Struct Reference                           | 1164 |
| 10.420   | Detailed Description  | 1164 |
| 10.420   | Constructor & Destructor Documentation                      | 1165 |
| 10.420.2 | 1diag2d   | 1165 |
| 10.420.3 | Member Function Documentation                               | 1165 |
| 10.420.3 | length  | 1165 |
| 10.421   | mln::win::line< M, i, C > Struct Template Reference         | 1165 |
| 10.421   | Detailed Description  | 1166 |
| 10.421   | Member Enumeration Documentation                            | 1166 |
| 10.421.2 | 1"@87   | 1166 |
| 10.421   | Constructor & Destructor Documentation                      | 1166 |
| 10.421.3 | iline   | 1166 |
| 10.421.4 | Member Function Documentation                               | 1167 |
| 10.421.4 | length  | 1167 |
| 10.421.4 | size  | 1167 |
| 10.422   | mln::win::multiple< W, F > Class Template Reference         | 1167 |
| 10.422   | Detailed Description  | 1167 |
| 10.423   | mln::win::multiple_size< n, W, F > Class Template Reference | 1167 |
| 10.423   | Detailed Description  | 1167 |
| 10.424   | mln::win::octagon2d Struct Reference                        | 1168 |
| 10.424   | Detailed Description  | 1168 |
| 10.424   | Constructor & Destructor Documentation                      | 1168 |
| 10.424.2 | 1octagon2d  | 1168 |

|             |  |      |
|-------------|--|------|
| 10.424.3    | Member Function Documentation              | 1169 |
| 10.424.3.1  | area                                       | 1169 |
| 10.424.3.2  | length                                     | 1169 |
| 10.425      | mln::win::rectangle2d Struct Reference     | 1169 |
| 10.425.1    | Detailed Description                       | 1169 |
| 10.425.2    | Constructor & Destructor Documentation     | 1170 |
| 10.425.2.1  | rectangle2d                                | 1170 |
| 10.425.3    | Member Function Documentation              | 1170 |
| 10.425.3.1  | area                                       | 1170 |
| 10.425.3.2  | height                                     | 1170 |
| 10.425.3.3  | std_vector                                 | 1170 |
| 10.425.3.4  | width                                      | 1170 |
| 10.426      | mln::Window< E > Struct Template Reference | 1170 |
| 10.426.1    | Detailed Description                       | 1171 |
| 10.427      | mln::window< D > Class Template Reference  | 1171 |
| 10.427.1    | Detailed Description                       | 1173 |
| 10.427.2    | Member Typedef Documentation               | 1173 |
| 10.427.2.1  | l_bkd_qiter                                | 1173 |
| 10.427.2.2  | l_fwd_qiter                                | 1173 |
| 10.427.2.3  | qiter                                      | 1173 |
| 10.427.2.4  | regular                                    | 1173 |
| 10.427.3    | Constructor & Destructor Documentation     | 1173 |
| 10.427.3.1  | lwindow                                    | 1173 |
| 10.427.4    | Member Function Documentation              | 1174 |
| 10.427.4.1  | lclear                                     | 1174 |
| 10.427.4.2  | ldelta                                     | 1174 |
| 10.427.4.3  | ldp  | 1174 |
| 10.427.4.4  | lhas                                       | 1174 |
| 10.427.4.5  | linsert                                    | 1174 |
| 10.427.4.6  | linsert                                    | 1174 |
| 10.427.4.7  | linsert                                    | 1175 |
| 10.427.4.8  | l_is_centered                              | 1175 |
| 10.427.4.9  | l_is_empty                                 | 1175 |
| 10.427.4.10 | l_is_symmetric                             | 1175 |
| 10.427.4.11 | lprint                                     | 1175 |
| 10.427.4.12 | lsize                                      | 1175 |

---

|  |      |
|--|------|
| 10.427.4.13d_vector  | 1176 |
| 10.427.4.14m   | 1176 |
| 10.427.5.Friends And Related Function Documentation                            | 1176 |
| 10.427.5.1operator==   | 1176 |
| 10.428.mln::world::inter_pixel::is_separator Struct Reference                  | 1176 |
| 10.428.Detailed Description  | 1177 |
| 10.429.trait::graph< I > Struct Template Reference                             | 1177 |
| 10.429.Detailed Description  | 1178 |
| 10.430.trait::graph< mln::complex_image< 1, G, V > > Struct Template Reference | 1178 |
| 10.430.Detailed Description  | 1178 |
| 10.431.trait::graph< mln::image2d< T > > Struct Template Reference             | 1178 |
| 10.431.Detailed Description  | 1178 |

# Chapter 1

## Documentation of milena

### 1.1 Introduction

This is the documentation of Milena.

### 1.2 Overview of Milena.

- [mln](#)
- [mln::accu](#)
- [mln::algebra](#)
- [mln::arith](#)
- [mln::binarization](#)
- [mln::border](#)
- [mln::canvas](#)
- [mln::convert](#)
- [mln::data](#)
- [mln::debug](#)
- [mln::display](#)
- [mln::draw](#)
- [mln::estim](#)
- [mln::extension](#)
- [mln::fun](#)
- [mln::geom](#)
- [mln::graph](#)
- [mln::histo](#)

- [mln::io](#)
- [mln::labeling](#)
- [mln::data](#)
- [mln::linear](#)
- [mln::literal](#)
- [mln::logical](#)
- [mln::make](#)
- [mln::math](#)
- [mln::metal](#)
- [mln::morpho](#)
- [mln::norm](#)
- [mln::opt](#)
- [mln::pw](#)
- [mln::registration](#)
- [mln::set](#)
- [mln::tag](#)
- [mln::test](#)
- [mln::topo](#)
- [mln::trace](#)
- [mln::trait](#)
- [mln::transform](#)
- [mln::util](#)
- [mln::value](#)
- [mln::win](#)

### 1.3 Copyright and License.

Copyright (C) 2007, 2008, 2009, 2010, 2011 EPITA Research and Development (LRDE)

This documentation is part of Olena.

Olena is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, version 2 of the License.

Olena is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with Olena. If not, see <http://www.gnu.org/licenses/>.



## Chapter 2

# Quick Reference Guide

- [Installation](#)
- [Foreword](#)
- [Site](#)
- [Site set](#)
- [Image](#)
- [Structural elements: Window and neighborhood](#)
- [Sites, psites and dpoints](#)
- [Iterators](#)
- [Memory management](#)
- [Basic routines](#)
- [Input / Output](#)
- [Graphs and images](#)
- [Useful global variables](#)
- [Useful macros](#)
- [Common Compilation Errors](#)

## 2.1 Installation

### 2.1.1 Requirements

2.1.1.1 To compile the user examples

2.1.1.2 To compile the documentation (Optional)

2.1.1.3 To develop in Olena

### 2.1.2 Getting Olena

### 2.1.3 Building Olena

## 2.2 Foreword

### 2.2.1 Generality

### 2.2.2 Directory hierarchy

### 2.2.3 Writing and compiling a program with Olena

## 2.3 Site

## 2.4 Site set

### Iterators

#### 2.4.1 Basic interface

#### 2.4.2 Optional interface

```
box2d b(2,3);

// The bbox can be retrived in constant time.
std::cout << b.bbox() << std::endl;

// nsites can be retrieved in constant time.
std::cout << "nsites = " << b.nsites() << std::endl;

[(0,0)..(1,2)]
nsites = 6

p_array<point2d> arr;
arr.insert(point2d(1,0));
arr.insert(point2d(1,1));

// The bbox is computed thanks to bbox() algorithm.
box2d box = geom::bbox(arr);
std::cout << box << std::endl;

// p_array provides nsites(),
```

```
// it can be retrieved in constant time.
std::cout << "nsites = " << arr.nsites() << std::endl;

[(1,0)..(1,1)]
nsites = 2
```

## 2.5 Image

### 2.5.1 Definition

### 2.5.2 Possible image types

### 2.5.3 Possible value types

### 2.5.4 Domain

```
// Define a box2d from (-2,-3) to (3,5).
box2d b = make::box2d(-2,-3, 3,5);
// Initialize an image with b as domain.
image2d<int> ima(b);

std::cout << "b = " << b << std::endl;
std::cout << "domain = " << ima.domain() << std::endl;

b = [(-2,-3)..(3,5)]
domain = [(-2,-3)..(3,5)]

// Create an image on a 2D box
// with 10 columns and 10 rows.
image2d<bool> ima(make::box2d(10, 10));

mIn_site_(image2d<bool>) p1(20, 20);
mIn_site_(image2d<bool>) p2(3, 3);

std::cout << "has(p1)? "
  << (ima.has(p1) ? "true" : "false")
  << std::endl;

std::cout << "has(p2)? "
  << (ima.has(p2) ? "true" : "false")
  << std::endl;

has(p1)? false
has(p2)? true

point2d p(9,9);

// At (9, 9), both values change.
ima1(p) = 'M';
ima2(p) = 'W';

bool b = (ima1(p) == ima2(p));
std::cout << (b ? "True" : "False") << std::endl;

False
```

## 2.5.5 Border and extension

### 2.5.5.1 Image border

```

bool vals[3][3] = { { 0, 1, 1 },
                   { 1, 0, 0 },
                   { 1, 1, 0 } };

image2d<bool> ima_def = make::image(vals);
border::fill(ima_def, false);
debug::println_with_border(ima_def);

std::cout << "======" << std::endl << std::endl;

border::thickness = 0;
image2d<bool> ima_bt0 = make::image(vals);
debug::println_with_border(ima_bt0);

```

```

- - - - -
- - - - -
- - - - -
- - - | | - -
- - | - - - -
- - | | - - -
- - - - -
- - - - -
- - - - -

```

```

=====

```

```

- | |
| - -
| | -

```

### 2.5.5.2 Generality on image extension

imamorphed

### 2.5.5.3 Different extensions

```

image2d<rgb8> lena;
io::ppm::load(lena, MLN_IMG_DIR "/small.ppm");
box2d bbox_enlarged = lena.domain();
bbox_enlarged.enlarge(border::thickness);
mln_VAR(ima_roi, lena | fun::p2b::big_chess<box2d>(lena.domain(), 10));

```

#### 2.5.5.3.1 Extension with a value

```

mln_VAR(ext_with_val, extended_to(extend(ima_roi, literal::blue), bbox_enlarged
));

```

#### 2.5.5.3.2 Extension with a function

```

namespace mln
{
    struct my_ext : public Function_v2v<my_ext>
    {

```

```

typedef value::rgb8 result;

value::rgb8 operator()(const point2d& p) const
{
    if ((p.row() + p.col()) % 20)
        return literal::black;
    return literal::white;
}

};

} // end of namespace mln

mln_VAR(ext_with_fun, extended_to(extend(ima_roi, my_ext()), bbox_enlarged));

```

### 2.5.5.3.3 Extension with an image

```

mln_VAR(ext_with_ima, extend(ima_roi, lena));

// Default border size is set to 0.

// Image defined on a box2d from
// (0, 0) to (2, 2)
image2d<int> ima1(2, 3);

std::cout << "ima1.has(0, 0) : "
            << ima1.has(point2d(0, 0)) << std::endl;

std::cout << "ima1.has(-3, 0) : "
            << ima1.has(point2d(-3, 0)) << std::endl;

std::cout << "ima1.has(2, 5) : "
            << ima1.has(point2d(2, 5)) << std::endl;

std::cout << "======" << std::endl;

// Set default border size to 0.
border::thickness = 0;

// Image defined on a box2d from
// (0, 0) to (2, 2)
image2d<int> ima2(2, 3);

std::cout << "ima2.has(0, 0) : "
            << ima2.has(point2d(0, 0)) << std::endl;

std::cout << "ima2.has(-3, 0) : "
            << ima2.has(point2d(-3, 0)) << std::endl;

std::cout << "ima2.has(2, 5) : "
            << ima2.has(point2d(2, 5)) << std::endl;

ima1.has(0, 0) : 1
ima1.has(-3, 0) : 1
ima1.has(2, 5) : 1
====
ima2.has(0, 0) : 1
ima2.has(-3, 0) : 0
ima2.has(2, 5) : 0

border::thickness = 30;

// Declare the image to be rotated.

```

```

image2d<value::rgb8> ima1(220, 220);
data::fill(ima1, literal::cyan);
border::fill(ima1, literal::yellow);
// Set an infinite extension.
mln_VAR(ima1, extend(ima1, pw::cst(literal::yellow)));

// Declare the output image.
image2d<value::rgb8> ima2(220, 220);
data::fill(ima2, literal::cyan);
border::fill(ima2, literal::yellow);

box2d extended_domain= ima1.domain();
extended_domain.enlarge(border::thickness);

// Draw the domain bounding box
draw::box(ima1, geom::bbox(ima1), literal::red);
// Save the image, including its border.
doc::ppmsave(ima1 | extended_domain, "ima2d-rot");

// Define and apply a point-wise rotation
fun::x2x::rotation<2,float> rot1(0.5, literal::zero);
image2d<value::rgb8>::fwd_piter p(ima1.domain());
for_all(p)
{
    algebra::vec<2,float> pv = p.to_site().to_vec();
    algebra::vec<2,float> v = rot1.inv()(pv);
    ima2(p) = ima1(v);
}

draw::box(ima2, ima2.bbox(), literal::red);
doc::ppmsave(extended_to(ima2, extended_domain), "ima2d-rot");

my_routine(ima | ima.domain());

```

## 2.5.6 Interface

### 2.5.7 Load and save images

```

image2d<bool> ima;
io::pbm::load(ima, MLN_DOC_DIR "/img/small.pbm");

io::pbm::save(ima, MLN_DOC_DIR "/figures/ima_save.pbm");

```

### 2.5.8 Create an image

```

// Build an empty image;
image2d<value::int_u8> imgl1;

// Build an image with 2 rows
// and 3 columns sites
image2d<value::int_u8> imglb(box2d(2, 3));
image2d<value::int_u8> imglc(2, 3);

bool vals[6][5] = {
    {0, 1, 1, 0, 0},
    {0, 1, 1, 0, 0},
    {0, 0, 0, 0, 0},
    {1, 1, 0, 1, 0},
    {1, 0, 1, 1, 1},
    {1, 0, 0, 0, 0}
};
image2d<bool> ima = make::image(vals);

```

```

image2d<value::int_u8> img2a(2, 3);
image2d<value::int_u8> img2b;

initialize(img2b, img2a);
data::fill(img2b, img2a);

```

## Fill

### 2.5.9 Access and modify values

```

box2d b(2,3);
image2d<value::int_u8> ima(b);

// On image2d, Site <=> point2d
point2d p(1, 2);

// Associate '9' as value for the site/point2d (1,2).
// The value is returned by reference and can be changed.
opt::at(ima, 1,2) = 9;
std::cout << "opt::at(ima, 1,2) = " << opt::at(ima, 1,2)
          << std::endl;
std::cout << "ima(p) = " << ima(p) << std::endl;

std::cout << "---" << std::endl;

// Associate '2' as value for the site/point2d (1,2).
// The value is returned by reference
// and can be changed as well.
ima(p) = 2;
std::cout << "opt::at(ima, 1,2) = " << opt::at(ima, 1,2)
          << std::endl;
std::cout << "ima(p) = " << ima(p) << std::endl;

opt::at(ima, 1,2) = 9
ima(p) = 9
---
opt::at(ima, 1,2) = 2
ima(p) = 2

```

## Iterators

### 2.5.10 Image size

```

image2d<int> ima(make::box2d(0,0, 10,12));

std::cout << "nrows = " << ima.nrows()
          << " - "
          << "ncols = " << ima.ncols()
          << std::endl;

nrows = 11 - ncols = 13

```

## 2.6 Structural elements: Window and neighborhood

### 2.6.1 Define an element

#### 2.6.1.1 Window

#### 2.6.1.2 Neighborhood

```
label_8 nlabels;
image2d<label_8> lbl = labeling::blobs(ima, c4(), nlabels);
```

#### 2.6.1.3 Custom structural elements

```
window2d win;
win.insert(-1, -1);
win.insert(-1, 0);
win.insert(-1, 1);
```

- o -
- o X
- o -

```
bool b[9] = { 1, 0, 0,
             1, 0, 0,
             1, 0, 0 };
```

```
bool b2[3][3] = { { 1, 0, 0 },
                  { 1, 0, 0 },
                  { 1, 0, 0 } };
```

```
window2d win = convert::to<window2d>(b);
window2d win2 = convert::to<window2d>(b2);
```

#### 2.6.1.4 Conversion between Neighborhoods and Windows

## 2.7 Sites, psites and dpoints

### 2.7.1 Need for site

```
c 0 1 2 3
r
+---+---+
0 | |x| | |
+---+---+
1 | | | | |
+---+---+
```

### 2.7.2 Need for psite

```
unsigned my_values(const mln::point2d& p)
{
    if (p.row() == 0)
        return 8;
    return 9;
}
```



```

p_array<point2d> arr;
arr.append(point2d(3, 6));
arr.append(point2d(3, 7));
arr.append(point2d(3, 8));
arr.append(point2d(4, 8));
arr.append(point2d(4, 9));

mln_VAR(ima, my_values | arr);

c  6 7 8 9
r
  +---+
3  | |x| |
  +---+
4     | | |
     +---+

arr[] = 0 1 2 3 4
        +---+
        | |x| | | |
        +---+

```

### 2.7.3 From psite to site

### 2.7.4 Dpoint

```

dpoint2d dp(-1,0);
point2d p(1,1);

std::cout << p + dp << std::endl;

(0,1)

```

## 2.8 Iterators

```

box2d b(3, 2);
mln_piter_(box2d) p(b);

for_all(p)
  std::cout << p; //prints every site coordinates.

(0,0) (0,1) (1,0) (1,1) (2,0) (2,1)

template <typename I>
void fill(I& ima, mln_value(I) v)
{
  mln_piter(I) p(ima.domain());
  for_all(p)
    ima(p) = v;
}

template <typename I, typename J>
void paste(const I& data, J& dest)
{
  mln_piter(I) p(data.domain());
  for_all(p)
    dest(p) = data(p);
}

```

### Useful macros

## 2.9 Memory management

```

image2d<int> ima1(box2d(2, 3));
image2d<int> ima2;
point2d p(1,2);

ima2 = ima1; // ima1.id() == ima2.id()
// and both point to the same memory area.

ima2(p) = 2; // ima1 is modified as well.

// prints "2 - 2"
std::cout << ima2(p) << " - " << ima1(p) << std::endl;
// prints "true"
std::cout << (ima2.id_() == ima1.id_()) << std::endl;

image2d<int> ima1(5, 5);
image2d<int> ima3 = duplicate(ima1); // Makes a deep copy.

point2d p(2, 2);
ima3(p) = 3;

std::cout << ima3(p) << " - " << ima1(p) << std::endl;
std::cout << (ima3.id_() == ima1.id_()) << std::endl;

3 - 0
0

```

## 2.10 Basic routines

### 2.10.1 Fill

```

image2d<char> imga(5, 5);

data::fill(imga, 'a');

data::fill((imga | box2d(1,2)).rw(), 'a');

```

### 2.10.2 Paste

```

image2d<unsigned char> imgb(make::box2d(5,5, 7,8));
// Initialize imga with the same domain as imgb.
image2d<unsigned char> imga(imgb.domain());

// Initialize the image values.
data::fill(imgb, 'b');

// Paste the content of imgb in imga.
data::paste(imgb, imga);

debug::println(imga);

98 98 98 98
98 98 98 98
98 98 98 98

image2d<int> ima1(5, 5);

```

```

image2d<int> ima2(10, 10);

std::cout << "ima1.domain() = " << ima1.domain()
<< std::endl;
std::cout << "ima2.domain() = " << ima2.domain()
<< std::endl;

image2d<int> ima1(5, 5);
image2d<int> ima2(10, 10);

std::cout << "ima1.domain() = " << ima1.domain()
<< std::endl;
std::cout << "ima2.domain() = " << ima2.domain()
<< std::endl;

```

### 2.10.3 Blobs

```

bool vals[6][5] = {
    {0, 1, 1, 0, 0},
    {0, 1, 1, 0, 0},
    {0, 0, 0, 0, 0},
    {1, 1, 0, 1, 0},
    {1, 0, 1, 1, 1},
    {1, 0, 0, 0, 0}
};
image2d<bool> ima = make::image(vals);

label_8 nlabels;
image2d<label_8> lbl = labeling::blobs(ima, c4(), nlabels);

```

### 2.10.4 Logical not

```

bool vals[5][5] = {
    {1, 0, 1, 0, 0},
    {0, 1, 0, 1, 0},
    {1, 0, 1, 0, 0},
    {0, 1, 0, 1, 0},
    {0, 1, 0, 1, 0}
};
image2d<bool> ima = make::image(vals);

image2d<bool> ima_neg = logical::not_(ima);

logical::not_inplace(ima);

```

### 2.10.5 Compute

#### 2.10.5.1 Accumulators

```

data::compute(accu::meta::stat::max, ima);

data::compute(accu::meta::stat::max(), ima);

```

#### 2.10.5.2 Example with labeling::compute()

```

bool vals[6][5] = {
    {0, 1, 1, 0, 0},

```

```

    {0, 1, 1, 0, 0},
    {0, 0, 0, 0, 0},
    {1, 1, 0, 1, 0},
    {1, 0, 1, 1, 1},
    {1, 0, 0, 0, 0}
};
image2d<bool> ima = make::image(vals);

label_8 nlabels;
image2d<label_8> lbl = labeling::blobs(ima, c4(), nlabels);

util::array<box2d> boxes =
    labeling::compute(accu::meta::shape::bbox(),
                     lbl,
                     nlabels);

for (unsigned i = 1; i <= nlabels; ++i)
    std::cout << boxes[i] << std::endl;

[(0,1)..(1,2)]
[(3,0)..(5,1)]
[(3,2)..(4,4)]

unsigned nsites = geom::nsites(ima);

```

## 2.10.6 Working with parts of an image

```

//function_p2b
bool my_function_p2b(mln::point2d p);

//function_p2v
//V is the value type used in the image.
template <typename V>
V my_function_p2v(mln::point2d p);

bool vals[6][5] = {
    {0, 1, 1, 0, 0},
    {0, 1, 1, 0, 0},
    {0, 0, 0, 0, 0},
    {1, 1, 0, 1, 0},
    {1, 0, 1, 1, 1},
    {1, 0, 0, 0, 0}
};
image2d<bool> ima = make::image(vals);

```

### 2.10.6.1 Restrict an image with a site set

```

p_array<point2d> arr;

// We add two points in the array.
arr.append(point2d(0, 1));
arr.append(point2d(4, 0));

// We restrict the image to the sites
// contained in arr and fill these ones
// with 0.
// We must call "rw()" here.
data::fill((ima | arr).rw(), 0);

debug::println((ima | arr));

```

```

mln_VAR(ima2, ima | arr);
// We do not need to call "rw()" here.
data::fill(ima2, 0);

```

-

-

-

-

### 2.10.6.2 Restrict an image with a predicate

```

label_8 nlabels;
image2d<label_8> lbl = labeling::blobs(ima, c4(), nlabels);

mln_VAR(lbl_2, lbl | (pw::value(lbl) == pw::cst(2u)));

image2d<rgb8> ima2;
initialize(ima2, ima);
data::fill(ima2, literal::black);

data::fill((ima2 | lbl_2.domain()).rw(), literal::red);

label_8 nlabels;
image2d<label_8> lab = labeling::blobs(ima, c4(), nlabels);

image2d<rgb8> ima2;
initialize(ima2, ima);
data::fill(ima2, literal::black);

data::fill((ima2 | (pw::value(lab) == pw::cst(2u))).rw(), literal::red);

```

### 2.10.6.3 Restrict an image with a C function

```

bool row_oddity(mln::point2d p)
{
    return p.row() % 2;
}

image2d<rgb8> ima2;
initialize(ima2, ima);
data::fill(ima2, literal::black);

data::fill((ima2 | row_oddity).rw(), literal::red);

ima | sub_D

0 1 0
1 1 1

mln_VAR(imab1, ima | (pw::value(ima) == pw::cst(1u)));

```

```

1
1 1 1

box2d b1(1,0, 1, 2);
mln_VAR(imac, imab1 | b1);

// Print:
// 1 1 1
debug::println(imac);

box2d b2(0,0, 1, 1);
// Will fail at runtime.
// ima.domain().has((0,0)) is false.
mln_VAR(imad, imab1 | b2);
debug::println(imad);

ima / sub_D

```

## 2.11 Input / Output

### 2.11.1 ImageMagick

### 2.11.2 GDCM

## 2.12 Graphs and images

### 2.12.1 Description

### 2.12.2 Example

```

      0 1 2 3 4
    .-----
0 | 0       2
1 |   \   / |
2 |     1   |
3 |       \ |
4 |         3-4

util::graph g;

for (unsigned i = 0; i < 5; ++i)
    g.add_vertex(); // Add vertex 'i';

g.add_edge(0, 1); // Associated to edge 0.
g.add_edge(1, 2); // Associated to edge 1.
g.add_edge(1, 3); // Associated to edge 2.
g.add_edge(3, 4); // Associated to edge 3.
g.add_edge(4, 2); // Associated to edge 4.

typedef fun::i2v::array<point2d> F;
F f(5); // We need to map 5 vertices.
f(0) = point2d(0, 0);
f(1) = point2d(2, 2);
f(2) = point2d(0, 4);
f(3) = point2d(4, 3);
f(4) = point2d(4, 4);

```

```

typedef p_vertices<util::graph, F> pv_t;
pv_t pv(g, f);

template <typename S>
struct viota_t : public mln::Function_v2v< viota_t<S> >
{
    typedef unsigned result;

    viota_t(unsigned size)
    {
        v_.resize(size);
        for(unsigned i = 0; i < size; ++i)
            v_[i] = 10 + i;
    }

    unsigned
    operator()(const mln_psite(S)& p) const
    {
        return v_[p.v().id()];
    }

protected:
    std::vector<result> v_;
};

// Constructs an image
viota_t<pv_t> viota(pv.nsites());
mln_VAR(graph_vertices_ima, viota | pv);

//Prints each vertex and its associated data.
mln_piter_(graph_vertices_ima_t) p(graph_vertices_ima.domain());
for_all(p)
    std::cout << "graph_vertices_ima(" << p << ") = "
                << graph_vertices_ima(p) << std::endl;

graph_vertices_ima((0,0)) = 10
graph_vertices_ima((2,2)) = 11
graph_vertices_ima((0,4)) = 12
graph_vertices_ima((4,3)) = 13
graph_vertices_ima((4,4)) = 14

// Function which maps sites to data.
viota_t viota(g.v_nmax());

// Iterator on vertices.
mln_vertex_iter_(util::graph) v(g);

// Prints each vertex and its associated value.
for_all(v)
    std::cout << v << " : " << viota(v) << std::endl;

0 : 10
1 : 11
2 : 12
3 : 13
4 : 14

// Iterator on vertices.
mln_vertex_iter_(util::graph) v(g);

// Iterator on v's edges.
mln_vertex_nbh_edge_iter_(util::graph) e(v);

```

```

// Prints the graph
// List all edges for each vertex.
for_all(v)
{
    std::cout << v << " : ";
    for_all(e)
        std::cout << e << " ";
    std::cout << std::endl;
}

0 : (0,1)
1 : (0,1) (1,2) (1,3)
2 : (1,2) (2,4)
3 : (1,3) (3,4)
4 : (3,4) (2,4)

// Iterator on edges.
mIn_edge_iter_(util::graph) e(g);

// Iterator on edges adjacent to e.
mIn_edge_nbh_edge_iter_(util::graph) ne(e);

// Prints the graph
// List all adjacent edges for each edge.
for_all(e)
{
    std::cout << e << " : ";
    for_all(ne)
        std::cout << ne << " ";
    std::cout << std::endl;
}

(0,1) : (1,2) (1,3)
(1,2) : (0,1) (1,3) (2,4)
(1,3) : (0,1) (1,2) (3,4)
(3,4) : (1,3) (2,4)
(2,4) : (1,2) (3,4)

// Iterator on vertices.
mIn_vertex_iter_(util::graph) v(g);

// Iterator on vertices adjacent to v.
mIn_vertex_nbh_vertex_iter_(util::graph) nv(v);

// Prints the graph
// List all adjacent edges for each edge.
for_all(v)
{
    std::cout << v << " : ";
    for_all(nv)
        std::cout << nv << " ";
    std::cout << std::endl;
}

0 : 1
1 : 0 2 3
2 : 1 4
3 : 1 4
4 : 3 2

```



## **2.13 Useful global variables**

## **2.14 Useful macros**

### **2.14.1 Variable declaration macros**

### **2.14.2 Iterator type macros**

#### **2.14.2.1 Default iterator types**

#### **2.14.2.2 Forward iterator types**

#### **2.14.2.3 Backward iterators**

#### **2.14.2.4 Graph iterators**

## **2.15 Common Compilation Errors**



# Chapter 3

## Tutorial

- [Welcome](#)
- [Installation](#)
- [Getting started with Milena](#)
- [Data representation](#)
- [Load and save images](#)
- [Create your first image](#)
- [Read and write images](#)
- [Regions of interest](#)

### 3.1 Welcome

#### 3.1.1 How to learn Milena

#### 3.1.2 Obtaining the library

#### 3.1.3 Downloading the library

##### 3.1.3.1 Downloading from SVN

[Installation](#) [Directory structure](#) [Join the mailing lists](#)

##### 3.1.3.2 Downloading packaged releases

[Installation](#) [Directory structure](#) [Join the mailing lists](#) [Documentation](#)

### **3.1.4 Join the mailing lists**

### **3.1.5 Directory structure**

### **3.1.6 Documentation**

### **3.1.7 Community and Support**

[Join the mailing lists](#) [Contacts](#)

### **3.1.8 Project status**

[Join the mailing lists](#)

### **3.1.9 A brief history of Milena**

### **3.1.10 Contacts**

[Installation](#)

## **3.2 Installation**

### **3.2.1 Bootstrap (SVN Sources)**

[Configure](#)

### **3.2.2 Configure**

### **3.2.3 Install**

[Optional compilation](#) [Installation content](#)

### **3.2.4 Optional compilation**

#### **3.2.4.1 Examples**

#### **3.2.4.2 Tools**

#### **3.2.4.3 Tests**

### **3.2.5 Installation content**

[Welcome](#) [Getting started with Milena](#)

## 3.3 Getting started with Milena

### 3.3.1 Getting familiar with genericity

```
// Java or C -like code

void fill(image *ima, unsigned char v)
{
    for (int i = 0; i < ima->nrows; ++i)
        for (int j = 0; j < ima->ncols; ++j)
            ima->data[i][j] = v;
}

template <typename I>
void fill(I& ima, mln_value(I) v)
{
    mln_piter(I) p(ima.domain());
    for_all(p)
        ima(p) = v;
}

fill(ima, literal::green);

box2d b(20,20);
fill((ima | b).rw(), literal::green);
```

### 3.3.2 First generic algorithm

```
#include <mln/core/image/image2d.hh>
#include <mln/core/image/dmorph/image_if.hh>
#include <mln/core/alias/neighb2d.hh>

#include <mln/data/fill.hh>

#include <mln/labeling/blobs.hh>
#include <mln/labeling/compute.hh>
#include <mln/labeling/blobs.hh>

#include <mln/data/compare.hh>

#include <mln/util/array.hh>

#include <mln/value/label_8.hh>

#include <mln/accu/math/count.hh>

#include <mln/pw/all.hh>

#include <tests/data.hh>
#include <doc/tools/sample_utils.hh>

namespace mln
{
    template <typename I, typename N>
    mln_concrete(I)
    my_algorithm(const Image<I>& ima_,
                const Neighborhood<N>& nbh_)
    {
        trace::entering("my_algorithm");

        const I& ima = exact(ima_);
        const N& nbh = exact(nbh_);
```

```

mln_precondition(ima.is_valid());
mln_precondition(nbh.is_valid());

typedef value::label_8 V;
V nlabels;
mln_ch_value(I,V) lbl = labeling::blobs(ima, nbh, nlabels);
util::array<unsigned>
    count = labeling::compute(accu::meta::math::count(),
                              lbl,
                              nlabels);

mln_concrete(I) output;
initialize(output, ima);
data::fill(output, literal::one);

for (unsigned i = 1; i <= nlabels; ++i)
    if (count[i] < 10u)
        data::fill((output | (pw::value(lbl) == pw::cst(i))).rw(),
                    literal::zero);

trace::exiting("my_algorithm");
return output;
}

} // end of namespace mln

template <typename I, typename N>
mln_concrete(I)
my_algorithm(const Image<I>& ima_,
             const Neighborhood<N>& nbh_)

    trace::entering("my_algorithm");

```

### Debug hints

```

const I& ima = exact(ima_);
const N& nbh = exact(nbh_);
mln_precondition(ima.is_valid());
mln_precondition(nbh.is_valid());

typedef value::label_8 V;
V nlabels;
mln_ch_value(I,V) lbl = labeling::blobs(ima, nbh, nlabels);
util::array<unsigned>
    count = labeling::compute(accu::meta::math::count(),
                              lbl,
                              nlabels);

mln_concrete(I) output;
initialize(output, ima);
data::fill(output, literal::one);

for (unsigned i = 1; i <= nlabels; ++i)
    if (count[i] < 10u)
        data::fill((output | (pw::value(lbl) == pw::cst(i))).rw(),
                    literal::zero);

trace::exiting("my_algorithm");
return output;

```

### 3.3.3 Compilation

#### 3.3.3.1 Include path

#### 3.3.3.2 Library linking

#### Input / Output

#### 3.3.3.3 Disable Debug

#### 3.3.3.4 Compiler optimization flags

##### 3.3.3.4.1 GCC

##### 3.3.3.4.2 Other compilers

### 3.3.4 Debug hints

#### 3.3.4.1 Using assertions and GDB

#### 3.3.4.2 Traces

```
// ...
trace::quiet = false;

labeling::blobs(ima, c4(), nlabels);

trace::quiet = true;

geom::bbox(ima);
// ...
```

#### 3.3.4.3 Debug routines

```
image2d<int_u8> ima(5,5);
data::fill(ima, 2);
debug::println(ima);

2 2 2 2 2
2 2 2 2 2
2 2 2 2 2
2 2 2 2 2
2 2 2 2 2

image2d<int_u8> ima(5,5);
data::fill(ima, 2);
border::fill(ima, 7);
debug::println_with_border(ima);

7 7 7 7 7 7 7 7 7 7
7 7 7 7 7 7 7 7 7 7
7 7 7 7 7 7 7 7 7 7
7 7 7 2 2 2 2 7 7 7
7 7 7 2 2 2 2 7 7 7
7 7 7 2 2 2 2 7 7 7
```

```

7 7 7 2 2 2 2 7 7 7
7 7 7 2 2 2 2 7 7 7
7 7 7 7 7 7 7 7 7 7
7 7 7 7 7 7 7 7 7 7
7 7 7 7 7 7 7 7 7 7

```

```

int_u8 vals[25] = { 100, 100, 200, 200, 230,
                  100, 100, 200, 230, 230,
                  140, 140, 140,  0,  0,
                  65, 186, 65, 127, 127,
                  65,  65, 65, 127, 127 };

image2d<int_u8> ima = make::image2d(vals);
image2d<rgb8> ima_color = labeling::colorize(rgb8(), ima, 230);

```

### Installation Data representation

## 3.4 Data representation

### 3.4.1 Sites

```

point2d p(3,3);
std::cout << p << std::endl;

```

```
(3,3)
```

### 3.4.2 Site sets

#### Site set

#### 3.4.2.1 Creating a site set

```
box2d b(4,4);
```

```
(0,0), (0,1), (0,2), (0,3), (1,0), (1,1), (1,2), (1,3), (2,0), (2,1), (2,2), (2,3),
(3,0), (3,1), (3,2), (3,3),
```

```

mln_piter_(p_array<point2d>) p(arr);
for_all(p)
  std::cout << p << ", ";
std::cout << std::endl;

```

```
(-2,-2), (-2,-1), (-2,0), (-2,1), (-2,2), (-1,-2), (-1,-1), (-1,0), (-1,1), (-1,2),
(0,-2), (0,-1), (0,0), (0,1), (0,2), (1,-2), (1,-1), (1,0), (1,1), (1,2), (2,-2),
(2,-1), (2,0), (2,1), (2,2),
```

```

mln_piter_(box2d) p(b);
for_all(p)
  std::cout << p << ", ";
std::cout << std::endl;

```

```
(2,2), (1,2),
```



### 3.4.2.2 Getting access to sites

## 3.4.3 Images

### 3.4.3.1 Creating an image

### 3.4.3.2 Reading an image from a file

### 3.4.3.3 Accessing data

[Getting started with Milena Load and save images](#)

## 3.5 Load and save images

```

image2d<bool> ima;
io::pbm::load(ima, MLN_DOC_DIR "/img/small.pbm");

io::pbm::save(ima, MLN_DOC_DIR "/figures/ima_save.pbm");

```

[Load and save images Data representation Create your first image](#)

## 3.6 Create your first image

See also

[tuto2\\_first\\_image.cc](#)

```

bool vals[13][21] = {
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0},
    {0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0},
    {0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0},
    {0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0},
    {0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0},
    {0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0},
    {0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0},
    {0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0},
    {0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
};

image2d<bool> ima = make::image(vals);

```

[Create an image](#)

```

debug::println(ima);

- - - - -
- | - | - | | | - | - - - | - - - - | - -
- | - | - | - - - | - - - | - - - | - | -
- | | | - | | | - | - - - | - - - | - | -
- | - | - | - - - | - - - | - - - | - | -

```

```

- | - | - | | | - | | | - | | | - - | - -
- - - - - - - - - - - - - - - - - - - -
- | - | - - | - - | | - - | - - - | | - -
- | - | - | - | - | - | - | - - - | - | -
- | - | - | - | - | | - - | - - - | - | -
- | | | - | - | - | - | - | - - - | - | -
- | - | - - | - - | - | - | | - | | - -
- - - - - - - - - - - - - - - - - - - -

```

```
doc::pbmsave(ima, "tuto2_first_image");
```

[Possible value types](#) [Load and save images](#) [Read and write images](#)

## 3.7 Read and write images

See also

[tuto3\\_rw\\_image.cc](#)

```

image2d<value::rgb8> ima(40, 40);

data::fill(ima, literal::red);

for (def::coord row = 20; row < 30; ++row)
  for (def::coord col = 20; col < 30; ++col)
    ima(point2d(row, col)) = literal::blue;

for (def::coord row = 20; row < 30; ++row)
  for (def::coord col = 20; col < 30; ++col)
    opt::at(ima, row, col) = literal::blue;

image2d<value::rgb8> lena;
io::ppm::load(lena, MLN_IMG_DIR "/small.ppm");

data::fill(ima, lena);

data::paste(ima, lena);

```

[Access and modify values](#) [Fill](#) [Paste](#) [Create your first image](#) [Regions of interest](#)

## 3.8 Regions of interest

See also

[tuto4\\_genericity\\_and\\_algorithms.cc](#)

```

image2d<value::rgb8> lena;
io::ppm::load(lena, MLN_IMG_DIR "/small.ppm");

namespace data
{
  template <typename I, typename D>
  void fill(Image<I>& ima, const D& data);
}

```

### 3.8.1 Image domain restricted by a site set

```
box2d roi = make::box2d(20, 20, 39, 39);

data::fill((lena | roi).rw(), literal::green);
```

### 3.8.2 Image domain restricted by a function

```
p_array<point2d> arr;
for (def::coord row = geom::min_row(lena); row < geom::max_row(lena); ++row)
  for (def::coord col = geom::min_col(lena); col < geom::max_col(lena); ++col)
    if (((row + col) % 2) == 0)
      arr.append(point2d(row, col));

for (def::coord row = geom::min_row(lena); row < geom::max_row(lena); ++row)
  for (def::coord col = geom::min_col(lena); col < geom::max_col(lena); ++col)
    if (((row + col) % 2) == 0)
      opt::at(lena, row, col) = literal::green;

data::fill((lena | fun::p2b::chess()).rw(), literal::green);
```

### 3.8.3 Image domain restricted by a mask

```
image2d<bool> mask;
initialize(mask, lena);
data::fill(mask, false);
data::fill((mask | make::box2d(10, 10, 14, 14)).rw(), true);
data::fill((mask | make::box2d(25, 15, 29, 18)).rw(), true);
data::fill((mask | make::box2d(50, 50, 54, 54)).rw(), true);

data::fill((lena | pw::value(mask)).rw(), literal::green);
```

### 3.8.4 Image domain restricted by a predicate

```
image2d<bool> lena_bw = binarization::binarization(lena, keep_specific_colors()
);
value::label_8 nlabels;
image2d<value::label_8> label = labeling::blobs(lena_bw, c8(), nlabels);

data::fill((lena | (pw::value(label) == pw::cst(0u))).rw(), literal::blue);
```

[Read and write images](#)



# Chapter 4

## Module Index

### 4.1 Modules

Here is a list of all modules:

|                                 |     |
|---------------------------------|-----|
| Types . . . . .                 | 98  |
| Graphes . . . . .               | 94  |
| Images . . . . .                | 95  |
| Basic types . . . . .           | 95  |
| Image morphers . . . . .        | 96  |
| Values morphers . . . . .       | 96  |
| Domain morphers . . . . .       | 97  |
| Identity morphers . . . . .     | 98  |
| Neighborhoods . . . . .         | 101 |
| 1D neighborhoods . . . . .      | 101 |
| 2D neighborhoods . . . . .      | 102 |
| 3D neighborhoods . . . . .      | 104 |
| Site sets . . . . .             | 108 |
| Basic types . . . . .           | 108 |
| Graph based . . . . .           | 109 |
| Complex based . . . . .         | 109 |
| Sparse types . . . . .          | 109 |
| Queue based . . . . .           | 110 |
| Utilities . . . . .             | 110 |
| Windows . . . . .               | 111 |
| 1D windows . . . . .            | 112 |
| 2D windows . . . . .            | 113 |
| 3D windows . . . . .            | 115 |
| N-D windows . . . . .           | 118 |
| Multiple windows . . . . .      | 118 |
| Accumulators . . . . .          | 99  |
| On site sets . . . . .          | 91  |
| On images . . . . .             | 91  |
| On values . . . . .             | 92  |
| Multiple accumulators . . . . . | 94  |
| Routines . . . . .              | 99  |
| Canvas . . . . .                | 99  |

|                             |     |
|-----------------------------|-----|
| Functions . . . . .         | 100 |
| v2w2v functions . . . . .   | 118 |
| v2w_w2v functions . . . . . | 119 |
| vv2b functions . . . . .    | 119 |

# Chapter 5

## Namespace Index

### 5.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

|   |     |
|---|-----|
| <a href="#">mln</a> ( <a href="#">Mln/convert/to_image.hh</a> ) . . . . .   | 121 |
| <a href="#">mln::accu</a> (Namespace of accumulators) . . . . .   | 164 |
| <a href="#">mln::accu::image</a> (Namespace of accumulator image routines) . . . . .  | 167 |
| <a href="#">mln::accu::impl</a> (Implementation namespace of accumulator namespace) . . . . .                                   | 167 |
| <a href="#">mln::accu::logic</a> (Namespace of logical accumulators) . . . . .  | 168 |
| <a href="#">mln::accu::math</a> (Namespace of mathematic accumulators) . . . . .  | 168 |
| <a href="#">mln::accu::meta::logic</a> (Namespace of logical meta-accumulators) . . . . .                                       | 169 |
| <a href="#">mln::accu::meta::math</a> (Namespace of mathematic meta-accumulators) . . . . .                                     | 169 |
| <a href="#">mln::accu::meta::shape</a> (Namespace of shape meta-accumulators) . . . . .   | 170 |
| <a href="#">mln::accu::meta::stat</a> (Namespace of statistical meta-accumulators) . . . . .                                    | 170 |
| <a href="#">mln::accu::shape</a> (Namespace of shape accumulators) . . . . .  | 171 |
| <a href="#">mln::accu::stat</a> (Namespace of statistical accumulators) . . . . .   | 171 |
| <a href="#">mln::algebra</a> (Namespace of algebraic structure) . . . . .   | 173 |
| <a href="#">mln::arith</a> (Namespace of arithmetic) . . . . .  | 175 |
| <a href="#">mln::arith::impl</a> (Implementation namespace of arith namespace) . . . . .  | 187 |
| <a href="#">mln::arith::impl::generic</a> (Generic implementation namespace of arith namespace) . . . . .                       | 188 |
| <a href="#">mln::binarization</a> (Namespace of "point-wise" expression tools) . . . . .  | 188 |
| <a href="#">mln::border</a> (Namespace of routines related to image virtual (outer) border) . . . . .                           | 189 |
| <a href="#">mln::border::impl</a> (Implementation namespace of border namespace) . . . . .                                      | 192 |
| <a href="#">mln::border::impl::generic</a> (Generic implementation namespace of border namespace) . . . . .                     | 193 |
| <a href="#">mln::canvas</a> (Namespace of canvas) . . . . .   | 193 |
| <a href="#">mln::canvas::browsing</a> (Namespace of browsing canvas) . . . . .  | 194 |
| <a href="#">mln::canvas::impl</a> (Implementation namespace of canvas namespace) . . . . .                                      | 195 |
| <a href="#">mln::canvas::labeling</a> (Namespace of labeling canvas) . . . . .  | 195 |
| <a href="#">mln::canvas::labeling::impl</a> (Implementation namespace of labeling canvas namespace) . . . . .                   | 196 |
| <a href="#">mln::canvas::morpho</a> (Namespace of morphological canvas) . . . . .   | 197 |
| <a href="#">mln::convert</a> (Namespace of conversion routines) . . . . .   | 197 |
| <a href="#">mln::data</a> (Namespace of image processing routines related to pixel data) . . . . .                              | 203 |
| <a href="#">mln::data::approx</a> (Namespace of image processing routines related to pixel levels with approximation) . . . . . | 215 |
| <a href="#">mln::data::approx::impl</a> (Implementation namespace of <a href="#">data::approx</a> namespace) . . . . .          | 217 |
| <a href="#">mln::data::impl</a> (Implementation namespace of data namespace) . . . . .  | 217 |
| <a href="#">mln::data::impl::generic</a> (Generic implementation namespace of data namespace) . . . . .                         | 220 |

|  |     |
|--|-----|
| <code>mln::data::naive</code> (Namespace of image processing routines related to pixel levels with naive approach) . . . . . | 223 |
| <code>mln::data::naive::impl</code> (Implementation namespace of <code>data::naive</code> namespace) . . . . .               | 224 |
| <code>mln::debug</code> (Namespace of routines that help to debug) . . . . .   | 224 |
| <code>mln::debug::impl</code> (Implementation namespace of debug namespace) . . . . .  | 230 |
| <code>mln::def</code> (Namespace for core definitions) . . . . .   | 230 |
| <code>mln::display</code> (Namespace of routines that help to display images) . . . . .                                      | 231 |
| <code>mln::display::impl</code> (Implementation namespace of display namespace) . . . . .                                    | 232 |
| <code>mln::display::impl::generic</code> (Generic implementation namespace of display namespace) . . . . .                   | 232 |
| <code>mln::doc</code> (The namespace <code>mln::doc</code> is only for documentation purpose) . . . . .                      | 232 |
| <code>mln::draw</code> (Namespace of drawing routines) . . . . .   | 234 |
| <code>mln::estim</code> (Namespace of estimation materials) . . . . .  | 237 |
| <code>mln::extension</code> (Namespace of extension tools) . . . . .   | 239 |
| <code>mln::fun</code> (Namespace of functions) . . . . .   | 242 |
| <code>mln::fun::access</code> (Namespace for access functions) . . . . .   | 243 |
| <code>mln::fun::i2v</code> (Namespace of integer-to-value functions) . . . . .   | 243 |
| <code>mln::fun::n2v</code> (Namespace of functions from nil to value) . . . . .  | 244 |
| <code>mln::fun::p2b</code> (Namespace of functions from point to boolean) . . . . .  | 244 |
| <code>mln::fun::p2p</code> (Namespace of functions from grid point to grid point) . . . . .                                  | 244 |
| <code>mln::fun::p2v</code> (Namespace of functions from point to value) . . . . .  | 245 |
| <code>mln::fun::stat</code> (Namespace of statistical functions) . . . . .   | 245 |
| <code>mln::fun::v2b</code> (Namespace of functions from value to logic value) . . . . .                                      | 245 |
| <code>mln::fun::v2i</code> (Namespace of value-to-integer functions) . . . . .   | 245 |
| <code>mln::fun::v2v</code> (Namespace of functions from value to value) . . . . .  | 246 |
| <code>mln::fun::v2w2v</code> (Namespace of bijective functions) . . . . .  | 247 |
| <code>mln::fun::v2w_w2v</code> (Namespace of functions from value to value) . . . . .  | 247 |
| <code>mln::fun::vv2b</code> (Namespace of functions from value to value) . . . . .   | 248 |
| <code>mln::fun::vv2v</code> (Namespace of functions from a couple of values to a value) . . . . .                            | 249 |
| <code>mln::fun::x2p</code> (Namespace of functions from point to value) . . . . .  | 249 |
| <code>mln::fun::x2v</code> (Namespace of functions from vector to value) . . . . .   | 250 |
| <code>mln::fun::x2x</code> (Namespace of functions from vector to vector) . . . . .  | 250 |
| <code>mln::geom</code> (Namespace of all things related to geometry) . . . . .   | 250 |
| <code>mln::geom::impl</code> (Implementation namespace of geom namespace) . . . . .  | 264 |
| <code>mln::graph</code> (Namespace of graph related routines) . . . . .  | 265 |
| <code>mln::grid</code> (Namespace of grids definitions) . . . . .  | 267 |
| <code>mln::histo</code> (Namespace of histograms) . . . . .  | 267 |
| <code>mln::histo::impl</code> (Implementation namespace of histo namespace) . . . . .  | 269 |
| <code>mln::histo::impl::generic</code> (Generic implementation namespace of histo namespace) . . . . .                       | 269 |
| <code>mln::impl</code> (Implementation namespace of mln namespace) . . . . .   | 269 |
| <code>mln::io</code> (Namespace of input/output handling) . . . . .  | 269 |
| <code>mln::io::cloud</code> (Namespace of cloud input/output handling) . . . . .   | 271 |
| <code>mln::io::dicom</code> (Namespace of DICOM input/output handling) . . . . .   | 272 |
| <code>mln::io::dump</code> (Namespace of dump input/output handling) . . . . .   | 273 |
| <code>mln::io::fits</code> (Namespace of fits input/output handling) . . . . .   | 274 |
| <code>mln::io::fld</code> (Namespace of pgm input/output handling) . . . . .   | 275 |
| <code>mln::io::magick</code> (Namespace of magick input/output handling) . . . . .   | 276 |
| <code>mln::io::off</code> (Namespace of off input/output handling) . . . . .   | 278 |
| <code>mln::io::pbm</code> (Namespace of pbm input/output handling) . . . . .   | 279 |
| <code>mln::io::pbm::impl</code> (Namespace of pbm implementation details) . . . . .  | 281 |
| <code>mln::io::pbms</code> (Namespace of pbms input/output handling) . . . . .   | 281 |
| <code>mln::io::pbms::impl</code> (Namespace of pbms implementation details) . . . . .  | 282 |
| <code>mln::io::pfm</code> (Namespace of pfm input/output handling) . . . . .   | 282 |
| <code>mln::io::pfm::impl</code> (Implementation namespace of pfm namespace) . . . . .  | 283 |
| <code>mln::io::pgm</code> (Namespace of pgm input/output handling) . . . . .   | 284 |



|   |     |
|---|-----|
| <a href="#">mln::io::pgms</a> (Namespace of pgms input/output handling ) . . . . .  | 285 |
| <a href="#">mln::io::plot</a> (Namespace of plot input/output handling ) . . . . .  | 285 |
| <a href="#">mln::io::pnm</a> (Namespace of pnm input/output handling ) . . . . .  | 287 |
| <a href="#">mln::io::pnm::impl</a> (Namespace of pnm's implementation details ) . . . . .   | 289 |
| <a href="#">mln::io::pnms</a> (Namespace of pnms input/output handling ) . . . . .  | 289 |
| <a href="#">mln::io::ppm</a> (Namespace of ppm input/output handling ) . . . . .  | 290 |
| <a href="#">mln::io::ppms</a> (Namespace of ppms input/output handling ) . . . . .  | 292 |
| <a href="#">mln::io::raw</a> (Namespace of raw input/output handling ) . . . . .  | 292 |
| <a href="#">mln::io::tiff</a> (Namespace of tiff input/output handling ) . . . . .  | 294 |
| <a href="#">mln::io::txt</a> (Namespace of txt input/output handling ) . . . . .  | 294 |
| <a href="#">mln::labeling</a> (Namespace of labeling routines ) . . . . .   | 295 |
| <a href="#">mln::labeling::impl</a> (Implementation namespace of labeling namespace ) . . . . .   | 310 |
| <a href="#">mln::labeling::impl::generic</a> (Generic implementation namespace of labeling namespace ) . . . . .                              | 312 |
| <a href="#">mln::linear</a> (Namespace of linear image processing routines ) . . . . .  | 314 |
| <a href="#">mln::linear::impl</a> (Namespace of linear image processing routines implementation details ) . . . . .                           | 318 |
| <a href="#">mln::linear::local</a> (Specializations of local linear routines ) . . . . .  | 318 |
| <a href="#">mln::linear::local::impl</a> (Namespace of local linear routines implementation details ) . . . . .                               | 319 |
| <a href="#">mln::literal</a> (Namespace of literals ) . . . . .   | 319 |
| <a href="#">mln::logical</a> (Namespace of logic ) . . . . .  | 326 |
| <a href="#">mln::logical::impl</a> (Implementation namespace of logical namespace ) . . . . .   | 329 |
| <a href="#">mln::logical::impl::generic</a> (Generic implementation namespace of logical namespace ) . . . . .                                | 329 |
| <a href="#">mln::make</a> (Namespace of routines that help to make Milena's objects ) . . . . .   | 330 |
| <a href="#">mln::math</a> (Namespace of mathematical routines ) . . . . .   | 354 |
| <a href="#">mln::metal</a> (Namespace of meta-programming tools ) . . . . .   | 355 |
| <a href="#">mln::metal::impl</a> (Implementation namespace of metal namespace ) . . . . .   | 356 |
| <a href="#">mln::metal::math</a> (Namespace of static mathematical functions ) . . . . .  | 356 |
| <a href="#">mln::metal::math::impl</a> (Implementation namespace of <a href="#">metal::math</a> namespace ) . . . . .                         | 357 |
| <a href="#">mln::morpho</a> (Namespace of mathematical morphology routines ) . . . . .  | 357 |
| <a href="#">mln::morpho::approx</a> (Namespace of approximate mathematical morphology routines ) . . . . .                                    | 367 |
| <a href="#">mln::morpho::attribute</a> (Namespace of attributes used in mathematical morphology ) . . . . .                                   | 367 |
| <a href="#">mln::morpho::closing::approx</a> (Namespace of approximate mathematical morphology closing routines ) . . . . .                   | 367 |
| <a href="#">mln::morpho::elementary</a> (Namespace of image processing routines of elementary mathematical morphology ) . . . . .             | 368 |
| <a href="#">mln::morpho::impl</a> (Namespace of mathematical morphology routines implementations ) . . . . .                                  | 370 |
| <a href="#">mln::morpho::impl::generic</a> (Namespace of mathematical morphology routines generic implementations ) . . . . .                 | 370 |
| <a href="#">mln::morpho::opening::approx</a> (Namespace of approximate mathematical morphology opening routines ) . . . . .                   | 370 |
| <a href="#">mln::morpho::reconstruction</a> (Namespace of morphological reconstruction routines ) . . . . .                                   | 371 |
| <a href="#">mln::morpho::reconstruction::by_dilation</a> (Namespace of morphological reconstruction by dilation routines ) . . . . .          | 371 |
| <a href="#">mln::morpho::reconstruction::by_erosion</a> (Namespace of morphological reconstruction by erosion routines ) . . . . .            | 371 |
| <a href="#">mln::morpho::tree</a> (Namespace of morphological tree-related routines ) . . . . .   | 372 |
| <a href="#">mln::morpho::tree::filter</a> (Namespace for attribute filtering ) . . . . .  | 379 |
| <a href="#">mln::morpho::watershed</a> (Namespace of morphological watershed routines ) . . . . .   | 382 |
| <a href="#">mln::morpho::watershed::watershed</a> (Namespace of morphological watershed routines implementations ) . . . . .                  | 384 |
| <a href="#">mln::morpho::watershed::watershed::generic</a> (Namespace of morphological watershed routines generic implementations ) . . . . . | 384 |
| <a href="#">mln::norm</a> (Namespace of norms ) . . . . .   | 384 |
| <a href="#">mln::norm::impl</a> (Implementation namespace of norm namespace ) . . . . .   | 386 |
| <a href="#">mln::opt</a> (Namespace of optional routines ) . . . . .  | 387 |

---

|  |     |
|--|-----|
| <a href="#">mln::opt::impl</a> (Implementation namespace of opt namespace ) . . . . .                | 388 |
| <a href="#">mln::pw</a> (Namespace of "point-wise" expression tools ) . . . . .                      | 389 |
| <a href="#">mln::registration</a> (Namespace of "point-wise" expression tools ) . . . . .            | 389 |
| <a href="#">mln::select</a> (Select namespace (FIXME doc) ) . . . . .                                | 392 |
| <a href="#">mln::set</a> (Namespace of image processing routines related to pixel sets ) . . . . .   | 392 |
| <a href="#">mln::subsampling</a> (Namespace of "point-wise" expression tools ) . . . . .             | 394 |
| <a href="#">mln::tag</a> (Namespace of image processing routines related to tags ) . . . . .         | 396 |
| <a href="#">mln::test</a> (Namespace of image processing routines related to pixel tests ) . . . . . | 396 |
| <a href="#">mln::test::impl</a> (Implementation namespace of test namespace ) . . . . .              | 398 |
| <a href="#">mln::topo</a> (Namespace of "point-wise" expression tools ) . . . . .                    | 398 |
| <a href="#">mln::trace</a> (Namespace of routines related to the trace mechanism ) . . . . .         | 408 |
| <a href="#">mln::trait</a> (Namespace where traits are defined ) . . . . .                           | 408 |
| <a href="#">mln::transform</a> (Namespace of transforms ) . . . . .                                  | 408 |
| <a href="#">mln::util</a> (Namespace of tools using for more complex algorithm ) . . . . .           | 413 |
| <a href="#">mln::util::impl</a> (Implementation namespace of util namespace ) . . . . .              | 420 |
| <a href="#">mln::value</a> (Namespace of materials related to pixel value types ) . . . . .          | 420 |
| <a href="#">mln::value::impl</a> (Implementation namespace of value namespace ) . . . . .            | 432 |
| <a href="#">mln::win</a> (Namespace of image processing routines related to win ) . . . . .          | 432 |

# Chapter 6

## Class Index

### 6.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

|  |     |
|--|-----|
| mln::Generalized_Pixel< bkd_pixter1d< I > >              |     |
| mln::Pixel_Iterator< bkd_pixter1d< I > > . . . . .       | 940 |
| pixel_iterator_base_< I, bkd_pixter1d< I > >             |     |
| backward_pixel_iterator_base_< I, bkd_pixter1d< I > >    |     |
| mln::bkd_pixter1d< I > . . . . .                         | 548 |
| mln::Generalized_Pixel< bkd_pixter2d< I > >              |     |
| mln::Pixel_Iterator< bkd_pixter2d< I > > . . . . .       | 940 |
| pixel_iterator_base_< I, bkd_pixter2d< I > >             |     |
| backward_pixel_iterator_base_< I, bkd_pixter2d< I > >    |     |
| mln::bkd_pixter2d< I > . . . . .                         | 549 |
| mln::Generalized_Pixel< bkd_pixter3d< I > >              |     |
| mln::Pixel_Iterator< bkd_pixter3d< I > > . . . . .       | 940 |
| pixel_iterator_base_< I, bkd_pixter3d< I > >             |     |
| backward_pixel_iterator_base_< I, bkd_pixter3d< I > >    |     |
| mln::bkd_pixter3d< I > . . . . .                         | 551 |
| mln::Generalized_Pixel< dpoints_bkd_pixter< I > >        |     |
| mln::Pixel_Iterator< dpoints_bkd_pixter< I > > . . . . . | 940 |
| mln::dpoints_bkd_pixter< I > . . . . .                   | 648 |
| mln::Generalized_Pixel< dpoints_fwd_pixter< I > >        |     |
| mln::Pixel_Iterator< dpoints_fwd_pixter< I > > . . . . . | 940 |
| mln::dpoints_fwd_pixter< I > . . . . .                   | 652 |
| mln::Generalized_Pixel< fwd_pixter1d< I > >              |     |
| mln::Pixel_Iterator< fwd_pixter1d< I > > . . . . .       | 940 |
| pixel_iterator_base_< I, fwd_pixter1d< I > >             |     |
| forward_pixel_iterator_base_< I, fwd_pixter1d< I > >     |     |
| mln::fwd_pixter1d< I > . . . . .                         | 719 |
| mln::Generalized_Pixel< fwd_pixter2d< I > >              |     |
| mln::Pixel_Iterator< fwd_pixter2d< I > > . . . . .       | 940 |
| pixel_iterator_base_< I, fwd_pixter2d< I > >             |     |
| forward_pixel_iterator_base_< I, fwd_pixter2d< I > >     |     |
| mln::fwd_pixter2d< I > . . . . .                         | 720 |
| mln::Generalized_Pixel< fwd_pixter3d< I > >              |     |
| mln::Pixel_Iterator< fwd_pixter3d< I > > . . . . .       | 940 |

|   |      |
|---|------|
| pixel_iterator_base_< I, fwd_pixter3d< I > >  |      |
| forward_pixel_iterator_base_< I, fwd_pixter3d< I > >  |      |
| mln::fwd_pixter3d< I > . . . . .  | 722  |
| mln::Generalized_Pixel< pixel< I > >  |      |
| mln::pixel< I > . . . . .   | 939  |
| mln::internal::image_base< algebra::vec< n, I::value >, I::domain_t, stack_image< n, I > >  |      |
| image_morpher< I, algebra::vec< n, I::value >, I::domain_t, stack_image< n, I > >   |      |
| image_value_morpher< I, algebra::vec< n, I::value >, stack_image< n, I > >  |      |
| mln::value::stack_image< n, I > . . . . .   | 1142 |
| mln::internal::image_base< const I::value, I::domain_t, E >   |      |
| image_morpher< const I, const I::value, I::domain_t, E >  |      |
| image_identity< const I, I::domain_t, E >   |      |
| mln::labeled_image_base< I, E > . . . . .   | 793  |
| mln::internal::image_base< const I::value, I::domain_t, labeled_image< I > >  |      |
| image_morpher< const I, const I::value, I::domain_t, labeled_image< I > >   |      |
| image_identity< const I, I::domain_t, labeled_image< I > >  |      |
| mln::labeled_image_base< I, labeled_image< I > > . . . . .  | 793  |
| mln::labeled_image< I > . . . . .   | 789  |
| mln::internal::image_base< F::result, I1::domain_t, thrubin_image< I1, I2, F > >  |      |
| image_morpher< I1, F::result, I1::domain_t, thrubin_image< I1, I2, F > >  |      |
| image_value_morpher< I1, F::result, thrubin_image< I1, I2, F > >  |      |
| mln::thrubin_image< I1, I2, F > . . . . .   | 975  |
| mln::internal::image_base< F::result, I::domain_t, fun_image< F, I > >  |      |
| image_morpher< I, F::result, I::domain_t, fun_image< F, I > >   |      |
| image_value_morpher< I, F::result, fun_image< F, I > >  |      |
| mln::fun_image< F, I > . . . . .  | 712  |
| mln::internal::image_base< F::result, S, E >  |      |
| image_primary< F::result, S, E >  |      |
| mln::internal::image_base< F::result, S, image< F, S > >  |      |
| image_primary< F::result, S, image< F, S > >  |      |
| image_base< F, S, image< F, S > >   |      |
| mln::pw::image< F, S > . . . . .  | 958  |
| mln::internal::image_base< fun::i2v::array< V >::result, p_edges< G, internal::efsite_selector< P, G >::site_function_t >, edge_image< P, V, G > >      |      |
| image_primary< fun::i2v::array< V >::result, p_edges< G, internal::efsite_selector< P, G >::site_function_t >, edge_image< P, V, G > >                  |      |
| image_base< fun::i2v::array< V >, p_edges< G, internal::efsite_selector< P, G >::site_function_t >, edge_image< P, V, G > >                             |      |
| mln::edge_image< P, V, G > . . . . .  | 655  |
| mln::internal::image_base< fun::i2v::array< V >::result, p_vertices< G, internal::vfsite_selector< P, G >::site_function_t >, vertex_image< P, V, G > > |      |
| image_primary< fun::i2v::array< V >::result, p_vertices< G, internal::vfsite_selector< P, G >::site_function_t >, vertex_image< P, V, G > >             |      |
| image_base< fun::i2v::array< V >, p_vertices< G, internal::vfsite_selector< P, G >::site_function_t >, vertex_image< P, V, G > >                        |      |
| mln::vertex_image< P, V, G > . . . . .  | 1150 |
| mln::internal::image_base< I::value, box2d, slice_image< I > >  |      |
| image_morpher< I, I::value, box2d, slice_image< I > >   |      |
| image_domain_morpher< I, box2d, slice_image< I > >  |      |
| mln::slice_image< I > . . . . .   | 969  |
| mln::internal::image_base< I::value, box2d_h, hexa< I > >   |      |
| image_morpher< I, I::value, box2d_h, hexa< I > >  |      |
| image_domain_morpher< I, box2d_h, hexa< I > >   |      |
| mln::hexa< I > . . . . .  | 758  |

|  |      |
|--|------|
| mln::internal::image_base< I::value, box< I::site >, extended< I > >                       |      |
| image_morpher< I, I::value, box< I::site >, extended< I > >                                |      |
| image_domain_morpher< I, box< I::site >, extended< I > >                                   |      |
| mln::extended< I > . . . . .   | 657  |
| mln::internal::image_base< I::value, D, unproject_image< I, D, F > >                       |      |
| image_morpher< I, I::value, D, unproject_image< I, D, F > >                                |      |
| image_domain_morpher< I, D, unproject_image< I, D, F > >                                   |      |
| mln::unproject_image< I, D, F > . . . . .  | 1033 |
| mln::internal::image_base< I::value, I::domain_t, decorated_image< I, D > >                |      |
| image_morpher< I, I::value, I::domain_t, decorated_image< I, D > >                         |      |
| image_identity< I, I::domain_t, decorated_image< I, D > >                                  |      |
| mln::decorated_image< I, D > . . . . .   | 596  |
| mln::internal::image_base< I::value, I::domain_t, extension_fun< I, F > >                  |      |
| image_morpher< I, I::value, I::domain_t, extension_fun< I, F > >                           |      |
| image_identity< I, I::domain_t, extension_fun< I, F > >                                    |      |
| mln::extension_fun< I, F > . . . . .   | 659  |
| mln::internal::image_base< I::value, I::domain_t, extension_ima< I, J > >                  |      |
| image_morpher< I, I::value, I::domain_t, extension_ima< I, J > >                           |      |
| image_identity< I, I::domain_t, extension_ima< I, J > >                                    |      |
| mln::extension_ima< I, J > . . . . .   | 661  |
| mln::internal::image_base< I::value, I::domain_t, extension_val< I > >                     |      |
| image_morpher< I, I::value, I::domain_t, extension_val< I > >                              |      |
| image_identity< I, I::domain_t, extension_val< I > >                                       |      |
| mln::extension_val< I > . . . . .  | 664  |
| mln::internal::image_base< I::value, I::domain_t, interpolated< I, F > >                   |      |
| image_morpher< I, I::value, I::domain_t, interpolated< I, F > >                            |      |
| image_identity< I, I::domain_t, interpolated< I, F > >                                     |      |
| mln::interpolated< I, F > . . . . .  | 784  |
| mln::internal::image_base< I::value, I::domain_t, p2p_image< I, F > >                      |      |
| image_morpher< I, I::value, I::domain_t, p2p_image< I, F > >                               |      |
| image_domain_morpher< I, I::domain_t, p2p_image< I, F > >                                  |      |
| mln::p2p_image< I, F > . . . . .   | 845  |
| mln::internal::image_base< I::value, I::domain_t, plain< I > >                             |      |
| image_morpher< I, I::value, I::domain_t, plain< I > >                                      |      |
| image_identity< I, I::domain_t, plain< I > >   |      |
| mln::plain< I > . . . . .  | 942  |
| mln::internal::image_base< I::value, I::domain_t, safe_image< I > >                        |      |
| image_morpher< I, I::value, I::domain_t, safe_image< I > >                                 |      |
| image_identity< I, I::domain_t, safe_image< I > >  |      |
| mln::safe_image< I > . . . . .   | 961  |
| mln::internal::image_base< I::value, p_if< I::domain_t, F >, image_if< I, F > >            |      |
| image_morpher< I, I::value, p_if< I::domain_t, F >, image_if< I, F > >                     |      |
| image_domain_morpher< I, p_if< I::domain_t, F >, image_if< I, F > >                        |      |
| mln::image_if< I, F > . . . . .  | 782  |
| mln::internal::image_base< I::value, p_if< S, fun::p2b::has< I > >, sub_image_if< I, S > > |      |
| image_morpher< I, I::value, p_if< S, fun::p2b::has< I > >, sub_image_if< I, S > >          |      |
| image_domain_morpher< I, p_if< S, fun::p2b::has< I > >, sub_image_if< I, S > >             |      |
| mln::sub_image_if< I, S > . . . . .  | 973  |
| mln::internal::image_base< I::value, S, E >  |      |
| image_morpher< I, I::value, S, E >   |      |
| mln::internal::image_base< I::value, S, sub_image< I, S > >                                |      |
| image_morpher< I, I::value, S, sub_image< I, S > >   |      |
| image_domain_morpher< I, S, sub_image< I, S > >  |      |
| mln::sub_image< I, S > . . . . .   | 972  |

mln::internal::image\_base< I::value, S, tr\_image< S, I, T > >  
   image\_morpher< I, I::value, S, tr\_image< S, I, T > >  
   image\_identity< I, S, tr\_image< S, I, T > >  
     mln::tr\_image< S, I, T > . . . . . 1030  
 mln::internal::image\_base< image2d< V >::value, box2d\_h, hexa< image2d< V > > >  
   image\_morpher< image2d< V >, image2d< V >::value, box2d\_h, hexa< image2d< V > > >  
   image\_domain\_morpher< image2d< V >, box2d\_h, hexa< image2d< V > > >  
     mln::hexa< image2d< V > > . . . . . 758  
     mln::image2d\_h< V > . . . . . 774  
 mln::internal::image\_base< mln::trait::ch\_value< I, F::result >::ret::value, I::domain\_t, lazy\_image< I, F, B > >  
   image\_morpher< mln::trait::ch\_value< I, F::result >::ret, mln::trait::ch\_value< I, F::result >::ret::value, I::domain\_t, lazy\_image< I, F, B > >  
   image\_identity< mln::trait::ch\_value< I, F::result >::ret, I::domain\_t, lazy\_image< I, F, B > >  
     mln::lazy\_image< I, F, B > . . . . . 796  
 mln::internal::image\_base< T, box1d, image1d< T > >  
   image\_primary< T, box1d, image1d< T > >  
     mln::image1d< T > . . . . . 764  
 mln::internal::image\_base< T, box3d, image3d< T > >  
   image\_primary< T, box3d, image3d< T > >  
     mln::image3d< T > . . . . . 777  
 mln::internal::image\_base< T, I::domain\_t, E >  
   image\_morpher< I, T, I::domain\_t, E >  
 mln::internal::image\_base< T, I::domain\_t, violent\_cast\_image< T, I > >  
   image\_morpher< I, T, I::domain\_t, violent\_cast\_image< T, I > >  
   image\_value\_morpher< I, T, violent\_cast\_image< T, I > >  
     mln::violent\_cast\_image< T, I > . . . . . 1152  
 mln::internal::image\_base< T, mln::box2d, image2d< T > >  
   image\_primary< T, mln::box2d, image2d< T > >  
     mln::image2d< T > . . . . . 769  
 mln::internal::image\_base< T, S, flat\_image< T, S > >  
   image\_primary< T, S, flat\_image< T, S > >  
     mln::flat\_image< T, S > . . . . . 669  
 mln::internal::image\_base< V, p\_complex< D, G >, complex\_image< D, G, V > >  
   image\_primary< V, p\_complex< D, G >, complex\_image< D, G, V > >  
     mln::complex\_image< D, G, V > . . . . . 584  
 mln::internal::check::image\_fastest< complex\_image< D, G, V >, mln::metal::equal< mln\_trait\_image\_speed(complex\_image< D, G, V >), trait::image::speed::fastest >::eval >  
 mln::internal::check::image\_fastest< decorated\_image< I, D >, mln::metal::equal< mln\_trait\_image\_speed(decorated\_image< I, D >), trait::image::speed::fastest >::eval >  
 mln::internal::check::image\_fastest< E, mln::metal::equal< mln\_trait\_image\_speed(E), trait::image::speed::fastest >::eval >  
 mln::internal::check::image\_fastest< edge\_image< P, V, G >, mln::metal::equal< mln\_trait\_image\_speed(edge\_image< P, V, G >), trait::image::speed::fastest >::eval >  
 mln::internal::check::image\_fastest< extended< I >, mln::metal::equal< mln\_trait\_image\_speed(extended< I >), trait::image::speed::fastest >::eval >  
 mln::internal::check::image\_fastest< extension\_fun< I, F >, mln::metal::equal< mln\_trait\_image\_speed(extension\_fun< I, F >), trait::image::speed::fastest >::eval >  
 mln::internal::check::image\_fastest< extension\_ima< I, J >, mln::metal::equal< mln\_trait\_image\_speed(extension\_ima< I, J >), trait::image::speed::fastest >::eval >  
 mln::internal::check::image\_fastest< extension\_val< I >, mln::metal::equal< mln\_trait\_image\_speed(extension\_val< I >), trait::image::speed::fastest >::eval >  
 mln::internal::check::image\_fastest< flat\_image< T, S >, mln::metal::equal< mln\_trait\_image\_

```

speed(flat_image< T, S >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< fun_image< F, I >, mln::metal::equal< mln_trait_image_
speed(fun_image< F, I >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< hexa< I >, mln::metal::equal< mln_trait_image_speed(hexa<
I >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< hexa< image2d< V > >, mln::metal::equal< mln_trait_
image_speed(hexa< image2d< V > >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< image1d< T >, mln::metal::equal< mln_trait_image_
speed(image1d< T >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< image2d< T >, mln::metal::equal< mln_trait_image_
speed(image2d< T >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< image3d< T >, mln::metal::equal< mln_trait_image_
speed(image3d< T >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< image< F, S >, mln::metal::equal< mln_trait_image_
speed(image< F, S >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< image_if< I, F >, mln::metal::equal< mln_trait_image_
speed(image_if< I, F >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< interpolated< I, F >, mln::metal::equal< mln_trait_image_
speed(interpolated< I, F >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< labeled_image< I >, mln::metal::equal< mln_trait_image_
speed(labeled_image< I >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< lazy_image< I, F, B >, mln::metal::equal< mln_trait_image_
speed(lazy_image< I, F, B >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< p2p_image< I, F >, mln::metal::equal< mln_trait_image_
speed(p2p_image< I, F >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< plain< I >, mln::metal::equal< mln_trait_image_speed(plain<
I >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< safe_image< I >, mln::metal::equal< mln_trait_image_
speed(safe_image< I >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< slice_image< I >, mln::metal::equal< mln_trait_image_
speed(slice_image< I >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< stack_image< n, I >, mln::metal::equal< mln_trait_image_
speed(stack_image< n, I >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< sub_image< I, S >, mln::metal::equal< mln_trait_image_
speed(sub_image< I, S >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< sub_image_if< I, S >, mln::metal::equal< mln_trait_image_
speed(sub_image_if< I, S >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< thrubin_image< I1, I2, F >, mln::metal::equal< mln_trait_
image_speed(thrubin_image< I1, I2, F >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< tr_image< S, I, T >, mln::metal::equal< mln_trait_image_
speed(tr_image< S, I, T >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< unproject_image< I, D, F >, mln::metal::equal< mln_trait_
image_speed(unproject_image< I, D, F >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< vertex_image< P, V, G >, mln::metal::equal< mln_trait_
image_speed(vertex_image< P, V, G >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< violent_cast_image< T, I >, mln::metal::equal< mln_trait_
image_speed(violent_cast_image< T, I >), trait::image::speed::fastest >::eval >
mln::internal::impl_selector< W:center_t, W::psite, graph_window_piter< S, W, I > >
    mln::graph_window_piter< S, W, I > . . . . . 755
mln::value::Integer< graylevel< n > > . . . . . 1125
    mln::value::graylevel< n > . . . . . 1112
mln::value::Integer< int_s< n > > . . . . . 1125
    mln::value::int_s< n > . . . . . 1118

```

|  |      |
|--|------|
| mln::value::Integer< int_u< n > > . . . . .  | 1125 |
| mln::value::int_u< n > . . . . .   | 1121 |
| mln::value::Integer< int_u_sat< n > > . . . . .  | 1125 |
| mln::value::int_u_sat< n > . . . . .   | 1123 |
| mln::value::Integer< object_id< Tag, V > > . . . . .   | 1125 |
| mln::util::object_id< Tag, V > . . . . .   | 1075 |
| mln::internal::is_masked_impl_selector< S, W::mask_t::domain_t, graph_window_if_piter< S, W, I > > . . . . . |      |
| mln::graph_window_if_piter< S, W, I > . . . . .  | 753  |
| mln::algebra::h_mat< d, T > . . . . .  | 544  |
| mln::algebra::h_vec< d, C > . . . . .  | 546  |
| mln::canvas::chamfer< F > . . . . .  | 583  |
| mln::category< R(*) (A) > . . . . .  | 583  |
| mln::Delta_Point_Site< void > . . . . .  | 599  |
| mln::doc::Accumulator< E > . . . . .   | 599  |
| mln::doc::Generalized_Pixel< E > . . . . .   | 614  |
| mln::doc::Pixel_Iterator< E > . . . . .  | 625  |
| mln::doc::Object< E > . . . . .  | 625  |
| mln::doc::Dpoint< E > . . . . .  | 604  |
| mln::doc::Image< E > . . . . .   | 616  |
| mln::doc::Fastest_Image< E > . . . . .   | 606  |
| mln::doc::Iterator< E > . . . . .  | 622  |
| mln::doc::Pixel_Iterator< E > . . . . .  | 625  |
| mln::doc::Site_Iterator< E > . . . . .   | 631  |
| mln::doc::Value_Iterator< E > . . . . .  | 635  |
| mln::doc::Neighborhood< E > . . . . .  | 623  |
| mln::doc::Site_Set< E > . . . . .  | 633  |
| mln::doc::Box< E > . . . . .   | 601  |
| mln::doc::Value_Set< E > . . . . .   | 636  |
| mln::doc::Weighted_Window< E > . . . . .   | 639  |
| mln::doc::Window< E > . . . . .  | 641  |
| mln::doc::Point_Site< E > . . . . .  | 628  |
| mln::Edge< E > . . . . .   | 655  |
| mln::fun::from_accu< A > . . . . .   | 671  |
| mln::fun::internal::ch_function_value_impl< F, V > . . . . .   |      |
| mln::fun::v2v::ch_function_value< F, V > . . . . .   | 677  |
| mln::fun::x2p::closest_point< P > . . . . .  | 702  |
| mln::fun::x2x::composed< T2, T1 > . . . . .  | 704  |
| mln::Function< void > . . . . .  | 715  |
| mln::Gdpoint< void > . . . . .   | 724  |
| mln::Generalized_Pixel< E > . . . . .  |      |
| mln::Pixel_Iterator< E > . . . . .   | 940  |
| mln::internal::pixel_iterator_base_ . . . . .  |      |
| mln::geom::complex_geometry< D, P > . . . . .  | 725  |
| mln::graph::attribute::card_t . . . . .  | 732  |
| mln::graph::attribute::representative_t . . . . .  | 732  |
| mln::histo::array< T > . . . . .   | 762  |
| mln::internal::image_base< T, S, E > . . . . .   |      |
| mln::internal::neighborhood_base< W, E > . . . . .   |      |
| mln::internal::neighb_base . . . . .   |      |
| mln::neighb< graph_elt_mixed_window< G, S, S2 > > . . . . .  | 842  |
| mln::graph_elt_mixed_neighborhood< G, S, S2 > . . . . .  | 733  |



|  |      |
|--|------|
| mln::neighb< graph_elt_window< G, S > > . . . . .            | 842  |
| mln::graph_elt_neighborhood< G, S > . . . . .                | 738  |
| mln::neighb< graph_elt_window_if< G, S, I > > . . . . .      | 842  |
| mln::graph_elt_neighborhood_if< G, S, I > . . . . .          | 740  |
| mln::internal::pixel_impl_< I, E >                           |      |
| mln::internal::pixel_iterator_base_                          |      |
| mln::io::dicom::dicom_header . . . . .                       | 786  |
| mln::io::dump::dump_header . . . . .                         | 786  |
| mln::io::fld::fld_header . . . . .                           | 787  |
| mln::io::raw::raw_header . . . . .                           | 787  |
| mln::metal::ands< E1, E2, E3, E4, E5, E6, E7, E8 > . . . . . | 828  |
| mln::metal::bool_< false >                                   |      |
| mln::metal::equal< T1::coord, T2::coord > . . . . .          | 828  |
| mln::metal::equal< T1::point, T2::point > . . . . .          | 828  |
| mln::metal::equal< T1, T2 > . . . . .                        | 828  |
| mln::metal::converts_to< T, U > . . . . .                    | 828  |
| mln::metal::goes_to< T, U > . . . . .                        | 829  |
| mln::metal::is< T, U > . . . . .                             | 829  |
| mln::metal::is_a< T, M > . . . . .                           | 830  |
| mln::metal::is_not< T, U > . . . . .                         | 830  |
| mln::metal::is_not_a< T, M > . . . . .                       | 830  |
| mln::Neighborhood< void > . . . . .                          | 844  |
| mln::Object< E > . . . . .                                   | 844  |
| mln::Browsing< E > . . . . .                                 | 568  |
| mln::Delta_Point_Site  |      |
| mln::Dpoint< E > . . . . .                                   | 643  |
| mln::Function< E > . . . . .                                 | 715  |
| mln::Function_n2v< E > . . . . .                             | 716  |
| mln::Function_v2v< E > . . . . .                             | 717  |
| mln::fun::x2v::bilinear< I > . . . . .                       | 703  |
| mln::fun::x2v::trilinear< I > . . . . .                      | 704  |
| mln::fun::x2x::linear< I > . . . . .                         | 705  |
| mln::Function_v2b< E > . . . . .                             | 716  |
| mln::Function_vv2b< E > . . . . .                            | 718  |
| mln::Function_vv2v< E > . . . . .                            | 718  |
| mln::Gdpoint< E > . . . . .                                  | 723  |
| mln::Graph< E > . . . . .                                    | 731  |
| mln::Image< E > . . . . .                                    | 762  |
| mln::Iterator< E > . . . . .                                 | 787  |
| mln::Pixel_Iterator< E > . . . . .                           | 940  |
| mln::topo::internal::complex_iterator_base                   |      |
| mln::topo::internal::complex_relative_iterator_base          |      |
| mln::Value_Iterator< E > . . . . .                           | 1146 |
| mln::Literal< E > . . . . .                                  | 799  |
| mln::Mesh< E > . . . . .                                     | 823  |
| mln::Regular_Grid< E > . . . . .                             | 960  |
| mln::Meta_Accumulator< E > . . . . .                         | 824  |
| mln::Meta_Function< E > . . . . .                            | 826  |
| mln::Meta_Function_v2v< E > . . . . .                        | 827  |
| mln::Meta_Function_vv2v< E > . . . . .                       | 827  |
| mln::Neighborhood< E > . . . . .                             | 844  |
| mln::Point_Site< E > . . . . .                               | 953  |
| mln::Proxy< E > . . . . .                                    | 957  |

|   |      |
|---|------|
| mln::Accumulator< E > . . . . .   | 542  |
| mln::accu::internal::base   |      |
| couple< accu::shape::bbox< P >, accu::math::count< P >, float, rectangularity< P >> |      |
| >   |      |
| mln::accu::site_set::rectangularity< P > . . . . .                                  | 509  |
| mln::accu::pair< min< V >, max< V >> . . . . .                                      | 499  |
| mln::accu::stat::min_max< V > . . . . .   | 527  |
| mln::Site_Proxy< E > . . . . .  | 965  |
| mln::Pseudo_Site< E > . . . . .   | 957  |
| mln::Site_Iterator< E > . . . . .   | 963  |
| mln::internal::site_iterator_base   |      |
| mln::internal::site_set_iterator_base   |      |
| mln::p_transformed_piter< Pi, S, F > . . . . .                                      | 927  |
| mln::Site< E > . . . . .  | 962  |
| mln::Gpoint< E > . . . . .  | 726  |
| mln::Site_Set< E > . . . . .  | 966  |
| mln::Box< E > . . . . .   | 560  |
| mln::Value< E > . . . . .   | 1107 |
| mln::Value_Set< E > . . . . .   | 1148 |
| mln::Weighted_Window< E > . . . . .   | 1158 |
| mln::Window< E > . . . . .  | 1170 |
| mln::graph_window_base< P, E > . . . . .  | 751  |
| mln::internal::window_base  |      |
| mln::Point_Site< void > . . . . .   | 956  |
| mln::Proxy< void > . . . . .  | 957  |
| mln::Pseudo_Site< void > . . . . .  | 958  |
| mln::registration::closest_point_basic< P > . . . . .                               | 959  |
| mln::registration::closest_point_with_map< P > . . . . .                            | 960  |
| mln::select::p_of< P > . . . . .  | 961  |
| mln::Site< void > . . . . .   | 963  |
| mln::Site_Proxy< void > . . . . .   | 966  |
| mln::Site_Set< void > . . . . .   | 969  |
| mln::thru_image< I, F > . . . . .   | 975  |
| mln::topo::complex< D > . . . . .   | 1004 |
| mln::topo::face< D > . . . . .  | 1007 |
| mln::topo::algebraic_face< D > . . . . .  | 991  |
| mln::topo::is_simple_2d_t< N > . . . . .  | 1015 |
| mln::topo::n_face< N, D > . . . . .   | 1019 |
| mln::topo::algebraic_n_face< N, D > . . . . .                                       | 996  |
| mln::topo::n_faces_set< N, D > . . . . .  | 1026 |
| mln::topo::skeleton::is_simple_point< N > . . . . .                                 | 1027 |
| mln::util::adjacency_matrix< V > . . . . .  | 1035 |
| mln::util::array< T > . . . . .   | 1036 |
| mln::util::branch< T > . . . . .  | 1043 |
| mln::util::branch_iter< T > . . . . .   | 1044 |
| mln::util::branch_iter_ind< T > . . . . .   | 1046 |
| mln::util::greater_point< I > . . . . .   | 1065 |
| mln::util::greater_psite< I > . . . . .   | 1066 |
| mln::util::head< T, R > . . . . .   | 1066 |
| mln::util::ilcell< T > . . . . .  | 1067 |
| mln::util::internal::edge_impl_< G >  |      |
| mln::util::edge< G > . . . . .  | 1050 |

|  |      |
|--|------|
| mln::util::internal::vertex_impl_< G >   |      |
| mln::util::vertex< G >   | 1102 |
| mln::util::node< T, R >  | 1075 |
| mln::util::ord< T >  | 1077 |
| mln::util::pix< I >  | 1080 |
| mln::util::tracked_ptr< T >  | 1093 |
| mln::util::tree< T >   | 1095 |
| mln::util::tree_node< T >  | 1097 |
| mln::value::float01  | 1109 |
| mln::value::Integer< E >   | 1125 |
| mln::value::Integer< void >  | 1126 |
| mln::value::internal::value_like_< V, C, N, E >  |      |
| mln::value::float01_f  | 1111 |
| mln::value::graylevel< n >   | 1112 |
| mln::value::graylevel_f  | 1116 |
| mln::value::int_s< n >   | 1118 |
| mln::value::int_u< n >   | 1121 |
| mln::value::int_u_sat< n >   | 1123 |
| mln::value::label< n >   | 1126 |
| mln::value::qt::rgb32  | 1135 |
| mln::value::rgb< n >   | 1137 |
| mln::value::set< T >   | 1139 |
| mln::value::sign   | 1140 |
| mln::value::super_value< sign >  | 1145 |
| mln::value::value_array< T, V >  | 1145 |
| mln::Vertex< E >   | 1149 |
| mln::internal::neighborhood_base< W, mixed_neighb< W > >   |      |
| neighb_base< W, mixed_neighb< W > >  |      |
| mln::mixed_neighb< W >   | 830  |
| mln::internal::neighborhood_base< W, neighb< W > >   |      |
| neighb_base< W, neighb< W > >  |      |
| mln::neighb< W >   | 842  |
| mln::Object< abs >   | 844  |
| mln::Meta_Function< abs >  | 826  |
| mln::Meta_Function_v2v< abs >  | 827  |
| mln::Object< abs< V > >  | 844  |
| mln::Function< abs< V > >  | 715  |
| mln::Function_v2v< abs< V > >  | 717  |
| mln::Object< accu_result >   | 844  |
| mln::Meta_Function< accu_result >  | 826  |
| mln::Meta_Function_v2v< accu_result >  | 827  |
| mln::Object< adj_higher_dim_connected_n_face_bkd_iter< D > >   | 844  |
| mln::Iterator< adj_higher_dim_connected_n_face_bkd_iter< D > >   | 787  |
| complex_iterator_base< algebraic_face< D >, adj_higher_dim_connected_n_face_bkd_iter< D > >                                    |      |
| complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_dim_connected_n_face_bkd_iter< D > >          |      |
| backward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_dim_connected_n_face_bkd_iter< D > > |      |
| mln::topo::adj_higher_dim_connected_n_face_bkd_iter< D >   | 977  |
| mln::Object< adj_higher_dim_connected_n_face_fwd_iter< D > >   | 844  |
| mln::Iterator< adj_higher_dim_connected_n_face_fwd_iter< D > >   | 787  |

|  |     |
|--|-----|
| complex_iterator_base< algebraic_face< D >, adj_higher_dim_connected_n_face_fwd_iter< D >>                                   |     |
| complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_dim_connected_n_face_fwd_iter< D >>         |     |
| forward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_dim_connected_n_face_fwd_iter< D >> |     |
| mln::topo::adj_higher_dim_connected_n_face_fwd_iter< D > . . . . .   | 978 |
| mln::Object< adj_higher_face_bkd_iter< D >> . . . . .  | 844 |
| mln::Iterator< adj_higher_face_bkd_iter< D >> . . . . .  | 787 |
| complex_iterator_base< algebraic_face< D >, adj_higher_face_bkd_iter< D >>   |     |
| complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_face_bkd_iter< D >>                         |     |
| backward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_face_bkd_iter< D >>                |     |
| mln::topo::adj_higher_face_bkd_iter< D > . . . . .   | 979 |
| mln::Object< adj_higher_face_fwd_iter< D >> . . . . .  | 844 |
| mln::Iterator< adj_higher_face_fwd_iter< D >> . . . . .  | 787 |
| complex_iterator_base< algebraic_face< D >, adj_higher_face_fwd_iter< D >>   |     |
| complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_face_fwd_iter< D >>                         |     |
| forward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_face_fwd_iter< D >>                 |     |
| mln::topo::adj_higher_face_fwd_iter< D > . . . . .   | 980 |
| mln::Object< adj_lower_dim_connected_n_face_bkd_iter< D >> . . . . .   | 844 |
| mln::Iterator< adj_lower_dim_connected_n_face_bkd_iter< D >> . . . . .   | 787 |
| complex_iterator_base< algebraic_face< D >, adj_lower_dim_connected_n_face_bkd_iter< D >>                                    |     |
| complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_lower_dim_connected_n_face_bkd_iter< D >>          |     |
| backward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_lower_dim_connected_n_face_bkd_iter< D >> |     |
| mln::topo::adj_lower_dim_connected_n_face_bkd_iter< D > . . . . .  | 981 |
| mln::Object< adj_lower_dim_connected_n_face_fwd_iter< D >> . . . . .   | 844 |
| mln::Iterator< adj_lower_dim_connected_n_face_fwd_iter< D >> . . . . .   | 787 |
| complex_iterator_base< algebraic_face< D >, adj_lower_dim_connected_n_face_fwd_iter< D >>                                    |     |
| complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_lower_dim_connected_n_face_fwd_iter< D >>          |     |
| forward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_lower_dim_connected_n_face_fwd_iter< D >>  |     |
| mln::topo::adj_lower_dim_connected_n_face_fwd_iter< D > . . . . .  | 982 |
| mln::Object< adj_lower_face_bkd_iter< D >> . . . . .   | 844 |
| mln::Iterator< adj_lower_face_bkd_iter< D >> . . . . .   | 787 |
| complex_iterator_base< algebraic_face< D >, adj_lower_face_bkd_iter< D >>  |     |
| complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_lower_face_bkd_iter< D >>                          |     |
| backward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_lower_face_bkd_iter< D >>                 |     |
| mln::topo::adj_lower_face_bkd_iter< D > . . . . .  | 984 |
| mln::Object< adj_lower_face_fwd_iter< D >> . . . . .   | 844 |
| mln::Iterator< adj_lower_face_fwd_iter< D >> . . . . .   | 787 |
| complex_iterator_base< algebraic_face< D >, adj_lower_face_fwd_iter< D >>  |     |

|  |      |
|--|------|
| complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_lower_face_  |      |
| fwd_iter< D > >  |      |
| forward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_     |      |
| lower_face_fwd_iter< D > >   |      |
| mln::topo::adj_lower_face_fwd_iter< D > . . . . .                                      | 985  |
| mln::Object< adj_lower_higher_face_bkd_iter< D > > . . . . .                           | 844  |
| mln::Iterator< adj_lower_higher_face_bkd_iter< D > > . . . . .                         | 787  |
| complex_relative_iterator_sequence< adj_higher_face_bkd_iter< D >, adj_lower_face_bkd_ |      |
| iter< D >, adj_lower_higher_face_bkd_iter< D > >                                       |      |
| mln::topo::adj_lower_higher_face_bkd_iter< D > . . . . .                               | 986  |
| mln::Object< adj_lower_higher_face_fwd_iter< D > > . . . . .                           | 844  |
| mln::Iterator< adj_lower_higher_face_fwd_iter< D > > . . . . .                         | 787  |
| complex_relative_iterator_sequence< adj_lower_face_fwd_iter< D >, adj_higher_face_fwd_ |      |
| iter< D >, adj_lower_higher_face_fwd_iter< D > >                                       |      |
| mln::topo::adj_lower_higher_face_fwd_iter< D > . . . . .                               | 987  |
| mln::Object< adj_m_face_bkd_iter< D > > . . . . .                                      | 844  |
| mln::Iterator< adj_m_face_bkd_iter< D > > . . . . .                                    | 787  |
| complex_iterator_base< algebraic_face< D >, adj_m_face_bkd_iter< D > >                 |      |
| complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_m_face_      |      |
| bkd_iter< D > >  |      |
| backward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >,         |      |
| adj_m_face_bkd_iter< D > >   |      |
| mln::topo::adj_m_face_bkd_iter< D > . . . . .  | 988  |
| mln::Object< adj_m_face_fwd_iter< D > > . . . . .                                      | 844  |
| mln::Iterator< adj_m_face_fwd_iter< D > > . . . . .                                    | 787  |
| complex_iterator_base< algebraic_face< D >, adj_m_face_fwd_iter< D > >                 |      |
| complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_m_face_      |      |
| fwd_iter< D > >  |      |
| forward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_     |      |
| m_face_fwd_iter< D > >   |      |
| mln::topo::adj_m_face_fwd_iter< D > . . . . .  | 989  |
| mln::Object< all_to< T > > . . . . .   | 844  |
| mln::Function< all_to< T > > . . . . .   | 715  |
| mln::Function_v2v< all_to< T > > . . . . .   | 717  |
| mln::Object< antilogy > . . . . .  | 844  |
| mln::Function< antilogy > . . . . .  | 715  |
| mln::Function_v2v< antilogy > . . . . .  | 717  |
| mln::Function_v2b< antilogy > . . . . .  | 716  |
| mln::fun::p2b::antilogy . . . . .  | 672  |
| mln::Object< array1d< T, Size > > . . . . .  | 844  |
| mln::Object< array2d< T, r, c > > . . . . .  | 844  |
| mln::Object< array3d< T, s, r, c > > . . . . .   | 844  |
| mln::Object< array_bkd_iter< T > > . . . . .   | 844  |
| mln::Proxy< array_bkd_iter< T > > . . . . .  | 957  |
| mln::Object< array_fwd_iter< T > > . . . . .   | 844  |
| mln::Proxy< array_fwd_iter< T > > . . . . .  | 957  |
| mln::Object< asc_propagation > . . . . .   | 844  |
| mln::Object< backdiag2d > . . . . .  | 844  |
| mln::Window< backdiag2d > . . . . .  | 1170 |
| window_base< dpoint2d, backdiag2d >  |      |

|   |      |
|---|------|
| classical_window_base< dpoint2d, backdiag2d >         |      |
| mln::win::backdiag2d                                  | 1159 |
| mln::Object< backdiagonal2d_t >                       | 844  |
| mln::Browsing< backdiagonal2d_t >                     | 568  |
| mln::canvas::browsing::backdiagonal2d_t               | 569  |
| mln::Object< ball< G, C > >                           | 844  |
| mln::Window< ball< G, C > >                           | 1170 |
| window_base< dpoint< G, C >, ball< G, C > >           |      |
| classical_window_base< dpoint< G, C >, ball< G, C > > |      |
| mln::win::ball< G, C >                                | 1160 |
| mln::Object< bbox >                                   | 844  |
| mln::Meta_Accumulator< bbox >                         | 824  |
| mln::accu::meta::shape::bbox                          | 482  |
| mln::Object< bbox< P > >                              | 844  |
| mln::Proxy< bbox< P > >                               | 957  |
| mln::Accumulator< bbox< P > >                         | 542  |
| base< const box< P > &, bbox< P > >                   |      |
| mln::accu::shape::bbox< P >                           | 504  |
| mln::Object< big_chess< B > >                         | 844  |
| mln::Function< big_chess< B > >                       | 715  |
| mln::Function_v2v< big_chess< B > >                   | 717  |
| mln::Function_v2b< big_chess< B > >                   | 716  |
| mln::Object< bin_off_loader >                         | 844  |
| mln::Object< bin_off_saver >                          | 844  |
| mln::Object< binary< Fun, T1, T2 > >                  | 844  |
| mln::Function< binary< Fun, T1, T2 > >                | 715  |
| mln::Function_v2v< binary< Fun, T1, T2 > >            | 717  |
| mln::Object< bkd_pixter1d< I > >                      | 844  |
| mln::Iterator< bkd_pixter1d< I > >                    | 787  |
| mln::Pixel_Iterator< bkd_pixter1d< I > >              | 940  |
| mln::Object< bkd_pixter2d< I > >                      | 844  |
| mln::Iterator< bkd_pixter2d< I > >                    | 787  |
| mln::Pixel_Iterator< bkd_pixter2d< I > >              | 940  |
| mln::Object< bkd_pixter3d< I > >                      | 844  |
| mln::Iterator< bkd_pixter3d< I > >                    | 787  |
| mln::Pixel_Iterator< bkd_pixter3d< I > >              | 940  |
| mln::Object< black_t >                                | 844  |
| mln::Literal< black_t >                               | 799  |
| mln::literal::black_t                                 | 801  |
| mln::Object< blue >                                   | 844  |
| mln::Meta_Function< blue >                            | 826  |
| mln::Meta_Function_v2v< blue >                        | 827  |
| mln::Object< blue_t >                                 | 844  |
| mln::Literal< blue_t >                                | 799  |
| mln::literal::blue_t                                  | 802  |
| mln::Object< box< P > >                               | 844  |
| mln::Site_Set< box< P > >                             | 966  |
| mln::Box< box< P > >                                  | 560  |

|  |      |
|--|------|
| mln::box< P > . . . . .  | 552  |
| mln::Object< box_runend_piter< P > > . . . . .   | 844  |
| mln::Proxy< box_runend_piter< P > > . . . . .  | 957  |
| mln::Site_Proxy< box_runend_piter< P > > . . . . .                                       | 965  |
| mln::Site_Iterator< box_runend_piter< P > > . . . . .                                    | 963  |
| site_iterator_base< box< P >, box_runend_piter< P > >                                    |      |
| site_set_iterator_base< box< P >, box_runend_piter< P > >                                |      |
| mln::box_runend_piter< P > . . . . .   | 566  |
| mln::Object< box_runstart_piter< P > > . . . . .   | 844  |
| mln::Proxy< box_runstart_piter< P > > . . . . .  | 957  |
| mln::Site_Proxy< box_runstart_piter< P > > . . . . .                                     | 965  |
| mln::Site_Iterator< box_runstart_piter< P > > . . . . .                                  | 963  |
| site_iterator_base< box< P >, box_runstart_piter< P > >                                  |      |
| site_set_iterator_base< box< P >, box_runstart_piter< P > >                              |      |
| mln::box_runstart_piter< P > . . . . .   | 567  |
| mln::Object< breadth_first_search_t > . . . . .  | 844  |
| mln::Browsing< breadth_first_search_t > . . . . .  | 568  |
| graph_first_search_t< breadth_first_search_t, std::queue >                               |      |
| mln::canvas::browsing::breadth_first_search_t . . . . .                                  | 571  |
| mln::Object< brown_t > . . . . .   | 844  |
| mln::Literal< brown_t > . . . . .  | 799  |
| mln::literal::brown_t . . . . .  | 802  |
| mln::Object< card< I > > . . . . .   | 844  |
| mln::Proxy< card< I > > . . . . .  | 957  |
| mln::Accumulator< card< I > > . . . . .  | 542  |
| base< unsigned, card< I > >  |      |
| mln::morpho::attribute::card< I > . . . . .  | 832  |
| mln::Object< cast< V > > . . . . .   | 844  |
| mln::Function< cast< V > > . . . . .   | 715  |
| mln::Function_v2v< cast< V > > . . . . .   | 717  |
| mln::Object< center > . . . . .  | 844  |
| mln::Meta_Accumulator< center > . . . . .  | 824  |
| mln::accu::meta::center . . . . .  | 463  |
| mln::Object< center< P, V > > . . . . .  | 844  |
| mln::Proxy< center< P, V > > . . . . .   | 957  |
| mln::Accumulator< center< P, V > > . . . . .   | 542  |
| base< V, center< P, V > >  |      |
| mln::accu::center< P, V > . . . . .  | 437  |
| mln::Object< center_only_iter< D > > . . . . .   | 844  |
| mln::Iterator< center_only_iter< D > > . . . . .   | 787  |
| complex_iterator_base< algebraic_face< D >, center_only_iter< D > >                      |      |
| complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, center_only_       |      |
| iter< D > >  |      |
| forward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >,            |      |
| center_only_iter< D > >  |      |
| mln::topo::center_only_iter< D > . . . . .   | 1001 |
| mln::Object< centered_bkd_iter_adapter< D, I > > . . . . .                               | 844  |
| mln::Iterator< centered_bkd_iter_adapter< D, I > > . . . . .                             | 787  |
| complex_relative_iterator_sequence< I, center_only_iter< D >, centered_bkd_iter_adapter< |      |
| D, I > >   |      |

|   |      |
|---|------|
| mln::topo::centered_bkd_iter_adapter< D, I > . . . . .  | 1002 |
| mln::Object< centered_fwd_iter_adapter< D, I > > . . . . .  | 844  |
| mln::Iterator< centered_fwd_iter_adapter< D, I > > . . . . .                                      | 787  |
| complex_relative_iterator_sequence< center_only_iter< D >, I, centered_fwd_iter_adapter< D, I > > |      |
| mln::topo::centered_fwd_iter_adapter< D, I > . . . . .  | 1003 |
| mln::Object< ch_function_value< F, V > > . . . . .  | 844  |
| mln::Function< ch_function_value< F, V > > . . . . .  | 715  |
| mln::Function_v2v< ch_function_value< F, V > > . . . . .  | 717  |
| mln::fun::v2v::ch_function_value< F, V > . . . . .  | 677  |
| mln::Object< ch_piter_image< I, Fwd > > . . . . .   | 844  |
| mln::Image< ch_piter_image< I, Fwd > > . . . . .  | 762  |
| mln::Object< chess > . . . . .  | 844  |
| mln::Function< chess > . . . . .  | 715  |
| mln::Function_v2v< chess > . . . . .  | 717  |
| mln::Function_v2b< chess > . . . . .  | 716  |
| mln::Object< col > . . . . .  | 844  |
| mln::Meta_Function< col > . . . . .   | 826  |
| mln::Meta_Function_v2v< col > . . . . .   | 827  |
| mln::Object< colorize > . . . . .   | 844  |
| mln::Function< colorize > . . . . .   | 715  |
| mln::Function_v2v< colorize > . . . . .   | 717  |
| mln::Object< comp > . . . . .   | 844  |
| mln::Meta_Function< comp > . . . . .  | 826  |
| mln::Meta_Function_v2v< comp > . . . . .  | 827  |
| mln::Object< comp_count > . . . . .   | 844  |
| mln::Meta_Function< comp_count > . . . . .  | 826  |
| mln::Meta_Function_v2v< comp_count > . . . . .  | 827  |
| mln::Object< complex_image< D, G, V > > . . . . .   | 844  |
| mln::Image< complex_image< D, G, V > > . . . . .  | 762  |
| mln::Object< complex_neighborhood_bkd_piter< I, G, N > > . . . . .                                | 844  |
| mln::Proxy< complex_neighborhood_bkd_piter< I, G, N > > . . . . .                                 | 957  |
| mln::Site_Proxy< complex_neighborhood_bkd_piter< I, G, N > > . . . . .                            | 965  |
| mln::Site_Iterator< complex_neighborhood_bkd_piter< I, G, N > > . . . . .                         | 963  |
| site_iterator_base< N, complex_neighborhood_bkd_piter< I, G, N > >                                |      |
| site_relative_iterator_base< N, complex_neighborhood_bkd_piter< I, G, N > >                       |      |
| mln::complex_neighborhood_bkd_piter< I, G, N > . . . . .  | 587  |
| mln::Object< complex_neighborhood_fwd_piter< I, G, N > > . . . . .                                | 844  |
| mln::Proxy< complex_neighborhood_fwd_piter< I, G, N > > . . . . .                                 | 957  |
| mln::Site_Proxy< complex_neighborhood_fwd_piter< I, G, N > > . . . . .                            | 965  |
| mln::Site_Iterator< complex_neighborhood_fwd_piter< I, G, N > > . . . . .                         | 963  |
| site_iterator_base< N, complex_neighborhood_fwd_piter< I, G, N > >                                |      |
| site_relative_iterator_base< N, complex_neighborhood_fwd_piter< I, G, N > >                       |      |
| mln::complex_neighborhood_fwd_piter< I, G, N > . . . . .  | 588  |
| mln::Object< complex_psite< D, G > > . . . . .  | 844  |
| mln::Proxy< complex_psite< D, G > > . . . . .   | 957  |
| mln::Site_Proxy< complex_psite< D, G > > . . . . .  | 965  |
| mln::Pseudo_Site< complex_psite< D, G > > . . . . .   | 957  |
| pseudo_site_base_< const G::site &, complex_psite< D, G > >                                       |      |



|   |     |
|---|-----|
| mln::complex_psite< D, G > . . . . .  | 590 |
| mln::Object< complex_window_bkd_piter< I, G, W > > . . . . .  | 844 |
| mln::Proxy< complex_window_bkd_piter< I, G, W > > . . . . .   | 957 |
| mln::Site_Proxy< complex_window_bkd_piter< I, G, W > > . . . . .  | 965 |
| mln::Site_Iterator< complex_window_bkd_piter< I, G, W > > . . . . .                                       | 963 |
| site_iterator_base< W, complex_window_bkd_piter< I, G, W > >  |     |
| site_relative_iterator_base< W, complex_window_bkd_piter< I, G, W > >                                     |     |
| mln::complex_window_bkd_piter< I, G, W > . . . . .  | 593 |
| mln::Object< complex_window_fwd_piter< I, G, W > > . . . . .  | 844 |
| mln::Proxy< complex_window_fwd_piter< I, G, W > > . . . . .   | 957 |
| mln::Site_Proxy< complex_window_fwd_piter< I, G, W > > . . . . .  | 965 |
| mln::Site_Iterator< complex_window_fwd_piter< I, G, W > > . . . . .                                       | 963 |
| site_iterator_base< W, complex_window_fwd_piter< I, G, W > >  |     |
| site_relative_iterator_base< W, complex_window_fwd_piter< I, G, W > >                                     |     |
| mln::complex_window_fwd_piter< I, G, W > . . . . .  | 595 |
| mln::Object< component< T, i > > . . . . .  | 844 |
| mln::Function< component< T, i > > . . . . .  | 715 |
| mln::Function_v2v< component< T, i > > . . . . .  | 717 |
| mln::fun::v2v::component< T, i > . . . . .  | 678 |
| mln::Object< compose > . . . . .  | 844 |
| mln::Meta_Function< compose > . . . . .   | 826 |
| mln::Meta_Function_vv2v< compose > . . . . .  | 827 |
| mln::Object< composition< mln::Meta_Function_v2v, F, mln::Meta_Function_v2v, G > > . . . . .              | 844 |
| mln::Meta_Function< composition< mln::Meta_Function_v2v, F, mln::Meta_Function_v2v, G > > . . . . .       | 826 |
| mln::Meta_Function_v2v< composition< mln::Meta_Function_v2v, F, mln::Meta_Function_v2v, G > > . . . . .   | 827 |
| mln::Object< composition< mln::Meta_Function_v2v, F, mln::Meta_Function_vv2v, G > > . . . . .             | 844 |
| mln::Meta_Function< composition< mln::Meta_Function_v2v, F, mln::Meta_Function_vv2v, G > > . . . . .      | 826 |
| mln::Meta_Function_vv2v< composition< mln::Meta_Function_v2v, F, mln::Meta_Function_vv2v, G > > . . . . . | 827 |
| mln::Object< concrete > . . . . .   | 844 |
| mln::Object< convert< V > > . . . . .   | 844 |
| mln::Function< convert< V > > . . . . .   | 715 |
| mln::Function_v2v< convert< V > > . . . . .   | 717 |
| mln::Object< convolve< T1, T2, R > > . . . . .  | 844 |
| mln::Proxy< convolve< T1, T2, R > > . . . . .   | 957 |
| mln::Accumulator< convolve< T1, T2, R > > . . . . .   | 542 |
| base< R, convolve< T1, T2, R > >  |     |
| mln::accu::convolve< T1, T2, R > . . . . .  | 439 |
| mln::Object< cos > . . . . .  | 844 |
| mln::Meta_Function< cos > . . . . .   | 826 |
| mln::Meta_Function_v2v< cos > . . . . .   | 827 |
| mln::Object< cos< V > > . . . . .   | 844 |
| mln::Function< cos< V > > . . . . .   | 715 |
| mln::Function_v2v< cos< V > > . . . . .   | 717 |
| mln::fun::v2w2v::cos< V > . . . . .   | 684 |
| mln::Object< count > . . . . .  | 844 |

|  |      |
|--|------|
| mln::Meta_Accumulator< count > . . . . .                       | 824  |
| mln::accu::meta::math::count . . . . .                         | 473  |
| mln::Object< count< T > > . . . . .                            | 844  |
| mln::Proxy< count< T > > . . . . .                             | 957  |
| mln::Accumulator< count< T > > . . . . .                       | 542  |
| base< unsigned, count< T > >                                   |      |
| mln::accu::math::count< T > . . . . .                          | 455  |
| mln::Object< count_adjacent_vertices > . . . . .               | 844  |
| mln::Meta_Accumulator< count_adjacent_vertices > . . . . .     | 824  |
| mln::accu::meta::count_adjacent_vertices . . . . .             | 463  |
| mln::Object< count_adjacent_vertices< F, S > > . . . . .       | 844  |
| mln::Proxy< count_adjacent_vertices< F, S > > . . . . .        | 957  |
| mln::Accumulator< count_adjacent_vertices< F, S > > . . . . .  | 542  |
| base< unsigned, count_adjacent_vertices< F, S > >              |      |
| mln::accu::count_adjacent_vertices< F, S > . . . . .           | 440  |
| mln::Object< count_adjacent_vertices< I > > . . . . .          | 844  |
| mln::Proxy< count_adjacent_vertices< I > > . . . . .           | 957  |
| mln::Accumulator< count_adjacent_vertices< I > > . . . . .     | 542  |
| base< unsigned, count_adjacent_vertices< I > >                 |      |
| mln::morpho::attribute::count_adjacent_vertices< I > . . . . . | 833  |
| mln::Object< count_labels > . . . . .                          | 844  |
| mln::Meta_Accumulator< count_labels > . . . . .                | 824  |
| mln::accu::meta::count_labels . . . . .                        | 464  |
| mln::Object< count_labels< L > > . . . . .                     | 844  |
| mln::Proxy< count_labels< L > > . . . . .                      | 957  |
| mln::Accumulator< count_labels< L > > . . . . .                | 542  |
| base< unsigned, count_labels< L > >                            |      |
| mln::accu::count_labels< L > . . . . .                         | 442  |
| mln::Object< count_value > . . . . .                           | 844  |
| mln::Meta_Accumulator< count_value > . . . . .                 | 824  |
| mln::accu::meta::count_value . . . . .                         | 465  |
| mln::Object< count_value< V > > . . . . .                      | 844  |
| mln::Proxy< count_value< V > > . . . . .                       | 957  |
| mln::Accumulator< count_value< V > > . . . . .                 | 542  |
| base< unsigned, count_value< V > >                             |      |
| mln::accu::count_value< V > . . . . .                          | 443  |
| mln::Object< couple< T, U > > . . . . .                        | 844  |
| mln::util::couple< T, U > . . . . .                            | 1047 |
| mln::Object< cube > . . . . .                                  | 844  |
| mln::Mesh< cube > . . . . .                                    | 823  |
| mln::Regular_Grid< cube > . . . . .                            | 960  |
| mln::Object< cube3d > . . . . .                                | 844  |
| mln::Window< cube3d > . . . . .                                | 1170 |
| window_base< dpoint3d, cube3d >                                |      |
| classical_window_base< dpoint3d, cube3d >                      |      |
| mln::win::cube3d . . . . .                                     | 1161 |
| mln::Object< cuboid3d > . . . . .                              | 844  |
| mln::Window< cuboid3d > . . . . .                              | 1170 |
| window_base< dpoint3d, cuboid3d >                              |      |

|   |      |
|---|------|
| classical_window_base< dpoint3d, cuboid3d >                   |      |
| mln::win::cuboid3d . . . . .                                  | 1162 |
| mln::Object< cyan_t > . . . . .                               | 844  |
| mln::Literal< cyan_t > . . . . .                              | 799  |
| mln::literal::cyan_t . . . . .                                | 803  |
| mln::Object< d_t > . . . . .                                  | 844  |
| mln::Function< d_t > . . . . .                                | 715  |
| mln::Function_vv2v< d_t > . . . . .                           | 718  |
| mln::Object< dark_gray_t > . . . . .                          | 844  |
| mln::Literal< dark_gray_t > . . . . .                         | 799  |
| mln::Object< dashed_line_f< I, dim > > . . . . .              | 844  |
| mln::Function< dashed_line_f< I, dim > > . . . . .            | 715  |
| mln::Function_v2v< dashed_line_f< I, dim > > . . . . .        | 717  |
| mln::Function_v2b< dashed_line_f< I, dim > > . . . . .        | 716  |
| mln::Object< decorated_image< I, D > > . . . . .              | 844  |
| mln::Image< decorated_image< I, D > > . . . . .               | 762  |
| mln::Object< depth1st_piter< T > > . . . . .                  | 844  |
| mln::Proxy< depth1st_piter< T > > . . . . .                   | 957  |
| mln::Site_Proxy< depth1st_piter< T > > . . . . .              | 965  |
| mln::Site_Iterator< depth1st_piter< T > > . . . . .           | 963  |
| mln::Object< depth_first_search_t > . . . . .                 | 844  |
| mln::Browsing< depth_first_search_t > . . . . .               | 568  |
| graph_first_search_t< depth_first_search_t, std::stack >      |      |
| mln::canvas::browsing::depth_first_search_t . . . . .         | 571  |
| mln::Object< desc_propagation > . . . . .                     | 844  |
| mln::Object< deviation > . . . . .                            | 844  |
| mln::Meta_Accumulator< deviation > . . . . .                  | 824  |
| mln::accu::stat::meta::deviation . . . . .                    | 523  |
| mln::Object< deviation< T, S, M > > . . . . .                 | 844  |
| mln::Proxy< deviation< T, S, M > > . . . . .                  | 957  |
| mln::Accumulator< deviation< T, S, M > > . . . . .            | 542  |
| base< M, deviation< T, S, M > >                               |      |
| mln::accu::stat::deviation< T, S, M > . . . . .               | 511  |
| mln::Object< diag2d > . . . . .                               | 844  |
| mln::Window< diag2d > . . . . .                               | 1170 |
| window_base< dpoint2d, diag2d >                               |      |
| classical_window_base< dpoint2d, diag2d >                     |      |
| mln::win::diag2d . . . . .                                    | 1164 |
| mln::Object< diagonal2d_t > . . . . .                         | 844  |
| mln::Browsing< diagonal2d_t > . . . . .                       | 568  |
| mln::canvas::browsing::diagonal2d_t . . . . .                 | 571  |
| mln::Object< diff_abs< V > > . . . . .                        | 844  |
| mln::Function< diff_abs< V > > . . . . .                      | 715  |
| mln::Function_vv2v< diff_abs< V > > . . . . .                 | 718  |
| mln::fun::vv2v::diff_abs< V > . . . . .                       | 694  |
| mln::Object< dir_struct_elt_incr_update_t > . . . . .         | 844  |
| mln::Browsing< dir_struct_elt_incr_update_t > . . . . .       | 568  |
| mln::canvas::browsing::dir_struct_elt_incr_update_t . . . . . | 573  |

|  |      |
|--|------|
| mln::Object< directional_t > . . . . .                   | 844  |
| mln::Browsing< directional_t > . . . . .                 | 568  |
| mln::canvas::browsing::directional_t . . . . .           | 574  |
| mln::Object< dist > . . . . .                            | 844  |
| mln::Function< dist > . . . . .                          | 715  |
| mln::Function_vv2v< dist > . . . . .                     | 718  |
| mln::Object< dist_t > . . . . .                          | 844  |
| mln::Function< dist_t > . . . . .                        | 715  |
| mln::Function_vv2v< dist_t > . . . . .                   | 718  |
| mln::Object< dn_leaf_piter< T > > . . . . .              | 844  |
| mln::Proxy< dn_leaf_piter< T > > . . . . .               | 957  |
| mln::Site_Proxy< dn_leaf_piter< T > > . . . . .          | 965  |
| mln::Site_Iterator< dn_leaf_piter< T > > . . . . .       | 963  |
| mln::Object< dn_node_piter< T > > . . . . .              | 844  |
| mln::Proxy< dn_node_piter< T > > . . . . .               | 957  |
| mln::Site_Proxy< dn_node_piter< T > > . . . . .          | 965  |
| mln::Site_Iterator< dn_node_piter< T > > . . . . .       | 963  |
| mln::Object< dn_site_piter< T > > . . . . .              | 844  |
| mln::Proxy< dn_site_piter< T > > . . . . .               | 957  |
| mln::Site_Proxy< dn_site_piter< T > > . . . . .          | 965  |
| mln::Site_Iterator< dn_site_piter< T > > . . . . .       | 963  |
| mln::Object< dpoint< G, C > > . . . . .                  | 844  |
| mln::Gdpoint< dpoint< G, C > > . . . . .                 | 723  |
| mln::dpoint< G, C > . . . . .                            | 644  |
| mln::Object< dpoints_bkd_pixter< I > > . . . . .         | 844  |
| mln::Iterator< dpoints_bkd_pixter< I > > . . . . .       | 787  |
| mln::Pixel_Iterator< dpoints_bkd_pixter< I > > . . . . . | 940  |
| mln::Object< dpoints_fwd_pixter< I > > . . . . .         | 844  |
| mln::Iterator< dpoints_fwd_pixter< I > > . . . . .       | 787  |
| mln::Pixel_Iterator< dpoints_fwd_pixter< I > > . . . . . | 940  |
| mln::Object< dpsites_bkd_piter< V > > . . . . .          | 844  |
| mln::Proxy< dpsites_bkd_piter< V > > . . . . .           | 957  |
| mln::Site_Proxy< dpsites_bkd_piter< V > > . . . . .      | 965  |
| mln::Site_Iterator< dpsites_bkd_piter< V > > . . . . .   | 963  |
| mln::Object< dpsites_fwd_piter< V > > . . . . .          | 844  |
| mln::Proxy< dpsites_fwd_piter< V > > . . . . .           | 957  |
| mln::Site_Proxy< dpsites_fwd_piter< V > > . . . . .      | 965  |
| mln::Site_Iterator< dpsites_fwd_piter< V > > . . . . .   | 963  |
| mln::Object< eat > . . . . .                             | 844  |
| mln::util::eat . . . . .                                 | 1049 |
| mln::Object< edge_bkd_iterator< G > > . . . . .          | 844  |
| mln::Proxy< edge_bkd_iterator< G > > . . . . .           | 957  |
| mln::Object< edge_fwd_iterator< G > > . . . . .          | 844  |
| mln::Proxy< edge_fwd_iterator< G > > . . . . .           | 957  |
| mln::Object< edge_image< P, V, G > > . . . . .           | 844  |
| mln::Image< edge_image< P, V, G > > . . . . .            | 762  |
| mln::Object< edge_nbh_edge_bkd_iterator< G > > . . . . . | 844  |
| mln::Proxy< edge_nbh_edge_bkd_iterator< G > > . . . . .  | 957  |

|  |     |
|--|-----|
| mln::Object< edge_nbh_edge_fwd_iterator< G > > . . . . . | 844 |
| mln::Proxy< edge_nbh_edge_fwd_iterator< G > > . . . . .  | 957 |
| mln::Object< edge_to_color< I, V > > . . . . .           | 844 |
| mln::Function< edge_to_color< I, V > > . . . . .         | 715 |
| mln::Function_v2v< edge_to_color< I, V > > . . . . .     | 717 |
| mln::Object< enc< V > > . . . . .                        | 844 |
| mln::Function< enc< V > > . . . . .                      | 715 |
| mln::Function_v2v< enc< V > > . . . . .                  | 717 |
| mln::Object< eq< L, R > > . . . . .                      | 844 |
| mln::Function< eq< L, R > > . . . . .                    | 715 |
| mln::Function_vv2b< eq< L, R > > . . . . .               | 718 |
| mln::fun::vv2b::eq< L, R > . . . . .                     | 689 |
| mln::Object< extended< I > > . . . . .                   | 844 |
| mln::Image< extended< I > > . . . . .                    | 762 |
| mln::Object< extension_fun< I, F > > . . . . .           | 844 |
| mln::Image< extension_fun< I, F > > . . . . .            | 762 |
| mln::Object< extension_ima< I, J > > . . . . .           | 844 |
| mln::Image< extension_ima< I, J > > . . . . .            | 762 |
| mln::Object< extension_val< I > > . . . . .              | 844 |
| mln::Image< extension_val< I > > . . . . .               | 762 |
| mln::Object< f_16_to_8 > . . . . .                       | 844 |
| mln::Function< f_16_to_8 > . . . . .                     | 715 |
| mln::Function_v2v< f_16_to_8 > . . . . .                 | 717 |
| mln::Object< f_box1d_t > . . . . .                       | 844 |
| mln::Function< f_box1d_t > . . . . .                     | 715 |
| mln::Function_v2v< f_box1d_t > . . . . .                 | 717 |
| mln::Function_v2b< f_box1d_t > . . . . .                 | 716 |
| mln::Object< f_box2d_t > . . . . .                       | 844 |
| mln::Function< f_box2d_t > . . . . .                     | 715 |
| mln::Function_v2v< f_box2d_t > . . . . .                 | 717 |
| mln::Function_v2b< f_box2d_t > . . . . .                 | 716 |
| mln::Object< f_box3d_t > . . . . .                       | 844 |
| mln::Function< f_box3d_t > . . . . .                     | 715 |
| mln::Function_v2v< f_box3d_t > . . . . .                 | 717 |
| mln::Function_v2b< f_box3d_t > . . . . .                 | 716 |
| mln::Object< f_hsi_to_rgb< T_rgb > > . . . . .           | 844 |
| mln::Function< f_hsi_to_rgb< T_rgb > > . . . . .         | 715 |
| mln::Function_v2v< f_hsi_to_rgb< T_rgb > > . . . . .     | 717 |
| mln::Object< f_hsl_to_rgb< T_rgb > > . . . . .           | 844 |
| mln::Function< f_hsl_to_rgb< T_rgb > > . . . . .         | 715 |
| mln::Function_v2v< f_hsl_to_rgb< T_rgb > > . . . . .     | 717 |
| mln::Object< f_rgb_to_hsi< T_hsi > > . . . . .           | 844 |
| mln::Function< f_rgb_to_hsi< T_hsi > > . . . . .         | 715 |
| mln::Function_v2v< f_rgb_to_hsi< T_hsi > > . . . . .     | 717 |
| mln::Object< f_rgb_to_hsl< T_hsl > > . . . . .           | 844 |
| mln::Function< f_rgb_to_hsl< T_hsl > > . . . . .         | 715 |
| mln::Function_v2v< f_rgb_to_hsl< T_hsl > > . . . . .     | 717 |
| mln::Object< face_bkd_iter< D > > . . . . .              | 844 |

|   |      |
|---|------|
| mln::Iterator< face_bkd_iter< D > > . . . . .                                   | 787  |
| complex_iterator_base< topo::face< D >, face_bkd_iter< D > >                    |      |
| complex_set_iterator_base< topo::face< D >, face_bkd_iter< D > >                |      |
| mln::topo::face_bkd_iter< D > . . . . .   | 1012 |
| mln::Object< face_fwd_iter< D > > . . . . .                                     | 844  |
| mln::Iterator< face_fwd_iter< D > > . . . . .                                   | 787  |
| complex_iterator_base< topo::face< D >, face_fwd_iter< D > >                    |      |
| complex_set_iterator_base< topo::face< D >, face_fwd_iter< D > >                |      |
| mln::topo::face_fwd_iter< D > . . . . .   | 1013 |
| mln::Object< faces_psite< N, D, P > > . . . . .                                 | 844  |
| mln::Proxy< faces_psite< N, D, P > > . . . . .                                  | 957  |
| mln::Site_Proxy< faces_psite< N, D, P > > . . . . .                             | 965  |
| mln::Pseudo_Site< faces_psite< N, D, P > > . . . . .                            | 957  |
| pseudo_site_base_< const P &, faces_psite< N, D, P > >                          |      |
| mln::faces_psite< N, D, P > . . . . .   | 666  |
| mln::Object< fibonacci_heap< P, T > > . . . . .                                 | 844  |
| mln::util::fibonacci_heap< P, T > . . . . .                                     | 1054 |
| mln::Object< flat_image< T, S > > . . . . .                                     | 844  |
| mln::Image< flat_image< T, S > > . . . . .                                      | 762  |
| mln::Object< float01 > . . . . .  | 844  |
| mln::Value< float01 > . . . . .   | 1107 |
| mln::Object< float01_f > . . . . .  | 844  |
| mln::Value< float01_f > . . . . .   | 1107 |
| mln::Object< float_off_loader > . . . . .                                       | 844  |
| mln::Object< float_off_saver > . . . . .  | 844  |
| mln::Object< fold< P, dir_0, dir_1, dir_2 > > . . . . .                         | 844  |
| mln::Function< fold< P, dir_0, dir_1, dir_2 > > . . . . .                       | 715  |
| mln::Function_v2v< fold< P, dir_0, dir_1, dir_2 > > . . . . .                   | 717  |
| mln::Object< from_accu< A > > . . . . .   | 844  |
| mln::Meta_Function< from_accu< A > > . . . . .                                  | 826  |
| mln::Meta_Function_v2v< from_accu< A > > . . . . .                              | 827  |
| mln::Object< fun_image< F, I > > . . . . .                                      | 844  |
| mln::Image< fun_image< F, I > > . . . . .                                       | 762  |
| mln::Object< function< meta::blue< value::rgb< n > > > > . . . . .              | 844  |
| mln::Function< function< meta::blue< value::rgb< n > > > > . . . . .            | 715  |
| mln::Function_v2v< function< meta::blue< value::rgb< n > > > > . . . . .        | 717  |
| mln::Object< function< meta::first< util::couple< T, U > > > > . . . . .        | 844  |
| mln::Function< function< meta::first< util::couple< T, U > > > > . . . . .      | 715  |
| mln::Function_v2v< function< meta::first< util::couple< T, U > > > > . . . . .  | 717  |
| mln::Object< function< meta::green< value::rgb< n > > > > . . . . .             | 844  |
| mln::Function< function< meta::green< value::rgb< n > > > > . . . . .           | 715  |
| mln::Function_v2v< function< meta::green< value::rgb< n > > > > . . . . .       | 717  |
| mln::Object< function< meta::red< value::rgb< n > > > > . . . . .               | 844  |
| mln::Function< function< meta::red< value::rgb< n > > > > . . . . .             | 715  |
| mln::Function_v2v< function< meta::red< value::rgb< n > > > > . . . . .         | 717  |
| mln::Object< function< meta::second< util::couple< T, U > > > > . . . . .       | 844  |
| mln::Function< function< meta::second< util::couple< T, U > > > > . . . . .     | 715  |
| mln::Function_v2v< function< meta::second< util::couple< T, U > > > > . . . . . | 717  |

|   |      |
|---|------|
| mln::Object< function< meta::to_enc< T > > > . . . . .                                    | 844  |
| mln::Function< function< meta::to_enc< T > > > . . . . .                                  | 715  |
| mln::Function_v2v< function< meta::to_enc< T > > > . . . . .                              | 717  |
| mln::Object< fwd_pixter1d< I > > . . . . .  | 844  |
| mln::Iterator< fwd_pixter1d< I > > . . . . .  | 787  |
| mln::Pixel_Iterator< fwd_pixter1d< I > > . . . . .  | 940  |
| mln::Object< fwd_pixter2d< I > > . . . . .  | 844  |
| mln::Iterator< fwd_pixter2d< I > > . . . . .  | 787  |
| mln::Pixel_Iterator< fwd_pixter2d< I > > . . . . .  | 940  |
| mln::Object< fwd_pixter3d< I > > . . . . .  | 844  |
| mln::Iterator< fwd_pixter3d< I > > . . . . .  | 787  |
| mln::Pixel_Iterator< fwd_pixter3d< I > > . . . . .  | 940  |
| mln::Object< fwd_t > . . . . .  | 844  |
| mln::Browsing< fwd_t > . . . . .  | 568  |
| mln::canvas::browsing::fwd_t . . . . .  | 576  |
| mln::Object< ge< L, R > > . . . . .   | 844  |
| mln::Function< ge< L, R > > . . . . .   | 715  |
| mln::Function_vv2b< ge< L, R > > . . . . .  | 718  |
| mln::fun::vv2b::ge< L, R > . . . . .  | 689  |
| mln::Object< graph > . . . . .  | 844  |
| mln::Graph< graph > . . . . .   | 731  |
| graph_base< graph > . . . . .   |      |
| mln::util::graph . . . . .  | 1058 |
| mln::Object< graph_elt_mixed_window< G, S, S2 > > . . . . .                               | 844  |
| mln::Window< graph_elt_mixed_window< G, S, S2 > > . . . . .                               | 1170 |
| mln::graph_window_base< S2::fun_t::result, graph_elt_mixed_window< G, S, S2 > > . . . . . | 751  |
| mln::graph_elt_mixed_window< G, S, S2 > . . . . .   | 734  |
| mln::Object< graph_elt_window< G, S > > . . . . .   | 844  |
| mln::Window< graph_elt_window< G, S > > . . . . .   | 1170 |
| mln::graph_window_base< S::fun_t::result, graph_elt_window< G, S > > . . . . .            | 751  |
| mln::graph_elt_window< G, S > . . . . .   | 742  |
| mln::Object< graph_elt_window_if< G, S, I > > . . . . .                                   | 844  |
| mln::Window< graph_elt_window_if< G, S, I > > . . . . .                                   | 1170 |
| mln::graph_window_base< S::fun_t::result, graph_elt_window_if< G, S, I > > . . . . .      | 751  |
| mln::graph_elt_window_if< G, S, I > . . . . .   | 746  |
| mln::Object< graph_window_if_piter< S, W, I > > . . . . .                                 | 844  |
| mln::Proxy< graph_window_if_piter< S, W, I > > . . . . .                                  | 957  |
| mln::Site_Proxy< graph_window_if_piter< S, W, I > > . . . . .                             | 965  |
| mln::Site_Iterator< graph_window_if_piter< S, W, I > > . . . . .                          | 963  |
| site_iterator_base< W, graph_window_if_piter< S, W, I > > . . . . .                       |      |
| site_relative_iterator_base< W, graph_window_if_piter< S, W, I > > . . . . .              |      |
| mln::graph_window_if_piter< S, W, I > . . . . .   | 753  |
| mln::Object< graph_window_piter< S, W, I > > . . . . .                                    | 844  |
| mln::Proxy< graph_window_piter< S, W, I > > . . . . .                                     | 957  |
| mln::Site_Proxy< graph_window_piter< S, W, I > > . . . . .                                | 965  |
| mln::Site_Iterator< graph_window_piter< S, W, I > > . . . . .                             | 963  |
| site_iterator_base< W, graph_window_piter< S, W, I > > . . . . .                          |      |
| site_relative_iterator_base< W, graph_window_piter< S, W, I >, W::center_t > . . . . .    |      |
| mln::graph_window_piter< S, W, I > . . . . .  | 755  |

|  |      |
|--|------|
| mln::Object< gray_f > . . . . .                | 844  |
| mln::Value< gray_f > . . . . .                 | 1107 |
| mln::Object< graylevel< n > > . . . . .        | 844  |
| mln::Value< graylevel< n > > . . . . .         | 1107 |
| mln::Object< graylevel_f > . . . . .           | 844  |
| mln::Value< graylevel_f > . . . . .            | 1107 |
| mln::Object< green > . . . . .                 | 844  |
| mln::Meta_Function< green > . . . . .          | 826  |
| mln::Meta_Function_v2v< green > . . . . .      | 827  |
| mln::Object< green_t > . . . . .               | 844  |
| mln::Literal< green_t > . . . . .              | 799  |
| mln::literal::green_t . . . . .                | 804  |
| mln::Object< gt< L, R > > . . . . .            | 844  |
| mln::Function< gt< L, R > > . . . . .          | 715  |
| mln::Function_vv2b< gt< L, R > > . . . . .     | 718  |
| mln::fun::vv2b::gt< L, R > . . . . .           | 690  |
| mln::Object< has< I > > . . . . .              | 844  |
| mln::Function< has< I > > . . . . .            | 715  |
| mln::Function_v2v< has< I > > . . . . .        | 717  |
| mln::Function_v2b< has< I > > . . . . .        | 716  |
| mln::Object< height > . . . . .                | 844  |
| mln::Meta_Accumulator< height > . . . . .      | 824  |
| mln::accu::meta::shape::height . . . . .       | 483  |
| mln::Object< height< I > > . . . . .           | 844  |
| mln::Proxy< height< I > > . . . . .            | 957  |
| mln::Accumulator< height< I > > . . . . .      | 542  |
| base< unsigned, height< I > >                  |      |
| mln::accu::shape::height< I > . . . . .        | 505  |
| mln::morpho::attribute::height< I > . . . . .  | 835  |
| mln::Object< hexa > . . . . .                  | 844  |
| mln::Mesh< hexa > . . . . .                    | 823  |
| mln::Regular_Grid< hexa > . . . . .            | 960  |
| mln::Object< hexa< I > > . . . . .             | 844  |
| mln::Image< hexa< I > > . . . . .              | 762  |
| mln::Object< hexa< image2d< V > > > . . . . .  | 844  |
| mln::Image< hexa< image2d< V > > > . . . . .   | 762  |
| mln::Object< histo > . . . . .                 | 844  |
| mln::Meta_Accumulator< histo > . . . . .       | 824  |
| mln::accu::meta::histo . . . . .               | 466  |
| mln::Object< histo3d_rgb > . . . . .           | 844  |
| mln::Meta_Accumulator< histo3d_rgb > . . . . . | 824  |
| mln::Object< histo3d_rgb< V > > . . . . .      | 844  |
| mln::Proxy< histo3d_rgb< V > > . . . . .       | 957  |
| mln::Accumulator< histo3d_rgb< V > > . . . . . | 542  |
| base< image3d< unsigned >, histo3d_rgb< V > >  |      |
| mln::accu::stat::histo3d_rgb< V > . . . . .    | 513  |
| mln::Object< histo< V > > . . . . .            | 844  |
| mln::Proxy< histo< V > > . . . . .             | 957  |



|   |      |
|---|------|
| mln::Accumulator< histo< V > > . . . . .              | 542  |
| base< const std::vector< unsigned > &, histo< V > >   |      |
| mln::accu::histo< V > . . . . .                       | 445  |
| mln::Object< hyper_directional_t > . . . . .          | 844  |
| mln::Browsing< hyper_directional_t > . . . . .        | 568  |
| mln::canvas::browsing::hyper_directional_t . . . . .  | 577  |
| mln::Object< id2element< G, Elt > > . . . . .         | 844  |
| mln::Function< id2element< G, Elt > > . . . . .       | 715  |
| mln::Function_v2v< id2element< G, Elt > > . . . . .   | 717  |
| mln::Object< identity_t > . . . . .                   | 844  |
| mln::Literal< identity_t > . . . . .                  | 799  |
| mln::literal::identity_t . . . . .                    | 805  |
| mln::Object< ignore > . . . . .                       | 844  |
| mln::util::ignore . . . . .                           | 1067 |
| mln::Object< image1d< T > > . . . . .                 | 844  |
| mln::Image< image1d< T > > . . . . .                  | 762  |
| mln::Object< image2d< T > > . . . . .                 | 844  |
| mln::Image< image2d< T > > . . . . .                  | 762  |
| mln::Object< image3d< T > > . . . . .                 | 844  |
| mln::Image< image3d< T > > . . . . .                  | 762  |
| mln::Object< image< F, S > > . . . . .                | 844  |
| mln::Image< image< F, S > > . . . . .                 | 762  |
| mln::Object< image_if< I, F > > . . . . .             | 844  |
| mln::Image< image_if< I, F > > . . . . .              | 762  |
| mln::Object< implies< L, R > > . . . . .              | 844  |
| mln::Function< implies< L, R > > . . . . .            | 715  |
| mln::Function_vv2b< implies< L, R > > . . . . .       | 718  |
| mln::fun::vv2b::implies< L, R > . . . . .             | 691  |
| mln::Object< index_of_value< bool > > . . . . .       | 844  |
| mln::Function< index_of_value< bool > > . . . . .     | 715  |
| mln::Function_v2v< index_of_value< bool > > . . . . . | 717  |
| mln::Object< index_of_value< T > > . . . . .          | 844  |
| mln::Function< index_of_value< T > > . . . . .        | 715  |
| mln::Function_v2v< index_of_value< T > > . . . . .    | 717  |
| mln::Object< inf > . . . . .                          | 844  |
| mln::Meta_Accumulator< inf > . . . . .                | 824  |
| mln::accu::meta::math::inf . . . . .                  | 474  |
| mln::Meta_Function< inf > . . . . .                   | 826  |
| mln::Meta_Function_vv2v< inf > . . . . .              | 827  |
| mln::Object< inf< T > > . . . . .                     | 844  |
| mln::Proxy< inf< T > > . . . . .                      | 957  |
| mln::Accumulator< inf< T > > . . . . .                | 542  |
| base< const T &, inf< T > >                           |      |
| mln::accu::math::inf< T > . . . . .                   | 457  |
| mln::Object< int_s< n > > . . . . .                   | 844  |
| mln::Value< int_s< n > > . . . . .                    | 1107 |
| mln::Object< int_u8_off_saver > . . . . .             | 844  |
| mln::Object< int_u< n > > . . . . .                   | 844  |

|   |      |
|---|------|
| mln::Value< int_u< n > > . . . . .                  | 1107 |
| mln::Object< int_u_sat< n > > . . . . .             | 844  |
| mln::Value< int_u_sat< n > > . . . . .              | 1107 |
| mln::Object< interpolated< I, F > > . . . . .       | 844  |
| mln::Image< interpolated< I, F > > . . . . .        | 762  |
| mln::Object< iota > . . . . .                       | 844  |
| mln::Function< iota > . . . . .                     | 715  |
| mln::Function_v2v< iota > . . . . .                 | 717  |
| mln::Object< is_dot > . . . . .                     | 844  |
| mln::Function< is_dot > . . . . .                   | 715  |
| mln::Function_v2v< is_dot > . . . . .               | 717  |
| mln::Function_v2b< is_dot > . . . . .               | 716  |
| mln::Object< is_edge > . . . . .                    | 844  |
| mln::Function< is_edge > . . . . .                  | 715  |
| mln::Function_v2v< is_edge > . . . . .              | 717  |
| mln::Function_v2b< is_edge > . . . . .              | 716  |
| mln::Object< is_n_face< N > > . . . . .             | 844  |
| mln::Function< is_n_face< N > > . . . . .           | 715  |
| mln::Function_v2v< is_n_face< N > > . . . . .       | 717  |
| mln::Function_v2b< is_n_face< N > > . . . . .       | 716  |
| mln::topo::is_n_face< N > . . . . .                 | 1014 |
| mln::Object< is_pixel > . . . . .                   | 844  |
| mln::Function< is_pixel > . . . . .                 | 715  |
| mln::Function_v2v< is_pixel > . . . . .             | 717  |
| mln::Function_v2b< is_pixel > . . . . .             | 716  |
| mln::Object< is_row_odd > . . . . .                 | 844  |
| mln::Function< is_row_odd > . . . . .               | 715  |
| mln::Function_v2v< is_row_odd > . . . . .           | 717  |
| mln::Function_v2b< is_row_odd > . . . . .           | 716  |
| mln::Object< is_separator > . . . . .               | 844  |
| mln::Function< is_separator > . . . . .             | 715  |
| mln::Function_v2v< is_separator > . . . . .         | 717  |
| mln::Function_v2b< is_separator > . . . . .         | 716  |
| mln::world::inter_pixel::is_separator . . . . .     | 1176 |
| mln::Object< is_simple_cell< I > > . . . . .        | 844  |
| mln::Function< is_simple_cell< I > > . . . . .      | 715  |
| mln::Function_v2v< is_simple_cell< I > > . . . . .  | 717  |
| mln::Function_v2b< is_simple_cell< I > > . . . . .  | 716  |
| mln::topo::is_simple_cell< I > . . . . .            | 1016 |
| mln::Object< ithcomp > . . . . .                    | 844  |
| mln::Meta_Function< ithcomp > . . . . .             | 826  |
| mln::Meta_Function_vv2v< ithcomp > . . . . .        | 827  |
| mln::Object< keep_specific_colors > . . . . .       | 844  |
| mln::Function< keep_specific_colors > . . . . .     | 715  |
| mln::Function_v2v< keep_specific_colors > . . . . . | 717  |
| mln::Function_v2b< keep_specific_colors > . . . . . | 716  |
| mln::Object< l1 > . . . . .                         | 844  |
| mln::Meta_Function< l1 > . . . . .                  | 826  |

|  |      |
|--|------|
| mln::Meta_Function_v2v< I1 > . . . . .                   | 827  |
| mln::Object< I1_norm< V > > . . . . .                    | 844  |
| mln::Function< I1_norm< V > > . . . . .                  | 715  |
| mln::Function_v2v< I1_norm< V > > . . . . .              | 717  |
| mln::Object< I1_norm< V, R > > . . . . .                 | 844  |
| mln::Function< I1_norm< V, R > > . . . . .               | 715  |
| mln::Function_v2v< I1_norm< V, R > > . . . . .           | 717  |
| mln::fun::v2v::I1_norm< V, R > . . . . .                 | 678  |
| mln::fun::v2w_w2v::I1_norm< V, R > . . . . .             | 685  |
| mln::Object< I2 > . . . . .                              | 844  |
| mln::Meta_Function< I2 > . . . . .                       | 826  |
| mln::Meta_Function_v2v< I2 > . . . . .                   | 827  |
| mln::Object< I2_norm< V, R > > . . . . .                 | 844  |
| mln::Function< I2_norm< V, R > > . . . . .               | 715  |
| mln::Function_v2v< I2_norm< V, R > > . . . . .           | 717  |
| mln::fun::v2v::I2_norm< V, R > . . . . .                 | 679  |
| mln::fun::v2w_w2v::I2_norm< V, R > . . . . .             | 687  |
| mln::Object< label< n > > . . . . .                      | 844  |
| mln::Value< label< n > > . . . . .                       | 1107 |
| mln::Object< label_used > . . . . .                      | 844  |
| mln::Meta_Accumulator< label_used > . . . . .            | 824  |
| mln::accu::meta::label_used . . . . .                    | 467  |
| mln::Object< label_used< L > > . . . . .                 | 844  |
| mln::Proxy< label_used< L > > . . . . .                  | 957  |
| mln::Accumulator< label_used< L > > . . . . .            | 542  |
| base< const fun::i2v::array< bool > &, label_used< L > > |      |
| mln::accu::label_used< L > . . . . .                     | 446  |
| mln::Object< labeled_image< I > > . . . . .              | 844  |
| mln::Image< labeled_image< I > > . . . . .               | 762  |
| mln::Object< land > . . . . .                            | 844  |
| mln::Meta_Accumulator< land > . . . . .                  | 824  |
| mln::accu::meta::logic::land . . . . .                   | 468  |
| mln::Proxy< land > . . . . .                             | 957  |
| mln::Accumulator< land > . . . . .                       | 542  |
| base< bool, land >                                       |      |
| mln::accu::logic::land . . . . .                         | 448  |
| mln::Object< land< L, R > > . . . . .                    | 844  |
| mln::Function< land< L, R > > . . . . .                  | 715  |
| mln::Function_vv2v< land< L, R > > . . . . .             | 718  |
| mln::fun::vv2v::land< L, R > . . . . .                   | 695  |
| mln::Object< land_basic > . . . . .                      | 844  |
| mln::Meta_Accumulator< land_basic > . . . . .            | 824  |
| mln::accu::meta::logic::land_basic . . . . .             | 469  |
| mln::Proxy< land_basic > . . . . .                       | 957  |
| mln::Accumulator< land_basic > . . . . .                 | 542  |
| base< bool, land_basic >                                 |      |
| mln::accu::logic::land_basic . . . . .                   | 449  |
| mln::Object< land_not< L, R > > . . . . .                | 844  |
| mln::Function< land_not< L, R > > . . . . .              | 715  |

|  |      |
|--|------|
| mln::Function_vv2v< land_not< L, R > > . . . . .         | 718  |
| mln::fun::vv2v::land_not< L, R > . . . . .               | 696  |
| mln::Object< lazy_image< I, F, B > > . . . . .           | 844  |
| mln::Image< lazy_image< I, F, B > > . . . . .            | 762  |
| mln::Object< le< L, R > > . . . . .                      | 844  |
| mln::Function< le< L, R > > . . . . .                    | 715  |
| mln::Function_vv2b< le< L, R > > . . . . .               | 718  |
| mln::fun::vv2b::le< L, R > . . . . .                     | 692  |
| mln::Object< light_gray_t > . . . . .                    | 844  |
| mln::Literal< light_gray_t > . . . . .                   | 799  |
| mln::literal::light_gray_t . . . . .                     | 806  |
| mln::Object< lime_t > . . . . .                          | 844  |
| mln::Literal< lime_t > . . . . .                         | 799  |
| mln::literal::lime_t . . . . .                           | 807  |
| mln::Object< line< M, i, C > > . . . . .                 | 844  |
| mln::Window< line< M, i, C > > . . . . .                 | 1170 |
| window_base< dpoint< M, C >, line< M, i, C > >           |      |
| classical_window_base< dpoint< M, C >, line< M, i, C > > |      |
| mln::win::line< M, i, C > . . . . .                      | 1165 |
| mln::Object< line_graph< G > > . . . . .                 | 844  |
| mln::Graph< line_graph< G > > . . . . .                  | 731  |
| graph_base< line_graph< G > >                            |      |
| mln::util::line_graph< G > . . . . .                     | 1068 |
| mln::Object< linear< V, T, R > > . . . . .               | 844  |
| mln::Function< linear< V, T, R > > . . . . .             | 715  |
| mln::Function_v2v< linear< V, T, R > > . . . . .         | 717  |
| mln::fun::v2v::linear< V, T, R > . . . . .               | 680  |
| mln::Object< linear_sat< V, T, R > > . . . . .           | 844  |
| mln::Function< linear_sat< V, T, R > > . . . . .         | 715  |
| mln::Function_v2v< linear_sat< V, T, R > > . . . . .     | 717  |
| mln::Object< linfty > . . . . .                          | 844  |
| mln::Meta_Function< linfty > . . . . .                   | 826  |
| mln::Meta_Function_v2v< linfty > . . . . .               | 827  |
| mln::Object< linfty_norm< V, R > > . . . . .             | 844  |
| mln::Function< linfty_norm< V, R > > . . . . .           | 715  |
| mln::Function_v2v< linfty_norm< V, R > > . . . . .       | 717  |
| mln::fun::v2v::linfty_norm< V, R > . . . . .             | 681  |
| mln::fun::v2w_w2v::linfty_norm< V, R > . . . . .         | 688  |
| mln::Object< lnot< V > > . . . . .                       | 844  |
| mln::Function< lnot< V > > . . . . .                     | 715  |
| mln::Function_v2v< lnot< V > > . . . . .                 | 717  |
| mln::Function_v2b< lnot< V > > . . . . .                 | 716  |
| mln::fun::v2b::lnot< V > . . . . .                       | 675  |
| mln::Object< lor > . . . . .                             | 844  |
| mln::Meta_Accumulator< lor > . . . . .                   | 824  |
| mln::accu::meta::logic::lor . . . . .                    | 470  |
| mln::Proxy< lor > . . . . .                              | 957  |
| mln::Accumulator< lor > . . . . .                        | 542  |

|   |      |
|---|------|
| base< bool, lor >                               |      |
| mln::accu::logic::lor . . . . .                 | 451  |
| mln::Object< lor< L, R > > . . . . .            | 844  |
| mln::Function< lor< L, R > > . . . . .          | 715  |
| mln::Function_vv2v< lor< L, R > > . . . . .     | 718  |
| mln::fun::vv2v::lor< L, R > . . . . .           | 697  |
| mln::Object< lor_basic > . . . . .              | 844  |
| mln::Meta_Accumulator< lor_basic > . . . . .    | 824  |
| mln::accu::meta::logic::lor_basic . . . . .     | 471  |
| mln::Proxy< lor_basic > . . . . .               | 957  |
| mln::Accumulator< lor_basic > . . . . .         | 542  |
| base< bool, lor_basic >                         |      |
| mln::accu::logic::lor_basic . . . . .           | 452  |
| mln::Object< lt< L, R > > . . . . .             | 844  |
| mln::Function< lt< L, R > > . . . . .           | 715  |
| mln::Function_vv2b< lt< L, R > > . . . . .      | 718  |
| mln::fun::vv2b::lt< L, R > . . . . .            | 693  |
| mln::Object< lut_vec< S, T > > . . . . .        | 844  |
| mln::Value_Set< lut_vec< S, T > > . . . . .     | 1148 |
| mln::value::lut_vec< S, T > . . . . .           | 1129 |
| mln::Object< lxor< L, R > > . . . . .           | 844  |
| mln::Function< lxor< L, R > > . . . . .         | 715  |
| mln::Function_vv2v< lxor< L, R > > . . . . .    | 718  |
| mln::fun::vv2v::lxor< L, R > . . . . .          | 698  |
| mln::Object< magenta_t > . . . . .              | 844  |
| mln::Literal< magenta_t > . . . . .             | 799  |
| mln::literal::magenta_t . . . . .               | 808  |
| mln::Object< mahalanobis< V > > . . . . .       | 844  |
| mln::Function< mahalanobis< V > > . . . . .     | 715  |
| mln::Function_v2v< mahalanobis< V > > . . . . . | 717  |
| mln::Object< maj_h > . . . . .                  | 844  |
| mln::Meta_Accumulator< maj_h > . . . . .        | 824  |
| mln::accu::meta::maj_h . . . . .                | 472  |
| mln::Object< maj_h< T > > . . . . .             | 844  |
| mln::Proxy< maj_h< T > > . . . . .              | 957  |
| mln::Accumulator< maj_h< T > > . . . . .        | 542  |
| base< const T &, maj_h< T > >                   |      |
| mln::accu::maj_h< T > . . . . .                 | 454  |
| mln::Object< mat< n, m, T > > . . . . .         | 844  |
| mln::Object< max > . . . . .                    | 844  |
| mln::Meta_Accumulator< max > . . . . .          | 824  |
| mln::accu::meta::stat::max . . . . .            | 485  |
| mln::Object< max< T > > . . . . .               | 844  |
| mln::Proxy< max< T > > . . . . .                | 957  |
| mln::Accumulator< max< T > > . . . . .          | 542  |
| base< const T &, max< T > >                     |      |
| mln::accu::stat::max< T > . . . . .             | 515  |
| mln::Object< max< V > > . . . . .               | 844  |
| mln::Function< max< V > > . . . . .             | 715  |

|   |     |
|---|-----|
| mln::Function_vv2v< max< V > > . . . . .                              | 718 |
| mln::fun::vv2v::max< V > . . . . .                                    | 699 |
| mln::Object< max_h > . . . . .  | 844 |
| mln::Meta_Accumulator< max_h > . . . . .                              | 824 |
| mln::accu::meta::stat::max_h . . . . .                                | 486 |
| mln::Object< max_h< V > > . . . . .                                   | 844 |
| mln::Proxy< max_h< V > > . . . . .                                    | 957 |
| mln::Accumulator< max_h< V > > . . . . .                              | 542 |
| base< const V &, max_h< V > >   |     |
| mln::accu::stat::max_h< V > . . . . .                                 | 517 |
| mln::Object< max_site > . . . . .                                     | 844 |
| mln::Meta_Accumulator< max_site > . . . . .                           | 824 |
| mln::accu::meta::max_site . . . . .                                   | 477 |
| mln::Object< max_site< I > > . . . . .                                | 844 |
| mln::Proxy< max_site< I > > . . . . .                                 | 957 |
| mln::Accumulator< max_site< I > > . . . . .                           | 542 |
| base< I::psite, max_site< I > >                                       |     |
| mln::accu::max_site< I > . . . . .                                    | 461 |
| mln::Object< max_t > . . . . .  | 844 |
| mln::Literal< max_t > . . . . .                                       | 799 |
| mln::literal::max_t . . . . .   | 809 |
| mln::Object< mean > . . . . .   | 844 |
| mln::Meta_Accumulator< mean > . . . . .                               | 824 |
| mln::accu::meta::stat::mean . . . . .                                 | 487 |
| mln::Meta_Function< mean > . . . . .                                  | 826 |
| mln::Meta_Function_v2v< mean > . . . . .                              | 827 |
| mln::Object< mean< T, S, M > > . . . . .                              | 844 |
| mln::Proxy< mean< T, S, M > > . . . . .                               | 957 |
| mln::Accumulator< mean< T, S, M > > . . . . .                         | 542 |
| base< M, mean< T, S, M > >  |     |
| mln::accu::stat::mean< T, S, M > . . . . .                            | 518 |
| mln::Object< median_alt< S > > . . . . .                              | 844 |
| mln::Proxy< median_alt< S > > . . . . .                               | 957 |
| mln::Accumulator< median_alt< S > > . . . . .                         | 542 |
| base< const S::value &, median_alt< S > >                             |     |
| mln::accu::stat::median_alt< S > . . . . .                            | 520 |
| mln::Object< median_alt< T > > . . . . .                              | 844 |
| mln::Meta_Accumulator< median_alt< T > > . . . . .                    | 824 |
| mln::accu::meta::stat::median_alt< T > . . . . .                      | 488 |
| mln::Object< median_alt< value::set< T > > > . . . . .                | 844 |
| mln::Proxy< median_alt< value::set< T > > > . . . . .                 | 957 |
| mln::Accumulator< median_alt< value::set< T > > > . . . . .           | 542 |
| base< const value::set< T >::value &, median_alt< value::set< T > > > |     |
| mln::accu::stat::median_alt< value::set< T > > . . . . .              | 520 |
| mln::Object< median_h > . . . . .                                     | 844 |
| mln::Meta_Accumulator< median_h > . . . . .                           | 824 |
| mln::accu::meta::stat::median_h . . . . .                             | 489 |
| mln::Object< median_h< V > > . . . . .                                | 844 |
| mln::Proxy< median_h< V > > . . . . .                                 | 957 |

|  |      |
|--|------|
| mln::Accumulator< median_h< V > > . . . . .        | 542  |
| base< const V &, median_h< V > >                   |      |
| mln::accu::stat::median_h< V > . . . . .           | 522  |
| mln::Object< medium_gray_t > . . . . .             | 844  |
| mln::Literal< medium_gray_t > . . . . .            | 799  |
| mln::Object< min > . . . . .                       | 844  |
| mln::Meta_Accumulator< min > . . . . .             | 824  |
| mln::accu::meta::stat::min . . . . .               | 490  |
| mln::Object< min< L, R > > . . . . .               | 844  |
| mln::Function< min< L, R > > . . . . .             | 715  |
| mln::Function_vv2v< min< L, R > > . . . . .        | 718  |
| mln::fun::vv2v::min< L, R > . . . . .              | 700  |
| mln::Object< min< T > > . . . . .                  | 844  |
| mln::Proxy< min< T > > . . . . .                   | 957  |
| mln::Accumulator< min< T > > . . . . .             | 542  |
| base< const T &, min< T > >                        |      |
| mln::accu::stat::min< T > . . . . .                | 524  |
| mln::Object< min_h > . . . . .                     | 844  |
| mln::Meta_Accumulator< min_h > . . . . .           | 824  |
| mln::accu::meta::stat::min_h . . . . .             | 491  |
| mln::Object< min_h< V > > . . . . .                | 844  |
| mln::Proxy< min_h< V > > . . . . .                 | 957  |
| mln::Accumulator< min_h< V > > . . . . .           | 542  |
| base< const V &, min_h< V > >                      |      |
| mln::accu::stat::min_h< V > . . . . .              | 526  |
| mln::Object< min_t > . . . . .                     | 844  |
| mln::Literal< min_t > . . . . .                    | 799  |
| mln::literal::min_t . . . . .                      | 810  |
| mln::Object< mirror< B > > . . . . .               | 844  |
| mln::Function< mirror< B > > . . . . .             | 715  |
| mln::Function_v2v< mirror< B > > . . . . .         | 717  |
| mln::Object< mixed_neighb< W > > . . . . .         | 844  |
| mln::Neighborhood< mixed_neighb< W > > . . . . .   | 844  |
| mln::Object< mln::util::set< T > > . . . . .       | 844  |
| mln::util::set< T > . . . . .                      | 1081 |
| mln::Object< multi_site< P > > . . . . .           | 844  |
| mln::Object< multiple< W, F > > . . . . .          | 844  |
| mln::Window< multiple< W, F > > . . . . .          | 1170 |
| window_base< W::dpsite, multiple< W, F > >         |      |
| mln::win::multiple< W, F > . . . . .               | 1167 |
| mln::Object< multiple_qiter< W, F > > . . . . .    | 844  |
| mln::Proxy< multiple_qiter< W, F > > . . . . .     | 957  |
| mln::Site_Proxy< multiple_qiter< W, F > >          | 965  |
| mln::Site_Iterator< multiple_qiter< W, F > >       | 963  |
| mln::Object< multiple_size< n, W, F > > . . . . .  | 844  |
| mln::Window< multiple_size< n, W, F > > . . . . .  | 1170 |
| window_base< W::dpsite, multiple_size< n, W, F > > |      |
| mln::win::multiple_size< n, W, F > . . . . .       | 1167 |

|   |      |
|---|------|
| mln::Object< multiple_size_qiter< n, W, F > > . . . . .                     | 844  |
| mln::Proxy< multiple_size_qiter< n, W, F > > . . . . .                      | 957  |
| mln::Site_Proxy< multiple_size_qiter< n, W, F > > . . . . .                 | 965  |
| mln::Site_Iterator< multiple_size_qiter< n, W, F > > . . . . .              | 963  |
| mln::Object< my_box2d > . . . . .   | 844  |
| mln::Function< my_box2d > . . . . .   | 715  |
| mln::Function_v2v< my_box2d > . . . . .                                     | 717  |
| mln::Function_v2b< my_box2d > . . . . .                                     | 716  |
| mln::Object< my_ext > . . . . .   | 844  |
| mln::Function< my_ext > . . . . .   | 715  |
| mln::Function_v2v< my_ext > . . . . .                                       | 717  |
| mln::Object< my_fun< G > > . . . . .  | 844  |
| mln::Function< my_fun< G > > . . . . .                                      | 715  |
| mln::Object< my_image2d< T > > . . . . .                                    | 844  |
| mln::Image< my_image2d< T > > . . . . .                                     | 762  |
| mln::Object< my_values_t > . . . . .  | 844  |
| mln::Function< my_values_t > . . . . .                                      | 715  |
| mln::Function_v2v< my_values_t > . . . . .                                  | 717  |
| mln::Object< myfun > . . . . .  | 844  |
| mln::Function< myfun > . . . . .  | 715  |
| mln::Function_vv2v< myfun > . . . . .                                       | 718  |
| mln::Object< mysqrt > . . . . .   | 844  |
| mln::Function< mysqrt > . . . . .   | 715  |
| mln::Function_v2v< mysqrt > . . . . .                                       | 717  |
| mln::Object< n_face_bkd_iter< D > > . . . . .                               | 844  |
| mln::Iterator< n_face_bkd_iter< D > > . . . . .                             | 787  |
| complex_iterator_base< topo::face< D >, n_face_bkd_iter< D > >              |      |
| complex_set_iterator_base< topo::face< D >, n_face_bkd_iter< D > >          |      |
| mln::topo::n_face_bkd_iter< D > . . . . .                                   | 1023 |
| mln::Object< n_face_fwd_iter< D > > . . . . .                               | 844  |
| mln::Iterator< n_face_fwd_iter< D > > . . . . .                             | 787  |
| complex_iterator_base< topo::face< D >, n_face_fwd_iter< D > >              |      |
| complex_set_iterator_base< topo::face< D >, n_face_fwd_iter< D > >          |      |
| mln::topo::n_face_fwd_iter< D > . . . . .                                   | 1024 |
| mln::Object< neighb< graph_elt_mixed_window< G, S, S2 > > > . . . . .       | 844  |
| mln::Neighborhood< neighb< graph_elt_mixed_window< G, S, S2 > > > . . . . . | 844  |
| mln::Object< neighb< graph_elt_window< G, S > > > . . . . .                 | 844  |
| mln::Neighborhood< neighb< graph_elt_window< G, S > > > . . . . .           | 844  |
| mln::Object< neighb< graph_elt_window_if< G, S, I > > > . . . . .           | 844  |
| mln::Neighborhood< neighb< graph_elt_window_if< G, S, I > > > . . . . .     | 844  |
| mln::Object< neighb< W > > . . . . .  | 844  |
| mln::Neighborhood< neighb< W > > . . . . .                                  | 844  |
| mln::Object< neighb_bkd_niter< W > > . . . . .                              | 844  |
| mln::Proxy< neighb_bkd_niter< W > > . . . . .                               | 957  |
| mln::Site_Proxy< neighb_bkd_niter< W > > . . . . .                          | 965  |
| mln::Site_Iterator< neighb_bkd_niter< W > > . . . . .                       | 963  |
| mln::Object< neighb_fwd_niter< W > > . . . . .                              | 844  |
| mln::Proxy< neighb_fwd_niter< W > > . . . . .                               | 957  |



|   |      |
|---|------|
| mln::Site_Proxy< neighb_fwd_niter< W > > . . . . .    | 965  |
| mln::Site_Iterator< neighb_fwd_niter< W > > . . . . . | 963  |
| mln::Object< nil > . . . . .                          | 844  |
| mln::Meta_Accumulator< nil > . . . . .                | 824  |
| mln::accu::meta::nil . . . . .                        | 478  |
| mln::util::nil . . . . .                              | 1074 |
| mln::Object< nil< T > > . . . . .                     | 844  |
| mln::Proxy< nil< T > > . . . . .                      | 957  |
| mln::Accumulator< nil< T > > . . . . .                | 542  |
| base< util::ignore, nil< T > >                        |      |
| mln::accu::nil< T > . . . . .                         | 496  |
| mln::Object< not_to_remove > . . . . .                | 844  |
| mln::Function< not_to_remove > . . . . .              | 715  |
| mln::Function_v2v< not_to_remove > . . . . .          | 717  |
| mln::Function_v2b< not_to_remove > . . . . .          | 716  |
| mln::Object< object_id< Tag, V > > . . . . .          | 844  |
| mln::Value< object_id< Tag, V > > . . . . .           | 1107 |
| mln::Object< octagon2d > . . . . .                    | 844  |
| mln::Window< octagon2d > . . . . .                    | 1170 |
| window_base< dpoint2d, octagon2d >                    |      |
| classical_window_base< dpoint2d, octagon2d >          |      |
| mln::win::octagon2d . . . . .                         | 1168 |
| mln::Object< olive_t > . . . . .                      | 844  |
| mln::Literal< olive_t > . . . . .                     | 799  |
| mln::literal::olive_t . . . . .                       | 811  |
| mln::Object< one_t > . . . . .                        | 844  |
| mln::Literal< one_t > . . . . .                       | 799  |
| mln::literal::one_t . . . . .                         | 812  |
| mln::Object< orange_t > . . . . .                     | 844  |
| mln::Literal< orange_t > . . . . .                    | 799  |
| mln::literal::orange_t . . . . .                      | 813  |
| mln::Object< ord_pair< T > > . . . . .                | 844  |
| mln::util::ord_pair< T > . . . . .                    | 1077 |
| mln::Object< origin_t > . . . . .                     | 844  |
| mln::Literal< origin_t > . . . . .                    | 799  |
| mln::literal::origin_t . . . . .                      | 814  |
| mln::Object< P > . . . . .                            | 844  |
| mln::Point_Site< P > . . . . .                        | 953  |
| mln::Point< P > . . . . .                             | 943  |
| mln::Object< p2p_image< I, F > > . . . . .            | 844  |
| mln::Image< p2p_image< I, F > > . . . . .             | 762  |
| mln::Object< p< A > > . . . . .                       | 844  |
| mln::Proxy< p< A > > . . . . .                        | 957  |
| mln::Accumulator< p< A > > . . . . .                  | 542  |
| base< const A::result &, p< A > >                     |      |
| mln::accu::p< A > . . . . .                           | 498  |
| mln::Object< p< mA > > . . . . .                      | 844  |
| mln::Meta_Accumulator< p< mA > > . . . . .            | 824  |

|  |     |
|--|-----|
| mln::accu::meta::p< mA > . . . . .                           | 479 |
| mln::Object< p_array< P > > . . . . .                        | 844 |
| mln::Site_Set< p_array< P > > . . . . .                      | 966 |
| site_set_base_< P, p_array< P > >                            |     |
| mln::p_array< P > . . . . .                                  | 847 |
| mln::Object< p_centered< W > > . . . . .                     | 844 |
| mln::Site_Set< p_centered< W > > . . . . .                   | 966 |
| site_set_base_< W::psite, p_centered< W > >                  |     |
| mln::p_centered< W > . . . . .                               | 852 |
| mln::Object< p_centered_piter< W > > . . . . .               | 844 |
| mln::Proxy< p_centered_piter< W > > . . . . .                | 957 |
| mln::Site_Proxy< p_centered_piter< W > > . . . . .           | 965 |
| mln::Site_Iterator< p_centered_piter< W > > . . . . .        | 963 |
| mln::Object< p_complex< D, G > > . . . . .                   | 844 |
| mln::Site_Set< p_complex< D, G > > . . . . .                 | 966 |
| site_set_base_< complex_psite< D, G >, p_complex< D, G > >   |     |
| mln::p_complex< D, G > . . . . .                             | 855 |
| mln::Object< p_double_piter< S, I1, I2 > > . . . . .         | 844 |
| mln::Proxy< p_double_piter< S, I1, I2 > > . . . . .          | 957 |
| mln::Site_Proxy< p_double_piter< S, I1, I2 > > . . . . .     | 965 |
| mln::Site_Iterator< p_double_piter< S, I1, I2 > > . . . . .  | 963 |
| mln::Object< p_double_psite< S, Sp > > . . . . .             | 844 |
| mln::Proxy< p_double_psite< S, Sp > > . . . . .              | 957 |
| mln::Site_Proxy< p_double_psite< S, Sp > > . . . . .         | 965 |
| mln::Pseudo_Site< p_double_psite< S, Sp > > . . . . .        | 957 |
| mln::Object< p_edges< G, F > > . . . . .                     | 844 |
| mln::Site_Set< p_edges< G, F > > . . . . .                   | 966 |
| site_set_base_< F::result, p_edges< G, F > >                 |     |
| mln::p_edges< G, F > . . . . .                               | 859 |
| mln::Object< p_edges_psite< G, F > > . . . . .               | 844 |
| mln::Proxy< p_edges_psite< G, F > > . . . . .                | 957 |
| mln::Site_Proxy< p_edges_psite< G, F > > . . . . .           | 965 |
| mln::Pseudo_Site< p_edges_psite< G, F > > . . . . .          | 957 |
| mln::Object< p_faces< N, D, P > > . . . . .                  | 844 |
| mln::Site_Set< p_faces< N, D, P > > . . . . .                | 966 |
| site_set_base_< faces_psite< N, D, P >, p_faces< N, D, P > > |     |
| mln::p_faces< N, D, P > . . . . .                            | 865 |
| mln::Object< p_graph_piter< S, I > > . . . . .               | 844 |
| mln::Proxy< p_graph_piter< S, I > > . . . . .                | 957 |
| mln::Site_Proxy< p_graph_piter< S, I > > . . . . .           | 965 |
| mln::Site_Iterator< p_graph_piter< S, I > > . . . . .        | 963 |
| site_iterator_base< S, p_graph_piter< S, I > >               |     |
| site_set_iterator_base< S, p_graph_piter< S, I > >           |     |
| mln::p_graph_piter< S, I > . . . . .                         | 868 |
| mln::Object< p_if< S, F > > . . . . .                        | 844 |
| mln::Site_Set< p_if< S, F > > . . . . .                      | 966 |
| site_set_base_< S::psite, p_if< S, F > >                     |     |
| mln::p_if< S, F > . . . . .                                  | 870 |
| mln::Object< p_image< I > > . . . . .                        | 844 |

|  |     |
|--|-----|
| mln::Site_Set< p_image< I > > . . . . .  | 966 |
| site_set_base_< I::psite, p_image< I > >   |     |
| mln::p_image< I > . . . . .  | 873 |
| mln::Object< p_indexed_bkd_piter< S > > . . . . .  | 844 |
| mln::Proxy< p_indexed_bkd_piter< S > > . . . . .   | 957 |
| mln::Site_Proxy< p_indexed_bkd_piter< S > > . . . . .  | 965 |
| mln::Site_Iterator< p_indexed_bkd_piter< S > > . . . . .   | 963 |
| site_iterator_base< S, p_indexed_bkd_piter< S > >  |     |
| site_set_iterator_base< S, p_indexed_bkd_piter< S > >  |     |
| mln::p_indexed_bkd_piter< S > . . . . .  | 877 |
| mln::Object< p_indexed_fwd_piter< S > > . . . . .  | 844 |
| mln::Proxy< p_indexed_fwd_piter< S > > . . . . .   | 957 |
| mln::Site_Proxy< p_indexed_fwd_piter< S > > . . . . .  | 965 |
| mln::Site_Iterator< p_indexed_fwd_piter< S > > . . . . .   | 963 |
| site_iterator_base< S, p_indexed_fwd_piter< S > >  |     |
| site_set_iterator_base< S, p_indexed_fwd_piter< S > >  |     |
| mln::p_indexed_fwd_piter< S > . . . . .  | 879 |
| mln::Object< p_indexed_psite< S > > . . . . .  | 844 |
| mln::Proxy< p_indexed_psite< S > > . . . . .   | 957 |
| mln::Site_Proxy< p_indexed_psite< S > > . . . . .  | 965 |
| mln::Pseudo_Site< p_indexed_psite< S > > . . . . .   | 957 |
| pseudo_site_base_< const S::element &, p_indexed_psite< S > >  |     |
| mln::p_indexed_psite< S > . . . . .  | 880 |
| mln::Object< p_key< K, P > > . . . . .   | 844 |
| mln::Site_Set< p_key< K, P > > . . . . .   | 966 |
| site_set_base_< P, p_key< K, P > >   |     |
| mln::p_key< K, P > . . . . .   | 880 |
| mln::Object< p_line2d > . . . . .  | 844 |
| mln::Site_Set< p_line2d > . . . . .  | 966 |
| site_set_base_< point2d, p_line2d >  |     |
| mln::p_line2d . . . . .  | 886 |
| mln::Object< p_mutable_array_of< S > > . . . . .   | 844 |
| mln::Site_Set< p_mutable_array_of< S > > . . . . .   | 966 |
| site_set_base_< S::site, p_mutable_array_of< S > >   |     |
| mln::p_mutable_array_of< S > . . . . .   | 890 |
| mln::Object< p_n_faces_bkd_piter< D, G > > . . . . .   | 844 |
| mln::Proxy< p_n_faces_bkd_piter< D, G > > . . . . .  | 957 |
| mln::Site_Proxy< p_n_faces_bkd_piter< D, G > > . . . . .   | 965 |
| mln::Site_Iterator< p_n_faces_bkd_piter< D, G > > . . . . .  | 963 |
| site_iterator_base< p_complex< D, G >, p_n_faces_bkd_piter< D, G > >   |     |
| site_set_iterator_base< p_complex< D, G >, p_n_faces_bkd_piter< D, G > >                                     |     |
| p_complex_piter_base_< topo::n_face_bkd_iter< D >, p_complex< D, G >, G::site, p_n_faces_bkd_piter< D, G > > |     |
| mln::p_n_faces_bkd_piter< D, G > . . . . .   | 893 |
| mln::Object< p_n_faces_fwd_piter< D, G > > . . . . .   | 844 |
| mln::Proxy< p_n_faces_fwd_piter< D, G > > . . . . .  | 957 |
| mln::Site_Proxy< p_n_faces_fwd_piter< D, G > > . . . . .   | 965 |
| mln::Site_Iterator< p_n_faces_fwd_piter< D, G > > . . . . .  | 963 |
| site_iterator_base< p_complex< D, G >, p_n_faces_fwd_piter< D, G > >   |     |
| site_set_iterator_base< p_complex< D, G >, p_n_faces_fwd_piter< D, G > >                                     |     |

|  |     |
|--|-----|
| p_complex_piter_base_< topo::n_face_fwd_iter< D >, p_complex< D, G >, G::site, p_n_faces_fwd_piter< D, G > > |     |
| mln::p_n_faces_fwd_piter< D, G > . . . . .   | 895 |
| mln::Object< p_priority< P, Q > > . . . . .  | 844 |
| mln::Site_Set< p_priority< P, Q > > . . . . .  | 966 |
| site_set_base_< Q::site, p_priority< P, Q > >  |     |
| mln::p_priority< P, Q > . . . . .  | 896 |
| mln::Object< p_queue< P > > . . . . .  | 844 |
| mln::Site_Set< p_queue< P > > . . . . .  | 966 |
| site_set_base_< P, p_queue< P > >  |     |
| mln::p_queue< P > . . . . .  | 902 |
| mln::Object< p_queue_fast< P > > . . . . .   | 844 |
| mln::Site_Set< p_queue_fast< P > > . . . . .   | 966 |
| site_set_base_< P, p_queue_fast< P > >   |     |
| mln::p_queue_fast< P > . . . . .   | 906 |
| mln::Object< p_run< P > > . . . . .  | 844 |
| mln::Site_Set< p_run< P > > . . . . .  | 966 |
| site_set_base_< P, p_run< P > >  |     |
| mln::p_run< P > . . . . .  | 912 |
| mln::Object< p_run_psite< P > > . . . . .  | 844 |
| mln::Proxy< p_run_psite< P > > . . . . .   | 957 |
| mln::Site_Proxy< p_run_psite< P > > . . . . .  | 965 |
| mln::Pseudo_Site< p_run_psite< P > > . . . . .   | 957 |
| mln::Object< p_set< P > > . . . . .  | 844 |
| mln::Site_Set< p_set< P > > . . . . .  | 966 |
| site_set_base_< P, p_set< P > >  |     |
| mln::p_set< P > . . . . .  | 916 |
| mln::Object< p_set_of< S > > . . . . .   | 844 |
| mln::Site_Set< p_set_of< S > > . . . . .   | 966 |
| site_set_base_< S::site, p_set_of< S > >   |     |
| mln::p_set_of< S > . . . . .   | 921 |
| mln::Object< p_transformed< S, F > > . . . . .   | 844 |
| mln::Site_Set< p_transformed< S, F > > . . . . .   | 966 |
| site_set_base_< S::psite, p_transformed< S, F > >  |     |
| mln::p_transformed< S, F > . . . . .   | 924 |
| mln::Object< p_transformed_piter< Pi, S, F > > . . . . .   | 844 |
| mln::Proxy< p_transformed_piter< Pi, S, F > > . . . . .  | 957 |
| mln::Site_Proxy< p_transformed_piter< Pi, S, F > > . . . . .   | 965 |
| mln::Site_Iterator< p_transformed_piter< Pi, S, F > > . . . . .  | 963 |
| mln::Object< p_vaccess< V, S > > . . . . .   | 844 |
| mln::Site_Set< p_vaccess< V, S > > . . . . .   | 966 |
| site_set_base_< S::site, p_vaccess< V, S > >   |     |
| mln::p_vaccess< V, S > . . . . .   | 928 |
| mln::Object< p_vertices< G, F > > . . . . .  | 844 |
| mln::Site_Set< p_vertices< G, F > > . . . . .  | 966 |
| site_set_base_< F::result, p_vertices< G, F > >  |     |
| mln::p_vertices< G, F > . . . . .  | 932 |
| mln::Object< p_vertices_psite< G, F > > . . . . .  | 844 |
| mln::Proxy< p_vertices_psite< G, F > > . . . . .   | 957 |

|  |      |
|--|------|
| mln::Site_Proxy< p_vertices_psite< G, F > > . . . . .                            | 965  |
| mln::Pseudo_Site< p_vertices_psite< G, F > > . . . . .                           | 957  |
| mln::Object< pair< A1, A2 > > . . . . .  | 844  |
| mln::Meta_Accumulator< pair< A1, A2 > > . . . . .                                | 824  |
| mln::accu::meta::pair< A1, A2 > . . . . .  | 480  |
| mln::Object< pair< A1, A2, T > > . . . . .                                       | 844  |
| mln::Proxy< pair< A1, A2, T > > . . . . .  | 957  |
| mln::Accumulator< pair< A1, A2, T > > . . . . .                                  | 542  |
| base< std::pair< A1::result, A2::result >, pair< A1, A2, T > >                   |      |
| mln::accu::pair< A1, A2, T > . . . . .   | 499  |
| mln::Object< pair< min< V >, max< V >, mln_argument(min< V >) > > . . . . .      | 844  |
| mln::Proxy< pair< min< V >, max< V >, mln_argument(min< V >) > > . . . . .       | 957  |
| mln::Accumulator< pair< min< V >, max< V >, mln_argument(min< V >) > > . . . . . | 542  |
| mln::Object< pink_t > . . . . .  | 844  |
| mln::Literal< pink_t > . . . . .   | 799  |
| mln::literal::pink_t . . . . .   | 815  |
| mln::Object< pixel< I > > . . . . .  | 844  |
| mln::pixel< I > . . . . .  | 939  |
| mln::Object< plain< I > > . . . . .  | 844  |
| mln::Image< plain< I > > . . . . .   | 762  |
| mln::Object< point< G, C > > . . . . .   | 844  |
| mln::Site< point< G, C > > . . . . .   | 962  |
| mln::Gpoint< point< G, C > > . . . . .   | 726  |
| mln::point< G, C > . . . . .   | 946  |
| mln::Object< point_from_value< T > > . . . . .                                   | 844  |
| mln::Function< point_from_value< T > > . . . . .                                 | 715  |
| mln::Function_v2v< point_from_value< T > > . . . . .                             | 717  |
| mln::Object< projection< P, dir > > . . . . .                                    | 844  |
| mln::Function< projection< P, dir > > . . . . .                                  | 715  |
| mln::Function_v2v< projection< P, dir > > . . . . .                              | 717  |
| mln::Object< proxy< I > > . . . . .  | 844  |
| mln::Proxy< proxy< I > > . . . . .   | 957  |
| mln::value::proxy< I > . . . . .   | 1133 |
| mln::Object< purple_t > . . . . .  | 844  |
| mln::Literal< purple_t > . . . . .   | 799  |
| mln::literal::purple_t . . . . .   | 816  |
| mln::Object< qrde > . . . . .  | 844  |
| mln::Function< qrde > . . . . .  | 715  |
| mln::Function_v2v< qrde > . . . . .  | 717  |
| mln::Object< qt_rgb_to_int_u< n > > . . . . .                                    | 844  |
| mln::Function< qt_rgb_to_int_u< n > > . . . . .                                  | 715  |
| mln::Function_v2v< qt_rgb_to_int_u< n > > . . . . .                              | 717  |
| mln::Object< quat > . . . . .  | 844  |
| mln::Value< quat > . . . . .   | 1107 |
| mln::Object< rank > . . . . .  | 844  |
| mln::Meta_Accumulator< rank > . . . . .  | 824  |
| mln::accu::meta::stat::rank . . . . .  | 492  |
| mln::Object< rank< bool > > . . . . .  | 844  |

|  |      |
|--|------|
| mln::Proxy< rank< bool > > . . . . .               | 957  |
| mln::Accumulator< rank< bool > > . . . . .         | 542  |
| base< bool, rank< bool > >                         |      |
| mln::accu::stat::rank< bool > . . . . .            | 532  |
| mln::Object< rank< T > > . . . . .                 | 844  |
| mln::Proxy< rank< T > > . . . . .                  | 957  |
| mln::Accumulator< rank< T > > . . . . .            | 542  |
| base< const T &, rank< T > >                       |      |
| mln::accu::stat::rank< T > . . . . .               | 530  |
| mln::Object< rank_high_quant > . . . . .           | 844  |
| mln::Meta_Accumulator< rank_high_quant > . . . . . | 824  |
| mln::accu::meta::stat::rank_high_quant . . . . .   | 493  |
| mln::Object< rank_high_quant< T > > . . . . .      | 844  |
| mln::Proxy< rank_high_quant< T > > . . . . .       | 957  |
| mln::Accumulator< rank_high_quant< T > > . . . . . | 542  |
| base< const T &, rank_high_quant< T > >            |      |
| mln::accu::stat::rank_high_quant< T > . . . . .    | 533  |
| mln::Object< rectangle2d > . . . . .               | 844  |
| mln::Window< rectangle2d > . . . . .               | 1170 |
| window_base< dpoint2d, rectangle2d >               |      |
| classical_window_base< dpoint2d, rectangle2d >     |      |
| mln::win::rectangle2d . . . . .                    | 1169 |
| mln::Object< rectangularity< P > > . . . . .       | 844  |
| mln::Proxy< rectangularity< P > > . . . . .        | 957  |
| mln::Accumulator< rectangularity< P > > . . . . .  | 542  |
| mln::Object< red > . . . . .                       | 844  |
| mln::Meta_Function< red > . . . . .                | 826  |
| mln::Meta_Function_v2v< red > . . . . .            | 827  |
| mln::Object< red_t > . . . . .                     | 844  |
| mln::Literal< red_t > . . . . .                    | 799  |
| mln::literal::red_t . . . . .                      | 817  |
| mln::Object< ref_data > . . . . .                  | 844  |
| mln::Function< ref_data > . . . . .                | 715  |
| mln::Function_v2v< ref_data > . . . . .            | 717  |
| mln::Object< rgb32 > . . . . .                     | 844  |
| mln::Value< rgb32 > . . . . .                      | 1107 |
| mln::Object< rgb8_off_loader > . . . . .           | 844  |
| mln::Object< rgb8_off_saver > . . . . .            | 844  |
| mln::Object< rgb8_to_rgbn< n > > . . . . .         | 844  |
| mln::Function< rgb8_to_rgbn< n > > . . . . .       | 715  |
| mln::Function_v2v< rgb8_to_rgbn< n > > . . . . .   | 717  |
| mln::fun::v2v::rgb8_to_rgbn< n > . . . . .         | 682  |
| mln::Object< rgb< n > > . . . . .                  | 844  |
| mln::Value< rgb< n > > . . . . .                   | 1107 |
| mln::Object< rgb_to_dist< T, n > > . . . . .       | 844  |
| mln::Function< rgb_to_dist< T, n > > . . . . .     | 715  |
| mln::Function_v2v< rgb_to_dist< T, n > > . . . . . | 717  |
| mln::Object< rgb_to_int_u< n > > . . . . .         | 844  |
| mln::Function< rgb_to_int_u< n > > . . . . .       | 715  |

|  |     |
|--|-----|
| mln::Function_v2v< rgb_to_int_u< n > > . . . . .     | 717 |
| mln::Object< rgb_to_luma< T_luma > > . . . . .       | 844 |
| mln::Function< rgb_to_luma< T_luma > > . . . . .     | 715 |
| mln::Function_v2v< rgb_to_luma< T_luma > > . . . . . | 717 |
| mln::Object< rgbn_to_lbl8< n > > . . . . .           | 844 |
| mln::Function< rgbn_to_lbl8< n > > . . . . .         | 715 |
| mln::Function_v2v< rgbn_to_lbl8< n > > . . . . .     | 717 |
| mln::Object< rms > . . . . .                         | 844 |
| mln::Meta_Accumulator< rms > . . . . .               | 824 |
| mln::accu::meta::rms . . . . .                       | 481 |
| mln::Object< rms< T, V > > . . . . .                 | 844 |
| mln::Proxy< rms< T, V > > . . . . .                  | 957 |
| mln::Accumulator< rms< T, V > > . . . . .            | 542 |
| base< V, rms< T, V > > . . . . .                     |     |
| mln::accu::rms< T, V > . . . . .                     | 502 |
| mln::Object< rotation< n, C > > . . . . .            | 844 |
| mln::Function< rotation< n, C > > . . . . .          | 715 |
| mln::Function_v2v< rotation< n, C > > . . . . .      | 717 |
| mln::fun::x2x::rotation< n, C > . . . . .            | 706 |
| mln::Object< round< R > > . . . . .                  | 844 |
| mln::Function< round< R > > . . . . .                | 715 |
| mln::Function_v2v< round< R > > . . . . .            | 717 |
| mln::Object< row > . . . . .                         | 844 |
| mln::Meta_Function< row > . . . . .                  | 826 |
| mln::Meta_Function_v2v< row > . . . . .              | 827 |
| mln::Object< safe_image< I > > . . . . .             | 844 |
| mln::Image< safe_image< I > > . . . . .              | 762 |
| mln::Object< saturate< V > > . . . . .               | 844 |
| mln::Function< saturate< V > > . . . . .             | 715 |
| mln::Function_v2v< saturate< V > > . . . . .         | 717 |
| mln::Object< saturate_rgb8 > . . . . .               | 844 |
| mln::Function< saturate_rgb8 > . . . . .             | 715 |
| mln::Function_v2v< saturate_rgb8 > . . . . .         | 717 |
| mln::Object< scomp< ith > > . . . . .                | 844 |
| mln::Meta_Function< scomp< ith > > . . . . .         | 826 |
| mln::Meta_Function_v2v< scomp< ith > > . . . . .     | 827 |
| mln::Object< set_bkd_iter< T > > . . . . .           | 844 |
| mln::Proxy< set_bkd_iter< T > > . . . . .            | 957 |
| mln::Object< set_fwd_iter< T > > . . . . .           | 844 |
| mln::Proxy< set_fwd_iter< T > > . . . . .            | 957 |
| mln::Object< sharpness< I > > . . . . .              | 844 |
| mln::Proxy< sharpness< I > > . . . . .               | 957 |
| mln::Accumulator< sharpness< I > > . . . . .         | 542 |
| base< double, sharpness< I > > . . . . .             |     |
| mln::morpho::attribute::sharpness< I > . . . . .     | 836 |
| mln::Object< shell< F, I > > . . . . .               | 844 |
| mln::Proxy< shell< F, I > > . . . . .                | 957 |
| mln::Object< sign > . . . . .                        | 844 |

|  |      |
|--|------|
| mln::Value< sign > . . . . .   | 1107 |
| mln::Object< site_pair< P > > . . . . .                                      | 844  |
| mln::util::site_pair< P > . . . . .  | 1087 |
| mln::Object< sli > . . . . .   | 844  |
| mln::Meta_Function< sli > . . . . .  | 826  |
| mln::Meta_Function_v2v< sli > . . . . .                                      | 827  |
| mln::Object< slice_image< I > > . . . . .                                    | 844  |
| mln::Image< slice_image< I > > . . . . .                                     | 762  |
| mln::Object< snake_fwd_t > . . . . .   | 844  |
| mln::Browsing< snake_fwd_t > . . . . .                                       | 568  |
| mln::canvas::browsing::snake_fwd_t . . . . .                                 | 579  |
| mln::Object< snake_generic_t > . . . . .                                     | 844  |
| mln::Browsing< snake_generic_t > . . . . .                                   | 568  |
| mln::canvas::browsing::snake_generic_t . . . . .                             | 580  |
| mln::Object< snake_vert_t > . . . . .  | 844  |
| mln::Browsing< snake_vert_t > . . . . .                                      | 568  |
| mln::canvas::browsing::snake_vert_t . . . . .                                | 582  |
| mln::Object< soft_heap< T, R > > . . . . .                                   | 844  |
| mln::util::soft_heap< T, R > . . . . .                                       | 1089 |
| mln::Object< sqrt > . . . . .  | 844  |
| mln::Function< sqrt > . . . . .  | 715  |
| mln::Function_v2v< sqrt > . . . . .  | 717  |
| mln::Object< square > . . . . .  | 844  |
| mln::Mesh< square > . . . . .  | 823  |
| mln::Regular_Grid< square > . . . . .  | 960  |
| mln::Object< stack_image< n, I > > . . . . .                                 | 844  |
| mln::Image< stack_image< n, I > > . . . . .                                  | 762  |
| mln::Object< static_n_face_bkd_iter< N, D > > . . . . .                      | 844  |
| mln::Iterator< static_n_face_bkd_iter< N, D > > . . . . .                    | 787  |
| complex_iterator_base< topo::face< D >, static_n_face_bkd_iter< N, D > >     |      |
| complex_set_iterator_base< topo::face< D >, static_n_face_bkd_iter< N, D > > |      |
| mln::topo::static_n_face_bkd_iter< N, D > . . . . .                          | 1028 |
| mln::Object< static_n_face_fwd_iter< N, D > > . . . . .                      | 844  |
| mln::Iterator< static_n_face_fwd_iter< N, D > > . . . . .                    | 787  |
| complex_iterator_base< topo::face< D >, static_n_face_fwd_iter< N, D > >     |      |
| complex_set_iterator_base< topo::face< D >, static_n_face_fwd_iter< N, D > > |      |
| mln::topo::static_n_face_fwd_iter< N, D > . . . . .                          | 1029 |
| mln::Object< sub_image< I, S > > . . . . .                                   | 844  |
| mln::Image< sub_image< I, S > > . . . . .                                    | 762  |
| mln::Object< sub_image_if< I, S > > . . . . .                                | 844  |
| mln::Image< sub_image_if< I, S > > . . . . .                                 | 762  |
| mln::Object< sum > . . . . .   | 844  |
| mln::Meta_Accumulator< sum > . . . . .                                       | 824  |
| mln::accu::meta::math::sum . . . . .   | 475  |
| mln::Object< sum< I, S > > . . . . .   | 844  |
| mln::Proxy< sum< I, S > > . . . . .  | 957  |
| mln::Accumulator< sum< I, S > > . . . . .                                    | 542  |
| base< S, sum< I, S > >   |      |



|   |      |
|---|------|
| mln::morpho::attribute::sum< I, S > . . . . .       | 839  |
| mln::Object< sum< T, S > > . . . . .                | 844  |
| mln::Proxy< sum< T, S > > . . . . .                 | 957  |
| mln::Accumulator< sum< T, S > > . . . . .           | 542  |
| base< const S &, sum< T, S > >                      |      |
| mln::accu::math::sum< T, S > . . . . .              | 458  |
| mln::Object< sup > . . . . .                        | 844  |
| mln::Meta_Accumulator< sup > . . . . .              | 824  |
| mln::accu::meta::math::sup . . . . .                | 476  |
| mln::Meta_Function< sup > . . . . .                 | 826  |
| mln::Meta_Function_vv2v< sup > . . . . .            | 827  |
| mln::Object< sup< T > > . . . . .                   | 844  |
| mln::Proxy< sup< T > > . . . . .                    | 957  |
| mln::Accumulator< sup< T > > . . . . .              | 542  |
| base< const T &, sup< T > >                         |      |
| mln::accu::math::sup< T > . . . . .                 | 460  |
| mln::Object< tautology > . . . . .                  | 844  |
| mln::Function< tautology > . . . . .                | 715  |
| mln::Function_v2v< tautology > . . . . .            | 717  |
| mln::Function_v2b< tautology > . . . . .            | 716  |
| mln::fun::p2b::tautology . . . . .                  | 673  |
| mln::Object< teal_t > . . . . .                     | 844  |
| mln::Literal< teal_t > . . . . .                    | 799  |
| mln::literal::teal_t . . . . .                      | 818  |
| mln::Object< test > . . . . .                       | 844  |
| mln::Function< test > . . . . .                     | 715  |
| mln::Function_v2v< test > . . . . .                 | 717  |
| mln::Object< threshold< V > > . . . . .             | 844  |
| mln::Function< threshold< V > > . . . . .           | 715  |
| mln::Function_v2v< threshold< V > > . . . . .       | 717  |
| mln::Function_v2b< threshold< V > > . . . . .       | 716  |
| mln::fun::v2b::threshold< V > . . . . .             | 676  |
| mln::Object< thru_image< I, F > > . . . . .         | 844  |
| mln::Image< thru_image< I, F > > . . . . .          | 762  |
| mln::Object< thrubin_image< I1, I2, F > > . . . . . | 844  |
| mln::Image< thrubin_image< I1, I2, F > > . . . . .  | 762  |
| mln::Object< tick > . . . . .                       | 844  |
| mln::Mesh< tick > . . . . .                         | 823  |
| mln::Regular_Grid< tick > . . . . .                 | 960  |
| mln::Object< timer > . . . . .                      | 844  |
| mln::Proxy< timer > . . . . .                       | 957  |
| mln::util::timer . . . . .                          | 1092 |
| mln::Object< to16bits > . . . . .                   | 844  |
| mln::Function< to16bits > . . . . .                 | 715  |
| mln::Function_v2v< to16bits > . . . . .             | 717  |
| mln::Object< to19bits > . . . . .                   | 844  |
| mln::Function< to19bits > . . . . .                 | 715  |
| mln::Function_v2v< to19bits > . . . . .             | 717  |

|  |     |
|--|-----|
| mln::Object< to23bits > . . . . .  | 844 |
| mln::Function< to23bits > . . . . .  | 715 |
| mln::Function_v2v< to23bits > . . . . .  | 717 |
| mln::Object< to27bits > . . . . .  | 844 |
| mln::Function< to27bits > . . . . .  | 715 |
| mln::Function_v2v< to27bits > . . . . .  | 717 |
| mln::Object< to8bits > . . . . .   | 844 |
| mln::Function< to8bits > . . . . .   | 715 |
| mln::Function_v2v< to8bits > . . . . .   | 717 |
| mln::Object< tofloat01 > . . . . .   | 844 |
| mln::Function< tofloat01 > . . . . .   | 715 |
| mln::Function_v2v< tofloat01 > . . . . .   | 717 |
| mln::Object< tr_image< S, I, T > > . . . . .   | 844 |
| mln::Image< tr_image< S, I, T > > . . . . .  | 762 |
| mln::Object< transformed_image< I, F > > . . . . .   | 844 |
| mln::Image< transformed_image< I, F > > . . . . .  | 762 |
| mln::Object< translation< n, C > > . . . . .   | 844 |
| mln::Function< translation< n, C > > . . . . .   | 715 |
| mln::Function_v2v< translation< n, C > > . . . . .   | 717 |
| mln::fun::x2x::translation< n, C > . . . . .   | 710 |
| mln::Object< translation_t< P > > . . . . .  | 844 |
| mln::Function< translation_t< P > > . . . . .  | 715 |
| mln::Function_v2v< translation_t< P > > . . . . .  | 717 |
| mln::Object< tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T)> > . . . . .   | 844 |
| mln::Proxy< tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T)> > . . . . .  | 957 |
| mln::Accumulator< tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T)> > . . . . .  | 542 |
| base< boost::tuple< BOOST_PP_REPEAT(10, RESULT_ACCU, Le Ricard ya que ca de<br>vrai!) >, tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T)> > . . . . . |     |
| mln::accu::tuple< A, n, > . . . . .  | 539 |
| mln::Object< tuple< n, BOOST_PP_ENUM_PARAMS(10, T)> > . . . . .  | 844 |
| mln::Meta_Accumulator< tuple< n, BOOST_PP_ENUM_PARAMS(10, T)> > . . . . .  | 824 |
| mln::accu::meta::tuple< n, > . . . . .   | 494 |
| mln::Object< unary< Fun, T > > . . . . .   | 844 |
| mln::Function< unary< Fun, T > > . . . . .   | 715 |
| mln::Function_v2v< unary< Fun, T > > . . . . .   | 717 |
| mln::Object< unproject_image< I, D, F > > . . . . .  | 844 |
| mln::Image< unproject_image< I, D, F > > . . . . .   | 762 |
| mln::Object< up_leaf_piter< T > > . . . . .  | 844 |
| mln::Proxy< up_leaf_piter< T > > . . . . .   | 957 |
| mln::Site_Proxy< up_leaf_piter< T > > . . . . .  | 965 |
| mln::Site_Iterator< up_leaf_piter< T > > . . . . .   | 963 |
| mln::Object< up_node_piter< T > > . . . . .  | 844 |
| mln::Proxy< up_node_piter< T > > . . . . .   | 957 |
| mln::Site_Proxy< up_node_piter< T > > . . . . .  | 965 |
| mln::Site_Iterator< up_node_piter< T > > . . . . .   | 963 |
| mln::Object< up_site_piter< T > > . . . . .  | 844 |
| mln::Proxy< up_site_piter< T > > . . . . .   | 957 |
| mln::Site_Proxy< up_site_piter< T > > . . . . .  | 965 |

|  |      |
|--|------|
| mln::Site_Iterator< up_site_piter< T > > . . . . .           | 963  |
| mln::Object< val< A > > . . . . .                            | 844  |
| mln::Proxy< val< A > > . . . . .                             | 957  |
| mln::Accumulator< val< A > > . . . . .                       | 542  |
| base< const A::result &, val< A > >                          |      |
| mln::accu::val< A > . . . . .                                | 541  |
| mln::Object< val< mA > > . . . . .                           | 844  |
| mln::Meta_Accumulator< val< mA > > . . . . .                 | 824  |
| mln::accu::meta::val< mA > . . . . .                         | 495  |
| mln::Object< value_at_index< bool > > . . . . .              | 844  |
| mln::Function< value_at_index< bool > > . . . . .            | 715  |
| mln::Function_v2v< value_at_index< bool > > . . . . .        | 717  |
| mln::Object< value_at_index< T > > . . . . .                 | 844  |
| mln::Function< value_at_index< T > > . . . . .               | 715  |
| mln::Function_v2v< value_at_index< T > > . . . . .           | 717  |
| mln::Object< var< T > > . . . . .                            | 844  |
| mln::Proxy< var< T > > . . . . .                             | 957  |
| mln::Accumulator< var< T > > . . . . .                       | 542  |
| base< algebra::mat< T::dim, T::dim, float >, var< T > >      |      |
| mln::accu::stat::var< T > . . . . .                          | 535  |
| mln::Object< variance< T, S, R > > . . . . .                 | 844  |
| mln::Proxy< variance< T, S, R > > . . . . .                  | 957  |
| mln::Accumulator< variance< T, S, R > > . . . . .            | 542  |
| base< R, variance< T, S, R > >                               |      |
| mln::accu::stat::variance< T, S, R > . . . . .               | 537  |
| mln::Object< vec< 1, T > > . . . . .                         | 844  |
| mln::Object< vec< 2, T > > . . . . .                         | 844  |
| mln::Object< vec< 3, T > > . . . . .                         | 844  |
| mln::Object< vec< 4, T > > . . . . .                         | 844  |
| mln::Object< vec< n, C > > . . . . .                         | 844  |
| mln::Object< vec< n, T > > . . . . .                         | 844  |
| mln::Object< vec< V > > . . . . .                            | 844  |
| mln::Function< vec< V > > . . . . .                          | 715  |
| mln::Function_vv2v< vec< V > > . . . . .                     | 718  |
| mln::fun::vv2v::vec< V > . . . . .                           | 701  |
| mln::Object< vertex< G > > . . . . .                         | 844  |
| mln::Site< vertex< G > > . . . . .                           | 962  |
| mln::util::vertex< G > . . . . .                             | 1102 |
| mln::Object< vertex_bkd_iterator< G > > . . . . .            | 844  |
| mln::Proxy< vertex_bkd_iterator< G > > . . . . .             | 957  |
| mln::Object< vertex_fwd_iterator< G > > . . . . .            | 844  |
| mln::Proxy< vertex_fwd_iterator< G > > . . . . .             | 957  |
| mln::Object< vertex_image< P, V, G > > . . . . .             | 844  |
| mln::Image< vertex_image< P, V, G > > . . . . .              | 762  |
| mln::Object< vertex_nbh_edge_bkd_iterator< G > > . . . . .   | 844  |
| mln::Proxy< vertex_nbh_edge_bkd_iterator< G > > . . . . .    | 957  |
| mln::Object< vertex_nbh_edge_fwd_iterator< G > > . . . . .   | 844  |
| mln::Proxy< vertex_nbh_edge_fwd_iterator< G > > . . . . .    | 957  |
| mln::Object< vertex_nbh_vertex_bkd_iterator< G > > . . . . . | 844  |

|  |      |
|--|------|
| mln::Proxy< vertex_nbh_vertex_bkd_iterator< G > > . . . . .  | 957  |
| mln::Object< vertex_nbh_vertex_fwd_iterator< G > > . . . . . | 844  |
| mln::Proxy< vertex_nbh_vertex_fwd_iterator< G > > . . . . .  | 957  |
| mln::Object< violent_cast_image< T, I > > . . . . .          | 844  |
| mln::Image< violent_cast_image< T, I > > . . . . .           | 762  |
| mln::Object< violet_t > . . . . .                            | 844  |
| mln::Literal< violet_t > . . . . .                           | 799  |
| mln::literal::violet_t . . . . .                             | 819  |
| mln::Object< viota_t > . . . . .                             | 844  |
| mln::Function< viota_t > . . . . .                           | 715  |
| mln::Function_v2v< viota_t > . . . . .                       | 717  |
| mln::Object< viota_t< S > > . . . . .                        | 844  |
| mln::Function< viota_t< S > > . . . . .                      | 715  |
| mln::Function_v2v< viota_t< S > > . . . . .                  | 717  |
| mln::Object< volume > . . . . .                              | 844  |
| mln::Meta_Accumulator< volume > . . . . .                    | 824  |
| mln::accu::meta::shape::volume . . . . .                     | 484  |
| mln::Object< volume< I > > . . . . .                         | 844  |
| mln::Proxy< volume< I > > . . . . .                          | 957  |
| mln::Accumulator< volume< I > > . . . . .                    | 542  |
| base< unsigned, volume< I > >                                |      |
| mln::accu::shape::volume< I > . . . . .                      | 507  |
| mln::morpho::attribute::volume< I > . . . . .                | 840  |
| mln::Object< W > . . . . .                                   | 844  |
| mln::Object< w_window< D, W > > . . . . .                    | 844  |
| mln::Weighted_Window< w_window< D, W > > . . . . .           | 1158 |
| weighted_window_base< mln::window< D >, w_window< D, W > >   |      |
| mln::w_window< D, W > . . . . .                              | 1154 |
| mln::Object< white_gaussian< V > > . . . . .                 | 844  |
| mln::Function< white_gaussian< V > > . . . . .               | 715  |
| mln::Function_n2v< white_gaussian< V > > . . . . .           | 716  |
| mln::fun::n2v::white_gaussian< V > . . . . .                 | 672  |
| mln::Object< white_t > . . . . .                             | 844  |
| mln::Literal< white_t > . . . . .                            | 799  |
| mln::literal::white_t . . . . .                              | 820  |
| mln::Object< window< D > > . . . . .                         | 844  |
| mln::Window< window< D > > . . . . .                         | 1170 |
| window_base< D, window< D > >                                |      |
| mln::window< D > . . . . .                                   | 1171 |
| mln::Object< wrap > . . . . .                                | 844  |
| mln::Function< wrap > . . . . .                              | 715  |
| mln::Function_v2v< wrap > . . . . .                          | 717  |
| mln::Object< wrap< L > > . . . . .                           | 844  |
| mln::Function< wrap< L > > . . . . .                         | 715  |
| mln::Function_v2v< wrap< L > > . . . . .                     | 717  |
| mln::Object< yellow_t > . . . . .                            | 844  |
| mln::Literal< yellow_t > . . . . .                           | 799  |
| mln::literal::yellow_t . . . . .                             | 821  |

|  |      |
|--|------|
| mln::Object< yes > . . . . .                             | 844  |
| mln::util::yes . . . . .                                 | 1107 |
| mln::Object< zero_t > . . . . .                          | 844  |
| mln::Literal< zero_t > . . . . .                         | 799  |
| mln::literal::zero_t . . . . .                           | 822  |
| mln::internal::pixel_impl_< I, bkd_pixter1d< I > >       |      |
| pixel_iterator_base_< I, bkd_pixter1d< I > >             |      |
| mln::internal::pixel_impl_< I, bkd_pixter2d< I > >       |      |
| pixel_iterator_base_< I, bkd_pixter2d< I > >             |      |
| mln::internal::pixel_impl_< I, bkd_pixter3d< I > >       |      |
| pixel_iterator_base_< I, bkd_pixter3d< I > >             |      |
| mln::internal::pixel_impl_< I, dpoints_bkd_pixter< I > > |      |
| mln::dpoints_bkd_pixter< I > . . . . .                   | 648  |
| mln::internal::pixel_impl_< I, dpoints_fwd_pixter< I > > |      |
| mln::dpoints_fwd_pixter< I > . . . . .                   | 652  |
| mln::internal::pixel_impl_< I, fwd_pixter1d< I > >       |      |
| pixel_iterator_base_< I, fwd_pixter1d< I > >             |      |
| mln::internal::pixel_impl_< I, fwd_pixter2d< I > >       |      |
| pixel_iterator_base_< I, fwd_pixter2d< I > >             |      |
| mln::internal::pixel_impl_< I, fwd_pixter3d< I > >       |      |
| pixel_iterator_base_< I, fwd_pixter3d< I > >             |      |
| mln::internal::pixel_impl_< I, pixel< I > >              |      |
| mln::pixel< I > . . . . .                                | 939  |
| trait::graph< I > . . . . .                              | 1177 |
| trait::graph< mln::complex_image< 1, G, V > > . . . . .  | 1178 |
| trait::graph< mln::image2d< T > > . . . . .              | 1178 |



# Chapter 7

## Class Index

### 7.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

|  |     |
|--|-----|
| <a href="#">mln::accu::center&lt; P, V &gt;</a> (Mass center accumulator) . . . . .  | 437 |
| <a href="#">mln::accu::convolve&lt; T1, T2, R &gt;</a> (Generic convolution accumulator class) . . . . .   | 439 |
| <a href="#">mln::accu::count_adjacent_vertices&lt; F, S &gt;</a> ( <a href="#">Accumulator</a> class counting the number of vertices adjacent to a set of <a href="#">mln::p_edges_psite</a> (i.e., a set of edges)) . . . . . | 440 |
| <a href="#">mln::accu::count_labels&lt; L &gt;</a> (Count the number of different labels in an image) . . . . .  | 442 |
| <a href="#">mln::accu::count_value&lt; V &gt;</a> (Define an accumulator that counts the occurrence of a given value) . . . . .  | 443 |
| <a href="#">mln::accu::histo&lt; V &gt;</a> (Generic histogram class over a value set with type $V$ ) . . . . .  | 445 |
| <a href="#">mln::accu::label_used&lt; L &gt;</a> (References all the labels used) . . . . .  | 446 |
| <a href="#">mln::accu::logic::land</a> ("Logical-and" accumulator) . . . . .   | 448 |
| <a href="#">mln::accu::logic::land_basic</a> ("Logical-and" accumulator) . . . . .   | 449 |
| <a href="#">mln::accu::logic::lor</a> ("Logical-or" accumulator) . . . . .   | 451 |
| <a href="#">mln::accu::logic::lor_basic</a> ("Logical-or" accumulator class) . . . . .   | 452 |
| <a href="#">mln::accu::maj_h&lt; T &gt;</a> (Compute the majority value) . . . . .   | 454 |
| <a href="#">mln::accu::math::count&lt; T &gt;</a> (Generic counter accumulator) . . . . .  | 455 |
| <a href="#">mln::accu::math::inf&lt; T &gt;</a> (Generic inf accumulator class) . . . . .  | 457 |
| <a href="#">mln::accu::math::sum&lt; T, S &gt;</a> (Generic sum accumulator class) . . . . .   | 458 |
| <a href="#">mln::accu::math::sup&lt; T &gt;</a> (Generic sup accumulator class) . . . . .  | 460 |
| <a href="#">mln::accu::max_site&lt; I &gt;</a> (Define an accumulator that computes the first site with the maximum value in an image) . . . . .   | 461 |
| <a href="#">mln::accu::meta::center</a> (Meta accumulator for center) . . . . .  | 463 |
| <a href="#">mln::accu::meta::count_adjacent_vertices</a> (Meta accumulator for <a href="#">count_adjacent_vertices</a> ) . . . . .   | 463 |
| <a href="#">mln::accu::meta::count_labels</a> (Meta accumulator for <a href="#">count_labels</a> ) . . . . .   | 464 |
| <a href="#">mln::accu::meta::count_value</a> (FIXME: How to write a meta accumulator with a constructor taking a generic argument? Meta accumulator for <a href="#">count_value</a> ) . . . . .                                | 465 |
| <a href="#">mln::accu::meta::histo</a> (Meta accumulator for <a href="#">histo</a> ) . . . . .   | 466 |
| <a href="#">mln::accu::meta::label_used</a> (Meta accumulator for <a href="#">label_used</a> ) . . . . .   | 467 |
| <a href="#">mln::accu::meta::logic::land</a> (Meta accumulator for <a href="#">land</a> ) . . . . .  | 468 |
| <a href="#">mln::accu::meta::logic::land_basic</a> (Meta accumulator for <a href="#">land_basic</a> ) . . . . .  | 469 |
| <a href="#">mln::accu::meta::logic::lor</a> (Meta accumulator for <a href="#">lor</a> ) . . . . .  | 470 |
| <a href="#">mln::accu::meta::logic::lor_basic</a> (Meta accumulator for <a href="#">lor_basic</a> ) . . . . .  | 471 |
| <a href="#">mln::accu::meta::maj_h</a> (Meta accumulator for <a href="#">maj_h</a> ) . . . . .   | 472 |
| <a href="#">mln::accu::meta::math::count</a> (Meta accumulator for <a href="#">count</a> ) . . . . .   | 473 |
| <a href="#">mln::accu::meta::math::inf</a> (Meta accumulator for <a href="#">inf</a> ) . . . . .   | 474 |

|   |     |
|---|-----|
| <code>mln::accu::meta::math::sum</code> (Meta accumulator for sum) . . . . .  | 475 |
| <code>mln::accu::meta::math::sup</code> (Meta accumulator for sup) . . . . .  | 476 |
| <code>mln::accu::meta::max_site</code> (Meta accumulator for <code>max_site</code> ) . . . . .  | 477 |
| <code>mln::accu::meta::nil</code> (Meta accumulator for nil) . . . . .  | 478 |
| <code>mln::accu::meta::p&lt; mA &gt;</code> (Meta accumulator for p) . . . . .  | 479 |
| <code>mln::accu::meta::pair&lt; A1, A2 &gt;</code> (Meta accumulator for pair) . . . . .  | 480 |
| <code>mln::accu::meta::rms</code> (Meta accumulator for rms) . . . . .  | 481 |
| <code>mln::accu::meta::shape::bbox</code> (Meta accumulator for bbox) . . . . .   | 482 |
| <code>mln::accu::meta::shape::height</code> (Meta accumulator for height) . . . . .   | 483 |
| <code>mln::accu::meta::shape::volume</code> (Meta accumulator for volume) . . . . .   | 484 |
| <code>mln::accu::meta::stat::max</code> (Meta accumulator for max) . . . . .  | 485 |
| <code>mln::accu::meta::stat::max_h</code> (Meta accumulator for max) . . . . .  | 486 |
| <code>mln::accu::meta::stat::mean</code> (Meta accumulator for mean) . . . . .  | 487 |
| <code>mln::accu::meta::stat::median_alt&lt; T &gt;</code> (Meta accumulator for <code>median_alt</code> ) . . . . .   | 488 |
| <code>mln::accu::meta::stat::median_h</code> (Meta accumulator for <code>median_h</code> ) . . . . .  | 489 |
| <code>mln::accu::meta::stat::min</code> (Meta accumulator for min) . . . . .  | 490 |
| <code>mln::accu::meta::stat::min_h</code> (Meta accumulator for min) . . . . .  | 491 |
| <code>mln::accu::meta::stat::rank</code> (Meta accumulator for rank) . . . . .  | 492 |
| <code>mln::accu::meta::stat::rank_high_quant</code> (Meta accumulator for <code>rank_high_quant</code> ) . . . . .  | 493 |
| <code>mln::accu::meta::tuple&lt; n, &gt;</code> (Meta accumulator for tuple) . . . . .  | 494 |
| <code>mln::accu::meta::val&lt; mA &gt;</code> (Meta accumulator for val) . . . . .  | 495 |
| <code>mln::accu::nil&lt; T &gt;</code> (Define an accumulator that does nothing) . . . . .  | 496 |
| <code>mln::accu::p&lt; A &gt;</code> (Generic p of accumulators) . . . . .  | 498 |
| <code>mln::accu::pair&lt; A1, A2, T &gt;</code> (Generic pair of accumulators) . . . . .  | 499 |
| <code>mln::accu::rms&lt; T, V &gt;</code> (Generic root mean square accumulator class) . . . . .  | 502 |
| <code>mln::accu::shape::bbox&lt; P &gt;</code> (Generic bounding box accumulator class) . . . . .   | 504 |
| <code>mln::accu::shape::height&lt; I &gt;</code> (Height accumulator) . . . . .   | 505 |
| <code>mln::accu::shape::volume&lt; I &gt;</code> (Volume accumulator class) . . . . .   | 507 |
| <code>mln::accu::site_set::rectangularity&lt; P &gt;</code> (Compute the rectangularity of a site set) . . . . .  | 509 |
| <code>mln::accu::stat::deviation&lt; T, S, M &gt;</code> (Generic standard deviation accumulator class) . . . . .   | 511 |
| <code>mln::accu::stat::histo3d_rgb&lt; V &gt;</code> (Define a histogram as accumulator which returns an <code>image3d</code> )                             | 513 |
| <code>mln::accu::stat::max&lt; T &gt;</code> (Generic max accumulator class) . . . . .  | 515 |
| <code>mln::accu::stat::max_h&lt; V &gt;</code> (Generic max function based on histogram over a value set with<br>type V) . . . . .                          | 517 |
| <code>mln::accu::stat::mean&lt; T, S, M &gt;</code> (Generic mean accumulator class) . . . . .  | 518 |
| <code>mln::accu::stat::median_alt&lt; S &gt;</code> (Generic <code>median_alt</code> function based on histogram over a value<br>set with type S) . . . . . | 520 |
| <code>mln::accu::stat::median_h&lt; V &gt;</code> (Generic median function based on histogram over a value set<br>with type V) . . . . .                    | 522 |
| <code>mln::accu::stat::meta::deviation</code> (Meta accumulator for deviation) . . . . .  | 523 |
| <code>mln::accu::stat::min&lt; T &gt;</code> (Generic min accumulator class) . . . . .  | 524 |
| <code>mln::accu::stat::min_h&lt; V &gt;</code> (Generic min function based on histogram over a value set with<br>type V) . . . . .                          | 526 |
| <code>mln::accu::stat::min_max&lt; V &gt;</code> (Generic min and max accumulator class) . . . . .  | 527 |
| <code>mln::accu::stat::rank&lt; T &gt;</code> (Generic rank accumulator class) . . . . .  | 530 |
| <code>mln::accu::stat::rank&lt; bool &gt;</code> (Rank accumulator class for Boolean) . . . . .   | 532 |
| <code>mln::accu::stat::rank_high_quant&lt; T &gt;</code> (Generic rank accumulator class) . . . . .   | 533 |
| <code>mln::accu::stat::var&lt; T &gt;</code> (Var accumulator class) . . . . .  | 535 |
| <code>mln::accu::stat::variance&lt; T, S, R &gt;</code> (Variance accumulator class) . . . . .  | 537 |
| <code>mln::accu::tuple&lt; A, n, &gt;</code> (Generic tuple of accumulators) . . . . .  | 539 |
| <code>mln::accu::val&lt; A &gt;</code> (Generic val of accumulators) . . . . .  | 541 |
| <code>mln::Accumulator&lt; E &gt;</code> (Base class for implementation of accumulators) . . . . .  | 542 |
| <code>mln::algebra::h_mat&lt; d, T &gt;</code> (N-Dimensional matrix with homogeneous coordinates) . . . . .  | 544 |
| <code>mln::algebra::h_vec&lt; d, C &gt;</code> (N-Dimensional vector with homogeneous coordinates) . . . . .  | 546 |



|  |     |
|--|-----|
| <a href="#">mln::bkd_pixter1d&lt; I &gt;</a> (Backward pixel iterator on a 1-D image with border)  | 548 |
| <a href="#">mln::bkd_pixter2d&lt; I &gt;</a> (Backward pixel iterator on a 2-D image with border)  | 549 |
| <a href="#">mln::bkd_pixter3d&lt; I &gt;</a> (Backward pixel iterator on a 3-D image with border)  | 551 |
| <a href="#">mln::box&lt; P &gt;</a> (Generic box class: site set containing points of a regular grid)  | 552 |
| <a href="#">mln::Box&lt; E &gt;</a> (Base class for implementation classes of boxes)   | 560 |
| <a href="#">mln::box_runend_piter&lt; P &gt;</a> (A generic backward iterator on points by lines)  | 566 |
| <a href="#">mln::box_runstart_piter&lt; P &gt;</a> (A generic forward iterator on points by lines)   | 567 |
| <a href="#">mln::Browsing&lt; E &gt;</a> (Base class for implementation classes that are browsings)  | 568 |
| <a href="#">mln::canvas::browsing::backdiagonal2d_t</a> ( <a href="#">Browsing</a> in a certain direction)                                       | 569 |
| <a href="#">mln::canvas::browsing::breadth_first_search_t</a> (Breadth-first search algorithm for graph, on vertices)                            | 571 |
| <a href="#">mln::canvas::browsing::depth_first_search_t</a> (Breadth-first search algorithm for graph, on vertices)                              | 571 |
| <a href="#">mln::canvas::browsing::diagonal2d_t</a> ( <a href="#">Browsing</a> in a certain direction)   | 571 |
| <a href="#">mln::canvas::browsing::dir_struct_elt_incr_update_t</a> ( <a href="#">Browsing</a> in a certain direction with a segment)            | 573 |
| <a href="#">mln::canvas::browsing::directional_t</a> ( <a href="#">Browsing</a> in a certain direction)  | 574 |
| <a href="#">mln::canvas::browsing::fwd_t</a> (Canvas for forward browsing)   | 576 |
| <a href="#">mln::canvas::browsing::hyper_directional_t</a> ( <a href="#">Browsing</a> in a certain direction)                                    | 577 |
| <a href="#">mln::canvas::browsing::snake_fwd_t</a> ( <a href="#">Browsing</a> in a snake-way, forward)   | 579 |
| <a href="#">mln::canvas::browsing::snake_generic_t</a> (Multidimensional <a href="#">Browsing</a> in a given-way)                                | 580 |
| <a href="#">mln::canvas::browsing::snake_vert_t</a> ( <a href="#">Browsing</a> in a snake-way, forward)  | 582 |
| <a href="#">mln::canvas::chamfer&lt; F &gt;</a> (Compute chamfer distance)   | 583 |
| <a href="#">mln::category&lt; R(*) (A) &gt;</a> (Category declaration for a unary C function)  | 583 |
| <a href="#">mln::complex_image&lt; D, G, V &gt;</a> ( <a href="#">Image</a> based on a complex)  | 584 |
| <a href="#">mln::complex_neighborhood_bkd_piter&lt; I, G, N &gt;</a> (Backward iterator on complex neighborhood)                                 | 587 |
| <a href="#">mln::complex_neighborhood_fwd_piter&lt; I, G, N &gt;</a> (Forward iterator on complex neighborhood)                                  | 588 |
| <a href="#">mln::complex_psite&lt; D, G &gt;</a> (Point site associated to a <a href="#">mln::p_complex</a> )                                    | 590 |
| <a href="#">mln::complex_window_bkd_piter&lt; I, G, W &gt;</a> (Backward iterator on complex window)   | 593 |
| <a href="#">mln::complex_window_fwd_piter&lt; I, G, W &gt;</a> (Forward iterator on complex window)  | 595 |
| <a href="#">mln::decorated_image&lt; I, D &gt;</a> ( <a href="#">Image</a> that can have additional features)                                    | 596 |
| <a href="#">mln::Delta_Point_Site&lt; void &gt;</a> (Delta point site category flag type)  | 599 |
| <a href="#">mln::doc::Accumulator&lt; E &gt;</a> (Documentation class for <a href="#">mln::Accumulator</a> )                                     | 599 |
| <a href="#">mln::doc::Box&lt; E &gt;</a> (Documentation class for <a href="#">mln::Box</a> )   | 601 |
| <a href="#">mln::doc::Dpoint&lt; E &gt;</a> (Documentation class for <a href="#">mln::Dpoint</a> )   | 604 |
| <a href="#">mln::doc::Fastest_Image&lt; E &gt;</a> (Documentation class for the concept of images that have the speed property set to "fastest") | 606 |
| <a href="#">mln::doc::Generalized_Pixel&lt; E &gt;</a> (Documentation class for <a href="#">mln::Generalized_Pixel</a> )                         | 614 |
| <a href="#">mln::doc::Image&lt; E &gt;</a> (Documentation class for <a href="#">mln::Image</a> )   | 616 |
| <a href="#">mln::doc::Iterator&lt; E &gt;</a> (Documentation class for <a href="#">mln::Iterator</a> )   | 622 |
| <a href="#">mln::doc::Neighborhood&lt; E &gt;</a> (Documentation class for <a href="#">mln::Neighborhood</a> )                                   | 623 |
| <a href="#">mln::doc::Object&lt; E &gt;</a> (Documentation class for <a href="#">mln::Object</a> )   | 625 |
| <a href="#">mln::doc::Pixel_Iterator&lt; E &gt;</a> (Documentation class for <a href="#">mln::Iterator</a> )                                     | 625 |
| <a href="#">mln::doc::Point_Site&lt; E &gt;</a> (Documentation class for <a href="#">mln::Point_Site</a> )                                       | 628 |
| <a href="#">mln::doc::Site_Iterator&lt; E &gt;</a> (Documentation class for <a href="#">mln::Site_Iterator</a> )                                 | 631 |
| <a href="#">mln::doc::Site_Set&lt; E &gt;</a> (Documentation class for <a href="#">mln::Site_Set</a> )   | 633 |
| <a href="#">mln::doc::Value_Iterator&lt; E &gt;</a> (Documentation class for <a href="#">mln::Value_Iterator</a> )                               | 635 |
| <a href="#">mln::doc::Value_Set&lt; E &gt;</a> (Documentation class for <a href="#">mln::Value_Set</a> )   | 636 |
| <a href="#">mln::doc::Weighted_Window&lt; E &gt;</a> (Documentation class for <a href="#">mln::Weighted_Window</a> )                             | 639 |
| <a href="#">mln::doc::Window&lt; E &gt;</a> (Documentation class for <a href="#">mln::Window</a> )   | 641 |
| <a href="#">mln::Dpoint&lt; E &gt;</a> (Base class for implementation of delta-point classes)  | 643 |
| <a href="#">mln::dpoint&lt; G, C &gt;</a> (Generic delta-point class)  | 644 |

|  |     |
|--|-----|
| <code>mln::dpoints_bkd_pixter&lt; I &gt;</code> (A generic backward iterator on the pixels of a dpoint-based window or neighborhood) . . . . .   | 648 |
| <code>mln::dpoints_fwd_pixter&lt; I &gt;</code> (A generic forward iterator on the pixels of a dpoint-based window or neighborhood) . . . . .  | 652 |
| <code>mln::Edge&lt; E &gt;</code> (Edge category flag type) . . . . .  | 655 |
| <code>mln::edge_image&lt; P, V, G &gt;</code> (Image based on graph edges) . . . . .   | 655 |
| <code>mln::extended&lt; I &gt;</code> (Makes an image become restricted by a point set) . . . . .  | 657 |
| <code>mln::extension_fun&lt; I, F &gt;</code> (Extends the domain of an image with a function) . . . . .   | 659 |
| <code>mln::extension_ima&lt; I, J &gt;</code> (Extends the domain of an image with an image) . . . . .   | 661 |
| <code>mln::extension_val&lt; I &gt;</code> (Extends the domain of an image with a value) . . . . .   | 664 |
| <code>mln::faces_psite&lt; N, D, P &gt;</code> (Point site associated to a <code>mln::p_faces</code> ) . . . . .   | 666 |
| <code>mln::flat_image&lt; T, S &gt;</code> (Image with a single value) . . . . .   | 669 |
| <code>mln::fun::from_accu&lt; A &gt;</code> (Wrap an accumulator into a function) . . . . .  | 671 |
| <code>mln::fun::n2v::white_gaussian&lt; V &gt;</code> (Generate a White Gaussian Noise) . . . . .  | 672 |
| <code>mln::fun::p2b::antilogy</code> (A <code>p2b</code> function always returning <code>false</code> ) . . . . .  | 672 |
| <code>mln::fun::p2b::tautology</code> (A <code>p2b</code> function always returning <code>true</code> ) . . . . .  | 673 |
| <code>mln::fun::v2b::lnot&lt; V &gt;</code> (Functor computing logical-not on a value) . . . . .   | 675 |
| <code>mln::fun::v2b::threshold&lt; V &gt;</code> (Threshold function) . . . . .  | 676 |
| <code>mln::fun::v2v::ch_function_value&lt; F, V &gt;</code> (Wrap a function <code>v2v</code> and convert its result to another type) . . . . .  | 677 |
| <code>mln::fun::v2v::component&lt; T, i &gt;</code> (Functor that accesses the <code>i</code> -th component of a value) . . . . .  | 678 |
| <code>mln::fun::v2v::l1_norm&lt; V, R &gt;</code> (L1-norm) . . . . .  | 678 |
| <code>mln::fun::v2v::l2_norm&lt; V, R &gt;</code> (L2-norm) . . . . .  | 679 |
| <code>mln::fun::v2v::linear&lt; V, T, R &gt;</code> (Linear function. $f(v) = a * v + b$ . <code>V</code> is the type of input values; <code>T</code> is the type used to compute the result; <code>R</code> is the result type) . . . . . | 680 |
| <code>mln::fun::v2v::linfty_norm&lt; V, R &gt;</code> (L-infty norm) . . . . .   | 681 |
| <code>mln::fun::v2v::rgb8_to_rgbn&lt; n &gt;</code> (Convert a <code>rgb8</code> value to a <code>rgn</code> , $n < 8$ ) . . . . .   | 682 |
| <code>mln::fun::v2w2v::cos&lt; V &gt;</code> (Cosinus bijective functor) . . . . .   | 684 |
| <code>mln::fun::v2w_w2v::l1_norm&lt; V, R &gt;</code> (L1-norm) . . . . .  | 685 |
| <code>mln::fun::v2w_w2v::l2_norm&lt; V, R &gt;</code> (L2-norm) . . . . .  | 687 |
| <code>mln::fun::v2w_w2v::linfty_norm&lt; V, R &gt;</code> (L-infty norm) . . . . .   | 688 |
| <code>mln::fun::vv2b::eq&lt; L, R &gt;</code> (Functor computing equal between two values) . . . . .   | 689 |
| <code>mln::fun::vv2b::ge&lt; L, R &gt;</code> (Functor computing "greater or equal than" between two values) . . . . .   | 689 |
| <code>mln::fun::vv2b::gt&lt; L, R &gt;</code> (Functor computing "greater than" between two values) . . . . .  | 690 |
| <code>mln::fun::vv2b::implies&lt; L, R &gt;</code> (Functor computing logical-implies between two values) . . . . .  | 691 |
| <code>mln::fun::vv2b::le&lt; L, R &gt;</code> (Functor computing "lower or equal than" between two values) . . . . .   | 692 |
| <code>mln::fun::vv2b::lt&lt; L, R &gt;</code> (Functor computing "lower than" between two values) . . . . .  | 693 |
| <code>mln::fun::vv2v::diff_abs&lt; V &gt;</code> (A functor computing the <code>diff_absimum</code> of two values) . . . . .   | 694 |
| <code>mln::fun::vv2v::land&lt; L, R &gt;</code> (Functor computing logical-and between two values) . . . . .   | 695 |
| <code>mln::fun::vv2v::land_not&lt; L, R &gt;</code> (Functor computing logical and-not between two values) . . . . .   | 696 |
| <code>mln::fun::vv2v::lor&lt; L, R &gt;</code> (Functor computing logical-or between two values) . . . . .   | 697 |
| <code>mln::fun::vv2v::lxor&lt; L, R &gt;</code> (Functor computing logical-xor between two values) . . . . .   | 698 |
| <code>mln::fun::vv2v::max&lt; V &gt;</code> (A functor computing the maximum of two values) . . . . .  | 699 |
| <code>mln::fun::vv2v::min&lt; L, R &gt;</code> (A functor computing the minimum of two values) . . . . .   | 700 |
| <code>mln::fun::vv2v::vec&lt; V &gt;</code> (A functor computing the vecimum of two values) . . . . .  | 701 |
| <code>mln::fun::x2p::closest_point&lt; P &gt;</code> (FIXME: doxygen + concept checking) . . . . .   | 702 |
| <code>mln::fun::x2v::bilinear&lt; I &gt;</code> (Represent a bilinear interolation of values from an underlying image) . . . . .   | 703 |
| <code>mln::fun::x2v::trilinear&lt; I &gt;</code> (Represent a trilinear interolation of values from an underlying image) . . . . .   | 704 |
| <code>mln::fun::x2x::composed&lt; T2, T1 &gt;</code> (Represent a composition of two transformations) . . . . .  | 704 |
| <code>mln::fun::x2x::linear&lt; I &gt;</code> (Represent a linear interolation of values from an underlying image) . . . . .   | 705 |
| <code>mln::fun::x2x::rotation&lt; n, C &gt;</code> (Represent a rotation function) . . . . .   | 706 |
| <code>mln::fun::x2x::translation&lt; n, C &gt;</code> (Translation function-object) . . . . .  | 710 |

|  |     |
|--|-----|
| <code>mln::fun_image&lt; F, I &gt;</code> ( <code>Image</code> read through a function ) . . . . .   | 712 |
| <code>mln::Function&lt; E &gt;</code> (Base class for implementation of function-objects ) . . . . .   | 715 |
| <code>mln::Function&lt; void &gt;</code> ( <code>Function</code> category flag type ) . . . . .  | 715 |
| <code>mln::Function_n2v&lt; E &gt;</code> (Base class for implementation of function-objects from Nil to value ) . . . . .   | 716 |
| <code>mln::Function_v2b&lt; E &gt;</code> (Base class for implementation of function-objects from a value to a Boolean ) . . . . .                                       | 716 |
| <code>mln::Function_v2v&lt; E &gt;</code> (Base class for implementation of function-objects from value to value ) . . . . .   | 717 |
| <code>mln::Function_vv2b&lt; E &gt;</code> (Base class for implementation of function-objects from a couple of values to a Boolean ) . . . . .                           | 718 |
| <code>mln::Function_vv2v&lt; E &gt;</code> (Base class for implementation of function-objects from a couple of values to a value ) . . . . .                             | 718 |
| <code>mln::fwd_pixter1d&lt; I &gt;</code> (Forward pixel iterator on a 1-D image with border ) . . . . .   | 719 |
| <code>mln::fwd_pixter2d&lt; I &gt;</code> (Forward pixel iterator on a 2-D image with border ) . . . . .   | 720 |
| <code>mln::fwd_pixter3d&lt; I &gt;</code> (Forward pixel iterator on a 3-D image with border ) . . . . .   | 722 |
| <code>mln::Gdpoint&lt; E &gt;</code> (FIXME: Doc! ) . . . . .  | 723 |
| <code>mln::Gdpoint&lt; void &gt;</code> (Delta point site category flag type ) . . . . .   | 724 |
| <code>mln::geom::complex_geometry&lt; D, P &gt;</code> (A functor returning the sites of the faces of a complex where the locations of each 0-face is stored ) . . . . . | 725 |
| <code>mln::Gpoint&lt; E &gt;</code> (Base class for implementation of point classes ) . . . . .  | 726 |
| <code>mln::Graph&lt; E &gt;</code> (Base class for implementation of graph classes ) . . . . .   | 731 |
| <code>mln::graph::attribute::card_t</code> (Compute the cardinality of every component in a graph ) . . . . .  | 732 |
| <code>mln::graph::attribute::representative_t</code> (Compute the representative vertex of every component in a graph ) . . . . .  | 732 |
| <code>mln::graph_elt_mixed_neighborhood&lt; G, S, S2 &gt;</code> (Elementary neighborhood on graph class ) . . . . .   | 733 |
| <code>mln::graph_elt_mixed_window&lt; G, S, S2 &gt;</code> (Elementary window on graph class ) . . . . .   | 734 |
| <code>mln::graph_elt_neighborhood&lt; G, S &gt;</code> (Elementary neighborhood on graph class ) . . . . .   | 738 |
| <code>mln::graph_elt_neighborhood_if&lt; G, S, I &gt;</code> (Elementary neighborhood_if on graph class ) . . . . .  | 740 |
| <code>mln::graph_elt_window&lt; G, S &gt;</code> (Elementary window on graph class ) . . . . .   | 742 |
| <code>mln::graph_elt_window_if&lt; G, S, I &gt;</code> (Custom window on graph class ) . . . . .   | 746 |
| <code>mln::graph_window_base&lt; P, E &gt;</code> . . . . .  | 751 |
| <code>mln::graph_window_if_piter&lt; S, W, I &gt;</code> (Forward iterator on line graph window ) . . . . .  | 753 |
| <code>mln::graph_window_piter&lt; S, W, I &gt;</code> (Forward iterator on line graph window ) . . . . .   | 755 |
| <code>mln::hexa&lt; I &gt;</code> (Hexagonal image class ) . . . . .   | 758 |
| <code>mln::histo::array&lt; T &gt;</code> (Generic histogram class over a value set with type T ) . . . . .  | 762 |
| <code>mln::Image&lt; E &gt;</code> (Base class for implementation of image classes ) . . . . .   | 762 |
| <code>mln::image1d&lt; T &gt;</code> (Basic 1D image class ) . . . . .   | 764 |
| <code>mln::image2d&lt; T &gt;</code> (Basic 2D image class ) . . . . .   | 769 |
| <code>mln::image2d_h&lt; V &gt;</code> (2d image based on an hexagonal mesh ) . . . . .  | 774 |
| <code>mln::image3d&lt; T &gt;</code> (Basic 3D image class ) . . . . .   | 777 |
| <code>mln::image_if&lt; I, F &gt;</code> ( <code>Image</code> which domain is restricted by a function 'site -> Boolean' ) . . . . .                                     | 782 |
| <code>mln::interpolated&lt; I, F &gt;</code> (Makes the underlying image being accessed with floating coordinates ) . . . . .  | 784 |
| <code>mln::io::dicom::dicom_header</code> (Store dicom file header ) . . . . .   | 786 |
| <code>mln::io::dump::dump_header</code> (Store dump file header ) . . . . .  | 786 |
| <code>mln::io::fld::fld_header</code> (Define the header structure of an AVS field data file ) . . . . .   | 787 |
| <code>mln::io::raw::raw_header</code> (Store raw file header ) . . . . .   | 787 |
| <code>mln::Iterator&lt; E &gt;</code> (Base class for implementation classes that are iterators ) . . . . .  | 787 |
| <code>mln::labeled_image&lt; I &gt;</code> (Morpher providing an improved interface for labeled image ) . . . . .  | 789 |
| <code>mln::labeled_image_base&lt; I, E &gt;</code> (Base class Morpher providing an improved interface for labeled image ) . . . . .                                     | 793 |
| <code>mln::lazy_image&lt; I, F, B &gt;</code> ( <code>Image</code> values are computed on the fly ) . . . . .  | 796 |
| <code>mln::Literal&lt; E &gt;</code> (Base class for implementation classes of literals ) . . . . .  | 799 |
| <code>mln::literal::black_t</code> (Type of literal black ) . . . . .  | 801 |
| <code>mln::literal::blue_t</code> (Type of literal blue ) . . . . .  | 802 |
| <code>mln::literal::brown_t</code> (Type of literal brown ) . . . . .  | 802 |

|  |     |
|--|-----|
| <code>mln::literal::cyan_t</code> (Type of literal cyan) . . . . .   | 803 |
| <code>mln::literal::green_t</code> (Type of literal green) . . . . .   | 804 |
| <code>mln::literal::identity_t</code> (Type of literal identity) . . . . .   | 805 |
| <code>mln::literal::light_gray_t</code> (Type of literal grays) . . . . .  | 806 |
| <code>mln::literal::lime_t</code> (Type of literal lime) . . . . .   | 807 |
| <code>mln::literal::magenta_t</code> (Type of literal magenta) . . . . .   | 808 |
| <code>mln::literal::max_t</code> (Type of literal max) . . . . .   | 809 |
| <code>mln::literal::min_t</code> (Type of literal min) . . . . .   | 810 |
| <code>mln::literal::olive_t</code> (Type of literal olive) . . . . .   | 811 |
| <code>mln::literal::one_t</code> (Type of literal one) . . . . .   | 812 |
| <code>mln::literal::orange_t</code> (Type of literal orange) . . . . .   | 813 |
| <code>mln::literal::origin_t</code> (Type of literal origin) . . . . .   | 814 |
| <code>mln::literal::pink_t</code> (Type of literal pink) . . . . .   | 815 |
| <code>mln::literal::purple_t</code> (Type of literal purple) . . . . .   | 816 |
| <code>mln::literal::red_t</code> (Type of literal red) . . . . .   | 817 |
| <code>mln::literal::teal_t</code> (Type of literal teal) . . . . .   | 818 |
| <code>mln::literal::violet_t</code> (Type of literal violet) . . . . .   | 819 |
| <code>mln::literal::white_t</code> (Type of literal white) . . . . .   | 820 |
| <code>mln::literal::yellow_t</code> (Type of literal yellow) . . . . .   | 821 |
| <code>mln::literal::zero_t</code> (Type of literal zero) . . . . .   | 822 |
| <code>mln::Mesh&lt; E &gt;</code> (Base class for implementation classes of meshes) . . . . .  | 823 |
| <code>mln::Meta_Accumulator&lt; E &gt;</code> (Base class for implementation of meta accumulators) . . . . .                             | 824 |
| <code>mln::Meta_Function&lt; E &gt;</code> (Base class for implementation of meta functions) . . . . .                                   | 826 |
| <code>mln::Meta_Function_v2v&lt; E &gt;</code> (Base class for implementation of function-objects from value to value) . . . . .         | 827 |
| <code>mln::Meta_Function_vv2v&lt; E &gt;</code> (Base class for implementation of function-objects from value to value) . . . . .        | 827 |
| <code>mln::metal::ands&lt; E1, E2, E3, E4, E5, E6, E7, E8 &gt;</code> (Ands type) . . . . .  | 828 |
| <code>mln::metal::converts_to&lt; T, U &gt;</code> ("converts-to" check) . . . . .   | 828 |
| <code>mln::metal::equal&lt; T1, T2 &gt;</code> (Definition of a static 'equal' test) . . . . .   | 828 |
| <code>mln::metal::goes_to&lt; T, U &gt;</code> ("goes-to" check) . . . . .   | 829 |
| <code>mln::metal::is&lt; T, U &gt;</code> ("is" check) . . . . .   | 829 |
| <code>mln::metal::is_a&lt; T, M &gt;</code> ("is_a" check) . . . . .   | 830 |
| <code>mln::metal::is_not&lt; T, U &gt;</code> ("is_not" check) . . . . .   | 830 |
| <code>mln::metal::is_not_a&lt; T, M &gt;</code> ("is_not_a" static Boolean expression) . . . . .   | 830 |
| <code>mln::mixed_neighb&lt; W &gt;</code> (Adapter class from window to neighborhood) . . . . .  | 830 |
| <code>mln::morpho::attribute::card&lt; I &gt;</code> (Cardinality accumulator class) . . . . .   | 832 |
| <code>mln::morpho::attribute::count_adjacent_vertices&lt; I &gt;</code> (Count_Adjacent_Vertices accumulator class) . . . . .            | 833 |
| <code>mln::morpho::attribute::height&lt; I &gt;</code> (Height accumulator class) . . . . .  | 835 |
| <code>mln::morpho::attribute::sharpness&lt; I &gt;</code> (Sharpness accumulator class) . . . . .  | 836 |
| <code>mln::morpho::attribute::sum&lt; I, S &gt;</code> (Suminality accumulator class) . . . . .  | 839 |
| <code>mln::morpho::attribute::volume&lt; I &gt;</code> (Volume accumulator class) . . . . .  | 840 |
| <code>mln::neighb&lt; W &gt;</code> (Adapter class from window to neighborhood) . . . . .  | 842 |
| <code>mln::Neighborhood&lt; E &gt;</code> (Base class for implementation classes that are neighborhoods) . . . . .                       | 844 |
| <code>mln::Neighborhood&lt; void &gt;</code> (Neighborhood category flag type) . . . . .   | 844 |
| <code>mln::Object&lt; E &gt;</code> (Base class for almost every class defined in Milena) . . . . .                                      | 844 |
| <code>mln::p2p_image&lt; I, F &gt;</code> (FIXME: Doc!) . . . . .  | 845 |
| <code>mln::p_array&lt; P &gt;</code> (Multi-set of sites) . . . . .  | 847 |
| <code>mln::p_centered&lt; W &gt;</code> (Site set corresponding to a window centered on a site) . . . . .                                | 852 |
| <code>mln::p_complex&lt; D, G &gt;</code> (A complex psite set based on the N-faces of a complex of dimension D (a D-complex)) . . . . . | 855 |
| <code>mln::p_edges&lt; G, F &gt;</code> (Site set mapping graph edges and image sites) . . . . .   | 859 |

|   |     |
|---|-----|
| <code>mln::p_faces&lt; N, D, P &gt;</code> (A complex psite set based on a the N-faces of a complex of dimension D (a D-complex) ) . . . . .          | 865 |
| <code>mln::p_graph_piter&lt; S, I &gt;</code> (Generic iterator on point sites of a <code>mln::S</code> ) . . . . .                                   | 868 |
| <code>mln::p_if&lt; S, F &gt;</code> (Site set restricted w.r.t ) . . . . .   | 870 |
| <code>mln::p_image&lt; I &gt;</code> (Site set based on an image of Booleans ) . . . . .  | 873 |
| <code>mln::p_indexed_bkd_piter&lt; S &gt;</code> (Backward iterator on sites of an indexed site set ) . . . . .                                       | 877 |
| <code>mln::p_indexed_fwd_piter&lt; S &gt;</code> (Forward iterator on sites of an indexed site set ) . . . . .  | 879 |
| <code>mln::p_indexed_psite&lt; S &gt;</code> (Psite class for indexed site sets such as <code>p_array</code> ) . . . . .                              | 880 |
| <code>mln::p_key&lt; K, P &gt;</code> (Priority queue class ) . . . . .   | 880 |
| <code>mln::p_line2d</code> (2D discrete line of points ) . . . . .  | 886 |
| <code>mln::p_mutable_array_of&lt; S &gt;</code> ( <code>P_mutable_array_of</code> is a mutable array of site sets ) . . . . .                         | 890 |
| <code>mln::p_n_faces_bkd_piter&lt; D, G &gt;</code> (Backward iterator on the n-faces sites of an <code>mln::p-complex&lt;D, G&gt;</code> ) . . . . . | 893 |
| <code>mln::p_n_faces_fwd_piter&lt; D, G &gt;</code> (Forward iterator on the n-faces sites of an <code>mln::p-complex&lt;D, G&gt;</code> ) . . . . .  | 895 |
| <code>mln::p_priority&lt; P, Q &gt;</code> (Priority queue ) . . . . .  | 896 |
| <code>mln::p_queue&lt; P &gt;</code> (Queue of sites (based on <code>std::deque</code> ) ) . . . . .  | 902 |
| <code>mln::p_queue_fast&lt; P &gt;</code> (Queue of sites class (based on <code>p_array</code> ) ) . . . . .  | 906 |
| <code>mln::p_run&lt; P &gt;</code> (Point set class in run ) . . . . .  | 912 |
| <code>mln::p_set&lt; P &gt;</code> (Mathematical set of sites (based on <code>util::set</code> ) ) . . . . .  | 916 |
| <code>mln::p_set_of&lt; S &gt;</code> ( <code>P_set_of</code> is a set of site sets ) . . . . .   | 921 |
| <code>mln::p_transformed&lt; S, F &gt;</code> (Site set transformed through a function ) . . . . .  | 924 |
| <code>mln::p_transformed_piter&lt; Pi, S, F &gt;</code> (Iterator on <code>p_transformed&lt;S,F&gt;</code> ) . . . . .                                | 927 |
| <code>mln::p_vaccess&lt; V, S &gt;</code> (Site set in which sites are grouped by their associated value ) . . . . .                                  | 928 |
| <code>mln::p_vertices&lt; G, F &gt;</code> (Site set based mapping graph vertices to sites ) . . . . .  | 932 |
| <code>mln::pixel&lt; I &gt;</code> (Generic pixel class ) . . . . .   | 939 |
| <code>mln::Pixel_Iterator&lt; E &gt;</code> (Base class for the implementation of pixel iterator classes ) . . . . .                                  | 940 |
| <code>mln::plain&lt; I &gt;</code> (Prevents an image from sharing its data ) . . . . .   | 942 |
| <code>mln::Point&lt; P &gt;</code> (Base class for implementation of point classes ) . . . . .  | 943 |
| <code>mln::point&lt; G, C &gt;</code> (Generic point class ) . . . . .  | 946 |
| <code>mln::Point_Site&lt; E &gt;</code> (Base class for implementation classes of the notion of "point site" ) . . . . .                              | 953 |
| <code>mln::Point_Site&lt; void &gt;</code> (Point site category flag type ) . . . . .   | 956 |
| <code>mln::Proxy&lt; E &gt;</code> (Base class for implementation classes of the notion of "proxy" ) . . . . .  | 957 |
| <code>mln::Proxy&lt; void &gt;</code> (Proxy category flag type ) . . . . .   | 957 |
| <code>mln::Pseudo_Site&lt; E &gt;</code> (Base class for implementation classes of the notion of "pseudo site" ) . . . . .                            | 957 |
| <code>mln::Pseudo_Site&lt; void &gt;</code> (Pseudo_Site category flag type ) . . . . .   | 958 |
| <code>mln::pw::image&lt; F, S &gt;</code> (A generic point-wise image implementation ) . . . . .  | 958 |
| <code>mln::registration::closest_point_basic&lt; P &gt;</code> (Closest point functor based on map distance ) . . . . .                               | 959 |
| <code>mln::registration::closest_point_with_map&lt; P &gt;</code> (Closest point functor based on map distance ) . . . . .                            | 960 |
| <code>mln::Regular_Grid&lt; E &gt;</code> (Base class for implementation classes of regular grids ) . . . . .   | 960 |
| <code>mln::safe_image&lt; I &gt;</code> (Makes an image accessible at undefined location ) . . . . .  | 961 |
| <code>mln::select::p_of&lt; P &gt;</code> (Structure <code>p_of</code> ) . . . . .  | 961 |
| <code>mln::Site&lt; E &gt;</code> (Base class for classes that are explicitly sites ) . . . . .   | 962 |
| <code>mln::Site&lt; void &gt;</code> (Site category flag type ) . . . . .   | 963 |
| <code>mln::Site_Iterator&lt; E &gt;</code> (Base class for implementation of classes of iterator on points ) . . . . .                                | 963 |
| <code>mln::Site_Proxy&lt; E &gt;</code> (Base class for implementation classes of the notion of "site proxy" ) . . . . .                              | 965 |
| <code>mln::Site_Proxy&lt; void &gt;</code> (Site_Proxy category flag type ) . . . . .   | 966 |
| <code>mln::Site_Set&lt; E &gt;</code> (Base class for implementation classes of site sets ) . . . . .   | 966 |
| <code>mln::Site_Set&lt; void &gt;</code> (Site_Set category flag type ) . . . . .   | 969 |
| <code>mln::slice_image&lt; I &gt;</code> (2D image extracted from a slice of a 3D image ) . . . . .   | 969 |
| <code>mln::sub_image&lt; I, S &gt;</code> (Image having its domain restricted by a site set ) . . . . .   | 972 |
| <code>mln::sub_image_if&lt; I, S &gt;</code> (Image having its domain restricted by a site set and a function ) . . . . .                             | 973 |
| <code>mln::thru_image&lt; I, F &gt;</code> (Morph image values through a function ) . . . . .   | 975 |
| <code>mln::thrubin_image&lt; I1, I2, F &gt;</code> (Morphes values from two images through a binary function ) . . . . .                              | 975 |

|   |      |
|---|------|
| <code>mln::topo::adj_higher_dim_connected_n_face_bkd_iter&lt; D &gt;</code> (Backward iterator on all the n-faces sharing an adjacent (n+1)-face with a (reference) n-face of an <code>mln::complex&lt;D&gt;</code> ) . . . . . | 977  |
| <code>mln::topo::adj_higher_dim_connected_n_face_fwd_iter&lt; D &gt;</code> (Forward iterator on all the n-faces sharing an adjacent (n+1)-face with a (reference) n-face of an <code>mln::complex&lt;D&gt;</code> ) . . . . .  | 978  |
| <code>mln::topo::adj_higher_face_bkd_iter&lt; D &gt;</code> (Backward iterator on all the adjacent (n+1)-faces of the n-face of an <code>mln::complex&lt;D&gt;</code> ) . . . . .   | 979  |
| <code>mln::topo::adj_higher_face_fwd_iter&lt; D &gt;</code> (Forward iterator on all the adjacent (n+1)-faces of the n-face of an <code>mln::complex&lt;D&gt;</code> ) . . . . .  | 980  |
| <code>mln::topo::adj_lower_dim_connected_n_face_bkd_iter&lt; D &gt;</code> (Backward iterator on all the n-faces sharing an adjacent (n-1)-face with a (reference) n-face of an <code>mln::complex&lt;D&gt;</code> ) . . . . .  | 981  |
| <code>mln::topo::adj_lower_dim_connected_n_face_fwd_iter&lt; D &gt;</code> (Forward iterator on all the n-faces sharing an adjacent (n-1)-face with a (reference) n-face of an <code>mln::complex&lt;D&gt;</code> ) . . . . .   | 982  |
| <code>mln::topo::adj_lower_face_bkd_iter&lt; D &gt;</code> (Backward iterator on all the adjacent (n-1)-faces of the n-face of an <code>mln::complex&lt;D&gt;</code> ) . . . . .  | 984  |
| <code>mln::topo::adj_lower_face_fwd_iter&lt; D &gt;</code> (Forward iterator on all the adjacent (n-1)-faces of the n-face of an <code>mln::complex&lt;D&gt;</code> ) . . . . .   | 985  |
| <code>mln::topo::adj_lower_higher_face_bkd_iter&lt; D &gt;</code> (Forward iterator on all the adjacent (n-1)-faces and (n+1)-faces of the n-face of an <code>mln::complex&lt;D&gt;</code> ) . . . . .                          | 986  |
| <code>mln::topo::adj_lower_higher_face_fwd_iter&lt; D &gt;</code> (Forward iterator on all the adjacent (n-1)-faces and (n+1)-faces of the n-face of an <code>mln::complex&lt;D&gt;</code> ) . . . . .                          | 987  |
| <code>mln::topo::adj_m_face_bkd_iter&lt; D &gt;</code> (Backward iterator on all the m-faces transitively adjacent to a (reference) n-face in a complex) . . . . .  | 988  |
| <code>mln::topo::adj_m_face_fwd_iter&lt; D &gt;</code> (Forward iterator on all the m-faces transitively adjacent to a (reference) n-face in a complex) . . . . .   | 989  |
| <code>mln::topo::algebraic_face&lt; D &gt;</code> (Algebraic face handle in a complex; the face dimension is dynamic) . . . . .   | 991  |
| <code>mln::topo::algebraic_n_face&lt; N, D &gt;</code> (Algebraic N-face handle in a complex) . . . . .   | 996  |
| <code>mln::topo::center_only_iter&lt; D &gt;</code> (Iterator on all the adjacent (n-1)-faces of the n-face of an <code>mln::complex&lt;D&gt;</code> ) . . . . .  | 1001 |
| <code>mln::topo::centered_bkd_iter_adapter&lt; D, I &gt;</code> (Forward complex relative iterator adapters adding the central (reference) point to the set of iterated faces) . . . . .  | 1002 |
| <code>mln::topo::centered_fwd_iter_adapter&lt; D, I &gt;</code> (Backward complex relative iterator adapters adding the central (reference) point to the set of iterated faces) . . . . .                                       | 1003 |
| <code>mln::topo::complex&lt; D &gt;</code> (General complex of dimension D) . . . . .   | 1004 |
| <code>mln::topo::face&lt; D &gt;</code> (Face handle in a complex; the face dimension is dynamic) . . . . .   | 1007 |
| <code>mln::topo::face_bkd_iter&lt; D &gt;</code> (Backward iterator on all the faces of an <code>mln::complex&lt;D&gt;</code> ) . . . . .   | 1012 |
| <code>mln::topo::face_fwd_iter&lt; D &gt;</code> (Forward iterator on all the faces of an <code>mln::complex&lt;D&gt;</code> ) . . . . .  | 1013 |
| <code>mln::topo::is_n_face&lt; N &gt;</code> (A functor testing wheter a <code>mln::complex_psite</code> is an N-face) . . . . .  | 1014 |
| <code>mln::topo::is_simple_2d_t&lt; N &gt;</code> (Test if a point is simple or not) . . . . .  | 1015 |
| <code>mln::topo::is_simple_cell&lt; I &gt;</code> (A predicate for the simplicity of a point based on the collapse property of the attachment) . . . . .  | 1016 |
| <code>mln::topo::n_face&lt; N, D &gt;</code> (N-face handle in a complex) . . . . .   | 1019 |
| <code>mln::topo::n_face_bkd_iter&lt; D &gt;</code> (Backward iterator on all the faces of an <code>mln::complex&lt;D&gt;</code> ) . . . . .   | 1023 |
| <code>mln::topo::n_face_fwd_iter&lt; D &gt;</code> (Forward iterator on all the faces of an <code>mln::complex&lt;D&gt;</code> ) . . . . .  | 1024 |
| <code>mln::topo::n_faces_set&lt; N, D &gt;</code> (Set of face handles of dimension N) . . . . .  | 1026 |
| <code>mln::topo::skeleton::is_simple_point&lt; N &gt;</code> . . . . .  | 1027 |
| <code>mln::topo::static_n_face_bkd_iter&lt; N, D &gt;</code> (Backward iterator on all the N-faces of a <code>mln::complex&lt;D&gt;</code> ) . . . . .  | 1028 |
| <code>mln::topo::static_n_face_fwd_iter&lt; N, D &gt;</code> (Forward iterator on all the N-faces of a <code>mln::complex&lt;D&gt;</code> ) . . . . .   | 1029 |

|   |      |
|---|------|
| <code>mln::tr_image&lt; S, I, T &gt;</code> (Transform an image by a given transformation ) . . . . .   | 1030 |
| <code>mln::unproject_image&lt; I, D, F &gt;</code> (Un-projects an image ) . . . . .  | 1033 |
| <code>mln::util::adjacency_matrix&lt; V &gt;</code> (A class of adjacency matrix ) . . . . .  | 1035 |
| <code>mln::util::array&lt; T &gt;</code> (A dynamic array class ) . . . . .   | 1036 |
| <code>mln::util::branch&lt; T &gt;</code> (Class of generic branch ) . . . . .  | 1043 |
| <code>mln::util::branch_iter&lt; T &gt;</code> (Basic 2D image class ) . . . . .  | 1044 |
| <code>mln::util::branch_iter_ind&lt; T &gt;</code> (Basic 2D image class ) . . . . .  | 1046 |
| <code>mln::util::couple&lt; T, U &gt;</code> (Definition of a couple ) . . . . .  | 1047 |
| <code>mln::util::eat</code> (Eat structure ) . . . . .  | 1049 |
| <code>mln::util::edge&lt; G &gt;</code> (Edge of a graph $G$ ) . . . . .  | 1050 |
| <code>mln::util::fibonacci_heap&lt; P, T &gt;</code> (Fibonacci heap ) . . . . .  | 1054 |
| <code>mln::util::graph</code> (Undirected graph ) . . . . .   | 1058 |
| <code>mln::util::greater_point&lt; I &gt;</code> (A "greater than" functor comparing points w.r.t ) . . . . .   | 1065 |
| <code>mln::util::greater_psite&lt; I &gt;</code> (A "greater than" functor comparing psites w.r.t ) . . . . .   | 1066 |
| <code>mln::util::head&lt; T, R &gt;</code> (Top structure of the soft heap ) . . . . .  | 1066 |
| <code>mln::util::ignore</code> (Ignore structure ) . . . . .  | 1067 |
| <code>mln::util::ilcell&lt; T &gt;</code> (Element of an item list. Store the data (key) used in <code>soft_heap</code> ) . . . . .                     | 1067 |
| <code>mln::util::line_graph&lt; G &gt;</code> (Undirected line graph of a graph of type $G$ ) . . . . .   | 1068 |
| <code>mln::util::nil</code> (Nil structure ) . . . . .  | 1074 |
| <code>mln::util::node&lt; T, R &gt;</code> (Meta-data of an element in the heap ) . . . . .   | 1075 |
| <code>mln::util::object_id&lt; Tag, V &gt;</code> (Base class of an object id ) . . . . .   | 1075 |
| <code>mln::util::ord&lt; T &gt;</code> (Function-object that defines an ordering between objects with type $T$ : <i>lhs</i><br><i>R rhs</i> ) . . . . . | 1077 |
| <code>mln::util::ord_pair&lt; T &gt;</code> (Ordered pair structure s.a ) . . . . .   | 1077 |
| <code>mln::util::pix&lt; I &gt;</code> (Structure pix ) . . . . .   | 1080 |
| <code>mln::util::set&lt; T &gt;</code> (An "efficient" mathematical set class ) . . . . .   | 1081 |
| <code>mln::util::site_pair&lt; P &gt;</code> (A pair of sites ) . . . . .   | 1087 |
| <code>mln::util::soft_heap&lt; T, R &gt;</code> (Soft heap ) . . . . .  | 1089 |
| <code>mln::util::timer</code> (Timer structure ) . . . . .  | 1092 |
| <code>mln::util::tracked_ptr&lt; T &gt;</code> (Smart pointer for shared data with tracking ) . . . . .   | 1093 |
| <code>mln::util::tree&lt; T &gt;</code> (Class of generic tree ) . . . . .  | 1095 |
| <code>mln::util::tree_node&lt; T &gt;</code> (Class of generic <code>tree_node</code> for tree ) . . . . .  | 1097 |
| <code>mln::util::vertex&lt; G &gt;</code> (Vertex of a graph $G$ ) . . . . .  | 1102 |
| <code>mln::util::yes</code> (Object that always says "yes" ) . . . . .  | 1107 |
| <code>mln::Value&lt; E &gt;</code> (Base class for implementation classes of values ) . . . . .   | 1107 |
| <code>mln::value::float01</code> (Class for floating values restricted to the interval $[0..1]$ and discretized with<br>$n$ bits ) . . . . .            | 1109 |
| <code>mln::value::float01_f</code> (Class for floating values restricted to the interval $[0..1]$ ) . . . . .   | 1111 |
| <code>mln::value::graylevel&lt; n &gt;</code> (General gray-level class on $n$ bits ) . . . . .   | 1112 |
| <code>mln::value::graylevel_f</code> (General gray-level class on $n$ bits ) . . . . .  | 1116 |
| <code>mln::value::int_s&lt; n &gt;</code> (Signed integer value class ) . . . . .   | 1118 |
| <code>mln::value::int_u&lt; n &gt;</code> (Unsigned integer value class ) . . . . .   | 1121 |
| <code>mln::value::int_u_sat&lt; n &gt;</code> (Unsigned integer value class with saturation behavior ) . . . . .  | 1123 |
| <code>mln::value::Integer&lt; E &gt;</code> (Concept of integer ) . . . . .   | 1125 |
| <code>mln::value::Integer&lt; void &gt;</code> (Category flag type ) . . . . .  | 1126 |
| <code>mln::value::label&lt; n &gt;</code> (Label value class ) . . . . .  | 1126 |
| <code>mln::value::lut_vec&lt; S, T &gt;</code> (Class that defines FIXME ) . . . . .  | 1129 |
| <code>mln::value::proxy&lt; I &gt;</code> (Generic proxy class for an image pixel value ) . . . . .   | 1133 |
| <code>mln::value::qt::rgb32</code> (Color class for red-green-blue where every component is $n$ -bit encoded ) . . . . .                                | 1135 |
| <code>mln::value::rgb&lt; n &gt;</code> (Color class for red-green-blue where every component is $n$ -bit encoded ) . . . . .                           | 1137 |
| <code>mln::value::set&lt; T &gt;</code> (Class that defines the set of values of type $T$ ) . . . . .   | 1139 |
| <code>mln::value::sign</code> (Value type composed by the set $\{-1, 0, 1\}$ sign value type is a subset of the int<br>value type ) . . . . .           | 1140 |
| <code>mln::value::stack_image&lt; n, I &gt;</code> (Stack image class ) . . . . .   | 1142 |

|  |      |
|--|------|
| <code>mln::value::super_value&lt; sign &gt;</code> (Specializations: ) . . . . .   | 1145 |
| <code>mln::value::value_array&lt; T, V &gt;</code> (Generic array class over indexed by a value set with type T) . . . . .             | 1145 |
| <code>mln::Value_Iterator&lt; E &gt;</code> (Base class for implementation of classes of iterator on values) . . . . .                 | 1146 |
| <code>mln::Value_Set&lt; E &gt;</code> (Base class for implementation classes of sets of values) . . . . .                             | 1148 |
| <code>mln::Vertex&lt; E &gt;</code> ( <code>Vertex</code> category flag type) . . . . .  | 1149 |
| <code>mln::vertex_image&lt; P, V, G &gt;</code> ( <code>Image</code> based on graph vertices) . . . . .                                | 1150 |
| <code>mln::violent_cast_image&lt; T, I &gt;</code> (Violently cast image values to a given type) . . . . .                             | 1152 |
| <code>mln::w_window&lt; D, W &gt;</code> (Generic <code>w_window</code> class) . . . . .   | 1154 |
| <code>mln::Weighted_Window&lt; E &gt;</code> (Base class for implementation classes that are weighted-windows) . . . . .               | 1158 |
| <code>mln::win::backdiag2d</code> (Diagonal line window defined on the 2D square grid) . . . . .                                       | 1159 |
| <code>mln::win::ball&lt; G, C &gt;</code> (Generic ball window defined on a given grid) . . . . .                                      | 1160 |
| <code>mln::win::cube3d</code> (Cube window defined on the 3D grid) . . . . .   | 1161 |
| <code>mln::win::cuboid3d</code> (Cuboid defined on the 3-D square grid) . . . . .  | 1162 |
| <code>mln::win::diag2d</code> (Diagonal line window defined on the 2D square grid) . . . . .   | 1164 |
| <code>mln::win::line&lt; M, i, C &gt;</code> (Generic line window defined on a given grid in the given dimension) . . . . .            | 1165 |
| <code>mln::win::multiple&lt; W, F &gt;</code> (Multiple window) . . . . .  | 1167 |
| <code>mln::win::multiple_size&lt; n, W, F &gt;</code> (Definition of a multiple-size window) . . . . .                                 | 1167 |
| <code>mln::win::octagon2d</code> (Octagon window defined on the 2D square grid) . . . . .  | 1168 |
| <code>mln::win::rectangle2d</code> (Rectangular window defined on the 2D square grid) . . . . .  | 1169 |
| <code>mln::Window&lt; E &gt;</code> (Base class for implementation classes that are windows) . . . . .                                 | 1170 |
| <code>mln::window&lt; D &gt;</code> (Generic window class) . . . . .   | 1171 |
| <code>mln::world::inter_pixel::is_separator</code> (Functor returning whether a site is a separator in an inter-pixel image) . . . . . | 1176 |
| <code>trait::graph&lt; I &gt;</code> (Graph traits) . . . . .  | 1177 |
| <code>trait::graph&lt; mln::complex_image&lt; 1, G, V &gt;&gt;</code> (Graph traits for 1-complexes images) . . . . .                  | 1178 |
| <code>trait::graph&lt; mln::image2d&lt; T &gt;&gt;</code> (Graph traits for <code>mln::image2d</code> ) . . . . .                      | 1178 |



# Chapter 8

## Module Documentation

### 8.1 On site sets

Accumulators working on site sets.

#### Classes

- struct `mln::accu::center< P, V >`  
*Mass center accumulator.*
- struct `mln::accu::math::count< T >`  
*Generic counter accumulator.*
- struct `mln::accu::shape::bbox< P >`  
*Generic bounding box accumulator class.*
- class `mln::accu::site_set::rectangularity< P >`  
*Compute the rectangularity of a site set.*

#### 8.1.1 Detailed Description

Accumulators working on site sets.

### 8.2 On images

Accumulators working on images.

#### Classes

- struct `mln::accu::count_adjacent_vertices< F, S >`  
*Accumulator class counting the number of vertices adjacent to a set of `mln::p_edges_psite` (i.e., a set of edges).*

- struct `mln::accu::max_site< I >`  
*Define an accumulator that computes the first site with the maximum value in an image.*
- struct `mln::accu::shape::height< I >`  
*Height accumulator.*
- struct `mln::accu::shape::volume< I >`  
*Volume accumulator class.*

### 8.2.1 Detailed Description

Accumulators working on images.

## 8.3 On values

Accumulators working on image values.

### Classes

- struct `mln::accu::convolve< T1, T2, R >`  
*Generic convolution accumulator class.*
- struct `mln::accu::count_labels< L >`  
*Count the number of different labels in an image.*
- struct `mln::accu::count_value< V >`  
*Define an accumulator that counts the occurrence of a given value.*
- struct `mln::accu::histo< V >`  
*Generic histogram class over a value set with type V.*
- struct `mln::accu::label_used< L >`  
*References all the labels used.*
- struct `mln::accu::logic::land`  
*"Logical-and" accumulator.*
- struct `mln::accu::logic::land_basic`  
*"Logical-and" accumulator.*
- struct `mln::accu::logic::lor`  
*"Logical-or" accumulator.*
- struct `mln::accu::logic::lor_basic`  
*"Logical-or" accumulator class.*

- struct `mln::accu::maj_h< T >`  
*Compute the majority value.*
- struct `mln::accu::math::inf< T >`  
*Generic inf accumulator class.*
- struct `mln::accu::math::sum< T, S >`  
*Generic sum accumulator class.*
- struct `mln::accu::math::sup< T >`  
*Generic sup accumulator class.*
- struct `mln::accu::rms< T, V >`  
*Generic root mean square accumulator class.*
- struct `mln::accu::stat::deviation< T, S, M >`  
*Generic standard deviation accumulator class.*
- struct `mln::accu::stat::histo3d_rgb< V >`  
*Define a histogram as accumulator which returns an `image3d`.*
- struct `mln::accu::stat::max< T >`  
*Generic max accumulator class.*
- struct `mln::accu::stat::max_h< V >`  
*Generic max function based on histogram over a value set with type `V`.*
- struct `mln::accu::stat::mean< T, S, M >`  
*Generic mean accumulator class.*
- struct `mln::accu::stat::median_alt< S >`  
*Generic `median_alt` function based on histogram over a value set with type `S`.*
- struct `mln::accu::stat::median_h< V >`  
*Generic median function based on histogram over a value set with type `V`.*
- struct `mln::accu::stat::min< T >`  
*Generic min accumulator class.*
- struct `mln::accu::stat::min_h< V >`  
*Generic min function based on histogram over a value set with type `V`.*
- struct `mln::accu::stat::min_max< V >`  
*Generic min and max accumulator class.*
- struct `mln::accu::stat::rank< T >`  
*Generic rank accumulator class.*
- struct `mln::accu::stat::rank< bool >`  
*rank accumulator class for Boolean.*

- struct `mln::accu::stat::rank_high_quant< T >`  
*Generic rank accumulator class.*
- struct `mln::accu::stat::var< T >`  
*Var accumulator class.*
- struct `mln::accu::stat::variance< T, S, R >`  
*Variance accumulator class.*

### 8.3.1 Detailed Description

Accumulators working on image values.

## 8.4 Multiple accumulators

Set of special accumulators for computing several accumulators at the same time.

### Classes

- struct `mln::accu::pair< A1, A2, T >`  
*Generic pair of accumulators.*
- struct `mln::accu::tuple< A, n, >`  
*Generic tuple of accumulators.*

### 8.4.1 Detailed Description

Set of special accumulators for computing several accumulators at the same time.

## 8.5 Graphes

All graphes implementations.

### Classes

- class `mln::util::graph`  
*Undirected graph.*
- class `mln::util::line_graph< G >`  
*Undirected line graph of a graph of type G.*

### 8.5.1 Detailed Description

All graphes implementations.

## 8.6 Images

All the generic image types provided in Olena.

### Modules

- [Basic types](#)  
*Concrete images.*
- [Image morphers](#)  
*Morpher on both image values and domain.*
- [Values morphers](#)  
*Morpher on image values.*
- [Domain morphers](#)  
*Morpher on image domain.*
- [Identity morphers](#)  
*Morpher adding new fonctionnalities.*

### 8.6.1 Detailed Description

All the generic image types provided in Olena.

## 8.7 Basic types

Concrete images.

### Classes

- class [mln::complex\\_image< D, G, V >](#)  
*Image based on a complex.*
- class [mln::edge\\_image< P, V, G >](#)  
*Image based on graph edges.*
- struct [mln::flat\\_image< T, S >](#)  
*Image with a single value.*
- struct [mln::imageId< T >](#)

*Basic 1D image class.*

- class `mln::image2d< T >`  
*Basic 2D image class.*
- struct `mln::image2d_h< V >`  
*2d image based on an hexagonal mesh.*
- struct `mln::image3d< T >`  
*Basic 3D image class.*
- class `mln::pw::image< F, S >`  
*A generic point-wise image implementation.*
- class `mln::vertex_image< P, V, G >`  
*Image based on graph vertices.*

### 8.7.1 Detailed Description

Concrete images.

## 8.8 Image morphers

Morpher on both image values and domain.

Morpher on both image values and domain.

## 8.9 Values morphers

Morpher on image values.

### Classes

- struct `mln::fun_image< F, I >`  
*Image read through a function.*
- class `mln::thru_image< I, F >`  
*Morph image values through a function.*
- class `mln::thrubin_image< I1, I2, F >`  
*Morphes values from two images through a binary function.*
- struct `mln::violent_cast_image< T, I >`  
*Violently cast image values to a given type.*

### 8.9.1 Detailed Description

Morpher on image values.

## 8.10 Domain morphers

Morpher on image domain.

### Classes

- struct `mln::extended< I >`  
*Makes an image become restricted by a point set.*
- class `mln::extension_fun< I, F >`  
*Extends the domain of an image with a function.*
- class `mln::extension_ima< I, J >`  
*Extends the domain of an image with an image.*
- class `mln::extension_val< I >`  
*Extends the domain of an image with a value.*
- struct `mln::hexa< I >`  
*hexagonal image class.*
- struct `mln::image_if< I, F >`  
*Image which domain is restricted by a function 'site -> Boolean'.*
- struct `mln::p2p_image< I, F >`  
*FIXME: Doc!*
- struct `mln::slice_image< I >`  
*2D image extracted from a slice of a 3D image.*
- class `mln::sub_image< I, S >`  
*Image having its domain restricted by a site set.*
- struct `mln::sub_image_if< I, S >`  
*Image having its domain restricted by a site set and a function.*
- struct `mln::unproject_image< I, D, F >`  
*Un-projects an image.*

### 8.10.1 Detailed Description

Morpher on image domain.

## 8.11 Identity morphers

Morpher adding new fonctionnalités.

### Classes

- struct `mln::decorated_image< I, D >`  
*Image that can have additional features.*
- class `mln::labeled_image< I >`  
*Morpher providing an improved interface for labeled image.*
- struct `mln::lazy_image< I, F, B >`  
*Image values are computed on the fly.*
- class `mln::plain< I >`  
*Prevents an image from sharing its data.*
- class `mln::safe_image< I >`  
*Makes an image accessible at undefined location.*
- struct `mln::tr_image< S, I, T >`  
*Transform an image by a given transformation.*

### 8.11.1 Detailed Description

Morpher adding new fonctionnalités.

## 8.12 Types

Milena Object types.

### Modules

- [Graphes](#)  
*All graphes implementations.*
- [Images](#)  
*All the generic image types provided in Olena.*
- [Neighborhoods](#)  
*All the predefined generic neighborhoods.*
- [Site sets](#)  
*All Site set types.*



- [Utilities](#)

*Miscellaneous useful containers/structures.*

- [Windows](#)

*All the predefined generic windows.*

### 8.12.1 Detailed Description

Milena Object types.

## 8.13 Accumulators

All accumulator types.

### Modules

- [On site sets](#)

*Accumulators working on site sets.*

- [On images](#)

*Accumulators working on images.*

- [On values](#)

*Accumulators working on image values.*

- [Multiple accumulators](#)

*Set of special accumulators for computing several accumulators at the same time.*

### 8.13.1 Detailed Description

All accumulator types.

## 8.14 Routines

All algorithms/routines provided in Milena.

All algorithms/routines provided in Milena.

## 8.15 Canvas

All canvas.

All canvas.

## 8.16 Functions

All predefined functions.

### Classes

- struct [mln::Function< E >](#)  
*Base class for implementation of function-objects.*
- struct [mln::Function\\_n2v< E >](#)  
*Base class for implementation of function-objects from Nil to value.*
- struct [mln::Function\\_v2b< E >](#)  
*Base class for implementation of function-objects from a value to a Boolean.*
- struct [mln::Function\\_v2v< E >](#)  
*Base class for implementation of function-objects from value to value.*
- struct [mln::Function\\_vv2b< E >](#)  
*Base class for implementation of function-objects from a couple of values to a Boolean.*
- struct [mln::Function\\_vv2v< E >](#)  
*Base class for implementation of function-objects from a couple of values to a value.*

### Namespaces

- namespace [mln::fun::i2v](#)  
*Namespace of integer-to-value functions.*
- namespace [mln::fun::n2v](#)  
*Namespace of functions from nil to value.*
- namespace [mln::fun::stat](#)  
*Namespace of statistical functions.*
- namespace [mln::fun::v2i](#)  
*Namespace of value-to-integer functions.*
- namespace [mln::fun::v2v](#)  
*Namespace of functions from value to value.*

### Modules

- [v2w2v functions](#)  
*All bijective functions.*

- [v2w\\_w2v functions](#)  
*All bijective function.*
- [vv2b functions](#)  
*All functions mapping two values to a logical value.*

### 8.16.1 Detailed Description

All predefined functions.

## 8.17 Neighborhoods

All the predefined generic neighborhoods.

### Modules

- [1D neighborhoods](#)  
*Predefined 1D neighborhoods.*
- [2D neighborhoods](#)  
*Predefined 2D neighborhoods.*
- [3D neighborhoods](#)  
*Predefined 3D neighborhoods.*

### 8.17.1 Detailed Description

All the predefined generic neighborhoods.

## 8.18 1D neighborhoods

Predefined 1D neighborhoods.

### Typedefs

- `typedef neighb< window1d > mln::neighb1d`  
*Type alias for a neighborhood defined on the 1D square grid with integer coordinates.*

### Functions

- `const neighb1d & mln::c2 ()`  
*2-connectivity neighborhood on the 1D grid.*

### 8.18.1 Detailed Description

Predefined 1D neighborhoods.

### 8.18.2 Typedef Documentation

#### 8.18.2.1 typedef neighb<window1d> mln::neighb1d

Type alias for a neighborhood defined on the 1D square grid with integer coordinates.

Definition at line 47 of file neighb1d.hh.

### 8.18.3 Function Documentation

#### 8.18.3.1 const neighb1d & mln::c2 ( ) [inline]

2-connectivity neighborhood on the 1D grid.

○ × ○

#### Returns

A neighb1d.

Definition at line 67 of file neighb1d.hh.

## 8.19 2D neighborhoods

Predefined 2D neighborhoods.

### Typedefs

- typedef neighb< window2d > [mln::neighb2d](#)  
*Type alias for a neighborhood defined on the 2D square grid with integer coordinates.*

### Functions

- const neighb2d & [mln::c2\\_col](#) ()  
*Vertical 2-connectivity neighborhood on the 2D grid.*
- const neighb2d & [mln::c2\\_row](#) ()  
*Horizontal 2-connectivity neighborhood on the 2D grid.*
- const neighb2d & [mln::c4](#) ()  
*4-connectivity neighborhood on the 2D grid.*
- const neighb2d & [mln::c8](#) ()  
*8-connectivity neighborhood on the 2D grid.*

## 8.19.1 Detailed Description

Predefined 2D neighborhoods.

## 8.19.2 Typedef Documentation

### 8.19.2.1 typedef neighb<window2d> mln::neighb2d

Type alias for a neighborhood defined on the 2D square grid with integer coordinates.

Definition at line 51 of file core/alias/neighb2d.hh.

## 8.19.3 Function Documentation

### 8.19.3.1 const neighb2d & mln::c2\_col ( ) [inline]

Vertical 2-connectivity neighborhood on the 2D grid.

```
- o -  
- x -  
- o -
```

#### Returns

A neighb2d.

Definition at line 190 of file core/alias/neighb2d.hh.

### 8.19.3.2 const neighb2d & mln::c2\_row ( ) [inline]

Horizontal 2-connectivity neighborhood on the 2D grid.

```
- - -  
o x o  
- - -
```

#### Returns

A neighb2d.

Definition at line 176 of file core/alias/neighb2d.hh.

### 8.19.3.3 const neighb2d & mln::c4 ( ) [inline]

4-connectivity neighborhood on the 2D grid.

```
- o -  
o x o  
- o -
```

**Returns**

A `neighb2d`.

Definition at line 148 of file `core/alias/neighb2d.hh`.

**8.19.3.4 const neighb2d & mln::c8 ( ) [inline]**

8-connectivity neighborhood on the 2D grid.

```

○ ○ ○
○ × ○
○ ○ ○

```

**Returns**

A `neighb2d`.

Definition at line 162 of file `core/alias/neighb2d.hh`.

**8.20 3D neighborhoods**

Predefined 3D neighborhoods.

**Typedefs**

- `typedef neighb< window3d > mln::neighb3d`  
*Type alias for a neighborhood defined on the 3D square grid with integer coordinates.*

**Functions**

- `const neighb3d & mln::c18 ( )`  
*18-connectivity neighborhood on the 3D grid.*
- `const neighb3d & mln::c26 ( )`  
*26-connectivity neighborhood on the 3D grid.*
- `const neighb3d & mln::c2_3d_sli ( )`  
*depth 2-connectivity neighborhood on the 3D grid.*
- `const neighb3d & mln::c4_3d ( )`  
*4-connectivity neighborhood on the 3D grid.*
- `const neighb3d & mln::c6 ( )`  
*6-connectivity neighborhood on the 3D grid.*
- `const neighb3d & mln::c8_3d ( )`  
*8-connectivity neighborhood on the 3D grid.*

### 8.20.1 Detailed Description

Predefined 3D neighborhoods.

### 8.20.2 Typedef Documentation

#### 8.20.2.1 `typedef neighb<window3d> mln::neighb3d`

Type alias for a neighborhood defined on the 3D square grid with integer coordinates.

Definition at line 50 of file `neighb3d.hh`.

### 8.20.3 Function Documentation

#### 8.20.3.1 `const neighb3d & mln::c18 ( ) [inline]`

18-connectivity neighborhood on the 3D grid.

```

  . o .
  o o o
  . o .

  o o o
  o x o
  o o o

  . o .
  o o o
  . o .

```

#### Returns

A `neighb3d`.

Definition at line 288 of file `neighb3d.hh`.

References `mln::c6()`, `mln::window< D >::insert()`, and `mln::win::sym()`.

Referenced by `mln::c26()`.

#### 8.20.3.2 `const neighb3d & mln::c26 ( ) [inline]`

26-connectivity neighborhood on the 3D grid.

```

  o o o
  o o o
  o o o

  o o o
  o x o
  o o o

  o o o
  o o o

```

o o o

### Returns

A neighb3d.

Definition at line 309 of file neighb3d.hh.

References `mln::c18()`, `mln::window< D >::insert()`, and `mln::win::sym()`.

#### 8.20.3.3 `const neighb3d & mln::c2_3d_sli ( ) [inline]`

depth 2-connectivity neighborhood on the 3D grid.

```

. . .
. o .
. . .

. . .
. x .
. . .

. . .
. o .
. . .

```

### Returns

A neighb3d.

Definition at line 226 of file neighb3d.hh.

References `mln::window< D >::insert()`.

#### 8.20.3.4 `const neighb3d & mln::c4_3d ( ) [inline]`

4-connectivity neighborhood on the 3D grid.

```

. . .
. . .
. . .

. o .
o x o
. o .

. . .
. . .
. . .

```

### Returns

A neighb3d.



Definition at line 241 of file neighb3d.hh.

References `mln::window< D >::insert()`, and `mln::win::sym()`.

### 8.20.3.5 `const neighb3d & mln::c6 ( ) [inline]`

6-connectivity neighborhood on the 3D grid.

```

. . .
. o .
. . .

. o .
o x o
. o .

. . .
. o .
. . .

```

#### Returns

A `neighb3d`.

Definition at line 271 of file neighb3d.hh.

References `mln::window< D >::insert()`, and `mln::win::sym()`.

Referenced by `mln::c18()`.

### 8.20.3.6 `const neighb3d & mln::c8_3d ( ) [inline]`

8-connectivity neighborhood on the 3D grid.

```

. . .
. . .
. . .

o o o
o x o
o o o

. . .
. . .
. . .

```

#### Returns

A `neighb3d`.

Definition at line 257 of file neighb3d.hh.

## 8.21 Site sets

All Site set types.

### Modules

- [Basic types](#)  
*Basic site sets.*
- [Graph based](#)  
*Site sets based on a graph.*
- [Complex based](#)  
*Site sets based on a complexes.*
- [Sparse types](#)  
*Sparse site sets.*
- [Queue based](#)  
*Site sets based on a queue.*

### 8.21.1 Detailed Description

All Site set types.

## 8.22 Basic types

Basic site sets.

### Classes

- class [mln::box< P >](#)  
*Generic box class: site set containing points of a regular grid.*
- class [mln::p\\_line2d](#)  
*2D discrete line of points.*
- class [mln::p\\_mutable\\_array\\_of< S >](#)  
*p\_mutable\_array\_of is a mutable array of site sets.*
- class [mln::p\\_run< P >](#)  
*Point set class in run.*

### 8.22.1 Detailed Description

Basic site sets.

## 8.23 Graph based

Site sets based on a graph.

### Classes

- class `mln::p_edges< G, F >`  
*Site set mapping graph edges and image sites.*
- struct `mln::p_faces< N, D, P >`  
*A complex psite set based on the N-faces of a complex of dimension D (a D-complex).*
- class `mln::p_vertices< G, F >`  
*Site set based mapping graph vertices to sites.*

### 8.23.1 Detailed Description

Site sets based on a graph.

## 8.24 Complex based

Site sets based on a complexes.

### Classes

- class `mln::p_complex< D, G >`  
*A complex psite set based on the N-faces of a complex of dimension D (a D-complex).*

### 8.24.1 Detailed Description

Site sets based on a complexes.

## 8.25 Sparse types

Sparse site sets.

### Classes

- class `mln::p_array< P >`  
*Multi-set of sites.*
- class `mln::p_centered< W >`  
*Site set corresponding to a window centered on a site.*

- class `mln::p_if< S, F >`  
*Site set restricted w.r.t.*
- class `mln::p_image< I >`  
*Site set based on an image of Booleans.*
- class `mln::p_set< P >`  
*Mathematical set of sites (based on `util::set`).*
- class `mln::p_transformed< S, F >`  
*Site set transformed through a function.*
- class `mln::p_vaccess< V, S >`  
*Site set in which sites are grouped by their associated value.*

### 8.25.1 Detailed Description

Sparse site sets.

## 8.26 Queue based

Site sets based on a queue.

### Classes

- class `mln::p_key< K, P >`  
*Priority queue class.*
- class `mln::p_priority< P, Q >`  
*Priority queue.*
- class `mln::p_queue< P >`  
*Queue of sites (based on `std::deque`).*
- class `mln::p_queue_fast< P >`  
*Queue of sites class (based on `p_array`).*

### 8.26.1 Detailed Description

Site sets based on a queue.

## 8.27 Utilities

Miscellaneous useful containers/structures.

## Classes

- class `mln::util::adjacency_matrix< V >`  
*A class of adjacency matrix.*
- class `mln::util::array< T >`  
*A dynamic array class.*
- class `mln::util::couple< T, U >`  
*Definition of a couple.*
- struct `mln::util::eat`  
*Eat structure.*
- class `mln::util::fibonacci_heap< P, T >`  
*Fibonacci heap.*
- struct `mln::util::ignore`  
*Ignore structure.*
- struct `mln::util::nil`  
*Nil structure.*
- struct `mln::util::ord_pair< T >`  
*Ordered pair structure s.a.*
- class `mln::util::set< T >`  
*An "efficient" mathematical set class.*
- class `mln::util::site_pair< P >`  
*A pair of sites.*
- class `mln::util::soft_heap< T, R >`  
*Soft heap.*
- struct `mln::util::tracked_ptr< T >`  
*Smart pointer for shared data with tracking.*
- struct `mln::util::yes`  
*Object that always says "yes".*

### 8.27.1 Detailed Description

Miscellaneous useful containers/structures.

## 8.28 Windows

All the predefined generic windows.

## Modules

- [1D windows](#)  
*Predefined 1D windows.*
- [2D windows](#)  
*Predefined 2D windows.*
- [3D windows](#)  
*Predefined 3D windows.*
- [N-D windows](#)  
*Predefined N-D windows.*
- [Multiple windows](#)  
*Generic multiple windows.*

### 8.28.1 Detailed Description

All the predefined generic windows.

## 8.29 1D windows

Predefined 1D windows.

### Typedefs

- typedef line< grid::tick, 0, def::coord > [mln::win::segment1d](#)  
*Segment window defined on the 1D grid.*
- typedef window< [mln::dpoint1d](#) > [mln::window1d](#)  
*Type alias for a window with arbitrary shape, defined on the 1D square grid with integer coordinates.*

### 8.29.1 Detailed Description

Predefined 1D windows.

### 8.29.2 Typedef Documentation

#### 8.29.2.1 typedef line<grid::tick, 0, def::coord> mln::win::segment1d

Segment window defined on the 1D grid.

An `segment1d` is centered and symmetric; so its height (length) is odd.

For instance:

○ × ○

is defined with length = 3.

Definition at line 56 of file segment1d.hh.

### 8.29.2.2 typedef window<mln::dpoint1d> mln::window1d

Type alias for a window with arbitrary shape, defined on the 1D square grid with integer coordinates.

Definition at line 47 of file window1d.hh.

## 8.30 2D windows

Predefined 2D windows.

### Classes

- struct [mln::win::backdiag2d](#)  
*Diagonal line window defined on the 2D square grid.*
- struct [mln::win::diag2d](#)  
*Diagonal line window defined on the 2D square grid.*
- struct [mln::win::octagon2d](#)  
*Octagon window defined on the 2D square grid.*
- struct [mln::win::rectangle2d](#)  
*Rectangular window defined on the 2D square grid.*

### Typedefs

- typedef ball< grid::square, def::coord > [mln::win::disk2d](#)  
*2D disk window; precisely, ball-shaped window defined on the 2D square grid.*
- typedef line< grid::square, 1, def::coord > [mln::win::hline2d](#)  
*Horizontal line window defined on the 2D square grid.*
- typedef line< grid::square, 0, def::coord > [mln::win::vline2d](#)  
*Vertical line window defined on the 2D square grid.*
- typedef window< [mln::dpoint2d](#) > [mln::window2d](#)  
*Type alias for a window with arbitrary shape, defined on the 2D square grid with integer coordinates.*

## Functions

- `const window2d & mln::win_c4p ()`  
*4-connectivity window on the 2D grid, including the center.*
- `const window2d & mln::win_c8p ()`  
*8-connectivity window on the 2D grid, including the center.*

### 8.30.1 Detailed Description

Predefined 2D windows.

### 8.30.2 Typedef Documentation

#### 8.30.2.1 `typedef ball<grid::square, def::coord> mln::win::disk2d`

2D disk window; precisely, ball-shaped window defined on the 2D square grid.

Definition at line 49 of file `disk2d.hh`.

#### 8.30.2.2 `typedef line<grid::square, 1, def::coord> mln::win::hline2d`

Horizontal line window defined on the 2D square grid.

An `hline2d` is centered and symmetric; so its height is 1 and its width (length) is odd.

For instance:

```
o o x o o
```

is defined with `length = 5`.

Definition at line 57 of file `hline2d.hh`.

#### 8.30.2.3 `typedef line<grid::square, 0, def::coord> mln::win::vline2d`

Vertical line window defined on the 2D square grid.

An `vline2d` is centered and symmetric; so its width is 1 and its height (length) is odd.

For instance:

```
o
x
o
```

is defined with `length = 3`.

Definition at line 58 of file `vline2d.hh`.



### 8.30.2.4 typedef window<mln::dpoint2d> mln::window2d

Type alias for a window with arbitrary shape, defined on the 2D square grid with integer coordinates.

Definition at line 49 of file window2d.hh.

## 8.30.3 Function Documentation

### 8.30.3.1 const window2d & mln::win\_c4p( ) [inline]

4-connectivity window on the 2D grid, including the center.

```

- o -
o x o
- o -

```

#### Returns

A window2d.

Definition at line 105 of file window2d.hh.

References mln::window< D >::insert(), and mln::window< D >::size().

### 8.30.3.2 const window2d & mln::win\_c8p( ) [inline]

8-connectivity window on the 2D grid, including the center.

```

o o o
o x o
o o o

```

#### Returns

A window2d.

Definition at line 121 of file window2d.hh.

References mln::window< D >::insert(), and mln::window< D >::size().

## 8.31 3D windows

Predefined 3D windows.

### Classes

- struct [mln::win::cube3d](#)  
*Cube window defined on the 3D grid.*
- struct [mln::win::cuboid3d](#)  
*Cuboid defined on the 3-D square grid.*

## Typedefs

- typedef line< grid::cube, 0, def::coord > mln::win::sline3d  
*Depth line window defined on the 3D cubic grid.*
- typedef ball< grid::cube, def::coord > mln::win::sphere3d  
*3D sphere window; precisely, ball-shaped window defined on the 3D cubic grid.*
- typedef window< mln::dpoint3d > mln::window3d  
*Type alias for a window with arbitrary shape, defined on the 3D square grid with integer coordinates.*

## Functions

- const window3d & mln::win\_c4p\_3d ()  
*4-connectivity window on the 3D grid, including the center.*
- const window3d & mln::win\_c8p\_3d ()  
*8-connectivity window on the 3D grid, including the center.*

### 8.31.1 Detailed Description

Predefined 3D windows.

### 8.31.2 Typedef Documentation

#### 8.31.2.1 typedef line<grid::cube, 0, def::coord> mln::win::sline3d

Depth line window defined on the 3D cubic grid.

An sline3d is centered and symmetric; so its height and its width are 1 and its depth is odd.

For instance:

```

. . .
. o .
. . .

. . .
. x .
. . .

. . .
. o .
. . .

```

is defined with length = 3.

Definition at line 68 of file sline3d.hh.

**8.31.2.2 typedef ball<grid::cube, def::coord> mln::win::sphere3d**

3D sphere window; precisely, ball-shaped window defined on the 3D cubic grid.

Definition at line 47 of file sphere3d.hh.

**8.31.2.3 typedef window<mln::dpoint3d> mln::window3d**

Type alias for a window with arbitrary shape, defined on the 3D square grid with integer coordinates.

Definition at line 48 of file window3d.hh.

**8.31.3 Function Documentation****8.31.3.1 const window3d & mln::win\_c4p\_3d ( ) [inline]**

4-connectivity window on the 3D grid, including the center.

```

- - -
- - -
- - -

- o -
o x o
- o -

- - -
- - -
- - -

```

**Returns**

A window3d.

Definition at line 119 of file window3d.hh.

References mln::window< D >::insert(), and mln::window< D >::size().

**8.31.3.2 const window3d & mln::win\_c8p\_3d ( ) [inline]**

8-connectivity window on the 3D grid, including the center.

```

- - -
- - -
- - -

o o o
o x o
o o o

- - -
- - -
- - -

```

**Returns**

A window3d.

Definition at line 135 of file window3d.hh.

References `mln::window< D >::insert()`, and `mln::window< D >::size()`.

## 8.32 N-D windows

Predefined N-D windows.

### Classes

- struct `mln::win::ball< G, C >`  
*Generic ball window defined on a given grid.*
- struct `mln::win::line< M, i, C >`  
*Generic line window defined on a given grid in the given dimension.*

### 8.32.1 Detailed Description

Predefined N-D windows.

## 8.33 Multiple windows

Generic multiple windows.

### Classes

- class `mln::win::multiple< W, F >`  
*Multiple window.*
- class `mln::win::multiple_size< n, W, F >`  
*Definition of a multiple-size window.*

### 8.33.1 Detailed Description

Generic multiple windows.

## 8.34 v2w2v functions

All bijective functions.

All bijective functions.

**8.35 v2w\_w2v functions**

All bijective function.

All bijective function.

**8.36 vv2b functions**

All functions mapping two values to a logical value.

All functions mapping two values to a logical value.



# Chapter 9

## Namespace Documentation

### 9.1 mln Namespace Reference

[mln/convert/to\\_image.hh](#)

#### Namespaces

- namespace [accu](#)  
*Namespace of accumulators.*
- namespace [algebra](#)  
*Namespace of algebraic structure.*
- namespace [arith](#)  
*Namespace of arithmetic.*
- namespace [binarization](#)  
*Namespace of "point-wise" expression tools.*
- namespace [border](#)  
*Namespace of routines related to image virtual (outer) border.*
- namespace [canvas](#)  
*Namespace of canvas.*
- namespace [convert](#)  
*Namespace of conversion routines.*
- namespace [data](#)  
*Namespace of image processing routines related to pixel data.*
- namespace [debug](#)  
*Namespace of routines that help to debug.*
- namespace [def](#)

*Namespace for core definitions.*

- namespace [display](#)  
*Namespace of routines that help to display images.*
- namespace [doc](#)  
*The namespace `mln::doc` is only for documentation purpose.*
- namespace [draw](#)  
*Namespace of drawing routines.*
- namespace [estim](#)  
*Namespace of estimation materials.*
- namespace [extension](#)  
*Namespace of extension tools.*
- namespace [fun](#)  
*Namespace of functions.*
- namespace [geom](#)  
*Namespace of all things related to geometry.*
- namespace [graph](#)  
*Namespace of graph related routines.*
- namespace [grid](#)  
*Namespace of grids definitions.*
- namespace [histo](#)  
*Namespace of histograms.*
- namespace [impl](#)  
*Implementation namespace of mln namespace.*
- namespace [io](#)  
*Namespace of input/output handling.*
- namespace [labeling](#)  
*Namespace of labeling routines.*
- namespace [linear](#)  
*Namespace of linear image processing routines.*
- namespace [literal](#)  
*Namespace of literals.*
- namespace [logical](#)  
*Namespace of logic.*



- namespace [make](#)  
*Namespace of routines that help to make Milena's objects.*
- namespace [math](#)  
*Namespace of mathematical routines.*
- namespace [metal](#)  
*Namespace of meta-programming tools.*
- namespace [morpho](#)  
*Namespace of mathematical morphology routines.*
- namespace [norm](#)  
*Namespace of norms.*
- namespace [opt](#)  
*Namespace of optional routines.*
- namespace [pw](#)  
*Namespace of "point-wise" expression tools.*
- namespace [registration](#)  
*Namespace of "point-wise" expression tools.*
- namespace [select](#)  
*Select namespace (FIXME doc).*
- namespace [set](#)  
*Namespace of image processing routines related to pixel sets.*
- namespace [subsampling](#)  
*Namespace of "point-wise" expression tools.*
- namespace [tag](#)  
*Namespace of image processing routines related to tags.*
- namespace [test](#)  
*Namespace of image processing routines related to pixel tests.*
- namespace [topo](#)  
*Namespace of "point-wise" expression tools.*
- namespace [trace](#)  
*Namespace of routines related to the trace mechanism.*
- namespace [trait](#)  
*Namespace where traits are defined.*
- namespace [transform](#)  
*Namespace of transforms.*

- namespace [util](#)  
*Namespace of tools using for more complex algorithm.*
- namespace [value](#)  
*Namespace of materials related to pixel value types.*
- namespace [win](#)  
*Namespace of image processing routines related to win.*

## Classes

- struct [Accumulator](#)  
*Base class for implementation of accumulators.*
- class [bkd\\_pixter1d](#)  
*Backward pixel iterator on a 1-D image with border.*
- class [bkd\\_pixter2d](#)  
*Backward pixel iterator on a 2-D image with border.*
- class [bkd\\_pixter3d](#)  
*Backward pixel iterator on a 3-D image with border.*
- class [box](#)  
*Generic box class: site set containing points of a regular grid.*
- struct [Box](#)  
*Base class for implementation classes of boxes.*
- class [box\\_runend\\_piter](#)  
*A generic backward iterator on points by lines.*
- class [box\\_runstart\\_piter](#)  
*A generic forward iterator on points by lines.*
- struct [Browsing](#)  
*Base class for implementation classes that are browsings.*
- struct [category< R\(\\*\) \(A\) >](#)  
*Category declaration for a unary C function.*
- class [complex\\_image](#)  
*Image based on a complex.*
- class [complex\\_neighborhood\\_bkd\\_piter](#)  
*Backward iterator on complex neighborhood.*

- class `complex_neighborhood_fwd_piter`  
*Forward iterator on complex neighborhood.*
- class `complex_psite`  
*Point site associated to a `mln::p_complex`.*
- class `complex_window_bkd_piter`  
*Backward iterator on complex window.*
- class `complex_window_fwd_piter`  
*Forward iterator on complex window.*
- struct `decorated_image`  
*Image that can have additional features.*
- struct `Delta_Point_Site< void >`  
*Delta point site category flag type.*
- struct `dpoint`  
*Generic delta-point class.*
- struct `Dpoint`  
*Base class for implementation of delta-point classes.*
- class `dpoints_bkd_pixter`  
*A generic backward iterator on the pixels of a dpoint-based window or neighborhood.*
- class `dpoints_fwd_pixter`  
*A generic forward iterator on the pixels of a dpoint-based window or neighborhood.*
- struct `Edge`  
*edge category flag type.*
- class `edge_image`  
*Image based on graph edges.*
- struct `extended`  
*Makes an image become restricted by a point set.*
- class `extension_fun`  
*Extends the domain of an image with a function.*
- class `extension_ima`  
*Extends the domain of an image with an image.*
- class `extension_val`  
*Extends the domain of an image with a value.*
- class `faces_psite`  
*Point site associated to a `mln::p_faces`.*

- struct [flat\\_image](#)  
*Image with a single value.*
- struct [fun\\_image](#)  
*Image read through a function.*
- struct [Function](#)  
*Base class for implementation of function-objects.*
- struct [Function< void >](#)  
*Function category flag type.*
- struct [Function\\_n2v](#)  
*Base class for implementation of function-objects from Nil to value.*
- struct [Function\\_v2b](#)  
*Base class for implementation of function-objects from a value to a Boolean.*
- struct [Function\\_v2v](#)  
*Base class for implementation of function-objects from value to value.*
- struct [Function\\_vv2b](#)  
*Base class for implementation of function-objects from a couple of values to a Boolean.*
- struct [Function\\_vv2v](#)  
*Base class for implementation of function-objects from a couple of values to a value.*
- class [fwd\\_pixter1d](#)  
*Forward pixel iterator on a 1-D image with border.*
- class [fwd\\_pixter2d](#)  
*Forward pixel iterator on a 2-D image with border.*
- class [fwd\\_pixter3d](#)  
*Forward pixel iterator on a 3-D image with border.*
- struct [Gdpoint](#)  
*FIXME: Doc!*
- struct [Gdpoint< void >](#)  
*Delta point site category flag type.*
- struct [Gpoint](#)  
*Base class for implementation of point classes.*
- struct [Graph](#)  
*Base class for implementation of graph classes.*
- struct [graph\\_elt\\_mixed\\_neighborhood](#)

*Elementary neighborhood on graph class.*

- class [graph\\_elt\\_mixed\\_window](#)  
*Elementary window on graph class.*
- struct [graph\\_elt\\_neighborhood](#)  
*Elementary neighborhood on graph class.*
- struct [graph\\_elt\\_neighborhood\\_if](#)  
*Elementary neighborhood\_if on graph class.*
- class [graph\\_elt\\_window](#)  
*Elementary window on graph class.*
- class [graph\\_elt\\_window\\_if](#)  
*Custom window on graph class.*
- class [graph\\_window\\_base](#)
- class [graph\\_window\\_if\\_piter](#)  
*Forward iterator on line graph window.*
- class [graph\\_window\\_piter](#)  
*Forward iterator on line graph window.*
- struct [hexa](#)  
*hexagonal image class.*
- struct [Image](#)  
*Base class for implementation of image classes.*
- struct [image1d](#)  
*Basic 1D image class.*
- class [image2d](#)  
*Basic 2D image class.*
- struct [image2d\\_h](#)  
*2d image based on an hexagonal mesh.*
- struct [image3d](#)  
*Basic 3D image class.*
- struct [image\\_if](#)  
*Image which domain is restricted by a function 'site -> Boolean'.*
- struct [interpolated](#)  
*Makes the underlying image being accessed with floating coordinates.*
- struct [Iterator](#)  
*Base class for implementation classes that are iterators.*

- class [labeled\\_image](#)  
*Morpher providing an improved interface for labeled image.*
- class [labeled\\_image\\_base](#)  
*Base class Morpher providing an improved interface for labeled image.*
- struct [lazy\\_image](#)  
*Image values are computed on the fly.*
- struct [Literal](#)  
*Base class for implementation classes of literals.*
- struct [Mesh](#)  
*Base class for implementation classes of meshes.*
- struct [Meta\\_Accumulator](#)  
*Base class for implementation of meta accumulators.*
- struct [Meta\\_Function](#)  
*Base class for implementation of meta functions.*
- struct [Meta\\_Function\\_v2v](#)  
*Base class for implementation of function-objects from value to value.*
- struct [Meta\\_Function\\_vv2v](#)  
*Base class for implementation of function-objects from value to value.*
- class [mixed\\_neighb](#)  
*Adapter class from window to neighborhood.*
- class [neighb](#)  
*Adapter class from window to neighborhood.*
- struct [Neighborhood](#)  
*Base class for implementation classes that are neighborhoods.*
- struct [Neighborhood< void >](#)  
*Neighborhood category flag type.*
- struct [Object](#)  
*Base class for almost every class defined in Milena.*
- struct [p2p\\_image](#)  
*FIXME: Doc!*
- class [p\\_array](#)  
*Multi-set of sites.*
- class [p\\_centered](#)

*Site set corresponding to a window centered on a site.*

- class [p\\_complex](#)  
*A complex psite set based on the N-faces of a complex of dimension D (a D-complex).*
- class [p\\_edges](#)  
*Site set mapping graph edges and image sites.*
- struct [p\\_faces](#)  
*A complex psite set based on a the N-faces of a complex of dimension D (a D-complex).*
- class [p\\_graph\\_piter](#)  
*Generic iterator on point sites of a `mln::S`.*
- class [p\\_if](#)  
*Site set restricted w.r.t.*
- class [p\\_image](#)  
*Site set based on an image of Booleans.*
- class [p\\_indexed\\_bkd\\_piter](#)  
*Backward iterator on sites of an indexed site set.*
- class [p\\_indexed\\_fwd\\_piter](#)  
*Forward iterator on sites of an indexed site set.*
- class [p\\_indexed\\_psite](#)  
*Psite class for indexed site sets such as [p\\_array](#).*
- class [p\\_key](#)  
*Priority queue class.*
- class [p\\_line2d](#)  
*2D discrete line of points.*
- class [p\\_mutable\\_array\\_of](#)  
*[p\\_mutable\\_array\\_of](#) is a mutable array of site sets.*
- class [p\\_n\\_faces\\_bkd\\_piter](#)  
*Backward iterator on the n-faces sites of an `mln::p_complex<D, G>`.*
- class [p\\_n\\_faces\\_fwd\\_piter](#)  
*Forward iterator on the n-faces sites of an `mln::p_complex<D, G>`.*
- class [p\\_priority](#)  
*Priority queue.*
- class [p\\_queue](#)  
*Queue of sites (based on `std::deque`).*

- class [p\\_queue\\_fast](#)  
*Queue of sites class (based on [p\\_array](#)).*
- class [p\\_run](#)  
*Point set class in run.*
- class [p\\_set](#)  
*Mathematical set of sites (based on [util::set](#)).*
- class [p\\_set\\_of](#)  
*[p\\_set\\_of](#) is a set of site sets.*
- class [p\\_transformed](#)  
*Site set transformed through a function.*
- struct [p\\_transformed\\_piter](#)  
*Iterator on [p\\_transformed](#)<S,F>.*
- class [p\\_vaccess](#)  
*Site set in which sites are grouped by their associated value.*
- class [p\\_vertices](#)  
*Site set based mapping graph vertices to sites.*
- struct [pixel](#)  
*Generic pixel class.*
- struct [Pixel\\_Iterator](#)  
*Base class for the implementation of pixel iterator classes.*
- class [plain](#)  
*Prevents an image from sharing its data.*
- struct [point](#)  
*Generic point class.*
- struct [Point](#)  
*Base class for implementation of point classes.*
- struct [Point\\_Site](#)  
*Base class for implementation classes of the notion of "point site".*
- struct [Point\\_Site< void >](#)  
*Point site category flag type.*
- struct [Proxy](#)  
*Base class for implementation classes of the notion of "proxy".*
- struct [Proxy< void >](#)  
*Proxy category flag type.*



- struct [Pseudo\\_Site](#)  
*Base class for implementation classes of the notion of "pseudo site".*
- struct [Pseudo\\_Site< void >](#)  
*Pseudo\_Site category flag type.*
- struct [Regular\\_Grid](#)  
*Base class for implementation classes of regular grids.*
- class [safe\\_image](#)  
*Makes an image accessible at undefined location.*
- struct [Site](#)  
*Base class for classes that are explicitly sites.*
- struct [Site< void >](#)  
*Site category flag type.*
- struct [Site\\_Iterator](#)  
*Base class for implementation of classes of iterator on points.*
- struct [Site\\_Proxy](#)  
*Base class for implementation classes of the notion of "site proxy".*
- struct [Site\\_Proxy< void >](#)  
*Site\_Proxy category flag type.*
- struct [Site\\_Set](#)  
*Base class for implementation classes of site sets.*
- struct [Site\\_Set< void >](#)  
*Site\_Set category flag type.*
- struct [slice\\_image](#)  
*2D image extracted from a slice of a 3D image.*
- class [sub\\_image](#)  
*Image having its domain restricted by a site set.*
- struct [sub\\_image\\_if](#)  
*Image having its domain restricted by a site set and a function.*
- class [thru\\_image](#)  
*Morph image values through a function.*
- class [thrubin\\_image](#)  
*Morphes values from two images through a binary function.*
- struct [tr\\_image](#)

*Transform an image by a given transformation.*

- struct [unproject\\_image](#)  
*Un-projects an image.*
- struct [Value](#)  
*Base class for implementation classes of values.*
- struct [Value\\_Iterator](#)  
*Base class for implementation of classes of iterator on values.*
- struct [Value\\_Set](#)  
*Base class for implementation classes of sets of values.*
- struct [Vertex](#)  
*Vertex category flag type.*
- class [vertex\\_image](#)  
*Image based on graph vertices.*
- struct [violent\\_cast\\_image](#)  
*Violently cast image values to a given type.*
- struct [w\\_window](#)  
*Generic [w\\_window](#) class.*
- struct [Weighted\\_Window](#)  
*Base class for implementation classes that are [weighted\\_windows](#).*
- class [window](#)  
*Generic window class.*
- struct [Window](#)  
*Base class for implementation classes that are windows.*

## Typedefs

- typedef [mln::complex\\_image](#)< 1, [mln::discrete\\_plane\\_1complex\\_geometry](#), bool > [bin\\_1complex\\_image2d](#)  
*Type alias for a binary image based on a 1-complex, where 0-faces are located at discrete (integer) 2-dimensional points.*
- typedef [mln::complex\\_image](#)< 2, [mln::space\\_2complex\\_geometry](#), bool > [bin\\_2complex\\_image3df](#)  
*Type alias for a binary image based on a 2-complex, where 0-faces are located at floating-point 3-dimensional points.*
- typedef [box](#)< [mln::point1d](#) > [box1d](#)  
*Type alias for a box defined on the 1D square grid with integer coordinates.*

- typedef [box](#)< [mln::point2d](#) > [box2d](#)  
*Type alias for a box defined on the 2D square grid with integer coordinates.*
- typedef [box](#)< [point2d\\_h](#) > [box2d\\_h](#)  
*FIXME.*
- typedef [box](#)< [point3d](#) > [box3d](#)  
*Type alias for a box defined on the 3D square grid with integer coordinates.*
- typedef [mln::geom::complex\\_geometry](#)< 1, [point2d](#) > [discrete\\_plane\\_1complex\\_geometry](#)  
*Type alias for the geometry of a 1-complex (e.g., a graph) located in a discrete 2-dimensional plane (with integer coordinates).*
- typedef [mln::geom::complex\\_geometry](#)< 2, [point2d](#) > [discrete\\_plane\\_2complex\\_geometry](#)  
*Type alias for the geometry of a 2-complex located in a discrete 2-dimensional plane (with integer coordinates).*
- typedef [dpoint](#)< [mln::grid::tick](#), [def::coord](#) > [dpoint1d](#)  
*Type alias for a delta-point defined on the 1D square grid with integer coordinates.*
- typedef [dpoint](#)< [mln::grid::square](#), [mln::def::coord](#) > [dpoint2d](#)  
*Type alias for a delta-point defined on the 2D square grid with integer coordinates.*
- typedef [dpoint](#)< [mln::grid::hexa](#), [def::coord](#) > [dpoint2d\\_h](#)  
*Type alias for a delta-point defined on the 2D square grid with integer coordinates.*
- typedef [dpoint](#)< [mln::grid::cube](#), [def::coord](#) > [dpoint3d](#)  
*Type alias for a delta-point defined on the 3D square grid with integer coordinates.*
- typedef [mln::complex\\_image](#)< 2, [mln::space\\_2complex\\_geometry](#), [float](#) > [float\\_2complex\\_image3df](#)  
*Type alias for a floating-point image based on a 2-complex, where 0-faces are located at floating-point 3-dimensional points.*
- typedef [mln::complex\\_image](#)< 1, [mln::discrete\\_plane\\_1complex\\_geometry](#), [mln::value::int\\_u8](#) > [int\\_u8\\_1complex\\_image2d](#)  
*Type alias for an 8-bit gray-level image based on a 1-complex, where 0-faces are located at discrete (integer) 2-dimensional points.*
- typedef [mln::complex\\_image](#)< 2, [mln::discrete\\_plane\\_2complex\\_geometry](#), [mln::value::int\\_u8](#) > [int\\_u8\\_2complex\\_image2d](#)  
*Type alias for an 8-bit gray-level image based on a 2-complex, where 0-faces are located at discrete (integer) 2-dimensional points.*
- typedef [mln::complex\\_image](#)< 2, [mln::space\\_2complex\\_geometry](#), [mln::value::int\\_u8](#) > [int\\_u8\\_2complex\\_image3df](#)  
*Type alias for an 8-bit gray-level image based on a 2-complex, where 0-faces are located at floating-point 3-dimensional points.*
- typedef [neighb](#)< [window1d](#) > [neighb1d](#)

*Type alias for a neighborhood defined on the 1D square grid with integer coordinates.*

- typedef [neighb](#)< [window2d](#) > [neighb2d](#)  
*Type alias for a neighborhood defined on the 2D square grid with integer coordinates.*
- typedef [neighb](#)< [window3d](#) > [neighb3d](#)  
*Type alias for a neighborhood defined on the 3D square grid with integer coordinates.*
- typedef [p\\_run](#)< [point2d](#) > [p\\_run2d](#)  
*Type alias for a run of 2d points.*
- typedef [p\\_set\\_of](#)< [p\\_run2d](#) > [p\\_runs2d](#)  
*Type alias for a set of runs of 2d points.*
- typedef [point](#)< [grid::tick](#), [def::coordf](#) > [point1df](#)  
*Type alias for a point defined on the 1D ruler with floating-point coordinates.*
- typedef [point](#)< [mln::grid::square](#), [mln::def::coordf](#) > [point2df](#)  
*Type alias for a point defined on the 2D square grid with floating-point coordinates.*
- typedef [point](#)< [grid::cube](#), [def::coordf](#) > [point3df](#)  
*Type alias for a point defined on the 3D square grid with floating-point coordinates.*
- typedef [mln::complex\\_image](#)< 2, [mln::space\\_2complex\\_geometry](#), [mln::value::rgb8](#) > [rgb8\\_2complex\\_image3df](#)  
*Type alias for a (3x8-bit) RGB image based on a 2-complex, where 0-faces are located at floating-point 3-dimensional points.*
- typedef [mln::geom::complex\\_geometry](#)< 2, [point3df](#) > [space\\_2complex\\_geometry](#)  
*Type alias for the geometry of a 2-complex located in a 3-dimensional space (with floating-point coordinates).*
- typedef [mln::complex\\_image](#)< 2, [mln::space\\_2complex\\_geometry](#), [unsigned](#) > [unsigned\\_2complex\\_image3df](#)  
*Type alias for a gray-level image based on a 2-complex, where 0-faces are located at floating-point 3-dimensional points.*
- typedef [algebra::vec](#)< 2u, [double](#) > [vec2d\\_d](#)  
*2D vector with double coordinates.*
- typedef [algebra::vec](#)< 2u, [float](#) > [vec2d\\_f](#)  
*2D vector with float coordinates.*
- typedef [algebra::vec](#)< 3u, [double](#) > [vec3d\\_d](#)  
*3D vector with double coordinates.*
- typedef [algebra::vec](#)< 3u, [float](#) > [vec3d\\_f](#)  
*3D vector with float coordinates.*
- typedef [w\\_window](#)< [dpoint1d](#), [float](#) > [w\\_window1d\\_float](#)

Type alias for a `w_window` with arbitrary shape, defined on the 1D grid (with integer coordinates) and whose weights are floating values.

- typedef `w_window< dpoint1d, int > w_window1d_int`  
Type alias for a `w_window` with arbitrary shape, defined on the 1D grid (with integer coordinates) and whose weights are integers.
- typedef `w_window< dpoint2d, float > w_window2d_float`  
Type alias for a `w_window` with arbitrary shape, defined on the 2D square grid (with integer coordinates) and whose weights are floating values.
- typedef `w_window< dpoint2d, int > w_window2d_int`  
Type alias for a `w_window` with arbitrary shape, defined on the 2D square grid (with integer coordinates) and whose weights are integers.
- typedef `w_window< dpoint3d, float > w_window3d_float`  
Type alias for a `w_window` with arbitrary shape, defined on the 3D grid (with integer coordinates) and whose weights are floating values.
- typedef `w_window< dpoint3d, int > w_window3d_int`  
Type alias for a `w_window` with arbitrary shape, defined on the 3D grid (with integer coordinates) and whose weights are integers.
- typedef `window< mln::dpoint1d > window1d`  
Type alias for a window with arbitrary shape, defined on the 1D square grid with integer coordinates.
- typedef `window< mln::dpoint2d > window2d`  
Type alias for a window with arbitrary shape, defined on the 2D square grid with integer coordinates.
- typedef `window< mln::dpoint3d > window3d`  
Type alias for a window with arbitrary shape, defined on the 3D square grid with integer coordinates.
- typedef `point< grid::tick, def::coord > point1d`  
Type alias for a point defined on the 1D ruler with integer coordinates.
- typedef `point< mln::grid::square, mln::def::coord > point2d`  
Type alias for a point defined on the 2D square grid with integer coordinates.
- typedef `point< grid::hexa, def::coord > point2d_h`  
Type alias for a point defined on the 2D hexagonal grid with integer coordinates.
- typedef `point< grid::cube, def::coord > point3d`  
Type alias for a point defined on the 3D square grid with integer coordinates.

## Functions

- template<typename I >  
I::site `a_point_of` (const `Image< I > &ima`)  
Give a point of an image.

- `template<typename I , typename F >`  
`p2p_image< I, F > apply_p2p (Image< I > &ima, const Function_v2v< F > &f)`  
*FIXME: Doc!*
- `template<typename I , typename F >`  
`p2p_image< const I, F > apply_p2p (const Image< I > &ima, const Function_v2v< F > &f)`  
*FIXME: Doc!*
- `const neighb3d & c18 ()`  
*18-connectivity neighborhood on the 3D grid.*
- `const neighb1d & c2 ()`  
*2-connectivity neighborhood on the 1D grid.*
- `const neighb3d & c26 ()`  
*26-connectivity neighborhood on the 3D grid.*
- `const neighb3d & c2_3d_sli ()`  
*depth 2-connectivity neighborhood on the 3D grid.*
- `const neighb2d & c2_col ()`  
*Vertical 2-connectivity neighborhood on the 2D grid.*
- `const neighb2d & c2_row ()`  
*Horizontal 2-connectivity neighborhood on the 2D grid.*
- `const neighb2d & c4 ()`  
*4-connectivity neighborhood on the 2D grid.*
- `const neighb3d & c4_3d ()`  
*4-connectivity neighborhood on the 3D grid.*
- `const neighb3d & c6 ()`  
*6-connectivity neighborhood on the 3D grid.*
- `const neighb2d & c8 ()`  
*8-connectivity neighborhood on the 2D grid.*
- `const neighb3d & c8_3d ()`  
*8-connectivity neighborhood on the 3D grid.*
- `template<typename T2 , typename T1 >`  
`fun::x2x::composed< T2, T1 > compose (T2 f, T1 g)`  
*Do a composition of two transformations.*
- `template<typename I >`  
`mln::trait::concrete< I >::ret duplicate (const Image< I > &model)`  
*Duplicate the image model with the values of the image data.*

- `template<typename I, typename F >`  
`extension_fun< const I, F > extend` (const `Image< I >` &ima, const `Function_v2v< F >` &fun)  
*Routines for domain extension with a function.*
- `template<typename I, typename J >`  
`extension_ima< const I, const J > extend` (const `Image< I >` &ima, const `Image< J >` &ext)  
*Routines for domain extension with an image.*
- `template<typename I >`  
`extension_val< const I > extend` (const `Image< I >` &ima, const `typename I::value` &val)  
*Routines for domain extension with a value.*
- `bool implies` (bool lexpr, bool rexpr)  
*Implication.*
- `template<typename I, typename J >`  
`void initialize` (`Image< I >` &target, const `Image< J >` &model)
- `template<typename P >`  
`box< P > larger_than` (const `box< P >` a, const `box< P >` b)  
*Return the minimum box including box a and box b.*
- `template<typename I, typename V, typename E >`  
`image2d< typename I::value > make_debug_graph_image` (const I &input, const V &ima\_v, const E &ima\_e, const `value::rgb8` &bg)  
*Draw a graph.*
- `mln_gen_complex_neighborhood` (`complex_higher_dim_connected_n_face_neighborhood`, `complex_higher_dim_connected_n_face_window`)  
*Neighborhood centered on an n-face of complex returning the n-faces sharing an (n+1)-face with the center n-face.*
- `mln_gen_complex_neighborhood` (`complex_m_face_neighborhood`, `complex_m_face_window`)  
*Neighborhood centered on an n-face of complex returning the m-faces transitively adjacent to this center n-face.*
- `mln_gen_complex_neighborhood` (`complex_lower_neighborhood`, `complex_lower_window`)  
*Neighborhood centered on an n-face of complex returning its adjacent (n-1)-faces.*
- `mln_gen_complex_neighborhood` (`complex_higher_neighborhood`, `complex_higher_window`)  
*Neighborhood centered on an n-face of complex returning its adjacent (n+1)-faces.*
- `mln_gen_complex_neighborhood` (`complex_lower_higher_neighborhood`, `complex_lower_higher_window`)  
*Neighborhood centered on an n-face of complex returning its adjacent (n-1)-faces and (n+1)-faces.*
- `mln_gen_complex_neighborhood` (`complex_lower_dim_connected_n_face_neighborhood`, `complex_lower_dim_connected_n_face_window`)  
*Neighborhood centered on an n-face of complex returning the n-faces sharing an (n-1)-face with the center n-face.*

- `mln_gen_complex_window` (`complex_lower_window`, `topo::adj_lower_face_fwd_iter`, `topo::adj_lower_face_bkd_iter`)  
*Window centered on an  $n$ -face of complex returning its adjacent  $(n-1)$ -faces.*
- `mln_gen_complex_window` (`complex_higher_window`, `topo::adj_higher_face_fwd_iter`, `topo::adj_higher_face_bkd_iter`)  
*Window centered on an  $n$ -face of complex returning its adjacent  $(n+1)$ -faces.*
- `mln_gen_complex_window` (`complex_lower_higher_window`, `topo::adj_lower_higher_face_fwd_iter`, `topo::adj_lower_higher_face_bkd_iter`)  
*Window centered on an  $n$ -face of complex returning its adjacent  $(n-1)$ -faces and  $(n+1)$ -faces.*
- `mln_gen_complex_window` (`complex_lower_dim_connected_n_face_window`, `topo::adj_lower_dim_connected_n_face_fwd_iter`, `topo::adj_lower_dim_connected_n_face_bkd_iter`)  
*Window centered on an  $n$ -face of complex returning the  $n$ -faces sharing an  $(n-1)$ -face with the center  $n$ -face.*
- `mln_gen_complex_window` (`complex_higher_dim_connected_n_face_window`, `topo::adj_higher_dim_connected_n_face_fwd_iter`, `topo::adj_higher_dim_connected_n_face_bkd_iter`)  
*Window centered on an  $n$ -face of complex returning the  $n$ -faces sharing an  $(n+1)$ -face with the center  $n$ -face.*
- `mln_gen_complex_window` (`complex_m_face_window`, `topo::adj_m_face_fwd_iter`, `topo::adj_m_face_bkd_iter`)  
*Window centered on an  $n$ -face of complex returning the  $m$ -faces transitively adjacent to this center  $n$ -face.*
- `mln_gen_complex_window_p` (`complex_lower_window_p`, `topo::adj_lower_face_fwd_iter`, `topo::adj_lower_face_bkd_iter`)  
*Window centered on an  $n$ -face of complex returning its adjacent  $(n-1)$ -faces as well as the center  $n$ -face.*
- `mln_gen_complex_window_p` (`complex_higher_window_p`, `topo::adj_higher_face_fwd_iter`, `topo::adj_higher_face_bkd_iter`)  
*Window centered on an  $n$ -face of complex returning its adjacent  $(n+1)$ -faces as well as the center  $n$ -face.*
- `mln_gen_complex_window_p` (`complex_lower_higher_window_p`, `topo::adj_lower_higher_face_fwd_iter`, `topo::adj_lower_higher_face_bkd_iter`)  
*Window centered on an  $n$ -face of complex returning its adjacent  $(n-1)$ -faces and  $(n+1)$ -faces as well as the center  $n$ -face.*
- `mln_gen_complex_window_p` (`complex_lower_dim_connected_n_face_window_p`, `topo::adj_lower_dim_connected_n_face_fwd_iter`, `topo::adj_lower_dim_connected_n_face_bkd_iter`)  
*Window centered on an  $n$ -face of complex returning the  $n$ -faces sharing an  $(n-1)$ -face with the center  $n$ -face, as well as this center  $n$ -face.*
- `mln_gen_complex_window_p` (`complex_higher_dim_connected_n_face_window_p`, `topo::adj_higher_dim_connected_n_face_fwd_iter`, `topo::adj_higher_dim_connected_n_face_bkd_iter`)  
*Window centered on an  $n$ -face of complex returning the  $n$ -faces sharing an  $(n+1)$ -face with the center  $n$ -face, as well as this center  $n$ -face.*
- `mln_gen_complex_window_p` (`complex_m_face_window_p`, `topo::adj_m_face_fwd_iter`, `topo::adj_m_face_bkd_iter`)  
*Window centered on an  $n$ -face of complex returning the  $m$ -faces transitively adjacent to this center  $n$ -face, as well as this center  $n$ -face.*



- `template<typename W1 , typename W2 >`  
`mln_reguler` (W1) operator-(const `Window`< W1 > &win1  
*Set difference between a couple of windows win1 and win2.*
- `template<typename O1 , typename O2 >`  
`mln_trait_op_geq` (O1, O2) operator>  
*General definition of the "greater than or equal to" operator.*
- `template<typename O1 , typename O2 >`  
`mln_trait_op_greater` (O1, O2) operator>(const `Object`< O1 > &lhs  
*General definition of the "greater than" operator.*
- `template<typename O1 , typename O2 >`  
`mln_trait_op_leq` (O1, O2) operator<  
*Default definition of the "less than or equal to" operator.*
- `template<typename O1 , typename O2 >`  
`mln_trait_op_neq` (O1, O2) operator!  
*General definition of the "not equal to" operator.*
- `template<typename P , typename S >`  
`P operator*` (const `Gpoint`< P > &p, const `value::scalar_< S >` &s)  
*Multiply a point p by a scalar s.*
- `template<typename S >`  
`S & operator++` (`value::Scalar< S >` &rhs)  
*Pre-incrementation for any scalar type.*
- `template<typename P , typename D >`  
`P operator-` (const `Gpoint`< P > &p, const `Gdpoint`< D > &dp)  
*Subtract a delta-point dp to a grid point p.*
- `template<typename N1 , typename N2 >`  
`neighb`< `typename N1::window::regular` > `operator-` (const `Neighborhood`< N1 > &nbh1, const `Neighborhood`< N2 > &nbh2)  
*Set difference between a couple of neighborhoods nbh1 and nbh2.*
- `template<typename S >`  
`S & operator--` (`value::Scalar< S >` &rhs)  
*Pre-decrementation for any scalar type.*
- `template<typename L , typename R >`  
`bool operator<` (const `Image`< L > &lhs, const `Image`< R > &rhs)  
*Point-wise test if the pixel values of lhs are point-wise less than the pixel values of rhs.*
- `template<typename I , typename G , typename W >`  
`std::ostream & operator<<` (`std::ostream` &ostr, const `complex_window_fwd_piter`< I, G, W > &p)  
*Print an mln::complex\_window\_fwd\_piter.*

- `template<typename I, typename G, typename W >`  
`std::ostream & operator<< (std::ostream &ostr, const complex\_window\_bkd\_piter< I, G, W >`  
`&p)`  
*Print an [mln::complex\\_window\\_bkd\\_piter](#).*
- `template<typename I, typename G, typename N >`  
`std::ostream & operator<< (std::ostream &ostr, const complex\_neighborhood\_fwd\_piter< I, G, N`  
`> &p)`  
*Print an [mln::complex\\_neighborhood\\_fwd\\_piter](#).*
- `template<typename I, typename G, typename N >`  
`std::ostream & operator<< (std::ostream &ostr, const complex\_neighborhood\_bkd\_piter< I, G, N`  
`> &p)`  
*Print an [mln::complex\\_neighborhood\\_bkd\\_piter](#).*
- `template<unsigned N, unsigned D, typename P >`  
`bool operator<= (const p\_faces< N, D, P > &lhs, const p\_faces< N, D, P > &rhs)`  
*Inclusion of a [mln::p\\_faces](#) in another one.*
- `template<typename G, typename F >`  
`bool operator<= (const p\_edges< G, F > &lhs, const p\_edges< G, F > &rhs)`  
*Inclusion of a [mln::p\\_edges](#) in another one.*
- `template<typename L, typename R >`  
`bool operator<= (const Image< L > &lhs, const Image< R > &rhs)`  
*Point-wise test if the pixel values of lhs are point-wise less than or equal to the pixel values of rhs.*
- `template<unsigned D, typename G >`  
`bool operator<= (const p\_complex< D, G > &lhs, const p\_complex< D, G > &rhs)`  
*Inclusion of a [mln::p\\_complex](#) in another one.*
- `template<typename G, typename F >`  
`bool operator<= (const p\_vertices< G, F > &lhs, const p\_vertices< G, F > &rhs)`  
*Inclusion of a [mln::p\\_vertices](#) in another one.*
- `template<typename G, typename F >`  
`bool operator== (const p\_edges< G, F > &lhs, const p\_edges< G, F > &rhs)`  
*Comparison between two [mln::p\\_edges](#)'s.*
- `template<typename L, typename R >`  
`bool operator== (const Image< L > &lhs, const Image< R > &rhs)`  
*Point-wise test if the pixel values of lhs are equal to the pixel values of rhs.*
- `template<unsigned D, typename G >`  
`bool operator== (const p\_complex< D, G > &lhs, const p\_complex< D, G > &rhs)`  
*Comparison between two [mln::p\\_complex](#)'s.*
- `template<unsigned N, unsigned D, typename P >`  
`bool operator== (const p\_faces< N, D, P > &lhs, const p\_faces< N, D, P > &rhs)`  
*Comparison between two [mln::p\\_faces](#)'s.*

- `template<typename G, typename F >`  
`bool operator== (const p_vertices< G, F > &lhs, const p_vertices< G, F > &rhs)`  
*Comparison between two `mln::p_vertices`'s.*
- `template<typename I, typename F >`  
`image_if< I, F > operator| (Image< I > &ima, const Function_v2b< F > &f)`  
*`ima` | `f` creates an `image_if` with the image `ima` and the function `f`.*
- `template<typename V, typename G, typename P >`  
`edge_image< P, V, G > operator| (const fun::i2v::array< V > &edge_values, const p_edges< G, fun::i2v::array< P > > &pe)`  
*Construct a edge image from a `fun::i2v::array` and a `p_edges`.*
- `template<typename S, typename F >`  
`p_if< S, F > operator| (const Site_Set< S > &s, const Function_v2b< F > &f)`  
*Restrict a site set `s` to points that verify `f`.*
- `template<typename F, typename S >`  
`pw::image< F, S > operator| (const Function_v2v< F > &f, const Site_Set< S > &ps)`  
*Construct an image from a function and a site set.*
- `template<typename I, typename F >`  
`image_if< const I, F > operator| (const Image< I > &ima, const Function_v2b< F > &f)`  
*`ima` | `f` creates an `image_if` with the image `ima` and the function `f`.*
- `template<typename V, typename G, typename P >`  
`vertex_image< P, V, G > operator| (const fun::i2v::array< V > &vertex_values, const p_vertices< G, fun::i2v::array< P > > &pv)`  
*Construct a vertex image from a `fun::i2v::array` and a `p_vertices`.*
- `template<typename I >`  
`const internal::primary_type< I >::ret & primary (const Image< I > &input)`  
*FIXME: Doc!*
- `template<typename S, typename F >`  
`p_transformed< S, F > ptransform (const Site_Set< S > &s, const Function_v2v< F > &f)`  
*Transform a site set `s` through the function `f`.*
- `const window2d & win_c4p ()`  
*4-connectivity window on the 2D grid, including the center.*
- `const window3d & win_c4p_3d ()`  
*4-connectivity window on the 3D grid, including the center.*
- `const window2d & win_c8p ()`  
*8-connectivity window on the 2D grid, including the center.*
- `const window3d & win_c8p_3d ()`  
*8-connectivity window on the 3D grid, including the center.*

- `template<unsigned N, unsigned D, typename P >`  
`bool operator== (const faces\_psite< N, D, P > &lhs, const faces\_psite< N, D, P > &rhs)`  
*Comparison of two instances of [mln::faces\\_psite](#).*
- `template<unsigned N, unsigned D, typename P >`  
`bool operator!= (const faces\_psite< N, D, P > &lhs, const faces\_psite< N, D, P > &rhs)`  
*Is lhs equal to rhs?*
- `template<unsigned N, unsigned D, typename P >`  
`bool operator< (const faces\_psite< N, D, P > &lhs, const faces\_psite< N, D, P > &rhs)`  
*Is lhs "less" than rhs?*
  
- `template<typename T >`  
`mln\_exact (T)*exact(T *ptr)`  
*Exact cast routine for mln objects.*
  
- `template<unsigned D, typename G >`  
`bool operator== (const complex\_psite< D, G > &lhs, const complex\_psite< D, G > &rhs)`  
*Comparison of two instances of [mln::complex\\_psite](#).*
- `template<unsigned D, typename G >`  
`bool operator!= (const complex\_psite< D, G > &lhs, const complex\_psite< D, G > &rhs)`  
*Is lhs not equal to rhs?*
- `template<unsigned D, typename G >`  
`bool operator< (const complex\_psite< D, G > &lhs, const complex\_psite< D, G > &rhs)`  
*Is lhs "less" than rhs?*

## Variables

- `const dpoint1d before = dpoint1d( -1 )`  
*Definition of a shortcut for delta point in 1d.*
- `const dpoint2d up = dpoint2d( -1, 0 )`  
*Definition of a shortcut for delta point in 2d.*
- `const dpoint3d sagittal\_dec = dpoint3d( 0, 0, -1)`  
*Definition of a shortcut for delta point in 3d.*

### 9.1.1 Detailed Description

[mln/convert/to\\_image.hh](#) This implementation is not an usual heap, it allows to set an error rate so that some nodes may be "corrupted".

Generic class for hierarchical queues.

Merge with [mln/topo/skeleton/is\\_simple\\_point.hh](#) The fastest version give an approximation of the result.

The generic dual input tree algorithm for high quantized image.

The dual input tree algorithm specialized for low quantized image.

[mln/linear/convolve\\_directional.hh](#)

Read AVS header from a file.

Define a function which aborts a process in io module.

Forward declaration.

[mln/core/def/all.hh](#)

The namespace mln corresponds to the Milena (mini-Olena) project.

This accumulator uses an [mln::util::pix](#) (pixel) to update the reference level, area and volume information of the component.

The class mln/accu/volume is not a general-purpose accumulator; it is used to implement volume-based connected filters.

#### See also

[mln::morpho::closing::volume](#)  
[mln::morpho::opening::volume](#)

The functor should provide the following methods:

- `template <typename g>=""> void init(const Graph<G>& g)` Will be called at the beginning.
- `bool to_be_treated(unsigned id)` Return whether this vertex has already been marked or if it may be a component representative.
- `void new_component_from_vertex(unsigned id)` will be called for the first vertex encountered for each component.
- `void process_vertex(unsigned id)` Will be called for each vertex queued.
- `bool to_be_queued(unsigned id)` Return whether this vertex has already been marked or if it can be added to the current component.
- `void added_to_queue(unsigned id)` Will be called for every vertex encountered in each component, except the first one.
- `void next_component()` Will be called after all vertices from a component have been treated.
- `void final()` Will be called at the end;

Conversions to [mln::Image](#).

FIXME: Re-write this description.

The contents of mln mimics the contents of the olenia project but in a simplified way. Some classes have the same name in both projects and roughly have the same behavior.

#### Warning

The Milena project is independent from the Olena project; the user has to choose between both the project she wants to work with.

File that includes all core definitions.

The set of operators defined in this file is:

```

l += r : l = l + r, -> l&
l -= r : l = l - r, -> l&
l *= r : l = l * r, -> l&
l /= r : l = l / r, -> l&
l %= r : l = l % r, -> l&

+ r    : -> r
- r    : -> (0 - r)

l ++   : t = l, ++l, -> t
l --   : t = l, --l, -> t

++ r   : r += 1, -> r&
-- r   : r -= 1, -> r&

l != r : -> ! (l == r)

l > r  : -> (r < l)
l >= r : -> (r <= l)
l <= r : -> ! (r < l)    warning: re-define when partial ordering

```

As a consequence, the set of operators to be defined along with a client class is:

```

l + r
l - r
l * r
l / r

l == r

l < r
l <= r in case of partial ordering

```

Convolution by a line-shaped (directional) kernel.

This implementation is based on P. Salembier algorithm using hierachical queues. This implies a low-quantized input image so that the number of queues is limited.

TODO: Think about how to extend f domain in a more generic way. The actual implementation doubles the size of the first dimension. It implies a boxed domain.

TODO: Use the less functor. The actual implementation is for max-tree.

TODO: During the canonization pass, we build the tree site set from the sorted site set of f, so that we compute twice f histogram (can be avoided).

This implementation is based on tarjan's union method, so that image quantization does not impact on the computation time.

TODO: Think about how to extend f domain in a more generic way. The actual implementation doubles the size of the first dimension. It implies a boxed domain.

TODO: Use the less functor. The actual implementation is for max-tree.

Do we want it to be exact?

Hierarchical queues are often used with connected operators (P. Salembier's max tree algorithm relies on these queues). To be efficient, the hiererachy is a static array and each are preallocated using an histogram.

FIXME: consider hqueues as a site set ?

A "corrupted node" means that its correct order is not totally preserved for performance reasons. Of course, it will have an impact on the returned values. As a result, be ware of not using this data structure if the element order is relevant for to you.

A corruption threshold can be passed to the constructor. This threshold means that if nodes have a rank higher than this threshold they can be "corrupted" and therefore their rank can be reduced. Tuning this threshold may have an impact on the structure entropy thus on the returned values order. It may also have an impact on the performance.

More implementation details are available in: "The soft heap: an approximate priority queue with optimal error rate", Bernard Chazelle, JACM, 2000.

URL: <http://www.cs.princeton.edu/~chazelle/pubs/sheap.pdf>

## 9.1.2 Typedef Documentation

### 9.1.2.1 `typedef mln::complex_image<1, mln::discrete_plane_1complex_geometry, bool> mln::bin_1complex_image2d`

Type alias for a binary image based on a 1-complex, where 0-faces are located at discrete (integer) 2-dimensional points.

Definition at line 53 of file `mln/core/alias/complex_image.hh`.

### 9.1.2.2 `typedef mln::complex_image<2, mln::space_2complex_geometry, bool> mln::bin_2complex_image3df`

Type alias for a binary image based on a 2-complex, where 0-faces are located at floating-point 3-dimensional points.

Definition at line 85 of file `mln/core/alias/complex_image.hh`.

### 9.1.2.3 `typedef box<mln::point1d> mln::box1d`

Type alias for a box defined on the 1D square grid with integer coordinates.

#### See also

`mln::win::rectangle1d`.

Definition at line 47 of file `core/alias/box1d.hh`.

### 9.1.2.4 `typedef box<mln::point2d> mln::box2d`

Type alias for a box defined on the 2D square grid with integer coordinates.

#### See also

`mln::win::rectangle2d`.

Definition at line 45 of file `core/alias/box2d.hh`.

**9.1.2.5 typedef box<point2d\_h> mln::box2d\_h**

FIXME.

Definition at line 47 of file core/alias/box2d\_h.hh.

**9.1.2.6 typedef box<point3d> mln::box3d**

Type alias for a box defined on the 3D square grid with integer coordinates.

**See also**

`mln::win::rectangle3d`.

Definition at line 45 of file core/alias/box3d.hh.

**9.1.2.7 typedef mln::geom::complex\_geometry<1, point2d> mln::discrete\_plane\_1complex\_geometry**

Type alias for the geometry of a 1-complex (e.g., a graph) located in a discrete 2-dimensional plane (with integer coordinates).

Definition at line 45 of file core/alias/complex\_geometry.hh.

**9.1.2.8 typedef mln::geom::complex\_geometry<2, point2d> mln::discrete\_plane\_2complex\_geometry**

Type alias for the geometry of a 2-complex located in a discrete 2-dimensional plane (with integer coordinates).

Definition at line 50 of file core/alias/complex\_geometry.hh.

**9.1.2.9 typedef dpoint<mln::grid::tick, def::coord> mln::dpoint1d**

Type alias for a delta-point defined on the 1D square grid with integer coordinates.

Definition at line 44 of file dpoint1d.hh.

**9.1.2.10 typedef dpoint<mln::grid::square, mln::def::coord> mln::dpoint2d**

Type alias for a delta-point defined on the 2D square grid with integer coordinates.

Definition at line 44 of file dpoint2d.hh.

**9.1.2.11 typedef dpoint<mln::grid::hexa, def::coord> mln::dpoint2d\_h**

Type alias for a delta-point defined on the 2D square grid with integer coordinates.

Definition at line 45 of file core/alias/dpoint2d\_h.hh.



**9.1.2.12 typedef dpoint<mln::grid::cube, def::coord> mln::dpoint3d**

Type alias for a delta-point defined on the 3D square grid with integer coordinates.

Definition at line 43 of file dpoint3d.hh.

**9.1.2.13 typedef mln::complex\_image<2, mln::space\_2complex\_geometry, float>  
mln::float\_2complex\_image3df**

Type alias for a floating-point image based on a 2-complex, where 0-faces are located at floating-point 3-dimensional points.

Definition at line 106 of file mln/core/alias/complex\_image.hh.

**9.1.2.14 typedef mln::complex\_image<1, mln::discrete\_plane\_1complex\_geometry,  
mln::value::int\_u8> mln::int\_u8\_1complex\_image2d**

Type alias for an 8-bit gray-level image based on a 1-complex, where 0-faces are located at discrete (integer) 2-dimensional points.

Definition at line 61 of file mln/core/alias/complex\_image.hh.

**9.1.2.15 typedef mln::complex\_image<2, mln::discrete\_plane\_2complex\_geometry,  
mln::value::int\_u8> mln::int\_u8\_2complex\_image2d**

Type alias for an 8-bit gray-level image based on a 2-complex, where 0-faces are located at discrete (integer) 2-dimensional points.

Definition at line 74 of file mln/core/alias/complex\_image.hh.

**9.1.2.16 typedef mln::complex\_image<2, mln::space\_2complex\_geometry, mln::value::int\_u8>  
mln::int\_u8\_2complex\_image3df**

Type alias for an 8-bit gray-level image based on a 2-complex, where 0-faces are located at floating-point 3-dimensional points.

Definition at line 92 of file mln/core/alias/complex\_image.hh.

**9.1.2.17 typedef p\_run<point2d> mln::p\_run2d**

Type alias for a run of 2d points.

Definition at line 42 of file p\_run2d.hh.

**9.1.2.18 typedef p\_set\_of<p\_run2d> mln::p\_runs2d**

Type alias for a set of runs of 2d points.

Definition at line 42 of file p\_runs2d.hh.

**9.1.2.19 typedef point< grid::tick, def::coord > mln::point1d**

Type alias for a point defined on the 1D ruler with integer coordinates.

Definition at line 45 of file point1d.hh.

**9.1.2.20 typedef point<grid::tick, def::coordf> mln::point1df**

Type alias for a point defined on the 1D ruler with floating-point coordinates.

Definition at line 49 of file point1d.hh.

**9.1.2.21 typedef point< grid::square, def::coord > mln::point2d**

Type alias for a point defined on the 2D square grid with integer coordinates.

Definition at line 45 of file point2d.hh.

**9.1.2.22 typedef point< grid::hexa, def::coord > mln::point2d\_h**

Type alias for a point defined on the 2D hexagonal grid with integer coordinates.

Definition at line 43 of file core/alias/point2d\_h.hh.

**9.1.2.23 typedef point<mln::grid::square, mln::def::coordf> mln::point2df**

Type alias for a point defined on the 2D square grid with floating-point coordinates.

Definition at line 49 of file point2d.hh.

**9.1.2.24 typedef point< grid::cube, def::coord > mln::point3d**

Type alias for a point defined on the 3D square grid with integer coordinates.

Definition at line 44 of file point3d.hh.

**9.1.2.25 typedef point<grid::cube, def::coordf> mln::point3df**

Type alias for a point defined on the 3D square grid with floating-point coordinates.

Definition at line 48 of file point3d.hh.

**9.1.2.26 typedef mln::complex\_image<2, mln::space\_2complex\_geometry, mln::value::rgb8>  
mln::rgb8\_2complex\_image3df**

Type alias for a (3x8-bit) RGB image based on a 2-complex, where 0-faces are located at floating-point 3-dimensional points.

Definition at line 113 of file mln/core/alias/complex\_image.hh.

**9.1.2.27 typedef mln::geom::complex\_geometry<2, point3df> mln::space\_2complex\_geometry**

Type alias for the geometry of a 2-complex located in a 3-dimensional space (with floating-point coordinates).

Definition at line 54 of file core/alias/complex\_geometry.hh.

**9.1.2.28 typedef mln::complex\_image<2, mln::space\_2complex\_geometry, unsigned> mln::unsigned\_2complex\_image3df**

Type alias for a gray-level image based on a 2-complex, where 0-faces are located at floating-point 3-dimensional points.

Definition at line 99 of file mln/core/alias/complex\_image.hh.

**9.1.2.29 typedef algebra::vec<2u,double> mln::vec2d\_d**

2D vector with double coordinates.

Definition at line 43 of file vec2d.hh.

**9.1.2.30 typedef algebra::vec<2u,float> mln::vec2d\_f**

2D vector with float coordinates.

Definition at line 40 of file vec2d.hh.

**9.1.2.31 typedef algebra::vec<3u,double> mln::vec3d\_d**

3D vector with double coordinates.

Definition at line 43 of file vec3d.hh.

**9.1.2.32 typedef algebra::vec<3u,float> mln::vec3d\_f**

3D vector with float coordinates.

Definition at line 40 of file vec3d.hh.

**9.1.2.33 typedef w\_window<dpoint1d, float> mln::w\_window1d\_float**

Type alias for a [w\\_window](#) with arbitrary shape, defined on the 1D grid (with integer coordinates) and whose weights are floating values.

Definition at line 45 of file w\_window1d\_float.hh.

**9.1.2.34 typedef w\_window<dpoint1d, int> mln::w\_window1d\_int**

Type alias for a [w\\_window](#) with arbitrary shape, defined on the 1D grid (with integer coordinates) and whose weights are integers.

Definition at line 45 of file core/alias/w\_window1d\_int.hh.

**9.1.2.35** `typedef w_window<dpoint2d, float> mln::w_window2d_float`

Type alias for a `w_window` with arbitrary shape, defined on the 2D square grid (with integer coordinates) and whose weights are floating values.

Definition at line 45 of file `w_window2d_float.hh`.

**9.1.2.36** `typedef w_window<dpoint2d, int> mln::w_window2d_int`

Type alias for a `w_window` with arbitrary shape, defined on the 2D square grid (with integer coordinates) and whose weights are integers.

Definition at line 45 of file `core/alias/w_window2d_int.hh`.

**9.1.2.37** `typedef w_window<dpoint3d, float> mln::w_window3d_float`

Type alias for a `w_window` with arbitrary shape, defined on the 3D grid (with integer coordinates) and whose weights are floating values.

Definition at line 45 of file `w_window3d_float.hh`.

**9.1.2.38** `typedef w_window<dpoint3d, int> mln::w_window3d_int`

Type alias for a `w_window` with arbitrary shape, defined on the 3D grid (with integer coordinates) and whose weights are integers.

Definition at line 45 of file `core/alias/w_window3d_int.hh`.

**9.1.3 Function Documentation****9.1.3.1** `template<typename I> I::site mln::a_point_of( const Image< I > & ima ) [inline]`

Give a point of an image.

Definition at line 51 of file `a_point_of.hh`.

**9.1.3.2** `template<typename I, typename F> p2p_image< I, F > mln::apply_p2p( Image< I > & ima, const Function_v2v< F > & f ) [inline]`

FIXME: Doc!

Definition at line 242 of file `p2p_image.hh`.

Referenced by `mln::debug::mosaic()`, and `mln::debug::slices_2d()`.

**9.1.3.3** `template<typename I, typename F> p2p_image< const I, F > mln::apply_p2p( const Image< I > & ima, const Function_v2v< F > & f ) [inline]`

FIXME: Doc!

Definition at line 256 of file `p2p_image.hh`.

**9.1.3.4** `template<typename T2 , typename T1 > fun::x2x::composed< T2, T1 > mln::compose ( T2 f, T1 g ) [inline]`

Do a composition of two transformations.

#### Parameters

[in] *f* The second transformation.

[in] *g* The first transformation.

#### Returns

The composed transformation fog.

Definition at line 306 of file composed.hh.

References `compose()`.

Referenced by `compose()`, and `mln::geom::rotate()`.

**9.1.3.5** `template<typename I > mln::trait::concrete< I >::ret mln::duplicate ( const Image< I > & model ) [inline]`

Duplicate the image `model` with the values of the image data.

#### Parameters

[in] *model* The image to be duplicated.

#### Returns

The duplicate.

#### Precondition

`model.is_valid`

Definition at line 56 of file core/routine/duplicate.hh.

References `mln::data::fill()`, and `initialize()`.

Referenced by `mln::registration::icp()`, `mln::plain< I >::operator I()`, `mln::geom::rotate()`, `mln::geom::impl::seeds2tiling()`, and `mln::labeling::superpose()`.

**9.1.3.6** `template<typename I , typename F > extension_fun< const I, F > mln::extend ( const Image< I > & ima, const Function_v2v< F > & fun ) [inline]`

Routines for domain extension with a function.

Definition at line 140 of file extend.hh.

Referenced by `mln::geom::translate()`.

**9.1.3.7** `template<typename I , typename J > extension_ima< const I, const J > mln::extend ( const Image< I > & ima, const Image< J > & ext )`

Routines for domain extension with an image.

Definition at line 162 of file extend.hh.

### 9.1.3.8 `template<typename I> extension_val< const I > mln::extend ( const Image< I > & ima, const typename I::value & val ) [inline]`

Routines for domain extension with a value.

Definition at line 175 of file extend.hh.

### 9.1.3.9 `bool mln::implies ( bool lexpr, bool rexpr ) [inline]`

Implication.

Definition at line 68 of file contract.hh.

Referenced by `mln::p_line2d::is_valid()`.

### 9.1.3.10 `template<typename I, typename J> void mln::initialize ( Image< I > & target, const Image< J > & model ) [inline]`

Initialize the image `target` with data extracted from image `model`.

#### Parameters

[in, out] *target* The image to be initialized.

[in] *model* The image to provide data for the initialization.

#### Precondition

(not `target.is_valid()` and `model.is_valid()`)

Definition at line 56 of file initialize.hh.

Referenced by `duplicate()`, `mln::histo::equalize()`, `mln::morpho::tree::filter::filter()`, `mln::linear::gaussian()`, `mln::linear::gaussian_1st_derivative()`, `mln::linear::gaussian_2nd_derivative()`, `mln::graph::labeling()`, `mln::io::magick::load()`, `mln::io::dicom::load()`, `make_debug_graph_image()`, `mln::morpho::tree::filter::max()`, `mln::morpho::meyer_wst()`, `mln::morpho::tree::filter::min()`, `mln::arith::min()`, `mln::arith::minus()`, `mln::arith::plus()`, `mln::arith::revert()`, `mln::geom::rotate()`, `mln::data::impl::stretch()`, `mln::morpho::watershed::topological()`, and `mln::data::impl::generic::transform()`.

### 9.1.3.11 `template<typename P> box< P > mln::larger_than ( const box< P > a, const box< P > > b ) [inline]`

Return the minimum box including box `a` and box `b`.

Definition at line 362 of file core/site\_set/box.hh.

References `mln::box< P >::pmax()`, and `mln::box< P >::pmin()`.

### 9.1.3.12 `template<typename I, typename V, typename E> image2d<typename I::value> mln::make_debug_graph_image ( const I & input, const V & ima_v, const E & ima_e, const value::rgb8 & bg ) [inline]`

Draw a graph.

Definition at line 123 of file demos/graph/region\_adjacency\_graph.cc.

References `mln::box< P >::crop_wrt()`, `mln::image2d< T >::domain()`, `mln::debug::draw_graph()`, `mln::data::fill()`, `mln::literal::green`, `initialize()`, and `mln::convert::to()`.

### 9.1.3.13 `template<typename T > mln::mln_exact ( T ) [inline]`

Exact cast routine for mln objects.

This set of routines can be used to downcast an object towards its exact type. The only argument, respectively `ptr` or `ref`, should be an [mln::Object](#).

The parameter `E` is the exact type of the object.

#### Returns

The return follows the nature of the argument (either a pointer or a reference, const or not).

Definition at line 92 of file `routine/exact.hh`.

Referenced by `mln::geom::rotate()`, `mln::Accumulator< E >::take_as_init()`, `mln::Accumulator< E >::take_n_times()`, `mln::convert::to()`, and `mln::geom::translate()`.

### 9.1.3.14 `mln::mln_gen_complex_neighborhood ( complex_higher_neighborhood , complex_higher_window )`

[Neighborhood](#) centered on an n-face of complex returning its adjacent (n+1)-faces.

### 9.1.3.15 `mln::mln_gen_complex_neighborhood ( complex_lower_higher_neighborhood , complex_lower_higher_window )`

[Neighborhood](#) centered on an n-face of complex returning its adjacent (n-1)-faces and (n+1)-faces.

### 9.1.3.16 `mln::mln_gen_complex_neighborhood ( complex_lower_dim_connected_n_face_neighborhood , complex_lower_dim_connected_n_face_window )`

[Neighborhood](#) centered on an n-face of complex returning the n-faces sharing an (n-1)-face with the center n-face.

### 9.1.3.17 `mln::mln_gen_complex_neighborhood ( complex_higher_dim_connected_n_face_neighborhood , complex_higher_dim_connected_n_face_window )`

[Neighborhood](#) centered on an n-face of complex returning the n-faces sharing an (n+1)-face with the center n-face.

### 9.1.3.18 `mln::mln_gen_complex_neighborhood ( complex_m_face_neighborhood , complex_m_face_window )`

[Neighborhood](#) centered on an n-face of complex returning the m-faces transitively adjacent to this center n-face.

**9.1.3.19** `mln::mln_gen_complex_neighborhood ( complex_lower_neighborhood , complex_lower_window )`

[Neighborhood](#) centered on an n-face of complex returning its adjacent (n-1)-faces.

**9.1.3.20** `mln::mln_gen_complex_window ( complex_lower_window , topo::adj_lower_face_fwd_iter , topo::adj_lower_face_bkd_iter )`

[Window](#) centered on an n-face of complex returning its adjacent (n-1)-faces.

**9.1.3.21** `mln::mln_gen_complex_window ( complex_higher_window , topo::adj_higher_face_fwd_iter , topo::adj_higher_face_bkd_iter )`

[Window](#) centered on an n-face of complex returning its adjacent (n+1)-faces.

**9.1.3.22** `mln::mln_gen_complex_window ( complex_lower_higher_window , topo::adj_lower_higher_face_fwd_iter , topo::adj_lower_higher_face_bkd_iter )`

[Window](#) centered on an n-face of complex returning its adjacent (n-1)-faces and (n+1)-faces.

**9.1.3.23** `mln::mln_gen_complex_window ( complex_lower_dim_connected_n_face_window , topo::adj_lower_dim_connected_n_face_fwd_iter , topo::adj_lower_dim_connected_n_face_bkd_iter )`

[Window](#) centered on an n-face of complex returning the n-faces sharing an (n-1)-face with the center n-face.

**9.1.3.24** `mln::mln_gen_complex_window ( complex_higher_dim_connected_n_face_window , topo::adj_higher_dim_connected_n_face_fwd_iter , topo::adj_higher_dim_connected_n_face_bkd_iter )`

[Window](#) centered on an n-face of complex returning the n-faces sharing an (n+1)-face with the center n-face.

**9.1.3.25** `mln::mln_gen_complex_window ( complex_m_face_window , topo::adj_m_face_fwd_iter , topo::adj_m_face_bkd_iter )`

[Window](#) centered on an n-face of complex returning the m-faces transitively adjacent to this center n-face.

**9.1.3.26** `mln::mln_gen_complex_window_p ( complex_lower_window_p , topo::adj_lower_face_fwd_iter , topo::adj_lower_face_bkd_iter )`

[Window](#) centered on an n-face of complex returning its adjacent (n-1)-faces as well as the center n-face.

**9.1.3.27** `mln::mln_gen_complex_window_p ( complex_higher_window_p , topo::adj_higher_face_fwd_iter , topo::adj_higher_face_bkd_iter )`

[Window](#) centered on an n-face of complex returning its adjacent (n+1)-faces as well as the center n-face.



**9.1.3.28** `mln::mln_gen_complex_window_p ( complex_lower_higher_window_p ,  
topo::adj_lower_higher_face_fwd_iter , topo::adj_lower_higher_face_bkd_iter )`

**Window** centered on an n-face of complex returning its adjacent (n-1)-faces and (n+1)-faces as well as the center n-face.

**9.1.3.29** `mln::mln_gen_complex_window_p ( complex_higher_dim_connected_  
n_face_window_p , topo::adj_higher_dim_connected_n_face_fwd_iter ,  
topo::adj_higher_dim_connected_n_face_bkd_iter )`

**Window** centered on an n-face of complex returning the n-faces sharing an (n+1)-face with the center n-face, as well as this center n-face.

**9.1.3.30** `mln::mln_gen_complex_window_p ( complex_lower_dim_connected_  
n_face_window_p , topo::adj_lower_dim_connected_n_face_fwd_iter ,  
topo::adj_lower_dim_connected_n_face_bkd_iter )`

**Window** centered on an n-face of complex returning the n-faces sharing an (n-1)-face with the center n-face, as well as this center n-face.

**9.1.3.31** `mln::mln_gen_complex_window_p ( complex_m_face_window_p ,  
topo::adj_m_face_fwd_iter , topo::adj_m_face_bkd_iter )`

**Window** centered on an n-face of complex returning the m-faces transitively adjacent to this center n-face, as well as this center n-face.

**9.1.3.32** `template<typename W1 , typename W2 > mln::mln_regular ( W1 ) const [inline]`

Set difference between a couple of windows `win1` and `win2`.

Inter a window `win` with a delta-point `dp`.

It just calls `mln::win::diff`.

**9.1.3.33** `template<typename O1 , typename O2 > mln::mln_trait_op_geq ( O1 , O2 )`

General definition of the "greater than or equal to" operator.

The "greater than or equal to" operator is here defined for every Milena objects. It relies on the definition of the "less than or equal to" operator. It returns "`rhs <= lhs`".

#### Warning

There shall not be any other definition of this operator in Milena when applying on a couple of `mln::Object`.

**9.1.3.34** `template<typename O1 , typename O2 > mln::mln_trait_op_greater ( O1 , O2 ) const`

General definition of the "greater than" operator.

The "greater than" operator is here defined for every milena objects. It relies on the definition of the "less than" operator. It returns "`rhs < lhs`".

**Warning**

There shall not be any other definition of this operator in Milena when applying on a couple of [mln::Object](#).

**9.1.3.35** `template<typename O1 , typename O2 > mln::mln_trait_op_leq ( O1 , O2 )`

Default definition of the "less than or equal to" operator.

A default version of the "less than or equal to" operator is defined for every Milena objects. It relies on the definition of the "less than" operator. It returns "not (rhs < lhs)".

**Warning**

In the case of partial ordering between objects, this operator has to be re-defined.

**9.1.3.36** `template<typename O1 , typename O2 > mln::mln_trait_op_neq ( O1 , O2 )`  
`[inline]`**Initial value:**

```
(const Object<O1>& lhs, const Object<O2>& rhs)
{
    return ! (exact(lhs) == exact(rhs));
}

template <typename O1, typename O2>
inline
mln_trait_op_greater(O1, O2)
operator>(const Object<O1>& lhs, const Object<O2>& rhs)
{
    return exact(rhs) < exact(lhs);
}

template <typename O1
```

General definition of the "not equal to" operator.

The "not equal to" operator is here defined for every milena objects. It relies on the definition of the "equal to" operator. It returns "not (lhs == rhs)".

**Warning**

There shall not be any other definition of this operator in Milena when applying on a couple of [mln::Object](#).

**9.1.3.37** `template<unsigned D, typename G > bool mln::operator!=( const complex_psite< D, G > & lhs, const complex_psite< D, G > & rhs )`

Is *lhs* not equal to *rhs*?

**Precondition**

Arguments *lhs* and *rhs* must belong to the same [mln::p\\_complex](#).

Definition at line 353 of file `complex_psite.hh`.

References `mln::complex_psite< D, G >::face()`, and `mln::complex_psite< D, G >::site_set()`.

**9.1.3.38** `template<unsigned N, unsigned D, typename P > bool mln::operator!= ( const faces_psite< N, D, P > & lhs, const faces_psite< N, D, P > & rhs )`

Is *lhs* equal to *rhs*?

#### Precondition

Arguments *lhs* and *rhs* must belong to the same `mln::complex`.

Definition at line 336 of file `faces_psite.hh`.

References `mln::faces_psite< N, D, P >::face()`, and `mln::faces_psite< N, D, P >::site_set()`.

**9.1.3.39** `template<typename P, typename S > P mln::operator* ( const Gpoint< P > & p, const value::scalar_< S > & s ) [inline]`

Multiply a point *p* by a scalar *s*.

Definition at line 405 of file `gpoint.hh`.

**9.1.3.40** `template<typename S > S & mln::operator++ ( value::Scalar< S > & rhs ) [inline]`

Pre-incrementation for any scalar type.

Definition at line 77 of file `concept/scalar.hh`.

References `mln::literal::one`.

**9.1.3.41** `template<typename N1, typename N2 > N2 neighb< typename N1::window::regular > mln::operator- ( const Neighborhood< N1 > & nbh1, const Neighborhood< N2 > & nbh2 )`

Set difference between a couple of neighborhoods *nbh1* and *nbh2*.

It just calls `mln::win::diff`.

Definition at line 155 of file `win/diff.hh`.

References `mln::win::diff()`.

**9.1.3.42** `template<typename P, typename D > P mln::operator- ( const Gpoint< P > & p, const Gdpoint< D > & dp ) [inline]`

Subtract a delta-point *dp* to a grid point *p*.

#### Parameters

[in] *p* A grid point.

[in] *dp* A delta-point.

The type of *dp* has to compatible with the type of *p*.

#### Returns

A point (temporary object).

**See also**

[mln::Gdpoint](#)  
[mln::Gdpoint](#)

Definition at line 395 of file `gpoint.hh`.

**9.1.3.43** `template<typename S > S & mln::operator-- ( value::Scalar< S > & rhs ) [inline]`

Pre-decrementation for any scalar type.

Definition at line 85 of file `concept/scalar.hh`.

References `mln::literal::one`.

**9.1.3.44** `template<unsigned D, typename G > bool mln::operator< ( const complex_psite< D, G > & lhs, const complex_psite< D, G > & rhs )`

Is *lhs* “less” than *rhs*?

This comparison is required by algorithms sorting psites.

**Precondition**

Arguments *lhs* and *rhs* must belong to the same `mln::p_complex`.

Definition at line 362 of file `complex_psite.hh`.

**9.1.3.45** `template<unsigned N, unsigned D, typename P > bool mln::operator< ( const faces_psite< N, D, P > & lhs, const faces_psite< N, D, P > & rhs )`

Is *lhs* “less” than *rhs*?

This comparison is required by algorithms sorting psites.

**Precondition**

Arguments *lhs* and *rhs* must belong to the same `mln::complex`.

Definition at line 345 of file `faces_psite.hh`.

**9.1.3.46** `template<typename L , typename R > bool mln::operator< ( const Image< L > & lhs, const Image< R > & rhs ) [inline]`

Point-wise test if the pixel values of *lhs* are point-wise less than the pixel values of *rhs*.

**Parameters**

[in] *lhs* A first image.

[in] *rhs* A second image.

**Precondition**

`lhs.domain == rhs.domain`

Definition at line 107 of file `compare.hh`.

References `mln::test::predicate()`.

**9.1.3.47** `template<typename I, typename G, typename W> std::ostream & mln::operator<< ( std::ostream & ostr, const complex_window_bkd_piter< I, G, W > & p ) [inline]`

Print an [mln::complex\\_window\\_bkd\\_piter](#).

Definition at line 401 of file `complex_window_piter.hh`.

**9.1.3.48** `template<typename I, typename G, typename W> std::ostream & mln::operator<< ( std::ostream & ostr, const complex_window_fwd_piter< I, G, W > & p ) [inline]`

Print an [mln::complex\\_window\\_fwd\\_piter](#).

Definition at line 292 of file `complex_window_piter.hh`.

**9.1.3.49** `template<typename I, typename G, typename N> std::ostream & mln::operator<< ( std::ostream & ostr, const complex_neighborhood_fwd_piter< I, G, N > & p ) [inline]`

Print an [mln::complex\\_neighborhood\\_fwd\\_piter](#).

Definition at line 292 of file `complex_neighborhood_piter.hh`.

**9.1.3.50** `template<typename I, typename G, typename N> std::ostream & mln::operator<< ( std::ostream & ostr, const complex_neighborhood_bkd_piter< I, G, N > & p ) [inline]`

Print an [mln::complex\\_neighborhood\\_bkd\\_piter](#).

Definition at line 399 of file `complex_neighborhood_piter.hh`.

**9.1.3.51** `template<unsigned D, typename G> bool mln::operator<= ( const p_complex< D, G > & lhs, const p_complex< D, G > & rhs )`

Inclusion of a [mln::p\\_complex](#) in another one.

This inclusion relation is very strict for the moment, since our infrastructure for complexes is simple: a [mln::p\\_complex](#) is included in another one if their are equal.

Definition at line 330 of file `p_complex.hh`.

**9.1.3.52** `template<typename G, typename F> bool mln::operator<= ( const p_edges< G, F > & lhs, const p_edges< G, F > & rhs )`

Inclusion of a [mln::p\\_edges](#) in another one.

Definition at line 338 of file `p_edges.hh`.

**9.1.3.53** `template<typename L, typename R> bool mln::operator<= ( const Image< L > & lhs, const Image< R > & rhs ) [inline]`

Point-wise test if the pixel values of `lhs` are point-wise less than or equal to the pixel values of `rhs`.

**Parameters**

[in] *lhs* A first image.

[in] *rhs* A second image.

**Precondition**

`lhs.domain == rhs.domain`

Definition at line 126 of file `compare.hh`.

References `mln::test::predicate()`.

### 9.1.3.54 `template<typename G , typename F > bool mln::operator<= ( const p_vertices< G, F > & lhs, const p_vertices< G, F > & rhs )`

Inclusion of a `mln::p_vertices` in another one.

This inclusion relation is very strict for the moment, since our infrastructure for graphs is simple: a `mln::p_vertices` is included in another one if their are equal.

Definition at line 399 of file `p_vertices.hh`.

### 9.1.3.55 `template<unsigned N, unsigned D, typename P > bool mln::operator<= ( const p_faces< N, D, P > & lhs, const p_faces< N, D, P > & rhs )`

Inclusion of a `mln::p_faces` in another one.

This inclusion relation is very strict for the moment, since our infrasturure for complexes is simple: a `mln::p_faces` is included in another one if their are equal.

Definition at line 287 of file `p_faces.hh`.

### 9.1.3.56 `template<unsigned N, unsigned D, typename P > bool mln::operator== ( const p_faces< N, D, P > & lhs, const p_faces< N, D, P > & rhs )`

Comparison between two `mln::p_faces`'s.

Two `mln::p_faces`'s are considered equal if they share the same complex.

Definition at line 278 of file `p_faces.hh`.

References `mln::p_faces< N, D, P >::cplx()`.

### 9.1.3.57 `template<unsigned N, unsigned D, typename P > bool mln::operator== ( const faces_psite< N, D, P > & lhs, const faces_psite< N, D, P > & rhs )`

Comparison of two instances of `mln::faces_psite`.

Is *lhs* equal to *rhs*?

**Precondition**

Arguments *lhs* and *rhs* must belong to the same `mln::complex`.

Definition at line 327 of file `faces_psite.hh`.

References `mln::faces_psite< N, D, P >::face()`, and `mln::faces_psite< N, D, P >::site_set()`.

**9.1.3.58** `template<typename L , typename R > bool mln::operator==( const Image< L > & lhs, const Image< R > & rhs ) [inline]`

Point-wise test if the pixel values of `lhs` are equal to the pixel values of `rhs`.

#### Parameters

[in] *lhs* A first image.

[in] *rhs* A second image.

#### Precondition

`lhs.domain == rhs.domain`

Definition at line 86 of file `compare.hh`.

References `mln::test::predicate()`.

**9.1.3.59** `template<unsigned D, typename G > bool mln::operator==( const p_complex< D, G > & lhs, const p_complex< D, G > & rhs )`

Comparison between two `mln::p_complex`'s.

Two `mln::p_complex`'s are considered equal if they share the same complex.

Definition at line 321 of file `p_complex.hh`.

References `mln::p_complex< D, G >::cplx()`.

**9.1.3.60** `template<unsigned D, typename G > bool mln::operator==( const complex_psite< D, G > & lhs, const complex_psite< D, G > & rhs )`

Comparison of two instances of `mln::complex_psite`.

Is *lhs* equal to *rhs*?

#### Precondition

Arguments *lhs* and *rhs* must belong to the same `mln::p_complex`.

Definition at line 344 of file `complex_psite.hh`.

References `mln::complex_psite< D, G >::face()`, and `mln::complex_psite< D, G >::site_set()`.

**9.1.3.61** `template<typename G , typename F > bool mln::operator==( const p_vertices< G, F > & lhs, const p_vertices< G, F > & rhs )`

Comparison between two `mln::p_vertices`'s.

Two `mln::p_vertices`'s are considered equal if they share the same graph.

Definition at line 392 of file `p_vertices.hh`.

References `mln::p_vertices< G, F >::graph()`.

**9.1.3.62** `template<typename G , typename F > bool mln::operator==( const p_edges< G, F > & lhs, const p_edges< G, F > & rhs )`

Comparison between two [mln::p\\_edges](#)'s.

Two [mln::p\\_edges](#)'s are considered equal if they share the same graph.

Definition at line 331 of file `p_edges.hh`.

References `mln::p_edges< G, F >::graph()`.

**9.1.3.63** `template<typename F , typename S > pw::image<F,S> mln::operator|( const Function_v2v< F > & f, const Site_Set< S > & ps )`

Construct an image from a function and a site set.

`image = function | site_set.`

Definition at line 138 of file `pw/image.hh`.

**9.1.3.64** `template<typename V , typename G , typename P > edge_image<P,V,G> mln::operator|( const fun::i2v::array< V > & edge_values, const p_edges< G, fun::i2v::array< P > > & pe )`

Construct a edge image from a `fun::i2v::array` and a [p\\_edges](#).

`image = fun::i2v::array | p_edges.`

Definition at line 217 of file `core/image/edge_image.hh`.

**9.1.3.65** `template<typename S , typename F > p_if<S, F> mln::operator|( const Site_Set< S > & s, const Function_v2b< F > & f )`

Restrict a site set `s` to points that verify `f`.

#### Parameters

[in] `s` A site set.

[in] `f` A function from point to Boolean.

#### Returns

A subset of points.

Definition at line 147 of file `p_if.hh`.

**9.1.3.66** `template<typename V , typename G , typename P > vertex_image<P,V,G> mln::operator|( const fun::i2v::array< V > & vertex_values, const p_vertices< G, fun::i2v::array< P > > & pv )`

Construct a vertex image from a `fun::i2v::array` and a [p\\_vertices](#).

`image = fun::i2v::array | p_vertices.`

Definition at line 215 of file `core/image/vertex_image.hh`.



**9.1.3.67** `template<typename I, typename F> image_if<I,F> mln::operator| ( Image< I > & ima, const Function_v2b< F > & f )`

`ima | f` creates an `image_if` with the image `ima` and the function `f`.

Definition at line 233 of file `image_if.hh`.

**9.1.3.68** `template<typename I, typename F> image_if<const I,F> mln::operator| ( const Image< I > & ima, const Function_v2b< F > & f )`

`ima | f` creates an `image_if` with the image `ima` and the function `f`.

Definition at line 242 of file `image_if.hh`.

**9.1.3.69** `template<typename I> const internal::primary_type< I >::ret & mln::primary ( const Image< I > & input ) [inline]`

FIXME: Doc!

Definition at line 129 of file `primary.hh`.

Referenced by `mln::border::resize()`.

**9.1.3.70** `template<typename S, typename F> p_transformed< S, F > mln::ptransform ( const Site_Set< S > & s, const Function_v2v< F > & f ) [inline]`

Transform a site set `s` through the function `f`.

#### Parameters

[in] `s` A site set.

[in] `f` A function from site to site.

#### Returns

The transformed site set.

Definition at line 145 of file `p_transformed.hh`.

## 9.1.4 Variable Documentation

**9.1.4.1** `const dpoint1d mln::before = dpoint1d( -1 )`

Definition of a shortcut for delta point in 1d.

Definition at line 73 of file `dpoint1d.hh`.

**9.1.4.2** `const dpoint3d mln::sagittal_dec = dpoint3d( 0, 0, -1 )`

Definition of a shortcut for delta point in 3d.

Definition at line 72 of file `dpoint3d.hh`.

### 9.1.4.3 `const dpoint2d mln::up = dpoint2d( -1, 0 )`

Definition of a shortcut for delta point in 2d.

Definition at line 76 of file `dpoint2d.hh`.

## 9.2 `mln::accu` Namespace Reference

Namespace of accumulators.

### Namespaces

- namespace [image](#)  
*Namespace of accumulator image routines.*
- namespace [impl](#)  
*Implementation namespace of accumulator namespace.*
- namespace [logic](#)  
*Namespace of logical accumulators.*
- namespace [math](#)  
*Namespace of mathematic accumulators.*
- namespace [shape](#)  
*Namespace of shape accumulators.*
- namespace [stat](#)  
*Namespace of statistical accumulators.*

### Classes

- struct [center](#)  
*Mass center accumulator.*
- struct [convolve](#)  
*Generic convolution accumulator class.*
- struct [count\\_adjacent\\_vertices](#)  
*Accumulator class counting the number of vertices adjacent to a set of `mln::p_edges_psite` (i.e., a set of edges).*
- struct [count\\_labels](#)  
*Count the number of different labels in an image.*
- struct [count\\_value](#)  
*Define an accumulator that counts the occurrence of a given value.*

- struct [histo](#)  
*Generic histogram class over a value set with type V.*
- struct [label\\_used](#)  
*References all the labels used.*
- struct [maj\\_h](#)  
*Compute the majority value.*
- struct [max\\_site](#)  
*Define an accumulator that computes the first site with the maximum value in an image.*
- struct [nil](#)  
*Define an accumulator that does nothing.*
- struct [p](#)  
*Generic p of accumulators.*
- struct [pair](#)  
*Generic pair of accumulators.*
- struct [rms](#)  
*Generic root mean square accumulator class.*
- struct [tuple](#)  
*Generic tuple of accumulators.*
- struct [val](#)  
*Generic val of accumulators.*

## Functions

- `template<typename A , typename I >`  
`A::result compute (const Accumulator< A > &a, const Image< I > &input)`  
*Make an accumulator compute the pixels of the image `input`.*
- `template<typename Meta_Accu , unsigned Dir, typename I , typename O >`  
`void line (const Image< I > &input, const typename I::site &p_start, unsigned len, unsigned half_length, Image< O > &output)`
- `template<typename A , typename I >`  
`mln\_meta\_accu\_result (A, util::pix< I >) compute(const Meta\_Accumulator< A > &a)`  
*Make an accumulator compute the pixels of the image `input`.*
- `template<typename A , typename I >`  
`void take (const Image< I > &input, Accumulator< A > &a)`  
*Make an accumulator take the pixels of the image `input`.*

## 9.2.1 Detailed Description

Namespace of accumulators.

## 9.2.2 Function Documentation

### 9.2.2.1 `template<typename A , typename I > A::result mln::accu::compute ( const Accumulator< A > & a, const Image< I > & input ) [inline]`

Make an accumulator compute the pixels of the image `input`.

#### Parameters

[in] *input* The input image.

[in] *a* An accumulator.

This routine runs:

```
a.take(make::pix(input, p));
```

on all pixels on the images.

#### Warning

This routine does not perform `a.init()`.

Definition at line 130 of file `accu/compute.hh`.

### 9.2.2.2 `template<typename Meta_Accu , unsigned Dir, typename I , typename O > void mln::accu::line ( const Image< I > & input, const typename I::site & p_start, unsigned len, unsigned half_length, Image< O > & output )`

Line an accumulator onto the pixel values of the image `input`.

#### Parameters

[in] *input* The input image.

[in] *p\_start* The starting site of the line.

[in] *len* The line length.

[in] *half\_length* The half length of the line.

[in, out] *output* The resulting image.

This routine runs:

```
tmp = a
```

```
tmp.init()
```

```
accu::take(input, tmp)
```

```
return tmp.to_result()
```

Definition at line 381 of file `accu/line.hh`.

**9.2.2.3** `template<typename A , typename I > mln::accu::mln_meta_accu_result ( A , util::pix< I > ) const [inline]`

Make an accumulator compute the pixels of the image `input`.

#### Parameters

[in] *input* The input image.

[in] *a* A meta accumulator.

This routine runs:

`a.take(make::pix(input, p));` on all pixels on the images.

#### Warning

This routine does not perform `a.init()`.

**9.2.2.4** `template<typename A , typename I > void mln::accu::take ( const Image< I > & input, Accumulator< A > & a ) [inline]`

Make an accumulator take the pixels of the image `input`.

#### Parameters

[in] *input* The input image.

[in, out] *a* The accumulator.

This routine runs:

for all `p` of `input`, `a.take( pix(input, p) )`

#### Warning

This routine does not perform `a.init()`.

Definition at line 87 of file `take.hh`.

## 9.3 mln::accu::image Namespace Reference

Namespace of accumulator image routines.

### 9.3.1 Detailed Description

Namespace of accumulator image routines.

## 9.4 mln::accu::impl Namespace Reference

Implementation namespace of accumulator namespace.

### 9.4.1 Detailed Description

Implementation namespace of accumulator namespace.

## 9.5 mln::accu::logic Namespace Reference

Namespace of logical accumulators.

### Classes

- struct [land](#)  
*"Logical-and" accumulator.*
- struct [land\\_basic](#)  
*"Logical-and" accumulator.*
- struct [lor](#)  
*"Logical-or" accumulator.*
- struct [lor\\_basic](#)  
*"Logical-or" accumulator class.*

### 9.5.1 Detailed Description

Namespace of logical accumulators.

## 9.6 mln::accu::math Namespace Reference

Namespace of mathematic accumulators.

### Classes

- struct [count](#)  
*Generic counter accumulator.*
- struct [inf](#)  
*Generic inf accumulator class.*
- struct [sum](#)  
*Generic sum accumulator class.*
- struct [sup](#)  
*Generic sup accumulator class.*

### 9.6.1 Detailed Description

Namespace of mathematic accumulators.

## 9.7 mln::accu::meta::logic Namespace Reference

Namespace of logical meta-accumulators.

### Classes

- struct [land](#)  
*Meta accumulator for land.*
- struct [land\\_basic](#)  
*Meta accumulator for [land\\_basic](#).*
- struct [lor](#)  
*Meta accumulator for lor.*
- struct [lor\\_basic](#)  
*Meta accumulator for [lor\\_basic](#).*

### 9.7.1 Detailed Description

Namespace of logical meta-accumulators.

## 9.8 mln::accu::meta::math Namespace Reference

Namespace of mathematic meta-accumulators.

### Classes

- struct [count](#)  
*Meta accumulator for count.*
- struct [inf](#)  
*Meta accumulator for inf.*
- struct [sum](#)  
*Meta accumulator for sum.*
- struct [sup](#)  
*Meta accumulator for sup.*

### 9.8.1 Detailed Description

Namespace of mathematic meta-accumulators.

## 9.9 mln::accu::meta::shape Namespace Reference

Namespace of shape meta-accumulators.

### Classes

- struct [bbox](#)  
*Meta accumulator for bbox.*
- struct [height](#)  
*Meta accumulator for height.*
- struct [volume](#)  
*Meta accumulator for volume.*

### 9.9.1 Detailed Description

Namespace of shape meta-accumulators.

## 9.10 mln::accu::meta::stat Namespace Reference

Namespace of statistical meta-accumulators.

### Classes

- struct [max](#)  
*Meta accumulator for max.*
- struct [max\\_h](#)  
*Meta accumulator for max.*
- struct [mean](#)  
*Meta accumulator for mean.*
- struct [median\\_alt](#)  
*Meta accumulator for [median\\_alt](#).*
- struct [median\\_h](#)  
*Meta accumulator for [median\\_h](#).*
- struct [min](#)



*Meta accumulator for min.*

- struct [min\\_h](#)  
*Meta accumulator for min.*
- struct [rank](#)  
*Meta accumulator for rank.*
- struct [rank\\_high\\_quant](#)  
*Meta accumulator for [rank\\_high\\_quant](#).*

### 9.10.1 Detailed Description

Namespace of statistical meta-accumulators.

## 9.11 mln::accu::shape Namespace Reference

Namespace of shape accumulators.

### Classes

- struct [bbox](#)  
*Generic bounding box accumulator class.*
- struct [height](#)  
*Height accumulator.*
- struct [volume](#)  
*Volume accumulator class.*

### 9.11.1 Detailed Description

Namespace of shape accumulators.

## 9.12 mln::accu::stat Namespace Reference

Namespace of statistical accumulators.

### Classes

- struct [deviation](#)  
*Generic standard deviation accumulator class.*
- struct [histo3d\\_rgb](#)

Define a histogram as accumulator which returns an *image3d*.

- struct [max](#)  
*Generic max accumulator class.*
- struct [max\\_h](#)  
*Generic max function based on histogram over a value set with type V.*
- struct [mean](#)  
*Generic mean accumulator class.*
- struct [median\\_alt](#)  
*Generic [median\\_alt](#) function based on histogram over a value set with type S.*
- struct [median\\_h](#)  
*Generic median function based on histogram over a value set with type V.*
- struct [min](#)  
*Generic min accumulator class.*
- struct [min\\_h](#)  
*Generic min function based on histogram over a value set with type V.*
- struct [min\\_max](#)  
*Generic min and max accumulator class.*
- struct [rank](#)  
*Generic rank accumulator class.*
- struct [rank< bool >](#)  
*rank accumulator class for Boolean.*
- struct [rank\\_high\\_quant](#)  
*Generic rank accumulator class.*
- struct [var](#)  
*Var accumulator class.*
- struct [variance](#)  
*Variance accumulator class.*

## Functions

- `template<typename V >`  
`bool operator==(const histo3d\_rgb< V > &histo1, const histo3d\_rgb< V > &histo2)`  
*Check whether a histogram is equal to another one.*

### 9.12.1 Detailed Description

Namespace of statistical accumulators.

### 9.12.2 Function Documentation

#### 9.12.2.1 `template<typename V > bool mln::accu::stat::operator==( const histo3d_rgb< V > & histo1, const histo3d_rgb< V > & histo2 )`

Check whether an histogram is equal to another one.

#### Parameters

- [in] *histo1* the first histogram to compare with.
- [in] *histo2* the second histogram.

The operator compares all the bins from the two histograms. Nobody uses this method unless unitary tests.

Definition at line 315 of file `histo3d_rgb.hh`.

References `mln::image3d< T >::domain()`.

## 9.13 mln::algebra Namespace Reference

Namespace of algebraic structure.

### Classes

- struct [h\\_mat](#)  
*N-Dimensional matrix with homogeneous coordinates.*
- class [h\\_vec](#)  
*N-Dimensional vector with homogeneous coordinates.*

### Functions

- `template<unsigned N, typename T > bool ldlt\_decomp( mat< N, N, T > &A, vec< N, T > &rdiag)`  
*Perform  $LDL^T$  decomposition of a symmetric positive definite matrix.*
- `template<unsigned N, typename T > void ldlt\_solve( const mat< N, N, T > &A, const vec< N, T > &rdiag, const vec< N, T > &B, vec< N, T > &x)`  
*Solve  $Ax = B$  after `mln::algebra::ldlt_decomp`.*
- `template<unsigned n, typename T, typename U > mln::trait::value_< typename mln::trait::op::times< T, U >::ret >::sum operator\*( const vec< n, T > &lhs, const vec< n, U > &rhs)`  
*Scalar product (dot product).*

- `template<typename T, typename U >`  
`vec< 3, typename mln::trait::op::times< T, U >::ret > vprod (const vec< 3, T > &lhs, const vec<`  
`3, U > &rhs)`  
*Vectorial product (cross product).*

### 9.13.1 Detailed Description

Namespace of algebraic structure.

### 9.13.2 Function Documentation

**9.13.2.1** `template<unsigned N, typename T > bool mln::algebra::ldlt_decomp ( mat< N, N, T >`  
`& A, vec< N, T > & rdiag ) [inline]`

Perform  $LDL^T$  decomposition of a symmetric positive definite matrix.

Like Cholesky, but no square roots. Overwrites lower triangle of matrix.

From Trimesh's `ldltdc` routine.

Definition at line 79 of file `misc.hh`.

Referenced by `mln::geom::mesh_curvature()`.

**9.13.2.2** `template<unsigned N, typename T > void mln::algebra::ldlt_solve ( const mat< N, N, T`  
`> & A, const vec< N, T > & rdiag, const vec< N, T > & B, vec< N, T > & x )`  
`[inline]`

Solve  $Ax = B$  after [mln::algebra::ldlt\\_decomp](#).

Definition at line 112 of file `misc.hh`.

Referenced by `mln::geom::mesh_curvature()`.

**9.13.2.3** `template<unsigned n, typename T, typename U > mln::trait::value_< typename`  
`mln::trait::op::times< T, U >::ret >::sum mln::algebra::operator* ( const vec< n, T >`  
`& lhs, const vec< n, U > & rhs ) [inline]`

Scalar product (dot product).

Definition at line 632 of file `algebra/vec.hh`.

References `mln::literal::zero`.

**9.13.2.4** `template<typename T, typename U > vec< 3, typename mln::trait::op::times< T, U`  
`>::ret > mln::algebra::vprod ( const vec< 3, T > & lhs, const vec< 3, U > & rhs )`  
`[inline]`

Vectorial product (cross product).

Definition at line 713 of file `algebra/vec.hh`.

References `vprod()`.

Referenced by mln::geom::mesh\_corner\_point\_area(), mln::geom::mesh\_curvature(), mln::geom::mesh\_normal(), and vprod().

## 9.14 mln::arith Namespace Reference

Namespace of arithmetic.

### Namespaces

- namespace [impl](#)  
*Implementation namespace of arith namespace.*

### Functions

- `template<typename I >  
mln::trait::concrete< I >::ret diff_abs (const Image< I > &lhs, const Image< I > &rhs)`  
*Point-wise absolute difference of images lhs and rhs.*
- `template<typename L , typename R , typename O >  
void div (const Image< L > &lhs, const Image< R > &rhs, Image< O > &output)`  
*Point-wise division of images lhs and rhs.*
- `template<typename I , typename V , typename O >  
void div_cst (const Image< I > &input, const V &val, Image< O > &output)`  
*Point-wise division of the value val to image input.*
- `template<typename L , typename R >  
void div_inplace (Image< L > &lhs, const Image< R > &rhs)`  
*Point-wise division of image rhs in image lhs.*
- `template<typename L , typename R >  
mln::trait::concrete< L >::ret min (const Image< L > &lhs, const Image< R > &rhs)`  
*Point-wise min of images lhs and rhs.*
- `template<typename L , typename R >  
void min_inplace (Image< L > &lhs, const Image< R > &rhs)`  
*Point-wise min of image lhs in image rhs.*
- `template<typename L , typename R >  
mln::trait::op::minus< L , R >::ret minus (const Image< L > &lhs, const Image< R > &rhs)`  
*Point-wise addition of images lhs and rhs.*
- `template<typename L , typename R , typename F >  
mln::trait::ch_value< L , typename F::result >::ret minus (const Image< L > &lhs, const Image< R > &rhs, const Function_v2v< F > &f)`  
*Point-wise addition of images lhs and rhs.*

- `template<typename V , typename L , typename R >`  
`mln::trait::ch_value< L, V >::ret minus (const Image< L > &lhs, const Image< R > &rhs)`  
*Point-wise addition of images lhs and rhs.*
- `template<typename I , typename V >`  
`mln::trait::op::minus< I, V >::ret minus\_cst (const Image< I > &input, const V &val)`  
*Point-wise addition of the value val to image input.*
- `template<typename I , typename V , typename F >`  
`mln::trait::ch_value< I, typename F::result >::ret minus\_cst (const Image< I > &input, const V &val, const Function\_v2v< F > &f)`  
*Point-wise addition of the value val to image input.*
- `template<typename I , typename V >`  
`I & minus\_cst\_inplace (Image< I > &input, const V &val)`  
*Point-wise addition of the value val to image input.*
- `template<typename L , typename R >`  
`void minus\_inplace (Image< L > &lhs, const Image< R > &rhs)`  
*Point-wise addition of image rhs in image lhs.*
- `template<typename L , typename R , typename F >`  
`mln::trait::ch_value< L, typename F::result >::ret plus (const Image< L > &lhs, const Image< R > &rhs, const Function\_v2v< F > &f)`  
*Point-wise addition of images lhs and rhs.*
- `template<typename V , typename L , typename R >`  
`mln::trait::ch_value< L, V >::ret plus (const Image< L > &lhs, const Image< R > &rhs)`  
*Point-wise addition of images lhs and rhs.*
- `template<typename L , typename R >`  
`mln::trait::op::plus< L, R >::ret plus (const Image< L > &lhs, const Image< R > &rhs)`  
*Point-wise addition of images lhs and rhs.*
- `template<typename I , typename V >`  
`mln::trait::op::plus< I, V >::ret plus\_cst (const Image< I > &input, const V &val)`  
*Point-wise addition of the value val to image input.*
- `template<typename I , typename V , typename F >`  
`mln::trait::ch_value< I, typename F::result >::ret plus\_cst (const Image< I > &input, const V &val, const Function\_v2v< F > &f)`  
*Point-wise addition of the value val to image input.*
- `template<typename W , typename I , typename V >`  
`mln::trait::ch_value< I, W >::ret plus\_cst (const Image< I > &input, const V &val)`  
*Point-wise addition of the value val to image input.*
- `template<typename I , typename V >`  
`I & plus\_cst\_inplace (Image< I > &input, const V &val)`  
*Point-wise addition of the value val to image input.*

- `template<typename L , typename R >`  
`void plus_inplace (Image< L > &lhs, const Image< R > &rhs)`  
*Point-wise addition of image rhs in image lhs.*
- `template<typename I >`  
`mln::trait::concrete< I >::ret revert (const Image< I > &input)`  
*Point-wise reversion of image input.*
- `template<typename I >`  
`void revert_inplace (Image< I > &input)`  
*Point-wise in-place reversion of image input.*
- `template<typename L , typename R , typename O >`  
`void times (const Image< L > &lhs, const Image< R > &rhs, Image< O > &output)`  
*Point-wise addition of images lhs and rhs.*
- `template<typename I , typename V , typename O >`  
`void times_cst (const Image< I > &input, const V &val, Image< O > &output)`  
*Point-wise addition of the value val to image input.*
- `template<typename L , typename R >`  
`void times_inplace (Image< L > &lhs, const Image< R > &rhs)`  
*Point-wise addition of image rhs in image lhs.*

## 9.14.1 Detailed Description

Namespace of arithmetic.

## 9.14.2 Function Documentation

### 9.14.2.1 `template<typename I > mln::trait::concrete< I >::ret mln::arith::diff_abs ( const Image< I > & lhs, const Image< I > & rhs ) [inline]`

Point-wise absolute difference of images lhs and rhs.

#### Parameters

- `[in] lhs` First operand image.
- `[in] rhs` Second operand image.

#### Returns

The result image.

#### Precondition

```
lhs.domain == rhs.domain
```

Definition at line 63 of file arith/diff\_abs.hh.

References `mln::data::transform()`.

**9.14.2.2** `template<typename L , typename R , typename O > void mln::arith::div ( const Image< L > & lhs, const Image< R > & rhs, Image< O > & output ) [inline]`

Point-wise division of images `lhs` and `rhs`.

#### Parameters

- [in] *lhs* First operand image.
- [in] *rhs* Second operand image.
- [out] *output* The result image.

#### Precondition

```
output.domain == lhs.domain == rhs.domain
```

Definition at line 227 of file `arith/div.hh`.

**9.14.2.3** `template<typename I , typename V , typename O > void mln::arith::div_cst ( const Image< I > & input, const V & val, Image< O > & output ) [inline]`

Point-wise division of the value `val` to image `input`.

#### Parameters

- [in] *input* The image.
- [in] *val* The value.
- [out] *output* The result image.

#### Precondition

```
output.domain == input.domain
```

Definition at line 242 of file `arith/div.hh`.

References `div_cst()`.

Referenced by `div_cst()`.

**9.14.2.4** `template<typename L , typename R > void mln::arith::div_inplace ( Image< L > & lhs, const Image< R > & rhs ) [inline]`

Point-wise division of image `rhs` in image `lhs`.

#### Parameters

- [in] *lhs* First operand image (subject to division).
- [in, out] *rhs* Second operand image (to div `lhs`).

This addition performs:

for all `p` of `rhs.domain`

`lhs(p) /= rhs(p)`



**Precondition**

```
rhs.domain <= lhs.domain
```

Definition at line 255 of file arith/div.hh.

References div\_inplace().

Referenced by div\_inplace().

**9.14.2.5 template<typename L , typename R > mln::trait::concrete< L >::ret mln::arith::min ( const Image< L > & lhs, const Image< R > & rhs ) [inline]**

Point-wise min of images lhs and rhs.

**Parameters**

[in] *lhs* First operand image.

[in] *rhs* Second operand image.

**Returns**

The result image.

**Precondition**

```
lhs.domain == rhs.domain
```

Definition at line 112 of file arith/min.hh.

References mln::initialize().

**9.14.2.6 template<typename L , typename R > void mln::arith::min\_inplace ( Image< L > & lhs, const Image< R > & rhs ) [inline]**

Point-wise min of image lhs in image rhs.

**Parameters**

[in, out] *lhs* First operand image.

[in] *rhs* Second operand image.

**Precondition**

```
rhs.domain == lhs.domain
```

Definition at line 128 of file arith/min.hh.

**9.14.2.7 template<typename L , typename R > mln::trait::op::minus< L, R >::ret mln::arith::minus ( const Image< L > & lhs, const Image< R > & rhs ) [inline]**

Point-wise addition of images lhs and rhs.

**Parameters**

[in] *lhs* First operand image.

[in] *rhs* Second operand image.

### Returns

The result image.

### Precondition

```
lhs.domain == rhs.domain
```

Definition at line 331 of file arith/minus.hh.

References mln::initialize().

**9.14.2.8** `template<typename L , typename R , typename F > mln::trait::ch_value< L, typename F::result >::ret mln::arith::minus ( const Image< L > & lhs, const Image< R > & rhs, const Function_v2v< F > & f ) [inline]`

Point-wise addition of images *lhs* and *rhs*.

### Parameters

[in] *lhs* First operand image.

[in] *rhs* Second operand image.

[in] *f* [Function](#).

### Returns

The result image.

### Precondition

```
lhs.domain == rhs.domain
```

Definition at line 350 of file arith/minus.hh.

References mln::initialize().

**9.14.2.9** `template<typename V , typename L , typename R > mln::trait::ch_value< L, V >::ret mln::arith::minus ( const Image< L > & lhs, const Image< R > & rhs ) [inline]`

Point-wise addition of images *lhs* and *rhs*.

### Parameters

[in] *lhs* First operand image.

[in] *rhs* Second operand image.

### Returns

The result image.

The free parameter *V* sets the destination value type.

### Precondition

```
lhs.domain == rhs.domain
```

Definition at line 369 of file arith/minus.hh.

**9.14.2.10** `template<typename I , typename V > mln::trait::op::minus< I, V >::ret  
mln::arith::minus_cst ( const Image< I > & input, const V & val ) [inline]`

Point-wise addition of the value *val* to image *input*.

#### Parameters

- [in] *input* The image.
- [in] *val* The value.

#### Returns

The result image.

#### Precondition

`input.is_valid`

Definition at line 387 of file arith/minus.hh.

**9.14.2.11** `template<typename I , typename V , typename F > mln::trait::ch_value< I, typename  
F::result >::ret mln::arith::minus_cst ( const Image< I > & input, const V & val,  
const Function_v2v< F > & f ) [inline]`

Point-wise addition of the value *val* to image *input*.

#### Parameters

- [in] *input* The image.
- [in] *val* The value.
- [in] *f* [Function](#).

#### Returns

The result image.

#### Precondition

`input.is_valid`

Definition at line 405 of file arith/minus.hh.

**9.14.2.12** `template<typename I , typename V > I & mln::arith::minus_cst_inplace ( Image< I >  
& input, const V & val ) [inline]`

Point-wise addition of the value *val* to image *input*.

#### Parameters

- [in, out] *input* The image.
- [in] *val* The value.

#### Precondition

`input.is_valid`

Definition at line 440 of file arith/minus.hh.

References `minus_cst_inplace()`, and `minus_inplace()`.

Referenced by `minus_cst_inplace()`.

**9.14.2.13** `template<typename L , typename R > void mln::arith::minus_inplace ( Image< L > & lhs, const Image< R > & rhs ) [inline]`

Point-wise addition of image `rhs` in image `lhs`.

#### Parameters

[in, out] *lhs* First operand image (subject to addition).

[in] *rhs* Second operand image (to be added to `lhs`).

This addition performs:

for all `p` of `rhs.domain`

`lhs(p) -= rhs(p)`

#### Precondition

```
rhs.domain == lhs.domain
```

Definition at line 424 of file arith/minus.hh.

References `minus_inplace()`.

Referenced by `minus_cst_inplace()`, and `minus_inplace()`.

**9.14.2.14** `template<typename L , typename R , typename F > mln::trait::ch_value< L , typename F::result >::ret mln::arith::plus ( const Image< L > & lhs, const Image< R > & rhs, const Function_v2v< F > & f ) [inline]`

Point-wise addition of images `lhs` and `rhs`.

#### Parameters

[in] *lhs* First operand image.

[in] *rhs* Second operand image.

[in] *f* [Function](#).

#### Returns

The result image.

#### Precondition

```
lhs.domain == rhs.domain
```

Definition at line 367 of file arith/plus.hh.

References `mln::initialize()`.

**9.14.2.15** `template<typename V , typename L , typename R > mln::trait::ch_value< L, V >::ret  
mln::arith::plus ( const Image< L > & lhs, const Image< R > & rhs ) [inline]`

Point-wise addition of images `lhs` and `rhs`.

#### Parameters

[in] *lhs* First operand image.

[in] *rhs* Second operand image.

#### Returns

The result image.

The free parameter `V` sets the destination value type.

#### Precondition

```
lhs.domain == rhs.domain
```

Definition at line 386 of file `arith/plus.hh`.

**9.14.2.16** `template<typename L , typename R > mln::trait::op::plus< L, R >::ret  
mln::arith::plus ( const Image< L > & lhs, const Image< R > & rhs ) [inline]`

Point-wise addition of images `lhs` and `rhs`.

#### Parameters

[in] *lhs* First operand image.

[in] *rhs* Second operand image.

#### Returns

The result image.

#### Precondition

```
lhs.domain == rhs.domain
```

Definition at line 348 of file `arith/plus.hh`.

References `mln::initialize()`.

**9.14.2.17** `template<typename I , typename V > mln::trait::op::plus< I, V >::ret  
mln::arith::plus_cst ( const Image< I > & input, const V & val ) [inline]`

Point-wise addition of the value `val` to image `input`.

#### Parameters

[in] *input* The image.

[in] *val* The value.

**Returns**

The result image.

**Precondition**

```
input.is_valid
```

Definition at line 404 of file arith/plus.hh.

Referenced by plus\_cst().

**9.14.2.18** `template<typename I, typename V, typename F > mln::trait::ch_value< I, typename F::result >::ret mln::arith::plus_cst ( const Image< I > & input, const V & val, const Function_v2v< F > & f ) [inline]`

Point-wise addition of the value `val` to image `input`.

**Parameters**

[in] *input* The image.

[in] *val* The value.

[in] *f* [Function](#).

**Returns**

The result image.

**Precondition**

```
input.is_valid
```

Definition at line 422 of file arith/plus.hh.

**9.14.2.19** `template<typename W, typename I, typename V > mln::trait::ch_value< I, W >::ret mln::arith::plus_cst ( const Image< I > & input, const V & val ) [inline]`

Point-wise addition of the value `val` to image `input`.

**Parameters**

[in] *input* The image.

[in] *val* The value.

**Returns**

The result image.

**Precondition**

```
input.is_valid
```

Definition at line 441 of file arith/plus.hh.

References plus\_cst().

**9.14.2.20** `template<typename I , typename V > I & mln::arith::plus_cst_inplace ( Image< I > & input, const V & val ) [inline]`

Point-wise addition of the value `val` to image `input`.

**Parameters**

[in, out] *input* The image.  
[in] *val* The value.

**Precondition**

```
input.is_valid
```

Definition at line 475 of file `arith/plus.hh`.

References `plus_cst_inplace()`, and `plus_inplace()`.

Referenced by `plus_cst_inplace()`.

**9.14.2.21** `template<typename L , typename R > void mln::arith::plus_inplace ( Image< L > & lhs, const Image< R > & rhs ) [inline]`

Point-wise addition of image `rhs` in image `lhs`.

**Parameters**

[in, out] *lhs* First operand image (subject to addition).  
[in] *rhs* Second operand image (to be added to `lhs`).

This addition performs:

for all `p` of `rhs.domain`

```
lhs(p) += rhs(p)
```

**Precondition**

```
rhs.domain == lhs.domain
```

Definition at line 459 of file `arith/plus.hh`.

References `plus_inplace()`.

Referenced by `plus_cst_inplace()`, and `plus_inplace()`.

**9.14.2.22** `template<typename I > mln::trait::concrete< I >::ret mln::arith::revert ( const Image< I > & input ) [inline]`

Point-wise reversion of image `input`.

**Parameters**

[in] *input* the input image.

**Returns**

The result image.

**Precondition**

```
input.is_valid
```

It performs:

for all  $p$  of `input.domain`

$output(p) = min + (max - input(p))$

Definition at line 158 of file `revert.hh`.

References `mln::initialize()`.

#### 9.14.2.23 `template<typename I > void mln::arith::revert_inplace ( Image< I > & input ) [inline]`

Point-wise in-place reversion of image `input`.

**Parameters**

[in, out] *input* The target image.

**Precondition**

```
input.is_valid
```

It performs:

for all  $p$  of `input.domain`

$input(p) = min + (max - input(p))$

Definition at line 174 of file `revert.hh`.

#### 9.14.2.24 `template<typename L , typename R , typename O > void mln::arith::times ( const Image< L > & lhs, const Image< R > & rhs, Image< O > & output ) [inline]`

Point-wise addition of images `lhs` and `rhs`.

**Parameters**

[in] *lhs* First operand image.

[in] *rhs* Second operand image.

[out] *output* The result image.

**Precondition**

```
output.domain == lhs.domain == rhs.domain
```

Definition at line 226 of file `arith/times.hh`.

#### 9.14.2.25 `template<typename I , typename V , typename O > void mln::arith::times_cst ( const Image< I > & input, const V & val, Image< O > & output ) [inline]`

Point-wise addition of the value `val` to image `input`.



**Parameters**

- [in] *input* The image.
- [in] *val* The value.
- [out] *output* The result image.

**Precondition**

```
output.domain == input.domain
```

Definition at line 241 of file arith/times.hh.

References times\_cst().

Referenced by times\_cst().

### 9.14.2.26 `template<typename L , typename R > void mln::arith::times_inplace ( Image< L > & lhs, const Image< R > & rhs ) [inline]`

Point-wise addition of image *rhs* in image *lhs*.

**Parameters**

- [in] *lhs* First operand image (subject to addition).
- [in, out] *rhs* Second operand image (to be added to *lhs*).

This addition performs:

for all *p* of *rhs*.domain

```
lhs(p) += rhs(p)
```

**Precondition**

```
rhs.domain <= lhs.domain
```

Definition at line 254 of file arith/times.hh.

References times\_inplace().

Referenced by times\_inplace().

## 9.15 mln::arith::impl Namespace Reference

Implementation namespace of arith namespace.

**Namespaces**

- namespace [generic](#)  
*Generic implementation namespace of arith namespace.*

### 9.15.1 Detailed Description

Implementation namespace of arith namespace.

## 9.16 mln::arith::impl::generic Namespace Reference

Generic implementation namespace of arith namespace.

### 9.16.1 Detailed Description

Generic implementation namespace of arith namespace.

## 9.17 mln::binarization Namespace Reference

Namespace of "point-wise" expression tools.

### Functions

- `template<typename I , typename F > mln::trait::ch_value< I, bool >::ret binarization (const Image< I > &input, const Function_v2b< F > &fun)`

*Thresholds the values of `input` so that they can be stored in the output binary image.*

- `template<typename I > mln::trait::ch_value< I, bool >::ret threshold (const Image< I > &input, const typename I::value threshold)`

*Thresholds the values of `input` so that they can be stored in the output binary image.*

### 9.17.1 Detailed Description

Namespace of "point-wise" expression tools.

### 9.17.2 Function Documentation

- 9.17.2.1** `template<typename I , typename F > mln::trait::ch_value< I, bool >::ret mln::binarization::binarization ( const Image< I > & input, const Function_v2b< F > & fun ) [inline]`

Thresholds the values of `input` so that they can be stored in the output binary image.

#### Parameters

- [in] *input* The input image.
- [in] *fun* The thresholding function, from value(I) to bool.

`for_all(p), output(p) = fun(p)`

Definition at line 86 of file `binarization.hh`.

Referenced by `threshold()`.

### 9.17.2.2 `template<typename I> mln::trait::ch_value< I, bool >::ret mln::binarization::threshold ( const Image< I > & input, const typename I::value threshold ) [inline]`

Thresholds the values of `input` so that they can be stored in the `output` binary image.

#### Parameters

- [in] *input* The input image.
- [in] *threshold* The threshold.

If `input(p)` is greater or equal than the threshold, the value in the output image in the same point will be TRUE, else FALSE.

Definition at line 61 of file `binarization/threshold.hh`.

References `binarization()`.

## 9.18 mln::border Namespace Reference

Namespace of routines related to image virtual (outer) border.

### Namespaces

- namespace [impl](#)  
*Implementation namespace of border namespace.*

### Functions

- `template<typename I>`  
void [adjust](#) (const [Image](#)< I > &ima, unsigned min\_thickness)
- `template<typename I>`  
void [duplicate](#) (const [Image](#)< I > &ima)
- `template<typename I, typename J>`  
void [equalize](#) (const [Image](#)< I > &ima1, const [Image](#)< J > &ima2, unsigned min\_thickness)
- `template<typename I>`  
void [fill](#) (const [Image](#)< I > &ima, const typename I::value &v)
- `template<typename I>`  
unsigned [find](#) (const [Image](#)< I > &ima)
- `template<typename I>`  
unsigned [get](#) (const [Image](#)< I > &ima)
- `template<typename I>`  
void [mirror](#) (const [Image](#)< I > &ima)
- `template<typename I>`  
void [resize](#) (const [Image](#)< I > &ima, unsigned thickness)

*Facade.*

### 9.18.1 Detailed Description

Namespace of routines related to image virtual (outer) border.

## 9.18.2 Function Documentation

### 9.18.2.1 `template<typename I> void mln::border::adjust ( const Image< I > & ima, unsigned min_thickness ) [inline]`

Adjust the virtual (outer) border of image *ima* so that its size is at least *min\_thickness*.

#### Parameters

- [in, out] *ima* The image whose border is to be adjusted.
- [in] *min\_thickness* The expected border minimum thickness.

#### Precondition

*ima* has to be initialized.

#### Warning

If the image border is already larger than *min\_thickness*, this routine is a no-op.

Definition at line 62 of file `border/adjust.hh`.

References `get()`, and `resize()`.

### 9.18.2.2 `template<typename I> void mln::border::duplicate ( const Image< I > & ima )`

Assign the virtual (outer) border of image *ima* with the duplicate of the inner border of this image.

#### Parameters

- [in, out] *ima* The image whose border is to be duplicated.

#### Precondition

*ima* has to be initialized.

Definition at line 252 of file `border/duplicate.hh`.

References `get()`.

### 9.18.2.3 `template<typename I, typename J> void mln::border::equalize ( const Image< I > & ima1, const Image< J > & ima2, unsigned min_thickness ) [inline]`

Equalize the virtual (outer) border of images *ima1* and *ima2* so that their size is equal and is at least *min\_thickness*.

#### Parameters

- [in, out] *ima1* The first image whose border is to be equalized.
- [in, out] *ima2* The second image whose border is to be equalized.
- [in] *min\_thickness* The expected border minimum thickness of both images.

#### Precondition

*ima1* has to be initialized.  
*ima2* has to be initialized.

### Warning

If both image borders already have the same thickness and if this thickness is larger than `min_thickness`, this routine is a no-op.

Definition at line 112 of file `border/equalize.hh`.

References `get()`.

#### 9.18.2.4 `template<typename I> void mln::border::fill ( const Image< I > & ima, const typename I::value & v ) [inline]`

Fill the virtual (outer) border of image `ima` with the single value `v`.

#### Parameters

[in, out] *ima* The image whose border is to be filled.

[in] *v* The value to assign to all border pixels.

#### Precondition

`ima` has to be initialized.

Definition at line 201 of file `border/fill.hh`.

#### 9.18.2.5 `template<typename I> unsigned mln::border::find ( const Image< I > & ima ) [inline]`

Find the virtual (outer) border thickness of image `ima`.

#### Parameters

[in] *ima* The image.

#### Returns

The border thickness (0 if there is no border).

#### Precondition

`ima` has to be initialized.

Definition at line 95 of file `find.hh`.

#### 9.18.2.6 `template<typename I> unsigned mln::border::get ( const Image< I > & ima ) [inline]`

Get the virtual (outer) border thickness of image `ima`.

#### Parameters

[in] *ima* The image.

#### Returns

The border thickness (0 if there is no border).

**Precondition**

*ima* has to be initialized.

Definition at line 90 of file border/get.hh.

Referenced by adjust(), duplicate(), and equalize().

**9.18.2.7** `template<typename I> void mln::border::mirror ( const Image< I > & ima )  
[inline]`

Mirror the virtual (outer) border of image *ima* with the (inner) level contents of this image.

**Parameters**

[in, out] *ima* The image whose border is to be mirrored.

**Precondition**

*ima* has to be initialized.

Definition at line 211 of file border/mirror.hh.

**9.18.2.8** `template<typename I> void mln::border::resize ( const Image< I > & ima, unsigned  
thickness ) [inline]`

Facade.

Resize the virtual (outer) border of image *ima* to exactly *thickness*.

**Parameters**

[in, out] *ima* The image whose border is to be resized.

[in] *thickness* The expected border thickness.

**Precondition**

*ima* has to be initialized.

**Warning**

If the image border already has the expected thickness, this routine is a no-op.

Definition at line 126 of file resize.hh.

References mln::primary(), and resize().

Referenced by adjust(), and resize().

## 9.19 mln::border::impl Namespace Reference

Implementation namespace of border namespace.

## Namespaces

- namespace [generic](#)

*Generic implementation namespace of border namespace.*

### 9.19.1 Detailed Description

Implementation namespace of border namespace.

## 9.20 mln::border::impl::generic Namespace Reference

Generic implementation namespace of border namespace.

### 9.20.1 Detailed Description

Generic implementation namespace of border namespace.

## 9.21 mln::canvas Namespace Reference

Namespace of canvas.

## Namespaces

- namespace [browsing](#)

*Namespace of browsing canvas.*

- namespace [impl](#)

*Implementation namespace of canvas namespace.*

- namespace [labeling](#)

*Namespace of labeling canvas.*

- namespace [morpho](#)

*Namespace of morphological canvas.*

## Classes

- struct [chamfer](#)

*Compute chamfer distance.*

## Functions

- `template<typename I, typename N, typename W, typename D, typename F > mln::trait::ch_value< I, D >::ret distance_front (const Image< I > &input, const Neighborhood< N > &nbh, const Weighted_Window< W > &w_win, D max, F &functor)`  
*Canvas of discrete distance computation by thick front propagation.*
- `template<typename I, typename N, typename D, typename F > mln::trait::ch_value< I, D >::ret distance_geodesic (const Image< I > &input, const Neighborhood< N > &nbh, D max, F &functor)`  
*Discrete geodesic distance canvas.*

### 9.21.1 Detailed Description

Namespace of canvas.

### 9.21.2 Function Documentation

- 9.21.2.1** `template<typename I, typename N, typename W, typename D, typename F > mln::trait::ch_value< I, D >::ret mln::canvas::distance_front ( const Image< I > &input, const Neighborhood< N > & nbh, const Weighted_Window< W > & w_win, D max, F & functor ) [inline]`

Canvas of discrete distance computation by thick front propagation.

Definition at line 397 of file canvas/distance\_front.hh.

Referenced by `mln::transform::influence_zone_front()`.

- 9.21.2.2** `template<typename I, typename N, typename D, typename F > mln::trait::ch_value< I, D >::ret mln::canvas::distance_geodesic ( const Image< I > &input, const Neighborhood< N > & nbh, D max, F & functor ) [inline]`

Discrete geodesic distance canvas.

Definition at line 321 of file canvas/distance\_geodesic.hh.

Referenced by `mln::transform::influence_zone_geodesic_saturated()`.

## 9.22 mln::canvas::browsing Namespace Reference

Namespace of browsing canvas.

### Classes

- struct `backdiagonal2d_t`  
*Browsing in a certain direction.*
- struct `breadth_first_search_t`



*Breadth-first search algorithm for graph, on vertices.*

- struct [depth\\_first\\_search\\_t](#)  
*Breadth-first search algorithm for graph, on vertices.*
- struct [diagonal2d\\_t](#)  
*Browsing in a certain direction.*
- struct [dir\\_struct\\_elt\\_incr\\_update\\_t](#)  
*Browsing in a certain direction with a segment.*
- struct [directional\\_t](#)  
*Browsing in a certain direction.*
- struct [fwd\\_t](#)  
*Canvas for forward browsing.*
- struct [hyper\\_directional\\_t](#)  
*Browsing in a certain direction.*
- struct [snake\\_fwd\\_t](#)  
*Browsing in a snake-way, forward.*
- struct [snake\\_generic\\_t](#)  
*Multidimensional Browsing in a given-way.*
- struct [snake\\_vert\\_t](#)  
*Browsing in a snake-way, forward.*

### 9.22.1 Detailed Description

Namespace of browsing canvas.

## 9.23 mln::canvas::impl Namespace Reference

Implementation namespace of canvas namespace.

### 9.23.1 Detailed Description

Implementation namespace of canvas namespace.

## 9.24 mln::canvas::labeling Namespace Reference

Namespace of labeling canvas.

## Namespaces

- namespace [impl](#)

*Implementation namespace of labeling canvas namespace.*

## Functions

- `template<typename I, typename N, typename L, typename F > mln::trait::ch_value< I, L >::ret blobs (const Image< I > &input_, const Neighborhood< N > &nbh_, L &nlabels, F &functor)`

*Canvas for connected component labeling of the binary objects of a binary image using a queue-based algorithm.*

### 9.24.1 Detailed Description

Namespace of labeling canvas.

### 9.24.2 Function Documentation

- 9.24.2.1** `template<typename I, typename N, typename L, typename F > mln::trait::ch_value< I, L >::ret mln::canvas::labeling::blobs ( const Image< I > & input_, const Neighborhood< N > & nbh_, L & nlabels, F & functor ) [inline]`

Canvas for connected component labeling of the binary objects of a binary image using a queue-based algorithm.

#### Parameters

- [in] *input* The input image.
- [in] *nbh* The connexity of the objects.
- [out] *nlabels* The Number of labels. Its value is set in the algorithms.
- [in, out] *functor* A functor computing data while labeling.

#### Returns

The label image.

#### Precondition

The input image has to be binary (checked at compile-time).

A fast queue is used so that the algorithm is not recursive and can handle large binary objects (blobs).

Definition at line 167 of file canvas/labeling/blobs.hh.

## 9.25 mln::canvas::labeling::impl Namespace Reference

Implementation namespace of labeling canvas namespace.

### 9.25.1 Detailed Description

Implementation namespace of labeling canvas namespace.

## 9.26 mln::canvas::morpho Namespace Reference

Namespace of morphological canvas.

### 9.26.1 Detailed Description

Namespace of morphological canvas.

## 9.27 mln::convert Namespace Reference

Namespace of conversion routines.

### Functions

- template<typename V >  
void [from\\_to](#) (const float &from, [Value](#)< V > &to)  
*Conversion of a float from towards a value to.*
- template<typename V >  
void [from\\_to](#) (const int &from, [Value](#)< V > &to)  
*Conversion of a int from towards a value to.*
- template<typename V >  
void [from\\_to](#) (const double &from, [Value](#)< V > &to)  
*Conversion of a double from towards a value to.*
- template<typename V >  
void [from\\_to](#) (const unsigned &from, [Value](#)< V > &to)  
*Conversion of an unsigned from towards a value to.*
- template<typename S >  
[mln\\_image\\_from\\_grid](#) (typename S::site::grid, bool) to\_image(const [Site\\_Set](#)< S > &pset)  
*Convert a point set pset into a binary image.*
- template<typename W >  
[mln\\_image\\_from\\_grid](#) (typename W::site::grid, bool) to\_image(const [Window](#)< W > &win)  
*Convert a window win into a binary image.*
- template<typename W >  
[mln\\_image\\_from\\_grid](#) (typename W::site::grid, [mln\\_weight](#)(W)) to\_image(const [Weighted\\_Window](#)< W > &w\_win)  
*Convert a weighted window w\_win into an image.*

- `template<typename N >`  
`mln_image_from_grid` (typename N::site::grid, bool) `to_image`(const `Neighborhood`< N > &nbh)  
*Convert a neighborhood nbh into a binary image.*
- `template<typename N >`  
`mln_window` (N) `to_window`(const `Neighborhood`< N > &nbh)  
*Convert a neighborhood nbh into a window.*
- `template<typename T , typename O >`  
`T to` (const O &from)  
*Conversion of the object from towards an object with type T.*
- `template<typename P >`  
`P::dpoint to_dpoint` (const `Point_Site`< P > &p)  
*Convert a point site p into a delta-point.*
- `template<typename I >`  
`pw::value_< I > to_fun` (const `Image`< I > &ima)  
*Convert an image into a function.*
- `template<typename T >`  
`imageId< unsigned > to_image` (const `histo::array`< T > &h)  
*Convert an histo h into an imageId<unsigned>.*
- `template<typename I >`  
`p_array< typename I::psite > to_p_array` (const `Image`< I > &img)  
*Convert an image img into a p\_array.*
- `template<typename S >`  
`p_array< typename S::psite > to_p_array` (const `Site_Set`< S > &pset)  
*Convert a point set pset into a p\_array (point set vector).*
- `template<typename W >`  
`p_array< typename W::psite > to_p_array` (const `Window`< W > &win, const typename W::psite &p)  
*Convert a window win centered at point p into a p\_array (point set vector).*
- `template<typename N >`  
`p_set< typename N::psite > to_p_set` (const `Neighborhood`< N > &nbh)  
*Convert a neighborhood nbh into a site set.*
- `template<typename I >`  
`p_set< typename I::psite > to_p_set` (const `Image`< I > &ima)  
*Convert a binary image ima into a site set.*
- `template<typename P , typename C >`  
`p_set< P > to_p_set` (const `std::set`< P, C > &s)  
*Convert an std::set s of sites into a site set.*
- `template<typename S >`  
`p_set< typename S::psite > to_p_set` (const `Site_Set`< S > &ps)

*Convert any site set `ps` into a 'mlnp\_set' site set.*

- `template<typename W >`  
`p_set< typename W::psite > to_p_set (const Window< W > &win)`  
*Convert a `Window` `win` into a site set.*
- `template<typename I >`  
`QImage to_qimage (const Image< I > &ima)`  
*Convert a Milena image to a `Qimage`.*
- `template<typename N >`  
`window< typename N::dpoint > to_upper_window (const Neighborhood< N > &nbh)`  
*Convert a neighborhood `nbh` into an upper window.*
- `template<typename W >`  
`window< typename W::dpsite > to_upper_window (const Window< W > &win)`  
*Convert a window `nbh` into an upper window.*
- `template<typename D , typename C >`  
`window< D > to_window (const std::set< D, C > &s)`  
*Convert an `std::set` `s` of delta-sites into a window.*
- `template<typename I >`  
`window< typename I::site::dpsite > to_window (const Image< I > &ima)`  
*Convert a binary image `ima` into a window.*
- `template<typename S >`  
`window< typename S::site::dpsite > to_window (const Site_Set< S > &pset)`  
*Convert a site set `pset` into a window.*

## Variables

- `fun::C< R(*) (A)> to_fun (R(*f)(A))`  
*Convert a `C` unary function into an `mln::fun::C`.*

### 9.27.1 Detailed Description

Namespace of conversion routines.

### 9.27.2 Function Documentation

#### 9.27.2.1 `template<typename V > void mln::convert::from_to ( const float & from, Value< V > & to )`

Conversion of a float `from` towards a value `to`.

**9.27.2.2** `template<typename V > void mln::convert::from_to ( const int & from, Value< V > & to )`

Conversion of a int *from* towards a value *to*.

**9.27.2.3** `template<typename V > void mln::convert::from_to ( const double & from, Value< V > & to )`

Conversion of a double *from* towards a value *to*.

**9.27.2.4** `template<typename V > void mln::convert::from_to ( const unsigned & from, Value< V > & to )`

Conversion of an unsigned *from* towards a value *to*.

**9.27.2.5** `template<typename S > mln::convert::mln_image_from_grid ( typename S::site::grid, bool ) const [inline]`

Convert a point set *pset* into a binary image.

Width of the converted image will be *pset.bbox* + 2 \* *border*.

**9.27.2.6** `template<typename W > mln::convert::mln_image_from_grid ( typename W::site::grid, bool ) const`

Convert a window *win* into a binary image.

**9.27.2.7** `template<typename W > mln::convert::mln_image_from_grid ( typename W::site::grid, mln_weight(W) ) const`

Convert a weighted window *w\_win* into an image.

**9.27.2.8** `template<typename N > mln::convert::mln_image_from_grid ( typename N::site::grid, bool ) const`

Convert a neighborhood *nbh* into a binary image.

**9.27.2.9** `template<typename N > mln::convert::mln_window ( N ) const [inline]`

Convert a neighborhood *nbh* into a window.

Definition at line 74 of file *to\_window.hh*.

**9.27.2.10** `template<typename T, typename O > T mln::convert::to ( const O & from ) [inline]`

Conversion of the object *from* towards an object with type *T*.

Definition at line 63 of file *to.hh*.

References mln::mln\_exact().

Referenced by mln::make\_debug\_graph\_image().

**9.27.2.11** `template<typename P > P::dpoint mln::convert::to_dpoint ( const Point_Site< P > & p ) [inline]`

Convert a point site `p` into a delta-point.

Definition at line 52 of file `to_dpoint.hh`.

**9.27.2.12** `template<typename I > pw::value_<I> mln::convert::to_fun ( const Image< I > & ima )`

Convert an image into a function.

Definition at line 64 of file `to_fun.hh`.

**9.27.2.13** `template<typename T > imageId<unsigned> mln::convert::to_image ( const histo::array< T > & h )`

Convert an histo `h` into an `imageId<unsigned>`.

**9.27.2.14** `template<typename I > p_array< typename I::psite > mln::convert::to_p_array ( const Image< I > & img ) [inline]`

Convert an image `img` into a `p_array`.

Definition at line 98 of file `to_p_array.hh`.

References `mln::p_array< P >::append()`.

**9.27.2.15** `template<typename S > p_array< typename S::psite > mln::convert::to_p_array ( const Site_Set< S > & pset ) [inline]`

Convert a point set `pset` into a `p_array` (point set vector).

Definition at line 68 of file `to_p_array.hh`.

References `mln::p_array< P >::append()`.

**9.27.2.16** `template<typename W > p_array< typename W::psite > mln::convert::to_p_array ( const Window< W > & win, const typename W::psite & p ) [inline]`

Convert a window `win` centered at point `p` into a `p_array` (point set vector).

Definition at line 82 of file `to_p_array.hh`.

References `mln::p_array< P >::append()`, and `mln::p_array< P >::reserve()`.

**9.27.2.17** `template<typename N > p_set< typename N::psite > mln::convert::to_p_set ( const Neighborhood< N > & nbh ) [inline]`

Convert a neighborhood `nbh` into a site set.

Definition at line 83 of file `to_p_set.hh`.

References `mln::p_set< P >::insert()`.

**9.27.2.18** `template<typename I > p_set< typename I::psite > mln::convert::to_p_set ( const Image< I > & ima ) [inline]`

Convert a binary image `ima` into a site set.

Definition at line 97 of file `to_p_set.hh`.

References `mln::p_set< P >::insert()`.

**9.27.2.19** `template<typename P , typename C > p_set< P > mln::convert::to_p_set ( const std::set< P, C > & s ) [inline]`

Convert an `std::set` `s` of sites into a site set.

`C` is the comparison functor.

Definition at line 130 of file `to_p_set.hh`.

References `mln::p_set< P >::insert()`.

**9.27.2.20** `template<typename S > p_set< typename S::psite > mln::convert::to_p_set ( const Site_Set< S > & ps ) [inline]`

Convert any site set `ps` into a 'mlnp\_set' site set.

Definition at line 144 of file `to_p_set.hh`.

References `mln::p_set< P >::insert()`.

**9.27.2.21** `template<typename W > p_set< typename W::psite > mln::convert::to_p_set ( const Window< W > & win ) [inline]`

Convert a [Window](#) `win` into a site set.

Definition at line 117 of file `to_p_set.hh`.

References `mln::p_set< P >::insert()`.

**9.27.2.22** `template<typename I > QImage mln::convert::to_qimage ( const Image< I > & ima ) [inline]`

Convert a Milena image to a QImage.

Definition at line 279 of file `to_qimage.hh`.

**9.27.2.23** `template<typename N > window< typename N::dpoint > mln::convert::to_upper_window ( const Neighborhood< N > & nbh ) [inline]`

Convert a neighborhood `nbh` into an upper window.

Definition at line 82 of file `to_upper_window.hh`.



References mln::window< D >::insert().

**9.27.2.24** `template<typename W > window< typename W::dpsite >  
mln::convert::to_upper_window ( const Window< W > & win ) [inline]`

Convert a window `nbh` into an upper window.

Definition at line 67 of file `to_upper_window.hh`.

References mln::window< D >::insert().

**9.27.2.25** `template<typename D, typename C > window< D > mln::convert::to_window ( const  
std::set< D, C > & s ) [inline]`

Convert an `std::set` `s` of delta-sites into a window.

Definition at line 128 of file `to_window.hh`.

References mln::window< D >::insert().

**9.27.2.26** `template<typename I > window< typename I::site::dpsite > mln::convert::to_window ( const  
Image< I > & ima ) [inline]`

Convert a binary image `ima` into a window.

Definition at line 94 of file `to_window.hh`.

References mln::window< D >::insert().

Referenced by `to_window()`.

**9.27.2.27** `template<typename S > window< typename S::site::dpsite > mln::convert::to_window  
( const Site_Set< S > & pset ) [inline]`

Convert a site set `pset` into a window.

Definition at line 117 of file `to_window.hh`.

References `to_window()`.

## 9.27.3 Variable Documentation

**9.27.3.1** `pw::value_< I > mln::convert::to_fun [inline]`

Convert a C unary function into an `mln::fun::C`.

Definition at line 45 of file `to_fun.hh`.

## 9.28 mln::data Namespace Reference

Namespace of image processing routines related to pixel data.

## Namespaces

- namespace [approx](#)  
*Namespace of image processing routines related to pixel levels with approximation.*
- namespace [impl](#)  
*Implementation namespace of data namespace.*
- namespace [naive](#)  
*Namespace of image processing routines related to pixel levels with naive approach.*

## Functions

- `template<typename I , typename O >`  
`void abs (const Image< I > &input, Image< O > &output)`
- `template<typename I >`  
`void abs\_inplace (Image< I > &input)`
- `template<typename I , typename F >`  
`void apply (Image< I > &input, const Function\_v2v< F > &f)`
- `template<typename A , typename I >`  
`A::result compute (const Accumulator< A > &a, const Image< I > &input)`  
*Compute an accumulator onto the pixel values of the image input.*
- `template<typename A , typename I >`  
`A::result compute (Accumulator< A > &a, const Image< I > &input)`  
*Compute an accumulator onto the pixel values of the image input.*
- `template<typename V , typename I >`  
`mln::trait::ch_value< I , V >::ret convert (const V &v, const Image< I > &input)`  
*Convert the image input by changing the value type.*
- `template<typename I , typename W , typename O >`  
`void fast\_median (const Image< I > &input, const Window< W > &win, Image< O > &output)`
- `template<typename I , typename D >`  
`void fill (Image< I > &ima, const D &data)`
- `template<typename I , typename J >`  
`void fill\_with\_image (Image< I > &ima, const Image< J > &data)`  
*Fill the image ima with the values of the image data.*
- `template<typename I , typename W >`  
`mln::trait::concrete< I >::ret median (const Image< I > &input, const Window< W > &win)`
- `template<typename A , typename I >`  
`mln\_meta\_accu\_result (A, typename I::value) compute(const Meta\_Accumulator< A > &a)`  
*Compute an accumulator onto the pixel values of the image input.*
- `template<typename I , typename J >`  
`void paste (const Image< I > &input, Image< J > &output)`  
*Paste the contents of image input into the image output.*

- `template<typename I, typename J >`  
`void paste\_without\_localization (const Image< I > &input, Image< J > &output)`  
*Paste the contents of image `input` into the image `output` without taking into account the localization of sites.*
  
- `template<typename I >`  
`void replace (Image< I > &input, const typename I::value &old_value, const typename I::value &new_value)`
- `template<typename I, typename V >`  
`mln::trait::ch_value< I, V >::ret saturate (const Image< I > &input, const V &min, const V &max)`
- `template<typename V, typename I >`  
`mln::trait::ch_value< I, V >::ret saturate (V v, const Image< I > &input)`
- `template<typename I >`  
`void saturate\_inplace (Image< I > &input, const typename I::value &min, const typename I::value &max)`
- `template<typename I >`  
`util::array< unsigned > sort\_offsets\_increasing (const Image< I > &input)`  
*Sort pixel offsets of the image `input` wrt increasing pixel values.*
  
- `template<typename I >`  
`p\_array< typename I::psite > sort\_psites\_decreasing (const Image< I > &input)`
- `template<typename I >`  
`p\_array< typename I::psite > sort\_psites\_increasing (const Image< I > &input)`
- `template<typename V, typename I >`  
`mln::trait::ch_value< I, V >::ret stretch (const V &v, const Image< I > &input)`
- `template<typename I, typename O >`  
`void to\_enc (const Image< I > &input, Image< O > &output)`
- `template<typename I1, typename I2, typename F >`  
`mln::trait::ch_value< I1, typename F::result >::ret transform (const Image< I1 > &input1, const Image< I2 > &input2, const Function\_vv2v< F > &f)`
- `template<typename I, typename F >`  
`mln::trait::ch_value< I, typename F::result >::ret transform (const Image< I > &input, const Function\_v2v< F > &f)`
- `template<typename I1, typename I2, typename F >`  
`void transform\_inplace (Image< I1 > &ima, const Image< I2 > &aux, const Function\_vv2v< F > &f)`
- `template<typename I, typename F >`  
`void transform\_inplace (Image< I > &ima, const Function\_v2v< F > &f)`
- `template<typename A, typename I >`  
`A::result update (Accumulator< A > &a, const Image< I > &input)`
- `template<typename V, typename I >`  
`mln::trait::ch_value< I, V >::ret wrap (const V &v, const Image< I > &input)`  
*Routine to wrap values such as  $0 \rightarrow 0$  and  $[1, lmax]$  maps to  $[1, Lmax]$  (using modulus).*
  
- `template<typename I, typename V >`  
`void fill\_with\_value (Image< I > &ima, const V &val)`  
*Fill the whole image `ima` with the single value `v`.*

### 9.28.1 Detailed Description

Namespace of image processing routines related to pixel data.

## 9.28.2 Function Documentation

**9.28.2.1** `template<typename I, typename O > void mln::data::abs ( const Image< I > & input, Image< O > & output ) [inline]`

Apply the absolute value (abs) function to image pixel values.

### Parameters

[in] *input* The input image.

[out] *output* The output image.

Definition at line 68 of file data/abs.hh.

References transform().

**9.28.2.2** `template<typename I > void mln::data::abs_inplace ( Image< I > & input ) [inline]`

Apply the absolute value (abs) function to image pixel values.

### Parameters

[in, out] *input* The input image.

Definition at line 80 of file data/abs.hh.

References apply().

**9.28.2.3** `template<typename I, typename F > void mln::data::apply ( Image< I > & input, const Function_v2v< F > & f ) [inline]`

Apply a function-object to the image *input*.

### Parameters

[in, out] *input* The input image.

[in] *f* The function-object.

This routine runs:

for all *p* of *input*,  $input(p) = f(input(p))$

This routine is equivalent to `data::transform(input, f, input)` but it is faster since a single iterator is required.

Definition at line 95 of file apply.hh.

Referenced by `abs_inplace()`, and `saturate_inplace()`.

**9.28.2.4** `template<typename A, typename I > A::result mln::data::compute ( const Accumulator< A > & a, const Image< I > & input ) [inline]`

Compute an accumulator onto the pixel values of the image *input*.

Be ware that the given accumulator won't be modified and won't store any result.

**Parameters**

- [in] *a* An accumulator.
- [in] *input* The input image.

**Returns**

The accumulator result.

It fully relies on [data::update](#).

Definition at line 93 of file data/compute.hh.

Referenced by `mln::labeled_image< I >::labeled_image()`, `mln::estim::mean()`, `mln::estim::min_max()`, and `mln::estim::sum()`.

**9.28.2.5** `template<typename A , typename I > A::result mln::data::compute ( Accumulator< A > & a, const Image< I > & input ) [inline]`

Compute an accumulator onto the pixel values of the image `input`.

**Parameters**

- [in, out] *a* An accumulator.
- [in] *input* The input image.

**Returns**

The accumulator result.

It fully relies on [data::update](#).

Definition at line 104 of file data/compute.hh.

**9.28.2.6** `template<typename V , typename I > mln::trait::ch_value< I, V >::ret mln::data::convert ( const V & v, const Image< I > & input ) [inline]`

Convert the image `input` by changing the value type.

**Parameters**

- [in] *v* A value of the destination type.
- [in] *input* The input image.

Definition at line 154 of file mln/data/convert.hh.

Referenced by `mln::morpho::watershed::superpose()`, and `mln::debug::superpose()`.

**9.28.2.7** `template<typename I , typename W , typename O > void mln::data::fast_median ( const Image< I > & input, const Window< W > & win, Image< O > & output ) [inline]`

Compute in `output` the median filter of image `input` by the window `win`.

**Parameters**

- [in] *input* The image to be filtered.
- [in] *win* The window.
- [in, out] *output* The output image.

**Precondition**

`input` and `output` have to be initialized.

Definition at line 167 of file `fast_median.hh`.

### 9.28.2.8 `template<typename I, typename D > void mln::data::fill ( Image< I > & ima, const D & data ) [inline]`

Fill the whole image `ima` with the data provided by `aux`.

**Parameters**

- [in, out] *ima* The image to be filled.
- [in] *data* The auxiliary data to fill the image `ima`.

**Precondition**

`ima` has to be initialized.

Definition at line 138 of file `data/fill.hh`.

Referenced by `mln::draw::box_plain()`, `mln::draw::dashed_line()`, `mln::topo::detach()`, `mln::util::display_branch()`, `mln::transform::distance_and_closest_point_geodesic()`, `mln::duplicate()`, `mln::make::edge_image()`, `mln::morpho::tree::filter::filter()`, `mln::transform::hough()`, `mln::registration::icp()`, `mln::accu::stat::histo3d_rgb< V >::init()`, `mln::graph::labeling()`, `mln::morpho::laplacian()`, `mln::make_debug_graph_image()`, `mln::morpho::tree::filter::max()`, `mln::geom::mesh_corner_point_area()`, `mln::geom::mesh_normal()`, `mln::morpho::meyer_wst()`, `mln::morpho::tree::filter::min()`, `mln::debug::mosaic()`, `mln::debug::slices_2d()`, `mln::morpho::watershed::superpose()`, `mln::debug::superpose()`, `mln::morpho::watershed::topological()`, and `mln::geom::translate()`.

### 9.28.2.9 `template<typename I, typename J > void mln::data::fill_with_image ( Image< I > & ima, const Image< J > & data ) [inline]`

Fill the image `ima` with the values of the image `data`.

**Parameters**

- [in, out] *ima* The image to be filled.
- [in] *data* The image.

**Warning**

The definition domain of `ima` has to be included in the one of `data`.

**Precondition**

`ima.domain <= data.domain`.

Definition at line 124 of file `fill_with_image.hh`.

**9.28.2.10** `template<typename I , typename V > void mln::data::fill_with_value ( Image< I > & ima, const V & val ) [inline]`

Fill the whole image `ima` with the single value `v`.

#### Parameters

[in, out] *ima* The image to be filled.

[in] *val* The value to assign to all sites.

#### Precondition

`ima` has to be initialized.

Definition at line 130 of file `fill_with_value.hh`.

Referenced by `mln::p_image< I >::clear()`.

**9.28.2.11** `template<typename I , typename W > mln::trait::concrete< I >::ret mln::data::median ( const Image< I > & input, const Window< W > & win )`

Compute in `output` the median filter of image `input` by the window `win`.

#### Parameters

[in] *input* The image to be filtered.

[in] *win* The window.

#### Precondition

`input` have to be initialized.

Definition at line 270 of file `median.hh`.

**9.28.2.12** `template<typename A , typename I > mln::data::mln_meta_accu_result ( A , typename I::value ) const [inline]`

Compute an accumulator onto the pixel values of the image `input`.

#### Parameters

[in] *a* A meta-accumulator.

[in] *input* The input image.

#### Returns

The accumulator result.

**9.28.2.13** `template<typename I , typename J > void mln::data::paste ( const Image< I > & input, Image< J > & output ) [inline]`

Paste the contents of image `input` into the image `output`.

**Parameters**

[in] *input* The input image providing pixels values.

[in, out] *output* The image in which values are assigned.

This routine runs:

for all  $p$  of `input`, `output(p) = input(p)`.

**Warning**

The definition domain of `input` has to be included in the one of `output`; so using [mln::safe\\_image](#) does not make pasting outside the output domain work.

**Precondition**

```
input.domain <= output.domain
```

Definition at line 137 of file `paste.hh`.

Referenced by `mln::make::image3d()`, `mln::draw::line()`, `mln::debug::mosaic()`, `mln::geom::rotate()`, `mln::debug::slices_2d()`, and `mln::labeling::superpose()`.

#### 9.28.2.14 `template<typename I , typename J > void mln::data::paste_without_localization ( const Image< I > & input, Image< J > & output ) [inline]`

Paste the contents of image `input` into the image `output` without taking into account the localization of sites.

**Parameters**

[in] *input* The input image providing pixels values.

[in, out] *output* The image in which values are assigned.

Definition at line 349 of file `paste_without_localization.hh`.

#### 9.28.2.15 `template<typename I > void mln::data::replace ( Image< I > & input, const typename I::value & old_value, const typename I::value & new_value )`

Replace `old_value` by `new_value` in the image `input`

**Parameters**

[in] *input* The input image.

[in] *old\_value* The value to be replaced...

[in] *new\_value* ...by this one.

Definition at line 88 of file `replace.hh`.

#### 9.28.2.16 `template<typename V , typename I > mln::trait::ch_value< I, V >::ret mln::data::saturate ( V v, const Image< I > & input ) [inline]`

Apply the saturate function to image pixel values.



**Parameters**

[in] *v* A value of the output type.

[in] *input* The input image.

The saturation is based on the min and max values of the output value type. This assumes that the range of values in the input image is larger than the one of the output image.

Definition at line 87 of file data/saturate.hh.

References transform().

**9.28.2.17** `template<typename I, typename V > mln::trait::ch_value< I, V >::ret  
mln::data::saturate ( const Image< I > & input, const V & min, const V & max )  
[inline]`

Apply the saturate function to image pixel values.

**Parameters**

[in] *input* The input image.

[in] *min* The minimum output value.

[in] *max* The maximum output value.

Definition at line 103 of file data/saturate.hh.

References transform().

**9.28.2.18** `template<typename I > void mln::data::saturate_inplace ( Image< I > & input, const  
typename I::value & min, const typename I::value & max ) [inline]`

Apply the saturate function to image pixel values.

**Parameters**

[in, out] *input* The input image.

[in] *min* The minimum output value.

[in] *max* The maximum output value

Definition at line 119 of file data/saturate.hh.

References apply().

**9.28.2.19** `template<typename I > util::array< unsigned > mln::data::sort_offsets_increasing ( const Image< I > & input ) [inline]`

Sort pixel offsets of the image *input* wrt increasing pixel values.

Definition at line 298 of file sort\_offsets.hh.

### 9.28.2.20 `template<typename I > p_array< typename I::psite > mln::data::sort_psites_decreasing ( const Image< I > & input ) [inline]`

Sort psites the image `input` through a function `f` to set the `output` image in decreasing way.

#### Parameters

[in] *input* The input image.

#### Precondition

`input.is_valid`

Definition at line 229 of file `sort_psites.hh`.

Referenced by `mln::morpho::tree::min_tree()`.

### 9.28.2.21 `template<typename I > p_array< typename I::psite > mln::data::sort_psites_increasing ( const Image< I > & input ) [inline]`

Sort psites the image `input` through a function `f` to set the `output` image in increasing way.

#### Parameters

[in] *input* The input image.

#### Precondition

`input.is_valid`

Definition at line 219 of file `sort_psites.hh`.

Referenced by `mln::morpho::tree::max_tree()`.

### 9.28.2.22 `template<typename V , typename I > mln::trait::ch_value< I, V >::ret mln::data::stretch ( const V & v, const Image< I > & input ) [inline]`

Stretch the values of `input` so that they can be stored in `output`.

#### Parameters

[in] *v* A value to set the output value type.

[in] *input* The input image.

#### Returns

A stretch image with values of the same type as `v`.

#### Precondition

`input.is_valid`

Definition at line 128 of file `stretch.hh`.

References `mln::data::impl::stretch()`.

**9.28.2.23** `template<typename I , typename O > void mln::data::to_enc ( const Image< I > & input, Image< O > & output ) [inline]`

Set the `output` image with the encoding values of the image `input` pixels.

#### Parameters

- [in] *input* The input image.
- [out] *output* The result image.

#### Precondition

```
output.domain >= input.domain
```

Definition at line 60 of file `data/to_enc.hh`.

References `transform()`.

**9.28.2.24** `template<typename I , typename F > mln::trait::ch_value< I, typename F::result >::ret mln::data::transform ( const Image< I > & input, const Function_v2v< F > & f ) [inline]`

Transform the image `input` through a function `f`.

#### Parameters

- [in] *input* The input image.
- [in] *f* The function.

This routine runs:

for all `p` of `input`, `output (p) = f ( input (p) )`.

Definition at line 202 of file `data/transform.hh`.

Referenced by `abs()`, `mln::logical::and_not()`, `mln::labeling::colorize()`, `mln::arith::diff_abs()`, `mln::labeling::fill_holes()`, `mln::linear::mln_ch_convolve()`, `mln::linear::mln_ch_convolve_grad()`, `mln::labeling::pack()`, `mln::labeling::pack_inplace()`, `mln::labeling::relabel()`, `saturate()`, `mln::data::impl::stretch()`, `to_enc()`, `mln::labeling::wrap()`, and `wrap()`.

**9.28.2.25** `template<typename I1 , typename I2 , typename F > mln::trait::ch_value< I1, typename F::result >::ret mln::data::transform ( const Image< I1 > & input1, const Image< I2 > & input2, const Function_vv2v< F > & f ) [inline]`

Transform two images `input1` `input2` through a function `f`.

#### Parameters

- [in] *input1* The 1st input image.
- [in] *input2* The 2nd input image.
- [in] *f* The function.

This routine runs:

for all `p` of `input`, `output (p) = f ( input1 (p) , input2 (p) )`.

Definition at line 219 of file `data/transform.hh`.

**9.28.2.26** `template<typename I1 , typename I2 , typename F > void mln::data::transform_inplace ( Image< I1 > & ima, const Image< I2 > & aux, const Function_vv2v< F > & f )`

Transform inplace the image *ima* with the image *aux* through a function *f*.

#### Parameters

[in] *ima* The image to be transformed.

[in] *aux* The auxiliary image.

[in] *f* The function.

This routine runs:

for all *p* of *ima*,  $ima(p) = f(ima(p), aux(p))$ .

Definition at line 502 of file `transform_inplace.hh`.

**9.28.2.27** `template<typename I , typename F > void mln::data::transform_inplace ( Image< I > & ima, const Function_v2v< F > & f )`

Transform inplace the image *ima* through a function *f*.

#### Parameters

[in, out] *ima* The image to be transformed.

[in] *f* The function.

This routine runs:

for all *p* of *ima*,  $ima(p) = f(ima(p))$ .

Definition at line 490 of file `transform_inplace.hh`.

Referenced by `mln::logical::and_inplace()`, `mln::logical::and_not_inplace()`, `mln::logical::not_inplace()`, `mln::logical::or_inplace()`, `mln::labeling::relabel_inplace()`, and `mln::logical::xor_inplace()`.

**9.28.2.28** `template<typename A , typename I > A::result mln::data::update ( Accumulator< A > & a, const Image< I > & input ) [inline]`

Update an accumulator with the pixel values of the image *input*.

#### Parameters

[in] *a* The accumulator.

[in] *input* The input image.

#### Returns

The accumulator result.

Definition at line 191 of file `update.hh`.

**9.28.2.29** `template<typename V, typename I> mln::trait::ch_value< I, V >::ret mln::data::wrap ( const V & v, const Image< I > & input )`

Routine to wrap values such as 0 -> 0 and [1, lmax] maps to [1, Lmax] (using modulus).

#### Parameters

- [in] *v* The target value type.
- [in] *input* Input image.

#### Returns

An image with wrapped values.

Definition at line 65 of file data/wrap.hh.

References transform().

## 9.29 mln::data::approx Namespace Reference

Namespace of image processing routines related to pixel levels with approximation.

### Namespaces

- namespace [impl](#)  
*Implementation namespace of [data::approx](#) namespace.*

### Functions

- `template<typename I> mln::trait::concrete< I >::ret median (const Image< I > &input, const win::rectangle2d &win)`
- `template<typename I> mln::trait::concrete< I >::ret median (const Image< I > &input, const win::octagon2d &win)`
- `template<typename I> mln::trait::concrete< I >::ret median (const Image< I > &input, const win::disk2d &win)`

#### 9.29.1 Detailed Description

Namespace of image processing routines related to pixel levels with approximation.

#### 9.29.2 Function Documentation

**9.29.2.1** `template<typename I> mln::trait::concrete< I >::ret mln::data::approx::median ( const Image< I > & input, const win::rectangle2d & win ) [inline]`

Compute in *output* an approximate of the median filter of image *input* by the 2D rectangle *win*.

#### Parameters

- [in] *input* The image to be filtered.

[in] *win* The rectangle.

The approximation is based on a vertical median ran after an horizontal median.

#### Precondition

`input` and `output` have to be initialized.

Definition at line 112 of file `approx/median.hh`.

Referenced by `median()`.

**9.29.2.2** `template<typename I> mln::trait::concrete< I >::ret mln::data::approx::median ( const Image< I > & input, const win::octagon2d & win ) [inline]`

Compute in `output` an approximate of the median filter of image `input` by the 2D octagon `win`.

#### Parameters

[in] *input* The image to be filtered.

[in] *win* The octagon.

The approximation is based on a vertical median and an horizontal median an two diagonal median.

#### Precondition

`input` and `output` have to be initialized.

Definition at line 159 of file `approx/median.hh`.

References `median()`.

**9.29.2.3** `template<typename I> mln::trait::concrete< I >::ret mln::data::approx::median ( const Image< I > & input, const win::disk2d & win ) [inline]`

Compute in `output` an approximate of the median filter of image `input` by the 2D disk `win`.

#### Parameters

[in] *input* The image to be filtered.

[in] *win* The disk.

The approximation is based on a vertical median and an horizontal median an two diagonal median.

#### Precondition

`input` and `output` have to be initialized.

Definition at line 132 of file `approx/median.hh`.

References `median()`.

## 9.30 mln::data::approx::impl Namespace Reference

Implementation namespace of [data::approx](#) namespace.

### 9.30.1 Detailed Description

Implementation namespace of [data::approx](#) namespace.

## 9.31 mln::data::impl Namespace Reference

Implementation namespace of data namespace.

### Namespaces

- namespace [generic](#)  
*Generic implementation namespace of data namespace.*

### Functions

- `template<typename I, typename J >`  
`void paste\_without\_localization\_fast (const Image< I > &input_, Image< J > &output_)`  
*Paste data to an image without using localization. Performs a point-wise copy.*
- `template<typename I, typename J >`  
`void paste\_without\_localization\_fastest (const Image< I > &input_, Image< J > &output_)`  
*Paste data to an image without using localization. Performs a one-block memory copy.*
- `template<typename I, typename J >`  
`void paste\_without\_localization\_lines (const Image< I > &input_, Image< J > &output_)`  
*Paste data to an image without using localization. Performs a line-per-line memory copy.*
- `template<typename V, typename I >`  
`mln::trait::ch_value< I, V >::ret stretch (const V &v, const Image< I > &input)`  
*Generic implementation of [data::stretch](#).*
- `template<typename I, typename F >`  
`void transform\_inplace\_lowq (Image< I > &input_, const Function\_v2v< F > &f_)`  
*Specialized implementation.*
- `template<typename A, typename I >`  
`A::result update\_fastest (Accumulator< A > &a_, const Image< I > &input_)`  
*Fastest implementation of [data::update](#).*

### 9.31.1 Detailed Description

Implementation namespace of data namespace.

## 9.31.2 Function Documentation

### 9.31.2.1 `template<typename I , typename J > void mln::data::impl::paste_without_localization_fast ( const Image< I > & input_ , Image< J > & output_ ) [inline]`

Paste data to an image without using localization. Performs a point-wise copy.

`input` and `output` must have both the following properties:

- `mln::trait::image::value_alignment::with_grid`
- `mln::trait::image::value_storage::one_block`
- `mln::trait::image::value_access::direct`
- `mln::trait::image::ext_domain::some`

They must also fulfill the following conditions:

- Same domain size.

Definition at line 220 of file `paste_without_localization.hh`.

### 9.31.2.2 `template<typename I , typename J > void mln::data::impl::paste_without_localization_fastest ( const Image< I > & input_ , Image< J > & output_ ) [inline]`

Paste data to an image without using localization. Performs a one-block memory copy.

`input` and `output` must have both the following properties:

- `mln::trait::image::value_alignment::with_grid`
- `mln::trait::image::value_storage::one_block`
- `mln::trait::image::value_access::direct`
- `mln::trait::image::ext_domain::some`

They must also fulfill the following conditions:

- Same border size.
- Same domain size.
- Same value type.

Definition at line 142 of file `paste_without_localization.hh`.



**9.31.2.3** `template<typename I , typename J > void mln::data::impl::paste_without_localization_lines ( const Image< I > & input_, Image< J > & output_ ) [inline]`

Paste data to an image without using localization. Performs a line-per-line memory copy.

`input` and `output` must have both the following properties:

- `mln::trait::image::value_alignment::with_grid`
- `mln::trait::image::value_storage::one_block`
- `mln::trait::image::value_access::direct`
- `mln::trait::image::ext_domain::some`

They must also fulfill the following conditions:

- Same domain size.
- Same value type.

Definition at line 179 of file `paste_without_localization.hh`.

**9.31.2.4** `template<typename V , typename I > mln::trait::ch_value< I , V >::ret mln::data::impl::stretch ( const V & v, const Image< I > & input ) [inline]`

Generic implementation of [data::stretch](#).

#### Parameters

- [in] `v` A value to set the output value type.
- [in] `input` The input image.

#### Returns

A stretch image with values of the same type as `v`.

Definition at line 83 of file `stretch.hh`.

References `mln::initialize()`, `mln::estim::min_max()`, and `mln::data::transform()`.

Referenced by `mln::data::stretch()`.

**9.31.2.5** `template<typename I , typename F > void mln::data::impl::transform_inplace_lowq ( Image< I > & input_, const Function_v2v< F > & f_ )`

Specialized implementation.

Definition at line 203 of file `transform_inplace.hh`.

### 9.31.2.6 `template<typename A , typename I > A ::result mln::data::impl::update_fastest ( Accumulator< A > & a_ , const Image< I > & input_ ) [inline]`

Fastest implementation of `data::update`.

#### Parameters

- [in] `a_` The accumulator.
- [in] `input_` The input image.

#### Returns

The accumulator result.

Definition at line 129 of file `update.hh`.

## 9.32 `mln::data::impl::generic` Namespace Reference

Generic implementation namespace of data namespace.

### Functions

- `template<typename I , typename J > void fill_with_image (Image< I > &ima_ , const Image< J > &data_)`  
*Generic implementation.*
- `template<typename I , typename V > void fill_with_value (Image< I > &ima_ , const V &val)`  
*Fill the whole image `ima` with the single value `v`.*
- `template<typename I , typename J > void paste (const Image< I > &input_ , Image< J > &output_)`  
*Generic implementation of `data::paste`.*
- `template<typename I , typename F > mln::trait::ch_value< I , typename F::result >::ret transform (const Image< I > &input_ , const Function_v2v< F > &f_)`  
*Generic implementation of `data::transform`.*
- `template<typename I1 , typename I2 , typename F > mln::trait::ch_value< I1 , typename F::result >::ret transform (const Image< I1 > &input1_ , const Image< I2 > &input2_ , const Function_vv2v< F > &f_)`  
*Generic implementation of `data::transform`.*
- `template<typename I1 , typename I2 , typename F > void transform_inplace (Image< I1 > &ima_ , const Image< I2 > &aux_ , const Function_vv2v< F > &f_)`  
*Generic implementation of `transform_inplace`.*
- `template<typename I , typename F > void transform_inplace (Image< I > &ima_ , const Function_v2v< F > &f_)`

*Generic implementation of transform\_inplace.*

- `template<typename A , typename I >`  
`A::result update (Accumulator< A > &a_, const Image< I > &input_)`  
*Generic implementation of data::update.*

### 9.32.1 Detailed Description

Generic implementation namespace of data namespace.

### 9.32.2 Function Documentation

**9.32.2.1** `template<typename I , typename J > void mln::data::impl::generic::fill_with_image (`  
`Image< I > & ima_ , const Image< J > & data_ )`

Generic implementation.

#### Parameters

- `[in, out] ima_` The image to be filled.
- `[in] data_` The image.

Definition at line 100 of file fill\_with\_image.hh.

**9.32.2.2** `template<typename I , typename V > void mln::data::impl::generic::fill_with_value (`  
`Image< I > & ima_ , const V & val )`

Fill the whole image `ima` with the single value `v`.

#### Parameters

- `[in, out] ima_` The image to be filled.
- `[in] val` The value to assign to all sites.

#### Precondition

`ima` has to be initialized.

Definition at line 103 of file fill\_with\_value.hh.

**9.32.2.3** `template<typename I , typename J > void mln::data::impl::generic::paste ( const`  
`Image< I > & input_ , Image< J > & output_ ) [inline]`

Generic implementation of `data::paste`.

#### Parameters

- `[in] input_` The input image providing pixels values.
- `[in, out] output_` The image in which values are assigned.

Definition at line 111 of file paste.hh.

**9.32.2.4** `template<typename I , typename F > mln::trait::ch_value< I , typename F ::result >::ret mln::data::impl::generic::transform ( const Image< I > & input_ , const Function_v2v< F > & f_ )`

Generic implementation of [data::transform](#).

#### Parameters

[in] *input\_* The input image.

[in] *f\_* The function.

Definition at line 137 of file data/transform.hh.

References `mln::initialize()`.

**9.32.2.5** `template<typename I1 , typename I2 , typename F > mln::trait::ch_value< I1 , typename F ::result >::ret mln::data::impl::generic::transform ( const Image< I1 > & input1_ , const Image< I2 > & input2_ , const Function_vv2v< F > & f_ )`

Generic implementation of [data::transform](#).

#### Parameters

[in] *input1\_* The 1st input image.

[in] *input2\_* The 2nd input image.

[in] *f\_* The function.

Definition at line 166 of file data/transform.hh.

References `mln::initialize()`.

**9.32.2.6** `template<typename I1 , typename I2 , typename F > void mln::data::impl::generic::transform_inplace ( Image< I1 > & ima_ , const Image< I2 > & aux_ , const Function_vv2v< F > & f_ )`

Generic implementation of `transform_inplace`.

#### Parameters

[in] *ima\_* The image to be transformed.

[in] *aux\_* The auxiliary image.

[in] *f\_* The function.

Definition at line 176 of file transform\_inplace.hh.

**9.32.2.7** `template<typename I , typename F > void mln::data::impl::generic::transform_inplace ( Image< I > & ima_ , const Function_v2v< F > & f_ )`

Generic implementation of `transform_inplace`.

#### Parameters

[in, out] *ima\_* The image to be transformed.

[in] *f\_* The function.

Definition at line 149 of file transform\_inplace.hh.

**9.32.2.8** `template<typename A , typename I > A ::result mln::data::impl::generic::update ( Accumulator< A > & a_ , const Image< I > & input_ ) [inline]`

Generic implementation of [data::update](#).

#### Parameters

[in] *a\_* The accumulator.

[in] *input\_* The input image.

#### Returns

The accumulator result.

Definition at line 100 of file update.hh.

## 9.33 mln::data::naive Namespace Reference

Namespace of image processing routines related to pixel levels with naive approach.

### Namespaces

- namespace [impl](#)  
*Implementation namespace of [data::naive](#) namespace.*

### Functions

- `template<typename I , typename W , typename O > void median (const Image< I > &input, const Window< W > &win, Image< O > &output)`  
*Compute in output the median filter of image input by the window win.*

#### 9.33.1 Detailed Description

Namespace of image processing routines related to pixel levels with naive approach.

#### 9.33.2 Function Documentation

**9.33.2.1** `template<typename I , typename W , typename O > void mln::data::naive::median ( const Image< I > & input , const Window< W > & win , Image< O > & output ) [inline]`

Compute in output the median filter of image input by the window win.

**Parameters**

- [in] *input* The image to be filtered.
- [in] *win* The window.
- [in, out] *output* The output image.

This is a NAIVE version for test / comparison purpose so do NOT use it.

**Precondition**

`input` and `output` have to be initialized.

**See also**

[mln::data::median](#)

Definition at line 99 of file naive/median.hh.

## 9.34 mln::data::naive::impl Namespace Reference

Implementation namespace of [data::naive](#) namespace.

### 9.34.1 Detailed Description

Implementation namespace of [data::naive](#) namespace.

## 9.35 mln::debug Namespace Reference

Namespace of routines that help to debug.

**Namespaces**

- namespace [impl](#)  
*Implementation namespace of debug namespace.*

**Functions**

- `template<typename I , typename G , typename F >`  
void [draw\\_graph](#) ([Image](#)< I > &ima, const [p\\_vertices](#)< G, F > &pv, typename I::value vcolor, typename I::value ecolor)  
*Draw an image `ima` from a `mln::p_vertices` `pv`, with value `vcolor` for vertices, value `ecolor` for edges and 0 for the background.*
- `template<typename I , typename G , typename F , typename V , typename E >`  
void [draw\\_graph](#) ([Image](#)< I > &ima, const [p\\_vertices](#)< G, F > &pv, const [Function](#)< V > &vcolor\_f\_, const [Function](#)< E > &ecolor\_f\_)  
*Draw an image `ima` from a `mln::p_vertices` `pv`.*

- `template<typename I, typename G, typename F, typename V, typename E >`  
`void draw_graph (Image< I > &ima, const p_vertices< util::line_graph< G >, F > &pv, const`  
`Function< V > &vcolor_f_, const Function< E > &ecolor_f_)`  
*Draw an image `ima` from a `mln::p_vertices` `pv`.*
- `std::string filename` (`const std::string &filename, int id`)  
*Constructs and returns a formatted output file name.*
- signed short `format` (`signed char v`)  
*Format a signed char to print it properly, i.e., like an integer value.*
- unsigned short `format` (`unsigned char v`)  
*Format an unsigned char to print it properly, i.e., like an integer value.*
- `template<typename T >`  
`const T & format` (`const T &v`)  
*Default version for formatting a value is a no-op.*
- `char format` (`bool v`)  
*Format a Boolean to print it nicely: "|" for true and "-" for false.*
- `template<typename I >`  
`void iota` (`Image< I > &input, unsigned base_index`)
- `template<typename I >`  
`mln::trait::concrete< I >::ret mosaic` (`const util::array< I > &input, unsigned n_horizontal, const`  
`typename I::value &bg`)  
*Create a single image from an array of image.*
- `template<typename I >`  
`void println` (`const std::string &msg, const Image< I > &input`)  
*Print the message `msg` and the image `input` on the standard output.*
- `template<typename I >`  
`void println` (`const Image< I > &input`)  
*Print the image `input` on the standard output.*
- `template<typename I >`  
`void println_with_border` (`const Image< I > &input`)  
*Print the image `input` on the standard output.*
- `void put_word` (`image2d< char > &inout, const point2d &word_start, const std::string &word`)  
*Put the word starting at location `word_start` in the image `inout`.*
- `template<typename I >`  
`image2d< typename I::value > slices_2d` (`const Image< I > &input, unsigned n_horizontal, un-`  
`signed n_vertical, const typename I::value &bg`)  
*Create a 2D image of the slices of the 3D image `input`.*

- `template<typename I >`  
`image2d< typename I::value > slices_2d (const Image< I > &input, float ratio_hv, const typename I::value &bg)`

*Create a 2D image of the slices of the 3D image input.*

- `template<typename I , typename J >`  
`mln::trait::ch_value< I, value::rgb8 >::ret superpose (const Image< I > &input, const Image< J > &object)`
- `template<typename I , typename J >`  
`mln::trait::ch_value< I, value::rgb8 >::ret superpose (const Image< I > &input_, const Image< J > &object_, const value::rgb8 &object_color)`

*Superpose two images.*

- `template<typename I >`  
`void z_order (Image< I > &input)`

### 9.35.1 Detailed Description

Namespace of routines that help to debug.

### 9.35.2 Function Documentation

- 9.35.2.1** `template<typename I , typename G , typename F > void mln::debug::draw_graph ( Image< I > & ima, const p_vertices< G, F > & pv, typename I::value vcolor, typename I::value ecolor ) [inline]`

Draw an image `ima` from a `mln::p_vertices` `pv`, with value `vcolor` for vertices, value `ecolor` for edges and 0 for the background.

Definition at line 142 of file `draw_graph.hh`.

References `mln::p_vertices< G, F >::graph()`, and `mln::draw::line()`.

Referenced by `mln::make_debug_graph_image()`.

- 9.35.2.2** `template<typename I , typename G , typename F , typename V , typename E > void mln::debug::draw_graph ( Image< I > & ima, const p_vertices< G, F > & pv, const Function< V > & vcolor_f_, const Function< E > & ecolor_f_ ) [inline]`

Draw an image `ima` from a `mln::p_vertices` `pv`.

Colors for vertices are defined through `vcolor_f_`. Colors for edges are defined though `ecolor_f_`.

Definition at line 173 of file `draw_graph.hh`.

References `mln::draw::box_plain()`, `mln::box< P >::crop_wrt()`, `mln::p_vertices< G, F >::graph()`, and `mln::draw::line()`.



**9.35.2.3** `template<typename I , typename G , typename F , typename V , typename E > void mln::debug::draw_graph ( Image< I > & ima, const p_vertices< util::line_graph< G >, F > & pv, const Function< V > & vcolor_f_, const Function< E > & ecolor_f_ ) [inline]`

Draw an image `ima` from a `mln::p_vertices` `pv`.

Colors for vertices are defined through `vcolor_f_`. Colors for edges are defined through `ecolor_f_`.

Definition at line 211 of file `draw_graph.hh`.

References `mln::p_line2d::begin()`, `mln::p_line2d::end()`, `mln::p_vertices< G, F >::graph()`, and `mln::draw::line()`.

**9.35.2.4** `std::string mln::debug::filename ( const std::string & filename, int id = -1 ) [inline]`

Constructs and returns a formatted output file name.

The file name is formatted as follow:

`'filename_prefix_'id_'filename'`

Where:

- `'filename_prefix'` can be set through the global variable `debug::internal::filename_prefix`.

`'postfix_id'` is autoincremented by default. Its value can be forced.

- `'filename'` is the given filename

Definition at line 86 of file `filename.hh`.

**9.35.2.5** `signed short mln::debug::format ( signed char v ) [inline]`

Format a signed char to print it properly, i.e., like an integer value.

Definition at line 78 of file `format.hh`.

**9.35.2.6** `unsigned short mln::debug::format ( unsigned char v ) [inline]`

Format an unsigned char to print it properly, i.e., like an integer value.

Definition at line 85 of file `format.hh`.

**9.35.2.7** `template<typename T > const T & mln::debug::format ( const T & v ) [inline]`

Default version for formatting a value is a no-op.

Definition at line 64 of file `format.hh`.

Referenced by `mln::value::operator<<()`, and `mln::Gpoint< E >::operator<<()`.

**9.35.2.8 char mln::debug::format ( bool v ) [inline]**

Format a Boolean to print it nicely: "|" for true and "-" for false.

Definition at line 71 of file format.hh.

**9.35.2.9 template<typename I> void mln::debug::iota ( Image< I > & input, unsigned base\_index ) [inline]**

Fill the image `input` with successive values.

**Parameters**

[in, out] *input* The image in which values are assigned.

Definition at line 88 of file debug/iota.hh.

References `iota()`.

Referenced by `iota()`.

**9.35.2.10 template<typename I> mln::trait::concrete< I >::ret mln::debug::mosaic ( const util::array< I > & input, unsigned n\_horizontal, const typename I::value & bg ) [inline]**

Create a single image from an array of image.

The size of the output image is defined by:

$$\text{width} = n\_horizontal * \max(\text{input}[i].\text{ncols}()) \quad \text{height} = (\text{input.size}() / n\_horizontal) * \max(\text{input}[i].\text{nrows}())$$
**Returns**

a single image where all the input images are displayed as a mosaic.

Definition at line 77 of file mosaic.hh.

References `mln::apply_p2p()`, `mln::data::fill()`, and `mln::data::paste()`.

**9.35.2.11 template<typename I> void mln::debug::println ( const std::string & msg, const Image< I > & input )**

Print the message `msg` and the image `input` on the standard output.

Definition at line 98 of file println.hh.

References `println()`.

**9.35.2.12 template<typename I> void mln::debug::println ( const Image< I > & input ) [inline]**

Print the image `input` on the standard output.

Definition at line 87 of file println.hh.

References `mln::geom::bbox()`.

Referenced by `println()`.

**9.35.2.13** `template<typename I> void mln::debug::println_with_border ( const Image< I> & input ) [inline]`

Print the image `input` on the standard output.

Definition at line 79 of file `println_with_border.hh`.

References `mln::geom::bbox()`.

**9.35.2.14** `void mln::debug::put_word ( image2d< char> & inout, const point2d & word_start, const std::string & word ) [inline]`

Put the `word` starting at location `word_start` in the image `inout`.

Definition at line 55 of file `put_word.hh`.

References `mln::image2d< T>::has()`, and `mln::point< G, C>::last_coord()`.

**9.35.2.15** `template<typename I> image2d< typename I::value> mln::debug::slices_2d ( const Image< I> & input, unsigned n_horizontal, unsigned n_vertical, const typename I::value & bg ) [inline]`

Create a 2D image of the slices of the 3D image `input`.

Definition at line 79 of file `slices_2d.hh`.

References `mln::apply_p2p()`, `mln::data::fill()`, and `mln::data::paste()`.

Referenced by `slices_2d()`.

**9.35.2.16** `template<typename I> image2d< typename I::value> mln::debug::slices_2d ( const Image< I> & input, float ratio_hv, const typename I::value & bg )`

Create a 2D image of the slices of the 3D image `input`.

Definition at line 162 of file `slices_2d.hh`.

References `slices_2d()`.

**9.35.2.17** `template<typename I, typename J> mln::trait::ch_value< I, value::rgb8>::ret mln::debug::superpose ( const Image< I> & input, const Image< J> & object )`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 106 of file `debug/superpose.hh`.

References `mln::literal::red`, and `superpose()`.

**9.35.2.18** `template<typename I, typename J> mln::trait::ch_value< I, value::rgb8>::ret mln::debug::superpose ( const Image< I> & input_, const Image< J> & object_, const value::rgb8 & object_color )`

Superpose two images.

**Parameters**

- [in] *input\_* An image. Its value type must be convertible toward `value::rgb8` thanks to a conversion operator or `convert::from_to`.
- [in] *object\_* A scalar or labeled image. Objects used for superposition. have their pixel values different from 0.
- [in] *object\_color* The color used to draw the objects in `object_`.

**Precondition**

`input_` and `object_` must have the same domain.

**Returns**

A color image.

Definition at line 79 of file `debug/superpose.hh`.

References `mln::data::convert()`, `mln::data::fill()`, and `mln::literal::zero`.

Referenced by `superpose()`.

**9.35.2.19** `template<typename I> void mln::debug::z_order ( Image<I> & input ) [inline]`

Fill the image `input` with Z-order (curve) values.

**Parameters**

- [in, out] *input* The image in which values are assigned.

Reference: [http://en.wikipedia.org/wiki/Z-order\\_\(curve\)](http://en.wikipedia.org/wiki/Z-order_(curve))

Definition at line 142 of file `z_order.hh`.

**9.36 mln::debug::impl Namespace Reference**

Implementation namespace of `debug` namespace.

**9.36.1 Detailed Description**

Implementation namespace of `debug` namespace.

**9.37 mln::def Namespace Reference**

Namespace for core definitions.

**Typedefs**

- typedef short `coord`

*Definition of the default coordinate type: 'short'.*

- typedef float [coordf](#)

*Definition of the floating coordinate type.*

## Enumerations

- enum

*Definition of the number of bits of the low quantization threshold.*

### 9.37.1 Detailed Description

Namespace for core definitions.

### 9.37.2 Typedef Documentation

#### 9.37.2.1 typedef short mln::def::coord

Definition of the default coordinate type: 'short'.

Definition at line 43 of file coord.hh.

#### 9.37.2.2 typedef float mln::def::coordf

Definition of the floating coordinate type.

Definition at line 41 of file coordf.hh.

### 9.37.3 Enumeration Type Documentation

#### 9.37.3.1 anonymous enum

Definition of the number of bits of the low quantization threshold.

Definition at line 43 of file low\_quant\_nbits.hh.

## 9.38 mln::display Namespace Reference

Namespace of routines that help to display images.

### Namespaces

- namespace [impl](#)

*Implementation namespace of display namespace.*

### 9.38.1 Detailed Description

Namespace of routines that help to display images.

## 9.39 mln::display::impl Namespace Reference

Implementation namespace of display namespace.

### Namespaces

- namespace [generic](#)  
*Generic implementation namespace of display namespace.*

### 9.39.1 Detailed Description

Implementation namespace of display namespace.

## 9.40 mln::display::impl::generic Namespace Reference

Generic implementation namespace of display namespace.

### 9.40.1 Detailed Description

Generic implementation namespace of display namespace.

## 9.41 mln::doc Namespace Reference

The namespace [mln::doc](#) is only for documentation purpose.

### Classes

- struct [Accumulator](#)  
*Documentation class for [mln::Accumulator](#).*
- struct [Box](#)  
*Documentation class for [mln::Box](#).*
- struct [Dpoint](#)  
*Documentation class for [mln::Dpoint](#).*
- struct [Fastest\\_Image](#)  
*Documentation class for the concept of images that have the speed property set to "fastest".*

- struct [Generalized\\_Pixel](#)  
*Documentation class for mln::Generalized\_Pixel.*
- struct [Image](#)  
*Documentation class for mln::Image.*
- struct [Iterator](#)  
*Documentation class for mln::Iterator.*
- struct [Neighborhood](#)  
*Documentation class for mln::Neighborhood.*
- struct [Object](#)  
*Documentation class for mln::Object.*
- struct [Pixel\\_Iterator](#)  
*Documentation class for mln::Iterator.*
- struct [Point\\_Site](#)  
*Documentation class for mln::Point\_Site.*
- struct [Site\\_Iterator](#)  
*Documentation class for mln::Site\_Iterator.*
- struct [Site\\_Set](#)  
*Documentation class for mln::Site\_Set.*
- struct [Value\\_Iterator](#)  
*Documentation class for mln::Value\_Iterator.*
- struct [Value\\_Set](#)  
*Documentation class for mln::Value\_Set.*
- struct [Weighted\\_Window](#)  
*Documentation class for mln::Weighted\_Window.*
- struct [Window](#)  
*Documentation class for mln::Window.*

### 9.41.1 Detailed Description

The namespace `mln::doc` is only for documentation purpose. Since concepts are not yet part of the C++ Standard, they are not explicitly expressed in code. Their documentation is handled by their respective ghost class, located in this namespace.

#### Warning

The ghost classes located in `mln::doc` should not be used by the client.

## 9.42 mln::draw Namespace Reference

Namespace of drawing routines.

### Functions

- `template<typename I, typename B >`  
`void box (Image< I > &ima, const Box< B > &b, const typename I::value &v)`
- `template<typename I, typename B >`  
`void box\_plain (Image< I > &ima, const Box< B > &b, const typename I::value &v)`
- `template<typename I >`  
`void dashed\_line (Image< I > &ima, const typename I::psite &beg, const typename I::psite &end, const typename I::value &v)`
- `template<typename I >`  
`void line (Image< I > &ima, const typename I::psite &beg, const typename I::psite &end, const typename I::value &v)`
- `template<typename I >`  
`void plot (Image< I > &ima, const typename I::point &p, const typename I::value &v)`
- `template<typename I >`  
`void polygon (Image< I > &ima, const p\_array< typename I::site > &par, const typename I::value &v, unsigned output_ratio)`
- `template<typename I, typename S >`  
`void site\_set (Image< I > &ima, const Site\_Set< S > &s, const typename I::value &v, unsigned output_ratio=1)`

### 9.42.1 Detailed Description

Namespace of drawing routines.

### 9.42.2 Function Documentation

**9.42.2.1** `template<typename I, typename B > void mln::draw::box ( Image< I > & ima, const Box< B > & b, const typename I::value & v )` [`inline`]

Draw a box at value *v* in image *ima*

#### Parameters

- [in, out] *ima* The image to be drawn.
- [in] *b* the box to draw.
- [in] *v* The value to assign to all drawn pixels.

#### Precondition

- ima* has to be initialized.
- ima* has *beg*.
- ima* has *end*.

Definition at line 72 of file draw/box.hh.

References `line()`.



**9.42.2.2** `template<typename I, typename B > void mln::draw::box_plain ( Image< I > & ima, const Box< B > & b, const typename I::value & v ) [inline]`

Draw a plain box at value *v* in image *ima*

#### Parameters

- [in, out] *ima* The image to be drawn.
- [in] *b* the box to draw.
- [in] *v* The value to assign to all drawn pixels.

#### Precondition

- ima* has to be initialized.
- ima* has beg.
- ima* has end.

Definition at line 71 of file box\_plain.hh.

References mln::data::fill().

Referenced by mln::debug::draw\_graph().

**9.42.2.3** `template<typename I > void mln::draw::dashed_line ( Image< I > & ima, const typename I::psite & beg, const typename I::psite & end, const typename I::value & v ) [inline]`

Draw a dashed line at level *v* in image *ima* between the points *beg* and *end*.

#### Parameters

- [in, out] *ima* The image to be drawn.
- [in] *beg* The start point to drawn dashed\_line.
- [in] *end* The end point to drawn dashed\_line.
- [in] *v* The value to assign to all drawn pixels.

#### Precondition

- ima* has to be initialized.
- ima* has beg.
- ima* has end.

Definition at line 91 of file dashed\_line.hh.

References mln::data::fill().

**9.42.2.4** `template<typename I > void mln::draw::line ( Image< I > & ima, const typename I::psite & beg, const typename I::psite & end, const typename I::value & v ) [inline]`

Draw a line at level *v* in image *ima* between the points *beg* and *end*.

#### Parameters

- [in, out] *ima* The image to be drawn.

- [in] *beg* The start point to drawn line.
- [in] *end* The end point to drawn line.
- [in] *v* The value to assign to all drawn pixels.

**Precondition**

*ima* has to be initialized.  
*ima* has *beg*.  
*ima* has *end*.

Definition at line 72 of file draw/line.hh.

References `mln::data::paste()`.

Referenced by `box()`, `mln::debug::draw_graph()`, and `polygon()`.

#### 9.42.2.5 `template<typename I> void mln::draw::plot ( Image< I > & ima, const typename I::point & p, const typename I::value & v )`

Plot a point at level *v* in image *ima*

**Parameters**

- [in, out] *ima* The image to be drawn.
- [in] *p* The point to be plotted.
- [in] *v* The value to assign to all drawn pixels.

**Precondition**

*ima* has to be initialized.  
*ima* has *p*.

#### 9.42.2.6 `template<typename I> void mln::draw::polygon ( Image< I > & ima, const p_array< typename I::site > & par, const typename I::value & v, unsigned output_ratio )`

Draw a polygon at level *v* in image *ima*.

**Parameters**

- [in, out] *ima* The image to be drawn.
- [in] *par* The polygon site set.
- [in] *v* The value to assign to all drawn pixels.

**Precondition**

*ima* has to be initialized.

Definition at line 70 of file polygon.hh.

References `line()`.

**9.42.2.7** `template<typename I, typename S > void mln::draw::site_set ( Image< I > & ima, const Site_Set< S > & s, const typename I::value & v, unsigned output_ratio = 1 )`

Draw a sites with value  $v$  in image  $ima$

#### Parameters

[in, out] **ima** The image to be drawn.

[in] **b** the site set to draw.

[in] **v** The value to assign to all drawn pixels.

[in] **output\_ratio** size ratio between output image and the image from which the bboxes were calculated.

#### Precondition

$s$  is included in  $ima$  domain.

Definition at line 65 of file mln/draw/site\_set.hh.

## 9.43 mln::estim Namespace Reference

Namespace of estimation materials.

### Functions

- `template<typename I > mln::value::props< typename I::value >::sum mean (const Image< I > &input)`  
*Compute the mean value of the pixels of image input.*
- `template<typename S, typename I, typename M > void mean (const Image< I > &input, M &result)`  
*Compute the mean value of the pixels of image input.*
- `template<typename I > void min_max (const Image< I > &input, typename I::value &min, typename I::value &max)`  
*Compute the min and max values of the pixels of image input.*
- `template<typename I > mln::value::props< typename I::value >::sum sum (const Image< I > &input)`  
*Compute the sum value of the pixels of image input.*
- `template<typename I, typename S > void sum (const Image< I > &input, S &result)`  
*Compute the sum value of the pixels of image input.*

### 9.43.1 Detailed Description

Namespace of estimation materials.

## 9.43.2 Function Documentation

### 9.43.2.1 `template<typename I> mln::value::props< typename I::value >::sum mln::estim::mean ( const Image< I > & input ) [inline]`

Compute the mean value of the pixels of image `input`.

#### Parameters

[in] *input* The image.

#### Returns

The mean value.

Definition at line 68 of file `estim/mean.hh`.

References `mln::data::compute()`.

### 9.43.2.2 `template<typename S , typename I , typename M > void mln::estim::mean ( const Image< I > & input, M & result ) [inline]`

Compute the mean value of the pixels of image `input`.

#### Parameters

[in] *input* The image.

[out] *result* The mean value.

The free parameter `S` is the type used to compute the summation.

Definition at line 76 of file `estim/mean.hh`.

References `mln::data::compute()`.

### 9.43.2.3 `template<typename I> void mln::estim::min_max ( const Image< I > & input, typename I::value & min, typename I::value & max ) [inline]`

Compute the min and max values of the pixels of image `input`.

#### Parameters

[in] *input* The image.

[out] *min* The minimum pixel value of `input`.

[out] *max* The maximum pixel value of `input`.

Definition at line 61 of file `estim/min_max.hh`.

References `mln::data::compute()`.

Referenced by `mln::data::impl::stretch()`, and `mln::make::voronoi()`.

#### 9.43.2.4 `template<typename I> mln::value::props< typename I::value >::sum mln::estim::sum ( const Image< I > & input ) [inline]`

Compute the sum value of the pixels of image `input`.

##### Parameters

[in] *input* The image.

##### Returns

The sum value.

Definition at line 67 of file `estim/sum.hh`.

References `mln::data::compute()`.

#### 9.43.2.5 `template<typename I, typename S> void mln::estim::sum ( const Image< I > & input, S & result ) [inline]`

Compute the sum value of the pixels of image `input`.

##### Parameters

[in] *input* The image.

[out] *result* The sum value.

Definition at line 75 of file `estim/sum.hh`.

References `mln::data::compute()`.

## 9.44 mln::extension Namespace Reference

Namespace of extension tools.

### Functions

- `template<typename I, typename W> void adjust (const Image< I > &ima, const Window< W > &win)`  
*Adjust the domain extension of image `ima` with the size of the window `win`.*
- `template<typename I, typename W> void adjust (const Image< I > &ima, const Weighted_Window< W > &wwin)`  
*Adjust the domain extension of image `ima` with the size of the weighted window `wwin`.*
- `template<typename I> void adjust (const Image< I > &ima, unsigned delta)`  
*Adjust the domain extension of image `ima` with the size `delta`.*
- `template<typename I, typename N> void adjust (const Image< I > &ima, const Neighborhood< N > &nbh)`

Adjust the domain extension of image *ima* with the size of the neighborhood *nbh*.

- `template<typename I, typename W >`  
`void adjust\_duplicate (const Image< I > &ima, const Window< W > &win)`  
*Adjust then duplicate.*
- `template<typename I, typename W >`  
`void adjust\_fill (const Image< I > &ima, const Window< W > &win, const typename I::value &val)`  
*Adjust then fill.*
- `template<typename I >`  
`void duplicate (const Image< I > &ima)`  
*Assign the contents of the domain extension by duplicating the values of the inner boundary of image *ima*.*
- `template<typename I >`  
`void fill (const Image< I > &ima, const typename I::value &val)`

### 9.44.1 Detailed Description

Namespace of extension tools.

### 9.44.2 Function Documentation

#### 9.44.2.1 `template<typename I, typename W > void mln::extension::adjust ( const Image< I > & ima, const Window< W > & win )`

Adjust the domain extension of image *ima* with the size of the window *win*.

Definition at line 89 of file `extension/adjust.hh`.

References `mln::geom::delta()`.

Referenced by `adjust()`, `adjust_duplicate()`, and `adjust_fill()`.

#### 9.44.2.2 `template<typename I, typename W > void mln::extension::adjust ( const Image< I > & ima, const Weighted\_Window< W > & wwin )`

Adjust the domain extension of image *ima* with the size of the weighted window *wwin*.

Definition at line 97 of file `extension/adjust.hh`.

References `adjust()`, and `mln::geom::delta()`.

#### 9.44.2.3 `template<typename I > void mln::extension::adjust ( const Image< I > & ima, unsigned delta )`

Adjust the domain extension of image *ima* with the size *delta*.

Definition at line 113 of file `extension/adjust.hh`.

References `adjust()`.

**9.44.2.4** `template<typename I , typename N > void mln::extension::adjust ( const Image< I > & ima, const Neighborhood< N > & nbh )`

Adjust the domain extension of image `ima` with the size of the neighborhood `nbh`.

Definition at line 105 of file `extension/adjust.hh`.

References `adjust()`, and `mln::geom::delta()`.

**9.44.2.5** `template<typename I , typename W > void mln::extension::adjust_duplicate ( const Image< I > & ima, const Window< W > & win )`

Adjust then duplicate.

Definition at line 70 of file `adjust_duplicate.hh`.

References `adjust()`, and `duplicate()`.

**9.44.2.6** `template<typename I , typename W > void mln::extension::adjust_fill ( const Image< I > & ima, const Window< W > & win, const typename I::value & val )`

Adjust then fill.

Definition at line 72 of file `adjust_fill.hh`.

References `adjust()`, and `fill()`.

**9.44.2.7** `template<typename I > void mln::extension::duplicate ( const Image< I > & ima )`

Assign the contents of the domain extension by duplicating the values of the inner boundary of image `ima`.

Definition at line 58 of file `extension/duplicate.hh`.

Referenced by `adjust_duplicate()`.

**9.44.2.8** `template<typename I > void mln::extension::fill ( const Image< I > & ima, const typename I::value & val )`

Fill the domain extension of image `ima` with the single value `v`.

**Parameters**

[in, out] *ima* The image whose domain extension is to be filled.

[in] *val* The value to assign.

**Precondition**

`ima` has to be initialized.

Definition at line 173 of file `extension/fill.hh`.

Referenced by `adjust_fill()`.

## 9.45 `mln::fun` Namespace Reference

Namespace of functions.

### Namespaces

- namespace [access](#)  
*Namespace for access functions.*
- namespace [i2v](#)  
*Namespace of integer-to-value functions.*
- namespace [n2v](#)  
*Namespace of functions from nil to value.*
- namespace [p2b](#)  
*Namespace of functions from point to boolean.*
- namespace [p2p](#)  
*Namespace of functions from grid point to grid point.*
- namespace [p2v](#)  
*Namespace of functions from point to value.*
- namespace [stat](#)  
*Namespace of statistical functions.*
- namespace [v2b](#)  
*Namespace of functions from value to logic value.*
- namespace [v2i](#)  
*Namespace of value-to-integer functions.*
- namespace [v2v](#)  
*Namespace of functions from value to value.*
- namespace [v2w2v](#)  
*Namespace of bijective functions.*
- namespace [v2w\\_w2v](#)  
*Namespace of functions from value to value.*
- namespace [vv2b](#)  
*Namespace of functions from value to value.*
- namespace [vv2v](#)  
*Namespace of functions from a couple of values to a value.*
- namespace [x2p](#)



*Namespace of functions from point to value.*

- namespace [x2v](#)

*Namespace of functions from vector to value.*

- namespace [x2x](#)

*Namespace of functions from vector to vector.*

## Classes

- struct [from\\_accu](#)

*Wrap an accumulator into a function.*

### 9.45.1 Detailed Description

Namespace of functions. Forward declarations.

`fun::i2v::array`

Forward declaration.

## 9.46 mln::fun::access Namespace Reference

Namespace for access functions.

### 9.46.1 Detailed Description

Namespace for access functions.

## 9.47 mln::fun::i2v Namespace Reference

Namespace of integer-to-value functions.

## Functions

- `template<typename T >`  
`std::ostream & operator<< (std::ostream &ostr, const array< T > &a)`

*Operator<<.*

### 9.47.1 Detailed Description

Namespace of integer-to-value functions.

## 9.47.2 Function Documentation

### 9.47.2.1 `template<typename T> std::ostream & mln::fun::i2v::operator<< ( std::ostream & ostr, const array< T > & a )`

Operator<<.

Definition at line 353 of file fun/i2v/array.hh.

## 9.48 `mln::fun::n2v` Namespace Reference

Namespace of functions from nil to value.

### Classes

- struct [white\\_gaussian](#)  
*Generate a White Gaussian Noise.*

### 9.48.1 Detailed Description

Namespace of functions from nil to value.

## 9.49 `mln::fun::p2b` Namespace Reference

Namespace of functions from point to boolean.

### Classes

- struct [antilogy](#)  
*A `p2b` function always returning `false`.*
- struct [tautology](#)  
*A `p2b` function always returning `true`.*

### 9.49.1 Detailed Description

Namespace of functions from point to boolean.

## 9.50 `mln::fun::p2p` Namespace Reference

Namespace of functions from grid point to grid point.

### 9.50.1 Detailed Description

Namespace of functions from grid point to grid point.

## 9.51 mln::fun::p2v Namespace Reference

Namespace of functions from point to value.

### 9.51.1 Detailed Description

Namespace of functions from point to value.

## 9.52 mln::fun::stat Namespace Reference

Namespace of statistical functions.

### 9.52.1 Detailed Description

Namespace of statistical functions.

## 9.53 mln::fun::v2b Namespace Reference

Namespace of functions from value to logic value.

### Classes

- struct [lnot](#)  
*Functor computing logical-not on a value.*
- struct [threshold](#)  
*Threshold function.*

### 9.53.1 Detailed Description

Namespace of functions from value to logic value.

## 9.54 mln::fun::v2i Namespace Reference

Namespace of value-to-integer functions.

### 9.54.1 Detailed Description

Namespace of value-to-integer functions.

## 9.55 mln::fun::v2v Namespace Reference

Namespace of functions from value to value.

### Classes

- class [ch\\_function\\_value](#)  
*Wrap a function  $v2v$  and convert its result to another type.*
- struct [component](#)  
*Functor that accesses the  $i$ -th component of a value.*
- struct [l1\\_norm](#)  
 *$L1$ -norm.*
- struct [l2\\_norm](#)  
 *$L2$ -norm.*
- struct [linear](#)  
*Linear function.  $f(v) = a * v + b$ .  $V$  is the type of input values;  $T$  is the type used to compute the result;  $R$  is the result type.*
- struct [linfty\\_norm](#)  
 *$L$ -infty norm.*
- struct [rgb8\\_to\\_rgn](#)  
*Convert a  $rgb8$  value to a  $rgn$ ,  $n < 8$ .*

### Variables

- [f\\_hsi\\_to\\_rgb\\_3x8\\_t f\\_hsi\\_to\\_rgb\\_3x8](#)  
*Global variable.*
- [f\\_hsl\\_to\\_rgb\\_3x8\\_t f\\_hsl\\_to\\_rgb\\_3x8](#)  
*Global variables.*
- [f\\_rgb\\_to\\_hsi\\_f\\_t f\\_rgb\\_to\\_hsi\\_f](#)  
*Global variables.*
- [f\\_rgb\\_to\\_hsl\\_f\\_t f\\_rgb\\_to\\_hsl\\_f](#)  
*Global variables.*

### 9.55.1 Detailed Description

Namespace of functions from value to value.

### 9.55.2 Variable Documentation

#### 9.55.2.1 f\_hsi\_to\_rgb\_3x8\_t mln::fun::v2v::f\_hsi\_to\_rgb\_3x8

Global variable.

Definition at line 74 of file hsi\_to\_rgb.hh.

#### 9.55.2.2 f\_hsl\_to\_rgb\_3x8\_t mln::fun::v2v::f\_hsl\_to\_rgb\_3x8

Global variables.

Definition at line 92 of file hsl\_to\_rgb.hh.

#### 9.55.2.3 f\_rgb\_to\_hsi\_f\_t mln::fun::v2v::f\_rgb\_to\_hsi\_f

Global variables.

Definition at line 66 of file rgb\_to\_hsi.hh.

#### 9.55.2.4 f\_rgb\_to\_hsl\_f\_t mln::fun::v2v::f\_rgb\_to\_hsl\_f

Global variables.

Definition at line 75 of file rgb\_to\_hsl.hh.

## 9.56 mln::fun::v2w2v Namespace Reference

Namespace of bijective functions.

### Classes

- struct [cos](#)

*Cosinus bijective functor.*

### 9.56.1 Detailed Description

Namespace of bijective functions.

## 9.57 mln::fun::v2w\_w2v Namespace Reference

Namespace of functions from value to value.

## Classes

- struct [l1\\_norm](#)  
*L1-norm.*
- struct [l2\\_norm](#)  
*L2-norm.*
- struct [linfty\\_norm](#)  
*L-infty norm.*

### 9.57.1 Detailed Description

Namespace of functions from value to value.

## 9.58 `mln::fun::vv2b` Namespace Reference

Namespace of functions from value to value.

## Classes

- struct [eq](#)  
*Functor computing equal between two values.*
- struct [ge](#)  
*Functor computing "greater or equal than" between two values.*
- struct [gt](#)  
*Functor computing "greater than" between two values.*
- struct [implies](#)  
*Functor computing logical-implies between two values.*
- struct [le](#)  
*Functor computing "lower or equal than" between two values.*
- struct [lt](#)  
*Functor computing "lower than" between two values.*

### 9.58.1 Detailed Description

Namespace of functions from value to value.

## 9.59 mln::fun::vv2v Namespace Reference

Namespace of functions from a couple of values to a value.

### Classes

- struct [diff\\_abs](#)  
*A functor computing the diff\_absimum of two values.*
- struct [land](#)  
*Functor computing logical-and between two values.*
- struct [land\\_not](#)  
*Functor computing logical and-not between two values.*
- struct [lor](#)  
*Functor computing logical-or between two values.*
- struct [lxor](#)  
*Functor computing logical-xor between two values.*
- struct [max](#)  
*A functor computing the maximum of two values.*
- struct [min](#)  
*A functor computing the minimum of two values.*
- struct [vec](#)  
*A functor computing the vecimum of two values.*

### 9.59.1 Detailed Description

Namespace of functions from a couple of values to a value.

## 9.60 mln::fun::x2p Namespace Reference

Namespace of functions from point to value.

### Classes

- struct [closest\\_point](#)  
*FIXME: doxygen + concept checking.*

### 9.60.1 Detailed Description

Namespace of functions from point to value.

## 9.61 `mln::fun::x2v` Namespace Reference

Namespace of functions from vector to value.

### Classes

- struct [bilinear](#)  
*Represent a bilinear interolation of values from an underlying image.*
- struct [trilinear](#)  
*Represent a trilinear interolation of values from an underlying image.*

### 9.61.1 Detailed Description

Namespace of functions from vector to value.

## 9.62 `mln::fun::x2x` Namespace Reference

Namespace of functions from vector to vector.

### Classes

- struct [composed](#)  
*Represent a composition of two transformations.*
- struct [linear](#)  
*Represent a linear interolation of values from an underlying image.*
- struct [rotation](#)  
*Represent a rotation function.*
- struct [translation](#)  
*Translation function-object.*

### 9.62.1 Detailed Description

Namespace of functions from vector to vector.

## 9.63 `mln::geom` Namespace Reference

Namespace of all things related to geometry.



## Namespaces

- namespace [impl](#)  
*Implementation namespace of geom namespace.*

## Classes

- class [complex\\_geometry](#)  
*A functor returning the sites of the faces of a complex where the locations of each 0-face is stored.*

## Functions

- `template<typename S >`  
`box< typename S::site > bbox (const Site_Set< S > &pset)`  
*Compute the precise bounding box of a point set pset.*
- `template<typename I >`  
`box< typename I::site > bbox (const Image< I > &ima)`  
*Compute the precise bounding box of a point set pset.*
- `template<typename W >`  
`box< typename W::psite > bbox (const Window< W > &win)`  
*Compute the precise bounding box of a window win.*
- `template<typename W >`  
`box< typename W::psite > bbox (const Weighted_Window< W > &win)`  
*Compute the precise bounding box of a weighted window win.*
- `template<typename I, typename W >`  
`mln::trait::ch_value< I, unsigned >::ret chamfer (const Image< I > &input_, const W &w_win_, unsigned max=mln_max(unsigned))`  
*Apply chamfer algorithm to a binary image.*
- `template<typename W >`  
`unsigned delta (const Window< W > &win)`  
*Compute the delta of a window win.*
- `template<typename W >`  
`unsigned delta (const Weighted_Window< W > &wwin)`  
*Compute the delta of a weighted window wwin.*
- `template<typename N >`  
`unsigned delta (const Neighborhood< N > &nbh)`  
*Compute the delta of a neighborhood nbh.*
- `template<typename I >`  
`mln::trait::concrete< I >::ret horizontal_symmetry (const Image< I > &input)`  
*Performs a horizontal symmetry.*

- `template<typename I >`  
`I::site::coord max\_col (const Image< I > &ima)`  
*Give the maximum column of an image.*
- `template<typename B >`  
`B::site::coord max\_col (const Box< B > &b)`  
*Give the maximum col of an box 2d or 3d.*
- `template<typename I >`  
`I::site::coord max\_ind (const Image< I > &ima)`  
*Give the maximum ind of an image.*
- `template<typename I >`  
`I::site::coord max\_row (const Image< I > &ima)`  
*Give the maximum row of an image.*
- `template<typename B >`  
`B::site::coord max\_row (const Box< B > &b)`  
*Give the maximum row of an box 2d or 3d.*
- `template<typename I >`  
`I::site::coord max\_sli (const Image< I > &ima)`  
*Give the maximum sli of an image.*
- `std::pair< complex\_image< 2, mln::space\_2complex\_geometry, algebra::vec< 3, float > >, complex\_image< 2, mln::space\_2complex\_geometry, float > > mesh\_corner\_point\_area (const p\_complex< 2, space\_2complex\_geometry > &mesh)`  
*Compute the area “belonging” to normals at vertices.*
- `std::pair< complex\_image< 2, mln::space\_2complex\_geometry, float >, complex\_image< 2, mln::space\_2complex\_geometry, float > > mesh\_curvature (const p\_complex< 2, space\_2complex\_geometry > &mesh)`  
*Compute the principal curvatures of a surface at vertices.*
- `complex\_image< 2, mln::space\_2complex\_geometry, algebra::vec< 3, float > > mesh\_normal (const p\_complex< 2, space\_2complex\_geometry > &mesh)`  
*Compute normals at vertices.*
- `template<typename I >`  
`I::site::coord min\_col (const Image< I > &ima)`  
*Give the minimum column of an image.*
- `template<typename B >`  
`B::site::coord min\_col (const Box< B > &b)`  
*Give the minimum column of an box 2d or 3d.*
- `template<typename I >`  
`I::site::coord min\_ind (const Image< I > &ima)`  
*Give the minimum ind of an image.*

- `template<typename I >`  
`I::site::coord min_row (const Image< I > &ima)`  
*Give the minimum row of an image.*
- `template<typename B >`  
`B::site::coord min_row (const Box< B > &b)`  
*Give the minimum row of an box 2d or 3d.*
- `template<typename I >`  
`I::site::coord min_sli (const Image< I > &ima)`  
*Give the minimum sli of an image.*
- `template<typename I >`  
`unsigned ncols (const Image< I > &ima)`  
*Give the number of columns of an image.*
- `template<typename B >`  
`unsigned ncols (const Box< B > &b)`  
*Give the number of cols of a box 2d or 3d.*
- `template<typename I >`  
`unsigned ninds (const Image< I > &ima)`  
*Give the number of inds of an image.*
- `template<typename I >`  
`unsigned nrows (const Image< I > &ima)`  
*Give the number of rows of an image.*
- `template<typename B >`  
`unsigned nrows (const Box< B > &b)`  
*Give the number of rows of a box 2d or 3d.*
- `template<typename I >`  
`unsigned nsites (const Image< I > &input)`  
*Compute the number of sites of the image input.*
- `template<typename I >`  
`unsigned nslis (const Image< I > &ima)`  
*Give the number of slices of an image.*
- `template<typename S >`  
`void pmin_pmax (const Site_Set< S > &s, typename S::site &pmin, typename S::site &pmax)`  
*Compute the minimum and maximum points, *pmin* and *max*, of point set *s*.*
- `template<typename S >`  
`std::pair< typename S::site, typename S::site > pmin_pmax (const Site_Set< S > &s)`  
*Compute the minimum and maximum points of point set *s*.*
- `template<typename I >`  
`std::pair< typename I::site, typename I::site > pmin_pmax (const Site_Iterator< I > &p)`  
*Compute the minimum and maximum points when browsing with iterator *p*.*

- `template<typename I >`  
`void pmin_pmax (const Site_Iterator< I > &p, typename I::site &pmin, typename I::site &pmax)`  
*Compute the minimum and maximum points, `pmin` and `max`, when browsing with iterator `p`.*
- `template<typename I, typename Ext >`  
`mln::trait::concrete< I >::ret rotate (const Image< I > &input, double angle, const Ext &extension)`
- `template<typename B >`  
`B rotate (const Box< B > &box, double angle)`  
*This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts. The rotation center `ref` is set to `box.pcenter()`.*
- `template<typename I, typename Ext, typename S >`  
`mln::trait::concrete< I >::ret rotate (const Image< I > &input, double angle, const Ext &extension, const Site_Set< S > &output_domain)`  
*Perform a rotation from the center of an image.*
- `template<typename I >`  
`mln::trait::concrete< I >::ret rotate (const Image< I > &input, double angle)`  
*This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts. Use `literal::zero` as default value for the extension.*
- `template<typename B >`  
`B rotate (const Box< B > &box_, double angle, const typename B::site &ref)`  
*Rotate a box.*
- `template<typename I, typename N >`  
`mln::trait::concrete< I >::ret seeds2tiling (const Image< I > &ima_, const Neighborhood< N > &nbh)`  
*Take a labeled image `ima_` with seeds and extend them until creating tiles.*
- `template<typename I, typename V, typename Ext, typename S >`  
`mln::trait::concrete< I >::ret translate (const Image< I > &input, const algebra::vec< I::site::dim, V > &ref, const Ext &extension, const Site_Set< S > &output_domain)`  
*Perform a translation from the center of an image.*
- `template<typename I, typename V >`  
`mln::trait::concrete< I >::ret translate (const Image< I > &input, const algebra::vec< I::site::dim, V > &ref)`  
*This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts. Use `literal::zero` as default value for the extension.*
- `template<typename I, typename V, typename Ext >`  
`mln::trait::concrete< I >::ret translate (const Image< I > &input, const algebra::vec< I::site::dim, V > &ref, const Ext &extension)`
- `template<typename I >`  
`mln::trait::concrete< I >::ret vertical_symmetry (const Image< I > &input)`  
*Performs a vertical symmetry.*
- `template<typename I, typename N >`  
`I seeds2tiling_roundness (Image< I > &ima_, const w_window2d_int &w_win, unsigned max, const Neighborhood< N > &nbh_)`

Take a labeled image *ima\_* with seeds and extend them until creating tiles rounder than the primary version.

### 9.63.1 Detailed Description

Namespace of all things related to geometry. Namespace of essential things related to geometry.

### 9.63.2 Function Documentation

**9.63.2.1** `template<typename S > box< typename S::site > mln::geom::bbox ( const Site_Set< S > & pset ) [inline]`

Compute the precise bounding box of a point set *pset*.

Definition at line 122 of file `geom/bbox.hh`.

Referenced by `bbox()`, `mln::transform::distance_and_closest_point_geodesic()`, `mln::registration::icp()`, `max_col()`, `max_row()`, `max_sli()`, `min_col()`, `min_row()`, `min_sli()`, `mln::debug::println()`, `mln::debug::println_with_border()`, and `rotate()`.

**9.63.2.2** `template<typename I > box< typename I::site > mln::geom::bbox ( const Image< I > & ima )`

Compute the precise bounding box of a point set *pset*.

Definition at line 133 of file `geom/bbox.hh`.

References `bbox()`.

**9.63.2.3** `template<typename W > box< typename W::psite > mln::geom::bbox ( const Window< W > & win )`

Compute the precise bounding box of a window *win*.

Definition at line 143 of file `geom/bbox.hh`.

References `mln::literal::origin`.

**9.63.2.4** `template<typename W > box< typename W::psite > mln::geom::bbox ( const Weighted_Window< W > & win )`

Compute the precise bounding box of a weighted window *win*.

Definition at line 156 of file `geom/bbox.hh`.

References `bbox()`.

**9.63.2.5** `template<typename I , typename W > mln::trait::ch_value< I, unsigned >::ret mln::geom::chamfer ( const Image< I > & input_, const W & w_win_, unsigned max = mln_max(unsigned) )`

Apply chamfer algorithm to a binary image.

Definition at line 113 of file geom/chamfer.hh.

#### 9.63.2.6 `template<typename W > unsigned mln::geom::delta ( const Window< W > & win )`

Compute the delta of a window `win`.

Definition at line 96 of file delta.hh.

Referenced by `mln::extension::adjust()`, and `delta()`.

#### 9.63.2.7 `template<typename W > unsigned mln::geom::delta ( const Weighted_Window< W > & wwin )`

Compute the delta of a weighted window `wwin`.

Definition at line 105 of file delta.hh.

References `delta()`.

#### 9.63.2.8 `template<typename N > unsigned mln::geom::delta ( const Neighborhood< N > & nbh )`

Compute the delta of a neighborhood `nbh`.

Definition at line 112 of file delta.hh.

References `delta()`.

#### 9.63.2.9 `template<typename I > mln::trait::concrete< I >::ret mln::geom::horizontal_symmetry ( const Image< I > & input )`

Performs a horizontal symmetry.

Definition at line 172 of file horizontal\_symmetry.hh.

#### 9.63.2.10 `template<typename I > I::site::coord mln::geom::max_col ( const Image< I > & ima ) [inline]`

Give the maximum column of an image.

Definition at line 56 of file max\_col.hh.

References `bbox()`.

Referenced by `mln::io::magick::load()`, and `ncols()`.

#### 9.63.2.11 `template<typename B > B::site::coord mln::geom::max_col ( const Box< B > & b ) [inline]`

Give the maximum col of an box 2d or 3d.

Definition at line 67 of file max\_col.hh.

**9.63.2.12** `template<typename I > I::site::coord mln::geom::max_ind ( const Image< I > & ima ) [inline]`

Give the maximum ind of an image.

Definition at line 51 of file max\_ind.hh.

Referenced by ninds().

**9.63.2.13** `template<typename I > I::site::coord mln::geom::max_row ( const Image< I > & ima ) [inline]`

Give the maximum row of an image.

Definition at line 57 of file max\_row.hh.

References bbox().

Referenced by mln::io::magick::load(), and nrows().

**9.63.2.14** `template<typename B > B::site::coord mln::geom::max_row ( const Box< B > & b ) [inline]`

Give the maximum row of an box 2d or 3d.

Definition at line 68 of file max\_row.hh.

**9.63.2.15** `template<typename I > I::site::coord mln::geom::max_sli ( const Image< I > & ima ) [inline]`

Give the maximum sli of an image.

Definition at line 53 of file max\_sli.hh.

References bbox().

Referenced by nslis().

**9.63.2.16** `std::pair< complex_image< 2, mln::space_2complex_geometry, algebra::vec<3, float> >, complex_image< 2, mln::space_2complex_geometry, float > > mln::geom::mesh_corner_point_area ( const p_complex< 2, space_2complex_geometry > & mesh ) [inline]`

Compute the area “belonging” to normals at vertices.

Inspired from the method Trimesh::need\_pointareas of the Trimesh library.

**See also**

<http://www.cs.princeton.edu/gfx/proj/trimesh2/>

From the documentation of Trimesh:

“Compute the area “belonging” to each vertex or each corner of a triangle (defined as Voronoi area restricted to the 1-ring of a vertex, or to the triangle).”

Definition at line 249 of file misc.hh.

References `mln::data::fill()`, `mln::norm::sqr_l2()`, `mln::algebra::vprod()`, and `mln::literal::zero`.

Referenced by `mesh_curvature()`.

**9.63.2.17** `std::pair< complex_image< 2, mln::space_2complex_geometry, float >, complex_image< 2, mln::space_2complex_geometry, float > >`  
`mln::geom::mesh_curvature ( const p_complex< 2, space_2complex_geometry > & mesh ) [inline]`

Compute the principal curvatures of a surface at vertices.

These principal curvatures are names `kappa_1` and `kappa_2` in

Sylvie Philipp-Foliguet, Michel Jordan Laurent Najman and Jean Cousty. Artwork 3D Model Database Indexing and Classification.

#### Parameters

[in] *mesh* The surface (triangle mesh) on which the curvature is to be computed.

Definition at line 486 of file `misc.hh`.

References `mln::algebra::ldlt_decomp()`, `mln::algebra::ldlt_solve()`, `mesh_corner_point_area()`, `mesh_normal()`, `mln::algebra::vprod()`, and `mln::literal::zero`.

**9.63.2.18** `complex_image< 2, mln::space_2complex_geometry, algebra::vec<3, float> >`  
`mln::geom::mesh_normal ( const p_complex< 2, space_2complex_geometry > & mesh ) [inline]`

Compute normals at vertices.

Inspired from the method `Trimesh::need_normals` of the Trimesh library.

#### See also

<http://www.cs.princeton.edu/gfx/proj/trimesh2/>

For simplicity purpose, and contrary to Trimesh, this routine only compute normals from a mesh, not from a cloud of points.

Definition at line 161 of file `misc.hh`.

References `mln::data::fill()`, `mln::norm::sqr_l2()`, `mln::algebra::vprod()`, and `mln::literal::zero`.

Referenced by `mesh_curvature()`.

**9.63.2.19** `template<typename I > I::site::coord mln::geom::min_col ( const Image< I > & ima ) [inline]`

Give the minimum column of an image.

Definition at line 57 of file `min_col.hh`.

References `bbox()`.

Referenced by `mln::transform::hough()`, `mln::io::magick::load()`, and `ncols()`.



**9.63.2.20** `template<typename B > B::site::coord mln::geom::min_col ( const Box< B > & b )`  
`[inline]`

Give the minimum column of an box 2d or 3d.

Definition at line 68 of file min\_col.hh.

**9.63.2.21** `template<typename I > I::site::coord mln::geom::min_ind ( const Image< I > & ima`  
`) [inline]`

Give the minimum ind of an image.

Definition at line 51 of file min\_ind.hh.

Referenced by ninds().

**9.63.2.22** `template<typename I > I::site::coord mln::geom::min_row ( const Image< I > & ima`  
`) [inline]`

Give the minimum row of an image.

Definition at line 60 of file min\_row.hh.

References bbox().

Referenced by mln::transform::hough(), mln::io::magick::load(), and nrows().

**9.63.2.23** `template<typename B > B::site::coord mln::geom::min_row ( const Box< B > & b )`  
`[inline]`

Give the minimum row of an box 2d or 3d.

Definition at line 71 of file min\_row.hh.

**9.63.2.24** `template<typename I > I::site::coord mln::geom::min_sli ( const Image< I > & ima )`  
`[inline]`

Give the minimum sli of an image.

Definition at line 53 of file min\_sli.hh.

References bbox().

Referenced by nslis().

**9.63.2.25** `template<typename I > unsigned mln::geom::ncols ( const Image< I > & ima )`  
`[inline]`

Give the number of columns of an image.

Definition at line 57 of file ncols.hh.

References max\_col(), and min\_col().

Referenced by mln::labeling::impl::compute\_fastest(), mln::subsampling::gaussian\_subsampling(), mln::transform::hough(), ncols(), mln::io::magick::save(), and mln::subsampling::subsampling().

**9.63.2.26** `template<typename B > unsigned mln::geom::ncols ( const Box< B > & b )`

Give the number of cols of a box 2d or 3d.

Definition at line 67 of file `ncols.hh`.

References `max_col()`, `min_col()`, and `ncols()`.

**9.63.2.27** `template<typename I > unsigned mln::geom::ninds ( const Image< I > & ima )`  
`[inline]`

Give the number of inds of an image.

Definition at line 52 of file `ninds.hh`.

References `max_ind()`, and `min_ind()`.

**9.63.2.28** `template<typename I > unsigned mln::geom::nrows ( const Image< I > & ima )`  
`[inline]`

Give the number of rows of an image.

Definition at line 57 of file `nrows.hh`.

References `max_row()`, and `min_row()`.

Referenced by `mln::subsampling::gaussian_subsampling()`, `mln::transform::hough()`, `nrows()`, `mln::io::magick::save()`, and `mln::subsampling::subsampling()`.

**9.63.2.29** `template<typename B > unsigned mln::geom::nrows ( const Box< B > & b )`

Give the number of rows of a box 2d or 3d.

Definition at line 66 of file `nrows.hh`.

References `max_row()`, `min_row()`, and `nrows()`.

**9.63.2.30** `template<typename I > unsigned mln::geom::nsites ( const Image< I > & input )`  
`[inline]`

Compute the number of sites of the image `input`.

Definition at line 52 of file `nsites.hh`.

Referenced by `pmin_pmax()`.

**9.63.2.31** `template<typename I > unsigned mln::geom::nslis ( const Image< I > & ima )`  
`[inline]`

Give the number of slices of an image.

Definition at line 53 of file `nslis.hh`.

References `max_sli()`, and `min_sli()`.

**9.63.2.32** `template<typename S > std::pair< typename S::site, typename S::site >  
mln::geom::pmin_pmax ( const Site_Set< S > & s ) [inline]`

Compute the minimum and maximum points of point set *s*.

Definition at line 157 of file pmin\_pmax.hh.

References nsites().

Referenced by pmin\_pmax().

**9.63.2.33** `template<typename S > void mln::geom::pmin_pmax ( const Site_Set< S > & s,  
typename S::site & pmin, typename S::site & pmax ) [inline]`

Compute the minimum and maximum points, *pmin* and *max*, of point set *s*.

Definition at line 148 of file pmin\_pmax.hh.

References nsites().

**9.63.2.34** `template<typename I > std::pair< typename I::site, typename I::site >  
mln::geom::pmin_pmax ( const Site_Iterator< I > & p ) [inline]`

Compute the minimum and maximum points when browsing with iterator *p*.

Definition at line 106 of file pmin\_pmax.hh.

References pmin\_pmax().

**9.63.2.35** `template<typename I > void mln::geom::pmin_pmax ( const Site_Iterator< I > & p,  
typename I::site & pmin, typename I::site & pmax ) [inline]`

Compute the minimum and maximum points, *pmin* and *max*, when browsing with iterator *p*.

Definition at line 84 of file pmin\_pmax.hh.

**9.63.2.36** `template<typename I > mln::trait::concrete< I >::ret mln::geom::rotate ( const  
Image< I > & input, double angle )`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts. Use [literal::zero](#) as default value for the extension.

Definition at line 197 of file rotate.hh.

References rotate(), and mln::literal::zero.

**9.63.2.37** `template<typename I, typename Ext > mln::trait::concrete< I >::ret  
mln::geom::rotate ( const Image< I > & input, double angle, const Ext & extension )`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 185 of file rotate.hh.

References rotate().

### 9.63.2.38 `template<typename B > B mln::geom::rotate ( const Box< B > & box, double angle )`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts. The rotation center `ref` is set to `box.pcenter()`.

Definition at line 252 of file `rotate.hh`.

References `rotate()`.

### 9.63.2.39 `template<typename I, typename Ext, typename S > mln::trait::concrete< I >::ret mln::geom::rotate ( const Image< I > & input, double angle, const Ext & extension, const Site_Set< S > & output_domain )`

Perform a rotation from the center of an image.

#### Parameters

[in] *input* An image.

[in] *angle* An angle in degrees.

[in] *extension* [Function](#), image or value which will be used as extension. This extension allows to map values to sites which where not part of the domain before the rotation.

[in] *output\_domain* The domain of the output image. An invalid domain, causes the routine to use a domain large enough to display the whole original image.

#### Returns

An image with the same domain as `input`.

Definition at line 123 of file `rotate.hh`.

References `bbox()`, `mln::compose()`, `mln::duplicate()`, `mln::initialize()`, `mln::mln_exact()`, `mln::literal::origin`, and `mln::data::paste()`.

Referenced by `rotate()`.

### 9.63.2.40 `template<typename B > B mln::geom::rotate ( const Box< B > & box_, double angle, const typename B::site & ref )`

Rotate a box.

FIXME: the return type may be too generic and may lead to invalid covariance.

Definition at line 205 of file `rotate.hh`.

References `mln::compose()`, `mln::literal::origin`, and `mln::accu::shape::bbox< P >::to_result()`.

### 9.63.2.41 `template<typename I, typename N > mln::trait::concrete< I >::ret mln::geom::seeds2tiling ( const Image< I > & ima_, const Neighborhood< N > & nbh ) [inline]`

Take a labeled image `ima_` with seeds and extend them until creating tiles.

#### Parameters

[in, out] *ima\_* The labeled image with seed.

[in] *nbh* The neighborhood to use on this algorithm.

### Returns

A tiled image.

### Precondition

*ima\_* has to be initialized.

Definition at line 136 of file seeds2tiling.hh.

References `mln::geom::impl::seeds2tiling()`.

**9.63.2.42** `template<typename I, typename N > I mln::geom::seeds2tiling_roundness ( Image< I > & ima_, const w_window2d_int & w_win, unsigned max, const Neighborhood< N > & nbh_ ) [inline]`

Take a labeled image *ima\_* with seeds and extend them until creating tiles rounder than the primary version.

### Parameters

[in, out] *ima\_* The labeled image with seed.

[in] *w\_win* The weight window using by `geom::chamfer` to compute distance.

[in] *max* Unsigned using by `geom::chamfer` to compute the distance.

[in] *nbh\_* The neighborhood to use on this algorithm.

### Precondition

*ima\_* has to be initialized.

Definition at line 128 of file seeds2tiling\_roundness.hh.

**9.63.2.43** `template<typename I, typename V > mln::trait::concrete< I >::ret mln::geom::translate ( const Image< I > & input, const algebra::vec< I::site::dim, V > & ref )`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts. Use `literal::zero` as default value for the extension.

Definition at line 146 of file translate.hh.

References `translate()`, and `mln::literal::zero`.

**9.63.2.44** `template<typename I, typename V, typename Ext, typename S > mln::trait::concrete< I >::ret mln::geom::translate ( const Image< I > & input, const algebra::vec< I::site::dim, V > & ref, const Ext & extension, const Site_Set< S > & output_domain )`

Perform a translation from the center of an image.

### Parameters

[in] *input* An image.

- [in] *ref* The translation vector.
- [in] *extension* [Function](#), image or value which will be used as extension. This extension allows to map values to sites which where not part of the domain before the translation.
- [in] *output\_domain* The domain of the output image. An invalid domain, causes the routine to use the translated `input_domain`.

### Returns

An image with the same domain as `input`.

Definition at line 99 of file `translate.hh`.

References `mln::extend()`, `mln::data::fill()`, and `mln::mln_exact()`.

Referenced by `translate()`.

**9.63.2.45** `template<typename I , typename V , typename Ext > mln::trait::concrete< I >::ret mln::geom::translate ( const Image< I > & input, const algebra::vec< I::site::dim, V > & ref, const Ext & extension )`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 132 of file `translate.hh`.

References `translate()`.

**9.63.2.46** `template<typename I > mln::trait::concrete< I >::ret mln::geom::vertical_symmetry ( const Image< I > & input )`

Performs a vertical symmetry.

Definition at line 174 of file `vertical_symmetry.hh`.

## 9.64 mln::geom::impl Namespace Reference

Implementation namespace of `geom` namespace.

### Functions

- `template<typename I , typename N > mln::trait::concrete< I >::ret seeds2tiling (const Image< I > &ima_, const Neighborhood< N > &nbh_)`

*Generic implementation of `geom::seed2tiling`.*

#### 9.64.1 Detailed Description

Implementation namespace of `geom` namespace.

## 9.64.2 Function Documentation

**9.64.2.1** `template<typename I , typename N > mln::trait::concrete< I >::ret  
mln::geom::impl::seeds2tiling ( const Image< I > & ima_ , const Neighborhood< N > &  
nbh_ ) [inline]`

Generic implementation of geom::seed2tiling.

### Parameters

[in, out] *ima\_* The labeled image with seed.

[in] *nbh\_* The neighborhood to use on this algorithm.

Definition at line 77 of file seeds2tiling.hh.

References `mln::duplicate()`, `mln::p_queue< P >::front()`, `mln::p_queue< P >::pop()`, and `mln::p_queue< P >::push()`.

Referenced by `mln::geom::seeds2tiling()`.

## 9.65 mln::graph Namespace Reference

Namespace of graph related routines.

### Functions

- `template<typename G , typename F >  
F::result compute (const Graph< G > &g_ , F &functor)`  
*Base routine to compute attributes on a graph.*
- `template<typename I , typename N , typename L >  
mln::trait::ch_value< I , L >::ret labeling (const Image< I > &graph_image_ , const Neighborhood<  
N > &nbh_ , L &nlabels)`  
*Label graph components.*
- `template<typename I , typename M >  
graph_elt_neighborhood_if< mln_graph(I) , typename I::domain_t , M > to_neighb (const Image< I  
> &graph_image_ , const Image< M > &graph_mask_image_)`  
*Make a custom graph neighborhood from a mask image.*
- `template<typename I , typename M >  
graph_elt_window_if< mln_graph(I) , typename I::domain_t , M > to_win (const Image< I >  
&graph_image_ , const Image< M > &graph_mask_image_)`  
*Make a custom graph window from a mask image.*

### 9.65.1 Detailed Description

Namespace of graph related routines.

## 9.65.2 Function Documentation

### 9.65.2.1 `template<typename G , typename F > F::result mln::graph::compute ( const Graph< G > & g_, F & functor )`

Base routine to compute attributes on a graph.

#### Parameters

- [in] *g\_* A graph.
- [in] *functor* A functor implementing the right interface.

#### Returns

The computed data.

#### See also

`canvas::browsing::depth_first_search`

Definition at line 63 of file `graph/compute.hh`.

### 9.65.2.2 `template<typename I , typename N , typename L > mln::trait::ch_value< I, L >::ret mln::graph::labeling ( const Image< I > & graph_image_, const Neighborhood< N > & nbh_, L & nlabels )`

Label graph components.

[Vertex](#) with id 0, usually used to represent the background component, will be labeled with an id different from 0. Therefore, the labeling starts from 1.

#### Parameters

- [in] *graph\_image\_* A graph image (

#### See also

[vertex\\_image](#), [edge\\_image](#)).

#### Parameters

- [in] *nbh\_* A graph neighborhood.
- [in, out] *nlabels* The number of labels found.

#### Returns

a [Graph](#) image of labels.

Definition at line 72 of file `labeling.hh`.

References `mln::labeling::blobs()`, `mln::data::fill()`, and `mln::initialize()`.

### 9.65.2.3 `template<typename I , typename M > graph_elt_neighborhood_if< mln_graph(I), typename I::domain_t, M > mln::graph::to_neighb ( const Image< I > & graph_image_, const Image< M > & graph_mask_image_ )`

Make a custom graph neighborhood from a mask image.



**Parameters**

[in] *graph\_image\_* A graph image (

**See also**

[vertex\\_image](#) and [edge\\_image](#)).

**Parameters**

[in] *graph\_mask\_image\_* A graph image of bool used as a mask.

**Returns**

A masked neighborhood on graph.

Definition at line 57 of file to\_neighb.hh.

**9.65.2.4** `template<typename I, typename M > graph_elt_window_if< mln_graph(I), typename I::domain_t, M > mln::graph::to_win ( const Image< I > & graph_image_, const Image< M > & graph_mask_image_ )`

Make a custom graph window from a mask image.

**Parameters**

[in] *graph\_image\_* A graph image (

**See also**

[vertex\\_image](#) and [edge\\_image](#)).

**Parameters**

[in] *graph\_mask\_image\_* A graph image of bool used as a mask.

**Returns**

A masked window on graph.

Definition at line 57 of file to\_win.hh.

## 9.66 mln::grid Namespace Reference

Namespace of grids definitions.

### 9.66.1 Detailed Description

Namespace of grids definitions. Compute the image::space trait from a point type.

## 9.67 mln::histo Namespace Reference

Namespace of histograms.

## Namespaces

- namespace [impl](#)  
*Implementation namespace of histo namespace.*

## Classes

- struct [array](#)  
*Generic histogram class over a value set with type T.*

## Functions

- `template<typename I >`  
`histo::array< typename I::value > compute (const Image< I > &input)`  
*Compute the histogram of image input.*
- `template<typename I >`  
`mln::trait::concrete< I >::ret equalize (const Image< I > &input)`  
*Equalizes the histogram of image input.*

### 9.67.1 Detailed Description

Namespace of histograms.

### 9.67.2 Function Documentation

#### 9.67.2.1 `template<typename I > histo::array< typename I::value > mln::histo::compute ( const Image< I > & input ) [inline]`

Compute the histogram of image `input`.

Definition at line 79 of file `histo/compute.hh`.

Referenced by `equalize()`.

#### 9.67.2.2 `template<typename I > mln::trait::concrete< I >::ret mln::histo::equalize ( const Image< I > & input )`

Equalizes the histogram of image `input`.

#### Author

J. Fabrizio, R. Levillain

Definition at line 55 of file `histo/equalize.hh`.

References `compute()`, and `mln::initialize()`.

## 9.68 mln::histo::impl Namespace Reference

Implementation namespace of histo namespace.

### Namespaces

- namespace [generic](#)  
*Generic implementation namespace of histo namespace.*

### 9.68.1 Detailed Description

Implementation namespace of histo namespace.

## 9.69 mln::histo::impl::generic Namespace Reference

Generic implementation namespace of histo namespace.

### 9.69.1 Detailed Description

Generic implementation namespace of histo namespace.

## 9.70 mln::impl Namespace Reference

Implementation namespace of mln namespace.

### 9.70.1 Detailed Description

Implementation namespace of mln namespace.

## 9.71 mln::io Namespace Reference

Namespace of input/output handling.

### Namespaces

- namespace [cloud](#)  
*Namespace of cloud input/output handling.*
- namespace [dicom](#)  
*Namespace of DICOM input/output handling.*
- namespace [dump](#)

*Namespace of dump input/output handling.*

- namespace [fits](#)  
*Namespace of fits input/output handling.*
- namespace [fld](#)  
*Namespace of pgm input/output handling.*
- namespace [magick](#)  
*Namespace of magick input/output handling.*
- namespace [off](#)  
*Namespace of off input/output handling.*
- namespace [pbm](#)  
*Namespace of pbm input/output handling.*
- namespace [pbms](#)  
*Namespace of pbms input/output handling.*
- namespace [pfm](#)  
*Namespace of pfm input/output handling.*
- namespace [pgm](#)  
*Namespace of pgm input/output handling.*
- namespace [pgms](#)  
*Namespace of pgms input/output handling.*
- namespace [plot](#)  
*Namespace of plot input/output handling.*
- namespace [pnm](#)  
*Namespace of pnm input/output handling.*
- namespace [pnms](#)  
*Namespace of pnms input/output handling.*
- namespace [ppm](#)  
*Namespace of ppm input/output handling.*
- namespace [ppms](#)  
*Namespace of ppms input/output handling.*
- namespace [raw](#)  
*Namespace of raw input/output handling.*
- namespace [tiff](#)  
*Namespace of tiff input/output handling.*

- namespace `txt`

*Namespace of txt input/output handling.*

### 9.71.1 Detailed Description

Namespace of input/output handling.

## 9.72 mln::io::cloud Namespace Reference

Namespace of cloud input/output handling.

### Functions

- `template<typename P > void load (p_array< P > &arr, const std::string &filename)`  
*Load a cloud of points.*
- `template<typename P > void save (const p_array< P > &arr, const std::string &filename)`  
*Load a cloud of points.*

### 9.72.1 Detailed Description

Namespace of cloud input/output handling.

### 9.72.2 Function Documentation

#### 9.72.2.1 `template<typename P > void mln::io::cloud::load ( p_array< P > & arr, const std::string & filename )`

Load a cloud of points.

#### Parameters

[in, out] *arr* the site set where to load the data.

[in] *filename* file to load.

Definition at line 88 of file cloud/load.hh.

#### 9.72.2.2 `template<typename P > void mln::io::cloud::save ( const p_array< P > & arr, const std::string & filename )`

Load a cloud of points.

**Parameters**

- [in] *arr* the cloud of points to save.
- [in] *filename* the destination.

Definition at line 83 of file cloud/save.hh.

**9.73 mln::io::dicom Namespace Reference**

Namespace of DICOM input/output handling.

**Classes**

- struct [dicom\\_header](#)  
*Store dicom file header.*

**Functions**

- [dicom\\_header get\\_header](#) (const std::string &filename)  
*Retrieve header in a dicom file.*
- template<typename I >  
void [load](#) ([Image](#)< I > &ima, const std::string &filename)

**9.73.1 Detailed Description**

Namespace of DICOM input/output handling.

**9.73.2 Function Documentation****9.73.2.1 dicom\_header mln::io::dicom::get\_header ( const std::string & filename )**

Retrieve header in a dicom file.

Definition at line 76 of file dicom/get\_header.hh.

References [mln::util::array< T >::append\(\)](#).

**9.73.2.2 template<typename I > void mln::io::dicom::load ( Image< I > & ima, const std::string & filename ) [inline]**

Load a DICOM file in a Milena image.

**Parameters**

- [out] *ima* A reference to the image which will receive data.
- [in] *filename* The source.

Common compilation flags to link to gdcem if this file is used:

```
-lgdcmCommon -lgdcmDICT -lgdcmDSED -lgdcmIOD -lgdcmMSFF -lgdcmexpat -lgdcmjpeg12 -  
-lgdcmjpeg16 -lgdcmjpeg8 -lgdcmopenjpeg -lgdcmuuid -lgdcmzlib
```

Definition at line 96 of file dicom/load.hh.

References mln::initialize().

## 9.74 mln::io::dump Namespace Reference

Namespace of dump input/output handling.

### Classes

- struct [dump\\_header](#)  
*Store dump file header.*

### Functions

- [dump\\_header get\\_header](#) (const std::string &filename)  
*Retrieve header in a dump file.*
- [template<typename I >  
void load](#) ([Image](#)< I > &ima\_, const std::string &filename)  
*Load a Milena image by dumped into a file.*
- [template<typename I >  
void save](#) (const [Image](#)< I > &ima\_, const std::string &filename)  
*Save a Milena image by dumping its data to a file.*

### 9.74.1 Detailed Description

Namespace of dump input/output handling.

### 9.74.2 Function Documentation

#### 9.74.2.1 [dump\\_header mln::io::dump::get\\_header](#) ( [const std::string & filename](#) )

Retrieve header in a dump file.

Definition at line 68 of file dump/get\_header.hh.

References mln::util::array< T >::resize().

### 9.74.2.2 `template<typename I> void mln::io::dump::load ( Image< I > & ima_, const std::string & filename )`

Load a Milena image by dumped into a file.

#### Parameters

[in, out] *ima\_* The image to load.

[in] *filename* the destination.

Definition at line 171 of file dump/load.hh.

### 9.74.2.3 `template<typename I> void mln::io::dump::save ( const Image< I > & ima_, const std::string & filename )`

Save a Milena image by dumping its data to a file.

#### Parameters

[in] *ima\_* The image to save.

[in] *filename* the destination.

Definition at line 131 of file dump/save.hh.

## 9.75 `mln::io::fits` Namespace Reference

Namespace of fits input/output handling.

### Functions

- void [load](#) ([image2d](#)< float > &ima, const std::string &filename)  
*Load a fits image in a Milena image.*
- [image2d](#)< float > [load](#) (const std::string &filename)  
*Load a fits image in a [image2d](#)<float>.*

### 9.75.1 Detailed Description

Namespace of fits input/output handling.

### 9.75.2 Function Documentation

#### 9.75.2.1 `void mln::io::fits::load ( image2d< float > & ima, const std::string & filename )` `[inline]`

Load a fits image in a Milena image.



### Parameters

- [out] *ima* A reference to the image2d<float> which will receive data.
- [in] *filename* The source.

Definition at line 132 of file fits/load.hh.

#### 9.75.2.2 image2d<float> mln::io::fits::load ( const std::string & filename ) [inline]

Load a fits image in a image2d<float>.

### Parameters

- [in] *filename* The image source.

### Returns

An image2d<float> which contains loaded data.

Definition at line 85 of file fits/load.hh.

## 9.76 mln::io::fld Namespace Reference

Namespace of pgm input/output handling.

### Classes

- struct [fld\\_header](#)  
*Define the header structure of an AVS field data file.*

### Functions

- template<typename I >  
void [load](#) (Image< I > &ima\_, const char \*filename)  
*Load an image from an AVS field file.*
- [fld\\_header read\\_header](#) (std::istream &ins)  
*Read the header form an AVS field file.*
- void [write\\_header](#) (std::ostream &file, const [fld\\_header](#) &h)  
*Write the AVS header in a file.*

#### 9.76.1 Detailed Description

Namespace of pgm input/output handling.

## 9.76.2 Function Documentation

### 9.76.2.1 `template<typename I> void mln::io::fld::load ( Image< I > & ima_, const char * filename ) [inline]`

Load an image from an AVS field file.

#### Parameters

- `[in, out]` *ima\_* The image to load.
- `[in]` *filename* The path to the AVS file.

Definition at line 198 of file fld/load.hh.

References `mln::box< P >::pmax()`, `mln::box< P >::pmin()`, and `read_header()`.

### 9.76.2.2 `fld_header mln::io::fld::read_header ( std::istream & ins ) [inline]`

Read the header form an AVS field file.

#### Parameters

- ins* The file to read.

#### Returns

- The header.

Definition at line 73 of file fld/load\_header.hh.

Referenced by `load()`.

### 9.76.2.3 `void mln::io::fld::write_header ( std::ostream & file, const fld_header & h ) [inline]`

Write the AVS header in a file.

#### Parameters

- file* The file to write.
- h* The AVS header.

Definition at line 58 of file write\_header.hh.

## 9.77 mln::io::magick Namespace Reference

Namespace of magick input/output handling.

### Functions

- `template<typename I> void load (Image< I > &ima, const std::string &filename)`

*Load data from a file into a Milena image using Magick++.*

- `template<typename I, typename J >`  
`void save (const Image< I > &ima, const Image< J > &opacity_mask, const std::string &filename)`

*Save a Milena image into a file using Magick++.*

- `template<typename I >`  
`void save (const Image< I > &ima, const std::string &filename)`

*Save a Milena image into a file using Magick++.*

## 9.77.1 Detailed Description

Namespace of magick input/output handling.

## 9.77.2 Function Documentation

### 9.77.2.1 `template<typename I > void mln::io::magick::load ( Image< I > & ima, const std::string & filename ) [inline]`

Load data from a file into a Milena image using Magick++.

#### Parameters

[out] *ima* The image data are loaded into.

[in] *filename* The name of the input file.

Definition at line 139 of file magick/load.hh.

References `mln::initialize()`, `mln::geom::max_col()`, `mln::geom::max_row()`, `mln::geom::min_col()`, and `mln::geom::min_row()`.

### 9.77.2.2 `template<typename I, typename J > void mln::io::magick::save ( const Image< I > & ima, const Image< J > & opacity_mask, const std::string & filename )`

Save a Milena image into a file using Magick++.

#### Parameters

[in] *ima* The image to save.

[in] *opacity\_mask* Mask used to set pixel opacity\_mask in output image. Output format must support this feature to be taken into account.

[in] *filename* The name of the output file.

Definition at line 228 of file magick/save.hh.

References `mln::geom::ncols()`, and `mln::geom::nrows()`.

### 9.77.2.3 `template<typename I> void mln::io::magick::save ( const Image< I > & ima, const std::string & filename ) [inline]`

Save a Milena image into a file using Magick++.

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

#### Parameters

- [in] *ima* The image to save.
- [in] *filename* The name of the output file.

Definition at line 288 of file magick/save.hh.

## 9.78 `mln::io::off` Namespace Reference

Namespace of off input/output handling.

### Functions

- void `load (bin_2complex_image3df &ima, const std::string &filename)`  
*Load a (binary) OFF image into a complex image.*
- void `save (const bin_2complex_image3df &ima, const std::string &filename)`  
*Save a (binary) OFF image into a complex image.*
- `template<typename I>`  
void `save_bin_alt (const I &ima, const std::string &filename)`  
*FIXME: Similar to `mln::io::off::save(const bin_2complex_image3df&, const std::string&)`, but does not save faces whose value is 'false'.*

### 9.78.1 Detailed Description

Namespace of off input/output handling.

### 9.78.2 Function Documentation

#### 9.78.2.1 `void mln::io::off::load ( bin_2complex_image3df & ima, const std::string & filename )`

Load a (binary) OFF image into a complex image.

Load a 3x8-bit RGB (color) OFF image into a complex image.

Load a floating-point OFF image into a complex image.

#### Parameters

- [out] *ima* A reference to the image to construct.
- [in] *filename* The name of the file to load.

The image is said binary since data only represent the existence of faces.

### Parameters

[out] *ima* A reference to the image to construct.

[in] *filename* The name of the file to load.

Read floating-point data is attached to 2-faces only; 1-faces and 0-faces are set to 0.0f.

Definition at line 195 of file off/load.hh.

#### 9.78.2.2 void mln::io::off::save ( const bin\_2complex\_image3df & ima, const std::string & filename )

Save a (binary) OFF image into a complex image.

Save a 3x8-bit RGB (color) OFF image into a complex image.

Save a floating-point value grey-level OFF image into a complex image.

Save an 8-bit grey-level OFF image into a complex image.

### Parameters

[in] *ima* The image to save.

[in] *filename* The name of the file where to save the image.

The image is said binary since data represent only the existence of faces.

### Parameters

[in] *ima* The image to save.

[in] *filename* The name of the file where to save the image.

Only data is attached to 2-faces is saved; the OFF file cannot store data attached to faces of other dimensions.

Definition at line 189 of file off/save.hh.

#### 9.78.2.3 template<typename I> void mln::io::off::save\_bin\_alt ( const I & ima, const std::string & filename )

FIXME: Similar to [mln::io::off::save\(const bin\\_2complex\\_image3df&, const std::string&\)](#), but does not save faces whose value is 'false'.

Definition at line 59 of file save\_bin\_alt.hh.

## 9.79 mln::io::pbm Namespace Reference

Namespace of pbm input/output handling.

## Namespaces

- namespace [impl](#)  
*Namespace of pbm implementation details.*

## Functions

- void [load](#) ([image2d](#)< bool > &ima, const std::string &filename)  
*Load a pbm image in a Milena image.*
- [image2d](#)< bool > [load](#) (const std::string &filename)  
*Load a pbm image in a image2d<float>.*
- template<typename I >  
void [save](#) (const [Image](#)< I > &ima, const std::string &filename)

### 9.79.1 Detailed Description

Namespace of pbm input/output handling.

### 9.79.2 Function Documentation

#### 9.79.2.1 void mln::io::pbm::load ( [image2d](#)< bool > & *ima*, const std::string & *filename* ) [\[inline\]](#)

Load a pbm image in a Milena image.

#### Parameters

- [out] *ima* A reference to the [image2d](#)<bool> which will receive data.  
[in] *filename* The source.

Definition at line 156 of file pbm/load.hh.

#### 9.79.2.2 [image2d](#)< bool > mln::io::pbm::load ( const std::string & *filename* ) [\[inline\]](#)

Load a pbm image in a [image2d](#)<float>.

#### Parameters

- [in] *filename* The image source.

#### Returns

An [image2d](#)<float> which contains loaded data.

Definition at line 128 of file pbm/load.hh.

**9.79.2.3** `template<typename I> void mln::io::pbm::save ( const Image< I > & ima, const std::string & filename ) [inline]`

Save a Milena image as a pbm image.

#### Parameters

- [in] *ima* The image to save.
- [in, out] *filename* the destination.

Definition at line 118 of file pbm/save.hh.

## 9.80 mln::io::pbm::impl Namespace Reference

Namespace of pbm implementation details.

### 9.80.1 Detailed Description

Namespace of pbm implementation details.

## 9.81 mln::io::pbms Namespace Reference

Namespace of pbms input/output handling.

### Namespaces

- namespace [impl](#)  
*Namespace of pbms implementation details.*

### Functions

- void [load](#) (image3d< bool > &ima, const util::array< std::string > &filenames)  
*Load pbms images as slices of a 3D Milena image.*

### 9.81.1 Detailed Description

Namespace of pbms input/output handling.

### 9.81.2 Function Documentation

**9.81.2.1** `void mln::io::pbms::load ( image3d< bool > & ima, const util::array< std::string > & filenames ) [inline]`

Load pbms images as slices of a 3D Milena image.

**Parameters**

[out] *ima* A reference to the 3D image which will receive data.

[in] *filenames* The list of 2D images to load..

Definition at line 65 of file pbms/load.hh.

**9.82 mln::io::pbms::impl Namespace Reference**

Namespace of pbms implementation details.

**9.82.1 Detailed Description**

Namespace of pbms implementation details.

**9.83 mln::io::pfm Namespace Reference**

Namespace of pfm input/output handling.

**Namespaces**

- namespace [impl](#)

*Implementation namespace of pfm namespace.*

**Functions**

- void [load](#) ([image2d](#)< float > &ima, const std::string &filename)

*Load a pfm image in a Milena image.*

- [image2d](#)< float > [load](#) (const std::string &filename)

*Load a pfm image in a image2d<float>.*

- template<typename I >

void [save](#) (const [Image](#)< I > &ima, const std::string &filename)

*Save a Milena image as a pfm image.*

**9.83.1 Detailed Description**

Namespace of pfm input/output handling.



## 9.83.2 Function Documentation

**9.83.2.1** `void mln::io::pfm::load ( image2d<float> & ima, const std::string & filename )`  
`[inline]`

Load a pfm image in a Milena image.

### Parameters

[out] *ima* A reference to the image2d<float> which will receive data.

[in] *filename* The source.

Definition at line 162 of file pfm/load.hh.

**9.83.2.2** `image2d<float> mln::io::pfm::load ( const std::string & filename )` `[inline]`

Load a pfm image in a image2d<float>.

### Parameters

[in] *filename* The image source.

### Returns

An image2d<float> which contains loaded data.

Definition at line 138 of file pfm/load.hh.

**9.83.2.3** `template<typename I> void mln::io::pfm::save ( const Image<I> & ima, const std::string & filename )` `[inline]`

Save a Milena image as a pfm image.

### Parameters

[in] *ima* The image to save.

[in, out] *filename* the destination.

Definition at line 101 of file pfm/save.hh.

## 9.84 mln::io::pfm::impl Namespace Reference

Implementation namespace of pfm namespace.

### 9.84.1 Detailed Description

Implementation namespace of pfm namespace.

## 9.85 mln::io::pgm Namespace Reference

Namespace of pgm input/output handling.

### Functions

- `template<typename I >`  
`void load (Image< I > &ima, const std::string &filename)`  
*Load a pgm image in a Milena image.*
- `template<typename V >`  
`image2d< V > load (const std::string &filename)`  
*Load a pgm image in a Milena image.*
- `template<typename I >`  
`void save (const Image< I > &ima, const std::string &filename)`

### 9.85.1 Detailed Description

Namespace of pgm input/output handling.

### 9.85.2 Function Documentation

#### 9.85.2.1 `template<typename I > void mln::io::pgm::load ( Image< I > & ima, const std::string & filename ) [inline]`

Load a pgm image in a Milena image.

#### Parameters

- [out] *ima* A reference to the image which will receive data.  
 [in] *filename* The source.

Definition at line 87 of file `pgm/load.hh`.

#### 9.85.2.2 `template<typename V > image2d< V > mln::io::pgm::load ( const std::string & filename ) [inline]`

Load a pgm image in a Milena image.

To use this routine, you should specialize the template with the value type of the image loaded. (ex : `load<value::int_u8>("...")`)

#### Parameters

- [in] *filename* The image source.

#### Returns

An `image2d` which contains loaded data.

Definition at line 77 of file `pgm/load.hh`.

**9.85.2.3** `template<typename I > void mln::io::pgms::save ( const Image< I > & ima, const std::string & filename ) [inline]`

Save a Milena image as a pgm image.

#### Parameters

- [in] *ima* The image to save.
- [in, out] *filename* the destination.

Definition at line 77 of file pgm/save.hh.

## 9.86 mln::io::pgms Namespace Reference

Namespace of pgms input/output handling.

### Functions

- `template<typename V > void load (image3d< V > &ima, const util::array< std::string > &filenames)`  
*Load pgm images as slices of a 3D Milena image.*

### 9.86.1 Detailed Description

Namespace of pgms input/output handling.

### 9.86.2 Function Documentation

**9.86.2.1** `template<typename V > void mln::io::pgms::load ( image3d< V > & ima, const util::array< std::string > & filenames ) [inline]`

Load pgm images as slices of a 3D Milena image.

#### Parameters

- [out] *ima* A reference to the 3D image which will receive data.
- [in] *filenames* The list of 2D images to load..

Definition at line 69 of file pgms/load.hh.

## 9.87 mln::io::plot Namespace Reference

Namespace of plot input/output handling.

## Functions

- `template<typename I >`  
`void load (util::array< I > &arr, const std::string &filename)`
- `template<typename T >`  
`void save (const histo::array< T > &arr, const std::string &filename)`
- `template<typename T >`  
`void save (const util::array< T > &arr, const std::string &filename, int start_value=0)`  
*Save a Milena array in a plot file.*
- `template<typename I >`  
`void save (const image1d< I > &ima, const std::string &filename)`  
*Save a Milena 1D image in a plot file.*

### 9.87.1 Detailed Description

Namespace of plot input/output handling.

### 9.87.2 Function Documentation

#### 9.87.2.1 `template<typename I > void mln::io::plot::load ( util::array< I > & arr, const std::string & filename ) [inline]`

Load a Milena 1D image from a plot file.

#### Parameters

- [in] *ima* A reference to the image to load.
- [out] *filename* The output file.
- [in] *start\_value* The start index value of the plot (optional).

Load a Milena array from a plot file.

#### Parameters

- [in] *arr* A reference to the array to load.
- [out] *filename* The output file.

Definition at line 93 of file plot/load.hh.

References `mln::util::array< T >::append()`, and `mln::util::array< T >::clear()`.

#### 9.87.2.2 `template<typename T > void mln::io::plot::save ( const histo::array< T > & arr, const std::string & filename ) [inline]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 103 of file plot/save.hh.

**9.87.2.3** `template<typename T > void mln::io::plot::save ( const util::array< T > & arr, const std::string & filename, int start_value = 0 ) [inline]`

Save a Milena array in a plot file.

#### Parameters

- [in] *arr* A reference to the array to save.
- [out] *filename* The output file.
- [in] *start\_value* The start index value of the plot (optional).

Definition at line 89 of file plot/save.hh.

References `mln::util::array< T >::nelements()`.

**9.87.2.4** `template<typename I > void mln::io::plot::save ( const image1d< I > & ima, const std::string & filename )`

Save a Milena 1D image in a plot file.

#### Parameters

- [in] *ima* A reference to the image to save.
- [out] *filename* The output file.

## 9.88 mln::io::pnm Namespace Reference

Namespace of pnm input/output handling.

### Namespaces

- namespace [impl](#)  
*Namespace of pnm's implementation details.*

### Functions

- `template<typename V > image2d< V > load (char type_, const std::string &filename)`  
*main function : load pnm format*
- `template<typename I > void load (char type_, Image< I > &ima_, const std::string &filename)`  
*An other way to load pnm files : the destination is an argument to check if the type match the file to load.*
- `template<typename I > void load_ascii_builtin (std::ifstream &file, I &ima)`  
*load\_ascii for builtin value types.*

- `template<typename I >`  
void `load_ascii_value` (`std::ifstream &file`, `I &ima`)  
*load\_ascii for Milena value types.*
- `template<typename I >`  
void `load_raw_2d` (`std::ifstream &file`, `I &ima`)  
*load\_raw\_2d.*
- `template<typename V >`  
unsigned int `max_component` (`const V &`)  
*Give the maximum value which can be stored as a component value type V.*
- `template<typename I >`  
void `save` (`char type`, `const Image< I > &ima_`, `const std::string &filename`)

### 9.88.1 Detailed Description

Namespace of pnm input/output handling.

### 9.88.2 Function Documentation

#### 9.88.2.1 `template<typename V > image2d<V> mln::io::pnm::load ( char type_, const std::string & filename ) [inline]`

main function : load pnm format

Definition at line 210 of file `pnm/load.hh`.

References `load_raw_2d()`, and `max_component()`.

#### 9.88.2.2 `template<typename I > void mln::io::pnm::load ( char type_, Image< I > & ima_, const std::string & filename ) [inline]`

An other way to load pnm files : the destination is an argument to check if the type match the file to load.

Definition at line 257 of file `pnm/load.hh`.

References `mln::make::box2d()`, `load_raw_2d()`, and `max_component()`.

#### 9.88.2.3 `template<typename I > void mln::io::pnm::load_ascii_builtin ( std::ifstream & file, I & ima ) [inline]`

`load_ascii` for builtin value types.

Definition at line 178 of file `pnm/load.hh`.

#### 9.88.2.4 `template<typename I > void mln::io::pnm::load_ascii_value ( std::ifstream & file, I & ima ) [inline]`

`load_ascii` for Milena value types.

Definition at line 164 of file `pnm/load.hh`.

**9.88.2.5** `template<typename I > void mln::io::pnm::load_raw_2d ( std::ifstream & file, I & ima ) [inline]`

load\_raw\_2d.

for all pnm 8/16 bits formats

Definition at line 198 of file pnm/load.hh.

Referenced by load().

**9.88.2.6** `template<typename V > unsigned int mln::io::pnm::max_component ( const V & ) [inline]`

Give the maximum value which can be stored as a component value type V.

Definition at line 56 of file max\_component.hh.

Referenced by load().

**9.88.2.7** `template<typename I > void mln::io::pnm::save ( char type, const Image<I > & ima_, const std::string & filename ) [inline]`

Save a Milena image as a pnm image.

#### Parameters

[in] *type* The type of the image to save (can be PPM, PGM, PBM).

[in] *ima\_* The image to save.

[in, out] *filename* the destination.

Definition at line 185 of file pnm/save.hh.

## 9.89 mln::io::pnm::impl Namespace Reference

Namespace of pnm's implementation details.

### 9.89.1 Detailed Description

Namespace of pnm's implementation details.

## 9.90 mln::io::pnms Namespace Reference

Namespace of pnms input/output handling.

### Functions

- `template<typename V > void load (char type, image3d< V > &ima, const util::array< std::string > &filenames)`

*Load pnm images as slices of a 3D Milena image.*

- void `load` (char type, `image3d`< bool > &ima, const `util::array`< std::string > &filenames)

### 9.90.1 Detailed Description

Namespace of pnms input/output handling.

### 9.90.2 Function Documentation

**9.90.2.1** `template<typename V > void mln::io::pnms::load ( char type, image3d< V > & ima, const util::array< std::string > & filenames ) [inline]`

Load pnm images as slices of a 3D Milena image.

#### Parameters

- [in] *type* The type of the pnm files.
- [out] *ima* A reference to the 3D image which will receive data.
- [in] *filenames* The list of 2D images to load..

Definition at line 79 of file pnms/load.hh.

References `mln::make::image3d()`, `mln::util::array< T >::is_empty()`, and `mln::util::array< T >::nelements()`.

Referenced by `load()`.

**9.90.2.2** `void mln::io::pnms::load ( char type, image3d< bool > & ima, const util::array< std::string > & filenames ) [inline]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 102 of file pnms/load.hh.

References `mln::make::image3d()`, `mln::util::array< T >::is_empty()`, `load()`, and `mln::util::array< T >::nelements()`.

## 9.91 mln::io::ppm Namespace Reference

Namespace of ppm input/output handling.

### Functions

- `template<typename I > void load (Image< I > &ima, const std::string &filename)`

*Load a ppm image in a Milena image.*



- `template<typename V >`  
`image2d< V > load` (const std::string &filename)  
*Load a ppm image in a Milena image.*
- `template<typename I >`  
`void save` (const `Image< I > &ima`, const std::string &filename)

### 9.91.1 Detailed Description

Namespace of ppm input/output handling.

### 9.91.2 Function Documentation

#### 9.91.2.1 `template<typename I > void mln::io::ppm::load ( Image< I > & ima, const std::string & filename ) [inline]`

Load a ppm image in a Milena image.

##### Parameters

- [out] *ima* A reference to the image which will receive data.
- [in] *filename* The source.

Definition at line 89 of file ppm/load.hh.

#### 9.91.2.2 `template<typename V > image2d< V > mln::io::ppm::load ( const std::string & filename ) [inline]`

Load a ppm image in a Milena image.

To use this routine, you should specialize the template with the value type of the image loaded. (ex : `load<value::int_u8>("...")`)

##### Parameters

- [in] *filename* The image source.

##### Returns

An `image2d` which contains loaded data.

Definition at line 79 of file ppm/load.hh.

#### 9.91.2.3 `template<typename I > void mln::io::ppm::save ( const Image< I > & ima, const std::string & filename ) [inline]`

Save a Milena image as a ppm image.

##### Parameters

- [in] *ima* The image to save.

[in, out] *filename* the destination.

Definition at line 65 of file ppm/save.hh.

Referenced by mln::registration::icp().

## 9.92 mln::io::ppms Namespace Reference

Namespace of ppms input/output handling.

### Functions

- `template<typename V > void load (image3d< V > &ima, const util::array< std::string > &filenames)`  
*Load ppm images as slices of a 3D Milena image.*

### 9.92.1 Detailed Description

Namespace of ppms input/output handling.

### 9.92.2 Function Documentation

#### 9.92.2.1 `template<typename V > void mln::io::ppms::load ( image3d< V > & ima, const util::array< std::string > & filenames ) [inline]`

Load ppm images as slices of a 3D Milena image.

#### Parameters

[out] *ima* A reference to the 3D image which will receive data.

[in] *filenames* The list of 2D images to load..

Definition at line 67 of file ppm/load.hh.

## 9.93 mln::io::raw Namespace Reference

Namespace of raw input/output handling.

### Classes

- struct `raw_header`  
*Store raw file header.*

## Functions

- `raw_header get_header` (const std::string &filename)  
*Retrieve header in a raw file.*
- `template<typename I > void load` (Image< I > &ima\_, const std::string &filename)  
*Load an image saved as a raw data file.*
- `template<typename I > void save` (const Image< I > &ima\_, const std::string &filename)  
*Save a Milena image as a raw data file.*

### 9.93.1 Detailed Description

Namespace of raw input/output handling.

### 9.93.2 Function Documentation

#### 9.93.2.1 raw\_header mln::io::raw::get\_header ( const std::string & filename )

Retrieve header in a raw file.

Definition at line 68 of file raw/get\_header.hh.

References mln::util::array< T >::resize().

#### 9.93.2.2 template<typename I > void mln::io::raw::load ( Image< I > & ima\_, const std::string & filename )

Load an image saved as a raw data file.

##### Parameters

[in, out] *ima\_* The image to load.

[in] *filename* the destination.

This routine try to read two input files: 'filename' and 'filename.info'. 'filename' is the raw data. 'filename.info' store various information about the image.

Definition at line 184 of file raw/load.hh.

#### 9.93.2.3 template<typename I > void mln::io::raw::save ( const Image< I > & ima\_, const std::string & filename )

Save a Milena image as a raw data file.

##### Parameters

[in] *ima\_* The image to save.

[in] *filename* the destination.

This routine produce two output files: 'filename' and 'filename.info'. 'filename' is the raw data. 'filename.info' store various information about the image.

Definition at line 135 of file raw/save.hh.

## 9.94 mln::io::tiff Namespace Reference

Namespace of tiff input/output handling.

### Functions

- `template<typename I >`  
`void load (Image< I > &ima_, const std::string &filename)`  
*Load a TIFF image to a Milena image.*

### 9.94.1 Detailed Description

Namespace of tiff input/output handling.

### 9.94.2 Function Documentation

- 9.94.2.1 `template<typename I > void mln::io::tiff::load ( Image< I > & ima_, const std::string & filename ) [inline]`

Load a TIFF image to a Milena image.

Definition at line 323 of file tiff/load.hh.

## 9.95 mln::io::txt Namespace Reference

Namespace of txt input/output handling.

### Functions

- `void save (const image2d< char > &ima, const std::string &filename)`  
*Save an image as txt file.*

### 9.95.1 Detailed Description

Namespace of txt input/output handling.

## 9.95.2 Function Documentation

**9.95.2.1** `void mln::io::txt::save ( const image2d< char > & ima, const std::string & filename )`  
`[inline]`

Save an image as txt file.

### Parameters

- [in] *ima* The image to save. Must be an image of char.
- [in] *filename* the destination.

Definition at line 63 of file txt/save.hh.

References `mln::image2d< T >::domain()`.

## 9.96 mln::labeling Namespace Reference

Namespace of labeling routines.

### Namespaces

- namespace [impl](#)  
*Implementation namespace of labeling namespace.*

### Functions

- `template<typename I, typename N, typename L >`  
`mln::trait::ch_value< I, L >::ret background (const Image< I > &input, const Neighborhood< N > &nbh, L &nlabels)`
- `template<typename I, typename N, typename L >`  
`mln::trait::ch_value< I, L >::ret blobs (const Image< I > &input, const Neighborhood< N > &nbh, L &nlabels)`  
*Connected component labeling of the binary objects of a binary image.*
- `template<typename I, typename N, typename L, typename A >`  
`util::couple< mln::trait::ch_value< I, L >::ret, util::couple< util::array< typename A::result >, util::array< A > > > blobs\_and\_compute (const Image< I > &input, const Neighborhood< N > &nbh, L &nlabels, const Accumulator< A > &accu)`
- `template<typename V, typename L >`  
`mln::trait::ch_value< L, V >::ret colorize (const V &value, const Image< L > &labeled_image, const typename L::value &nlabels)`  
*Create a new color image from a labeled image and fill each component with a random color.*
- `template<typename V, typename L >`  
`mln::trait::ch_value< L, V >::ret colorize (const V &value, const Image< L > &labeled_image)`
- `template<typename L >`  
`mln::trait::ch_value< L, mln::value::rgb8 >::ret colorize (const Image< L > &input, const typename L::value &nlabels)`

- `template<typename A , typename I , typename L >`  
`util::array< mln_meta_accu_result(A, typename I::value)> compute (const Meta_Accumulator< A`  
`> &a, const Image< I > &input, const Image< L > &label, const typename L::value &nlabels)`  
*Compute an accumulator onto the pixel values of the image `input`.*
- `template<typename A , typename L >`  
`util::array< typename A::result > compute (const Accumulator< A > &a, const Image< L > &la-`  
`bel, const typename L::value &nlabels)`  
*Compute an accumulator onto the pixel sites of each component domain of `label`.*
- `template<typename A , typename L >`  
`util::array< mln_meta_accu_result(A, typename L::psite)> compute (const Meta_Accumulator< A`  
`> &a, const Image< L > &label, const typename L::value &nlabels)`  
*Compute an accumulator onto the pixel sites of each component domain of `label`.*
- `template<typename A , typename I , typename L >`  
`util::array< typename A::result > compute (util::array< A > &a, const Image< I > &input, const`  
`Image< L > &label, const typename L::value &nlabels)`  
*Compute an accumulator onto the pixel values of the image `input`.*
- `template<typename A , typename I , typename L >`  
`util::array< typename A::result > compute (const Accumulator< A > &a, const Image< I > &in-`  
`put, const Image< L > &label, const typename L::value &nlabels)`  
*Compute an accumulator onto the pixel values of the image `input`.*
- `template<typename A , typename I , typename L >`  
`mln::trait::ch_value< L, typename A::result >::ret compute_image (const util::array< typename`  
`A::result > &a, const Image< I > &input, const Image< L > &labels, const typename L::value`  
`&nlabels)`  
*Compute an accumulator onto the pixel values of the image `input`.*
- `template<typename A , typename I , typename L >`  
`mln::trait::ch_value< L, typename A::result >::ret compute_image (const Accumulator< A >`  
`&accu, const Image< I > &input, const Image< L > &labels, const typename L::value &nlabels)`  
*Compute an accumulator onto the pixel values of the image `input`.*
- `template<typename A , typename I , typename L >`  
`mln::trait::ch_value< L, typename mln::internal::meta_accu_ret_result_helper< A, typename`  
`I::value >::result >::ret compute_image (const Meta_Accumulator< A > &accu, const Image<`  
`I > &input, const Image< L > &labels, const typename L::value &nlabels)`  
*Compute an accumulator onto the pixel values of the image `input`.*
- `template<typename I , typename N , typename L >`  
`mln::trait::concrete< I >::ret fill_holes (const Image< I > &input, const Neighborhood< N >`  
`&nbh, L &nlabels)`  
*Filling holes of a single object in a binary image.*
- `template<typename I , typename N , typename L >`  
`mln::trait::ch_value< I, L >::ret flat_zones (const Image< I > &input, const Neighborhood< N >`  
`&nbh, L &nlabels)`  
*Connected component labeling of the flat zones of an image.*

- `template<typename I, typename N, typename L >`  
`mln::trait::ch_value< I, L >::ret foreground (const Image< I > &input, const Neighborhood< N > &nbh, L &nlabels)`
- `template<typename I >`  
`mln::trait::concrete< I >::ret pack (const Image< I > &label, typename I::value &new_nlabels, fun::i2v::array< typename I::value > &repack_fun)`

*Relabel a labeled image in order to have a contiguous labeling.*

- `template<typename I >`  
`mln::trait::concrete< I >::ret pack (const Image< I > &label, typename I::value &new_nlabels)`
- `template<typename I >`  
`void pack_inplace (Image< I > &label, typename I::value &new_nlabels)`
- `template<typename I >`  
`void pack_inplace (Image< I > &label, typename I::value &new_nlabels, fun::i2v::array< typename I::value > &repack_fun)`

*Relabel inplace a labeled image in order to have a contiguous labeling.*

- `template<typename I, typename N, typename L >`  
`mln::trait::ch_value< I, L >::ret regional_maxima (const Image< I > &input, const Neighborhood< N > &nbh, L &nlabels)`
- `template<typename I, typename N, typename L >`  
`mln::trait::ch_value< I, L >::ret regional_minima (const Image< I > &input, const Neighborhood< N > &nbh, L &nlabels)`
- `template<typename I, typename F >`  
`mln::trait::concrete< I >::ret relabel (const Image< I > &label, const typename I::value &nlabels, typename I::value &new_nlabels, const Function_v2b< F > &fv2b)`

*Remove components and relabel a labeled image.*

- `template<typename I, typename F >`  
`mln::trait::concrete< I >::ret relabel (const Image< I > &label, const typename I::value &nlabels, const Function_v2v< F > &fv2v)`

*Remove components and relabel a labeled image.*

- `template<typename I, typename F >`  
`void relabel_inplace (Image< I > &label, const typename I::value &nlabels, const Function_v2v< F > &fv2v)`

*Remove components and relabel a labeled image inplace.*

- `template<typename I, typename F >`  
`void relabel_inplace (Image< I > &label, const typename I::value &nlabels, const Function_v2b< F > &fv2b)`

*Remove components and relabel a labeled image inplace.*

- `template<typename I, typename J >`  
`mln::trait::concrete< I >::ret superpose (const Image< I > &lhs, const typename I::value &lhs_nlabels, const Image< J > &rhs, const typename J::value &rhs_nlabels, typename I::value &new_nlabels)`

*Superpose two labeled image.*

- `template<typename I, typename N, typename L >`  
`mln::trait::ch_value< I, L >::ret value` (const `Image< I >` &input, const `typename I::value` &val, const `Neighborhood< N >` &nbh, L &nlabels)  
*Connected component labeling of the image sites at a given value.*
- `template<typename I, typename N, typename L, typename A >`  
`util::couple< mln::trait::ch_value< I, L >::ret, util::couple< util::array< typename A::result >, util::array< A > > >` `value_and_compute` (const `Image< I >` &input, const `typename I::value` &val, const `Neighborhood< N >` &nbh, L &nlabels, const `Accumulator< A >` &accu)  
*Connected component labeling of the image sites at a given value.*
- `template<typename V, typename I >`  
`mln::trait::ch_value< I, V >::ret wrap` (const `V` &value\_type, const `Image< I >` &input)  
*Wrap labels such as 0 -> 0 and [1, lmax] maps to [1, Lmax] (using modulus).*
- `template<typename I >`  
`mln::trait::ch_value< I, mln::value::label_8 >::ret wrap` (const `Image< I >` &input)  
*Wrap labels such as 0 -> 0 and [1, lmax] maps to [1, Lmax] (using modulus).*

## 9.96.1 Detailed Description

Namespace of labeling routines.

## 9.96.2 Function Documentation

**9.96.2.1** `template<typename I, typename N, typename L > mln::trait::ch_value< I, L >::ret mln::labeling::background ( const Image< I > & input, const Neighborhood< N > & nbh, L & nlabels ) [inline]`

Connected component labeling of the background part in a binary image.

### Parameters

- [in] *input* The input image.
- [in] *nbh* The connexity of the background.
- [out] *nlabels* The number of labels.

### Returns

The label image.

### Precondition

The input image has to be binary (checked at compile-time).

This routine actually calls `mln::labeling::value` with the value set to `false`.

### See also

[mln::labeling::value](#)



Definition at line 69 of file background.hh.

References value().

Referenced by fill\_holes().

**9.96.2.2** `template<typename I , typename N , typename L > mln::trait::ch_value< I, L >::ret  
mln::labeling::blobs ( const Image< I > & input, const Neighborhood< N > & nbh, L  
& nlabels ) [inline]`

Connected component labeling of the binary objects of a binary image.

#### Parameters

[in] *input* The input image.

[in] *nbh* The connexity of the objects.

[out] *nlabels* The Number of labels. Its value is set in the algorithms.

#### Returns

The label image.

#### Precondition

The input image has to be binary (checked at compile-time).

A fast queue is used so that the algorithm is not recursive and can handle large binary objects (blobs).

Definition at line 102 of file labeling/blobs.hh.

Referenced by blobs\_and\_compute(), and mln::graph::labeling().

**9.96.2.3** `template<typename I , typename N , typename L , typename A > util::couple<  
mln::trait::ch_value< I, L >::ret, util::couple< util::array< typename A::result > ,  
util::array< A > > > mln::labeling::blobs_and_compute ( const Image< I > & input,  
const Neighborhood< N > & nbh, L & nlabels, const Accumulator< A > & accu )`

Label an image and compute given accumulators.

#### Parameters

[in] *input* A binary image.

[in] *nbh* A neighborhood used for labeling.

[in, out] *nlabels* The number of labels found.

[in] *accu* An accumulator to be computed while labeling.

#### Returns

The labeled image, computed attributes for each regions and an array of the accumulators used to compute the attributes.

Definition at line 160 of file blobs\_and\_compute.hh.

References blobs(), and mln::make::couple().

**9.96.2.4** `template<typename V , typename L > mln::trait::ch_value< L, V >::ret  
mln::labeling::colorize ( const V & value, const Image< L > & labeled_image, const  
typename L::value & nlabels ) [inline]`

Create a new color image from a labeled image and fill each component with a random color.

`litera::black` is used for component 0, e.g. the background. Min and max values for RGB values can be set through the global variables `mln::labeling::colorize_::min_value` and `mln::labeling::colorize_::max_value`.

#### Parameters

- [in] *value* value type used in the returned image.
- [in] *labeled\_image* A labeled image (

#### See also

[labeling::blobs](#)).

#### Parameters

- [in] *nlabels* Number of labels.

Definition at line 190 of file `colorize.hh`.

References `mln::literal::black`, and `mln::data::transform()`.

Referenced by `colorize()`.

**9.96.2.5** `template<typename V , typename L > mln::trait::ch_value< L, V >::ret  
mln::labeling::colorize ( const V & value, const Image< L > & labeled_image )  
[inline]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 228 of file `colorize.hh`.

References `colorize()`, and `compute()`.

**9.96.2.6** `template<typename L > mln::trait::ch_value< L, mln::value::rgb8 >::ret  
mln::labeling::colorize ( const Image< L > & input, const typename L::value & nlabels  
) [inline]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 247 of file `colorize.hh`.

References `colorize()`.

**9.96.2.7** `template<typename A , typename I , typename L > util::array<  
mln_meta_accu_result(A, typename I::value)> mln::labeling::compute ( const  
Meta_Accumulator< A > & a, const Image< I > & input, const Image< L > & label,  
const typename L::value & nlabels ) [inline]`

Compute an accumulator onto the pixel values of the image `input`.

for each component of the image `label`.

**Parameters**

- [in] *a* A meta-accumulator.
- [in] *input* The input image.
- [in] *label* The labeled image.
- [in] *nlabels* The number of labels in *label*.

**Returns**

A [util::array](#) of accumulator result (one result per label).

Definition at line 734 of file labeling/compute.hh.

References [compute\(\)](#).

**9.96.2.8** `template<typename A , typename L > util::array< typename A::result >  
mln::labeling::compute ( const Accumulator< A > & a, const Image< L > & label,  
const typename L::value & nlabels ) [inline]`

Compute an accumulator onto the pixel sites of each component domain of *label*.

**Parameters**

- [in] *a* An accumulator.
- [in] *label* The labeled image.
- [in] *nlabels* The number of labels in *label*.

**Returns**

A [util::array](#) of accumulator result (one result per label).

Definition at line 771 of file labeling/compute.hh.

**9.96.2.9** `template<typename A , typename L > util::array< mln_meta_accu_result(A, typename  
L::psite)> mln::labeling::compute ( const Meta_Accumulator< A > & a, const Image<  
L > & label, const typename L::value & nlabels ) [inline]`

Compute an accumulator onto the pixel sites of each component domain of *label*.

**Parameters**

- [in] *a* A meta-accumulator.
- [in] *label* The labeled image.
- [in] *nlabels* The number of labels in *label*.

**Returns**

A [util::array](#) of accumulator result (one result per label).

Definition at line 792 of file labeling/compute.hh.

References [compute\(\)](#).

**9.96.2.10** `template<typename A , typename I , typename L > util::array< typename A::result > mln::labeling::compute ( util::array< A > & a, const Image< I > & input, const Image< L > & label, const typename L::value & nlabels ) [inline]`

Compute an accumulator onto the pixel values of the image `input`.  
for each component of the image `label`.

#### Parameters

- [in] *a* An array of accumulator.
- [in] *input* The input image.
- [in] *label* The labeled image.
- [in] *nlabels* The number of labels in `label`.

#### Returns

A `util::array` of accumulator result (one result per label).

Definition at line 696 of file `labeling/compute.hh`.

**9.96.2.11** `template<typename A , typename I , typename L > util::array< typename A::result > mln::labeling::compute ( const Accumulator< A > & a, const Image< I > & input, const Image< L > & label, const typename L::value & nlabels ) [inline]`

Compute an accumulator onto the pixel values of the image `input`.  
for each component of the image `label`.

#### Parameters

- [in] *a* An accumulator.
- [in] *input* The input image.
- [in] *label* The labeled image.
- [in] *nlabels* The number of labels in `label`.

#### Returns

A `util::array` of accumulator result (one result per label).

Definition at line 715 of file `labeling/compute.hh`.

Referenced by `colorize()`, `compute()`, `compute_image()`, `fill_holes()`, `mln::make::p_edges_with_mass_centers()`, `mln::make::p_vertices_with_mass_centers()`, `pack()`, and `pack_inplace()`.

**9.96.2.12** `template<typename A , typename I , typename L > mln::trait::ch_value< L , typename A ::result >::ret mln::labeling::compute_image ( const util::array< typename A::result > & a, const Image< I > & input, const Image< L > & labels, const typename L::value & nlabels )`

Compute an accumulator onto the pixel values of the image `input`.  
for each component of the image `label`.

**Parameters**

- [in] *a* The [mln::p\\_array](#) of accumulator result.
- [in] *input* The input image (values).
- [in] *labels* The label image.
- [in] *nlabels* The count of labels.

**Returns**

The image where labels are replaced by the result of the accumulator.

Referenced by `compute_image()`.

**9.96.2.13** `template<typename A , typename I , typename L > mln::trait::ch_value< L, typename A::result >::ret mln::labeling::compute_image ( const Accumulator< A > & accu, const Image< I > & input, const Image< L > & labels, const typename L::value & nlabels ) [inline]`

Compute an accumulator onto the pixel values of the image `input`.  
for each component of the image `label`.

**Parameters**

- [in] *accu* The accumulator.
- [in] *input* The input image (values).
- [in] *labels* The label image.
- [in] *nlabels* The count of labels.

**Returns**

The image where labels are replaced by the result of the accumulator.

Definition at line 161 of file `compute_image.hh`.

References `compute()`, and `compute_image()`.

**9.96.2.14** `template<typename A , typename I , typename L > mln::trait::ch_value< L, typename mln::internal::meta_accu_ret_result_helper< A, typename I::value >::result >::ret mln::labeling::compute_image ( const Meta_Accumulator< A > & accu, const Image< I > & input, const Image< L > & labels, const typename L::value & nlabels ) [inline]`

Compute an accumulator onto the pixel values of the image `input`.  
for each component of the image `label`.

**Parameters**

- [in] *accu* The meta-accumulator.
- [in] *input* The input image (values).
- [in] *labels* The label image.
- [in] *nlabels* The count of labels.

**Returns**

The image where labels are replaced by the result of the accumulator.

Definition at line 181 of file `compute_image.hh`.

References `compute()`, and `compute_image()`.

**9.96.2.15** `template<typename I, typename N, typename L> mln::trait::concrete<I>::ret  
mln::labeling::fill_holes ( const Image<I> & input, const Neighborhood<N> &  
nbh, L & nlabels ) [inline]`

Filling holes of a single object in a binary image.

**Parameters**

[in] *input* The input image.

[in] *nbh* The connexity of the background.

[out] *nlabels* The number of labels.

**Returns**

The binary image with a simple object without holes.

**Precondition**

The input image has to be binary (checked at compile-time).

This routine actually calls [mln::labeling::background](#)

**See also**

[mln::labeling::background](#)

Definition at line 73 of file `fill_holes.hh`.

References `background()`, `compute()`, `mln::util::array<T>::nelements()`, and `mln::data::transform()`.

**9.96.2.16** `template<typename I, typename N, typename L> mln::trait::ch_value<I, L>::ret  
mln::labeling::flat_zones ( const Image<I> & input, const Neighborhood<N> &  
nbh, L & nlabels )`

Connected component labeling of the flat zones of an image.

**Parameters**

[in] *input* The input image.

[in] *nbh* The connexity of the flat zones.

[out] *nlabels* The number of labels.

**Returns**

The label image.

Definition at line 124 of file `flat_zones.hh`.

**9.96.2.17** `template<typename I, typename N, typename L > mln::trait::ch_value< I, L >::ret mln::labeling::foreground ( const Image< I > & input, const Neighborhood< N > & nbh, L & nlabels ) [inline]`

Connected component labeling of the object part in a binary image.

#### Parameters

- [in] *input* The input image.
- [in] *nbh* The connexity of the foreground.
- [out] *nlabels* The number of labels.

#### Returns

The label image.

#### Precondition

The input image has to be binary (checked at compile-time).

This routine actually calls [mln::labeling::value](#) with the value set to `true`.

#### See also

[mln::labeling::value](#)

Definition at line 69 of file `foreground.hh`.

References `value()`.

**9.96.2.18** `template<typename I > mln::trait::concrete< I >::ret mln::labeling::pack ( const Image< I > & label, typename I::value & new_nlabels, fun::i2v::array< typename I::value > & repack_fun )`

Relabel a labeled image in order to have a contiguous labeling.

#### Parameters

- [in] *label* The labeled image.
- [out] *new\_nlabels* The number of labels after relabeling.
- [out] *repack\_fun* The function used to repack the labels.

#### Returns

The relabeled image.

Definition at line 124 of file `pack.hh`.

References `compute()`, `mln::make::relabelfun()`, and `mln::data::transform()`.

Referenced by `pack()`.

**9.96.2.19** `template<typename I > mln::trait::concrete< I >::ret mln::labeling::pack ( const Image< I > & label, typename I::value & new_nlabels )`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 115 of file pack.hh.

References pack().

**9.96.2.20** `template<typename I > void mln::labeling::pack_inplace ( Image< I > & label, typename I::value & new_nlabels )`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 148 of file pack.hh.

References pack\_inplace().

**9.96.2.21** `template<typename I > void mln::labeling::pack_inplace ( Image< I > & label, typename I::value & new_nlabels, fun::i2v::array< typename I::value > & repack_fun )`

Relabel inplace a labeled image in order to have a contiguous labeling.

#### Parameters

- [in] *label* The labeled image.
- [out] *new\_nlabels* The number of labels after relabeling.
- [out] *repack\_fun* The function used to repack the labels.

Definition at line 157 of file pack.hh.

References compute(), mln::make::relabelfun(), and mln::data::transform().

Referenced by pack\_inplace().

**9.96.2.22** `template<typename I, typename N, typename L > mln::trait::ch_value< I, L >::ret mln::labeling::regional_maxima ( const Image< I > & input, const Neighborhood< N > & nbh, L & nlabels )`

Connected component labeling of the regional maxima of an image.

#### Parameters

- [in] *input* The input image.
- [in] *nbh* The connexity of the regional maxima.
- [out] *nlabels* The number of labeled regions.

#### Returns

The label image.

Definition at line 130 of file regional\_maxima.hh.



**9.96.2.23** `template<typename I , typename N , typename L > mln::trait::ch_value< I, L >::ret  
mln::labeling::regional_minima ( const Image< I > & input, const Neighborhood< N  
> & nbh, L & nlabels )`

Connected component labeling of the regional minima of an image.

#### Parameters

- [in] *input* The input image.
- [in] *nbh* The connexity of the regional minima.
- [out] *nlabels* The number of labeled regions.

#### Returns

The label image.

Definition at line 140 of file regional\_minima.hh.

Referenced by mln::morpho::meyer\_wst().

**9.96.2.24** `template<typename I , typename F > mln::trait::concrete< I >::ret  
mln::labeling::relabel ( const Image< I > & label, const typename I::value & nlabels,  
typename I::value & new_nlabels, const Function_v2b< F > & fv2b ) [inline]`

Remove components and relabel a labeled image.

#### Parameters

- [in] *label* the labeled image.
- [in] *nlabels* the number of labels in label.
- [out] *new\_nlabels* the number of labels after relabeling.
- [in] *fv2b* function returning whether a label must be replaced by the background.

#### Returns

the relabeled image.

Definition at line 200 of file relabel.hh.

References mln::make::relabelfun().

Referenced by superpose().

**9.96.2.25** `template<typename I , typename F > mln::trait::concrete< I >::ret  
mln::labeling::relabel ( const Image< I > & label, const typename I::value & nlabels,  
const Function_v2v< F > & fv2v ) [inline]`

Remove components and relabel a labeled image.

#### Parameters

- [in] *label* the labeled image.
- [in] *nlabels* the number of labels in label.

[in] *fv2v* function returning the new component id for each pixel value.

### Returns

the relabeled image.

Definition at line 179 of file relabel.hh.

References `mln::data::transform()`.

**9.96.2.26** `template<typename I , typename F > void mln::labeling::relabel_inplace ( Image< I > & label, const typename I::value & nlabels, const Function_v2v< F > & fv2v )`  
**[inline]**

Remove components and relabel a labeled image inplace.

### Parameters

[in, out] *label* the labeled image.

[in] *nlabels* the number of labels in *label*.

[in] *fv2v* function returning the new component id for each pixel value.

Definition at line 221 of file relabel.hh.

References `mln::data::transform_inplace()`.

**9.96.2.27** `template<typename I , typename F > void mln::labeling::relabel_inplace ( Image< I > & label, const typename I::value & nlabels, const Function_v2b< F > & fv2b )`  
**[inline]**

Remove components and relabel a labeled image inplace.

### Parameters

[in, out] *label* the labeled image.

[in] *nlabels* the number of labels in *label*.

[in] *fv2b* function returning whether a label must be replaced by the background.

Definition at line 240 of file relabel.hh.

References `mln::make::relabelfun()`.

Referenced by `mln::labeled_image_base< I, E >::relabel()`.

**9.96.2.28** `template<typename I , typename J > mln::trait::concrete< I >::ret mln::labeling::superpose ( const Image< I > & lhs, const typename I::value & lhs_nlabels, const Image< J > & rhs, const typename J::value & rhs_nlabels, typename I::value & new_nlabels )`

Superpose two labeled image.

Labels in *lhs* are preserved in the output. Labels of *rhs* are renumbered from the last label value of *lhs*. It avoids duplicate label values in several components.

**Parameters**

- [in] *lhs* A labeled image.
- [in] *lhs\_nlabels* The number of labels in lhs.
- [in] *rhs* A labeled image.
- [in] *rhs\_nlabels* The number of labels in rhs.
- [out] *new\_nlabels* The number of labels in the output image.

**Returns**

An image with all the components of rhs and lhs.

**Precondition**

- rhs and lhs must have the same domain.
- The value type of rhs must be convertible towards lhs's.

Definition at line 83 of file labeling/superpose.hh.

References mln::duplicate(), mln::value::equiv(), mln::data::paste(), relabel(), and mln::literal::zero.

**9.96.2.29** `template<typename I , typename N , typename L > mln::trait::ch_value< I, L >::ret  
mln::labeling::value ( const Image< I > & input, const typename I::value & val, const  
Neighborhood< N > & nbh, L & nlabels )`

Connected component labeling of the image sites at a given value.

**Parameters**

- [in] *input* The input image.
- [in] *val* The value to consider.
- [in] *nbh* The connectivity of components.
- [out] *nlabels* The number of labels.

**Returns**

The label image.

Definition at line 149 of file labeling/value.hh.

Referenced by background(), and foreground().

**9.96.2.30** `template<typename I , typename N , typename L , typename A > util::couple<  
mln::trait::ch_value< I, L >::ret, util::couple< util::array< typename A::result > ,  
util::array< A > > > mln::labeling::value_and_compute ( const Image< I > & input,  
const typename I::value & val, const Neighborhood< N > & nbh, L & nlabels, const  
Accumulator< A > & accu )`

Connected component labeling of the image sites at a given value.

**Parameters**

- [in] *input* The input image.

- [in] *val* The value to consider.
- [in] *nbh* The connectivity of components.
- [out] *nlabels* The number of labels.

**Returns**

The label image.

Definition at line 212 of file `value_and_compute.hh`.

References `mln::make::couple()`.

**9.96.2.31** `template<typename V , typename I > mln::trait::ch_value< I, V >::ret  
mln::labeling::wrap ( const V & value_type, const Image< I > & input ) [inline]`

Wrap labels such as 0 -> 0 and [1, lmax] maps to [1, Lmax] (using modulus).

**Parameters**

- [in] *value\_type* The type used to wrap the label type.
- [in] *input* The label image.

**Returns**

A new image with values wrapped with type V.

Definition at line 75 of file `labeling/wrap.hh`.

References `mln::data::transform()`.

Referenced by `wrap()`.

**9.96.2.32** `template<typename I > mln::trait::ch_value< I, mln::value::label_8 >::ret  
mln::labeling::wrap ( const Image< I > & input ) [inline]`

Wrap labels such as 0 -> 0 and [1, lmax] maps to [1, Lmax] (using modulus).

Use `label_8` as label type.

**Parameters**

- [in] *input* The label image.

**Returns**

A new image with values wrapped with type `label_8`.

Definition at line 93 of file `labeling/wrap.hh`.

References `wrap()`.

## 9.97 mln::labeling::impl Namespace Reference

Implementation namespace of labeling namespace.

## Namespaces

- namespace [generic](#)

*Generic implementation namespace of labeling namespace.*

## Functions

- `template<typename A , typename I , typename L > util::array< typename A::result > compute_fastest (const Accumulator< A > &a_, const Image< I > &input_, const Image< L > &label_, const typename L::value &nlabels)`

*Fastest implementation of [labeling::compute](#).*

- `template<typename A , typename I , typename L > util::array< typename A::result > compute_fastest (util::array< A > &accus, const Image< I > &input_, const Image< L > &label_, const typename L::value &nlabels)`

*Fastest implementation of [labeling::compute](#).*

### 9.97.1 Detailed Description

Implementation namespace of labeling namespace.

### 9.97.2 Function Documentation

**9.97.2.1** `template<typename A , typename I , typename L > util::array<typename A ::result> mln::labeling::impl::compute_fastest ( const Accumulator< A > & a_, const Image< I > & input_, const Image< L > & label_, const typename L::value & nlabels )`  
**[inline]**

Fastest implementation of [labeling::compute](#).

#### Parameters

- [in] *a\_* An accumulator.
- [in] *input\_* The input image.
- [in] *label\_* The labeled image.
- [in] *nlabels* The number of labels in *label*.

#### Returns

A [util::array](#) of accumulator result (one result per label).

Definition at line 373 of file `labeling/compute.hh`.

References `mln::geom::ncols()`.

**9.97.2.2** `template<typename A , typename I , typename L > util::array<typename A ::result> mln::labeling::impl::compute_fastest ( util::array< A > & accus, const Image< I > & input_, const Image< L > & label_, const typename L::value & nlabels ) [inline]`

Fastest implementation of `labeling::compute`.

#### Parameters

- [in] *accus* An array of accumulators.
- [in] *input\_* The input image.
- [in] *label\_* The labeled image.
- [in] *nlabels* The number of labels in `label`.

#### Returns

A `util::array` of accumulator result (one result per label).

Definition at line 427 of file `labeling/compute.hh`.

References `mln::geom::ncols()`, `mln::util::array< T >::resize()`, and `mln::util::array< T >::size()`.

## 9.98 mln::labeling::impl::generic Namespace Reference

Generic implementation namespace of labeling namespace.

### Functions

- `template<typename A , typename L > util::array< typename A::result > compute (const Accumulator< A > &a_, const Image< L > &label_, const typename L::value &nlabels)`  
*Generic implementation of `labeling::compute`.*
- `template<typename A , typename L > util::array< typename A::result > compute (util::array< A > &accus, const Image< L > &label_, const typename L::value &nlabels)`  
*Generic implementation of `labeling::compute`.*
- `template<typename A , typename I , typename L > util::array< typename A::result > compute (util::array< A > &accus, const Image< I > &input_, const Image< L > &label_, const typename L::value &nlabels)`  
*Generic implementation of `labeling::compute`.*
- `template<typename A , typename I , typename L > util::array< typename A::result > compute (const Accumulator< A > &a_, const Image< I > &input_, const Image< L > &label_, const typename L::value &nlabels)`  
*Generic implementation of `labeling::compute`.*

### 9.98.1 Detailed Description

Generic implementation namespace of labeling namespace.

## 9.98.2 Function Documentation

**9.98.2.1** `template<typename A , typename L > util::array<typename A ::result>  
mln::labeling::impl::generic::compute ( const Accumulator< A > & a_ , const Image<  
L > & label_ , const typename L::value & nlabels ) [inline]`

Generic implementation of [labeling::compute](#).

### Parameters

- [in] *a\_* An accumulator.
- [in] *label\_* The labeled image.
- [in] *nlabels* The number of labels in *label*.

### Returns

A [util::array](#) of accumulator result (one result per label).

Definition at line 204 of file `labeling/compute.hh`.

**9.98.2.2** `template<typename A , typename L > util::array<typename A ::result>  
mln::labeling::impl::generic::compute ( util::array< A > & accus , const Image< L >  
& label_ , const typename L::value & nlabels ) [inline]`

Generic implementation of [labeling::compute](#).

### Parameters

- [in] *accus\_* An array of accumulators. If the size is set to *nlabels* + 1, the accumulators are considered as initialized. Otherwise, the size is adjusted.
- [in] *label\_* The labeled image.
- [in] *nlabels* The number of labels in *label*.

### Returns

A [util::array](#) of accumulator result (one result per label).

Definition at line 241 of file `labeling/compute.hh`.

References `mln::util::array< T >::resize()`, and `mln::util::array< T >::size()`.

**9.98.2.3** `template<typename A , typename I , typename L > util::array<typename A ::result>  
mln::labeling::impl::generic::compute ( util::array< A > & accus , const Image< I > &  
input_ , const Image< L > & label_ , const typename L::value & nlabels ) [inline]`

Generic implementation of [labeling::compute](#).

### Parameters

- [in] *accus* An array of accumulators.
- [in] *input\_* The input image.
- [in] *label\_* The labeled image.

[in] *nlabels* The number of labels in `label`.

### Returns

A `util::array` of accumulator result (one result per label).

Definition at line 321 of file `labeling/compute.hh`.

References `mln::util::array< T >::resize()`, and `mln::util::array< T >::size()`.

**9.98.2.4** `template<typename A , typename I , typename L > util::array<typename A ::result>  
mln::labeling::impl::generic::compute ( const Accumulator< A > & a_, const Image< I  
> & input_, const Image< L > & label_, const typename L::value & nlabels )  
[inline]`

Generic implementation of `labeling::compute`.

### Parameters

[in] *a\_* An accumulator.

[in] *input\_* The input image.

[in] *label\_* The labeled image.

[in] *nlabels* The number of labels in `label`.

### Returns

A `util::array` of accumulator result (one result per label).

Definition at line 283 of file `labeling/compute.hh`.

## 9.99 mln::linear Namespace Reference

Namespace of linear image processing routines.

### Namespaces

- namespace `impl`  
*Namespace of linear image processing routines implementation details.*
- namespace `local`  
*Specializations of local linear routines.*

### Functions

- `template<typename I >  
mln::trait::concrete< I >::ret gaussian (const Image< I > &input, float sigma)`  
*Gaussian filter of an image `input`.*



- `template<typename I >`  
`mln::trait::concrete< I >::ret gaussian (const Image< I > &input, float sigma, int dir)`
- `template<typename I >`  
`mln::trait::concrete< I >::ret gaussian_1st_derivative (const Image< I > &input, float sigma)`
- `template<typename I >`  
`mln::trait::concrete< I >::ret gaussian_1st_derivative (const Image< I > &input, float sigma, int dir)`
- `template<typename I >`  
`mln::trait::concrete< I >::ret gaussian_2nd_derivative (const Image< I > &input, float sigma, int dir)`
- `template<typename I , typename W , unsigned Sh, unsigned Sv>`  
`mln_ch_convolve (I, W) convolve_2x1d(const Image< I > &input`
- `template<typename I , typename W >`  
`mln_ch_convolve (I, W) convolve(const Image< I > &input`
- `template<typename I , typename W , unsigned S>`  
`mln_ch_convolve (I, W) convolve_directional(const Image< I > &input`
- `template<typename I >`  
`mln_ch_convolve_grad (I, int) sobel_2d(const Image< I > &input)`  
*Compute the vertical component of the 2D Sobel gradient.*
  
- `template<typename I >`  
`mln_ch_convolve (I, int) sobel_2d_h(const Image< I > &input)`  
*Sobel\_2d gradient components.*

## 9.99.1 Detailed Description

Namespace of linear image processing routines.

## 9.99.2 Function Documentation

### 9.99.2.1 `template<typename I > mln::trait::concrete< I >::ret mln::linear::gaussian ( const Image< I > & input, float sigma ) [inline]`

Gaussian filter of an image `input`.

#### Precondition

`output.domain = input.domain`

Apply an approximated gaussian filter of `sigma` on `input`. This filter is applied in all the input image direction.

#### Precondition

`input.is_valid`

Definition at line 750 of file `gaussian.hh`.

References `mln::initialize()`.

Referenced by `mln::subsampling::gaussian_subsampling()`.

### 9.99.2.2 `template<typename I> mln::trait::concrete< I >::ret mln::linear::gaussian ( const Image< I > & input, float sigma, int dir ) [inline]`

Apply an approximated gaussian filter of `sigma` on `input`. on a specific direction `dir` if `dir = 0`, the filter is applied on the first image dimension. if `dir = 1`, the filter is applied on the second image dimension. And so on...

#### Precondition

```
input.is_valid
dir < dimension(input)
```

Definition at line 653 of file `gaussian.hh`.

References `mln::initialize()`.

### 9.99.2.3 `template<typename I> mln::trait::concrete< I >::ret mln::linear::gaussian_1st_derivative ( const Image< I > & input, float sigma ) [inline]`

Apply an approximated first derivative gaussian filter of `sigma` on `input` This filter is applied in all the input image direction.

#### Precondition

```
input.is_valid
```

Definition at line 779 of file `gaussian.hh`.

References `mln::initialize()`.

### 9.99.2.4 `template<typename I> mln::trait::concrete< I >::ret mln::linear::gaussian_1st_derivative ( const Image< I > & input, float sigma, int dir ) [inline]`

Apply an approximated first derivative gaussian filter of `sigma` on `input`. on a specific direction `dir` if `dir = 0`, the filter is applied on the first image dimension. if `dir = 1`, the filter is applied on the second image dimension. And so on...

#### Precondition

```
input.is_valid
dir < dimension(input)
```

Definition at line 685 of file `gaussian.hh`.

References `mln::initialize()`.

### 9.99.2.5 `template<typename I> mln::trait::concrete< I >::ret mln::linear::gaussian_2nd_derivative ( const Image< I > & input, float sigma, int dir ) [inline]`

Apply an approximated second derivative gaussian filter of `sigma` on `input`. on a specific direction `dir` if `dir = 0`, the filter is applied on the first image dimension. if `dir = 1`, the filter is applied on the second image dimension. And so on...

**Precondition**

input.is\_valid  
 dir < dimension(input)

Definition at line 718 of file gaussian.hh.

References mln::initialize().

**9.99.2.6 template<typename I > mln::linear::mln\_ch\_convolve ( I, int ) const [inline]**

Sobel\_2d gradient components.

Compute the L1 norm of the 2D Sobel gradient.

Compute the vertical component of the 2D Sobel gradient.

Compute the horizontal component of the 2D Sobel gradient.

Definition at line 142 of file sobel\_2d.hh.

References mln\_ch\_convolve(), mln\_ch\_convolve\_grad(), and mln::data::transform().

**9.99.2.7 template<typename I , typename W , unsigned Sh, unsigned Sv> mln::linear::mln\_ch\_convolve ( I, W ) const**

Convolution of an image `input` by two weighted line-shapes windows.

**Warning**

The weighted window is used as-is, considering that its symmetrization is handled by the client.

**Precondition**

input.is\_valid

**9.99.2.8 template<typename I , typename W > mln::linear::mln\_ch\_convolve ( I, W ) const**

Convolution of an image `input` by the weighted window `w_win`.

**Warning**

Computation of `output(p)` is performed with the value type of `output`.

The weighted window is used as-is, considering that its symmetrization is handled by the client.

**Precondition**

input.is\_valid

Referenced by mln\_ch\_convolve(), and mln\_ch\_convolve\_grad().

**9.99.2.9 template<typename I , typename W , unsigned S> mln::linear::mln\_ch\_convolve ( I, W ) const [inline]**

Convolution of an image `input` by a line-shaped (directional) weighted window defined by the array of `weights`.

**Warning**

Computation of `output(p)` is performed with the value type of `output`.  
The weighted window is used as-is, considering that its symmetrization is handled by the client.

**Precondition**

`input.is_valid`

**9.99.2.10** `template<typename I > mln::linear::mln_ch_convolve_grad ( I, int ) const`

Compute the vertical component of the 2D Sobel gradient.

Definition at line 124 of file `sobel_2d.hh`.

References `mln_ch_convolve()`, and `mln::data::transform()`.

Referenced by `mln_ch_convolve()`.

**9.100 mln::linear::impl Namespace Reference**

Namespace of linear image processing routines implementation details.

**9.100.1 Detailed Description**

Namespace of linear image processing routines implementation details.

**9.101 mln::linear::local Namespace Reference**

Specializations of local linear routines.

**Namespaces**

- namespace [impl](#)  
*Namespace of local linear routines implementation details.*

**Functions**

- `template<typename I, typename P, typename W, typename R >`  
`void convolve (const Image< I > &input, const Site< P > &p, const Weighted_Window< W >`  
`&w_win, R &result)`
- `template<typename P, typename W, typename R >`  
`void convolve (const Generalized_Pixel< P > &p, const Weighted_Window< W > &w_win, R`  
`&result)`

**9.101.1 Detailed Description**

Specializations of local linear routines.

## 9.101.2 Function Documentation

**9.101.2.1** `template<typename I , typename P , typename W , typename R > void  
mln::linear::local::convolve ( const Image< I > & input, const Site< P > & p, const  
Weighted_Window< W > & w_win, R & result ) [inline]`

Local convolution of image `input` at point `p` by the weighted window `w_win`.

### Warning

Computation of the `result` is performed with the type `R`.  
The weighted window is used as-is, considering that its symmetrization is handled by the client.

Definition at line 149 of file `linear/local/convolve.hh`.

Referenced by `convolve()`.

**9.101.2.2** `template<typename P , typename W , typename R > void mln::linear::local::convolve (   
const Generalized_Pixel< P > & p, const Weighted_Window< W > & w_win, R &  
result ) [inline]`

Local convolution around (generalized) pixel by the weighted window `w_win`.

### Warning

Computation of the `result` is performed with the type `R`.  
The weighted window is used as-is, considering that its symmetrization is handled by the client.

Definition at line 161 of file `linear/local/convolve.hh`.

References `convolve()`.

## 9.102 mln::linear::local::impl Namespace Reference

Namespace of local linear routines implementation details.

### 9.102.1 Detailed Description

Namespace of local linear routines implementation details.

## 9.103 mln::literal Namespace Reference

Namespace of literals.

### Classes

- struct `black_t`  
*Type of literal black.*

- struct [blue\\_t](#)  
*Type of literal blue.*
- struct [brown\\_t](#)  
*Type of literal brown.*
- struct [cyan\\_t](#)  
*Type of literal cyan.*
- struct [green\\_t](#)  
*Type of literal green.*
- struct [identity\\_t](#)  
*Type of literal identity.*
- struct [light\\_gray\\_t](#)  
*Type of literal grays.*
- struct [lime\\_t](#)  
*Type of literal lime.*
- struct [magenta\\_t](#)  
*Type of literal magenta.*
- struct [max\\_t](#)  
*Type of literal max.*
- struct [min\\_t](#)  
*Type of literal min.*
- struct [olive\\_t](#)  
*Type of literal olive.*
- struct [one\\_t](#)  
*Type of literal one.*
- struct [orange\\_t](#)  
*Type of literal orange.*
- struct [origin\\_t](#)  
*Type of literal origin.*
- struct [pink\\_t](#)  
*Type of literal pink.*
- struct [purple\\_t](#)  
*Type of literal purple.*
- struct [red\\_t](#)  
*Type of literal red.*

- struct [teal\\_t](#)  
*Type of literal teal.*
- struct [violet\\_t](#)  
*Type of literal violet.*
- struct [white\\_t](#)  
*Type of literal white.*
- struct [yellow\\_t](#)  
*Type of literal yellow.*
- struct [zero\\_t](#)  
*Type of literal zero.*

## Variables

- const [black\\_t](#) & [black](#) = [black\\_t\(\)](#)  
*Literal black.*
- const [blue\\_t](#) & [blue](#) = [blue\\_t\(\)](#)  
*Literal blue.*
- const [brown\\_t](#) & [brown](#) = [brown\\_t\(\)](#)  
*Literal brown.*
- const [cyan\\_t](#) & [cyan](#) = [cyan\\_t\(\)](#)  
*Literal cyan.*
- const [dark\\_gray\\_t](#) & [dark\\_gray](#) = [dark\\_gray\\_t\(\)](#)  
*Literal dark gray.*
- const [green\\_t](#) & [green](#) = [green\\_t\(\)](#)  
*Literal green.*
- const [identity\\_t](#) & [identity](#) = [identity\\_t\(\)](#)  
*Literal identity.*
- const [light\\_gray\\_t](#) & [light\\_gray](#) = [light\\_gray\\_t\(\)](#)  
*Literal light gray.*
- const [lime\\_t](#) & [lime](#) = [lime\\_t\(\)](#)  
*Literal lime.*
- const [magenta\\_t](#) & [magenta](#) = [magenta\\_t\(\)](#)  
*Literal magenta.*

- const `max_t` & `max` = `max_t()`  
*Literal max.*
- const `medium_gray_t` & `medium_gray` = `medium_gray_t()`  
*Literal medium\_gray.*
- const `min_t` & `min` = `min_t()`  
*Literal min.*
- const `olive_t` & `olive` = `olive_t()`  
*Literal olive.*
- const `one_t` & `one` = `one_t()`  
*Literal one.*
- const `orange_t` & `orange` = `orange_t()`  
*Literal orange.*
- const `origin_t` & `origin` = `origin_t()`  
*Literal origin.*
- const `pink_t` & `pink` = `pink_t()`  
*Literal pink.*
- const `purple_t` & `purple` = `purple_t()`  
*Literal purple.*
- const `red_t` & `red` = `red_t()`  
*Literal red.*
- const `teal_t` & `teal` = `teal_t()`  
*Literal teal.*
- const `violet_t` & `violet` = `violet_t()`  
*Literal violet.*
- const `white_t` & `white` = `white_t()`  
*Literal white.*
- const `yellow_t` & `yellow` = `yellow_t()`  
*Literal yellow.*
- const `zero_t` & `zero` = `zero_t()`  
*Literal zero.*

### 9.103.1 Detailed Description

Namespace of literals.



## 9.103.2 Variable Documentation

### 9.103.2.1 `const black_t & mln::literal::black = black_t()`

[Literal](#) black.

Definition at line 60 of file black.hh.

Referenced by `mln::labeling::colorize()`, and `mln::registration::icp()`.

### 9.103.2.2 `const blue_t & mln::literal::blue = blue_t()`

[Literal](#) blue.

Definition at line 162 of file colors.hh.

### 9.103.2.3 `const brown_t & mln::literal::brown = brown_t()`

[Literal](#) brown.

Definition at line 164 of file colors.hh.

### 9.103.2.4 `const cyan_t & mln::literal::cyan = cyan_t()`

[Literal](#) cyan.

Definition at line 178 of file colors.hh.

### 9.103.2.5 `const dark_gray_t & mln::literal::dark_gray = dark_gray_t()`

[Literal](#) dark gray.

Definition at line 70 of file grays.hh.

### 9.103.2.6 `const green_t & mln::literal::green = green_t()`

[Literal](#) green.

Definition at line 160 of file colors.hh.

Referenced by `mln::registration::icp()`, and `mln::make_debug_graph_image()`.

### 9.103.2.7 `const identity_t & mln::literal::identity = identity_t()`

[Literal](#) identity.

Definition at line 54 of file identity.hh.

### 9.103.2.8 `const light_gray_t & mln::literal::light_gray = light_gray_t()`

[Literal](#) light gray.

Definition at line 66 of file grays.hh.

**9.103.2.9** `const lime_t & mln::literal::lime = lime_t()`

[Literal](#) lime.

Definition at line 166 of file colors.hh.

**9.103.2.10** `const magenta_t & mln::literal::magenta = magenta_t()`

[Literal](#) magenta.

Definition at line 180 of file colors.hh.

**9.103.2.11** `const max_t & mln::literal::max = max_t()`

[Literal](#) max.

Definition at line 66 of file literal/max.hh.

**9.103.2.12** `const medium_gray_t & mln::literal::medium_gray = medium_gray_t()`

[Literal](#) medium\_gray.

Definition at line 68 of file grays.hh.

**9.103.2.13** `const min_t & mln::literal::min = min_t()`

[Literal](#) min.

Definition at line 66 of file literal/min.hh.

**9.103.2.14** `const olive_t & mln::literal::olive = olive_t()`

[Literal](#) olive.

Definition at line 184 of file colors.hh.

**9.103.2.15** `const one_t & mln::literal::one = one_t()`

[Literal](#) one.

Definition at line 69 of file one.hh.

Referenced by `mln::algebra::h_vec< d, C >::h_vec()`, `mln::operator++()`, and `mln::operator--()`.

**9.103.2.16** `const orange_t & mln::literal::orange = orange_t()`

[Literal](#) orange.

Definition at line 168 of file colors.hh.

**9.103.2.17 const origin\_t & mln::literal::origin = origin\_t()**

[Literal](#) origin.

Definition at line 55 of file origin.hh.

Referenced by mln::win::ball< G, C >::ball(), mln::geom::bbox(), mln::box< P >::box(), mln::geom::rotate(), and mln::make::w\_window().

**9.103.2.18 const pink\_t & mln::literal::pink = pink\_t()**

[Literal](#) pink.

Definition at line 170 of file colors.hh.

**9.103.2.19 const purple\_t & mln::literal::purple = purple\_t()**

[Literal](#) purple.

Definition at line 172 of file colors.hh.

**9.103.2.20 const red\_t & mln::literal::red = red\_t()**

[Literal](#) red.

Definition at line 158 of file colors.hh.

Referenced by mln::morpho::watershed::superpose(), and mln::debug::superpose().

**9.103.2.21 const teal\_t & mln::literal::teal = teal\_t()**

[Literal](#) teal.

Definition at line 174 of file colors.hh.

**9.103.2.22 const violet\_t & mln::literal::violet = violet\_t()**

[Literal](#) violet.

Definition at line 176 of file colors.hh.

**9.103.2.23 const white\_t & mln::literal::white = white\_t()**

[Literal](#) white.

Definition at line 60 of file white.hh.

Referenced by mln::registration::icp().

**9.103.2.24 const yellow\_t & mln::literal::yellow = yellow\_t()**

[Literal](#) yellow.

Definition at line 182 of file colors.hh.

### 9.103.2.25 `const zero_t & mln::literal::zero = zero_t()`

[Literal](#) zero.

Definition at line 69 of file zero.hh.

Referenced by `mln::transform::influence_zone_geodesic_saturated()`, `mln::accu::shape::volume< I >::init()`, `mln::accu::stat::variance< T, S, R >::init()`, `mln::morpho::attribute::sum< I, S >::init()`, `mln::accu::math::sum< T, S >::init()`, `mln::accu::rms< T, V >::init()`, `mln::accu::stat::histo3d_rgb< V >::init()`, `mln::accu::convolve< T1, T2, R >::init()`, `mln::accu::center< P, V >::init()`, `mln::window< D >::is_centered()`, `mln::accu::stat::variance< T, S, R >::mean()`, `mln::accu::stat::var< T >::mean()`, `mln::geom::mesh_corner_point_area()`, `mln::geom::mesh_curvature()`, `mln::geom::mesh_normal()`, `mln::morpho::meyer_wst()`, `mln::algebra::operator*()`, `mln::test::positive()`, `mln::make::relabelfun()`, `mln::geom::rotate()`, `mln::accu::shape::volume< I >::set_value()`, `mln::morpho::watershed::superpose()`, `mln::labeling::superpose()`, `mln::debug::superpose()`, `mln::accu::stat::var< T >::to_result()`, `mln::geom::translate()`, and `mln::make::w_window_directional()`.

## 9.104 `mln::logical` Namespace Reference

Namespace of logic.

### Namespaces

- namespace [impl](#)

*Implementation namespace of logical namespace.*

### Functions

- `template<typename L, typename R >`  
`void and\_inplace (Image< L > &lhs, const Image< R > &rhs)`
- `template<typename L, typename R >`  
`mln::trait::ch_value< L, typename mln::fun::vv2v::land\_not< typename L::value, typename R::value >::result >::ret and\_not (const Image< L > &lhs, const Image< R > &rhs)`
- `template<typename L, typename R >`  
`void and\_not\_inplace (Image< L > &lhs, const Image< R > &rhs)`
- `template<typename I >`  
`void not\_inplace (Image< I > &input)`
- `template<typename L, typename R >`  
`void or\_inplace (Image< L > &lhs, const Image< R > &rhs)`
- `template<typename L, typename R >`  
`void xor\_inplace (Image< L > &lhs, const Image< R > &rhs)`

### 9.104.1 Detailed Description

Namespace of logic.

## 9.104.2 Function Documentation

### 9.104.2.1 `template<typename L , typename R > void mln::logical::and_inplace ( Image< L > & lhs, const Image< R > & rhs ) [inline]`

Point-wise in-place "logical and" of image `rhs` in image `lhs`.

#### Parameters

[in, out] *lhs* First operand image.

[in] *rhs* Second operand image.

It performs:

for all `p` of `rhs.domain`

`lhs(p) = lhs(p) and rhs(p)`

#### Precondition

```
rhs.domain >= lhs.domain
```

Definition at line 91 of file `logical/and.hh`.

References `mln::data::transform_inplace()`.

### 9.104.2.2 `template<typename L , typename R > mln::trait::ch_value< L, typename mln::fun::vv2v::land_not< typename L::value, typename R::value >::result >::ret mln::logical::and_not ( const Image< L > & lhs, const Image< R > & rhs ) [inline]`

Point-wise "logical and-not" between images `lhs` and `rhs`.

#### Parameters

[in] *lhs* First operand image.

[in] *rhs* Second operand image.

#### Returns

The result image.

#### Precondition

```
lhs.domain == rhs.domain
```

Definition at line 76 of file `and_not.hh`.

References `mln::data::transform()`.

### 9.104.2.3 `template<typename L , typename R > void mln::logical::and_not_inplace ( Image< L > & lhs, const Image< R > & rhs ) [inline]`

Point-wise in-place "logical and-not" of image `rhs` in image `lhs`.

**Parameters**

[in, out] *lhs* First operand image.

[in] *rhs* Second operand image.

It performs:

for all  $p$  of  $rhs.domain$

$lhs(p) = lhs(p)$  and not  $rhs(p)$

**Precondition**

$rhs.domain \geq lhs.domain$

Definition at line 91 of file `and_not.hh`.

References `mln::data::transform_inplace()`.

#### 9.104.2.4 `template<typename I > void mln::logical::not_inplace ( Image< I > & input )` **[inline]**

Point-wise in-place "logical not" of image `input`.

**Parameters**

[in, out] *input* The target image.

It performs:

for all  $p$  of  $input.domain$

$input(p) = \text{not } input(p)$

**Precondition**

`input.is_valid`

Definition at line 88 of file `logical/not.hh`.

References `mln::data::transform_inplace()`.

#### 9.104.2.5 `template<typename L , typename R > void mln::logical::or_inplace ( Image< L > & lhs, const Image< R > & rhs )` **[inline]**

Point-wise in-place "logical or" of image `rhs` in image `lhs`.

**Parameters**

[in, out] *lhs* First operand image.

[in] *rhs* Second operand image.

It performs:

for all  $p$  of  $rhs.domain$

$lhs(p) = lhs(p)$  or  $rhs(p)$

**Precondition**

```
rhs.domain >= lhs.domain
```

Definition at line 91 of file logical/or.hh.

References mln::data::transform\_inplace().

**9.104.2.6 template<typename L , typename R > void mln::logical::xor\_inplace ( Image< L > & lhs, const Image< R > & rhs ) [inline]**

Point-wise in-place "logical xor" of image rhs in image lhs.

**Parameters**

[in, out] *lhs* First operand image.

[in] *rhs* Second operand image.

It performs:

for all p of rhs.domain

lhs(p) = lhs(p) xor rhs(p)

**Precondition**

```
rhs.domain >= lhs.domain
```

Definition at line 91 of file logical/xor.hh.

References mln::data::transform\_inplace().

**9.105 mln::logical::impl Namespace Reference**

Implementation namespace of logical namespace.

**Namespaces**

- namespace [generic](#)

*Generic implementation namespace of logical namespace.*

**9.105.1 Detailed Description**

Implementation namespace of logical namespace.

**9.106 mln::logical::impl::generic Namespace Reference**

Generic implementation namespace of logical namespace.

## 9.106.1 Detailed Description

Generic implementation namespace of logical namespace.

## 9.107 mln::make Namespace Reference

Namespace of routines that help to make Milena's objects.

### Functions

- `template<unsigned D, typename G, typename V >`  
`p_set< complex_psite< D, G > > attachment` (const `complex_psite< D, G >` &f, const `complex_image< D, G, V >` &ima)  
*Compute the attachment of the cell corresponding to the facet f to the image ima.*
- `mln::box1d box1d` (`def::coord` min\_ind, `def::coord` max\_ind)  
*Create an mln::box1d.*
- `mln::box1d box1d` (unsigned ninds)  
*Create an mln::box1d.*
- `mln::box2d box2d` (unsigned nrows, unsigned ncols)  
*Create an mln::box2d.*
- `mln::box2d box2d` (`def::coord` min\_row, `def::coord` min\_col, `def::coord` max\_row, `def::coord` max\_col)  
*Create an mln::box2d.*
- `mln::box2d_h box2d_h` (`def::coord` min\_row, `def::coord` min\_col, `def::coord` max\_row, `def::coord` max\_col)  
*Create an mln::box2d\_h.*
- `mln::box2d_h box2d_h` (unsigned nrows, unsigned ncols)  
*Create an mln::box2d\_h.*
- `mln::box3d box3d` (unsigned nslis, unsigned nrows, unsigned ncols)  
*Create an mln::box3d.*
- `mln::box3d box3d` (`def::coord` min\_sli, `def::coord` min\_row, `def::coord` min\_col, `def::coord` max\_sli, `def::coord` max\_row, `def::coord` max\_col)  
*Create an mln::box3d.*
- `template<unsigned D, typename G >`  
`p_set< complex_psite< D, G > > cell` (const `complex_psite< D, G >` &f)  
*Compute the set of faces of the cell corresponding to the facet f.*
- `template<typename T, typename U >`  
`util::couple< T, U > couple` (const T &val1, const T &val2)  
*Construct an mln::util::couple on-the-fly.*



- `template<unsigned D, typename G, typename V >`  
`p_set< complex_site< D, G > > detachment (const complex_site< D, G > &f, const complex_image< D, G, V > &ima)`  
*Compute the detachment of the cell corresponding to the facet f to the image ima.*
- `mln::dpoint2d_h dpoint2d_h (def::coord row, def::coord col)`  
*Create an mln::dpoint2d\_h.*
- `template<typename G, typename P >`  
`p_edges< G, pw::cst_< P > > dummy_p_edges (const Graph< G > &g_, const P &dummy_site)`  
*Create a p\_edges which associate a graph element to a constant site.*
- `template<typename G >`  
`p_edges< G > dummy_p_edges (const Graph< G > &g)`  
*Create a p\_edges which associate a graph element to a constant site.*
- `template<typename G, typename P >`  
`p_vertices< G, pw::cst_< P > > dummy_p_vertices (const Graph< G > &g_, const P &dummy_site)`  
*Create a p\_vertices which associate a graph element to a constant site.*
- `template<typename G >`  
`p_vertices< G > dummy_p_vertices (const Graph< G > &g)`  
*Create a p\_vertices which associate a graph element to a constant site.*
- `template<typename V, typename G >`  
`mln::edge_image< void, V, G > edge_image (const Graph< G > &g, const fun::i2v::array< V > &fv)`  
*Construct an edge image.*
- `template<typename FV, typename G >`  
`mln::edge_image< void, typename FV::result, G > edge_image (const Graph< G > &g, const Function_v2v< FV > &fv)`  
*Construct an edge image.*
- `template<typename FP, typename FV, typename G >`  
`mln::edge_image< typename FP::result, typename FV::result, G > edge_image (const Graph< G > &g_, const Function_v2v< FP > &fp, const Function_v2v< FV > &fv)`  
*Construct an edge image.*
- `template<typename P, typename V, typename G, typename FP, typename FV >`  
`mln::edge_image< typename FP::result, typename FV::result, G > edge_image (const mln::vertex_image< P, V, G > &v_ima_, const p_edges< G, FP > pe, const Function_vv2v< FV > &fv_)`  
*Construct an edge image.*
- `template<typename P, typename V, typename G, typename FV >`  
`mln::edge_image< void, typename FV::result, G > edge_image (const mln::vertex_image< P, V, G > &v_ima_, const Function_vv2v< FV > &fv_)`  
*Construct an edge image.*

- `template<typename P, typename V, typename G, typename F >`  
`mln::edge_image< void, bool, G > edge_image (const mln::vertex_image< P, V, G > &v_ima_,`  
`const Function_v2b< F > &fv_)`  
*Construct an edge image.*
- `template<typename T, unsigned N>`  
`algebra::h_mat< mlc_sqrt_int(N), T > h_mat (const T(&tab)[N])`  
*Create an `mln::algebra::mat<n,n,T>`.*
- `template<typename V, unsigned L>`  
`mln::image1d< V > image (V(&values)[L])`  
*Create an `image1d` from an 1D array of values.*
- `template<typename V, unsigned R, unsigned C>`  
`mln::image2d< V > image (V(&values)[R][C])`  
*Create an `image2d` from an 2D array of values.*
- `template<typename V, unsigned S, unsigned R, unsigned C>`  
`mln::image3d< V > image (V(&values)[S][R][C])`  
*Create an `image3d` from an 3D array of values.*
- `template<typename V, unsigned S>`  
`mln::image2d< V > image2d (V(&values)[S])`  
*Create an `image2d` from an 2D array of values.*
- `template<typename I >`  
`mln::image3d< typename I::value > image3d (const Image< I > &ima)`  
*Create an `image3d` from a 2D image.*
- `template<typename I >`  
`mln::image3d< typename I::value > image3d (const util::array< I > &ima)`  
*Create an `image3d` from an array of 2D images.*
- `template<typename I, typename N >`  
`util::graph_influence_zone_adjacency_graph (const Image< I > &iz_, const Neighborhood< N >`  
`&nbh, const typename I::value &nlabels)`  
*Create a graph from an influence zone image.*
- `template<unsigned n, unsigned m, typename T >`  
`algebra::mat< n, m, T > mat (const T(&tab)[n * m])`  
*Create an `mln::algebra::mat<n,m,T>`.*
- `template<typename T >`  
`util::ord_pair< T > ord_pair (const T &val1, const T &val2)`  
*Construct an `mln::util::ord_pair` on-the-fly.*
- `template<typename W, typename G >`  
`p_edges< G, fun::i2v::array< util::site_pair< typename W::site > > > p_edges_with_mass_centers`  
`(const Image< W > &wst_, const Graph< G > &g_)`  
*Construct a `p_edges` from a watershed image and a region adjacency graph (RAG).*

- `template<typename W , typename G >`  
`p_vertices< G, fun::i2v::array< typename W::site > > p_vertices_with_mass_centers` (const `Image< W >` &wst\_, const `Graph< G >` &g\_)  
*Create a `p_vertices` from a watershed image and a region adjacency graph (RAG).*
- `template<typename I >`  
`mln::util::pix< I > pix` (const `Image< I >` &ima, const typename `I::psite` &p)  
*Create an `mln::util::pix` from an image `ima` and a psite `p`.*
- `template<typename I >`  
`mln::pixel< I > pixel` (`Image< I >` &ima, const typename `I::psite` &p)  
*Create a `mln::pixel` from a mutable image `ima` and a point `p`.*
- `template<typename I >`  
`mln::pixel< const I > pixel` (const `Image< I >` &ima, const typename `I::psite` &p)  
*Create a `mln::pixel` from a constant image `ima` and a point `p`.*
- `mln::point2d_h point2d_h` (def::coord row, def::coord col)  
*Create an `mln::point2d_h`.*
- `template<typename I , typename N >`  
`util::couple< util::graph, typename mln::trait::concrete< I >::ret > rag_and_labeled_wsl` (const `Image< I >` &wshd\_, const `Neighborhood< N >` &nbh\_, const typename `I::value` &nbasins)  
*Create a region adjacency graph and a label image of the watershed line from a watershed image.*
- `template<typename I , typename N >`  
`util::graph region_adjacency_graph` (const `Image< I >` &wshd\_, const `Neighborhood< N >` &nbh, const typename `I::value` &nbasins)  
*Create a region adjacency graph from a watershed image.*
- `template<typename V , typename F >`  
`fun::i2v::array< V > relabelfun` (const `Function_v2b< F >` &fv2b, const `V` &nlabels, `V` &new\_nlabels)  
*Create a `i2v` function from a `v2b` function.*
- `template<typename V , typename F >`  
`fun::i2v::array< V > relabelfun` (const `Function_v2v< F >` &fv2v, const `V` &nlabels, `V` &new\_nlabels)  
*Create a `i2v` function from a `v2v` function.*
- `template<typename T >`  
`algebra::vec< 1, T > vec` (const `T` &v\_0)  
*Create an `mln::algebra::vec<n,T>`.*
- `template<typename T >`  
`algebra::vec< 4, T > vec` (const `T` &v\_0, const `T` &v\_1, const `T` &v\_2, const `T` &v\_3)  
*Create an `mln::algebra::vec<4,T>`.*
- `template<typename T >`  
`algebra::vec< 3, T > vec` (const `T` &v\_0, const `T` &v\_1, const `T` &v\_2)  
*Create an `mln::algebra::vec<3,T>`.*

- `template<typename T >`  
`algebra::vec< 2, T > vec (const T &v_0, const T &v_1)`  
*Create an `mln::algebra::vec<2,T>`.*
  
- `template<typename G , typename FV >`  
`mln::vertex\_image< void, typename FV::result, G > vertex\_image (const Graph< G > &g, const Function\_v2v< FV > &fv)`  
*Construct a vertex image.*
  
- `template<typename FP , typename FV , typename G >`  
`mln::vertex\_image< typename FP::result, typename FV::result, G > vertex\_image (const Graph< G > &g_, const Function\_v2v< FP > &fp, const Function\_v2v< FV > &fv)`  
*Construct a vertex image.*
  
- `template<typename I , typename N >`  
`p\_vertices< util::graph, fun::i2v::array< typename I::site > > voronoi (Image< I > &ima_, Image< I > &orig_, const Neighborhood< N > &nbh)`  
*Apply the Voronoi algorithm on `ima_` with the original image `orig_` for node computing with neighborhood `nbh`.*
  
- `template<typename W , typename F >`  
`mln::w\_window< typename W::dpsite, typename F::result > w\_window (const Window< W > &win, const Function\_v2v< F > &wei)`  
*Create a `mln::w_window` from a window and a weight function.*
  
- `template<typename W , unsigned M>`  
`mln::w\_window< mln::dpoint1d, W > w\_window1d (W(&weights)[M])`  
*Create a 1D `mln::w_window` from an array of weights.*
  
- `template<unsigned M>`  
`mln::w\_window1d\_int w\_window1d\_int (int(&weights)[M])`  
*Create a `mln::w_window1d_int`.*
  
- `template<typename W , unsigned S>`  
`mln::w\_window< mln::dpoint2d, W > w\_window2d (W(&weights)[S])`  
*Create a 2D `mln::w_window` from an array of weights.*
  
- `template<unsigned M>`  
`mln::w\_window2d\_int w\_window2d\_int (int(&weights)[M])`  
*Create a `mln::w_window2d_int`.*
  
- `template<typename W , unsigned M>`  
`mln::w\_window< mln::dpoint3d, W > w\_window3d (W(&weights)[M])`  
*Create a 3D `mln::w_window` from an array of weights.*
  
- `template<unsigned M>`  
`mln::w\_window3d\_int w\_window3d\_int (int(&weights)[M])`  
*Create a `mln::w_window3d_int`.*

- `template<typename D, typename W, unsigned L>`  
`mln::w_window< D, W > w_window_directional (const Gdpoint< D > &dp, W(&weights)[L])`  
*Create a directional centered weighted window.*

### 9.107.1 Detailed Description

Namespace of routines that help to make Milena's objects.

### 9.107.2 Function Documentation

- 9.107.2.1** `template<unsigned D, typename G, typename V > p_set< complex_psite< D, G > >`  
`mln::make::attachment ( const complex_psite< D, G > & f, const complex_image< D,`  
`G, V > & ima ) [inline]`

Compute the attachment of the cell corresponding to the facet  $f$  to the image  $ima$ .

#### Precondition

$f$  is a facet (it does not belong to any face of higher dimension).  
 $ima$  is an image of Boolean values.

#### Returns

a set of faces containing the attachment.

We do not use the formal definition of the attachment here (see `coupric.08.pami`). We use the following (equivalent) definition: an  $N$ -face  $F$  in  $CELL$  is in the attachment of  $CELL$  to  $IMA$  if it is adjacent to at least an  $(N-1)$ -face or an  $(N+1)$ -face that does not belong to  $CELL$ .

Definition at line 68 of file `attachment.hh`.

References `cell()`, and `mln::topo::is_facet()`.

Referenced by `mln::topo::is_simple_cell< I >::operator()`.

- 9.107.2.2** `mln::box1d mln::make::box1d ( def::coord min_ind, def::coord max_ind )`  
`[inline]`

Create an `mln::box1d`.

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

#### Parameters

[in] *min\_ind* Minimum index.

[in] *max\_ind* Maximum index.

#### Precondition

`max_ind >= min_ind`.

#### Returns

A 1D box.

Definition at line 79 of file `make/box1d.hh`.

References `box1d()`.

### 9.107.2.3 `mln::box1d mln::make::box1d ( unsigned ninds ) [inline]`

Create an `mln::box1d`.

#### Parameters

[in] *ninds* Number of indices.

#### Precondition

`ninds != 0` and `ncols != 0`.

#### Returns

A 1D box.

Definition at line 70 of file `make/box1d.hh`.

Referenced by `box1d()`, and `mln::image1d< T >::image1d()`.

### 9.107.2.4 `mln::box2d mln::make::box2d ( unsigned nrows, unsigned ncols ) [inline]`

Create an `mln::box2d`.

#### Parameters

[in] *nrows* Number of rows.

[in] *ncols* Number of columns.

#### Precondition

`nrows != 0` and `ncols != 0`.

#### Returns

A 2D box.

Definition at line 78 of file `make/box2d.hh`.

Referenced by `mln::image2d< T >::image2d()`, and `mln::io::pnm::load()`.

### 9.107.2.5 `mln::box2d mln::make::box2d ( def::coord min_row, def::coord min_col, def::coord max_row, def::coord max_col ) [inline]`

Create an `mln::box2d`.

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

#### Parameters

[in] *min\_row* Index of the top most row.

- [in] *min\_col* Index of the left most column.
- [in] *max\_row* Index of the botton most row.
- [in] *max\_col* Index of the right most column.

**Precondition**

`max_row >= min_row and max_col >= min_col.`

**Returns**

A 2D box.

Definition at line 88 of file make/box2d.hh.

**9.107.2.6 mln::box2d\_h mln::make::box2d\_h ( def::coord *min\_row*, def::coord *min\_col*, def::coord *max\_row*, def::coord *max\_col* ) [inline]**

Create an [mln::box2d\\_h](#).

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**Parameters**

- [in] *min\_row* Index of the top most row.
- [in] *min\_col* Index of the left most column.
- [in] *max\_row* Index of the botton most row.
- [in] *max\_col* Index of the right most column.

**Precondition**

`max_row >= min_row and max_col >= min_col.`

**Returns**

A 2D\_H box.

Definition at line 85 of file make/box2d\_h.hh.

References [point2d\\_h\(\)](#).

**9.107.2.7 mln::box2d\_h mln::make::box2d\_h ( unsigned *nrows*, unsigned *ncols* ) [inline]**

Create an [mln::box2d\\_h](#).

**Parameters**

- [in] *nrows* Number of rows.
- [in] *ncols* Number of columns.

**Precondition**

`nrows != 0 and ncols != 0.`

**Returns**

A 2D\_H box.

Definition at line 75 of file make/box2d\_h.hh.

References [point2d\\_h\(\)](#).

### 9.107.2.8 `mln::box3d mln::make::box3d ( unsigned nslis, unsigned nrows, unsigned ncols ) [inline]`

Create an [mln::box3d](#).

**Parameters**

- [in] *nslis* Number of slices.
- [in] *nrows* Number of rows.
- [in] *ncols* Number of columns.

**Precondition**

`ninds != 0 and ncols != 0 and nslis != 0.`

**Returns**

A 3D box.

Definition at line 80 of file make/box3d.hh.

Referenced by [image3d\(\)](#), and [mln::image3d< T >::image3d\(\)](#).

### 9.107.2.9 `mln::box3d mln::make::box3d ( def::coord min_sli, def::coord min_row, def::coord min_col, def::coord max_sli, def::coord max_row, def::coord max_col ) [inline]`

Create an [mln::box3d](#).

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**Parameters**

- [in] *min\_sli* Index of the lowest slice.
- [in] *min\_row* Index of the top most row.
- [in] *min\_col* Index of the left most column.
- [in] *max\_sli* Index of the highest slice.
- [in] *max\_row* Index of the botton most row.
- [in] *max\_col* Index of the right most column.

**Precondition**

```
max_sli >= min_sli.
max_row >= min_row.
max_col >= min_col.
```

**Returns**

A 3D box.

Definition at line 92 of file make/box3d.hh.



**9.107.2.10** `template<unsigned D, typename G > p_set< complex_psite< D, G > >  
mln::make::cell ( const complex_psite< D, G > & f ) [inline]`

Compute the set of faces of the cell corresponding to the facet *f*.

#### Precondition

*f* is a facet (it does not belong to any face of higher dimension).

#### Returns

An [mln::p\\_set](#) of sites (faces) containing the attachment.

Definition at line 63 of file cell.hh.

References [mln::topo::is\\_facet\(\)](#), and [mln::complex\\_psite< D, G >::n\(\)](#).

Referenced by [attachment\(\)](#), and [detachment\(\)](#).

**9.107.2.11** `template<typename T, typename U > util::couple<T,U> mln::make::couple ( const  
T & val1, const T & val2 )`

Construct an [mln::util::couple](#) on-the-fly.

Referenced by [mln::labeling::blobs\\_and\\_compute\(\)](#), [mln::transform::distance\\_and\\_closest\\_point\\_geodesic\(\)](#), [mln::transform::distance\\_and\\_influence\\_zone\\_geodesic\(\)](#), and [mln::labeling::value\\_and\\_compute\(\)](#).

**9.107.2.12** `template<unsigned D, typename G, typename V > p_set< complex_psite< D, G > >  
mln::make::detachment ( const complex_psite< D, G > & f, const complex_image<  
D, G, V > & ima ) [inline]`

Compute the detachment of the cell corresponding to the facet *f* to the image *ima*.

#### Precondition

*f* is a facet (it does not belong to any face of higher dimension).  
*ima* is an image of Boolean values.

#### Returns

a set of faces containing the detachment.

We do not use the formal definition of the detachment here (see [coupric.08.pami](#)). We use the following (equivalent) definition: an N-face F in CELL is not in the detachment of CELL from IMA if it is adjacent to at least an (N-1)-face or an (N+1)-face that does not belong to CELL.

Definition at line 68 of file detachment.hh.

References [cell\(\)](#), and [mln::topo::is\\_facet\(\)](#).

Referenced by [mln::topo::detach\(\)](#).

**9.107.2.13** `mln::dpoint2d_h mln::make::dpoint2d_h ( def::coord row, def::coord col )  
[inline]`

Create an [mln::dpoint2d\\_h](#).

**Parameters**

- [in] *row* Row coordinate.
- [in] *col* Column coordinate.

**Returns**

A 2D dpoint.

Definition at line 55 of file make/dpoint2d\_h.hh.

**9.107.2.14** `template<typename G , typename P > p_edges< G, pw::cst_< P > >  
mln::make::dummy_p_edges ( const Graph< G > & g_, const P & dummy_site )`

Create a [p\\_edges](#) which associate a graph element to a constant site.

**Parameters**

- [in] *g\_* A graph.
- [in] *dummy\_site* The dummy site mapped to graph edges.

**Returns**

A [p\\_edges](#).

Definition at line 77 of file dummy\_p\_edges.hh.

**9.107.2.15** `template<typename G > p_edges< G > mln::make::dummy_p_edges ( const Graph<  
G > & g )`

Create a [p\\_edges](#) which associate a graph element to a constant site.

0 (int) is used as dummy site.

**Parameters**

- [in] *g* A graph.

**Returns**

A [p\\_edges](#).

Definition at line 93 of file dummy\_p\_edges.hh.

**9.107.2.16** `template<typename G , typename P > p_vertices< G, pw::cst_< P > >  
mln::make::dummy_p_vertices ( const Graph< G > & g_, const P & dummy_site )`

Create a [p\\_vertices](#) which associate a graph element to a constant site.

**Parameters**

- [in] *g\_* A graph.
- [in] *dummy\_site* The dummy site mapped to graph vertices.

**Returns**

A [p\\_vertices](#).

Definition at line 77 of file dummy\_p\_vertices.hh.

**9.107.2.17** `template<typename G > p_vertices< G > mln::make::dummy_p_vertices ( const Graph< G > & g )`

Create a [p\\_vertices](#) which associate a graph element to a constant site.

0 (int) is used as dummy site.

**Parameters**

[in] *g* A graph.

**Returns**

A [p\\_vertices](#).

Definition at line 93 of file dummy\_p\_vertices.hh.

**9.107.2.18** `template<typename V , typename G > mln::edge_image< void, V, G > mln::make::edge_image ( const Graph< G > & g, const fun::i2v::array< V > & fv ) [inline]`

Construct an edge image.

**Parameters**

[in] *g* A graph.

[in] *fv* A function mapping edge ids to values.

**Returns**

an edge image.

Definition at line 141 of file make/edge\_image.hh.

**9.107.2.19** `template<typename FV , typename G > mln::edge_image< void, typename FV::result, G > mln::make::edge_image ( const Graph< G > & g, const Function_v2v< FV > & fv )`

Construct an edge image.

**Parameters**

[in] *g* A graph.

[in] *fv* A function mapping edge ids to values.

**Returns**

an edge image.

Definition at line 155 of file make/edge\_image.hh.

**9.107.2.20** `template<typename FP , typename FV , typename G > mln::edge_image< typename FP::result, typename FV::result, G > mln::make::edge_image ( const Graph< G > & g_, const Function_v2v< FP > & fp, const Function_v2v< FV > & fv ) [inline]`

Construct an edge image.

#### Parameters

- [in] *g\_* A graph.
- [in] *fp* A function mapping edge ids to sites.
- [in] *fv* A function mapping edge ids to values.

#### Returns

an edge image.

Definition at line 179 of file make/edge\_image.hh.

**9.107.2.21** `template<typename P , typename V , typename G , typename FP , typename FV > mln::edge_image< typename FP::result, typename FV::result, G > mln::make::edge_image ( const mln::vertex_image< P, V, G > & v_ima_, const p_edges< G, FP > pe, const Function_vv2v< FV > & fv_ ) [inline]`

Construct an edge image.

#### Parameters

- [in] *v\_ima\_* A vertex image.
- [in] *pe* A [p\\_edges](#) mapping graph elements to sites.
- [in] *fv\_* A function mapping two vertex ids to a value. The result is associated to the corresponding edge.

#### Returns

an edge image.

Definition at line 199 of file make/edge\_image.hh.

**9.107.2.22** `template<typename P , typename V , typename G , typename FV > mln::edge_image< void, typename FV::result, G > mln::make::edge_image ( const mln::vertex_image< P, V, G > & v_ima_, const Function_vv2v< FV > & fv_ ) [inline]`

Construct an edge image.

#### Parameters

- [in] *v\_ima\_* A vertex image.
- [in] *fv\_* A function mapping two vertices' values to a value. The result is associated to the corresponding edge.

#### Returns

an edge image without localization information mapped to graph elements.

Definition at line 225 of file make/edge\_image.hh.

**9.107.2.23** `template<typename P , typename V , typename G , typename F > mln::edge_image< void, bool, G > mln::make::edge_image ( const mln::vertex_image< P, V, G > & v_ima_ , const Function_v2b< F > & fv_ ) [inline]`

Construct an edge image.

#### Parameters

[in] *v\_ima\_* A vertex image.

[in] *fv\_* A predicate on a vertex's value. The (Boolean) result is associated to the edges adjacent to the vertex.

#### Returns

an edge image without localization information mapped to graph elements.

Definition at line 250 of file make/edge\_image.hh.

References mln::data::fill().

**9.107.2.24** `template<typename T , unsigned N> algebra::h_mat< mlc_sqrt_int(N), T > mln::make::h_mat ( const T(&) tab[N] ) [inline]`

Create an mln::algebra::mat<n,n,T>.

Definition at line 60 of file make/h\_mat.hh.

Referenced by mln::fun::x2x::rotation< n, C >::rotation().

**9.107.2.25** `template<typename V , unsigned L> mln::image1d< V > mln::make::image ( V(&) values[L] )`

Create an [image1d](#) from an 1D array of values.

#### Parameters

[in] *values* 1D array.

#### Returns

A 1D image.

Definition at line 85 of file make/image.hh.

**9.107.2.26** `template<typename V , unsigned R, unsigned C> mln::image2d< V > mln::make::image ( V(&) values[R][C] )`

Create an [image2d](#) from an 2D array of values.

#### Parameters

[in] *values* 2D array.

#### Returns

A 2D image.

Definition at line 97 of file make/image.hh.

References `mln::opt::at()`.

**9.107.2.27** `template<typename V , unsigned S, unsigned R, unsigned C> mln::image3d< V > mln::make::image ( V(&) values[S][R][C] )`

Create an [image3d](#) from an 3D array of values.

#### Parameters

[in] *values* 3D array.

#### Returns

A 3D image.

Definition at line 112 of file make/image.hh.

References `mln::opt::at()`.

**9.107.2.28** `template<typename V , unsigned S> mln::image2d< V > mln::make::image2d ( V(&) values[S] )`

Create an [image2d](#) from an 2D array of values.

#### Parameters

[in] *values* 2D array.

#### Returns

A 2D image.

Definition at line 61 of file make/image2d.hh.

**9.107.2.29** `template<typename I > mln::image3d< typename I::value > mln::make::image3d ( const Image< I > & ima ) [inline]`

Create an [image3d](#) from a 2D image.

Definition at line 92 of file make/image3d.hh.

References `box3d()`, and `mln::data::paste()`.

**9.107.2.30** `template<typename I > mln::image3d< typename I::value > mln::make::image3d ( const util::array< I > & ima ) [inline]`

Create an [image3d](#) from an array of 2D images.

Definition at line 70 of file make/image3d.hh.

References `box3d()`, `mln::util::array< T >::is_empty()`, `mln::util::array< T >::nelements()`, `mln::data::paste()`, `mln::box< P >::pmax()`, and `mln::box< P >::pmin()`.

Referenced by `mln::io::pnms::load()`.

**9.107.2.31** `template<typename I , typename N > util::graph mln::make::influence_zone_adjacency_graph ( const Image< I > & iz_, const Neighborhood< N > & nbh, const typename I::value & nlabels ) [inline]`

Create a graph from an influence zone image.

#### Parameters

- [in] *iz* influence zone image.
- [in] *nbh* A neighborhood.
- [in] *nlabels* number of influence zone in *iz*.

#### Returns

[util::graph Graph](#) based on the adjacency of the influence zones.

Definition at line 175 of file `influence_zone_adjacency_graph.hh`.

**9.107.2.32** `template<unsigned n, unsigned m, typename T > algebra::mat< n, m, T > mln::make::mat ( const T(&) tab[n *m] ) [inline]`

Create an `mln::algebra::mat<n,m,T>`.

#### Parameters

- [in] *tab* Array of values.

#### Precondition

The array dimension has to be  $n * m$ .

Definition at line 61 of file `make/mat.hh`.

**9.107.2.33** `template<typename T > util::ord_pair< T > mln::make::ord_pair ( const T & val1, const T & val2 ) [inline]`

Construct an [mln::util::ord\\_pair](#) on-the-fly.

Definition at line 277 of file `ord_pair.hh`.

**9.107.2.34** `template<typename W , typename G > p_edges< G, fun::i2v::array< util::site_pair< typename W::site > > > mln::make::p_edges_with_mass_centers ( const Image< W > & wst_, const Graph< G > & g_ ) [inline]`

Construct a [p\\_edges](#) from a watershed image and a region adjacency graph (RAG).

Map each graph edge to a pair of mass centers of two adjacent regions.

#### Parameters

- wst\_* A watershed image.
- g\_* A region adjacency graph.

**Returns**

A [p\\_edges](#).

**See also**

[edge\\_image](#), [p\\_edges](#), [make::region\\_adjacency\\_graph](#)

Definition at line 81 of file `p_edges_with_mass_centers.hh`.

References `mln::labeling::compute()`.

**9.107.2.35** `template<typename W , typename G > p_vertices< G, fun::i2v::array< typename W::site > > mln::make::p_vertices_with_mass_centers ( const Image< W > & wst_, const Graph< G > & g_ ) [inline]`

Construct a [p\\_vertices](#) from a watershed image and a region adjacency graph (RAG).

Map each graph vertex to the mass center of its corresponding region.

**Parameters**

*wst\_* A watershed image.

*g\_* A region adjacency graph.

**Returns**

A [p\\_vertices](#).

**See also**

[edge\\_image](#), [vertex\\_image](#), [p\\_vertices](#), [p\\_edges](#), [make::region\\_adjacency\\_graph](#)

Definition at line 77 of file `p_vertices_with_mass_centers.hh`.

References `mln::labeling::compute()`.

**9.107.2.36** `template<typename I > mln::util::pix< I > mln::make::pix ( const Image< I > & ima, const typename I::psite & p ) [inline]`

Create an [mln::util::pix](#) from an image `ima` and a psite `p`.

**Parameters**

[in] *ima* The input image.

[in] *p* The point site.

**Returns**

An [mln::util::pix](#).

Definition at line 58 of file `make/pix.hh`.

**9.107.2.37** `template<typename I > mln::pixel< I > mln::make::pixel ( Image< I > & ima, const typename I::psite & p ) [inline]`

Create a [mln::pixel](#) from a mutable image `ima` and a point `p`.

Definition at line 63 of file `make/pixel.hh`.



### 9.107.2.38 `template<typename I> mln::pixel< const I> mln::make::pixel ( const Image< I> & ima, const typename I::psite & p ) [inline]`

Create a [mln::pixel](#) from a constant image `ima` and a point `p`.

Definition at line 55 of file `make/pixel.hh`.

### 9.107.2.39 `mln::point2d_h mln::make::point2d_h ( def::coord row, def::coord col ) [inline]`

Create an [mln::point2d\\_h](#).

#### Parameters

- [in] `row` Row coordinate.
- [in] `col` Column coordinate.

#### Returns

A 2D point.

Definition at line 55 of file `make/point2d_h.hh`.

Referenced by `box2d_h()`.

### 9.107.2.40 `template<typename I, typename N> util::couple< util::graph, typename mln::trait::concrete< I>::ret> mln::make::rag_and_labeled_wsl ( const Image< I> & wshd_, const Neighborhood< N> & nbh_, const typename I::value & nbasins ) [inline]`

Create a region adjacency graph and a label image of the watershed line from a watershed image.

#### Parameters

- [in] `wshd_` Watershed image.
- [in] `nbh_` [Neighborhood](#)
- [in] `nbasins` Number of influence zone in `wshd`.

#### Returns

A couple. First element is the graph, second element is an image with a labeled watershed line.

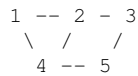
```

|-----|           |-----|
| 1 1 1 0 2 2 0 3 |           | . . . 1 . . 2 . |
| 1 1 0 2 2 2 0 3 |           | . . 1 . . . 2 . |
| 1 0 4 0 2 0 3 3 |           | . 1 . 3 . 4 . . |
| 0 4 4 4 0 5 0 3 |           | 1 . . . 5 . 6 . |
|-----|           |-----|

```

Watershed image                      Labeled watershed line  
(watershed line labeled with 0)

|  
|  
|  
v



Region Adjacency graph (RAG)

Definition at line 229 of file rag\_and\_labeled\_wsl.hh.

**9.107.2.41** `template<typename I, typename N > util::graph mln::make::region_adjacency_graph ( const Image< I > & wshd_, const Neighborhood< N > & nbh, const typename I::value & nbasins ) [inline]`

Create a region adjacency graph from a watershed image.

#### Parameters

- [in] *wshd\_* watershed image.
- [in] *nbh* A neighborhood.
- [in] *nbasins* number of influence zone in wshd.

#### Returns

[util::graph Graph](#) based on the adjacency of the influence zones.

Definition at line 179 of file region\_adjacency\_graph.hh.

**9.107.2.42** `template<typename V, typename F > fun::i2v::array< V > mln::make::relabelfun ( const Function_v2b< F > & fv2b, const V & nlabels, V & new_nlabels ) [inline]`

Create a i2v function from a v2b function.

This function can be used to relabel a labeled image.

#### Parameters

- [in] *fv2b* A v2b function.
- [in] *nlabels* The number of labels.
- [in] *new\_nlabels* The number of labels after relabeling.

#### Returns

a i2v function.

#### See also

[mln::labeling::relabel](#)

Definition at line 83 of file relabelfun.hh.

References `mln::literal::zero`.

Referenced by `mln::labeling::pack()`, `mln::labeling::pack_inplace()`, `mln::labeling::relabel()`, `mln::labeled_image_base< I, E >::relabel()`, and `mln::labeling::relabel_inplace()`.

**9.107.2.43** `template<typename V , typename F > fun::i2v::array< V > mln::make::relabelfun ( const Function_v2v< F > & fv2v, const V & nlabels, V & new_nlabels ) [inline]`

Create a i2v function from a v2v function.

This function can be used to relabel a labeled image.

#### Parameters

- [in] *fv2v* A v2v function. This function maps an id to an already existing one.
- [in] *nlabels* The number of labels.
- [in] *new\_nlabels* The number of labels after relabeling.

#### Returns

a i2v function.

#### See also

[mln::labeling::relabel](#)

Definition at line 106 of file relabelfun.hh.

References mln::literal::zero.

**9.107.2.44** `template<typename T > algebra::vec< 1, T > mln::make::vec ( const T & v_0 ) [inline]`

Create an mln::algebra::vec<n,T>.

#### Parameters

- [in] *v\_0* First coordinate.

#### Returns

A 1D vector.

Definition at line 91 of file make/vec.hh.

**9.107.2.45** `template<typename T > algebra::vec< 4, T > mln::make::vec ( const T & v_0, const T & v_1, const T & v_2, const T & v_3 ) [inline]`

Create an mln::algebra::vec<4,T>.

#### Parameters

- [in] *v\_0* First coordinate.
- [in] *v\_1* Second coordinate.
- [in] *v\_2* Third coordinate.
- [in] *v\_3* Fourth coordinate.

#### Returns

A 4D vector.

Definition at line 121 of file make/vec.hh.

**9.107.2.46** `template<typename T > algebra::vec< 3, T > mln::make::vec ( const T & v_0, const T & v_1, const T & v_2 ) [inline]`

Create an `mln::algebra::vec<3,T>`.

#### Parameters

[in] `v_0` First coordinate.

[in] `v_1` Second coordinate.

[in] `v_2` Third coordinate.

#### Returns

A 3D vector.

Definition at line 110 of file `make/vec.hh`.

**9.107.2.47** `template<typename T > algebra::vec< 2, T > mln::make::vec ( const T & v_0, const T & v_1 ) [inline]`

Create an `mln::algebra::vec<2,T>`.

#### Parameters

[in] `v_0` First coordinate.

[in] `v_1` Second coordinate.

#### Returns

A 2D vector.

Definition at line 100 of file `make/vec.hh`.

**9.107.2.48** `template<typename G , typename FV > mln::vertex_image< void, typename FV::result, G > mln::make::vertex_image ( const Graph< G > & g, const Function_v2v< FV > & fv )`

Construct a vertex image.

#### Parameters

[in] `g` A graph.

[in] `fv` A function mapping vertex ids to values.

#### Returns

A vertex image.

Definition at line 77 of file `make/vertex_image.hh`.

**9.107.2.49** `template<typename FP , typename FV , typename G > mln::vertex_image< typename FP::result, typename FV::result, G > mln::make::vertex_image ( const Graph< G > & g_, const Function_v2v< FP > & fp, const Function_v2v< FV > & fv )`

Construct a vertex image.

#### Parameters

- [in] *g\_* A graph.
- [in] *fp* A function mapping vertex ids to sites.
- [in] *fv* A function mapping vertex ids to values.

#### Returns

A vertex image.

Definition at line 92 of file make/vertex\_image.hh.

**9.107.2.50** `template<typename I , typename N > p_vertices< util::graph, fun::i2v::array< typename I::site > > mln::make::voronoi ( Image< I > & ima_, Image< I > & orig_, const Neighborhood< N > & nbh ) [inline]`

Apply the Voronoi algorithm on *ima\_* with the original image *orig\_* for node computing with neighborhood *nbh*.

#### Parameters

- [in] *ima\_* The labeling image.
- [in] *orig\_* The original image.
- [in] *nbh* The neighborhood for computing algorithm.

#### Returns

The computed graph.

Definition at line 68 of file voronoi.hh.

References `mln::util::graph::add_edge()`, `mln::util::graph::add_vertex()`, and `mln::estim::min_max()`.

**9.107.2.51** `template<typename W , typename F > mln::w_window< typename W::dpsite, typename F::result > mln::make::w_window ( const Window< W > & win, const Function_v2v< F > & wei ) [inline]`

Create a [mln::w\\_window](#) from a window and a weight function.

#### Parameters

- [in] *win* A simple window.
- [in] *wei* A weight function.

#### Returns

A weighted window.

Definition at line 63 of file make/w\_window.hh.

References `mln::w_window< D, W >::insert()`, and `mln::literal::origin`.

**9.107.2.52** `template<typename W , unsigned M> mln::w_window< mln::dpoint1d, W >  
mln::make::w_window1d ( W(&) weights[M] ) [inline]`

Create a 1D [mln::w\\_window](#) from an array of weights.

#### Parameters

[in] *weights* Array.

#### Precondition

The array size, M, has to be a square of an odd integer.

#### Returns

A 1D weighted window.

Definition at line 63 of file `w_window1d.hh`.

References `mln::w_window< D, W >::insert()`.

Referenced by `w_window1d_int()`.

**9.107.2.53** `template<unsigned M> mln::w_window1d_int mln::make::w_window1d_int ( int(&)  
weights[M] ) [inline]`

Create a [mln::w\\_window1d\\_int](#).

#### Parameters

[in] *weights* Array of integers.

#### Precondition

The array size, M, has to be a square of an odd integer.

#### Returns

A 1D int-weighted window.

Definition at line 61 of file `make/w_window1d_int.hh`.

References `w_window1d()`.

**9.107.2.54** `template<typename W , unsigned S> mln::w_window< mln::dpoint2d, W >  
mln::make::w_window2d ( W(&) weights[S] ) [inline]`

Create a 2D [mln::w\\_window](#) from an array of weights.

#### Parameters

[in] *weights* Array.

#### Precondition

The array size, S, has to be a square of an odd integer.

**Returns**

A 2D weighted window.

Definition at line 62 of file w\_window2d.hh.

Referenced by w\_window2d\_int().

**9.107.2.55** `template<unsigned M> mln::w_window2d_int mln::make::w_window2d_int ( int(&)  
weights[M] ) [inline]`

Create a [mln::w\\_window2d\\_int](#).

**Parameters**

[in] *weights* Array of integers.

**Precondition**

The array size, M, has to be a square of an odd integer.

**Returns**

A 2D int-weighted window.

Definition at line 61 of file make/w\_window2d\_int.hh.

References w\_window2d().

**9.107.2.56** `template<typename W , unsigned M> mln::w_window< mln::dpoint3d, W >  
mln::make::w_window3d ( W(&) weights[M] ) [inline]`

Create a 3D [mln::w\\_window](#) from an array of weights.

**Parameters**

[in] *weights* Array.

**Precondition**

The array size, M, has to be a cube of an odd integer.

**Returns**

A 3D weighted window.

Definition at line 64 of file w\_window3d.hh.

References mln::w\_window< D, W >::insert().

Referenced by w\_window3d\_int().

**9.107.2.57** `template<unsigned M> mln::w_window3d_int mln::make::w_window3d_int ( int(&)  
weights[M] ) [inline]`

Create a [mln::w\\_window3d\\_int](#).

**Parameters**

[in] *weights* Array of integers.

**Precondition**

The array size, M, has to be a cube of an odd integer.

**Returns**

A 3D int-weighted window.

Definition at line 61 of file `make/w_window3d_int.hh`.

References `w_window3d()`.

**9.107.2.58** `template<typename D , typename W , unsigned L> mln::w_window< D, W >`  
`mln::make::w_window_directional ( const Gdpoint< D > & dp, W(&) weights[L] )`  
**[inline]**

Create a directional centered weighted window.

**Parameters**

[in] *dp* A delta-point to set the orientation.

[in] *weights* An array of weights.

**Returns**

A weighted window.

The window length `L` has to be odd.

Definition at line 61 of file `w_window_directional.hh`.

References `mln::w_window< D, W >::insert()`, and `mln::literal::zero`.

## 9.108 mln::math Namespace Reference

Namespace of mathematical routines.

**Functions**

- `template<typename T >`  
`T abs (const T &v)`  
*Generic version.*
- `template<unsigned n>`  
`value::int_u< n > abs (const value::int_u< n > &v)`  
*Specialization for `mln::value::int_u`.*
- `int abs (int v)`  
*Specializations for existing overloads of `std::abs`.*



## 9.108.1 Detailed Description

Namespace of mathematical routines.

## 9.108.2 Function Documentation

### 9.108.2.1 `template<typename T > T mln::math::abs ( const T & v ) [inline]`

Generic version.

Definition at line 74 of file math/abs.hh.

Referenced by `abs()`, and `mln::morpho::line_gradient()`.

### 9.108.2.2 `int mln::math::abs ( int v ) [inline]`

Specializations for existing overloads of `std::abs`.

Reference: ISO/IEC 14882:2003 C++ standard, section 26.5 (C Library, [lib.c.math]).

Definition at line 79 of file math/abs.hh.

References `abs()`.

### 9.108.2.3 `template<unsigned n> value::int_u< n > mln::math::abs ( const value::int_u< n > & v ) [inline]`

Specialization for `mln::value::int_u`.

Definition at line 88 of file math/abs.hh.

## 9.109 mln::metal Namespace Reference

Namespace of meta-programming tools.

### Namespaces

- namespace `impl`  
*Implementation namespace of metal namespace.*
- namespace `math`  
*Namespace of static mathematical functions.*

### Classes

- struct `ands`  
*Ands type.*
- struct `converts_to`

*"converts-to" check.*

- struct [equal](#)  
*Definition of a static 'equal' test.*
- struct [goes\\_to](#)  
*"goes-to" check.*
- struct [is](#)  
*"is" check.*
- struct [is\\_a](#)  
*"is\_a" check.*
- struct [is\\_not](#)  
*"is\_not" check.*
- struct [is\\_not\\_a](#)  
*"is\_not\_a" static Boolean expression.*

### 9.109.1 Detailed Description

Namespace of meta-programming tools.

## 9.110 `mln::metal::impl` Namespace Reference

Implementation namespace of metal namespace.

### 9.110.1 Detailed Description

Implementation namespace of metal namespace.

## 9.111 `mln::metal::math` Namespace Reference

Namespace of static mathematical functions.

### Namespaces

- namespace [impl](#)  
*Implementation namespace of [metal::math](#) namespace.*

### 9.111.1 Detailed Description

Namespace of static mathematical functions.

## 9.112 mln::metal::math::impl Namespace Reference

Implementation namespace of [metal::math](#) namespace.

### 9.112.1 Detailed Description

Implementation namespace of [metal::math](#) namespace.

## 9.113 mln::morpho Namespace Reference

Namespace of mathematical morphology routines.

### Namespaces

- namespace [approx](#)  
*Namespace of approximate mathematical morphology routines.*
- namespace [attribute](#)  
*Namespace of attributes used in mathematical morphology.*
- namespace [elementary](#)  
*Namespace of image processing routines of elementary mathematical morphology.*
- namespace [impl](#)  
*Namespace of mathematical morphology routines implementations.*
- namespace [reconstruction](#)  
*Namespace of morphological reconstruction routines.*
- namespace [tree](#)  
*Namespace of morphological tree-related routines.*
- namespace [watershed](#)  
*Namespace of morphological watershed routines.*

### Functions

- `template<typename I >`  
`mln::trait::concrete< I >::ret complementation (const Image< I > &input)`
- `template<typename I >`  
`void complementation\_inplace (Image< I > &input)`
- `template<typename I, typename W >`  
`mln::trait::concrete< I >::ret contrast (const Image< I > &input, const Window< W > &win)`
- `template<typename I, typename W >`  
`mln::trait::concrete< I >::ret dilation (const Image< I > &input, const Window< W > &win)`  
*Morphological dilation.*

- `template<typename I, typename W >`  
`mln::trait::concrete< I >::ret erosion (const Image< I > &input, const Window< W > &win)`  
*Morphological erosion.*
- `template<typename Op, typename I, typename W >`  
`mln::trait::concrete< I >::ret general (const Op &op, const Image< I > &input, const Window< W > &win)`  
*Morphological general routine.*
- `template<typename I, typename W >`  
`mln::trait::concrete< I >::ret gradient (const Image< I > &input, const Window< W > &win)`  
*Morphological gradient.*
- `template<typename I, typename W >`  
`mln::trait::concrete< I >::ret gradient\_external (const Image< I > &input, const Window< W > &win)`  
*Morphological external gradient.*
- `template<typename I, typename W >`  
`mln::trait::concrete< I >::ret gradient\_internal (const Image< I > &input, const Window< W > &win)`  
*Morphological internal gradient.*
- `template<typename I, typename Wh, typename Wm >`  
`mln::trait::concrete< I >::ret hit\_or\_miss (const Image< I > &input, const Window< Wh > &win_hit, const Window< Wm > &win_miss)`  
*Morphological hit-or-miss.*
- `template<typename I, typename Wh, typename Wm >`  
`mln::trait::concrete< I >::ret hit\_or\_miss\_background\_closing (const Image< I > &input, const Window< Wh > &win_hit, const Window< Wm > &win_miss)`  
*Morphological hit-or-miss closing of the background.*
- `template<typename I, typename Wh, typename Wm >`  
`mln::trait::concrete< I >::ret hit\_or\_miss\_background\_opening (const Image< I > &input, const Window< Wh > &win_hit, const Window< Wm > &win_miss)`  
*Morphological hit-or-miss opening of the background.*
- `template<typename I, typename Wh, typename Wm >`  
`mln::trait::concrete< I >::ret hit\_or\_miss\_closing (const Image< I > &input, const Window< Wh > &win_hit, const Window< Wm > &win_miss)`  
*Morphological hit-or-miss closing.*
- `template<typename I, typename Wh, typename Wm >`  
`mln::trait::concrete< I >::ret hit\_or\_miss\_opening (const Image< I > &input, const Window< Wh > &win_hit, const Window< Wm > &win_miss)`  
*Morphological hit-or-miss opening.*
- `template<typename I, typename W, typename O >`  
`void laplacian (const Image< I > &input, const Window< W > &win, Image< O > &output)`

- `template<typename V >`  
`edge_image< util::site_pair< point2d >, V, util::graph > line_gradient (const mln::image2d< V > &ima)`  
*Create a line graph image representing the gradient norm of a `mln::image2d`.*
- `template<typename L, typename I, typename N >`  
`mln::trait::ch_value< I, L >::ret meyer_wst (const Image< I > &input, const Neighborhood< N > &nbh, L &nbasins)`  
*Meyer's Watershed Transform (WST) algorithm.*
- `template<typename L, typename I, typename N >`  
`mln::trait::ch_value< I, L >::ret meyer_wst (const Image< I > &input, const Neighborhood< N > &nbh)`  
*Meyer's Watershed Transform (WST) algorithm, with no count of basins.*
- `template<typename I, typename J >`  
`mln::trait::concrete< I >::ret min (const Image< I > &lhs, const Image< J > &rhs)`
- `template<typename I, typename J >`  
`void min_inplace (Image< I > &lhs, const Image< J > &rhs)`
- `template<typename I, typename J >`  
`mln::trait::concrete< I >::ret minus (const Image< I > &lhs, const Image< J > &rhs)`
- `template<typename I, typename J >`  
`mln::trait::concrete< I >::ret plus (const Image< I > &lhs, const Image< J > &rhs)`
- `template<typename I, typename W >`  
`mln::trait::concrete< I >::ret rank_filter (const Image< I > &input, const Window< W > &win, unsigned k)`  
*Morphological rank\_filter.*
- `template<typename I, typename Wfg, typename Wbg >`  
`mln::trait::concrete< I >::ret thick_miss (const Image< I > &input, const Window< Wfg > &win_fg, const Window< Wbg > &win_bg)`
- `template<typename I, typename Wfg, typename Wbg >`  
`mln::trait::concrete< I >::ret thickening (const Image< I > &input, const Window< Wfg > &win_fg, const Window< Wbg > &win_bg)`
- `template<typename I, typename Wfg, typename Wbg >`  
`mln::trait::concrete< I >::ret thin_fit (const Image< I > &input, const Window< Wfg > &win_fg, const Window< Wbg > &win_bg)`
- `template<typename I, typename Wfg, typename Wbg >`  
`mln::trait::concrete< I >::ret thinning (const Image< I > &input, const Window< Wfg > &win_fg, const Window< Wbg > &win_bg)`  
*Morphological thinning.*
- `template<typename I, typename W >`  
`mln::trait::concrete< I >::ret top_hat_black (const Image< I > &input, const Window< W > &win)`  
*Morphological black top-hat (for background / dark objects).*
- `template<typename I, typename W >`  
`mln::trait::concrete< I >::ret top_hat_self_complementary (const Image< I > &input, const Window< W > &win)`  
*Morphological self-complementary top-hat.*

- `template<typename I, typename W > mln::trait::concrete< I >::ret top_hat_white (const Image< I > &input, const Window< W > &win)`

*Morphological white top-hat (for object / light objects).*

### 9.113.1 Detailed Description

Namespace of mathematical morphology routines.

### 9.113.2 Function Documentation

#### 9.113.2.1 `template<typename I > mln::trait::concrete< I >::ret mln::morpho::complementation ( const Image< I > & input ) [inline]`

Morphological complementation: either a logical "not" (if morpho on sets) or an arithmetical complementation (if morpho on functions).

Definition at line 116 of file complementation.hh.

Referenced by `hit_or_miss_background_closing()`, `hit_or_miss_background_opening()`, `hit_or_miss_closing()`, and `thinning()`.

#### 9.113.2.2 `template<typename I > void mln::morpho::complementation_inplace ( Image< I > & input ) [inline]`

Morphological complementation, inplace version: either a logical "not" (if morpho on sets) or an arithmetical complementation (if morpho on functions).

Definition at line 130 of file complementation.hh.

#### 9.113.2.3 `template<typename I, typename W > mln::trait::concrete< I >::ret mln::morpho::contrast ( const Image< I > & input, const Window< W > & win ) [inline]`

Morphological contrast operator (based on top-hats).

This operator is  $Id + wth\_B - bth\_B$ .

Definition at line 57 of file contrast.hh.

References `plus()`, `top_hat_black()`, and `top_hat_white()`.

#### 9.113.2.4 `template<typename I, typename W > mln::trait::concrete< I >::ret mln::morpho::dilation ( const Image< I > & input, const Window< W > & win ) [inline]`

Morphological dilation.

Definition at line 162 of file dilation.hh.

References `general()`.

Referenced by `gradient()`, `gradient_external()`, `hit_or_miss_background_opening()`, `hit_or_miss_opening()`, `laplacian()`, `mln::morpho::opening::approx::structural()`, and `mln::morpho::closing::approx::structural()`.

**9.113.2.5** `template<typename I , typename W > mln::trait::concrete< I >::ret  
mln::morpho::erosion ( const Image< I > & input, const Window< W > & win )  
[inline]`

Morphological erosion.

Definition at line 163 of file `erosion.hh`.

References `general()`.

Referenced by `gradient()`, `gradient_internal()`, `laplacian()`, `mln::morpho::opening::approx::structural()`, and `mln::morpho::closing::approx::structural()`.

**9.113.2.6** `template<typename Op , typename I , typename W > mln::trait::concrete< I >::ret  
mln::morpho::general ( const Op & op, const Image< I > & input, const Window< W > & win ) [inline]`

Morphological general routine.

Definition at line 168 of file `general.hh`.

Referenced by `dilation()`, and `erosion()`.

**9.113.2.7** `template<typename I , typename W > mln::trait::concrete< I >::ret  
mln::morpho::gradient ( const Image< I > & input, const Window< W > & win )  
[inline]`

Morphological gradient.

This operator is  $d_B - e_B$ .

Definition at line 76 of file `gradient.hh`.

References `dilation()`, `erosion()`, `minus()`, and `mln::test::positive()`.

**9.113.2.8** `template<typename I , typename W > mln::trait::concrete< I >::ret  
mln::morpho::gradient_external ( const Image< I > & input, const Window< W > &  
win ) [inline]`

Morphological external gradient.

This operator is  $d_B - Id$ .

Definition at line 110 of file `gradient.hh`.

References `dilation()`, `minus()`, and `mln::test::positive()`.

**9.113.2.9** `template<typename I , typename W > mln::trait::concrete< I >::ret  
mln::morpho::gradient_internal ( const Image< I > & input, const Window< W > &  
win ) [inline]`

Morphological internal gradient.

This operator is  $\text{Id} - e_B$ .

Definition at line 93 of file `gradient.hh`.

References `erosion()`, `minus()`, and `mln::test::positive()`.

**9.113.2.10** `template<typename I , typename Wh , typename Wm > mln::trait::concrete< I >::ret  
mln::morpho::hit_or_miss ( const Image< I > & input, const Window< Wh > &  
win_hit, const Window< Wm > & win_miss ) [inline]`

Morphological hit-or-miss.

This operator is  $\text{HMT}_-(B_h, B_m) = e_{B_h} \wedge (e_{B_m} \circ C)$ .

Definition at line 276 of file `hit_or_miss.hh`.

Referenced by `thickening()`, and `thinning()`.

**9.113.2.11** `template<typename I , typename Wh , typename Wm > mln::trait::concrete< I >::ret  
mln::morpho::hit_or_miss_background_closing ( const Image< I > & input, const  
Window< Wh > & win_hit, const Window< Wm > & win_miss ) [inline]`

Morphological hit-or-miss closing of the background.

This operator is  $C \circ \text{HMT}_{\text{TopeBG}} \circ C$ .

Definition at line 357 of file `hit_or_miss.hh`.

References `complementation()`, `hit_or_miss_background_opening()`, and `hit_or_miss_closing()`.

**9.113.2.12** `template<typename I , typename Wh , typename Wm > mln::trait::concrete< I >::ret  
mln::morpho::hit_or_miss_background_opening ( const Image< I > & input, const  
Window< Wh > & win_hit, const Window< Wm > & win_miss ) [inline]`

Morphological hit-or-miss opening of the background.

This operator is  $\text{HMT}_{\text{TopeBG}} = \text{HMT}_{\text{Tope}}(B_m, B_h) \circ C = d_-(B_m) \circ \text{HMT}_-(B_h, B_m)$ .

Definition at line 314 of file `hit_or_miss.hh`.

References `complementation()`, `dilation()`, `hit_or_miss_opening()`, and `mln::win::sym()`.

Referenced by `hit_or_miss_background_closing()`, and `thick_miss()`.

**9.113.2.13** `template<typename I , typename Wh , typename Wm > mln::trait::concrete< I >::ret  
mln::morpho::hit_or_miss_closing ( const Image< I > & input, const Window< Wh  
> & win_hit, const Window< Wm > & win_miss ) [inline]`

Morphological hit-or-miss closing.

This operator is  $C \circ \text{HMT}_{\text{Tope}} \circ C$ .

Definition at line 337 of file `hit_or_miss.hh`.

References `complementation()`, and `hit_or_miss_opening()`.

Referenced by `hit_or_miss_background_closing()`.



**9.113.2.14** `template<typename I , typename Wh , typename Wm > mln::trait::concrete< I >::ret mln::morpho::hit_or_miss_opening ( const Image< I > & input, const Window< Wh > & win_hit, const Window< Wm > & win_miss ) [inline]`

Morphological hit-or-miss opening.

This operator is  $HMT_{\text{ope}}(B_h, B_m) = d_{-}(B_h) \circ HMT_{-}(B_h, B_m)$ .

Definition at line 294 of file hit\_or\_miss.hh.

References dilation(), and mln::win::sym().

Referenced by hit\_or\_miss\_background\_opening(), hit\_or\_miss\_closing(), and thin\_fit().

**9.113.2.15** `template<typename I , typename W , typename O > void mln::morpho::laplacian ( const Image< I > & input, const Window< W > & win, Image< O > & output ) [inline]`

Morphological laplacian.

This operator is  $(d_B - Id) - (Id - e_B)$ .

Definition at line 63 of file laplacian.hh.

References dilation(), erosion(), mln::data::fill(), and minus().

**9.113.2.16** `template<typename V > edge_image< util::site_pair< point2d >, V, util::graph > mln::morpho::line_gradient ( const mln::image2d< V > & ima )`

Create a line graph image representing the gradient norm of a [mln::image2d](#).

Definition at line 70 of file line\_gradient.hh.

References mln::math::abs(), mln::image2d< T >::domain(), mln::box< P >::has(), mln::window< D >::insert(), and mln::Box< E >::nsites().

**9.113.2.17** `template<typename L , typename I , typename N > mln::trait::ch_value< I, L >::ret mln::morpho::meyer_wst ( const Image< I > & input, const Neighborhood< N > & nbh, L & nbasins )`

Meyer's Watershed Transform (WST) algorithm.

#### Parameters

[in] *input* The input image.

[in] *nbh* The connexity of markers.

[out] *nbasins* The number of basins.

- *L* is the type of labels, used to number the watershed itself (with the minimal value), and the basins.
- *I* is the exact type of the input image.
- *N* is the exact type of the neighborhood used to express *input's* connexity.

Definition at line 108 of file meyer\_wst.hh.

References mln::data::fill(), mln::p\_priority< P, Q >::front(), mln::initialize(), mln::p\_priority< P, Q >::pop(), mln::p\_priority< P, Q >::push(), mln::labeling::regional\_minima(), and mln::literal::zero.

**9.113.2.18** `template<typename L , typename I , typename N > mln::trait::ch_value< I , L >::ret mln::morpho::meyer_wst ( const Image< I > & input, const Neighborhood< N > & nbh )`

Meyer's Watershed Transform (WST) algorithm, with no count of basins.

#### Parameters

[in] *input* The input image.

[in] *nbh* The connexity of markers.

- *L* is the type of labels, used to number the watershed itself (with the minimal value), and the basins.
- *I* is the exact type of the input image.
- *N* is the exact type of the neighborhood used to express *input*'s connexity.

Note that the first parameter, *L*, is not automatically valued from the type of the actual argument during implicit instantiation: you have to explicitly pass this parameter at call sites.

Definition at line 202 of file meyer\_wst.hh.

**9.113.2.19** `template<typename I , typename J > mln::trait::concrete< I >::ret mln::morpho::min ( const Image< I > & lhs, const Image< J > & rhs ) [inline]`

Morphological min: either a logical "and" (if morpho on sets) or an arithmetical min (if morpho on functions).

Definition at line 112 of file morpho/min.hh.

**9.113.2.20** `template<typename I , typename J > void mln::morpho::min_inplace ( Image< I > & lhs, const Image< J > & rhs ) [inline]`

Morphological min, inplace version: either a logical "and" (if morpho on sets) or an arithmetical min (if morpho on functions).

Definition at line 125 of file morpho/min.hh.

**9.113.2.21** `template<typename I , typename J > mln::trait::concrete< I >::ret mln::morpho::minus ( const Image< I > & lhs, const Image< J > & rhs ) [inline]`

Morphological minus: either a logical "and not" (if morpho on sets) or an arithmetical minus (if morpho on functions).

Definition at line 87 of file morpho/minus.hh.

Referenced by `gradient()`, `gradient_external()`, `gradient_internal()`, `laplacian()`, `thin_fit()`, `thinning()`, `top_hat_black()`, `mln::morpho::elementary::top_hat_black()`, `top_hat_self_complementary()`, `top_hat_white()`, and `mln::morpho::elementary::top_hat_white()`.

**9.113.2.22** `template<typename I , typename J > mln::trait::concrete< I >::ret mln::morpho::plus ( const Image< I > & lhs, const Image< J > & rhs ) [inline]`

Morphological plus: either a "logical or" (if morpho on sets) or an "arithmetical plus" (if morpho on functions).

Definition at line 86 of file morpho/plus.hh.

Referenced by `contrast()`, `thick_miss()`, and `thickening()`.

**9.113.2.23** `template<typename I , typename W > mln::trait::concrete< I >::ret mln::morpho::rank_filter ( const Image< I > & input, const Window< W > & win, unsigned k ) [inline]`

Morphological `rank_filter`.

Definition at line 213 of file rank\_filter.hh.

**9.113.2.24** `template<typename I , typename Wfg , typename Wbg > mln::trait::concrete< I >::ret mln::morpho::thick_miss ( const Image< I > & input, const Window< Wfg > & win_fg, const Window< Wbg > & win_bg ) [inline]`

Morphological thick-miss.

This operator is  $THICK\_B = Id + HMTopeBG\_B$ , where  $B = (Bfg, Bbg)$ .

Definition at line 59 of file thick\_miss.hh.

References `hit_or_miss_background_opening()`, and `plus()`.

**9.113.2.25** `template<typename I , typename Wfg , typename Wbg > mln::trait::concrete< I >::ret mln::morpho::thickening ( const Image< I > & input, const Window< Wfg > & win_fg, const Window< Wbg > & win_bg ) [inline]`

Morphological thickening.

This operator is  $THICK\_B = Id + HMT\_B$ , where  $B = (Bfg, Bbg)$ .

Definition at line 88 of file thickening.hh.

References `hit_or_miss()`, and `plus()`.

Referenced by `thinning()`.

**9.113.2.26** `template<typename I , typename Wfg , typename Wbg > mln::trait::concrete< I >::ret mln::morpho::thin_fit ( const Image< I > & input, const Window< Wfg > & win_fg, const Window< Wbg > & win_bg ) [inline]`

Morphological thin-fit.

This operator is  $THIN\_B = Id - HMTope\_B$  where  $B = (Bfg, Bbg)$ .

Definition at line 87 of file thin\_fit.hh.

References `hit_or_miss_opening()`, and `minus()`.

**9.113.2.27** `template<typename I , typename Wfg , typename Wbg > mln::trait::concrete< I >::ret mln::morpho::thinning ( const Image< I > & input, const Window< Wfg > & win_fg, const Window< Wbg > & win_bg ) [inline]`

Morphological thinning.

This operator is  $\text{THIN\_B} = \text{Id} - \text{HMT\_B}$ , where  $\text{B} = (\text{Bfg}, \text{Bbg})$ .

Definition at line 90 of file `thinning.hh`.

References `complementation()`, `hit_or_miss()`, `minus()`, and `thickening()`.

**9.113.2.28** `template<typename I , typename W > mln::trait::concrete< I >::ret mln::morpho::top_hat_black ( const Image< I > & input, const Window< W > & win ) [inline]`

Morphological black top-hat (for background / dark objects).

This operator is  $\text{clo\_B} - \text{Id}$ .

Definition at line 102 of file `top_hat.hh`.

References `minus()`, and `mln::test::positive()`.

Referenced by `contrast()`.

**9.113.2.29** `template<typename I , typename W > mln::trait::concrete< I >::ret mln::morpho::top_hat_self_complementary ( const Image< I > & input, const Window< W > & win ) [inline]`

Morphological self-complementary top-hat.

This operator is

$= \text{top\_hat\_white} + \text{top\_hat\_black}$

$= (\text{input} - \text{opening}) + (\text{closing} - \text{input})$

$= \text{closing} - \text{opening}$ .

Definition at line 121 of file `top_hat.hh`.

References `minus()`, and `mln::test::positive()`.

**9.113.2.30** `template<typename I , typename W > mln::trait::concrete< I >::ret mln::morpho::top_hat_white ( const Image< I > & input, const Window< W > & win ) [inline]`

Morphological white top-hat (for object / light objects).

This operator is  $\text{Id} - \text{ope\_B}$ .

Definition at line 83 of file `top_hat.hh`.

References `minus()`, and `mln::test::positive()`.

Referenced by `contrast()`.

## 9.114 mln::morpho::approx Namespace Reference

Namespace of approximate mathematical morphology routines.

### 9.114.1 Detailed Description

Namespace of approximate mathematical morphology routines.

## 9.115 mln::morpho::attribute Namespace Reference

Namespace of attributes used in mathematical morphology.

### Classes

- class [card](#)  
*Cardinality accumulator class.*
- struct [count\\_adjacent\\_vertices](#)  
*Count\_Adjacent\_Vertices accumulator class.*
- struct [height](#)  
*Height accumulator class.*
- struct [sharpness](#)  
*Sharpness accumulator class.*
- class [sum](#)  
*Suminality accumulator class.*
- struct [volume](#)  
*Volume accumulator class.*

### 9.115.1 Detailed Description

Namespace of attributes used in mathematical morphology.

## 9.116 mln::morpho::closing::approx Namespace Reference

Namespace of approximate mathematical morphology closing routines.

### Functions

- `template<typename I, typename W >  
mln::trait::concrete< I >::ret structural (const Image< I > &input, const Window< W > &win)`  
*Approximate of morphological structural closing.*

### 9.116.1 Detailed Description

Namespace of approximate mathematical morphology closing routines.

### 9.116.2 Function Documentation

**9.116.2.1** `template<typename I , typename W > mln::trait::concrete< I >::ret  
mln::morpho::closing::approx::structural ( const Image< I > & input, const  
Window< W > & win ) [inline]`

Approximate of morphological structural closing.

This operator is  $e_{-B} \circ d_B$ .

Definition at line 65 of file closing/approx/structural.hh.

References `mln::morpho::dilation()`, `mln::morpho::erosion()`, and `mln::win::sym()`.

## 9.117 mln::morpho::elementary Namespace Reference

Namespace of image processing routines of elementary mathematical morphology.

### Functions

- `template<typename I , typename N >  
mln::trait::concrete< I >::ret closing (const Image< I > &input, const Neighborhood< N >  
&nbh)`  
*Morphological elementary closing.*
- `template<typename I , typename N >  
mln_trait_op_minus_twice (typename mln::trait::concrete< I >::ret) laplacian(const Image< I >  
&input)`  
*Morphological elementary laplacian.*
- `template<typename I , typename N >  
mln::trait::concrete< I >::ret opening (const Image< I > &input, const Neighborhood< N >  
&nbh)`  
*Morphological elementary opening.*
- `template<typename I , typename N >  
mln::trait::concrete< I >::ret top_hat_black (const Image< I > &input, const Neighborhood< N >  
&nbh)`  
*Morphological elementary black top-hat (for background / dark objects).*
- `template<typename I , typename N >  
mln::trait::concrete< I >::ret top_hat_white (const Image< I > &input, const Neighborhood< N >  
&nbh)`  
*Morphological elementary white top-hat (for object / light objects).*

## 9.117.1 Detailed Description

Namespace of image processing routines of elementary mathematical morphology.

## 9.117.2 Function Documentation

**9.117.2.1** `template<typename I , typename N > mln::trait::concrete< I >::ret  
mln::morpho::elementary::closing ( const Image< I > & input, const Neighborhood<  
N > & nbh ) [inline]`

Morphological elementary closing.

This operator is e o d.

Definition at line 58 of file closing.hh.

Referenced by top\_hat\_black().

**9.117.2.2** `template<typename I , typename N > mln::morpho::elementary::mln_  
trait_op_minus_twice ( typename mln::trait::concrete< I >::ret ) const  
[inline]`

Morphological elementary laplacian.

This operator is (d - id) - (id - e).

**9.117.2.3** `template<typename I , typename N > mln::trait::concrete< I >::ret  
mln::morpho::elementary::opening ( const Image< I > & input, const Neighborhood<  
N > & nbh ) [inline]`

Morphological elementary opening.

This operator is d o e.

Definition at line 58 of file opening.hh.

Referenced by top\_hat\_white().

**9.117.2.4** `template<typename I , typename N > mln::trait::concrete< I >::ret  
mln::morpho::elementary::top_hat_black ( const Image< I > & input, const  
Neighborhood< N > & nbh ) [inline]`

Morphological elementary black top-hat (for background / dark objects).

This operator is clo - Id.

Definition at line 105 of file elementary/top\_hat.hh.

References closing(), mln::morpho::minus(), and mln::test::positive().

**9.117.2.5** `template<typename I , typename N > mln::trait::concrete< I >::ret  
mln::morpho::elementary::top_hat_white ( const Image< I > & input, const  
Neighborhood< N > & nbh ) [inline]`

Morphological elementary white top-hat (for object / light objects).

This operator is Id - ope.

Definition at line 85 of file elementary/top\_hat.hh.

References `mln::morpho::minus()`, `opening()`, and `mln::test::positive()`.

## 9.118 `mln::morpho::impl` Namespace Reference

Namespace of mathematical morphology routines implementations.

### Namespaces

- namespace [generic](#)

*Namespace of mathematical morphology routines generic implementations.*

#### 9.118.1 Detailed Description

Namespace of mathematical morphology routines implementations.

## 9.119 `mln::morpho::impl::generic` Namespace Reference

Namespace of mathematical morphology routines generic implementations.

#### 9.119.1 Detailed Description

Namespace of mathematical morphology routines generic implementations.

## 9.120 `mln::morpho::opening::approx` Namespace Reference

Namespace of approximate mathematical morphology opening routines.

### Functions

- `template<typename I , typename W > mln::trait::concrete< I >::ret structural (const Image< I > &input, const Window< W > &win)`

*Approximate of morphological structural opening.*

#### 9.120.1 Detailed Description

Namespace of approximate mathematical morphology opening routines.



## 9.120.2 Function Documentation

**9.120.2.1** `template<typename I , typename W > mln::trait::concrete< I >::ret  
mln::morpho::opening::approx::structural ( const Image< I > & input, const  
Window< W > & win ) [inline]`

Approximate of morphological structural opening.

This operator is  $d_{\{-B\}} \circ e_B$ .

Definition at line 64 of file opening/approx/structural.hh.

References `mln::morpho::dilation()`, `mln::morpho::erosion()`, and `mln::win::sym()`.

## 9.121 mln::morpho::reconstruction Namespace Reference

Namespace of morphological reconstruction routines.

### Namespaces

- namespace [by\\_dilation](#)

*Namespace of morphological reconstruction by dilation routines.*

- namespace [by\\_erosion](#)

*Namespace of morphological reconstruction by erosion routines.*

### 9.121.1 Detailed Description

Namespace of morphological reconstruction routines.

## 9.122 mln::morpho::reconstruction::by\_dilation Namespace Reference

Namespace of morphological reconstruction by dilation routines.

### 9.122.1 Detailed Description

Namespace of morphological reconstruction by dilation routines.

## 9.123 mln::morpho::reconstruction::by\_erosion Namespace Reference

Namespace of morphological reconstruction by erosion routines.

### 9.123.1 Detailed Description

Namespace of morphological reconstruction by erosion routines.

## 9.124 mln::morpho::tree Namespace Reference

Namespace of morphological tree-related routines.

### Namespaces

- namespace [filter](#)  
*Namespace for attribute filtering.*

### Functions

- `template<typename A , typename T >`  
`mln::trait::ch_value< typename T::function, typename A::result >::ret compute\_attribute\_image`  
`(const Accumulator< A > &a, const T &t, mln::trait::ch_value< typename T::function, A >::ret`  
`*accu_image=0)`  
*Compute an attribute image using tree with a parent relationship between sites.*
- `template<typename A , typename T , typename V >`  
`mln::trait::ch_value< typename T::function, typename A::result >::ret compute\_attribute\_image\_`  
`from (const Accumulator< A > &a, const T &t, const Image< V > &values, mln::trait::ch_value<`  
`typename T::function, A >::ret *accu_image=0)`  
*The same as `compute_attribute_image` but uses the values stored by `values` image instead.*
- `template<typename I , typename N , typename S >`  
`mln::trait::ch_value< I, typename I::psite >::ret compute\_parent (const Image< I > &f, const`  
`Neighborhood< N > &nbh, const Site\_Set< S > &s)`  
*Compute a tree with a parent relationship between sites.*
- `template<typename I , typename N >`  
`data< I, p\_array< typename I::psite > > dual\_input\_max\_tree (const Image< I > &f, const Image<`  
`I > &m, const Neighborhood< N > &nbh)`  
*Compute the dual input max tree using mask-based connectivity.*
- `template<typename I , typename N >`  
`data< I, p\_array< typename I::psite > > max\_tree (const Image< I > &f, const Neighborhood< N`  
`> &nbh)`  
*Compute a canonized max-tree.*
- `template<typename I , typename N >`  
`data< I, p\_array< typename I::psite > > min\_tree (const Image< I > &f, const Neighborhood< N`  
`> &nbh)`  
*Compute a canonized min-tree.*

- `template<typename T, typename A, typename P, typename W >`  
`void propagate_if (const T &tree, Image< A > &a_, const way_of_propagation< W > &prop_,`  
`const Function_v2b< P > &pred_, const typename A::value &v)`
- `template<typename T, typename A, typename P >`  
`void propagate_if (const T &tree, Image< A > &a_, const desc_propagation &prop_, const`  
`Function_v2b< P > &pred_)`
- `template<typename T, typename A, typename W >`  
`void propagate_if_value (const T &tree, Image< A > &a_, const way_of_propagation< W >`  
`&prop_, const typename A::value &v)`
- `template<typename T, typename A, typename W >`  
`void propagate_if_value (const T &tree, Image< A > &a_, const way_of_propagation< W >`  
`&prop_, const typename A::value &v, const typename A::value &v_prop)`
- `template<typename T, typename A >`  
`void propagate_node_to_ancestors (typename A::psite n, const T &t, Image< A > &a_, const type-`  
`name A::value &v)`
- `template<typename T, typename A >`  
`void propagate_node_to_ancestors (typename A::psite n, const T &t, Image< A > &a_)`
- `template<typename T, typename A >`  
`void propagate_node_to_descendants (typename A::psite n, const T &t, Image< A > &a_, const`  
`typename A::value &v, unsigned *nb_leaves=0)`
- `template<typename T, typename A >`  
`void propagate_node_to_descendants (typename A::psite &n, const T &t, Image< A > &a_, un-`  
`signed *nb_leaves=0)`
- `template<typename T, typename F >`  
`void propagate_representative (const T &t, Image< F > &f_)`

*Propagate the representative node's value to non-representative points of the component.*

### 9.124.1 Detailed Description

Namespace of morphological tree-related routines.

### 9.124.2 Function Documentation

**9.124.2.1** `template<typename A, typename T > mln::trait::ch_value< typename T::function,`  
`typename A::result >::ret mln::morpho::tree::compute_attribute_image ( const`  
`Accumulator< A > & a, const T & t, mln::trait::ch_value< typename T::function, A`  
`>::ret * accu_image = 0 ) [inline]`

Compute an attribute image using tree with a parent relationship between sites.

In the attribute image, the resulting value at a node is the 'sum' of its sub-components value + the attribute value at this node.

Warning: `s` translates the ordering related to the "natural" childhood relationship. The parenthood is thus inverted w.r.t. to `s`.

It is very convenient since all processing upon the parent tree are performed following `s` (in the default "forward" way).

FIXME: Put it more clearly...

The parent result image verifies:

- $p$  is root iff  $\text{parent}(p) == p$
- $p$  is a node iff either  $p$  is root or  $f(\text{parent}(p)) \neq f(p)$ .

### Parameters

- [in]  $a$  Attribute.
- [in]  $t$  Component tree.
- [out]  $accu\_image$  Optional argument used to store image of attribute accumulator.

### Returns

The attribute image.

Definition at line 216 of file `compute_attribute_image.hh`.

Referenced by `compute_attribute_image_from()`.

**9.124.2.2** `template<typename A , typename T , typename V > mln::trait::ch_value< typename T::function, typename A::result >::ret mln::morpho::tree::compute_attribute_image_from ( const Accumulator< A > & a, const T & t, const Image< V > & values, mln::trait::ch_value< typename T::function, A >::ret * accu_image = 0 ) [inline]`

The same as `compute_attribute_image` but uses the values stored by `values` image instead.

### Parameters

- [in]  $a$  Attribute.
- [in]  $t$  Component tree.
- [in]  $values$  Value image.
- [out]  $accu\_image$  Optional argument used to store image.

### Returns

Definition at line 233 of file `compute_attribute_image.hh`.

References `compute_attribute_image()`.

**9.124.2.3** `template<typename I , typename N , typename S > mln::trait::ch_value< I, typename I::psite >::ret mln::morpho::tree::compute_parent ( const Image< I > & f, const Neighborhood< N > & nbh, const Site_Set< S > & s ) [inline]`

Compute a tree with a parent relationship between sites.

Warning:  $s$  translates the ordering related to the "natural" childhood relationship. The parenthood is thus inverted w.r.t. to  $s$ .

It is very convenient since most processing routines upon the parent tree are performed following  $s$  (in the default "forward" way). Indeed that is the way to propagate information from parents to children.

The parent result image verifies:

- $p$  is root iff  $\text{parent}(p) == p$

- $p$  is a node iff either  $p$  is root or  $f(\text{parent}(p)) \neq f(p)$ .

The choice "s means childhood" is consistent with labeling in binary images. In that particular case, while browsing the image in forward scan (video), we expect to find first a tree root (a first point, representative of a component) and then the other component points. Please note that it leads to increasing values of labels in the "natural" video scan.

Since mathematical morphology on functions is related to morphology on sets, we clearly want to keep the equivalence between "component labeling" and "component filtering" using trees.

FIXME: Put it more clearly... Insert pictures!

A binary image:

- `||--`
- `||-|`
- `-----`
- `-||-`

where `'|'` means true and `'-'` means false.

Its labeling:

```
0 1 1 0 0
0 1 1 0 2
0 0 0 0 0
0 0 3 3 0
```

The corresponding forest:

```
x o . x x
x . . x o
x x x x x
x x o . x
```

where `'x'` means "no data", `'o'` is a tree root (representative point for a component), and `'.'` is a tree regular (non-root) point (in a component by not its representative point).

The forest, with the parent relationship looks like:

```
o < .
^ r
. . o
o < .
```

Definition at line 286 of file compute\_parent.hh.

**9.124.2.4** `template<typename I , typename N > morpho::tree::data< I , p_array< typename I::psite > > mln::morpho::tree::dual_input_max_tree ( const Image< I > & f, const Image< I > & m, const Neighborhood< N > & nbh ) [inline]`

Compute the dual input max tree using mask-based connectivity.

**Parameters**

- [in] *f* The original image.
- [in] *m* The connectivity mask.
- [in] *nbh* The neighborhood of the mask.

**Returns**

The computed tree.

Definition at line 109 of file dual\_input\_tree.hh.

**9.124.2.5** `template<typename I , typename N > data< I, p_array< typename I::psite > >  
mln::morpho::tree::max_tree ( const Image< I > & f, const Neighborhood< N > &  
nbh ) [inline]`

Compute a canonized max-tree.

**Parameters**

- [in] *f* The input image.
- [in] *nbh* The neighborhood.

**Returns**

The corresponding max-tree structure.

Definition at line 100 of file component\_tree.hh.

References mln::data::sort\_psites\_increasing().

**9.124.2.6** `template<typename I , typename N > data< I, p_array< typename I::psite > >  
mln::morpho::tree::min_tree ( const Image< I > & f, const Neighborhood< N > &  
nbh ) [inline]`

Compute a canonized min-tree.

**Parameters**

- [in] *f* The input image.
- [in] *nbh* The neighborhood.

**Returns**

The corresponding min-tree structure.

Definition at line 77 of file component\_tree.hh.

References mln::data::sort\_psites\_decreasing().

**9.124.2.7** `template<typename T , typename A , typename P , typename W > void  
mln::morpho::tree::propagate_if ( const T & tree, Image< A > & a_, const  
way_of_propagation< W > & prop_, const Function_v2b< P > & pred_, const  
typename A::value & v ) [inline]`

Propagate nodes checking the predicate `pred` in the way defined by `way_of_propagation`.

#### Parameters

- tree* Component tree used for propagation.
- a\_* Attributed image where values are propagated.
- prop\_* Propagate node in ascendant or descendant way.
- pred\_* Predicate that node must check to be propagated.
- v* Value to be propagated. (By default `v` is the value at the node being propagated).

Definition at line 293 of file `propagate_if.hh`.

Referenced by `propagate_if()`, `propagate_if_value()`, and `mln::morpho::tree::filter::subtractive()`.

**9.124.2.8** `template<typename T , typename A , typename P > void  
mln::morpho::tree::propagate_if ( const T & tree, Image< A > & a_, const  
desc_propagation & prop_, const Function_v2b< P > & pred_ ) [inline]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 310 of file `propagate_if.hh`.

References `propagate_if()`.

**9.124.2.9** `template<typename T , typename A , typename W > void  
mln::morpho::tree::propagate_if_value ( const T & tree, Image< A > & a_, const  
way_of_propagation< W > & prop_, const typename A::value & v ) [inline]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 278 of file `propagate_if.hh`.

References `propagate_if()`.

**9.124.2.10** `template<typename T , typename A , typename W > void  
mln::morpho::tree::propagate_if_value ( const T & tree, Image< A > & a_, const  
way_of_propagation< W > & prop_, const typename A::value & v, const typename  
A::value & v_prop ) [inline]`

Propagate nodes having the value `v` in the way defined by `way_of_propagation`.

#### Parameters

- tree* Component tree used for propagation.
- a\_* Attributed image where values are propagated.
- prop\_* Propagate node in ascendant or descendant way.

$v$  [Value](#) that node must have to be propagated.

$v_{prop}$  [Value](#) to propagate (By default it is the value at the node being propagated).

Definition at line 262 of file propagate\_if.hh.

References propagate\_if().

**9.124.2.11** `template<typename T , typename A > void mln::morpho::tree::propagate_node_to_ancestors ( typename A::psite  $n$ , const T &  $t$ , Image< A > &  $a_$ , const typename A::value &  $v$  )`

Propagate a value  $v$  from a node  $n$  to its ancestors.

#### Parameters

[in]  $n$  Node to propagate.

[in]  $t$  Component tree used for propagation.

[in]  $a_$  Attribute image where values are propagated.

[in]  $v$  [Value](#) to propagate.

Definition at line 170 of file propagate\_node.hh.

Referenced by propagate\_node\_to\_ancestors().

**9.124.2.12** `template<typename T , typename A > void mln::morpho::tree::propagate_node_to_ancestors ( typename A::psite  $n$ , const T &  $t$ , Image< A > &  $a_$  ) [inline]`

Propagate the node's value to its ancestors.

#### Parameters

[in]  $n$  Node to propagate.

[in]  $t$  Component tree used for propagation.

[in, out]  $a_$  Attribute image where values are propagated.

Definition at line 197 of file propagate\_node.hh.

References propagate\_node\_to\_ancestors().

**9.124.2.13** `template<typename T , typename A > void mln::morpho::tree::propagate_node_to_descendants ( typename A::psite  $n$ , const T &  $t$ , Image< A > &  $a_$ , const typename A::value &  $v$ , unsigned *  $nb\_leaves = 0$  ) [inline]`

Propagate a value  $v$  from a node  $n$  to its descendants.

#### Parameters

[in]  $n$  Node to propagate.

[in]  $t$  Component tree used for propagation.

[in]  $a_$  Attribute image where values are propagated.



[in] *v* [Value](#) to propagate.

[out] *nb\_leaves* Optional. Store the number of leaves in the component.

Definition at line 120 of file propagate\_node.hh.

**9.124.2.14** `template<typename T , typename A > void mln::morpho::tree::propagate_node_to_descendants ( typename A::psite & n, const T & t, Image< A > & a_, unsigned * nb_leaves = 0 ) [inline]`

Propagate the node's value to its descendants.

#### Parameters

[in] *n* Node to propagate.

[in] *t* Component tree used for propagation.

[in] *a\_* Attribute image where values are propagated.

[out] *nb\_leaves* Optional. Store the number of leaves in the component.

**9.124.2.15** `template<typename T , typename F > void mln::morpho::tree::propagate_representative ( const T & t, Image< F > & f_ ) [inline]`

Propagate the representative node's value to non-representative points of the component.

#### Parameters

*t* Component tree.

*f\_* [Value](#) image.

Definition at line 65 of file propagate\_representative.hh.

## 9.125 mln::morpho::tree::filter Namespace Reference

Namespace for attribute filtering.

### Functions

- `template<typename T , typename F , typename P > void direct (const T &tree, Image< F > &f_, const Function_v2b< P > &pred_)`  
*Direct non-pruning strategy.*
- `template<typename T , typename F , typename P > void filter (const T &tree, Image< F > &f_, const Function_v2b< P > &pred_, const typename F::value &v)`  
*Filter the image f\_ with a given value.*
- `template<typename T , typename F , typename P > void max (const T &tree, Image< F > &f_, const Function_v2b< P > &pred_)`

*Max pruning strategy.*

- `template<typename T, typename F, typename P >`  
`void min (const T &tree, Image< F > &f_, const Function_v2b< P > &pred_)`

*Min pruning strategy.*

- `template<typename T, typename F, typename P >`  
`void subtractive (const T &tree, Image< F > &f_, const Function_v2b< P > &pred_)`

*Subtractive pruning strategy.*

## 9.125.1 Detailed Description

Namespace for attribute filtering.

## 9.125.2 Function Documentation

**9.125.2.1** `template<typename T, typename F, typename P > void`  
`mln::morpho::tree::filter::direct ( const T & tree, Image< F > & f_, const`  
`Function_v2b< P > & pred_ ) [inline]`

Direct non-pruning strategy.

A node is removed if it does not verify the predicate. The sub-components remain intact.

### Parameters

- [in] *tree* Component tree.
- [out] *f\_* Image to filter.
- [in] *pred\_* Filtering criterion.

Definition at line 73 of file direct.hh.

**9.125.2.2** `template<typename T, typename F, typename P > void mln::morpho::tree::filter::filter`  
`( const T & tree, Image< F > & f_, const Function_v2b< P > & pred_, const`  
`typename F::value & v ) [inline]`

Filter the image *f\_* with a given value.

The sub-components of nodes that does not match the predicate *pred\_* are filled with the given value *v*.

### Parameters

- tree* Component tree.
- f\_* Image function.
- pred\_* Predicate.
- v* Value to propagate.

Definition at line 73 of file morpho/tree/filter/filter.hh.

References `mln::data::fill()`, and `mln::initialize()`.

**9.125.2.3** `template<typename T , typename F , typename P > void mln::morpho::tree::filter::max ( const T & tree, Image< F > & f_, const Function_v2b< P > & pred_ ) [inline]`

Max pruning strategy.

A node is removed iif all of its children are removed or if it does not verify the predicate `pred_`.

#### Parameters

- [in] *tree* Component tree.
- [out] *f\_ Image* to filter.
- [in] *pred\_* Filtering criterion.

Definition at line 74 of file `morpho/tree/filter/max.hh`.

References `mln::data::fill()`, and `mln::initialize()`.

**9.125.2.4** `template<typename T , typename F , typename P > void mln::morpho::tree::filter::min ( const T & tree, Image< F > & f_, const Function_v2b< P > & pred_ ) [inline]`

Min pruning strategy.

A node is removed iif its parent is removed or if it does not verify the predicate `pred_`.

#### Parameters

- [in] *tree* Component tree.
- [out] *f\_ Image* to filter.
- [in] *pred\_* Filtering criterion.

Definition at line 75 of file `morpho/tree/filter/min.hh`.

References `mln::data::fill()`, and `mln::initialize()`.

**9.125.2.5** `template<typename T , typename F , typename P > void mln::morpho::tree::filter::subtractive ( const T & tree, Image< F > & f_, const Function_v2b< P > & pred_ ) [inline]`

Subtractive pruning strategy.

The node is removed if it does not verify the predicate. The sub-components values are set to the value of the removed component.

#### Parameters

- [in] *tree* Component tree.
- [out] *f\_ Image* to filter.
- [in] *pred\_* Filtering criterion.

Definition at line 77 of file `subtractive.hh`.

References `mln::morpho::tree::propagate_if()`.

## 9.126 mln::morpho::watershed Namespace Reference

Namespace of morphological watershed routines.

### Namespaces

- namespace [watershed](#)  
*Namespace of morphological watershed routines implementations.*

### Functions

- `template<typename L , typename I , typename N > mln::trait::ch_value< I, L >::ret flooding (const Image< I > &input, const Neighborhood< N > &nbh, L &n_basins)`  
*Meyer's Watershed Transform (WST) algorithm.*
- `template<typename L , typename I , typename N > mln::trait::ch_value< I, L >::ret flooding (const Image< I > &input, const Neighborhood< N > &nbh)`  
*Meyer's Watershed Transform (WST) algorithm, with no count of basins.*
- `template<typename I , typename J > mln::trait::ch_value< I, value::rgb8 >::ret superpose (const Image< I > &input, const Image< J > &ws_ima)`  
*Convert an image to a rgb8 image and draw the watershed lines.*
- `template<typename I , typename J > mln::trait::ch_value< I, value::rgb8 >::ret superpose (const Image< I > &input_, const Image< J > &ws_ima_, const value::rgb8 &wsl_color)`  
*Convert an image to a rgb8 image and draw the watershed lines.*
- `template<class T > T::image_t topological (T &tree)`  
*Compute a toological watershed transform from tree.*

### 9.126.1 Detailed Description

Namespace of morphological watershed routines.

### 9.126.2 Function Documentation

- 9.126.2.1** `template<typename L , typename I , typename N > mln::trait::ch_value< I, L >::ret mln::morpho::watershed::flooding ( const Image< I > & input, const Neighborhood< N > & nbh, L & n_basins ) [inline]`

Meyer's Watershed Transform (WST) algorithm.

**Parameters**

- [in] *input* The input image.
- [in] *nbh* The connexity of markers.
- [out] *n\_basins* The number of basins.

- L is the type of labels, used to number the watershed itself (with the minimal value), and the basins.
- I is the exact type of the input image.
- N is the exact type of the neighborhood used to express *input*'s connexity.

Definition at line 381 of file flooding.hh.

**9.126.2.2** `template<typename L , typename I , typename N > mln::trait::ch_value< I, L >::ret  
mln::morpho::watershed::flooding ( const Image< I > & input, const Neighborhood<  
N > & nbh )`

Meyer's Watershed Transform (WST) algorithm, with no count of basins.

**Parameters**

- [in] *input* The input image.
- [in] *nbh* The connexity of markers.

- L is the type of labels, used to number the watershed itself (with the minimal value), and the basins.
- I is the exact type of the input image.
- N is the exact type of the neighborhood used to express *input*'s connexity.

Note that the first parameter, L, is not automatically valued from the type of the actual argument during implicit instantiation: you have to explicitly pass this parameter at call sites.

Definition at line 396 of file flooding.hh.

**9.126.2.3** `template<typename I , typename J > mln::trait::ch_value< I, value::rgb8 >::ret  
mln::morpho::watershed::superpose ( const Image< I > & input, const Image< J > &  
ws_ima ) [inline]`

Convert an image to a rgb8 image and draw the watershed lines.

Definition at line 109 of file morpho/watershed/superpose.hh.

References mln::literal::red, and superpose().

**9.126.2.4** `template<typename I , typename J > mln::trait::ch_value< I, value::rgb8 >::ret  
mln::morpho::watershed::superpose ( const Image< I > & input_, const Image< J > &  
& ws_ima_, const value::rgb8 & wsl_color ) [inline]`

Convert an image to a rgb8 image and draw the watershed lines.

Definition at line 85 of file morpho/watershed/superpose.hh.

References mln::data::convert(), mln::data::fill(), and mln::literal::zero.

Referenced by superpose().

### 9.126.2.5 `template<class T > T::image_t mln::morpho::watershed::topological ( T & tree )`

Compute a toological watershed transform from *tree*.

Definition at line 675 of file `topological.hh`.

References `mln::data::fill()`, `mln::p_priority< P, Q >::front()`, `mln::initialize()`, `mln::p_priority< P, Q >::pop()`, `mln::p_priority< P, Q >::push()`, and `topological()`.

Referenced by `topological()`.

## 9.127 `mln::morpho::watershed::watershed` Namespace Reference

Namespace of morphological watershed routines implementations.

### Namespaces

- namespace [generic](#)

*Namespace of morphological watershed routines generic implementations.*

### 9.127.1 Detailed Description

Namespace of morphological watershed routines implementations.

## 9.128 `mln::morpho::watershed::watershed::generic` Namespace Reference

Namespace of morphological watershed routines generic implementations.

### 9.128.1 Detailed Description

Namespace of morphological watershed routines generic implementations.

## 9.129 `mln::norm` Namespace Reference

Namespace of norms.

### Namespaces

- namespace [impl](#)

*Implementation namespace of norm namespace.*

## Functions

- `template<unsigned n, typename C >`  
`mln::trait::value_< typename mln::trait::op::times< C, C >::ret >::sum l2` (`const C(&vec)[n]`)  
*L2-norm of a vector vec.*
- `template<unsigned n, typename C >`  
`mln::trait::value_< typename mln::trait::op::times< C, C >::ret >::sum l1` (`const C(&vec)[n]`)  
*L1-norm of a vector vec.*
- `template<unsigned n, typename C >`  
`mln::trait::value_< typename mln::trait::op::times< C, C >::ret >::sum l1_distance` (`const C(&vec1)[n], const C(&vec2)[n]`)  
*L1-norm distance between vectors vec1 and vec2.*
- `template<unsigned n, typename C >`  
`mln::trait::value_< typename mln::trait::op::times< C, C >::ret >::sum sqr_l2` (`const C(&vec)[n]`)  
*Squared L2-norm of a vector vec.*
- `template<unsigned n, typename C >`  
`mln::trait::value_< typename mln::trait::op::times< C, C >::ret >::sum l2_distance` (`const C(&vec1)[n], const C(&vec2)[n]`)  
*L2-norm distance between vectors vec1 and vec2.*
- `template<unsigned n, typename C >`  
`C linfy` (`const C(&vec)[n]`)  
*L-infinity-norm of a vector vec.*
- `template<unsigned n, typename C >`  
`C linfy_distance` (`const C(&vec1)[n], const C(&vec2)[n]`)  
*L-infinity-norm distance between vectors vec1 and vec2.*

### 9.129.1 Detailed Description

Namespace of norms.

### 9.129.2 Function Documentation

**9.129.2.1** `template<unsigned n, typename C > mln::trait::value_< typename mln::trait::op::times< C, C >::ret >::sum mln::norm::l1 ( const C(&) vec[n] )`  
**[inline]**

L1-norm of a vector *vec*.

Definition at line 108 of file l1.hh.

**9.129.2.2** `template<unsigned n, typename C > mln::trait::value_< typename  
mln::trait::op::times< C, C >::ret >::sum mln::norm::l1_distance ( const C(&  
vec1[n], const C(&) vec2[n] ) [inline]`

L1-norm distance between vectors *vec1* and *vec2*.

Definition at line 124 of file l1.hh.

**9.129.2.3** `template<unsigned n, typename C > mln::trait::value_< typename  
mln::trait::op::times< C, C >::ret >::sum mln::norm::l2 ( const C(&) vec[n] )  
[inline]`

L2-norm of a vector *vec*.

Definition at line 139 of file l2.hh.

**9.129.2.4** `template<unsigned n, typename C > mln::trait::value_< typename  
mln::trait::op::times< C, C >::ret >::sum mln::norm::l2_distance ( const C(&  
vec1[n], const C(&) vec2[n] ) [inline]`

L2-norm distance between vectors *vec1* and *vec2*.

Definition at line 173 of file l2.hh.

**9.129.2.5** `template<unsigned n, typename C > C mln::norm::linfty ( const C(&) vec[n] )  
[inline]`

L-infinity-norm of a vector *vec*.

Definition at line 113 of file linfty.hh.

**9.129.2.6** `template<unsigned n, typename C > C mln::norm::linfty_distance ( const C(&  
vec1[n], const C(&) vec2[n] ) [inline]`

L-infinity-norm distance between vectors *vec1* and *vec2*.

Definition at line 127 of file linfty.hh.

**9.129.2.7** `template<unsigned n, typename C > mln::trait::value_< typename  
mln::trait::op::times< C, C >::ret >::sum mln::norm::sqr_l2 ( const C(&) vec[n] )  
[inline]`

Squared L2-norm of a vector *vec*.

Definition at line 156 of file l2.hh.

Referenced by `mln::geom::mesh_corner_point_area()`, and `mln::geom::mesh_normal()`.

## 9.130 mln::norm::impl Namespace Reference

Implementation namespace of norm namespace.



### 9.130.1 Detailed Description

Implementation namespace of norm namespace.

## 9.131 mln::opt Namespace Reference

Namespace of optional routines.

### Namespaces

- namespace [impl](#)

*Implementation namespace of opt namespace.*

### Functions

- `template<typename I >`  
`I::rvalue at (const Image< I > &ima, def::coord ind)`  
*One dimension Read-only access to the ima value located at (ind).*
- `template<typename I >`  
`I::lvalue at (Image< I > &ima, def::coord ind)`  
*Read-write access to the ima value located at (ind).*
- `template<typename I >`  
`I::lvalue at (Image< I > &ima, def::coord row, def::coord col)`  
*Read-write access to the ima value located at (row, col).*
- `template<typename I >`  
`I::rvalue at (const Image< I > &ima, def::coord sli, def::coord row, def::coord col)`  
*Three dimensions Read-only access to the ima value located at (sli, row, col).*
- `template<typename I >`  
`I::rvalue at (const Image< I > &ima, def::coord row, def::coord col)`  
*Two dimensions Read-only access to the ima value located at (row, col).*
- `template<typename I >`  
`I::lvalue at (Image< I > &ima, def::coord sli, def::coord row, def::coord col)`  
*Read-write access to the ima value located at (sli, row, col).*

### 9.131.1 Detailed Description

Namespace of optional routines.

## 9.131.2 Function Documentation

### 9.131.2.1 `template<typename I > I::rvalue mln::opt::at ( const Image< I > & ima, def::coord ind ) [inline]`

One dimension Read-only access to the `ima` value located at (`ind`).

Definition at line 151 of file `at.hh`.

Referenced by `mln::transform::hough()`, and `mln::make::image()`.

### 9.131.2.2 `template<typename I > I::lvalue mln::opt::at ( Image< I > & ima, def::coord ind )`

Read-write access to the `ima` value located at (`ind`).

Definition at line 160 of file `at.hh`.

### 9.131.2.3 `template<typename I > I::lvalue mln::opt::at ( Image< I > & ima, def::coord row, def::coord col )`

Read-write access to the `ima` value located at (`row`, `col`).

Definition at line 245 of file `at.hh`.

### 9.131.2.4 `template<typename I > I::rvalue mln::opt::at ( const Image< I > & ima, def::coord sli, def::coord row, def::coord col ) [inline]`

Three dimensions Read-only access to the `ima` value located at (`sli`, `row`, `col`).

Definition at line 320 of file `at.hh`.

### 9.131.2.5 `template<typename I > I::rvalue mln::opt::at ( const Image< I > & ima, def::coord row, def::coord col ) [inline]`

Two dimensions Read-only access to the `ima` value located at (`row`, `col`).

Definition at line 236 of file `at.hh`.

### 9.131.2.6 `template<typename I > I::lvalue mln::opt::at ( Image< I > & ima, def::coord sli, def::coord row, def::coord col )`

Read-write access to the `ima` value located at (`sli`, `row`, `col`).

Definition at line 330 of file `at.hh`.

## 9.132 `mln::opt::impl` Namespace Reference

Implementation namespace of `opt` namespace.

### 9.132.1 Detailed Description

Implementation namespace of opt namespace. Three dimensions.

Two dimensions.

One dimension.

## 9.133 mln::pw Namespace Reference

Namespace of "point-wise" expression tools.

### Classes

- class [image](#)  
*A generic point-wise image implementation.*

### 9.133.1 Detailed Description

Namespace of "point-wise" expression tools.

## 9.134 mln::registration Namespace Reference

Namespace of "point-wise" expression tools.

### Classes

- class [closest\\_point\\_basic](#)  
*Closest point functor based on map distance.*
- class [closest\\_point\\_with\\_map](#)  
*Closest point functor based on map distance.*

### Functions

- `template<typename P, typename F >`  
`algebra::quat get\_rot (const p\_array< P > &P_, const vec3d\_f &mu_P, const vec3d\_f &mu_Yk,`  
`const F &closest_point, const algebra::quat &qR, const vec3d\_f &qT)`  
*FIXME: work only for 3d images.*
- `template<typename P, typename F >`  
`std::pair< algebra::quat, mln\_vec(P)> icp (const p\_array< P > &P_, const p\_array< P > &X, const`  
`F &closest_point, const algebra::quat &initial_rot, const mln\_vec(P)&initial_translation)`  
*Base version of the ICP algorithm. It is called in other variants.*

- `template<typename P , typename F >`  
`composed< translation< P::dim, float >, rotation< P::dim, float > > icp (const p_array< P > &P_, const p_array< P > &X, const F &closest_point)`
- `template<typename P >`  
`composed< translation< P::dim, float >, rotation< P::dim, float > > registration1 (const box< P > &domain, const p_array< P > &P_, const p_array< P > &X)`  
*Call ICP once and return the resulting transformation.*
- `template<typename P >`  
`composed< translation< P::dim, float >, rotation< P::dim, float > > registration2 (const box< P > &domain, const p_array< P > &P_, const p_array< P > &X)`  
*Call ICP 10 times.*
- `template<typename P >`  
`composed< translation< P::dim, float >, rotation< P::dim, float > > registration3 (const box< P > &domain, const p_array< P > &P_, const p_array< P > &X)`  
*Call ICP 10 times.*

### 9.134.1 Detailed Description

Namespace of "point-wise" expression tools.

### 9.134.2 Function Documentation

**9.134.2.1** `template<typename P , typename F > algebra::quat mln::registration::get_rot ( const p_array< P > & P_, const vec3d_f & mu_P, const vec3d_f & mu_Yk, const F & closest_point, const algebra::quat & qR, const vec3d_f & qT )`

FIXME: work only for 3d images.

Definition at line 527 of file icp.hh.

References `mln::p_array< P >::nsites()`.

**9.134.2.2** `template<typename P , typename F > std::pair< algebra::quat, mln_vec(P)> mln::registration::icp ( const p_array< P > & P_, const p_array< P > & X, const F & closest_point, const algebra::quat & initial_rot, const mln_vec(P)& initial_translation )`  
**[inline]**

Base version of the ICP algorithm. It is called in other variants.

Register point in `c` using a function of closest points `closest_point`. This overload allows to specify initial transformations.

#### Parameters

- [in] `P_` The cloud of points.
- [in] `X` the reference surface.
- [in] `closest_point` The function of closest points.
- [in] `initial_rot` An initial rotation.

[in] *initial\_translation* An initial translation.

### Returns

the rigid transformation which may be use later to create a registered image.

WARNING: the function `closest_point` \*MUST\* take float/double vector as arguments. Otherwise the resulting transformation may be wrong due to the truncation of the vector coordinate values.

### Precondition

`P_` and `X` must not be empty.

Reference article: "A Method for Registration of 3-D Shapes", Paul J. Besl and Neil D. McKay, IEEE, 2, February 1992.

Definition at line 612 of file `icp.hh`.

References `mln::geom::bbox()`, `mln::literal::black`, `mln::set::compute()`, `mln::duplicate()`, `mln::box< P >::enlarge()`, `mln::data::fill()`, `mln::literal::green`, `mln::io::ppm::save()`, and `mln::literal::white`.

**9.134.2.3** `template<typename P , typename F > composed< translation<P::dim,float>,rotation<P::dim,float> > mln::registration::icp ( const p_array< P > & P_, const p_array< P > & X, const F & closest_point )`

Register point in `c` using a function of closest points `closest_point`.

### Parameters

[in] `P_` The cloud of points.

[in] `X` the reference surface.

[in] *closest\_point* The function of closest points.

### Returns

the rigid transformation which may be use later to create a registered image.

**9.134.2.4** `template<typename P > composed< translation< P::dim, float >, rotation< P::dim, float > > mln::registration::registration1 ( const box< P > & domain, const p_array< P > & P_, const p_array< P > & X ) [inline]`

Call ICP once and return the resulting transformation.

Definition at line 325 of file `registration.hh`.

**9.134.2.5** `template<typename P > composed< translation< P::dim, float >, rotation< P::dim, float > > mln::registration::registration2 ( const box< P > & domain, const p_array< P > & P_, const p_array< P > & X ) [inline]`

Call ICP 10 times.

Do the first call to ICP with all sites then work on a subset of which size is decreasing. For each call, a distance criterion is computed on a subset. Sites part of the subset which are too far or too close are removed. Removed sites are \*NOT\* reused later in the subset.

Definition at line 345 of file `registration.hh`.

**9.134.2.6** `template<typename P > composed< translation< P::dim, float >, rotation< P::dim, float > > mln::registration::registration3 ( const box< P > & domain, const p_array< P > & P_, const p_array< P > & X ) [inline]`

Call ICP 10 times.

Do the first call to ICP with all sites then work on a subset. For each call, a distance criterion is computed on a subset. A new subset is computed from the whole set of points according to this distance. It will be used in the next call. Removed Sites *\*MAY\** be reintegrated.

Definition at line 365 of file registration.hh.

## 9.135 mln::select Namespace Reference

Select namespace (FIXME doc).

### Classes

- struct [p\\_of](#)  
*Structure [p\\_of](#).*

### 9.135.1 Detailed Description

Select namespace (FIXME doc).

## 9.136 mln::set Namespace Reference

Namespace of image processing routines related to pixel sets.

### Functions

- `template<typename S > unsigned card (const Site_Set< S > &s)`  
*Compute the cardinality of the site set *s*.*
- `template<typename A , typename S > A::result compute (const Accumulator< A > &a, const Site_Set< S > &s)`  
*Compute an accumulator onto a site set.*
- `template<typename A , typename I > A::result compute_with_weights (const Accumulator< A > &a, const Image< I > &w)`  
*Compute an accumulator on a site set described by an image.*
- `template<typename S > S::site get (const Site_Set< S > &s, size_t index)`  
*FIXME.*

- `template<typename S >`  
`bool has (const Site_Set< S > &s, const typename S::site &e)`  
*FIXME.*
- `template<typename A , typename I >`  
`mln_meta_accu_result (A, typename I::site) compute_with_weights(const Meta_Accumulator< A > &a`  
`> &a`  
*Compute an accumulator on a site set described by an image.*
- `template<typename A , typename S >`  
`mln_meta_accu_result (A, typename S::site) compute(const Meta_Accumulator< A > &a`  
`&a`  
*Compute an accumulator onto a site set.*

### 9.136.1 Detailed Description

Namespace of image processing routines related to pixel sets.

### 9.136.2 Function Documentation

#### 9.136.2.1 `template<typename S > unsigned mln::set::card ( const Site_Set< S > & s )` `[inline]`

Compute the cardinality of the site set *s*.

Definition at line 134 of file set/card.hh.

#### 9.136.2.2 `template<typename A , typename S > A::result mln::set::compute ( const Accumulator< A > & a, const Site_Set< S > & s )` `[inline]`

Compute an accumulator onto a site set.

#### Parameters

`[in]` *a* An accumulator.

`[in]` *s* A site set.

#### Returns

The accumulator result.

Definition at line 112 of file set/compute.hh.

Referenced by `mln::registration::icp()`.

#### 9.136.2.3 `template<typename A , typename I > A::result mln::set::compute_with_weights ( const Accumulator< A > & a, const Image< I > & w )` `[inline]`

Compute an accumulator on a site set described by an image.

#### Parameters

`[in]` *a* An accumulator.

[in]  $w$  An image of weights (a site  $\rightarrow$  a weight).

### Returns

The accumulator result.

Definition at line 217 of file compute\_with\_weights.hh.

#### 9.136.2.4 `template<typename S > S::site mln::set::get ( const Site_Set< S > & s, size_t index )`

FIXME.

Definition at line 56 of file set/get.hh.

#### 9.136.2.5 `template<typename S > bool mln::set::has ( const Site_Set< S > & s, const typename S::site & e )`

FIXME.

Definition at line 56 of file set/has.hh.

#### 9.136.2.6 `template<typename A, typename I > mln::set::mln_meta_accu_result ( A, typename I::site ) const [inline]`

Compute an accumulator on a site set described by an image.

### Parameters

[in]  $a$  A meta-accumulator.

[in]  $w$  An image of weights (a site  $\rightarrow$  a weight).

### Returns

The accumulator result.

#### 9.136.2.7 `template<typename A, typename S > mln::set::mln_meta_accu_result ( A, typename S::site ) const`

Compute an accumulator onto a site set.

### Parameters

[in]  $a$  A meta-accumulator.

[in]  $s$  A site set.

## 9.137 `mln::subsampling` Namespace Reference

Namespace of "point-wise" expression tools.



## Functions

- `template<typename I >`  
`mln::trait::concrete< I >::ret antialiased (const Image< I > &input, unsigned factor, const typename I::domain_t &output_domain, unsigned border_thickness)`  
*Antialiased subsampling.*
- `template<typename I >`  
`mln::trait::concrete< I >::ret antialiased (const Image< I > &input, unsigned factor)`
- `template<typename I >`  
`mln::trait::concrete< I >::ret gaussian\_subsampling (const Image< I > &input, float sigma, const typename I::dpsite &first_p, const typename I::site::coord &gap)`  
*Gaussian subsampling FIXME : doxy.*
- `template<typename I >`  
`mln::trait::concrete< I >::ret subsampling (const Image< I > &input, const typename I::site::delta &first_p, const typename I::site::coord &gap)`  
*Subsampling FIXME : doxy.*

### 9.137.1 Detailed Description

Namespace of "point-wise" expression tools.

### 9.137.2 Function Documentation

#### 9.137.2.1 `template<typename I > mln::trait::concrete< I >::ret mln::subsampling::antialiased ( const Image< I > & input, unsigned factor, const typename I::domain_t & output_domain, unsigned border_thickness ) [inline]`

Antialiased subsampling.

#### Parameters

- [in] *input* A gray-level image.
- [in] *factor* Subsampling ratio. Must be divisible by 2 or 3.
- [in] *output\_domain* Force output domain.
- [in] *border\_thickness* Force output border thickness.

Definition at line 418 of file `antialiased.hh`.

Referenced by `antialiased()`.

#### 9.137.2.2 `template<typename I > mln::trait::concrete< I >::ret mln::subsampling::antialiased ( const Image< I > & input, unsigned factor ) [inline]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 443 of file `antialiased.hh`.

References `antialiased()`.

**9.137.2.3** `template<typename I> mln::trait::concrete< I >::ret mln::subsampling::gaussian_subsampling ( const Image< I > & input, float sigma, const typename I::dpsite & first_p, const typename I::site::coord & gap ) [inline]`

Gaussian subsampling FIXME : doxy.

Definition at line 62 of file gaussian\_subsampling.hh.

References `mln::linear::gaussian()`, `mln::geom::ncols()`, and `mln::geom::nrows()`.

**9.137.2.4** `template<typename I> mln::trait::concrete< I >::ret mln::subsampling::subsampling ( const Image< I > & input, const typename I::site::delta & first_p, const typename I::site::coord & gap ) [inline]`

Subsampling FIXME : doxy.

Definition at line 91 of file subsampling.hh.

References `mln::geom::ncols()`, and `mln::geom::nrows()`.

## 9.138 mln::tag Namespace Reference

Namespace of image processing routines related to tags.

### 9.138.1 Detailed Description

Namespace of image processing routines related to tags.

## 9.139 mln::test Namespace Reference

Namespace of image processing routines related to pixel tests.

### Namespaces

- namespace [impl](#)  
*Implementation namespace of test namespace.*

### Functions

- `template<typename I>`  
`bool positive (const Image< I > &input)`  
*Test if an image only contains positive values.*
- `template<typename S, typename F>`  
`bool predicate (const Site_Set< S > &pset, const Function_v2b< F > &f)`  
*Test if all points of pset verify the predicate f.*

- `template<typename I, typename J, typename F >`  
`bool predicate (const Image< I > &lhs, const Image< J > &rhs, const Function_vv2b< F > &f)`  
*Test if all pixel values of lhs and rhs verify the predicate f.*
- `template<typename I, typename F >`  
`bool predicate (const Image< I > &ima, const Function_v2b< F > &f)`  
*Test if all pixel values of ima verify the predicate f.*

### 9.139.1 Detailed Description

Namespace of image processing routines related to pixel tests.

### 9.139.2 Function Documentation

#### 9.139.2.1 `template<typename I > bool mln::test::positive ( const Image< I > & input )` `[inline]`

Test if an image only contains positive values.

Definition at line 54 of file positive.hh.

References `predicate()`, and `mln::literal::zero`.

Referenced by `mln::morpho::gradient()`, `mln::morpho::gradient_external()`, `mln::morpho::gradient_internal()`, `mln::morpho::top_hat_black()`, `mln::morpho::elementary::top_hat_black()`, `mln::morpho::top_hat_self_complementary()`, `mln::morpho::top_hat_white()`, and `mln::morpho::elementary::top_hat_white()`.

#### 9.139.2.2 `template<typename S, typename F > bool mln::test::predicate ( const Site_Set< S > & pset, const Function_v2b< F > & f )` `[inline]`

Test if all points of `pset` verify the predicate `f`.

#### Parameters

[in] *pset* The point set.

[in] *f* The predicate.

Definition at line 242 of file predicate.hh.

#### 9.139.2.3 `template<typename I, typename J, typename F > bool mln::test::predicate ( const Image< I > & lhs, const Image< J > & rhs, const Function_vv2b< F > & f )` `[inline]`

Test if all pixel values of `lhs` and `rhs` verify the predicate `f`.

#### Parameters

[in] *lhs* The image.

[in] *rhs* The image.

[in] *f* The predicate.

Definition at line 222 of file predicate.hh.

**9.139.2.4** `template<typename I, typename F > bool mln::test::predicate ( const Image< I > & ima, const Function_v2b< F > & f ) [inline]`

Test if all pixel values of *ima* verify the predicate *f*.

#### Parameters

[in] *ima* The image.

[in] *f* The predicate.

Definition at line 207 of file predicate.hh.

Referenced by `mln::operator<()`, `mln::operator<=()`, `mln::operator==()`, and `positive()`.

## 9.140 mln::test::impl Namespace Reference

Implementation namespace of test namespace.

### 9.140.1 Detailed Description

Implementation namespace of test namespace.

## 9.141 mln::topo Namespace Reference

Namespace of "point-wise" expression tools.

### Classes

- class [adj\\_higher\\_dim\\_connected\\_n\\_face\\_bkd\\_iter](#)  
*Backward iterator on all the n-faces sharing an adjacent (n+1)-face with a (reference) n-face of an mln::complex<D>.*
- class [adj\\_higher\\_dim\\_connected\\_n\\_face\\_fwd\\_iter](#)  
*Forward iterator on all the n-faces sharing an adjacent (n+1)-face with a (reference) n-face of an mln::complex<D>.*
- class [adj\\_higher\\_face\\_bkd\\_iter](#)  
*Backward iterator on all the adjacent (n+1)-faces of the n-face of an mln::complex<D>.*
- class [adj\\_higher\\_face\\_fwd\\_iter](#)  
*Forward iterator on all the adjacent (n+1)-faces of the n-face of an mln::complex<D>.*
- class [adj\\_lower\\_dim\\_connected\\_n\\_face\\_bkd\\_iter](#)

*Backward iterator on all the  $n$ -faces sharing an adjacent  $(n-1)$ -face with a (reference)  $n$ -face of an `mln::complex<D>`.*

- class [adj\\_lower\\_dim\\_connected\\_n\\_face\\_fwd\\_iter](#)  
*Forward iterator on all the  $n$ -faces sharing an adjacent  $(n-1)$ -face with a (reference)  $n$ -face of an `mln::complex<D>`.*
- class [adj\\_lower\\_face\\_bkd\\_iter](#)  
*Backward iterator on all the adjacent  $(n-1)$ -faces of the  $n$ -face of an `mln::complex<D>`.*
- class [adj\\_lower\\_face\\_fwd\\_iter](#)  
*Forward iterator on all the adjacent  $(n-1)$ -faces of the  $n$ -face of an `mln::complex<D>`.*
- class [adj\\_lower\\_higher\\_face\\_bkd\\_iter](#)  
*Forward iterator on all the adjacent  $(n-1)$ -faces and  $(n+1)$ -faces of the  $n$ -face of an `mln::complex<D>`.*
- class [adj\\_lower\\_higher\\_face\\_fwd\\_iter](#)  
*Forward iterator on all the adjacent  $(n-1)$ -faces and  $(n+1)$ -faces of the  $n$ -face of an `mln::complex<D>`.*
- class [adj\\_m\\_face\\_bkd\\_iter](#)  
*Backward iterator on all the  $m$ -faces transitively adjacent to a (reference)  $n$ -face in a complex.*
- class [adj\\_m\\_face\\_fwd\\_iter](#)  
*Forward iterator on all the  $m$ -faces transitively adjacent to a (reference)  $n$ -face in a complex.*
- class [algebraic\\_face](#)  
*Algebraic face handle in a complex; the face dimension is dynamic.*
- class [algebraic\\_n\\_face](#)  
*Algebraic  $N$ -face handle in a complex.*
- class [center\\_only\\_iter](#)  
*Iterator on all the adjacent  $(n-1)$ -faces of the  $n$ -face of an `mln::complex<D>`.*
- class [centered\\_bkd\\_iter\\_adapter](#)  
*Forward complex relative iterator adapters adding the central (reference) point to the set of iterated faces.*
- class [centered\\_fwd\\_iter\\_adapter](#)  
*Backward complex relative iterator adapters adding the central (reference) point to the set of iterated faces.*
- class [complex](#)  
*General complex of dimension  $D$ .*
- class [face](#)  
*Face handle in a complex; the face dimension is dynamic.*
- class [face\\_bkd\\_iter](#)  
*Backward iterator on all the faces of an `mln::complex<D>`.*
- class [face\\_fwd\\_iter](#)

*Forward iterator on all the faces of an `mln::complex<D>`.*

- struct [is\\_n\\_face](#)  
*A functor testing wheter a `mln::complex_site` is an  $N$ -face.*
- struct [is\\_simple\\_2d\\_t](#)  
*Test if a point is simple or not.*
- class [is\\_simple\\_cell](#)  
*A predicate for the simplicity of a point based on the collapse property of the attachment.*
- class [n\\_face](#)  
 *$N$ -face handle in a complex.*
- class [n\\_face\\_bkd\\_iter](#)  
*Backward iterator on all the faces of an `mln::complex<D>`.*
- class [n\\_face\\_fwd\\_iter](#)  
*Forward iterator on all the faces of an `mln::complex<D>`.*
- class [n\\_faces\\_set](#)  
*Set of face handles of dimension  $N$ .*
- class [static\\_n\\_face\\_bkd\\_iter](#)  
*Backward iterator on all the  $N$ -faces of a `mln::complex<D>`.*
- class [static\\_n\\_face\\_fwd\\_iter](#)  
*Forward iterator on all the  $N$ -faces of a `mln::complex<D>`.*

## Functions

- `template<unsigned D, typename G >`  
`void detach (const complex\_site< D, G > &f, complex\_image< D, G, bool > &ima)`  
*Detach the cell corresponding to  $f$  from  $ima$ .*
- `template<unsigned D, typename G >`  
`bool is_facet (const complex\_site< D, G > &f)`  
*Is  $f$  a facet, i.e., a face not “included in” (adjacent to) a face of higher dimension?*
- `template<unsigned D>`  
`algebraic\_face< D > make\_algebraic\_face (const face< D > &f, bool sign)`  
*Create an algebraic face handle of a  $D$ -complex.*
- `template<unsigned D>`  
`std::ostream & operator<< (std::ostream &ostr, const face< D > &f)`  
*Print an `mln::topo::face`.*
- `template<unsigned N, unsigned D>`  
`std::ostream & operator<< (std::ostream &ostr, const n\_face< N, D > &f)`

Print an *mln::topo::n\_face*.

- template<unsigned D>  
std::ostream & **operator<<** (std::ostream &ostr, const **complex**< D > &c)  
*Pretty print a complex.*
- template<unsigned D>  
std::ostream & **operator<<** (std::ostream &ostr, const **algebraic\_face**< D > &f)  
*Print an mln::topo::algebraic\_face.*
- template<unsigned N, unsigned D>  
std::ostream & **operator<<** (std::ostream &ostr, const **algebraic\_n\_face**< N, D > &f)  
*Print an mln::topo::algebraic\_n\_face.*
- template<unsigned D>  
bool **operator==** (const **complex**< D > &lhs, const **complex**< D > &rhs)  
*Compare two complexes for equality.*
- template<unsigned D>  
**algebraic\_face**< D > **operator-** (const **face**< D > &f)  
*Inversion operators.*
- template<unsigned D>  
bool **operator==** (const **algebraic\_face**< D > &lhs, const **algebraic\_face**< D > &rhs)  
*Comparison of two instances of mln::topo::algebraic\_face.*
- template<unsigned D>  
bool **operator!=** (const **algebraic\_face**< D > &lhs, const **algebraic\_face**< D > &rhs)  
*Is lhs different from rhs?*
- template<unsigned D>  
bool **operator<** (const **algebraic\_face**< D > &lhs, const **algebraic\_face**< D > &rhs)  
*Is lhs "less" than rhs?*
- template<unsigned N, unsigned D>  
**algebraic\_n\_face**< N, D > **operator-** (const **n\_face**< N, D > &f)  
*Inversion operators.*
- template<unsigned N, unsigned D>  
bool **operator==** (const **algebraic\_n\_face**< N, D > &lhs, const **algebraic\_n\_face**< N, D > &rhs)  
*Comparison of two instances of mln::topo::algebraic\_n\_face.*
- template<unsigned N, unsigned D>  
bool **operator!=** (const **algebraic\_n\_face**< N, D > &lhs, const **algebraic\_n\_face**< N, D > &rhs)  
*Is lhs different from rhs?*
- template<unsigned N, unsigned D>  
bool **operator<** (const **algebraic\_n\_face**< N, D > &lhs, const **algebraic\_n\_face**< N, D > &rhs)

*Is lhs "less" than rhs?*

- `template<unsigned D>`  
`algebraic_n_face< 1, D > edge (const n_face< 0, D > &f1, const n_face< 0, D > &f2)`  
*Helpers.*
  
- `template<unsigned D>`  
`bool operator== (const face< D > &lhs, const face< D > &rhs)`  
*Comparison of two instances of `mln::topo::face`.*
  
- `template<unsigned D>`  
`bool operator!= (const face< D > &lhs, const face< D > &rhs)`  
*Is lhs different from rhs?*
  
- `template<unsigned D>`  
`bool operator< (const face< D > &lhs, const face< D > &rhs)`  
*Is lhs "less" than rhs?*
  
  
- `template<unsigned N, unsigned D>`  
`bool operator== (const n_face< N, D > &lhs, const n_face< N, D > &rhs)`  
*Comparison of two instances of `mln::topo::n_face`.*
  
- `template<unsigned N, unsigned D>`  
`bool operator!= (const n_face< N, D > &lhs, const n_face< N, D > &rhs)`  
*Is lhs different from rhs?*
  
- `template<unsigned N, unsigned D>`  
`bool operator< (const n_face< N, D > &lhs, const n_face< N, D > &rhs)`  
*Is lhs "less" than rhs?*
  
  
- `template<unsigned N, unsigned D>`  
`n_faces_set< N, D > operator+ (const algebraic_n_face< N, D > &f1, const algebraic_n_face< N, D > &f2)`  
*Addition.*
  
- `template<unsigned N, unsigned D>`  
`n_faces_set< N, D > operator- (const algebraic_n_face< N, D > &f1, const algebraic_n_face< N, D > &f2)`  
*Subtraction.*

### 9.141.1 Detailed Description

Namespace of "point-wise" expression tools.



## 9.141.2 Function Documentation

**9.141.2.1** `template<unsigned D, typename G > void mln::topo::detach ( const complex_psite< D, G > & f, complex_image< D, G, bool > & ima ) [inline]`

Detach the cell corresponding to *f* from *ima*.

### Precondition

*f* is a facet (it does not belong to any face of higher dimension).  
*ima* is an image of Boolean values.

Definition at line 58 of file detach.hh.

References mln::make::detachment(), mln::data::fill(), and is\_facet().

**9.141.2.2** `template<unsigned D> algebraic_n_face< 1, D > mln::topo::edge ( const n_face< 0, D > & f1, const n_face< 0, D > & f2 )`

Helpers.

Return the algebraic 1-face (edge) linking the 0-faces (vertices) *f1* and *f2*. If there is no 1-face between *f1* and *f2*, return an invalid 1-face.

### Precondition

*f1* and *f2* must belong to the same complex.

Note: this routine assumes the complex is not degenerated, i.e.,

- it does not check that *f1* and *f2* are the only 0-faces adjacent to an hypothetical 1-face; it just checks that *f1* and *f2* share a common 1-face;
- if there are several adjacent 1-faces shared by *f1* and *f2* (if the complex is ill-formed), there is no guarantee on the returned 1-face (the current implementation return the first 1-face found, but client code should not rely on this implementation-defined behavior).

Definition at line 286 of file algebraic\_n\_face.hh.

References mln::topo::n\_face< N, D >::higher\_dim\_adj\_faces().

**9.141.2.3** `template<unsigned D, typename G > bool mln::topo::is_facet ( const complex_psite< D, G > & f ) [inline]`

Is *f* a facet, i.e., a face not “included in” (adjacent to) a face of higher dimension?

Definition at line 58 of file is\_facet.hh.

Referenced by mln::make::attachment(), mln::make::cell(), detach(), and mln::make::detachment().

**9.141.2.4** `template<unsigned D> algebraic_face< D > mln::topo::make_algebraic_face ( const face< D > & f, bool sign )`

Create an algebraic face handle of a *D-complex*.

Definition at line 211 of file algebraic\_face.hh.

**9.141.2.5** `template<unsigned D> bool mln::topo::operator!=( const algebraic_face< D > & lhs, const algebraic_face< D > & rhs ) [inline]`

Is *lhs* different from *rhs*?

#### Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

Definition at line 250 of file algebraic\_face.hh.

References mln::topo::face< D >::cplx().

**9.141.2.6** `template<unsigned D> bool mln::topo::operator!=( const face< D > & lhs, const face< D > & rhs ) [inline]`

Is *lhs* different from *rhs*?

#### Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

Definition at line 394 of file face.hh.

References mln::topo::face< D >::cplx().

**9.141.2.7** `template<unsigned N, unsigned D> bool mln::topo::operator!=( const algebraic_n_face< N, D > & lhs, const algebraic_n_face< N, D > & rhs ) [inline]`

Is *lhs* different from *rhs*?

#### Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

Definition at line 250 of file algebraic\_n\_face.hh.

References mln::topo::n\_face< N, D >::cplx().

**9.141.2.8** `template<unsigned N, unsigned D> bool mln::topo::operator!=( const n_face< N, D > & lhs, const n_face< N, D > & rhs ) [inline]`

Is *lhs* different from *rhs*?

#### Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

Definition at line 291 of file n\_face.hh.

References mln::topo::n\_face< N, D >::cplx().

**9.141.2.9** `template<unsigned N, unsigned D> n_faces_set< N, D > mln::topo::operator+ ( const algebraic_n_face< N, D > & f1, const algebraic_n_face< N, D > & f2 ) [inline]`

Addition.

Definition at line 206 of file n\_faces\_set.hh.

References `mln::topo::n_faces_set< N, D >::add()`.

**9.141.2.10** `template<unsigned N, unsigned D> algebraic_n_face< N, D > mln::topo::operator- ( const n_face< N, D > & f )`

Inversion operators.

Definition at line 221 of file algebraic\_n\_face.hh.

**9.141.2.11** `template<unsigned N, unsigned D> n_faces_set< N, D > mln::topo::operator- ( const algebraic_n_face< N, D > & f1, const algebraic_n_face< N, D > & f2 ) [inline]`

Subtraction.

Definition at line 284 of file n\_faces\_set.hh.

References `mln::topo::n_faces_set< N, D >::add()`.

**9.141.2.12** `template<unsigned D> algebraic_face< D > mln::topo::operator- ( const face< D > & f )`

Inversion operators.

Definition at line 219 of file algebraic\_face.hh.

**9.141.2.13** `template<unsigned N, unsigned D> bool mln::topo::operator< ( const n_face< N, D > & lhs, const n_face< N, D > & rhs ) [inline]`

Is *lhs* “less” than *rhs*?

This comparison is required by algorithms sorting face handles.

#### Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

Definition at line 301 of file n\_face.hh.

**9.141.2.14** `template<unsigned D> bool mln::topo::operator< ( const face< D > & lhs, const face< D > & rhs ) [inline]`

Is *lhs* “less” than *rhs*?

This comparison is required by algorithms sorting face handles.

#### Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

Arguments *lhs* and *rhs* must have the same dimension.

Definition at line 404 of file face.hh.

**9.141.2.15** `template<unsigned D> bool mln::topo::operator< ( const algebraic_face< D > & lhs, const algebraic_face< D > & rhs ) [inline]`

Is *lhs* “less” than *rhs*?

This comparison is required by algorithms sorting algebraic face handles.

#### Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).  
Arguments *lhs* and *rhs* must have the same dimension.

Definition at line 260 of file algebraic\_face.hh.

**9.141.2.16** `template<unsigned N, unsigned D> bool mln::topo::operator< ( const algebraic_n_face< N, D > & lhs, const algebraic_n_face< N, D > & rhs ) [inline]`

Is *lhs* “less” than *rhs*?

This comparison is required by algorithms sorting algebraic face handles.

#### Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

Definition at line 261 of file algebraic\_n\_face.hh.

**9.141.2.17** `template<unsigned N, unsigned D> std::ostream & mln::topo::operator<< ( std::ostream & ostr, const algebraic_n_face< N, D > & f ) [inline]`

Print an [mln::topo::algebraic\\_n\\_face](#).

Definition at line 273 of file algebraic\_n\_face.hh.

**9.141.2.18** `template<unsigned D> std::ostream & mln::topo::operator<< ( std::ostream & ostr, const face< D > & f ) [inline]`

Print an [mln::topo::face](#).

Definition at line 416 of file face.hh.

**9.141.2.19** `template<unsigned N, unsigned D> std::ostream & mln::topo::operator<< ( std::ostream & ostr, const n_face< N, D > & f ) [inline]`

Print an [mln::topo::n\\_face](#).

Definition at line 312 of file n\_face.hh.

**9.141.2.20** `template<unsigned D> std::ostream & mln::topo::operator<< ( std::ostream & ostr, const complex< D > & c ) [inline]`

Pretty print a complex.

Definition at line 670 of file complex.hh.

References `mln::topo::complex< D >::print()`.

**9.141.2.21** `template<unsigned D> std::ostream & mln::topo::operator<< ( std::ostream & ostr, const algebraic_face< D > & f ) [inline]`

Print an `mln::topo::algebraic_face`.

Definition at line 273 of file algebraic\_face.hh.

**9.141.2.22** `template<unsigned N, unsigned D> bool mln::topo::operator==( const algebraic_n_face< N, D > & lhs, const algebraic_n_face< N, D > & rhs ) [inline]`

Comparison of two instances of `mln::topo::algebraic_n_face`.

Is *lhs* equal to *rhs*?

#### Precondition

Arguments *lhs* and *rhs* must belong to the same `mln::topo::complex`.

Definition at line 239 of file algebraic\_n\_face.hh.

References `mln::topo::n_face< N, D >::cplx()`, `mln::topo::n_face< N, D >::face_id()`, and `mln::topo::algebraic_n_face< N, D >::sign()`.

**9.141.2.23** `template<unsigned D> bool mln::topo::operator==( const complex< D > & lhs, const complex< D > & rhs ) [inline]`

Compare two complexes for equality.

Definition at line 657 of file complex.hh.

**9.141.2.24** `template<unsigned D> bool mln::topo::operator==( const face< D > & lhs, const face< D > & rhs ) [inline]`

Comparison of two instances of `mln::topo::face`.

Is *lhs* equal to *rhs*?

#### Precondition

Arguments *lhs* and *rhs* must belong to the same `mln::topo::complex`.

Definition at line 384 of file face.hh.

References `mln::topo::face< D >::cplx()`, `mln::topo::face< D >::face_id()`, and `mln::topo::face< D >::n()`.

**9.141.2.25** `template<unsigned D> bool mln::topo::operator==( const algebraic_face< D > & lhs, const algebraic_face< D > & rhs ) [inline]`

Comparison of two instances of [mln::topo::algebraic\\_face](#).

Is *lhs* equal to *rhs*?

#### Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

Definition at line 237 of file `algebraic_face.hh`.

References `mln::topo::face< D >::cplx()`, `mln::topo::face< D >::face_id()`, `mln::topo::face< D >::n()`, and `mln::topo::algebraic_face< D >::sign()`.

**9.141.2.26** `template<unsigned N, unsigned D> bool mln::topo::operator==( const n_face< N, D > & lhs, const n_face< N, D > & rhs ) [inline]`

Comparison of two instances of [mln::topo::n\\_face](#).

Is *lhs* equal to *rhs*?

#### Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

Definition at line 281 of file `n_face.hh`.

References `mln::topo::n_face< N, D >::cplx()`, and `mln::topo::n_face< N, D >::face_id()`.

## 9.142 mln::trace Namespace Reference

Namespace of routines related to the trace mechanism.

### 9.142.1 Detailed Description

Namespace of routines related to the trace mechanism.

## 9.143 mln::trait Namespace Reference

Namespace where traits are defined.

### 9.143.1 Detailed Description

Namespace where traits are defined. Namespace for image traits.

## 9.144 mln::transform Namespace Reference

Namespace of transforms.

## Functions

- `template<typename I, typename N, typename D >`  
`util::couple< mln::trait::ch_value< I, D >::ret, mln::trait::ch_value< I, typename I::psite >::ret >`  
`distance_and_closest_point_geodesic` (const `Image< I >` &input, const `Neighborhood< N >` &nbh,  
 D max)  
*Discrete geodesic distance transform.*
- `template<typename P, typename N, typename D >`  
`util::couple< mln_image_from_grid(mln_grid(P), D), mln_image_from_grid(mln_grid(P),`  
`unsigned)> distance_and_closest_point_geodesic` (const `p_array< P >` &pset, const `box< P`  
`>` &closest\_point\_domain, const `Neighborhood< N >` &nbh, D max)  
*Discrete geodesic distance transform.*
- `template<typename I, typename N, typename D >`  
`util::couple< mln::trait::ch_value< I, D >::ret, I >` `distance_and_influence_zone_geodesic` (const  
`Image< I >` &input, const `Neighborhood< N >` &nbh, D max)  
*Discrete geodesic distance transform.*
- `template<typename I, typename N, typename W, typename D >`  
`mln::trait::ch_value< I, D >::ret` `distance_front` (const `Image< I >` &input, const `Neighborhood<`  
`N >` &nbh, const `Weighted_Window< W >` &w\_win, D max)  
*Discrete front distance transform.*
- `template<typename I, typename N, typename D >`  
`mln::trait::ch_value< I, D >::ret` `distance_geodesic` (const `Image< I >` &input, const `Neighbor-`  
`hood< N >` &nbh, D max)  
*Discrete geodesic distance transform.*
- `template<typename I >`  
`image2d< float >` `hough` (const `Image< I >` &input\_)  
*Compute the hough transform from a binary image.*
- `template<typename I, typename N, typename W >`  
`mln::trait::concrete< I >::ret` `influence_zone_front` (const `Image< I >` &input, const `Neighbor-`  
`hood< N >` &nbh, const `Weighted_Window< W >` &w\_win)  
*Influence zone transform.*
- `template<typename I, typename N, typename W, typename D >`  
`mln::trait::concrete< I >::ret` `influence_zone_front` (const `Image< I >` &input, const `Neighbor-`  
`hood< N >` &nbh, const `Weighted_Window< W >` &w\_win, D max)  
*Influence zone transform.*
- `template<typename I, typename N >`  
`mln::trait::concrete< I >::ret` `influence_zone_geodesic` (const `Image< I >` &input, const `Neighbor-`  
`hood< N >` &nbh)  
*Geodesic influence zone transform.*
- `template<typename I, typename N, typename D >`  
`mln::trait::concrete< I >::ret` `influence_zone_geodesic_saturated` (const `Image< I >` &input, const  
`Neighborhood< N >` &nbh, const D &max, const typename I::value &background\_value)  
*Geodesic influence zone transform.*

- `template<typename I , typename N , typename D >`  
`mln::trait::concrete< I >::ret influence\_zone\_geodesic\_saturated (const Image< I > &input, const Neighborhood< N > &nbh, const D &max)`

### 9.144.1 Detailed Description

Namespace of transforms.

### 9.144.2 Function Documentation

- 9.144.2.1** `template<typename I , typename N , typename D > util::couple<`  
`mln::trait::ch_value< I, D >::ret, mln::trait::ch_value< I, typename I::psite >::ret >`  
`mln::transform::distance_and_closest_point_geodesic ( const Image< I > & input,`  
`const Neighborhood< N > & nbh, D max ) \[inline\]`

Discrete geodesic distance transform.

#### Parameters

- `[in] input` [Image](#) from which the geodesic distance is computed.
- `[in] nbh` [Neighborhood](#)
- `[in] max` Max distance of propagation.

#### Returns

a couple of images. The first one is the distance map and the second one is the closest point image. The closest point image contains sites.

#### Postcondition

The returned images have the same domain as `input`.

Definition at line 90 of file `distance_and_closest_point_geodesic.hh`.

References `mln::make::couple()`, and `distance_geodesic()`.

- 9.144.2.2** `template<typename P , typename N , typename D > util::couple<`  
`mln_image_from_grid(mln_grid(P), D), mln_image_from_grid(mln_grid(P),`  
`unsigned)> mln::transform::distance_and_closest_point_geodesic ( const p\_array< P`  
`> & pset, const box< P > & closest_point_domain, const Neighborhood< N > & nbh,`  
`D max ) \[inline\]`

Discrete geodesic distance transform.

#### Parameters

- `[in] pset` an array of sites.
- `[in] closest_point_domain` domain of the returned image.
- `[in] nbh` neighborhood
- `[in] max` max distance of propagation.



**Returns**

A couple of images. The first one is the distance map and the second one is the closest point image. The closest point image contains site indexes.

**Postcondition**

The returned image domains are defined on `closest_point_domain`.

Definition at line 110 of file `distance_and_closest_point_geodesic.hh`.

References `mln::geom::bbox()`, `mln::make::couple()`, `distance_geodesic()`, `mln::data::fill()`, and `mln::box<P>::is_valid()`.

**9.144.2.3** `template<typename I, typename N, typename D > util::couple< mln::trait::ch_value< I, D >::ret, I > mln::transform::distance_and_influence_zone_geodesic ( const Image< I > & input, const Neighborhood< N > & nbh, D max ) [inline]`

Discrete geodesic distance transform.

**Parameters**

[in] *input* Image from which the geodesic distance is computed.

[in] *nbh* Neighborhood

[in] *max* Max distance of propagation.

**Returns**

a couple of images. The first one is the distance map and the second one is the closest point image. The closest point image contains sites.

**Postcondition**

The returned images have the same domain as `input`.

Definition at line 69 of file `distance_and_influence_zone_geodesic.hh`.

References `mln::make::couple()`, and `distance_geodesic()`.

**9.144.2.4** `template<typename I, typename N, typename W, typename D > mln::trait::ch_value< I, D >::ret mln::transform::distance_front ( const Image< I > & input, const Neighborhood< N > & nbh, const Weighted_Window< W > & w_win, D max ) [inline]`

Discrete front distance transform.

Definition at line 56 of file `transform/distance_front.hh`.

**9.144.2.5** `template<typename I, typename N, typename D > mln::trait::ch_value< I, D >::ret mln::transform::distance_geodesic ( const Image< I > & input, const Neighborhood< N > & nbh, D max ) [inline]`

Discrete geodesic distance transform.

Definition at line 55 of file `transform/distance_geodesic.hh`.

Referenced by `distance_and_closest_point_geodesic()`, and `distance_and_influence_zone_geodesic()`.

### 9.144.2.6 `template<typename I > image2d< float > mln::transform::hough ( const Image< I > & input_ )`

Compute the hough transform from a binary image.

Objects used for computation must be set to 'true'.

#### Parameters

[in] *input\_* A binary image.

#### Returns

A 2D image of float. Rows are used for the distance and columns are used for the angles. Angles go from 0 to 359. Distance goes from 0 to the maximum distance between the center and a corner. The site having the maximum value indicates through its column index the document inclination.

Definition at line 98 of file `hough.hh`.

References `mln::opt::at()`, `mln::data::fill()`, `mln::geom::min_col()`, `mln::geom::min_row()`, `mln::geom::ncols()`, and `mln::geom::nrows()`.

### 9.144.2.7 `template<typename I , typename N , typename W > mln::trait::concrete< I >::ret mln::transform::influence_zone_front ( const Image< I > & input, const Neighborhood< N > & nbh, const Weighted_Window< W > & w_win )`

Influence zone transform.

Definition at line 78 of file `influence_zone_front.hh`.

References `influence_zone_front()`.

### 9.144.2.8 `template<typename I , typename N , typename W , typename D > mln::trait::concrete< I >::ret mln::transform::influence_zone_front ( const Image< I > & input, const Neighborhood< N > & nbh, const Weighted_Window< W > & w_win, D max )`

Influence zone transform.

Definition at line 60 of file `influence_zone_front.hh`.

References `mln::canvas::distance_front()`.

Referenced by `influence_zone_front()`.

### 9.144.2.9 `template<typename I , typename N > mln::trait::concrete< I >::ret mln::transform::influence_zone_geodesic ( const Image< I > & input, const Neighborhood< N > & nbh )`

Geodesic influence zone transform.

#### Parameters

[in] *input* An image.

[in] *nbh* A neighborhood.

#### Returns

An image of influence zone.

Definition at line 216 of file influence\_zone\_geodesic.hh.

**9.144.2.10** `template<typename I , typename N , typename D > mln::trait::concrete< I >::ret  
mln::transform::influence_zone_geodesic_saturated ( const Image< I > & input,  
const Neighborhood< N > & nbh, const D & max, const typename I::value &  
background_value )`

Geodesic influence zone transform.

#### Parameters

- [in] *input* An image.
- [in] *nbh* A neighborhood.
- [in] *max* The maximum influence zone distance.
- [in] *background\_value* The value used as background (i.e. not propagated).

#### Returns

An image of influence zone.

Definition at line 73 of file influence\_zone\_geodesic\_saturated.hh.

References `mln::canvas::distance_geodesic()`.

Referenced by `influence_zone_geodesic_saturated()`.

**9.144.2.11** `template<typename I , typename N , typename D > mln::trait::concrete< I >::ret  
mln::transform::influence_zone_geodesic_saturated ( const Image< I > & input,  
const Neighborhood< N > & nbh, const D & max )`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 93 of file influence\_zone\_geodesic\_saturated.hh.

References `influence_zone_geodesic_saturated()`, and `mln::literal::zero`.

## 9.145 mln::util Namespace Reference

Namespace of tools using for more complex algorithm.

### Namespaces

- namespace [impl](#)  
*Implementation namespace of util namespace.*

### Classes

- class [adjacency\\_matrix](#)  
*A class of adjacency matrix.*

- class [array](#)  
*A dynamic array class.*
- class [branch](#)  
*Class of generic branch.*
- class [branch\\_iter](#)  
*Basic 2D image class.*
- class [branch\\_iter\\_ind](#)  
*Basic 2D image class.*
- class [couple](#)  
*Definition of a couple.*
- struct [eat](#)  
*Eat structure.*
- class [edge](#)  
*Edge of a graph  $G$ .*
- class [fibonacci\\_heap](#)  
*Fibonacci heap.*
- class [graph](#)  
*Undirected graph.*
- class [greater\\_point](#)  
*A “greater than” functor comparing points w.r.t.*
- class [greater\\_psite](#)  
*A “greater than” functor comparing psites w.r.t.*
- class [head](#)  
*Top structure of the soft heap.*
- struct [ignore](#)  
*Ignore structure.*
- struct [ilcell](#)  
*Element of an item list. Store the data (key) used in [soft\\_heap](#).*
- class [line\\_graph](#)  
*Undirected line graph of a graph of type  $G$ .*
- struct [nil](#)  
*Nil structure.*
- class [node](#)

*Meta-data of an element in the heap.*

- class [object\\_id](#)  
*Base class of an object id.*
- struct [ord](#)  
*Function-object that defines an ordering between objects with type  $T$ : lhs R rhs.*
- struct [ord\\_pair](#)  
*Ordered pair structure s.a.*
- struct [pix](#)  
*Structure pix.*
- class [set](#)  
*An "efficient" mathematical set class.*
- class [site\\_pair](#)  
*A pair of sites.*
- class [soft\\_heap](#)  
*Soft heap.*
- class [timer](#)  
*Timer structure.*
- struct [tracked\\_ptr](#)  
*Smart pointer for shared data with tracking.*
- class [tree](#)  
*Class of generic tree.*
- class [tree\\_node](#)  
*Class of generic [tree\\_node](#) for tree.*
- class [vertex](#)  
*Vertex of a graph  $G$ .*
- struct [yes](#)  
*Object that always says "yes".*

## Typedefs

- typedef [object\\_id](#)< [vertex\\_tag](#), unsigned > [vertex\\_id\\_t](#)  
*Vertex id type.*

## Functions

- `template<typename I, typename J >`  
`void display_branch (const Image< J > &ima_, tree_node< I > *tree_node)`  
*Display an arborescence from `tree_node`.*
- `template<typename I, typename J >`  
`void display_tree (const Image< J > &ima_, tree< I > &tree)`  
*Display a tree.*
- `template<typename I >`  
`I::psite lemmings (const Image< I > &ima, const typename I::psite &pt, const typename I::psite::delta &dpt, const typename I::value &val)`  
*Launch a lemmings on an image.*
- `template<typename I >`  
`greater_point< I > make_greater_point (const Image< I > &ima)`  
*Helper to build a `mln::util::greater_point`.*
- `template<typename I >`  
`greater_psite< I > make_greater_psite (const Image< I > &ima)`  
*Helper to build a `mln::util::greater_psite`.*
- `template<typename G >`  
`bool operator< (const vertex< G > &lhs, const vertex< G > &rhs)`  
*Less operator. Test whether `lhs.id() < rhs.id()`.*
- `template<typename G >`  
`std::ostream & operator<< (std::ostream &ostr, const vertex< G > &v)`  
*Push the vertex `v` in the output stream `ostr`.*
- `template<typename T >`  
`std::ostream & operator<< (std::ostream &ostr, const array< T > &a)`  
*Operator<<.*
- `template<typename G >`  
`bool operator== (const vertex< G > &v1, const vertex< G > &v2)`  
*Equality operator.*
- `template<typename T >`  
`bool operator== (const array< T > &lhs, const array< T > &rhs)`  
*Operator==.*
- `template<typename T >`  
`bool ord_strict (const T &lhs, const T &rhs)`  
*Routine to test if `lhs` is strictly "less-than" `rhs`.*
- `template<typename T >`  
`bool ord_weak (const T &lhs, const T &rhs)`  
*Routine to test if `lhs` is "less-than or equal-to" `rhs`.*

- `template<typename T, typename I >`  
`void tree_fast_to_image (tree_fast< T > &tree, Image< I > &output_)`
- `template<typename T >`  
`tree_fast< T > tree_to_fast (tree< T > &input)`  
*Facade.*
- `template<typename T, typename I >`  
`void tree_to_image (tree< T > &tree, Image< I > &output_)`  
*Convert a tree into an image.*

### 9.145.1 Detailed Description

Namespace of tools using for more complex algorithm. Forward declaration.

### 9.145.2 Typedef Documentation

#### 9.145.2.1 typedef object\_id<vertex\_tag, unsigned> mln::util::vertex\_id\_t

Vertex id type.

Definition at line 43 of file graph\_ids.hh.

### 9.145.3 Function Documentation

#### 9.145.3.1 `template<typename I, typename J > void mln::util::display_branch ( const Image< J > & ima_, tree_node< I > * tree_node ) [inline]`

Display an arborescence from `tree_node`.

##### Parameters

- [in] `ima_` The domain of output image.
- [in] `tree_node` The root `tree_node` to display.

Definition at line 210 of file tree\_to\_image.hh.

References `mln::data::fill()`.

#### 9.145.3.2 `template<typename I, typename J > void mln::util::display_tree ( const Image< J > & ima_, tree< I > & tree ) [inline]`

Display a tree.

##### Parameters

- [in] `ima_` The domain of output image.
- [in] `tree` The tree to display.

Definition at line 192 of file tree\_to\_image.hh.

References `mln::util::tree< T >::root()`.

**9.145.3.3** `template<typename I> I::psite mln::util::lemmings ( const Image< I > & ima, const typename I::psite & pt, const typename I::psite::delta & dpt, const typename I::value & val )`

Launch a lemmings on an image.

A lemmings is the point `pt` that you put on an image `ima`. This point will move through the image using the delta-point `dpt` while consider his value on the given image.

#### Returns

The first point that is not in the domain `domain` or which value on the given image is different to the value `val`.

#### Precondition

The domain `domain` must be contained in the domain of `ima`.

Definition at line 104 of file `lemmings.hh`.

**9.145.3.4** `template<typename I> greater_point< I > mln::util::make_greater_point ( const Image< I > & ima )`

Helper to build a [mln::util::greater\\_point](#).

Definition at line 82 of file `greater_point.hh`.

**9.145.3.5** `template<typename I> greater_psite< I > mln::util::make_greater_psite ( const Image< I > & ima )`

Helper to build a [mln::util::greater\\_psite](#).

Definition at line 82 of file `greater_psite.hh`.

**9.145.3.6** `template<typename G> bool mln::util::operator< ( const vertex< G > & lhs, const vertex< G > & rhs ) [inline]`

Less operator. Test whether `lhs.id() < rhs.id()`.

Definition at line 390 of file `vertex.hh`.

**9.145.3.7** `template<typename G> std::ostream & mln::util::operator<< ( std::ostream & ostr, const vertex< G > & v ) [inline]`

Push the vertex `v` in the output stream `ostr`.

Definition at line 372 of file `vertex.hh`.

**9.145.3.8** `template<typename T> std::ostream & mln::util::operator<< ( std::ostream & ostr, const array< T > & a )`

Operator<<.

Definition at line 796 of file `util/array.hh`.

References `mln::util::array< T >::nelements()`.



**9.145.3.9** `template<typename G > bool mln::util::operator==( const vertex< G > & v1, const vertex< G > & v2 ) [inline]`

Equality operator.

Test whether two vertices have the same id.

Definition at line 380 of file vertex.hh.

References `mln::util::vertex< G >::graph()`, and `mln::util::vertex< G >::id()`.

**9.145.3.10** `template<typename T > bool mln::util::operator==( const array< T > & lhs, const array< T > & rhs )`

Operator==.

Definition at line 815 of file util/array.hh.

References `mln::util::array< T >::std_vector()`.

**9.145.3.11** `template<typename T > bool mln::util::ord_strict ( const T & lhs, const T & rhs ) [inline]`

Routine to test if *lhs* is strictly "less-than" *rhs*.

Definition at line 90 of file util/ord.hh.

Referenced by `mln::util::ord_pair< T >::change_both()`, `mln::util::ord_pair< T >::change_first()`, and `mln::util::ord_pair< T >::change_second()`.

**9.145.3.12** `template<typename T > bool mln::util::ord_weak ( const T & lhs, const T & rhs ) [inline]`

Routine to test if *lhs* is "less-than or equal-to" *rhs*.

Definition at line 101 of file util/ord.hh.

Referenced by `mln::util::ord_pair< T >::change_both()`, `mln::util::ord_pair< T >::change_first()`, `mln::util::ord_pair< T >::change_second()`, and `mln::box< P >::is_valid()`.

**9.145.3.13** `template<typename T, typename I > void mln::util::tree_fast_to_image ( tree_fast< T > & tree, Image< I > & output_ ) [inline]`

Convert a `tree_fast` into an image.

#### Parameters

[in] *tree* The tree to convert.

[out] *output\_* The image containing tree informations.

Definition at line 99 of file `tree_fast_to_image.hh`.

**9.145.3.14** `template<typename T > tree_fast< T > mln::util::tree_to_fast ( tree< T > & input ) [inline]`

Facade.

Convert a tree into an `tree_fast`.

#### Parameters

[in] *input* The tree to convert.

#### Returns

The `tree_fast` containing tree informations.

Definition at line 90 of file `tree_to_fast.hh`.

References `mln::util::tree< T >::root()`.

#### 9.145.3.15 `template<typename T , typename I > void mln::util::tree_to_image ( tree< T > & tree, Image< I > & output_ ) [inline]`

Convert a tree into an image.

#### Parameters

[in] *tree* The tree to convert.

[out] *output\_* The image containing tree information.

Definition at line 178 of file `tree_to_image.hh`.

## 9.146 `mln::util::impl` Namespace Reference

Implementation namespace of util namespace.

### 9.146.1 Detailed Description

Implementation namespace of util namespace.

## 9.147 `mln::value` Namespace Reference

Namespace of materials related to pixel value types.

### Namespaces

- namespace [impl](#)  
*Implementation namespace of value namespace.*

### Classes

- class [float01](#)  
*Class for floating values restricted to the interval [0..1] and discretized with n bits.*

- struct [float01\\_f](#)  
*Class for floating values restricted to the interval [0..1].*
- struct [graylevel](#)  
*General gray-level class on n bits.*
- struct [graylevel\\_f](#)  
*General gray-level class on n bits.*
- struct [int\\_s](#)  
*Signed integer value class.*
- struct [int\\_u](#)  
*Unsigned integer value class.*
- struct [int\\_u\\_sat](#)  
*Unsigned integer value class with saturation behavior.*
- struct [Integer](#)  
*Concept of integer.*
- struct [Integer< void >](#)  
*Category flag type.*
- struct [label](#)  
*Label value class.*
- struct [lut\\_vec](#)  
*Class that defines FIXME.*
- class [proxy](#)  
*Generic proxy class for an image pixel value.*
- struct [rgb](#)  
*Color class for red-green-blue where every component is n-bit encoded.*
- struct [set](#)  
*Class that defines the set of values of type T.*
- class [sign](#)  
*The sign class represents the value type composed by the set (-1, 0, 1) sign value type is a subset of the int value type.*
- struct [stack\\_image](#)  
*Stack image class.*
- struct [super\\_value< sign >](#)  
*Specializations:*

- struct [value\\_array](#)  
*Generic array class over indexed by a value set with type T.*

## Typedefs

- typedef [float01\\_< 16 > float01\\_16](#)  
*Alias for 16 bit [float01](#).*
- typedef [float01\\_< 8 > float01\\_8](#)  
*Alias for 8 bit [float01](#).*
- typedef [graylevel< 16 > gl16](#)  
*Alias for 16 bit graylevel.*
- typedef [graylevel< 8 > gl8](#)  
*Alias for 8 bit graylevel.*
- typedef [graylevel\\_f glf](#)  
*Alias for graylevels encoded by float.*
- typedef [int\\_s< 16 > int\\_s16](#)  
*Alias for signed 16-bit integers.*
- typedef [int\\_s< 32 > int\\_s32](#)  
*Alias for signed 32-bit integers.*
- typedef [int\\_s< 8 > int\\_s8](#)  
*Alias for signed 8-bit integers.*
- typedef [int\\_u< 12 > int\\_u12](#)  
*Alias for unsigned 12-bit integers.*
- typedef [int\\_u< 16 > int\\_u16](#)  
*Alias for unsigned 16-bit integers.*
- typedef [mln::value::int\\_u< 32 > int\\_u32](#)  
*Alias for unsigned 32-bit integers.*
- typedef [mln::value::int\\_u< 8 > int\\_u8](#)  
*Alias for unsigned 8-bit integers.*
- typedef [label< 16 > label\\_16](#)  
*Alias for 16-bit integers.*
- typedef [label< 32 > label\\_32](#)  
*Alias for 32-bit integers.*
- typedef [mln::value::label< 8 > label\\_8](#)

*Alias for 8-bit labels.*

- typedef `rgb< 16 > rgb16`  
*Color class for red-green-blue where every component is 16-bit encoded.*
- typedef `rgb< 8 > rgb8`  
*Color class for red-green-blue where every component is 8-bit encoded.*

## Functions

- template<typename Dest , typename Src >  
Dest `cast` (const Src &src)  
*Cast a value `src` from type `Src` to type `Dest`.*
- template<typename V >  
internal::equiv\_< V >::ret `equiv` (const mln::Value< V > &v)  
*Access to the equivalent value.*
- template<unsigned n>  
`rgb< n >::interop operator+` (const `rgb< n >` &lhs, const `rgb< n >` &rhs)  
*Addition.*
- template<typename H , typename S , typename L >  
`hsl_< H, S, L > operator+` (const `hsl_< H, S, L >` &lhs, const `hsl_< H, S, L >` &rhs)  
*Addition.*
- template<unsigned n>  
std::ostream & `operator<<` (std::ostream &ostr, const `label< n >` &l)  
*Print a label `l` into the output stream `ostr`.*
- template<unsigned n>  
std::ostream & `operator<<` (std::ostream &ostr, const `rgb< n >` &c)  
*Print an rgb `c` into the output stream `ostr`.*
- std::ostream & `operator<<` (std::ostream &ostr, const `graylevel_f` &g)  
*Op<<.*
- template<typename T >  
std::ostream & `operator<<` (std::ostream &ostr, const `scalar_< T >` &s)  
*Print a scalar `s` in an output stream `ostr`.*
- template<typename H , typename S , typename L >  
std::ostream & `operator<<` (std::ostream &ostr, const `hsl_< H, S, L >` &c)  
*Print an hsl `c` into the output stream `ostr`.*
- template<unsigned n>  
std::ostream & `operator<<` (std::ostream &ostr, const `graylevel< n >` &g)  
*Op<<.*

- `template<unsigned n>`  
`std::ostream & operator<<< (std::ostream &ostr, const float01_<n> &f)`  
*Op<<<.*
- `std::ostream & operator<<< (std::ostream &ostr, const sign &i)`  
*Print an signed integer i into the output stream ostr.*
- `template<unsigned n>`  
`std::ostream & operator<<< (std::ostream &ostr, const int_u<n> &i)`  
*Print an unsigned integer i into the output stream ostr.*
- `template<unsigned n>`  
`std::ostream & operator<<< (std::ostream &ostr, const int_s<n> &i)`  
*Print an signed integer i into the output stream ostr.*
- `template<unsigned n>`  
`std::ostream & operator<<< (std::ostream &ostr, const int_u_sat<n> &i)`  
*Print a saturated unsigned integer i into the output stream ostr.*
- `bool operator== (const sign &lhs, const sign &rhs)`  
*Comparison operator.*
- `template<typename V >`  
`V other (const V &val)`  
*Give an other value than val.*
- `template<typename H, typename S, typename L >`  
`hsl_<H, S, L> operator- (const hsl_<H, S, L> &lhs, const hsl_<H, S, L> &rhs)`  
*Subtraction.*
- `template<typename H, typename S, typename L, typename S2 >`  
`hsl_<H, S, L> operator* (const hsl_<H, S, L> &lhs, const mln::value::scalar_<S2> &s)`  
*Product.*
- `template<typename H, typename S, typename L, typename S2 >`  
`hsl_<H, S, L> operator/ (const hsl_<H, S, L> &lhs, const mln::value::scalar_<S2> &s)`  
*Division.*
- `template<typename H, typename S, typename L >`  
`bool operator== (const hsl_<H, S, L> &lhs, const hsl_<H, S, L> &rhs)`  
*Comparison.*
- `template<unsigned n>`  
`rgb<n>::interop operator- (const rgb<n> &lhs, const rgb<n> &rhs)`  
*Subtraction.*

- template<unsigned n, typename S >  
[rgb](#)< n >::interop [operator\\*](#) (const [rgb](#)< n > &lhs, const mln::value::scalar\_< S > &s)  
*Product.*
- template<unsigned n, typename S >  
[rgb](#)< n >::interop [operator/](#) (const [rgb](#)< n > &lhs, const mln::value::scalar\_< S > &s)  
*Division.*
- template<typename I >  
[stack\\_image](#)< 2, const I > [stack](#) (const [Image](#)< I > &ima1, const [Image](#)< I > &ima2)  
*Shortcut to build a stack with two images.*

### 9.147.1 Detailed Description

Namespace of materials related to pixel value types.

### 9.147.2 Typedef Documentation

#### 9.147.2.1 typedef float01\_<16> mln::value::float01\_16

Alias for 16 bit [float01](#).

Definition at line 45 of file float01\_16.hh.

#### 9.147.2.2 typedef float01\_<8> mln::value::float01\_8

Alias for 8 bit [float01](#).

Definition at line 45 of file float01\_8.hh.

#### 9.147.2.3 typedef graylevel<16> mln::value::gl16

Alias for 16 bit graylevel.

Definition at line 45 of file gl16.hh.

#### 9.147.2.4 typedef graylevel<8> mln::value::gl8

Alias for 8 bit graylevel.

Definition at line 45 of file gl8.hh.

#### 9.147.2.5 typedef graylevel\_f mln::value::glf

Alias for graylevels encoded by float.

Definition at line 44 of file glf.hh.

**9.147.2.6 typedef int\_s<16> mln::value::int\_s16**

Alias for signed 16-bit integers.

Definition at line 45 of file int\_s16.hh.

**9.147.2.7 typedef int\_s<32> mln::value::int\_s32**

Alias for signed 32-bit integers.

Definition at line 45 of file int\_s32.hh.

**9.147.2.8 typedef int\_s<8> mln::value::int\_s8**

Alias for signed 8-bit integers.

Definition at line 45 of file int\_s8.hh.

**9.147.2.9 typedef int\_u<12> mln::value::int\_u12**

Alias for unsigned 12-bit integers.

Definition at line 45 of file int\_u12.hh.

**9.147.2.10 typedef int\_u<16> mln::value::int\_u16**

Alias for unsigned 16-bit integers.

Definition at line 45 of file int\_u16.hh.

**9.147.2.11 typedef mln::value::int\_u<32> mln::value::int\_u32**

Alias for unsigned 32-bit integers.

Definition at line 45 of file int\_u32.hh.

**9.147.2.12 typedef mln::value::int\_u<8> mln::value::int\_u8**

Alias for unsigned 8-bit integers.

Definition at line 44 of file int\_u8.hh.

**9.147.2.13 typedef label<16> mln::value::label\_16**

Alias for 16-bit integers.

Definition at line 44 of file label\_16.hh.

**9.147.2.14 typedef label<32> mln::value::label\_32**

Alias for 32-bit integers.

Definition at line 44 of file label\_32.hh.



**9.147.2.15 typedef mln::value::label<8> mln::value::label\_8**

Alias for 8-bit labels.

Definition at line 44 of file label\_8.hh.

**9.147.2.16 typedef rgb<16> mln::value::rgb16**

Color class for red-green-blue where every component is 16-bit encoded.

Definition at line 45 of file rgb16.hh.

**9.147.2.17 typedef rgb<8> mln::value::rgb8**

Color class for red-green-blue where every component is 8-bit encoded.

Definition at line 45 of file rgb8.hh.

**9.147.3 Function Documentation****9.147.3.1 template<typename Dest , typename Src > Dest mln::value::cast ( const Src & src )  
[inline]**

Cast a value `src` from type `Src` to type `Dest`.

Definition at line 85 of file value/cast.hh.

**9.147.3.2 template<typename V > internal::equiv\_< V >::ret mln::value::equiv ( const  
mln::Value< V > & v ) [inline]**

Access to the equivalent value.

Definition at line 153 of file equiv.hh.

Referenced by `mln::labeling::superpose()`.

**9.147.3.3 template<unsigned n, typename S > rgb< n >::interop mln::value::operator\* ( const  
rgb< n > & lhs, const mln::value::scalar\_< S > & s ) [inline]**

Product.

Definition at line 717 of file value/rgb.hh.

**9.147.3.4 template<typename H , typename S , typename L , typename S2 > hsl\_< H, S, L >  
mln::value::operator\* ( const hsl\_< H, S, L > & lhs, const mln::value::scalar\_< S2 >  
& s )**

Product.

Definition at line 357 of file hsl.hh.

**9.147.3.5** `template<unsigned n> rgb< n >::interop mln::value::operator+ ( const rgb< n > & lhs, const rgb< n > & rhs ) [inline]`

Addition.

{

Definition at line 663 of file value/rgb.hh.

**9.147.3.6** `template<typename H , typename S , typename L > hsl_< H, S, L > mln::value::operator+ ( const hsl_< H, S, L > & lhs, const hsl_< H, S, L > & rhs )`

Addition.

{

Definition at line 337 of file hsl.hh.

**9.147.3.7** `template<unsigned n> rgb< n >::interop mln::value::operator- ( const rgb< n > & lhs, const rgb< n > & rhs ) [inline]`

Subtraction.

Definition at line 690 of file value/rgb.hh.

**9.147.3.8** `template<typename H , typename S , typename L > hsl_< H, S, L > mln::value::operator- ( const hsl_< H, S, L > & lhs, const hsl_< H, S, L > & rhs )`

Subtraction.

Definition at line 347 of file hsl.hh.

**9.147.3.9** `template<unsigned n, typename S > rgb< n >::interop mln::value::operator/ ( const rgb< n > & lhs, const mln::value::scalar_< S > & s ) [inline]`

Division.

Definition at line 735 of file value/rgb.hh.

**9.147.3.10** `template<typename H , typename S , typename L , typename S2 > hsl_< H, S, L > mln::value::operator/ ( const hsl_< H, S, L > & lhs, const mln::value::scalar_< S2 > & s )`

Division.

Definition at line 367 of file hsl.hh.

**9.147.3.11** `template<typename T > std::ostream & mln::value::operator<< ( std::ostream & ostr, const scalar_< T > & s ) [inline]`

Print a scalar `s` in an output stream `ostr`.

Definition at line 130 of file scalar.hh.

**9.147.3.12** `std::ostream & mln::value::operator<< ( std::ostream & ostr, const sign & i )`  
`[inline]`

Print an signed integer `i` into the output stream `ostr`.

**Parameters**

[in, out] `ostr` An output stream.

[in] `i` An sign value

**Returns**

The modified output stream `ostr`.

Definition at line 184 of file `value/sign.hh`.

References `mln::debug::format()`.

**9.147.3.13** `template<unsigned n> std::ostream & mln::value::operator<< ( std::ostream & ostr,`  
`const int_s<n> & i ) [inline]`

Print an signed integer `i` into the output stream `ostr`.

**Parameters**

[in, out] `ostr` An output stream.

[in] `i` An signed integer.

**Returns**

The modified output stream `ostr`.

Definition at line 260 of file `int_s.hh`.

References `mln::debug::format()`.

**9.147.3.14** `template<unsigned n> std::ostream & mln::value::operator<< ( std::ostream & ostr,`  
`const graylevel<n> & g ) [inline]`

`Op<<`.

Definition at line 591 of file `graylevel.hh`.

**9.147.3.15** `template<unsigned n> std::ostream & mln::value::operator<< ( std::ostream & ostr,`  
`const int_u<n> & i ) [inline]`

Print an unsigned integer `i` into the output stream `ostr`.

**Parameters**

[in, out] `ostr` An output stream.

[in] `i` An unsigned integer.

**Returns**

The modified output stream `ostr`.

Definition at line 357 of file `int_u.hh`.

References `mln::debug::format()`.

**9.147.3.16** `template<unsigned n> std::ostream & mln::value::operator<< ( std::ostream & ostr, const int_u_sat<n> & i ) [inline]`

Print a saturated unsigned integer `i` into the output stream `ostr`.

**Parameters**

[in, out] `ostr` An output stream.

[in] `i` A saturated unsigned integer.

**Returns**

The modified output stream `ostr`.

Definition at line 220 of file `int_u_sat.hh`.

References `mln::debug::format()`.

**9.147.3.17** `template<unsigned n> std::ostream & mln::value::operator<< ( std::ostream & ostr, const rgb<n> & c ) [inline]`

Print an rgb `c` into the output stream `ostr`.

**Parameters**

[in, out] `ostr` An output stream.

[in] `c` An rgb.

**Returns**

The modified output stream `ostr`.

Definition at line 743 of file `value/rgb.hh`.

References `mln::debug::format()`.

**9.147.3.18** `template<unsigned n> std::ostream & mln::value::operator<< ( std::ostream & ostr, const float01_<n> & f ) [inline]`

Op<<.

Definition at line 253 of file `float01_.hh`.

**9.147.3.19** `template<typename H , typename S , typename L > std::ostream & mln::value::operator<< ( std::ostream & ostr, const hsl_< H, S, L > & c ) [inline]`

Print an hsl *c* into the output stream *ostr*.

#### Parameters

[in, out] *ostr* An output stream.

[in] *c* An rgb.

#### Returns

The modified output stream *ostr*.

Definition at line 326 of file hsl.hh.

References mln::debug::format().

**9.147.3.20** `template<unsigned n> std::ostream & mln::value::operator<< ( std::ostream & ostr, const label< n > & l ) [inline]`

Print a label *l* into the output stream *ostr*.

#### Parameters

[in, out] *ostr* An output stream.

[in] *l* A label.

#### Returns

The modified output stream *ostr*.

Definition at line 353 of file label.hh.

References mln::debug::format().

**9.147.3.21** `std::ostream & mln::value::operator<< ( std::ostream & ostr, const graylevel_f & g ) [inline]`

Op<<.

Definition at line 458 of file graylevel\_f.hh.

References mln::value::graylevel\_f::value().

**9.147.3.22** `template<typename H , typename S , typename L > bool mln::value::operator==( const hsl_< H, S, L > & lhs, const hsl_< H, S, L > & rhs )`

Comparison.

Definition at line 376 of file hsl.hh.

### 9.147.3.23 `bool mln::value::operator==( const sign & lhs, const sign & rhs ) [inline]`

Comparison operator.

Definition at line 190 of file value/sign.hh.

### 9.147.3.24 `template<typename V > V mln::value::other ( const V & val ) [inline]`

Give an other value than `val`.

Definition at line 115 of file other.hh.

### 9.147.3.25 `template<typename I > stack_image< 2, const I > mln::value::stack ( const Image< I > & ima1, const Image< I > & ima2 ) [inline]`

Shortcut to build a stack with two images.

Definition at line 306 of file stack.hh.

## 9.148 `mln::value::impl` Namespace Reference

Implementation namespace of value namespace.

### 9.148.1 Detailed Description

Implementation namespace of value namespace.

## 9.149 `mln::win` Namespace Reference

Namespace of image processing routines related to win.

### Classes

- struct [backdiag2d](#)  
*Diagonal line window defined on the 2D square grid.*
- struct [ball](#)  
*Generic ball window defined on a given grid.*
- struct [cube3d](#)  
*Cube window defined on the 3D grid.*
- struct [cuboid3d](#)  
*Cuboid defined on the 3-D square grid.*
- struct [diag2d](#)  
*Diagonal line window defined on the 2D square grid.*

- struct [line](#)  
*Generic line window defined on a given grid in the given dimension.*
- class [multiple](#)  
*Multiple window.*
- class [multiple\\_size](#)  
*Definition of a multiple-size window.*
- struct [octagon2d](#)  
*Octagon window defined on the 2D square grid.*
- struct [rectangle2d](#)  
*Rectangular window defined on the 2D square grid.*

## Typedefs

- typedef [ball](#)< grid::square, [def::coord](#) > [disk2d](#)  
*2D disk window; precisely, ball-shaped window defined on the 2D square grid.*
- typedef [line](#)< grid::square, 1, [def::coord](#) > [hline2d](#)  
*Horizontal line window defined on the 2D square grid.*
- typedef [line](#)< grid::tick, 0, [def::coord](#) > [segment1d](#)  
*Segment window defined on the 1D grid.*
- typedef [line](#)< grid::cube, 0, [def::coord](#) > [sline3d](#)  
*Depth line window defined on the 3D cubic grid.*
- typedef [ball](#)< grid::cube, [def::coord](#) > [sphere3d](#)  
*3D sphere window; precisely, ball-shaped window defined on the 3D cubic grid.*
- typedef [line](#)< grid::square, 0, [def::coord](#) > [vline2d](#)  
*Vertical line window defined on the 2D square grid.*

## Functions

- template<typename N1 , typename N2 >  
[neighb](#)< typename N1::window::regular > [diff](#) (const [Neighborhood](#)< N1 > &nbh1, const [Neighborhood](#)< N2 > &nbh2)  
*Set difference between a couple of neighborhoods nbh1 and nbh2.*
- template<typename W1 , typename W2 >  
[mln\\_regular](#) (W1) [diff](#)(const [Window](#)< W1 > &win1  
*Set difference between a couple of windows win1 and win2.*
- template<typename W >  
[mln\\_regular](#) (W) [shift](#)(const [Window](#)< W > &win

Shift a window *win* with a delta-point *dp*.

- `template<typename W >`  
`W sym (const Window< W > &win)`  
*Give the symmetrical window of *win*.*
- `template<typename W >`  
`W sym (const Weighted_Window< W > &w_win)`  
*Give the symmetrical weighted window of *w\_win*.*

### 9.149.1 Detailed Description

Namespace of image processing routines related to *win*.

### 9.149.2 Function Documentation

**9.149.2.1** `template<typename N1 , typename N2 > N2 neighb< typename N1::window::regular >`  
`mln::win::diff ( const Neighborhood< N1 > & nbh1, const Neighborhood< N2 > &`  
`nbh2 )`

Set difference between a couple of neighborhoods *nbh1* and *nbh2*.

Definition at line 132 of file `win/diff.hh`.

Referenced by `mln::operator()`.

**9.149.2.2** `template<typename W1 , typename W2 > mln::win::mln_regular ( W1 ) const`  
`[inline]`

Set difference between a couple of windows *win1* and *win2*.

**9.149.2.3** `template<typename W > mln::win::mln_regular ( W ) const [inline]`

Shift a window *win* with a delta-point *dp*.

**9.149.2.4** `template<typename W > W mln::win::sym ( const Window< W > & win )`  
`[inline]`

Give the symmetrical window of *win*.

Definition at line 59 of file `sym.hh`.

Referenced by `mln::c18()`, `mln::c26()`, `mln::c4_3d()`, `mln::c6()`, `mln::morpho::hit_or_miss_background_opening()`, `mln::morpho::hit_or_miss_opening()`, `mln::morpho::opening::approx::structural()`, and `mln::morpho::closing::approx::structural()`.

**9.149.2.5** `template<typename W > W mln::win::sym ( const Weighted_Window< W > & w_win`  
`) [inline]`

Give the symmetrical weighted window of *w\_win*.



Definition at line 71 of file sym.hh.



# Chapter 10

## Class Documentation

### 10.1 `mln::accu::center< P, V >` Struct Template Reference

Mass center accumulator.

```
#include <center.hh>
```

Inherits `base< V, center< P, V > >`.

#### Public Member Functions

- `bool is_valid () const`  
*Check whether this accu is able to return a result.*
- `unsigned nsites () const`  
*Return the number of sites taken in consideration.*
- `void take_as_init (const T &t)`  
*Take as initialization the value  $t$ .*
- `void take_n_times (unsigned n, const T &t)`  
*Take  $n$  times the value  $t$ .*
- `V to_result () const`  
*Get the value of the accumulator.*
  
- `void init ()`  
*Manipulators.*

#### 10.1.1 Detailed Description

```
template<typename P, typename V = typename P::vec> struct mln::accu::center< P, V >
```

Mass center accumulator.

## Template Parameters

*P* the type of site.

*V* the type of vector to be used as result. The default vector type is the one provided by P.

Definition at line 55 of file center.hh.

## 10.1.2 Member Function Documentation

**10.1.2.1** `template<typename P , typename V > void mln::accu::center< P, V >::init ( )`  
**[inline]**

Manipulators.

Definition at line 116 of file center.hh.

References mln::literal::zero.

**10.1.2.2** `template<typename P , typename V > bool mln::accu::center< P, V >::is_valid ( )`  
**const [inline]**

Check whether this accu is able to return a result.

Definition at line 160 of file center.hh.

Referenced by mln::accu::center< P, V >::to\_result().

**10.1.2.3** `template<typename P , typename V > unsigned mln::accu::center< P, V >::nsites ( )`  
**const [inline]**

Return the number of sites taken in consideration.

Definition at line 168 of file center.hh.

**10.1.2.4** `void mln::Accumulator< center< P, V > >::take_as_init ( const T & t )`  
**[inherited]**

Take as initialization the value *t*.

Dev note: this is a final method; override if needed by take\_as\_init\_ (ending with '\_').

**10.1.2.5** `void mln::Accumulator< center< P, V > >::take_n_times ( unsigned n, const T & t )`  
**[inherited]**

Take *n* times the value *t*.

Dev note: this is a final method; override if needed by take\_as\_init\_ (ending with '\_').

**10.1.2.6** `template<typename P , typename V > V mln::accu::center< P, V >::to_result ( ) const`  
**[inline]**

Get the value of the accumulator.

Definition at line 142 of file center.hh.

References mln::accu::center< P, V >::is\_valid().

## 10.2 mln::accu::convolve< T1, T2, R > Struct Template Reference

Generic convolution accumulator class.

```
#include <convolve.hh>
```

Inherits base< R, convolve< T1, T2, R > >.

### Public Member Functions

- bool [is\\_valid](#) () const  
*Check whether this accu is able to return a result.*
- void [take\\_as\\_init](#) (const T &t)  
*Take as initialization the value t.*
- void [take\\_n\\_times](#) (unsigned n, const T &t)  
*Take n times the value t.*
- R [to\\_result](#) () const  
*Get the value of the accumulator.*
- void [init](#) ()  
*Manipulators.*

### 10.2.1 Detailed Description

```
template<typename T1, typename T2, typename R = typename mln::trait::value_< typename
mln::trait::op::times< T1, T2 >::ret >::sum> struct mln::accu::convolve< T1, T2, R >
```

Generic convolution accumulator class. Parameters T1 and T2 are the type of values to be convolved. Parameter R is the result type.

Definition at line 54 of file accu/convolve.hh.

### 10.2.2 Member Function Documentation

```
10.2.2.1 template<typename T1, typename T2, typename R > void mln::accu::convolve< T1,
T2, R >::init ( ) [inline]
```

Manipulators.

Definition at line 96 of file accu/convolve.hh.

References mln::literal::zero.

### 10.2.2.2 `template<typename T1, typename T2, typename R > bool mln::accu::convolve< T1, T2, R >::is_valid ( ) const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 137 of file accu/convolve.hh.

### 10.2.2.3 `void mln::Accumulator< convolve< T1, T2, R > >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

### 10.2.2.4 `void mln::Accumulator< convolve< T1, T2, R > >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

### 10.2.2.5 `template<typename T1, typename T2, typename R > R mln::accu::convolve< T1, T2, R >::to_result ( ) const [inline]`

Get the value of the accumulator.

Definition at line 129 of file accu/convolve.hh.

## 10.3 `mln::accu::count_adjacent_vertices< F, S > Struct Template Reference`

[Accumulator](#) class counting the number of vertices adjacent to a set of `mln::p_edges_psite` (i.e., a set of edges).

```
#include <count_adjacent_vertices.hh>
```

Inherits base< unsigned, count\_adjacent\_vertices< F, S > >.

### Public Member Functions

- bool [is\\_valid](#) () const  
*Return whether this accu can return a result.*
- void [take\\_as\\_init](#) (const T &t)  
*Take as initialization the value  $t$ .*
- void [take\\_n\\_times](#) (unsigned n, const T &t)  
*Take  $n$  times the value  $t$ .*
- unsigned [to\\_result](#) () const

*Get the value of the accumulator.*

- void `init()`  
*Manipulators.*
- void `set_value(unsigned c)`  
*Force the value of the counter to c.*

### 10.3.1 Detailed Description

`template<typename F, typename S> struct mln::accu::count_adjacent_vertices< F, S >`

**Accumulator** class counting the number of vertices adjacent to a set of `mln::p_edges_psite` (i.e., a set of edges). The type to be count is `mln::util::pix< pw::image<F, S> >` where `F` and `S` are the parameters of this class.

This accumulator is used by `mln::closing_area_on_vertices` and `mln::opening_area_on_vertices`.

Definition at line 58 of file `accu/count_adjacent_vertices.hh`.

### 10.3.2 Member Function Documentation

**10.3.2.1** `template<typename F, typename S > void mln::accu::count_adjacent_vertices< F, S >::init( ) [inline]`

Manipulators.

Definition at line 123 of file `accu/count_adjacent_vertices.hh`.

**10.3.2.2** `template<typename F, typename S > bool mln::accu::count_adjacent_vertices< F, S >::is_valid( ) const [inline]`

Return whether this `accu` can return a result.

Definition at line 177 of file `accu/count_adjacent_vertices.hh`.

**10.3.2.3** `template<typename F, typename S > void mln::accu::count_adjacent_vertices< F, S >::set_value( unsigned c ) [inline]`

Force the value of the counter to `c`.

Definition at line 159 of file `accu/count_adjacent_vertices.hh`.

**10.3.2.4** `void mln::Accumulator< count_adjacent_vertices< F, S > >::take_as_init( const T & t ) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.3.2.5** `void mln::Accumulator< count_adjacent_vertices< F, S > >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.3.2.6** `template<typename F , typename S > unsigned mln::accu::count_adjacent_vertices< F, S >::to_result ( ) const [inline]`

Get the value of the accumulator.

Definition at line 151 of file `accu/count_adjacent_vertices.hh`.

## 10.4 `mln::accu::count_labels< L >` Struct Template Reference

Count the number of different labels in an image.

```
#include <count_labels.hh>
```

Inherits `base< unsigned, count_labels< L > >`.

### Public Member Functions

- `bool is_valid () const`  
*Check whether this `accu` is able to return a result.*
- `void take_as_init (const T &t)`  
*Take as initialization the value  $t$ .*
- `void take_n_times (unsigned n, const T &t)`  
*Take  $n$  times the value  $t$ .*
- `unsigned to_result () const`  
*Get the value of the accumulator.*
- `void init ()`  
*Manipulators.*
- `void set_value (unsigned c)`  
*Force the value of the counter to  $c$ .*

### 10.4.1 Detailed Description

`template<typename L> struct mln::accu::count_labels< L >`

Count the number of different labels in an image. The parameter  $L$  is the label type to be count.

Definition at line 52 of file `count_labels.hh`.



## 10.4.2 Member Function Documentation

### 10.4.2.1 `template<typename L > void mln::accu::count_labels< L >::init ( ) [inline]`

Manipulators.

Definition at line 113 of file count\_labels.hh.

### 10.4.2.2 `template<typename L > bool mln::accu::count_labels< L >::is_valid ( ) const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 163 of file count\_labels.hh.

### 10.4.2.3 `template<typename L > void mln::accu::count_labels< L >::set_value ( unsigned c ) [inline]`

Force the value of the counter to *c*.

Definition at line 155 of file count\_labels.hh.

### 10.4.2.4 `void mln::Accumulator< count_labels< L > >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

### 10.4.2.5 `void mln::Accumulator< count_labels< L > >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take *n* times the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

### 10.4.2.6 `template<typename L > unsigned mln::accu::count_labels< L >::to_result ( ) const [inline]`

Get the value of the accumulator.

Definition at line 146 of file count\_labels.hh.

## 10.5 mln::accu::count\_value< V > Struct Template Reference

Define an accumulator that counts the occurrence of a given value.

```
#include <count_value.hh>
```

Inherits `base< unsigned, count_value< V > >`.

## Public Member Functions

- `bool is_valid () const`  
*Check whether this `accu` is able to return a result.*
- `void take_as_init (const T &t)`  
*Take as initialization the value  $t$ .*
- `void take_n_times (unsigned n, const T &t)`  
*Take  $n$  times the value  $t$ .*
- `unsigned to_result () const`  
*Get the value of the accumulator.*
- `void init ()`  
*Manipulators.*
- `void set_value (unsigned c)`  
*Force the value of the counter to  $c$ .*

### 10.5.1 Detailed Description

`template<typename V> struct mln::accu::count_value< V >`

Define an accumulator that counts the occurrence of a given value.

Definition at line 72 of file `count_value.hh`.

### 10.5.2 Member Function Documentation

**10.5.2.1** `template<typename V > void mln::accu::count_value< V >::init ( ) [inline]`

Manipulators.

Definition at line 153 of file `count_value.hh`.

**10.5.2.2** `template<typename V > bool mln::accu::count_value< V >::is_valid ( ) const [inline]`

Check whether this `accu` is able to return a result.

Always true here.

Definition at line 216 of file `count_value.hh`.

**10.5.2.3** `template<typename V > void mln::accu::count_value< V >::set_value ( unsigned c ) [inline]`

Force the value of the counter to  $c$ .

Definition at line 208 of file `count_value.hh`.

#### 10.5.2.4 void mln::Accumulator< count\_value< V > >::take\_as\_init ( const T & t ) [inherited]

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by take\_as\_init\_ (ending with '\_').

#### 10.5.2.5 void mln::Accumulator< count\_value< V > >::take\_n\_times ( unsigned n, const T & t ) [inherited]

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by take\_as\_init\_ (ending with '\_').

#### 10.5.2.6 template<typename V > unsigned mln::accu::count\_value< V >::to\_result ( ) const [inline]

Get the value of the accumulator.

Definition at line 199 of file count\_value.hh.

## 10.6 mln::accu::histo< V > Struct Template Reference

Generic histogram class over a value set with type  $V$ .

```
#include <histo.hh>
```

Inherits base< const std::vector< unsigned > &, histo< V > >.

### Public Member Functions

- bool [is\\_valid](#) () const  
*Check whether this accu is able to return a result.*
- void [take\\_as\\_init](#) (const T &t)  
*Take as initialization the value  $t$ .*
- void [take\\_n\\_times](#) (unsigned n, const T &t)  
*Take  $n$  times the value  $t$ .*
- void [take](#) (const argument &t)  
*Manipulators.*
- const std::vector< unsigned > & [vect](#) () const  
*Get the value of the accumulator.*

## 10.6.1 Detailed Description

**template<typename V> struct mln::accu::histo< V >**

Generic histogram class over a value set with type V.

Definition at line 56 of file accu/histo.hh.

## 10.6.2 Member Function Documentation

**10.6.2.1 template<typename V > bool mln::accu::histo< V >::is\_valid ( ) const [inline]**

Check whether this accu is able to return a result.

Always true here.

Definition at line 236 of file accu/histo.hh.

**10.6.2.2 template<typename V > void mln::accu::histo< V >::take ( const argument & t ) [inline]**

Manipulators.

Definition at line 129 of file accu/histo.hh.

**10.6.2.3 void mln::Accumulator< histo< V > >::take\_as\_init ( const T & t ) [inherited]**

Take as initialization the value t.

Dev note: this is a final method; override if needed by take\_as\_init\_ (ending with '\_').

**10.6.2.4 void mln::Accumulator< histo< V > >::take\_n\_times ( unsigned n, const T & t ) [inherited]**

Take n times the value t.

Dev note: this is a final method; override if needed by take\_as\_init\_ (ending with '\_').

**10.6.2.5 template<typename V > const std::vector< unsigned > & mln::accu::histo< V >::vect ( ) const [inline]**

Get the value of the accumulator.

Definition at line 201 of file accu/histo.hh.

## 10.7 mln::accu::label\_used< L > Struct Template Reference

References all the labels used.

```
#include <label_used.hh>
```

Inherits base< const fun::i2v::array< bool > &, label\_used< L > >.

## Public Member Functions

- void `init()`  
*Initialize accumulator attributes.*
- bool `is_valid()` const  
*Check whether this accu is able to return a result.*
- void `take_as_init(const T &t)`  
*Take as initialization the value  $t$ .*
- void `take_n_times(unsigned n, const T &t)`  
*Take  $n$  times the value  $t$ .*
- const fun::i2v::array< bool > & `to_result()` const  
*Get the value of the accumulator.*
- void `take(const argument &)`  
*Manipulators.*

### 10.7.1 Detailed Description

`template<typename L> struct mln::accu::label_used< L >`

References all the labels used. The parameter  $L$  is the label type.

Definition at line 53 of file label\_used.hh.

### 10.7.2 Member Function Documentation

**10.7.2.1** `template<typename L > void mln::accu::label_used< L >::init( ) [inline]`

Initialize accumulator attributes.

Definition at line 110 of file label\_used.hh.

**10.7.2.2** `template<typename L > bool mln::accu::label_used< L >::is_valid( ) const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 150 of file label\_used.hh.

**10.7.2.3** `template<typename L > void mln::accu::label_used< L >::take( const argument & l ) [inline]`

Manipulators.

Definition at line 118 of file label\_used.hh.

#### 10.7.2.4 void mln::Accumulator< label\_used< L > >::take\_as\_init ( const T & t ) [inherited]

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by take\_as\_init\_ (ending with '\_').

#### 10.7.2.5 void mln::Accumulator< label\_used< L > >::take\_n\_times ( unsigned n, const T & t ) [inherited]

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by take\_as\_init\_ (ending with '\_').

#### 10.7.2.6 template<typename L > const fun::i2v::array< bool > & mln::accu::label\_used< L >::to\_result ( ) const [inline]

Get the value of the accumulator.

Definition at line 142 of file label\_used.hh.

## 10.8 mln::accu::logic::land Struct Reference

"Logical-and" accumulator.

```
#include <land.hh>
```

Inherits base< bool, land >.

### Public Member Functions

- bool [is\\_valid](#) () const  
*Check whether this accu is able to return a result.*
- void [take\\_as\\_init](#) (const T &t)  
*Take as initialization the value  $t$ .*
- void [take\\_n\\_times](#) (unsigned n, const T &t)  
*Take  $n$  times the value  $t$ .*
- bool [to\\_result](#) () const  
*Get the value of the accumulator.*
- void [init](#) ()  
*Manipulators.*

## 10.8.1 Detailed Description

"Logical-and" accumulator.

Definition at line 96 of file accu/logic/land.hh.

## 10.8.2 Member Function Documentation

### 10.8.2.1 void mln::accu::logic::land::init ( ) [inline]

Manipulators.

Definition at line 136 of file accu/logic/land.hh.

### 10.8.2.2 bool mln::accu::logic::land::is\_valid ( ) const [inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 185 of file accu/logic/land.hh.

### 10.8.2.3 void mln::Accumulator< land >::take\_as\_init ( const T & t ) [inherited]

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by take\_as\_init\_ (ending with '\_').

### 10.8.2.4 void mln::Accumulator< land >::take\_n\_times ( unsigned n, const T & t ) [inherited]

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by take\_as\_init\_ (ending with '\_').

### 10.8.2.5 bool mln::accu::logic::land::to\_result ( ) const [inline]

Get the value of the accumulator.

Definition at line 178 of file accu/logic/land.hh.

## 10.9 mln::accu::logic::land\_basic Struct Reference

"Logical-and" accumulator.

```
#include <land_basic.hh>
```

Inherits base< bool, land\_basic >.

### Public Member Functions

- bool [can\\_stop](#) () const

*Test if it is worth for this accumulator to take extra data.*

- bool `is_valid()` const

*Check whether this accu is able to return a result.*

- void `take_as_init(const T &t)`

*Take as initialization the value  $t$ .*

- void `take_n_times(unsigned n, const T &t)`

*Take  $n$  times the value  $t$ .*

- bool `to_result()` const

*Get the value of the accumulator.*

- void `init()`

*Manipulators.*

## 10.9.1 Detailed Description

"Logical-and" accumulator. Conversely to `accu::logic::land`, this version does not have the 'untake' method but features the 'can\_stop' method.

Definition at line 99 of file `land_basic.hh`.

## 10.9.2 Member Function Documentation

### 10.9.2.1 `bool mln::accu::logic::land_basic::can_stop() const [inline]`

Test if it is worth for this accumulator to take extra data.

If the result is already 'false' (because this accumulator has already taken a 'false' value), `can_stop` returns true.

Definition at line 181 of file `land_basic.hh`.

### 10.9.2.2 `void mln::accu::logic::land_basic::init() [inline]`

Manipulators.

Definition at line 140 of file `land_basic.hh`.

### 10.9.2.3 `bool mln::accu::logic::land_basic::is_valid() const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 174 of file `land_basic.hh`.



**10.9.2.4 void mln::Accumulator< land\_basic >::take\_as\_init ( const T & t ) [inherited]**

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by take\_as\_init\_ (ending with '\_').

**10.9.2.5 void mln::Accumulator< land\_basic >::take\_n\_times ( unsigned n, const T & t ) [inherited]**

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by take\_as\_init\_ (ending with '\_').

**10.9.2.6 bool mln::accu::logic::land\_basic::to\_result ( ) const [inline]**

Get the value of the accumulator.

Definition at line 167 of file land\_basic.hh.

**10.10 mln::accu::logic::lor Struct Reference**

"Logical-or" accumulator.

```
#include <lor.hh>
```

Inherits base< bool, lor >.

**Public Member Functions**

- bool [is\\_valid](#) () const  
*Check whether this accu is able to return a result.*
- void [take\\_as\\_init](#) (const T &t)  
*Take as initialization the value  $t$ .*
- void [take\\_n\\_times](#) (unsigned n, const T &t)  
*Take  $n$  times the value  $t$ .*
- bool [to\\_result](#) () const  
*Get the value of the accumulator.*
- void [init](#) ()  
*Manipulators.*

**10.10.1 Detailed Description**

"Logical-or" accumulator.

Definition at line 95 of file accu/logic/lor.hh.

## 10.10.2 Member Function Documentation

### 10.10.2.1 void mln::accu::logic::lor::init ( ) [inline]

Manipulators.

Definition at line 134 of file accu/logic/lor.hh.

### 10.10.2.2 bool mln::accu::logic::lor::is\_valid ( ) const [inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 183 of file accu/logic/lor.hh.

### 10.10.2.3 void mln::Accumulator< lor >::take\_as\_init ( const T & t ) [inherited]

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

### 10.10.2.4 void mln::Accumulator< lor >::take\_n\_times ( unsigned n, const T & t ) [inherited]

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

### 10.10.2.5 bool mln::accu::logic::lor::to\_result ( ) const [inline]

Get the value of the accumulator.

Definition at line 176 of file accu/logic/lor.hh.

## 10.11 mln::accu::logic::lor\_basic Struct Reference

"Logical-or" accumulator class.

```
#include <lor_basic.hh>
```

Inherits base< bool, lor\_basic >.

### Public Member Functions

- bool [can\\_stop](#) () const  
*Test if it is worth for this accumulator to take extra data.*
- bool [is\\_valid](#) () const  
*Check whether this accu is able to return a result.*
- void [take\\_as\\_init](#) (const T &t)

Take as initialization the value  $t$ .

- void [take\\_n\\_times](#) (unsigned n, const T &t)

Take  $n$  times the value  $t$ .

- bool [to\\_result](#) () const

Get the value of the accumulator.

- void [init](#) ()

Manipulators.

### 10.11.1 Detailed Description

"Logical-or" accumulator class. Conversely to [accu::logic::lor](#), this version does not have the 'untake' method but features the 'can\_stop' method.

Definition at line 98 of file lor\_basic.hh.

### 10.11.2 Member Function Documentation

#### 10.11.2.1 bool mln::accu::logic::lor\_basic::can\_stop ( ) const [inline]

Test if it is worth for this accumulator to take extra data.

If the result is already 'true' (because this accumulator has already taken a 'true' value), can\_stop returns true.

Definition at line 180 of file lor\_basic.hh.

#### 10.11.2.2 void mln::accu::logic::lor\_basic::init ( ) [inline]

Manipulators.

Definition at line 139 of file lor\_basic.hh.

#### 10.11.2.3 bool mln::accu::logic::lor\_basic::is\_valid ( ) const [inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 173 of file lor\_basic.hh.

#### 10.11.2.4 void mln::Accumulator<lor\_basic>::take\_as\_init ( const T & t ) [inherited]

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by take\_as\_init\_ (ending with '\_').

### 10.11.2.5 void mln::Accumulator< lor\_basic >::take\_n\_times ( unsigned n, const T & t ) [inherited]

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

### 10.11.2.6 bool mln::accu::logic::lor\_basic::to\_result ( ) const [inline]

Get the value of the accumulator.

Definition at line 166 of file `lor_basic.hh`.

## 10.12 mln::accu::maj\_h< T > Struct Template Reference

Compute the majority value.

```
#include <maj_h.hh>
```

Inherits base< const T &, maj\_h< T > >.

### Public Member Functions

- bool [is\\_valid](#) () const  
*Check whether this accu is able to return a result.*
- void [take\\_as\\_init](#) (const T &t)  
*Take as initialization the value  $t$ .*
- void [take\\_n\\_times](#) (unsigned n, const T &t)  
*Take  $n$  times the value  $t$ .*
- const T & [to\\_result](#) () const  
*Get the value of the accumulator.*
- void [init](#) ()  
*Manipulators.*

### 10.12.1 Detailed Description

```
template<typename T> struct mln::accu::maj_h< T >
```

Compute the majority value. It is based on a histogram. The parameter  $T$  is the type of values.

Definition at line 57 of file `maj_h.hh`.

## 10.12.2 Member Function Documentation

### 10.12.2.1 `template<typename T> void mln::accu::maj_h< T >::init ( ) [inline]`

Manipulators.

Definition at line 129 of file maj\_h.hh.

### 10.12.2.2 `template<typename T> bool mln::accu::maj_h< T >::is_valid ( ) const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 197 of file maj\_h.hh.

### 10.12.2.3 `void mln::Accumulator< maj_h< T > >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

### 10.12.2.4 `void mln::Accumulator< maj_h< T > >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

### 10.12.2.5 `template<typename T> const T & mln::accu::maj_h< T >::to_result ( ) const [inline]`

Get the value of the accumulator.

Definition at line 187 of file maj\_h.hh.

## 10.13 mln::accu::math::count< T > Struct Template Reference

Generic counter accumulator.

```
#include <count.hh>
```

Inherits `base< unsigned, count< T > >`.

### Public Member Functions

- `bool is_valid () const`  
*Check whether this accu is able to return a result.*
- `void take_as_init (const T &t)`  
*Take as initialization the value  $t$ .*

- void `take_n_times` (unsigned `n`, const `T` &`t`)  
*Take  $n$  times the value  $t$ .*
- unsigned `to_result` () const  
*Get the value of the accumulator.*
- void `init` ()  
*Manipulators.*
- void `set_value` (unsigned `c`)  
*Force the value of the counter to  $c$ .*

### 10.13.1 Detailed Description

**template<typename T> struct mln::accu::math::count< T >**

Generic counter accumulator. The parameter  $T$  is the type to be count.

Definition at line 100 of file count.hh.

### 10.13.2 Member Function Documentation

**10.13.2.1 template<typename T > void mln::accu::math::count< T >::init ( ) [inline]**

Manipulators.

Definition at line 145 of file count.hh.

**10.13.2.2 template<typename T > bool mln::accu::math::count< T >::is\_valid ( ) const [inline]**

Check whether this accu is able to return a result.

Always true here.

Definition at line 203 of file count.hh.

**10.13.2.3 template<typename T > void mln::accu::math::count< T >::set\_value ( unsigned  $c$  ) [inline]**

Force the value of the counter to  $c$ .

Definition at line 195 of file count.hh.

**10.13.2.4 void mln::Accumulator< count< T > >::take\_as\_init ( const T &  $t$  ) [inherited]**

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.13.2.5** void mln::Accumulator< count< T > >::take\_n\_times ( unsigned n, const T & t )  
[inherited]

Take n times the value t.

Dev note: this is a final method; override if needed by take\_as\_init\_ (ending with '\_').

**10.13.2.6** template<typename T > unsigned mln::accu::math::count< T >::to\_result ( ) const  
[inline]

Get the value of the accumulator.

Definition at line 187 of file count.hh.

## 10.14 mln::accu::math::inf< T > Struct Template Reference

Generic inf accumulator class.

```
#include <inf.hh>
```

Inherits base< const T &, inf< T > >.

### Public Member Functions

- bool [is\\_valid](#) () const  
*Check whether this accu is able to return a result.*
- void [take\\_as\\_init](#) (const T &t)  
*Take as initialization the value t.*
- void [take\\_n\\_times](#) (unsigned n, const T &t)  
*Take n times the value t.*
- const T & [to\\_result](#) () const  
*Get the value of the accumulator.*
- void [init](#) ()  
*Manipulators.*

### 10.14.1 Detailed Description

```
template<typename T> struct mln::accu::math::inf< T >
```

Generic inf accumulator class. The parameter T is the type of values.

Definition at line 56 of file accu/math/inf.hh.

## 10.14.2 Member Function Documentation

### 10.14.2.1 `template<typename T> void mln::accu::math::inf< T >::init ( ) [inline]`

Manipulators.

Definition at line 126 of file `accu/math/inf.hh`.

### 10.14.2.2 `template<typename T> bool mln::accu::math::inf< T >::is_valid ( ) const [inline]`

Check whether this `accu` is able to return a result.

Always true here.

Definition at line 164 of file `accu/math/inf.hh`.

### 10.14.2.3 `void mln::Accumulator< inf< T > >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

### 10.14.2.4 `void mln::Accumulator< inf< T > >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

### 10.14.2.5 `template<typename T> const T & mln::accu::math::inf< T >::to_result ( ) const [inline]`

Get the value of the accumulator.

Definition at line 156 of file `accu/math/inf.hh`.

## 10.15 `mln::accu::math::sum< T, S >` Struct Template Reference

Generic sum accumulator class.

```
#include <sum.hh>
```

Inherits `base< const S &, sum< T, S > >`.

### Public Member Functions

- `bool is_valid () const`  
*Check whether this `accu` is able to return a result.*
- `void take_as_init (const T &t)`  
*Take as initialization the value `t`.*



- void `take_n_times` (unsigned `n`, const `T` &`t`)  
*Take `n` times the value `t`.*
- const `S` & `to_result` () const  
*Get the value of the accumulator.*
- void `init` ()  
*Manipulators.*

### 10.15.1 Detailed Description

**template<typename T, typename S = typename mln::value::props< T >::sum> struct mln::accu::math::sum< T, S >**

Generic sum accumulator class. Parameter `T` is the type of values that we sum. Parameter `S` is the type to store the value sum; the default type of `S` is the summation type (property) of `T`.

Definition at line 112 of file `accu/math/sum.hh`.

### 10.15.2 Member Function Documentation

**10.15.2.1 template<typename T, typename S > void mln::accu::math::sum< T, S >::init ( )**  
**[inline]**

Manipulators.

Definition at line 161 of file `accu/math/sum.hh`.

References `mln::literal::zero`.

**10.15.2.2 template<typename T, typename S > bool mln::accu::math::sum< T, S >::is\_valid ( )**  
**const [inline]**

Check whether this `accu` is able to return a result.

Always true here.

Definition at line 222 of file `accu/math/sum.hh`.

**10.15.2.3 void mln::Accumulator< sum< T, S > >::take\_as\_init ( const T & *t* )** **[inherited]**

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.15.2.4 void mln::Accumulator< sum< T, S > >::take\_n\_times ( unsigned *n*, const T & *t* )**  
**[inherited]**

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

### 10.15.2.5 `template<typename T , typename S > const S & mln::accu::math::sum< T, S >::to_result ( ) const [inline]`

Get the value of the accumulator.

Definition at line 206 of file `accu/math/sum.hh`.

## 10.16 `mln::accu::math::sup< T >` Struct Template Reference

Generic sup accumulator class.

```
#include <sup.hh>
```

Inherits `base< const T &, sup< T > >`.

### Public Member Functions

- `bool is_valid ( ) const`  
*Check whether this accu is able to return a result.*
- `void take_as_init (const T &t)`  
*Take as initialization the value  $t$ .*
- `void take_n_times (unsigned n, const T &t)`  
*Take  $n$  times the value  $t$ .*
- `const T & to_result ( ) const`  
*Get the value of the accumulator.*
- `void init ( )`  
*Manipulators.*

### 10.16.1 Detailed Description

```
template<typename T> struct mln::accu::math::sup< T >
```

Generic sup accumulator class. The parameter `T` is the type of values.

Definition at line 56 of file `accu/math/sup.hh`.

### 10.16.2 Member Function Documentation

#### 10.16.2.1 `template<typename T > void mln::accu::math::sup< T >::init ( ) [inline]`

Manipulators.

Definition at line 128 of file `accu/math/sup.hh`.

**10.16.2.2** `template<typename T > bool mln::accu::math::sup< T >::is_valid ( ) const`  
**[inline]**

Check whether this accu is able to return a result.

Always true here.

Definition at line 166 of file accu/math/sup.hh.

**10.16.2.3** `void mln::Accumulator< sup< T > >::take_as_init ( const T & t )` **[inherited]**

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.16.2.4** `void mln::Accumulator< sup< T > >::take_n_times ( unsigned n, const T & t )`  
**[inherited]**

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.16.2.5** `template<typename T > const T & mln::accu::math::sup< T >::to_result ( ) const`  
**[inline]**

Get the value of the accumulator.

Definition at line 158 of file accu/math/sup.hh.

## 10.17 mln::accu::max\_site< I > Struct Template Reference

Define an accumulator that computes the first site with the maximum value in an image.

```
#include <max_site.hh>
```

Inherits `base< I::psite, max_site< I > >`.

### Public Member Functions

- `bool is_valid () const`  
*Check whether this accu is able to return a result.*
- `void take_as_init (const T &t)`  
*Take as initialization the value  $t$ .*
- `void take_n_times (unsigned n, const T &t)`  
*Take  $n$  times the value  $t$ .*
- `I::psite to_result () const`  
*Get the value of the accumulator.*

- void `init()`  
*Manipulators.*

### 10.17.1 Detailed Description

**template<typename I> struct mln::accu::max\_site< I >**

Define an accumulator that computes the first site with the maximum value in an image.

Definition at line 53 of file max\_site.hh.

### 10.17.2 Member Function Documentation

**10.17.2.1 template<typename I > void mln::accu::max\_site< I >::init ( ) [inline]**

Manipulators.

Definition at line 114 of file max\_site.hh.

**10.17.2.2 template<typename I > bool mln::accu::max\_site< I >::is\_valid ( ) const [inline]**

Check whether this accu is able to return a result.

Always true here.

Definition at line 174 of file max\_site.hh.

**10.17.2.3 void mln::Accumulator< max\_site< I > >::take\_as\_init ( const T & t ) [inherited]**

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.17.2.4 void mln::Accumulator< max\_site< I > >::take\_n\_times ( unsigned n, const T & t ) [inherited]**

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.17.2.5 template<typename I > I::psite mln::accu::max\_site< I >::to\_result ( ) const [inline]**

Get the value of the accumulator.

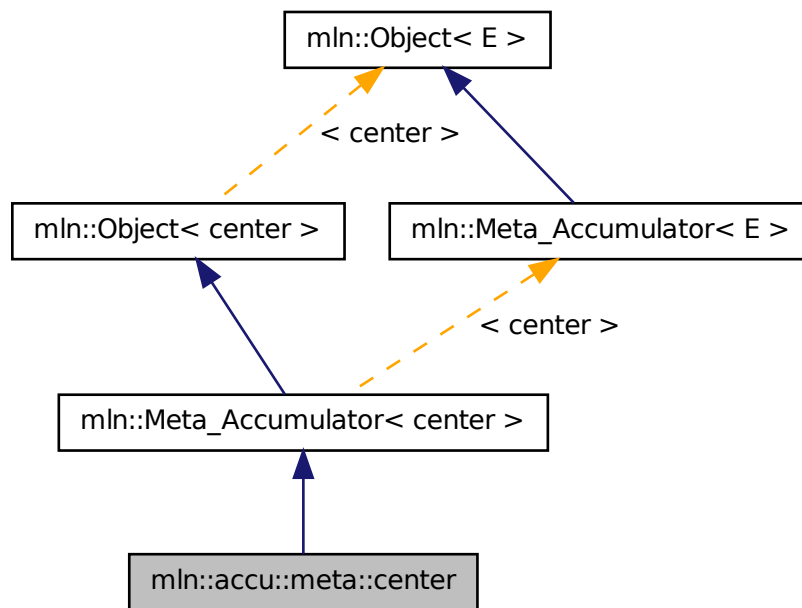
Definition at line 150 of file max\_site.hh.

## 10.18 mln::accu::meta::center Struct Reference

Meta accumulator for center.

```
#include <center.hh>
```

Inheritance diagram for mln::accu::meta::center:



### 10.18.1 Detailed Description

Meta accumulator for center.

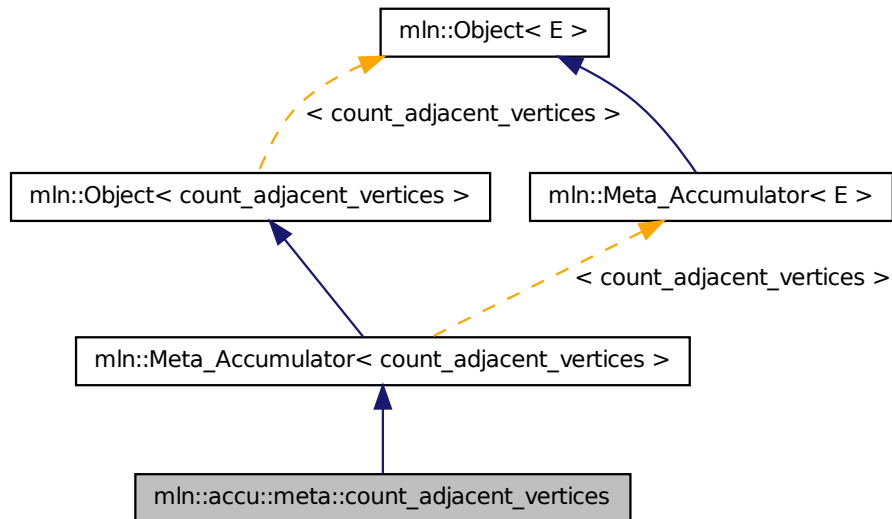
Definition at line 89 of file center.hh.

## 10.19 mln::accu::meta::count\_adjacent\_vertices Struct Reference

Meta accumulator for [count\\_adjacent\\_vertices](#).

```
#include <count_adjacent_vertices.hh>
```

Inheritance diagram for `mln::accu::meta::count_adjacent_vertices`:



### 10.19.1 Detailed Description

Meta accumulator for [count\\_adjacent\\_vertices](#).

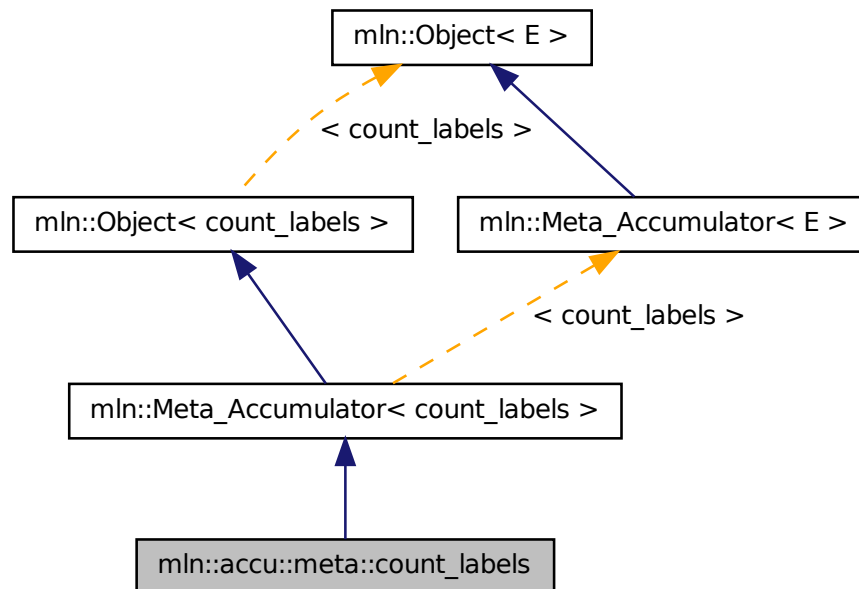
Definition at line 98 of file `accu/count_adjacent_vertices.hh`.

## 10.20 mln::accu::meta::count\_labels Struct Reference

Meta accumulator for [count\\_labels](#).

```
#include <count_labels.hh>
```

Inheritance diagram for mln::accu::meta::count\_labels:



### 10.20.1 Detailed Description

Meta accumulator for [count\\_labels](#).

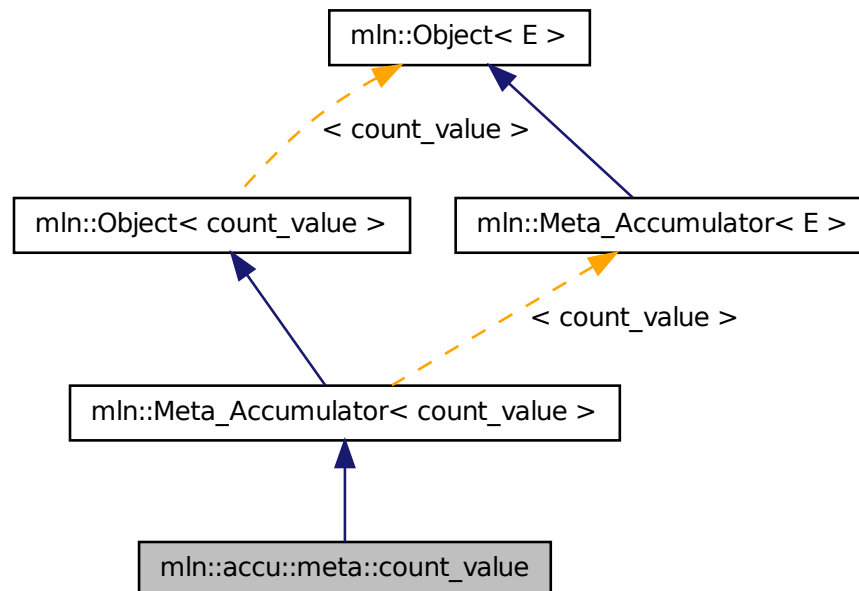
Definition at line 88 of file count\_labels.hh.

## 10.21 mln::accu::meta::count\_value Struct Reference

FIXME: How to write a meta accumulator with a constructor taking a generic argument? Meta accumulator for [count\\_value](#).

```
#include <count_value.hh>
```

Inheritance diagram for mln::accu::meta::count\_value:



### 10.21.1 Detailed Description

FIXME: How to write a meta accumulator with a constructor taking a generic argument? Meta accumulator for [count\\_value](#).

Definition at line 116 of file count\_value.hh.

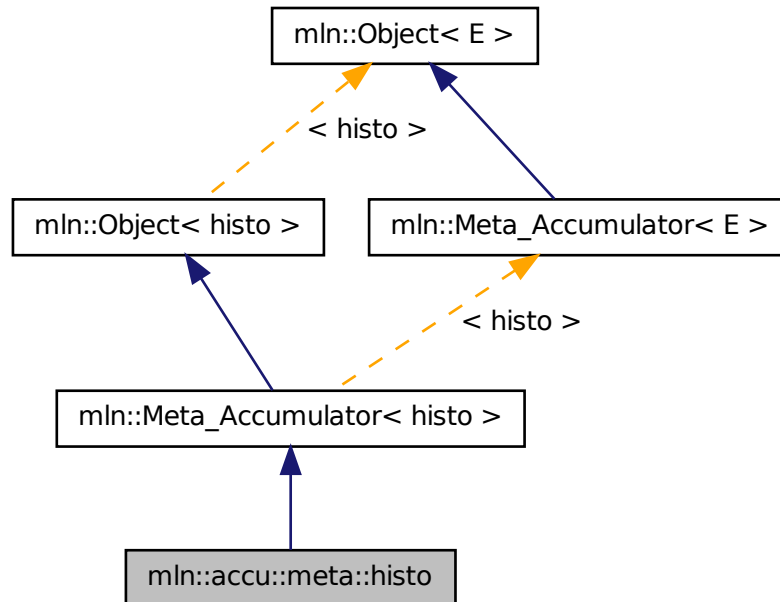
## 10.22 mln::accu::meta::histo Struct Reference

Meta accumulator for histo.

```
#include <histo.hh>
```



Inheritance diagram for mln::accu::meta::histo:



### 10.22.1 Detailed Description

Meta accumulator for histo.

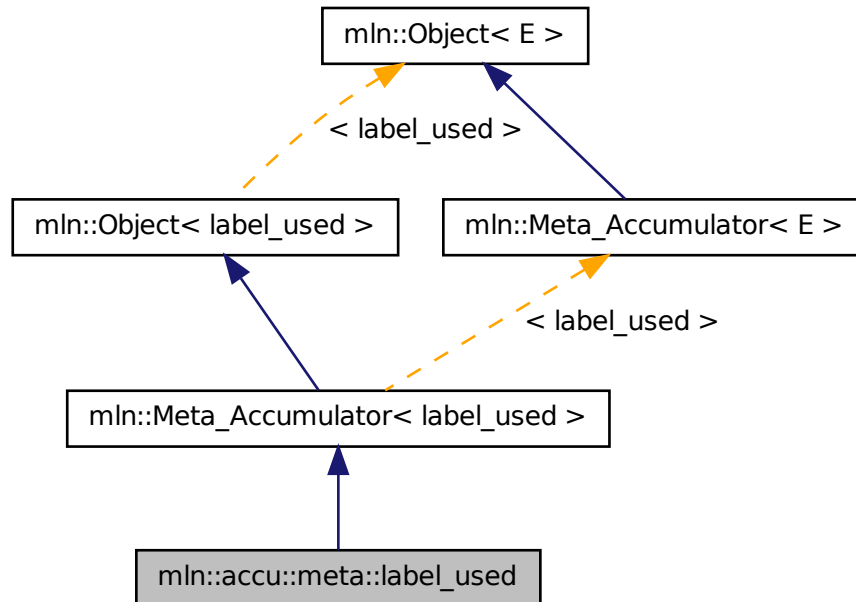
Definition at line 102 of file `accu/histo.hh`.

## 10.23 mln::accu::meta::label\_used Struct Reference

Meta accumulator for [label\\_used](#).

```
#include <label_used.hh>
```

Inheritance diagram for `mln::accu::meta::label_used`:



### 10.23.1 Detailed Description

Meta accumulator for [label\\_used](#).

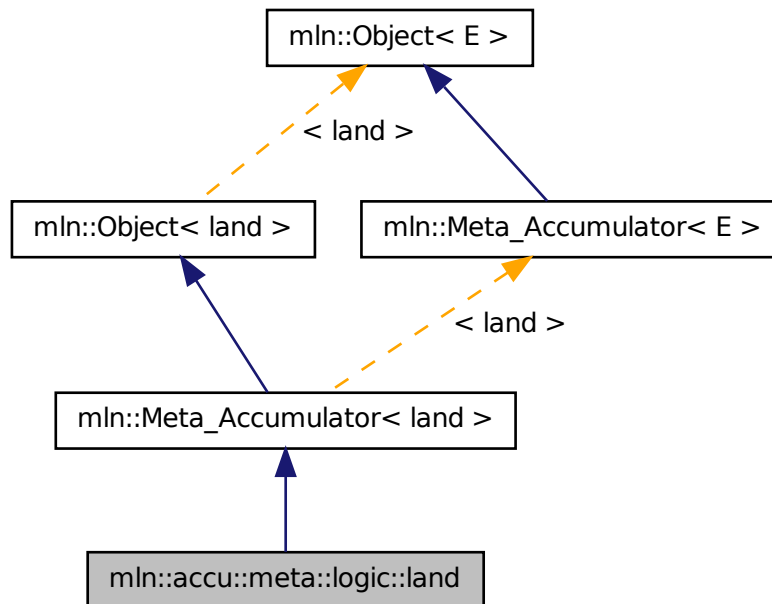
Definition at line 85 of file `label_used.hh`.

## 10.24 mln::accu::meta::logic::land Struct Reference

Meta accumulator for `land`.

```
#include <land.hh>
```

Inheritance diagram for mln::accu::meta::logic::land:



### 10.24.1 Detailed Description

Meta accumulator for land.

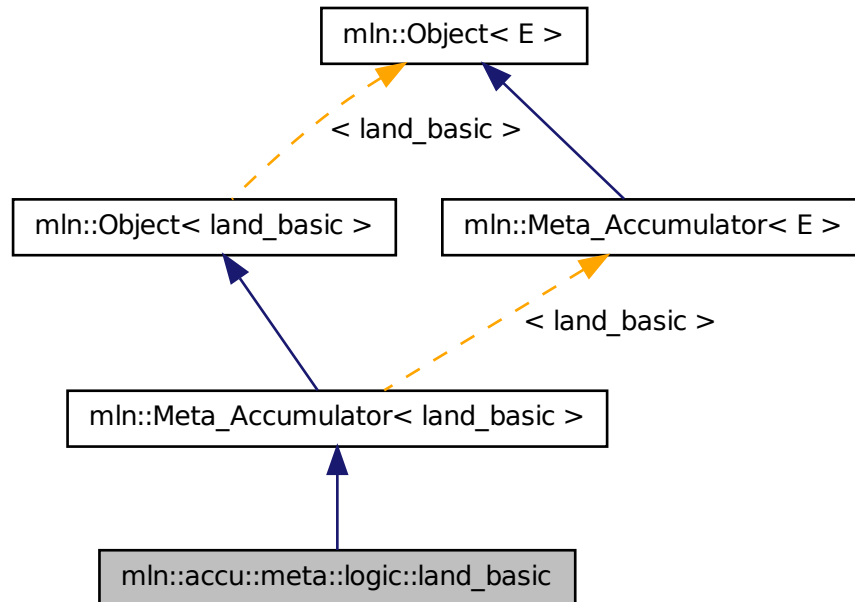
Definition at line 76 of file `accu/logic/land.hh`.

## 10.25 mln::accu::meta::logic::land\_basic Struct Reference

Meta accumulator for [land\\_basic](#).

```
#include <land_basic.hh>
```

Inheritance diagram for mln::accu::meta::logic::land\_basic:



### 10.25.1 Detailed Description

Meta accumulator for [land\\_basic](#).

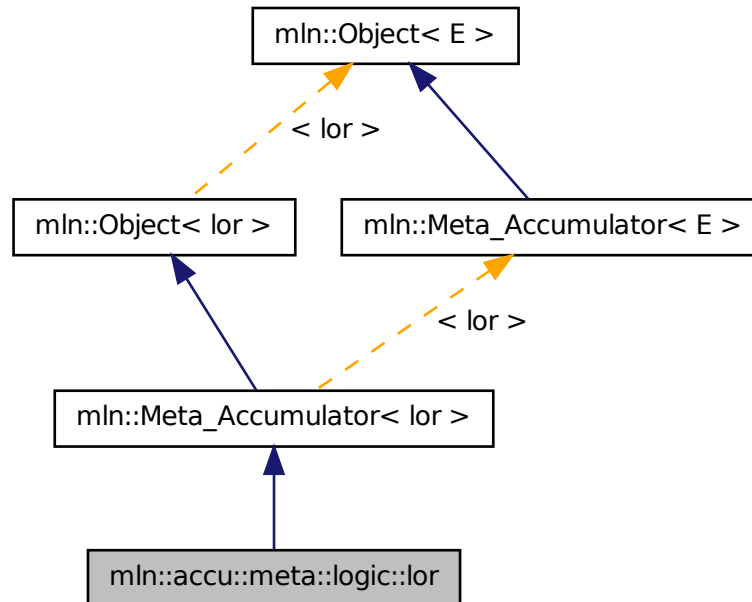
Definition at line 77 of file land\_basic.hh.

## 10.26 mln::accu::meta::logic::lor Struct Reference

Meta accumulator for lor.

```
#include <lor.hh>
```

Inheritance diagram for mln::accu::meta::logic::lor:



### 10.26.1 Detailed Description

Meta accumulator for lor.

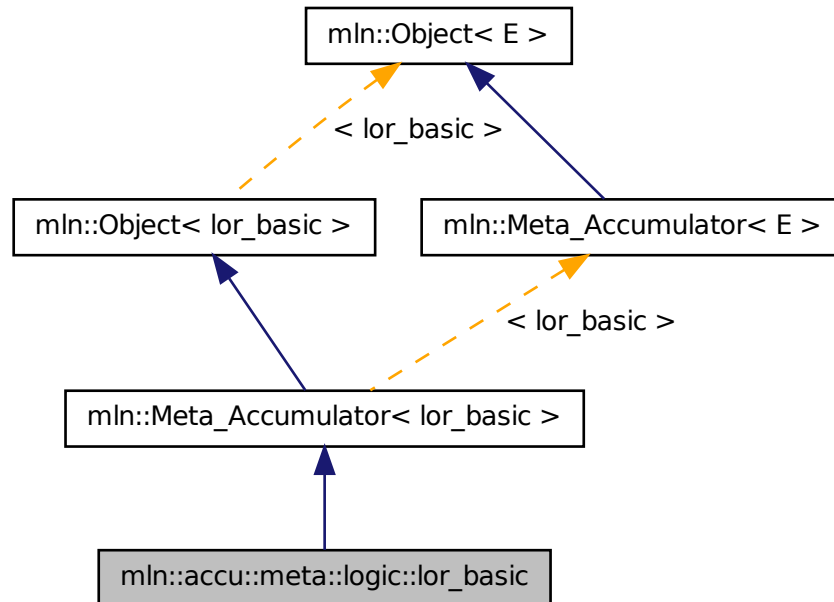
Definition at line 76 of file `accu/logic/lor.hh`.

## 10.27 mln::accu::meta::logic::lor\_basic Struct Reference

Meta accumulator for [lor\\_basic](#).

```
#include <lor_basic.hh>
```

Inheritance diagram for `mln::accu::meta::logic::lor_basic`:



### 10.27.1 Detailed Description

Meta accumulator for [lor\\_basic](#).

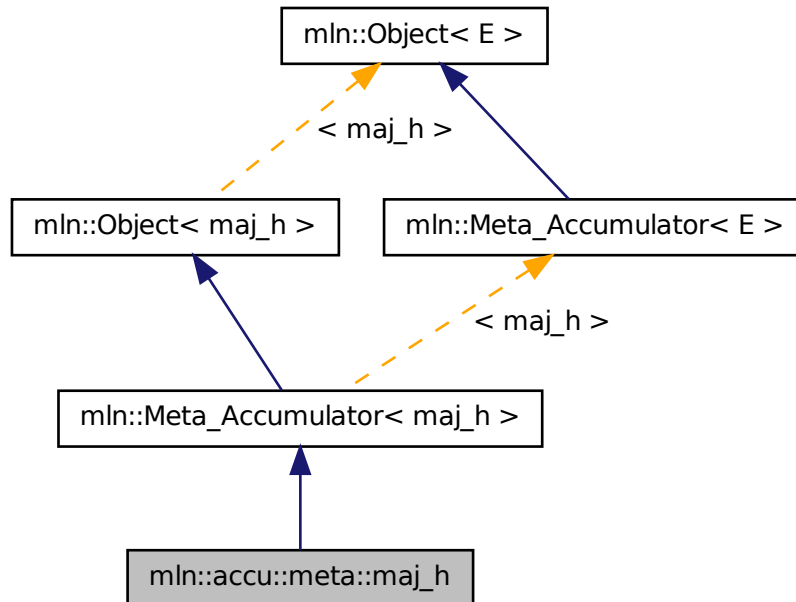
Definition at line 76 of file `lor_basic.hh`.

## 10.28 mln::accu::meta::maj\_h Struct Reference

Meta accumulator for [maj\\_h](#).

```
#include <maj_h.hh>
```

Inheritance diagram for mln::accu::meta::maj\_h:



### 10.28.1 Detailed Description

Meta accumulator for [maj\\_h](#).

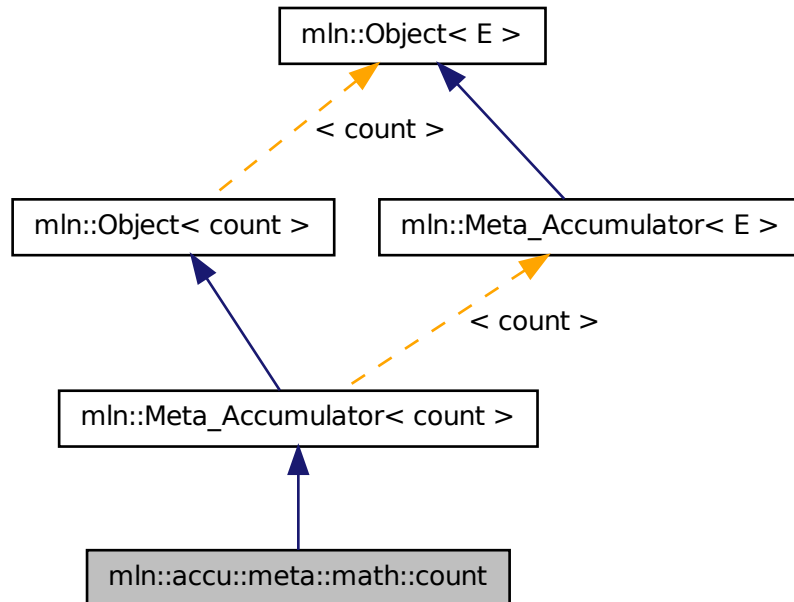
Definition at line 101 of file `maj_h.hh`.

## 10.29 mln::accu::meta::math::count Struct Reference

Meta accumulator for count.

```
#include <count.hh>
```

Inheritance diagram for `mln::accu::meta::math::count`:



### 10.29.1 Detailed Description

Meta accumulator for count.

Definition at line 77 of file count.hh.

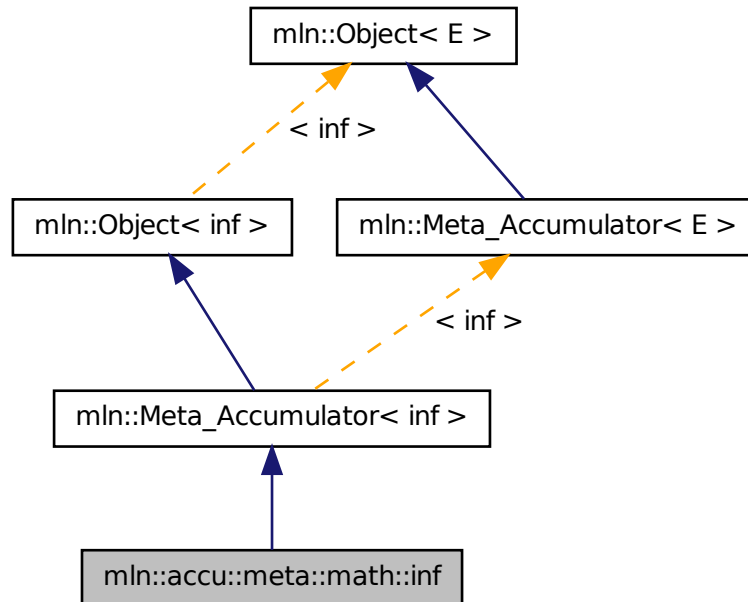
### 10.30 mln::accu::meta::math::inf Struct Reference

Meta accumulator for inf.

```
#include <inf.hh>
```



Inheritance diagram for mln::accu::meta::math::inf:



### 10.30.1 Detailed Description

Meta accumulator for inf.

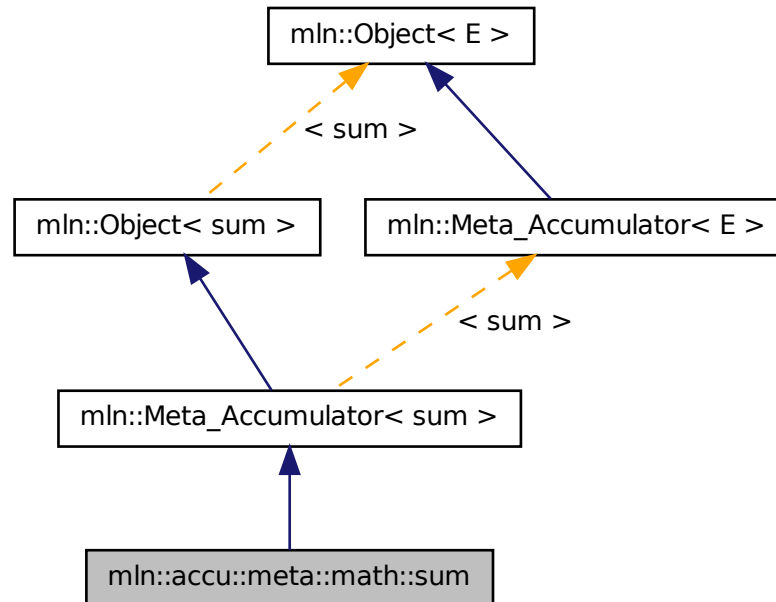
Definition at line 97 of file `accu/math/inf.hh`.

## 10.31 mln::accu::meta::math::sum Struct Reference

Meta accumulator for sum.

```
#include <sum.hh>
```

Inheritance diagram for `mln::accu::meta::math::sum`:



### 10.31.1 Detailed Description

Meta accumulator for sum.

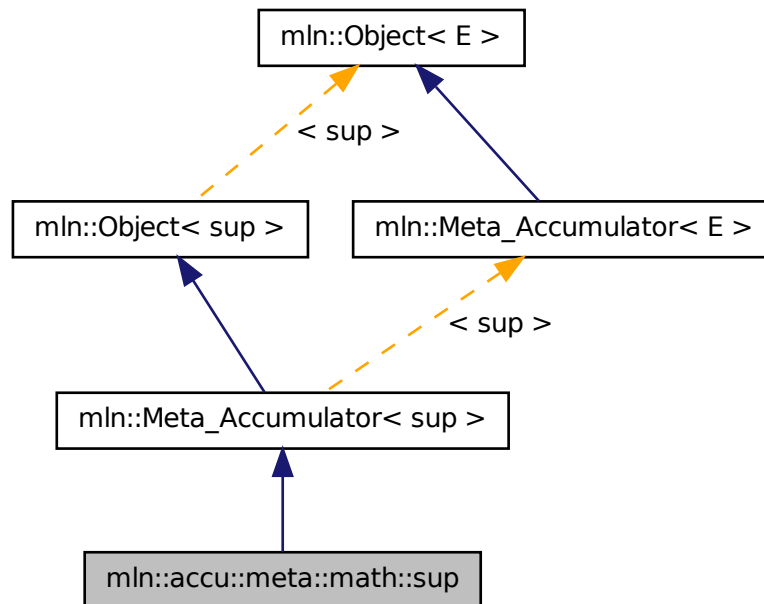
Definition at line 66 of file `accu/math/sum.hh`.

## 10.32 `mln::accu::meta::math::sup` Struct Reference

Meta accumulator for sup.

```
#include <sup.hh>
```

Inheritance diagram for mln::accu::meta::math::sup:



### 10.32.1 Detailed Description

Meta accumulator for `sup`.

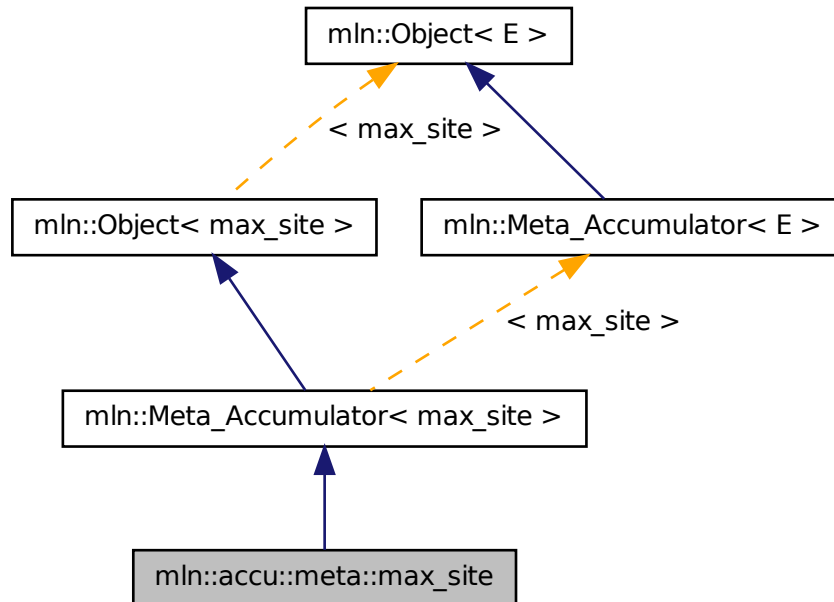
Definition at line 98 of file `accu/math/sup.hh`.

## 10.33 mln::accu::meta::max\_site Struct Reference

Meta accumulator for `max_site`.

```
#include <max_site.hh>
```

Inheritance diagram for `mln::accu::meta::max_site`:



### 10.33.1 Detailed Description

Meta accumulator for [max\\_site](#).

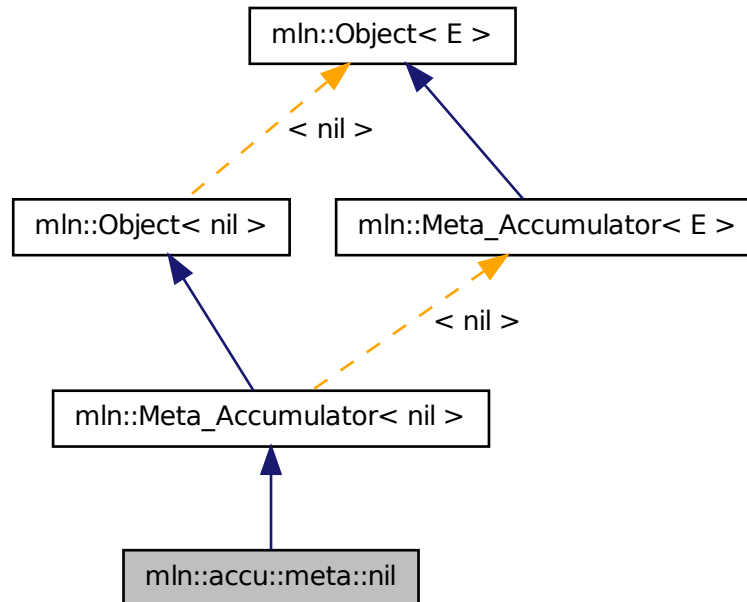
Definition at line 90 of file `max_site.hh`.

## 10.34 mln::accu::meta::nil Struct Reference

Meta accumulator for nil.

```
#include <nil.hh>
```

Inheritance diagram for mln::accu::meta::nil:



### 10.34.1 Detailed Description

Meta accumulator for nil.

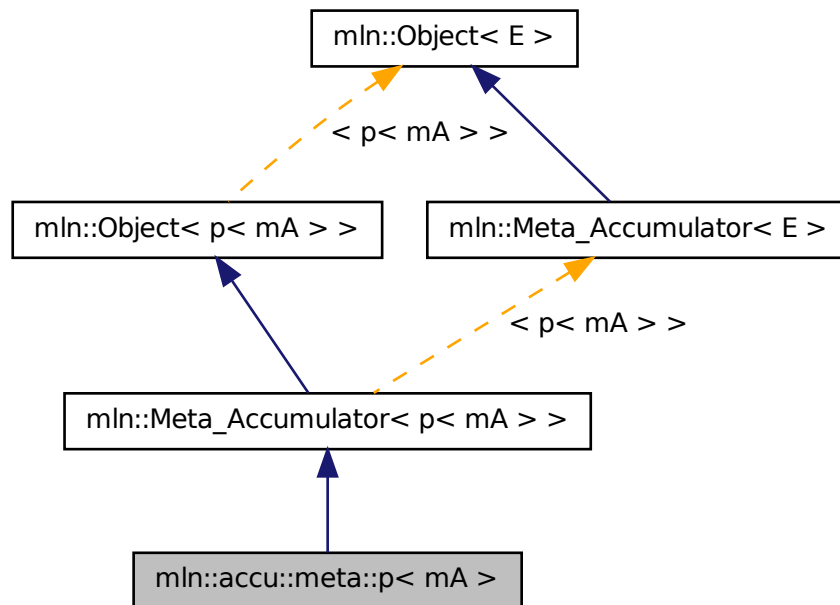
Definition at line 77 of file accu/nil.hh.

## 10.35 mln::accu::meta::p< mA > Struct Template Reference

Meta accumulator for p.

```
#include <p.hh>
```

Inheritance diagram for `mln::accu::meta::p< mA >`:



### 10.35.1 Detailed Description

```
template<typename mA> struct mln::accu::meta::p< mA >
```

Meta accumulator for p.

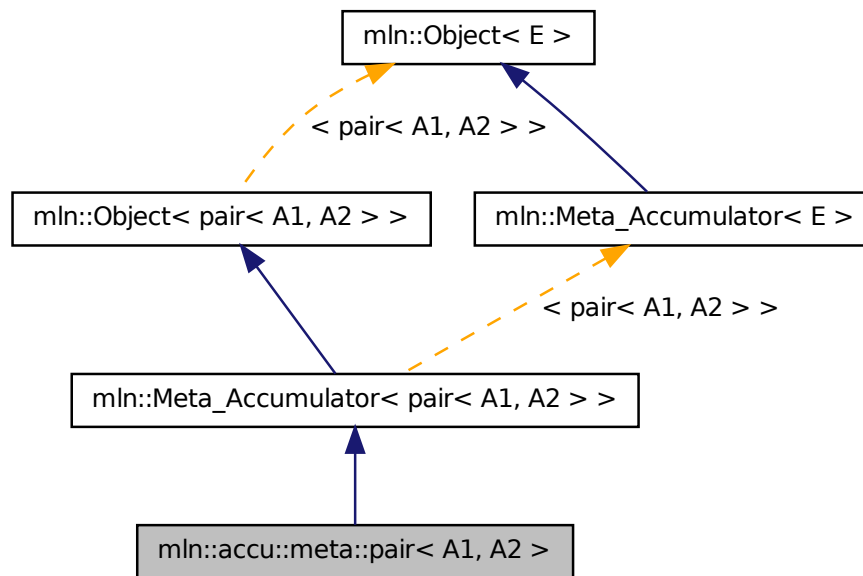
Definition at line 83 of file p.hh.

## 10.36 `mln::accu::meta::pair< A1, A2 >` Struct Template Reference

Meta accumulator for pair.

```
#include <pair.hh>
```

Inheritance diagram for mln::accu::meta::pair< A1, A2 >:



### 10.36.1 Detailed Description

```
template<typename A1, typename A2> struct mln::accu::meta::pair< A1, A2 >
```

Meta accumulator for pair.

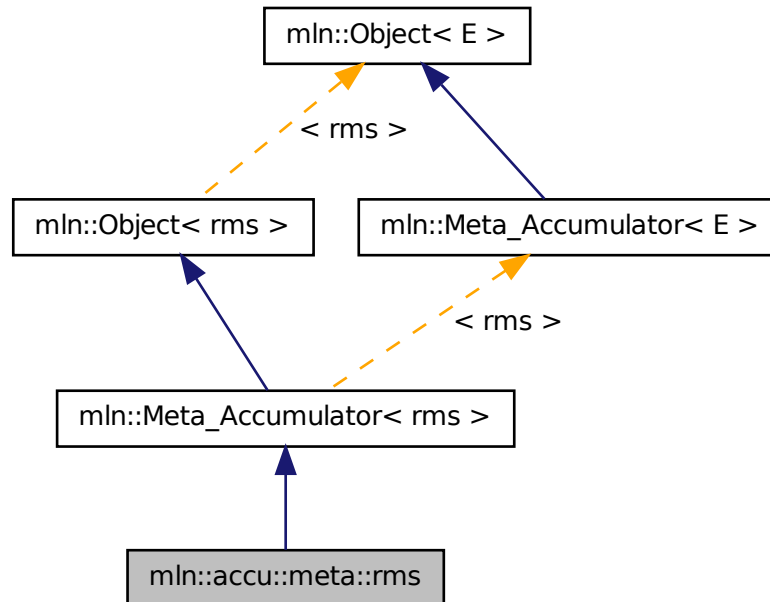
Definition at line 110 of file pair.hh.

## 10.37 mln::accu::meta::rms Struct Reference

Meta accumulator for rms.

```
#include <rms.hh>
```

Inheritance diagram for mln::accu::meta::rms:



### 10.37.1 Detailed Description

Meta accumulator for rms.

Definition at line 88 of file accu/rms.hh.

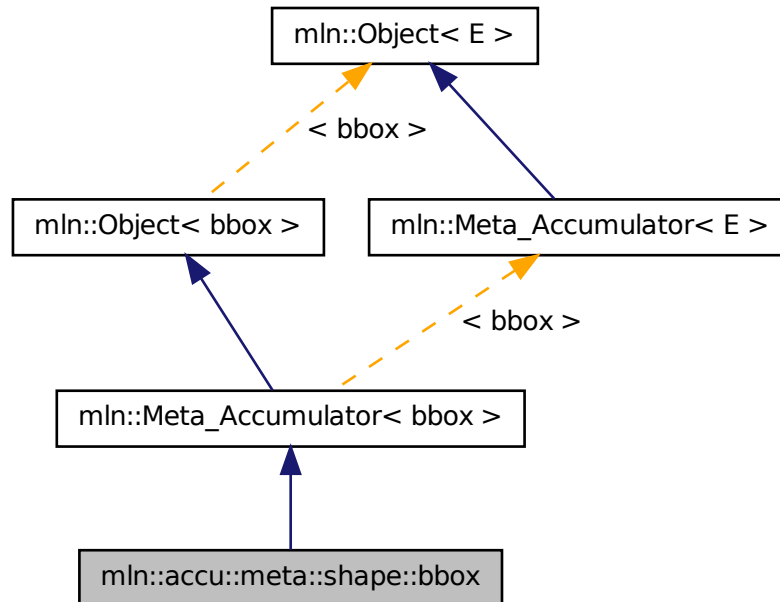
## 10.38 mln::accu::meta::shape::bbox Struct Reference

Meta accumulator for bbox.

```
#include <bbox.hh>
```



Inheritance diagram for mln::accu::meta::shape::bbox:



### 10.38.1 Detailed Description

Meta accumulator for bbox.

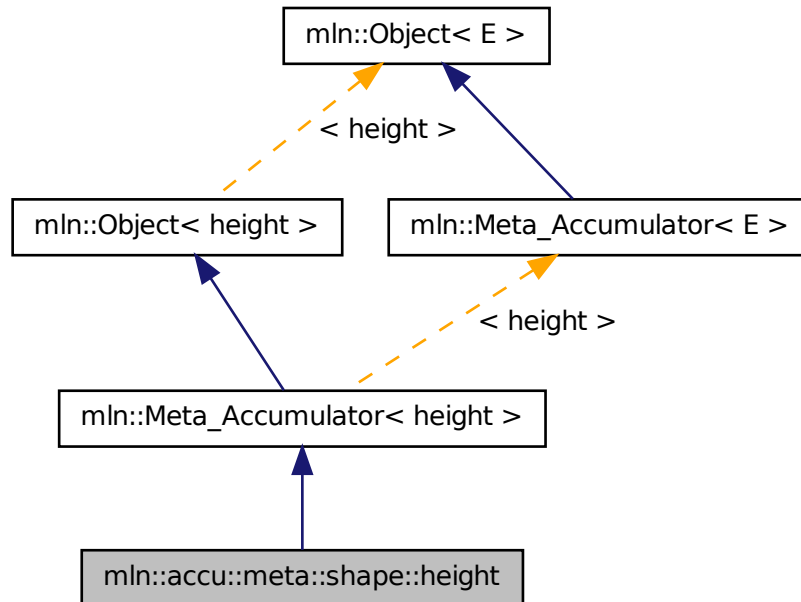
Definition at line 94 of file `accu/shape/bbox.hh`.

## 10.39 mln::accu::meta::shape::height Struct Reference

Meta accumulator for height.

```
#include <height.hh>
```

Inheritance diagram for mln::accu::meta::shape::height:



### 10.39.1 Detailed Description

Meta accumulator for height.

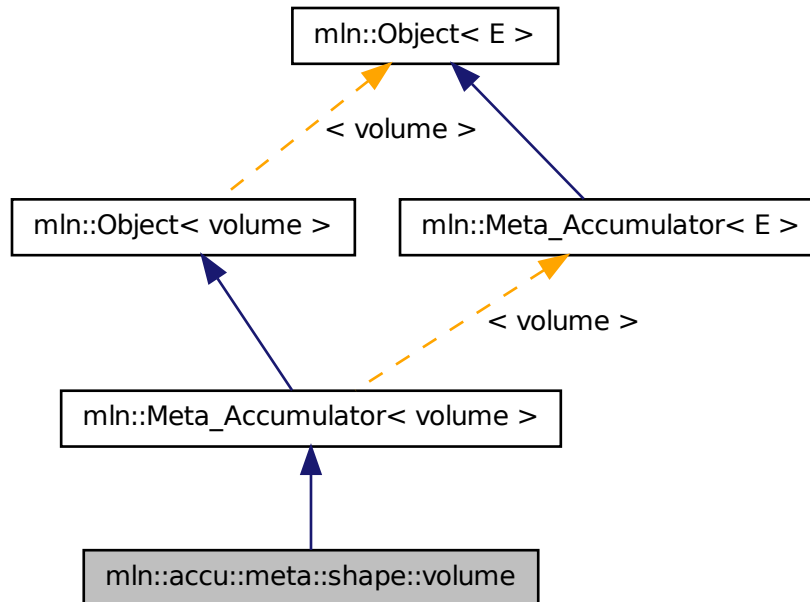
Definition at line 121 of file accu/shape/height.hh.

## 10.40 mln::accu::meta::shape::volume Struct Reference

Meta accumulator for volume.

```
#include <volume.hh>
```

Inheritance diagram for mln::accu::meta::shape::volume:



### 10.40.1 Detailed Description

Meta accumulator for volume.

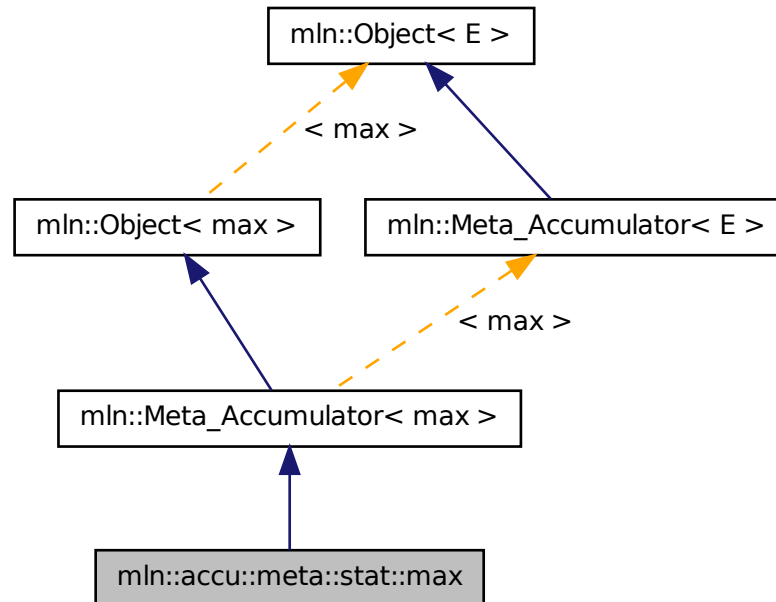
Definition at line 120 of file `accu/shape/volume.hh`.

## 10.41 mln::accu::meta::stat::max Struct Reference

Meta accumulator for max.

```
#include <max.hh>
```

Inheritance diagram for mln::accu::meta::stat::max:



### 10.41.1 Detailed Description

Meta accumulator for max.

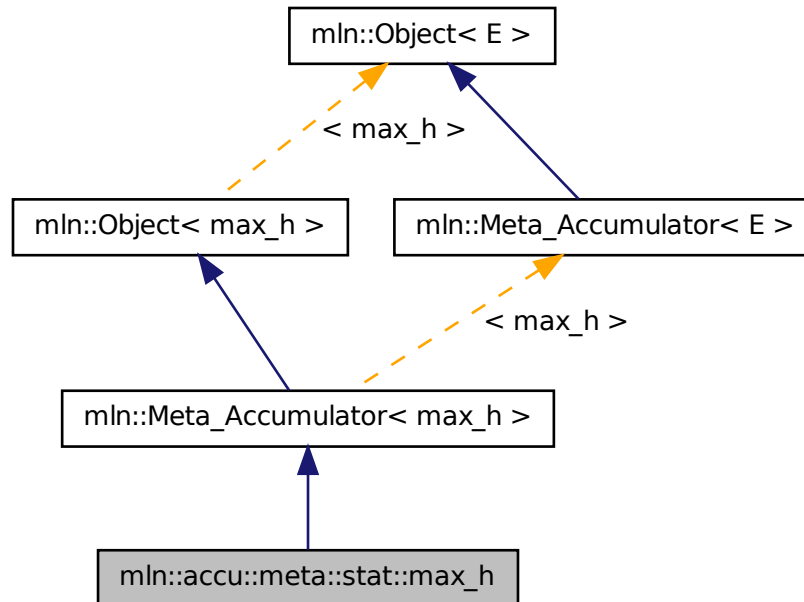
Definition at line 78 of file accu/stat/max.hh.

## 10.42 mln::accu::meta::stat::max\_h Struct Reference

Meta accumulator for max.

```
#include <max_h.hh>
```

Inheritance diagram for mln::accu::meta::stat::max\_h:



### 10.42.1 Detailed Description

Meta accumulator for max.

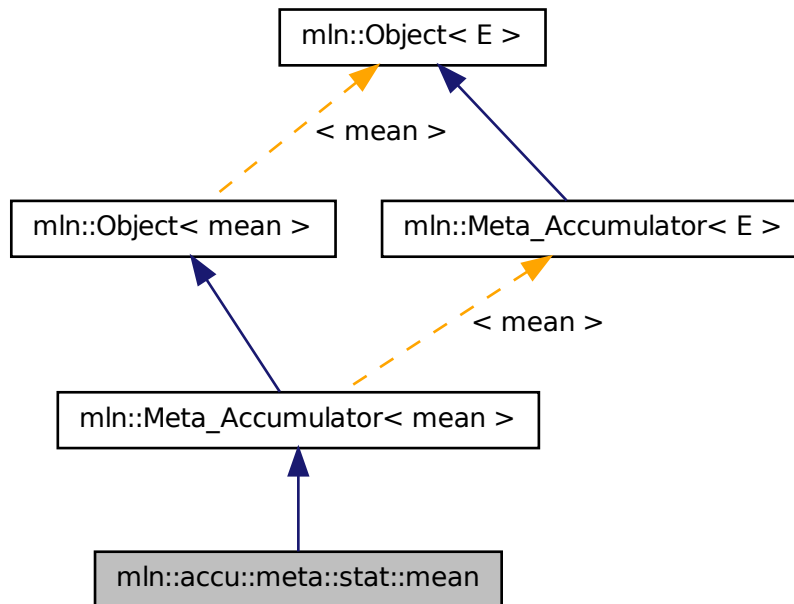
Definition at line 77 of file `max_h.hh`.

## 10.43 mln::accu::meta::stat::mean Struct Reference

Meta accumulator for mean.

```
#include <mean.hh>
```

Inheritance diagram for mln::accu::meta::stat::mean:



### 10.43.1 Detailed Description

Meta accumulator for mean.

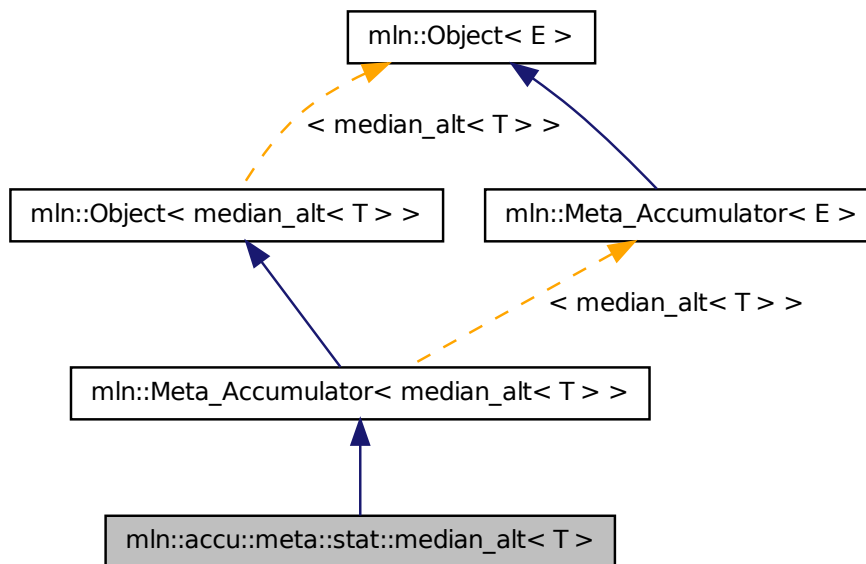
Definition at line 65 of file accu/stat/mean.hh.

## 10.44 mln::accu::meta::stat::median\_alt< T > Struct Template Reference

Meta accumulator for [median\\_alt](#).

```
#include <median_alt.hh>
```

Inheritance diagram for mln::accu::meta::stat::median\_alt< T >:



### 10.44.1 Detailed Description

`template<typename T> struct mln::accu::meta::stat::median_alt< T >`

Meta accumulator for [median\\_alt](#).

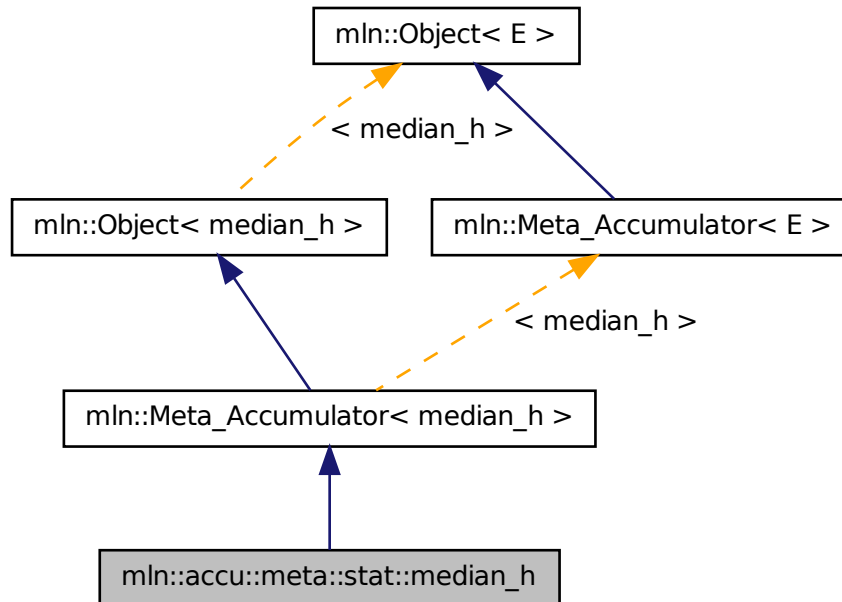
Definition at line 122 of file `median_alt.hh`.

## 10.45 mln::accu::meta::stat::median\_h Struct Reference

Meta accumulator for [median\\_h](#).

```
#include <median_h.hh>
```

Inheritance diagram for mln::accu::meta::stat::median\_h:



### 10.45.1 Detailed Description

Meta accumulator for [median\\_h](#).

Definition at line 61 of file median\_h.hh.

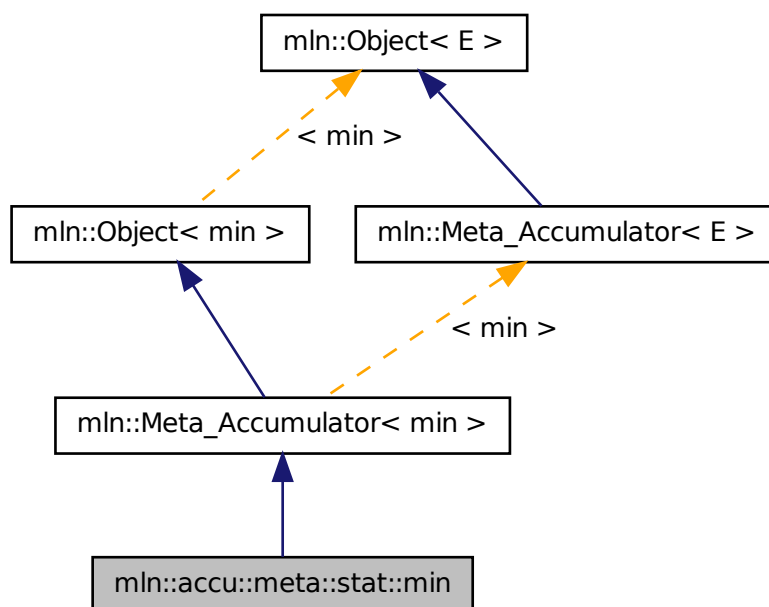
## 10.46 mln::accu::meta::stat::min Struct Reference

Meta accumulator for min.

```
#include <min.hh>
```



Inheritance diagram for mln::accu::meta::stat::min:



### 10.46.1 Detailed Description

Meta accumulator for min.

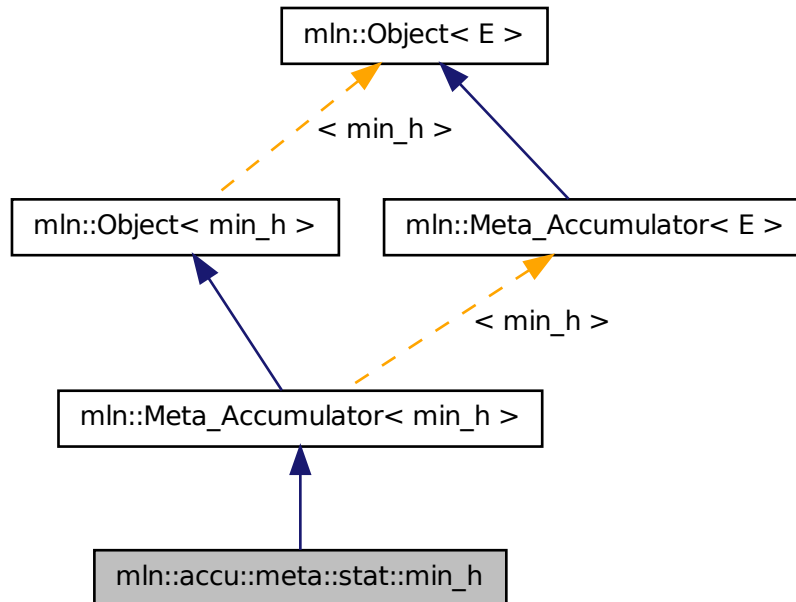
Definition at line 78 of file accu/stat/min.hh.

## 10.47 mln::accu::meta::stat::min\_h Struct Reference

Meta accumulator for min.

```
#include <min_h.hh>
```

Inheritance diagram for mln::accu::meta::stat::min\_h:



### 10.47.1 Detailed Description

Meta accumulator for min.

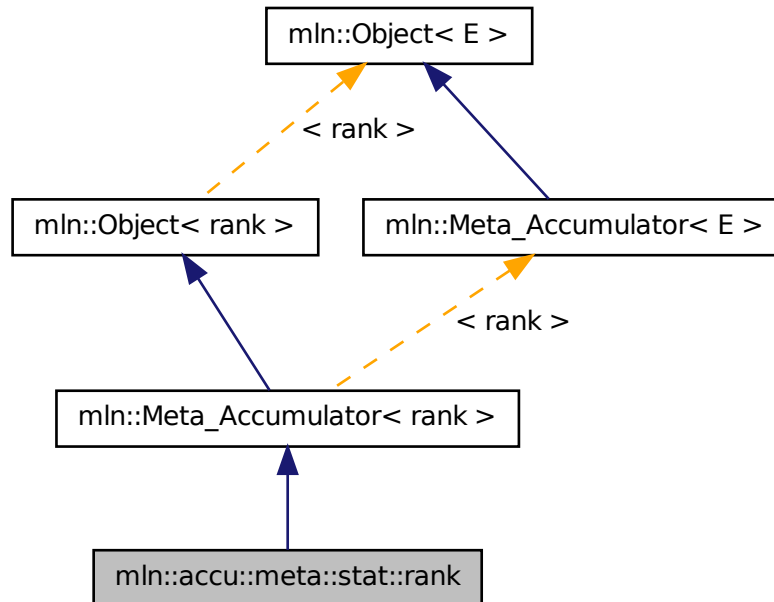
Definition at line 77 of file min\_h.hh.

## 10.48 mln::accu::meta::stat::rank Struct Reference

Meta accumulator for rank.

```
#include <rank.hh>
```

Inheritance diagram for mln::accu::meta::stat::rank:



### 10.48.1 Detailed Description

Meta accumulator for rank.

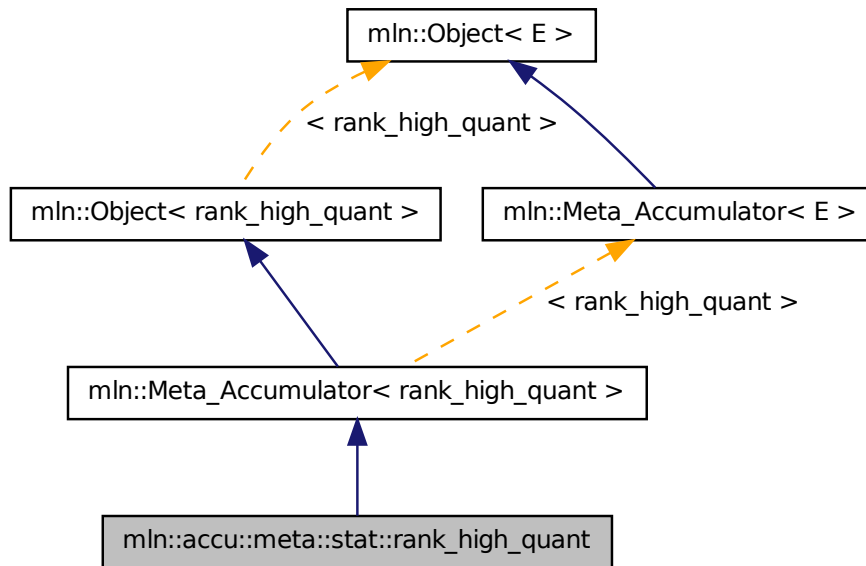
Definition at line 123 of file rank.hh.

## 10.49 mln::accu::meta::stat::rank\_high\_quant Struct Reference

Meta accumulator for [rank\\_high\\_quant](#).

```
#include <rank_high_quant.hh>
```

Inheritance diagram for `mln::accu::meta::stat::rank_high_quant`:



### 10.49.1 Detailed Description

Meta accumulator for [rank\\_high\\_quant](#).

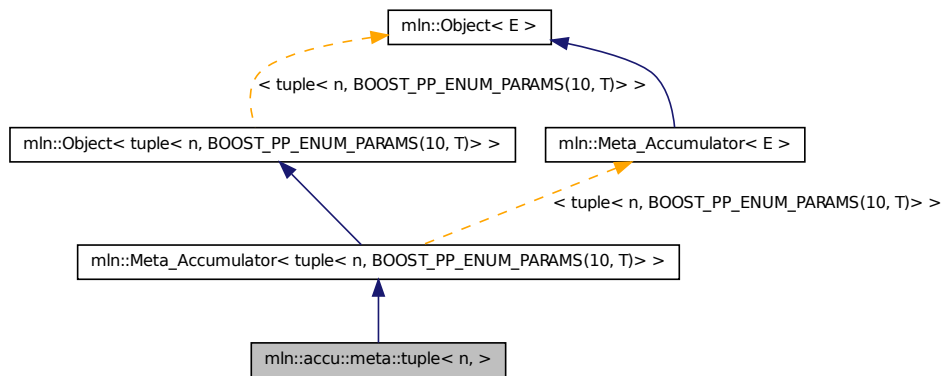
Definition at line 102 of file `rank_high_quant.hh`.

## 10.50 `mln::accu::meta::tuple< n, >` Struct Template Reference

Meta accumulator for tuple.

```
#include <tuple.hh>
```

Inheritance diagram for mln::accu::meta::tuple< n, >:



### 10.50.1 Detailed Description

```
template<unsigned n, BOOST_PP_ENUM_PARAMS_WITH_A_DEFAULT(10, typename T,
boost::tuples::null_type)> struct mln::accu::meta::tuple< n, >
```

Meta accumulator for tuple.

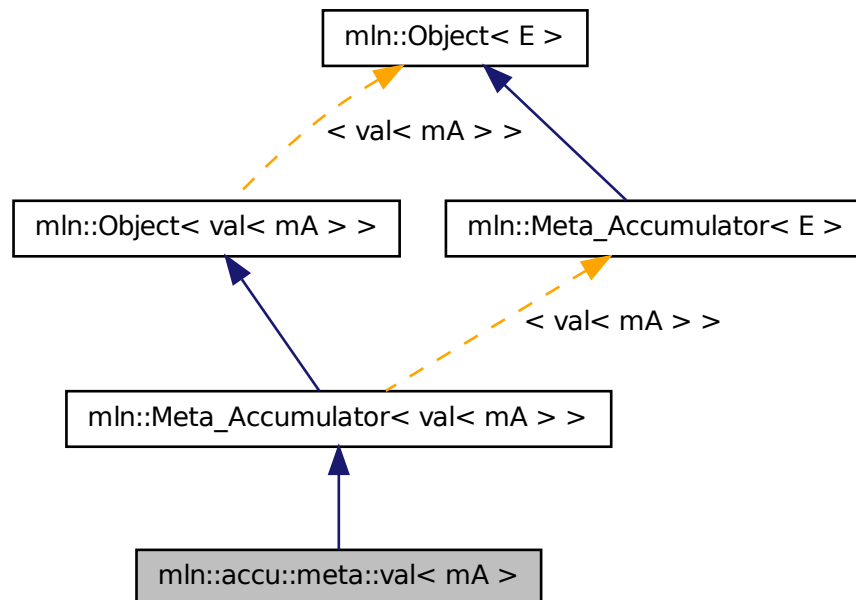
Definition at line 111 of file tuple.hh.

## 10.51 mln::accu::meta::val< mA > Struct Template Reference

Meta accumulator for val.

```
#include <v.hh>
```

Inheritance diagram for `mln::accu::meta::val< mA >`:



### 10.51.1 Detailed Description

```
template<typename mA> struct mln::accu::meta::val< mA >
```

Meta accumulator for val.

Definition at line 87 of file v.hh.

## 10.52 mln::accu::nil< T > Struct Template Reference

Define an accumulator that does nothing.

```
#include <nil.hh>
```

Inherits base< util::ignore, nil< T > >.

### Public Member Functions

- bool [is\\_valid](#) () const  
*Check whether this accu is able to return a result.*
- void [take\\_as\\_init](#) (const T &t)

Take as initialization the value  $t$ .

- void `take_n_times` (unsigned  $n$ , const T & $t$ )

Take  $n$  times the value  $t$ .

- `util::ignore to_result` () const

Get the value of the accumulator.

- void `init` ()

Manipulators.

### 10.52.1 Detailed Description

**template<typename T> struct mln::accu::nil< T >**

Define an accumulator that does nothing.

Definition at line 49 of file accu/nil.hh.

### 10.52.2 Member Function Documentation

**10.52.2.1 template<typename T > void mln::accu::nil< T >::init ( ) [inline]**

Manipulators.

Definition at line 100 of file accu/nil.hh.

**10.52.2.2 template<typename T > bool mln::accu::nil< T >::is\_valid ( ) const [inline]**

Check whether this accu is able to return a result.

Always true here.

Definition at line 136 of file accu/nil.hh.

**10.52.2.3 void mln::Accumulator< nil< T > >::take\_as\_init ( const T &  $t$  ) [inherited]**

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.52.2.4 void mln::Accumulator< nil< T > >::take\_n\_times ( unsigned  $n$ , const T &  $t$  ) [inherited]**

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.52.2.5** `template<typename T > util::ignore mln::accu::nil< T >::to_result ( ) const`  
`[inline]`

Get the value of the accumulator.

Definition at line 128 of file `accu/nil.hh`.

## 10.53 `mln::accu::p< A >` Struct Template Reference

Generic p of accumulators.

```
#include <p.hh>
```

Inherits base< const A::result &, p< A > >.

### Public Member Functions

- bool `is_valid` () const  
*Check whether this accu is able to return a result.*
- void `take_as_init` (const T &t)  
*Take as initialization the value t.*
- void `take_n_times` (unsigned n, const T &t)  
*Take n times the value t.*
- const A::result & `to_result` () const  
*Get the value of the accumulator.*
- void `init` ()  
*Manipulators.*

### 10.53.1 Detailed Description

`template<typename A> struct mln::accu::p< A >`

Generic p of accumulators. The parameter V is the type of values.

Definition at line 50 of file `p.hh`.

### 10.53.2 Member Function Documentation

**10.53.2.1** `template<typename A > void mln::accu::p< A >::init ( ) [inline]`

Manipulators.

Definition at line 115 of file `p.hh`.



**10.53.2.2** `template<typename A > bool mln::accu::p< A >::is_valid ( ) const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 155 of file p.hh.

**10.53.2.3** `void mln::Accumulator< p< A > >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.53.2.4** `void mln::Accumulator< p< A > >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take *n* times the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.53.2.5** `template<typename A > const A::result & mln::accu::p< A >::to_result ( ) const [inline]`

Get the value of the accumulator.

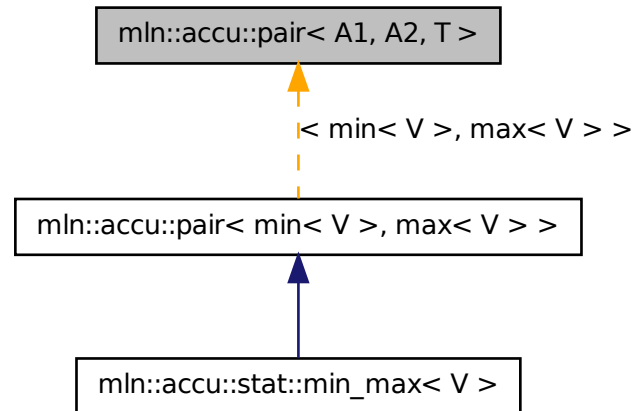
Definition at line 147 of file p.hh.

**10.54** `mln::accu::pair< A1, A2, T > Struct Template Reference`

Generic pair of accumulators.

```
#include <pair.hh>
```

Inheritance diagram for `mln::accu::pair< A1, A2, T >`:



## Public Member Functions

- `A1::result first () const`  
*Return the result of the first accumulator.*
- `A1 first_accu () const`  
*Return the first accumulator.*
- `bool is_valid () const`  
*Check whether this accu is able to return a result.*
- `A2::result second () const`  
*Return the result of the second accumulator.*
- `A2 second_accu () const`  
*Return the second accumulator.*
- `void take_as_init (const T &t)`  
*Take as initialization the value  $t$ .*
- `void take_n_times (unsigned n, const T &t)`  
*Take  $n$  times the value  $t$ .*
- `void init ()`  
*Manipulators.*

- `std::pair< typename A1::result, typename A2::result > to_result () const`  
*Get the value of the accumulator.*

### 10.54.1 Detailed Description

`template<typename A1, typename A2, typename T = mln_argument(A1)> struct mln::accu::pair< A1, A2, T >`

Generic pair of accumulators. The parameter T is the type of values.

Definition at line 58 of file pair.hh.

### 10.54.2 Member Function Documentation

**10.54.2.1** `template<typename A1 , typename A2 , typename T > A1::result mln::accu::pair< A1, A2, T >::first ( ) const [inline]`

Return the result of the first accumulator.

Definition at line 191 of file pair.hh.

**10.54.2.2** `template<typename A1 , typename A2 , typename T > A1 mln::accu::pair< A1, A2, T >::first_accu ( ) const [inline]`

Return the first accumulator.

Definition at line 209 of file pair.hh.

**10.54.2.3** `template<typename A1 , typename A2 , typename T > void mln::accu::pair< A1, A2, T >::init ( ) [inline]`

Manipulators.

Definition at line 136 of file pair.hh.

**10.54.2.4** `template<typename A1 , typename A2 , typename T > bool mln::accu::pair< A1, A2, T >::is_valid ( ) const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 226 of file pair.hh.

**10.54.2.5** `template<typename A1 , typename A2 , typename T > A2::result mln::accu::pair< A1, A2, T >::second ( ) const [inline]`

Return the result of the second accumulator.

Definition at line 199 of file pair.hh.

**10.54.2.6** `template<typename A1 , typename A2 , typename T > A2 mln::accu::pair< A1, A2, T >::second_accu ( ) const [inline]`

Return the second accumulator.

Definition at line 217 of file pair.hh.

**10.54.2.7** `void mln::Accumulator< pair< A1, A2, T > >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.54.2.8** `void mln::Accumulator< pair< A1, A2, T > >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.54.2.9** `template<typename A1 , typename A2 , typename T > std::pair< typename A1::result, typename A2::result > mln::accu::pair< A1, A2, T >::to_result ( ) const [inline]`

Get the value of the accumulator.

Definition at line 172 of file pair.hh.

## 10.55 mln::accu::rms< T, V > Struct Template Reference

Generic root mean square accumulator class.

```
#include <rms.hh>
```

Inherits base< V, rms< T, V > >.

### Public Member Functions

- bool `is_valid` () const  
*Check whether this accu is able to return a result.*
- void `take_as_init` (const T &t)  
*Take as initialization the value `t`.*
- void `take_n_times` (unsigned n, const T &t)  
*Take `n` times the value `t`.*
- V `to_result` () const  
*Get the value of the accumulator.*

- void `init()`  
*Manipulators.*

### 10.55.1 Detailed Description

`template<typename T, typename V> struct mln::accu::rms< T, V >`

Generic root mean square accumulator class. The parameter T is the type of the root mean square value.

Definition at line 52 of file `accu/rms.hh`.

### 10.55.2 Member Function Documentation

**10.55.2.1** `template<typename T , typename V > void mln::accu::rms< T, V >::init ( )`  
`[inline]`

Manipulators.

Definition at line 112 of file `accu/rms.hh`.

References `mln::literal::zero`.

**10.55.2.2** `template<typename T , typename V > bool mln::accu::rms< T, V >::is_valid ( ) const`  
`[inline]`

Check whether this `accu` is able to return a result.

Always true here.

Definition at line 166 of file `accu/rms.hh`.

**10.55.2.3** `void mln::Accumulator< rms< T, V > >::take_as_init ( const T & t )` `[inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.55.2.4** `void mln::Accumulator< rms< T, V > >::take_n_times ( unsigned n, const T & t )`  
`[inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.55.2.5** `template<typename T , typename V > V mln::accu::rms< T, V >::to_result ( ) const`  
`[inline]`

Get the value of the accumulator.

Definition at line 148 of file `accu/rms.hh`.

## 10.56 mln::accu::shape::bbox< P > Struct Template Reference

Generic bounding box accumulator class.

```
#include <bbox.hh>
```

Inherits base< const box< P > &, bbox< P > >.

### Public Member Functions

- bool [is\\_valid](#) () const  
*Check whether this accu is able to return a result.*
- void [take\\_as\\_init](#) (const T &t)  
*Take as initialization the value t.*
- void [take\\_n\\_times](#) (unsigned n, const T &t)  
*Take n times the value t.*
- const [box](#)< P > & [to\\_result](#) () const  
*Get the value of the accumulator.*
- void [init](#) ()  
*Manipulators.*

### 10.56.1 Detailed Description

**template<typename P> struct mln::accu::shape::bbox< P >**

Generic bounding box accumulator class. The parameter P is the type of points.

Definition at line 55 of file accu/shape/bbox.hh.

### 10.56.2 Member Function Documentation

**10.56.2.1 template<typename P > void mln::accu::shape::bbox< P >::init ( ) [inline]**

Manipulators.

Definition at line 124 of file accu/shape/bbox.hh.

**10.56.2.2 template<typename P > bool mln::accu::shape::bbox< P >::is\_valid ( ) const [inline]**

Check whether this accu is able to return a result.

Always true here.

Definition at line 224 of file accu/shape/bbox.hh.

**10.56.2.3 void mln::Accumulator< bbox< P > >::take\_as\_init ( const T & t ) [inherited]**

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.56.2.4 void mln::Accumulator< bbox< P > >::take\_n\_times ( unsigned n, const T & t ) [inherited]**

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.56.2.5 template<typename P > const box< P > & mln::accu::shape::bbox< P >::to\_result ( ) const [inline]**

Get the value of the accumulator.

Definition at line 215 of file `accu/shape/bbox.hh`.

Referenced by `mln::geom::rotate()`.

**10.57 mln::accu::shape::height< I > Struct Template Reference**

Height accumulator.

```
#include <height.hh>
```

Inherits `base< unsigned, height< I > >`.

**Public Types**

- typedef `util::pix< I > argument`  
*The accumulated data type.*
- typedef `argument::value value`  
*The value type associated to the pixel type.*

**Public Member Functions**

- bool `is_valid () const`  
*Check whether this accu is able to return a result.*
- void `take_as_init (const T &t)`  
*Take as initialization the value  $t$ .*
- void `take_n_times (unsigned n, const T &t)`  
*Take  $n$  times the value  $t$ .*
- unsigned `to_result () const`

*Get the value of the accumulator.*

- void `init()`  
*Manipulators.*
- void `set_value(unsigned h)`  
*Force the value of the counter to h.*

### 10.57.1 Detailed Description

`template<typename I> struct mln::accu::shape::height< I >`

Height accumulator. The parameter `I` is the image type on which the accumulator of pixels is built.

Definition at line 68 of file `accu/shape/height.hh`.

### 10.57.2 Member Typedef Documentation

**10.57.2.1** `template<typename I> typedef util::pix<I> mln::accu::shape::height< I >::argument`

The accumulated data type.

The height of component is represented by the height of its root pixel. See `mln::morpho::closing_height` and `mln::morpho::opening_height` for actual uses of this accumulator. **FIXME:** Replaced by `mln::morpho::attribute::height`

Definition at line 78 of file `accu/shape/height.hh`.

**10.57.2.2** `template<typename I> typedef argument::value mln::accu::shape::height< I >::value`

The value type associated to the pixel type.

Definition at line 80 of file `accu/shape/height.hh`.

### 10.57.3 Member Function Documentation

**10.57.3.1** `template<typename I> void mln::accu::shape::height< I >::init( ) [inline]`

Manipulators.

Definition at line 150 of file `accu/shape/height.hh`.

**10.57.3.2** `template<typename I> bool mln::accu::shape::height< I >::is_valid( ) const [inline]`

Check whether this `accu` is able to return a result.

Always true here.

Definition at line 199 of file `accu/shape/height.hh`.



**10.57.3.3** `template<typename I> void mln::accu::shape::height< I >::set_value ( unsigned h )`  
`[inline]`

Force the value of the counter to *h*.

Definition at line 188 of file accu/shape/height.hh.

**10.57.3.4** `void mln::Accumulator< height< I > >::take_as_init ( const T & t )` `[inherited]`

Take as initialization the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.57.3.5** `void mln::Accumulator< height< I > >::take_n_times ( unsigned n, const T & t )`  
`[inherited]`

Take *n* times the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.57.3.6** `template<typename I> unsigned mln::accu::shape::height< I >::to_result ( ) const`  
`[inline]`

Get the value of the accumulator.

Definition at line 180 of file accu/shape/height.hh.

## 10.58 mln::accu::shape::volume< I > Struct Template Reference

Volume accumulator class.

```
#include <volume.hh>
```

Inherits `base< unsigned, volume< I > >`.

### Public Types

- typedef `util::pix< I > argument`  
*The accumulated data type.*
- typedef `argument::value value`  
*The value type associated to the pixel type.*

### Public Member Functions

- bool `is_valid () const`  
*Check whether this accu is able to return a result.*
- void `take_as_init (const T &t)`  
*Take as initialization the value *t*.*

- void [take\\_n\\_times](#) (unsigned n, const T &t)  
*Take n times the value t.*
- unsigned [to\\_result](#) () const  
*Get the value of the accumulator.*
- void [init](#) ()  
*Manipulators.*
- void [set\\_value](#) (unsigned v)  
*Force the value of the counter to v.*

### 10.58.1 Detailed Description

**template<typename I> struct mln::accu::shape::volume< I >**

Volume accumulator class. The parameter I is the image type on which the accumulator of pixels is built. Definition at line 66 of file accu/shape/volume.hh.

### 10.58.2 Member Typedef Documentation

**10.58.2.1 template<typename I> typedef util::pix<I> mln::accu::shape::volume< I >::argument**

The accumulated data type.

The volume of component is represented by the volume of its root pixel. See [mln::morpho::closing\\_volume](#) and [mln::morpho::opening\\_volume](#) for actual uses of this accumulator. FIXME: Replaced by [mln::morpho::attribute::volume](#)

Definition at line 76 of file accu/shape/volume.hh.

**10.58.2.2 template<typename I> typedef argument::value mln::accu::shape::volume< I >::value**

The value type associated to the pixel type.

Definition at line 78 of file accu/shape/volume.hh.

### 10.58.3 Member Function Documentation

**10.58.3.1 template<typename I> void mln::accu::shape::volume< I >::init ( ) [inline]**

Manipulators.

Definition at line 148 of file accu/shape/volume.hh.

References [mln::literal::zero](#).

**10.58.3.2** `template<typename I > bool mln::accu::shape::volume< I >::is_valid ( ) const`  
**[inline]**

Check whether this accu is able to return a result.

Always true here.

Definition at line 215 of file accu/shape/volume.hh.

**10.58.3.3** `template<typename I > void mln::accu::shape::volume< I >::set_value ( unsigned v )`  
**[inline]**

Force the value of the counter to *v*.

Definition at line 204 of file accu/shape/volume.hh.

References mln::literal::zero.

**10.58.3.4** `void mln::Accumulator< volume< I > >::take_as_init ( const T & t )` **[inherited]**

Take as initialization the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.58.3.5** `void mln::Accumulator< volume< I > >::take_n_times ( unsigned n, const T & t )`  
**[inherited]**

Take *n* times the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.58.3.6** `template<typename I > unsigned mln::accu::shape::volume< I >::to_result ( ) const`  
**[inline]**

Get the value of the accumulator.

Definition at line 196 of file accu/shape/volume.hh.

## 10.59 mln::accu::site\_set::rectangularity< P > Class Template Reference

Compute the rectangularity of a site set.

```
#include <rectangularity.hh>
```

Inherits couple< accu::shape::bbox< P >, accu::math::count< P >, float, rectangularity< P > >.

### Public Member Functions

- A2::result [area](#) ( ) const

*Return the site set area.*

- `A1::result bbox () const`  
*Return the site set bounding box.*
- `rectangularity ()`  
*Constructor.*
- `template<typename T >`  
`void take_as_init (const T &t)`  
*Take as initialization the value  $t$ .*
- `template<typename T >`  
`void take_n_times (unsigned n, const T &t)`  
*Take  $n$  times the value  $t$ .*
- `result to_result () const`  
*Return the rectangularity value.*

### 10.59.1 Detailed Description

`template<typename P> class mln::accu::site_set::rectangularity< P >`

Compute the rectangularity of a site set.

Definition at line 51 of file rectangularity.hh.

### 10.59.2 Constructor & Destructor Documentation

**10.59.2.1** `template<typename P > mln::accu::site_set::rectangularity< P >::rectangularity ( )`  
`[inline]`

Constructor.

Definition at line 91 of file rectangularity.hh.

### 10.59.3 Member Function Documentation

**10.59.3.1** `template<typename P > rectangularity< P >::A2::result`  
`mln::accu::site_set::rectangularity< P >::area ( ) const [inline]`

Return the site set area.

Definition at line 107 of file rectangularity.hh.

**10.59.3.2** `template<typename P > rectangularity< P >::A1::result`  
`mln::accu::site_set::rectangularity< P >::bbox ( ) const [inline]`

Return the site set bounding box.

Definition at line 98 of file rectangularity.hh.

**10.59.3.3** `template<typename E > template<typename T > void mln::Accumulator< E >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Definition at line 186 of file `accumulator.hh`.

References `mln::mln_exact()`.

**10.59.3.4** `template<typename E > template<typename T > void mln::Accumulator< E >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Definition at line 213 of file `accumulator.hh`.

References `mln::mln_exact()`.

**10.59.3.5** `template<typename P > rectangularity< P >::result mln::accu::site_set::rectangularity< P >::to_result ( ) const [inline]`

Return the rectangularity value.

Definition at line 116 of file `rectangularity.hh`.

## 10.60 mln::accu::stat::deviation< T, S, M > Struct Template Reference

Generic standard deviation accumulator class.

```
#include <deviation.hh>
```

Inherits `base< M, deviation< T, S, M > >`.

### Public Member Functions

- `bool is_valid () const`  
*Check whether this `accu` is able to return a result.*
- `void take_as_init (const T &t)`  
*Take as initialization the value  $t$ .*
- `void take_n_times (unsigned n, const T &t)`  
*Take  $n$  times the value  $t$ .*
- `M to_result () const`  
*Get the value of the accumulator.*

- void `init()`  
*Manipulators.*

### 10.60.1 Detailed Description

**template<typename T, typename S = typename mln::value::props< T >::sum, typename M = S>  
struct mln::accu::stat::deviation< T, S, M >**

Generic standard deviation accumulator class. Parameter T is the type of values that we sum. Parameter S is the type to store the standard deviation; the default type of S is the summation type (property) of T. Parameter M is the type of the mean value; the default type of M is S.

Definition at line 62 of file deviation.hh.

### 10.60.2 Member Function Documentation

**10.60.2.1 template<typename T, typename S, typename M > void mln::accu::stat::deviation< T, S, M >::init( ) [inline]**

Manipulators.

Definition at line 132 of file deviation.hh.

**10.60.2.2 template<typename T, typename S, typename M > bool mln::accu::stat::deviation< T, S, M >::is\_valid( ) const [inline]**

Check whether this accu is able to return a result.

Always true here.

Definition at line 177 of file deviation.hh.

**10.60.2.3 void mln::Accumulator< deviation< T, S, M > >::take\_as\_init( const T & t ) [inherited]**

Take as initialization the value t.

Dev note: this is a final method; override if needed by take\_as\_init\_ (ending with '\_').

**10.60.2.4 void mln::Accumulator< deviation< T, S, M > >::take\_n\_times( unsigned n, const T & t ) [inherited]**

Take n times the value t.

Dev note: this is a final method; override if needed by take\_as\_init\_ (ending with '\_').

**10.60.2.5 template<typename T, typename S, typename M > M mln::accu::stat::deviation< T, S, M >::to\_result( ) const [inline]**

Get the value of the accumulator.

Definition at line 159 of file deviation.hh.

## 10.61 mln::accu::stat::histo3d\_rgb< V > Struct Template Reference

Define a histogram as accumulator which returns an [image3d](#).

```
#include <histo3d_rgb.hh>
```

Inherits base< [image3d](#)< unsigned >, [histo3d\\_rgb](#)< V > >.

### Public Member Functions

- [bool is\\_valid](#) () const  
*Check whether this accumulator is able to return a result.*
- [void take\\_as\\_init](#) (const T &t)  
*Take as initialization the value t.*
- [void take\\_n\\_times](#) (unsigned n, const T &t)  
*Take n times the value t.*
- [histo3d\\_rgb](#) ()  
*Constructors.*
- [void init](#) ()  
*Manipulators.*
- [void take](#) (const argument &t)  
*Update the histogram with the RGB pixel t.*
- [void take](#) (const [histo3d\\_rgb](#)< V > &other)  
*Update the histogram with an other histogram.*
- [result to\\_result](#) () const  
*Accessors.*

### 10.61.1 Detailed Description

```
template<typename V> struct mln::accu::stat::histo3d_rgb< V >
```

Define a histogram as accumulator which returns an [image3d](#). Param V defines the type of the input image value. It is in this space that we count the values. For instance, this histogram works well for [image2d](#)< rgb<2> > or with [image2d](#)< rgb<7> >. The number of bins depends directly the values V. For 8 bits there is 256x3 bins. Note that less quantification works too.

Definition at line 167 of file [histo3d\\_rgb.hh](#).

## 10.61.2 Constructor & Destructor Documentation

**10.61.2.1** `template<typename V > mln::accu::stat::histo3d_rgb< V >::histo3d_rgb ( )`  
`[inline]`

Constructors.

Infer the size of the resulting `image3d` domain. By evaluating the minimum and the maximum of `V`, we define the domain of the resulting `image3d`.

Definition at line 244 of file `histo3d_rgb.hh`.

## 10.61.3 Member Function Documentation

**10.61.3.1** `template<typename V > void mln::accu::stat::histo3d_rgb< V >::init ( )` `[inline]`

Manipulators.

Initialize the histogram with zero value. This method must be called just before starting the use of the histogram. If it's not, resulting values won't converge to the density.

Definition at line 268 of file `histo3d_rgb.hh`.

References `mln::data::fill()`, and `mln::literal::zero`.

**10.61.3.2** `template<typename V > bool mln::accu::stat::histo3d_rgb< V >::is_valid ( ) const`  
`[inline]`

Check whether this accumulator is able to return a result.

Depends if the resulting `image1d` is valid. We can assume it is quite always the case.

Definition at line 307 of file `histo3d_rgb.hh`.

**10.61.3.3** `template<typename V > void mln::accu::stat::histo3d_rgb< V >::take ( const`  
`argument & t )` `[inline]`

Update the histogram with the RGB pixel `t`.

### Parameters

`[in]` `t` a graylevel pixel of type `V`.

The end user shouldn't call this method. In place of it, he can go through the data compute interface.

Definition at line 275 of file `histo3d_rgb.hh`.

**10.61.3.4** `template<typename V > void mln::accu::stat::histo3d_rgb< V >::take ( const`  
`histo3d_rgb< V > & other )` `[inline]`

Update the histogram with an other histogram.

### Parameters

`[in]` `other` the other histogram.



The end user shouldn't call this method. This is part of data compute interface mechanism.

Definition at line 286 of file histo3d\_rgb.hh.

**10.61.3.5** void mln::Accumulator< histo3d\_rgb< V > >::take\_as\_init ( const T & t )  
[inherited]

Take as initialization the value t.

Dev note: this is a final method; override if needed by take\_as\_init\_ (ending with '\_').

**10.61.3.6** void mln::Accumulator< histo3d\_rgb< V > >::take\_n\_times ( unsigned n, const T & t ) [inherited]

Take n times the value t.

Dev note: this is a final method; override if needed by take\_as\_init\_ (ending with '\_').

**10.61.3.7** template<typename V > histo3d\_rgb< V >::result mln::accu::stat::histo3d\_rgb< V >::to\_result ( ) const [inline]

Accessors.

Return the histogram as an RGB [image3d](#). This is the machinery to communicate with data compute interface. The end user should'nt use it.

Definition at line 293 of file histo3d\_rgb.hh.

## 10.62 mln::accu::stat::max< T > Struct Template Reference

Generic max accumulator class.

```
#include <max.hh>
```

Inherits base< const T &, max< T > >.

### Public Member Functions

- bool [is\\_valid](#) () const  
*Check whether this accu is able to return a result.*
- void [set\\_value](#) (const T &t)  
*Force the value of the min to t.*
- void [take\\_as\\_init](#) (const T &t)  
*Take as initialization the value t.*
- void [take\\_n\\_times](#) (unsigned n, const T &t)  
*Take n times the value t.*
- const T & [to\\_result](#) () const  
*Get the value of the accumulator.*

- void `init()`  
*Manipulators.*

### 10.62.1 Detailed Description

**template<typename T> struct mln::accu::stat::max< T >**

Generic max accumulator class. The parameter T is the type of values.

Definition at line 101 of file `accu/stat/max.hh`.

### 10.62.2 Member Function Documentation

**10.62.2.1 template<typename T > void mln::accu::stat::max< T >::init ( ) [inline]**

Manipulators.

Definition at line 146 of file `accu/stat/max.hh`.

**10.62.2.2 template<typename T > bool mln::accu::stat::max< T >::is\_valid ( ) const [inline]**

Check whether this accu is able to return a result.

Always true here.

Definition at line 196 of file `accu/stat/max.hh`.

**10.62.2.3 template<typename T > void mln::accu::stat::max< T >::set\_value ( const T & t ) [inline]**

Force the value of the min to *t*.

Definition at line 180 of file `accu/stat/max.hh`.

**10.62.2.4 void mln::Accumulator< max< T > >::take\_as\_init ( const T & t ) [inherited]**

Take as initialization the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.62.2.5 void mln::Accumulator< max< T > >::take\_n\_times ( unsigned n, const T & t ) [inherited]**

Take *n* times the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

### 10.62.2.6 `template<typename T > const T & mln::accu::stat::max< T >::to_result ( ) const [inline]`

Get the value of the accumulator.

Definition at line 188 of file `accu/stat/max.hh`.

## 10.63 `mln::accu::stat::max_h< V > Struct Template Reference`

Generic max function based on histogram over a value set with type `V`.

```
#include <max_h.hh>
```

Inherits `base< const V &, max_h< V > >`.

### Public Member Functions

- `bool is_valid () const`  
*Check whether this `accu` is able to return a result.*
- `void take_as_init (const T &t)`  
*Take as initialization the value `t`.*
- `void take_n_times (unsigned n, const T &t)`  
*Take `n` times the value `t`.*
- `const argument & to_result () const`  
*Get the value of the accumulator.*
- `void init ()`  
*Manipulators.*

### 10.63.1 Detailed Description

```
template<typename V> struct mln::accu::stat::max_h< V >
```

Generic max function based on histogram over a value set with type `V`.

Definition at line 100 of file `max_h.hh`.

### 10.63.2 Member Function Documentation

#### 10.63.2.1 `template<typename V > void mln::accu::stat::max_h< V >::init ( ) [inline]`

Manipulators.

Definition at line 280 of file `max_h.hh`.

### 10.63.2.2 `template<typename V > bool mln::accu::stat::max_h< V >::is_valid ( ) const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 323 of file max\_h.hh.

### 10.63.2.3 `void mln::Accumulator< max_h< V > >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

### 10.63.2.4 `void mln::Accumulator< max_h< V > >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

### 10.63.2.5 `template<typename V > const max_h< V >::argument & mln::accu::stat::max_h< V >::to_result ( ) const [inline]`

Get the value of the accumulator.

Definition at line 304 of file max\_h.hh.

## 10.64 `mln::accu::stat::mean< T, S, M >` Struct Template Reference

Generic mean accumulator class.

```
#include <mean.hh>
```

Inherits `base< M, mean< T, S, M > >`.

### Public Member Functions

- `accu::math::count< T >::result count () const`  
*Get the cardinality.*
- `bool is_valid () const`  
*Check whether this accu is able to return a result.*
- `accu::math::sum< T >::result sum () const`  
*Get the sum of values.*
- `void take_as_init (const T &t)`  
*Take as initialization the value  $t$ .*
- `void take_n_times (unsigned n, const T &t)`

Take  $n$  times the value  $\tau$ .

- M `to_result()` const  
Get the value of the accumulator.
- void `init()`  
Manipulators.

### 10.64.1 Detailed Description

**template<typename T, typename S = typename mln::value::props< T >::sum, typename M = S>  
struct mln::accu::stat::mean< T, S, M >**

Generic mean accumulator class. Parameter `T` is the type of values that we sum. Parameter `S` is the type to store the sum of values; the default type of `S` is the summation type (property) of `T`. Parameter `M` is the type of the mean value; the default type of `M` is `S`.

Definition at line 119 of file `accu/stat/mean.hh`.

### 10.64.2 Member Function Documentation

**10.64.2.1 template<typename T, typename S, typename M > mln::math::count< T >::result  
mln::accu::stat::mean< T, S, M >::count ( ) const [inline]**

Get the cardinality.

Definition at line 242 of file `accu/stat/mean.hh`.

**10.64.2.2 template<typename T, typename S, typename M > void mln::accu::stat::mean< T, S,  
M >::init ( ) [inline]**

Manipulators.

Definition at line 173 of file `accu/stat/mean.hh`.

**10.64.2.3 template<typename T, typename S, typename M > bool mln::accu::stat::mean< T, S,  
M >::is\_valid ( ) const [inline]**

Check whether this `accu` is able to return a result.

Always true here.

Definition at line 234 of file `accu/stat/mean.hh`.

**10.64.2.4 template<typename T, typename S, typename M > mln::math::sum< T >::result  
mln::accu::stat::mean< T, S, M >::sum ( ) const [inline]**

Get the sum of values.

Definition at line 251 of file `accu/stat/mean.hh`.

**10.64.2.5** `void mln::Accumulator< mean< T, S, M > >::take_as_init ( const T & t )`  
**[inherited]**

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.64.2.6** `void mln::Accumulator< mean< T, S, M > >::take_n_times ( unsigned n, const T & t )`  
**[inherited]**

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.64.2.7** `template<typename T, typename S, typename M > M mln::accu::stat::mean< T, S, M >::to_result ( ) const` **[inline]**

Get the value of the accumulator.

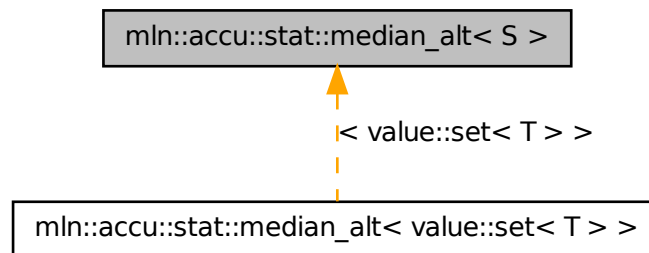
Definition at line 216 of file `accu/stat/mean.hh`.

## 10.65 `mln::accu::stat::median_alt< S >` Struct Template Reference

Generic `median_alt` function based on histogram over a value set with type `S`.

```
#include <median_alt.hh>
```

Inheritance diagram for `mln::accu::stat::median_alt< S >`:



### Public Member Functions

- `bool is_valid ( ) const`  
*Check whether this `accu` is able to return a result.*
- `void take_as_init ( const T &t)`

Take as initialization the value  $t$ .

- void [take\\_n\\_times](#) (unsigned  $n$ , const T & $t$ )

Take  $n$  times the value  $t$ .

- const argument & [to\\_result](#) () const

Get the value of the accumulator.

- void [take](#) (const argument & $t$ )

Manipulators.

### 10.65.1 Detailed Description

`template<typename S> struct mln::accu::stat::median_alt< S >`

Generic [median\\_alt](#) function based on histogram over a value set with type  $S$ .

Definition at line 54 of file median\_alt.hh.

### 10.65.2 Member Function Documentation

**10.65.2.1** `template<typename S > bool mln::accu::stat::median_alt< S >::is_valid ( ) const`  
**[inline]**

Check whether this accu is able to return a result.

Always true here.

Definition at line 282 of file median\_alt.hh.

**10.65.2.2** `template<typename S > void mln::accu::stat::median_alt< S >::take ( const argument`  
**& t ) [inline]**

Manipulators.

Definition at line 165 of file median\_alt.hh.

**10.65.2.3** `void mln::Accumulator< median_alt< S > >::take_as_init ( const T & t )`  
**[inherited]**

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.65.2.4** `void mln::Accumulator< median_alt< S > >::take_n_times ( unsigned n, const T & t`  
**) [inherited]**

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

### 10.65.2.5 `template<typename S > const median_alt< S >::argument & mln::accu::stat::median_alt< S >::to_result ( ) const [inline]`

Get the value of the accumulator.

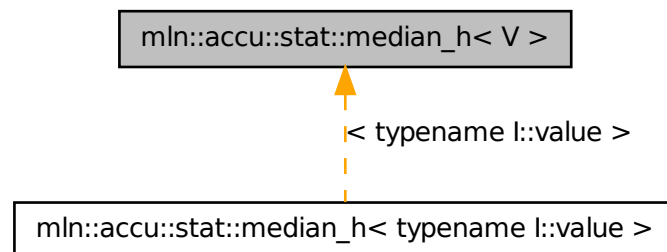
Definition at line 274 of file median\_alt.hh.

## 10.66 `mln::accu::stat::median_h< V > Struct Template Reference`

Generic median function based on histogram over a value set with type `V`.

```
#include <median_h.hh>
```

Inheritance diagram for `mln::accu::stat::median_h< V >`:



### Public Member Functions

- `bool is_valid () const`  
*Check whether this accu is able to return a result.*
- `void take_as_init (const T &t)`  
*Take as initialization the value  $\tau$ .*
- `void take_n_times (unsigned n, const T &t)`  
*Take  $n$  times the value  $\tau$ .*
- `const argument & to_result () const`  
*Get the value of the accumulator.*
- `void init ()`  
*Manipulators.*



### 10.66.1 Detailed Description

```
template<typename V> struct mln::accu::stat::median_h< V >
```

Generic median function based on histogram over a value set with type V.

Definition at line 82 of file median\_h.hh.

### 10.66.2 Member Function Documentation

**10.66.2.1** `template<typename V > void mln::accu::stat::median_h< V >::init ( ) [inline]`

Manipulators.

Definition at line 267 of file median\_h.hh.

**10.66.2.2** `template<typename V > bool mln::accu::stat::median_h< V >::is_valid ( ) const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 299 of file median\_h.hh.

**10.66.2.3** `void mln::Accumulator< median_h< V > >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value t.

Dev note: this is a final method; override if needed by take\_as\_init\_ (ending with '\_').

**10.66.2.4** `void mln::Accumulator< median_h< V > >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take n times the value t.

Dev note: this is a final method; override if needed by take\_as\_init\_ (ending with '\_').

**10.66.2.5** `template<typename V > const median_h< V >::argument & mln::accu::stat::median_h< V >::to_result ( ) const [inline]`

Get the value of the accumulator.

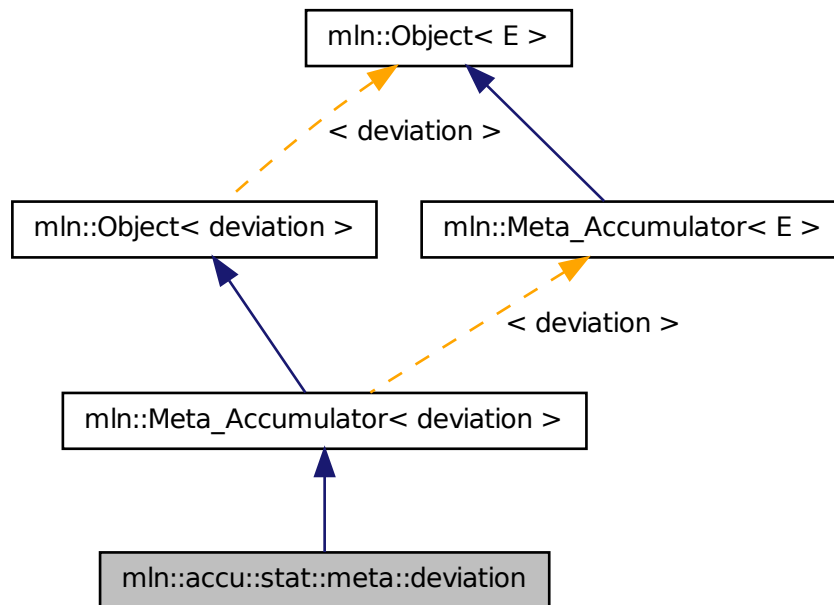
Definition at line 281 of file median\_h.hh.

## 10.67 mln::accu::stat::meta::deviation Struct Reference

Meta accumulator for deviation.

```
#include <deviation.hh>
```

Inheritance diagram for mln::accu::stat::meta::deviation:



### 10.67.1 Detailed Description

Meta accumulator for deviation.

Definition at line 105 of file deviation.hh.

## 10.68 mln::accu::stat::min< T > Struct Template Reference

Generic min accumulator class.

```
#include <min.hh>
```

Inherits base< const T &, min< T > >.

### Public Member Functions

- bool [is\\_valid](#) () const  
*Check whether this accu is able to return a result.*
- void [set\\_value](#) (const T &t)  
*Force the value of the min to t.*

- void `take_as_init` (const T &t)  
*Take as initialization the value  $t$ .*
- void `take_n_times` (unsigned n, const T &t)  
*Take  $n$  times the value  $t$ .*
- const T & `to_result` () const  
*Get the value of the accumulator.*
- void `init` ()  
*Manipulators.*

### 10.68.1 Detailed Description

`template<typename T> struct mln::accu::stat::min< T >`

Generic min accumulator class. The parameter T is the type of values.

Definition at line 102 of file `accu/stat/min.hh`.

### 10.68.2 Member Function Documentation

**10.68.2.1** `template<typename T > void mln::accu::stat::min< T >::init ( ) [inline]`

Manipulators.

Definition at line 147 of file `accu/stat/min.hh`.

**10.68.2.2** `template<typename T > bool mln::accu::stat::min< T >::is_valid ( ) const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 195 of file `accu/stat/min.hh`.

**10.68.2.3** `template<typename T > void mln::accu::stat::min< T >::set_value ( const T & t ) [inline]`

Force the value of the min to  $t$ .

Definition at line 179 of file `accu/stat/min.hh`.

**10.68.2.4** `void mln::Accumulator< min< T > >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.68.2.5** `void mln::Accumulator< min< T > >::take_n_times ( unsigned n, const T & t )`  
**[inherited]**

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.68.2.6** `template<typename T > const T & mln::accu::stat::min< T >::to_result ( ) const`  
**[inline]**

Get the value of the accumulator.

Definition at line 187 of file `accu/stat/min.hh`.

## 10.69 mln::accu::stat::min\_h< V > Struct Template Reference

Generic min function based on histogram over a value set with type  $V$ .

```
#include <min_h.hh>
```

Inherits `base< const V &, min_h< V > >`.

### Public Member Functions

- `bool is_valid () const`  
*Check whether this accu is able to return a result.*
- `void take_as_init (const T &t)`  
*Take as initialization the value  $t$ .*
- `void take_n_times (unsigned n, const T &t)`  
*Take  $n$  times the value  $t$ .*
- `const argument & to_result () const`  
*Get the value of the accumulator.*
- `void init ()`  
*Manipulators.*

### 10.69.1 Detailed Description

`template<typename V> struct mln::accu::stat::min_h< V >`

Generic min function based on histogram over a value set with type  $V$ .

Definition at line 100 of file `min_h.hh`.

## 10.69.2 Member Function Documentation

**10.69.2.1** `template<typename V> void mln::accu::stat::min_h< V >::init ( ) [inline]`

Manipulators.

Definition at line 256 of file min\_h.hh.

**10.69.2.2** `template<typename V> bool mln::accu::stat::min_h< V >::is_valid ( ) const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 298 of file min\_h.hh.

**10.69.2.3** `void mln::Accumulator< min_h< V > >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.69.2.4** `void mln::Accumulator< min_h< V > >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take *n* times the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.69.2.5** `template<typename V> const min_h< V >::argument & mln::accu::stat::min_h< V >::to_result ( ) const [inline]`

Get the value of the accumulator.

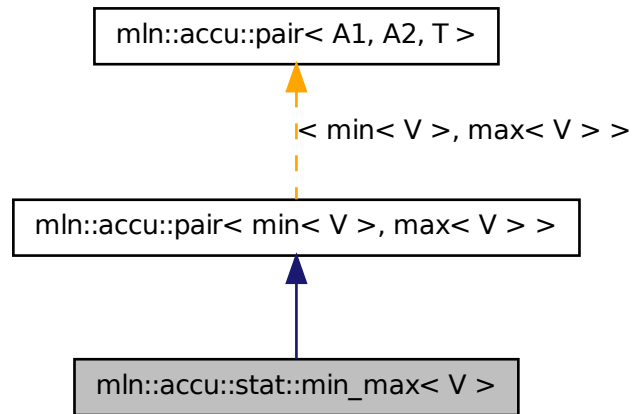
Definition at line 280 of file min\_h.hh.

## 10.70 mln::accu::stat::min\_max< V > Struct Template Reference

Generic min and max accumulator class.

```
#include <min_max.hh>
```

Inheritance diagram for `mln::accu::stat::min_max< V >`:



## Public Member Functions

- `min< V >::result first () const`  
*Return the result of the first accumulator.*
- `min< V > first_accu () const`  
*Return the first accumulator.*
- `bool is_valid () const`  
*Check whether this accu is able to return a result.*
- `max< V >::result second () const`  
*Return the result of the second accumulator.*
- `max< V > second_accu () const`  
*Return the second accumulator.*
- `template<typename T >`  
`void take_as_init (const T &t)`  
*Take as initialization the value  $t$ .*
- `template<typename T >`  
`void take_n_times (unsigned n, const T &t)`  
*Take  $n$  times the value  $t$ .*
- `void init ()`  
*Manipulators.*

- `std::pair< typename min< V >::result, typename max< V >::result > to_result () const`  
*Get the value of the accumulator.*

### 10.70.1 Detailed Description

`template<typename V> struct mln::accu::stat::min_max< V >`

Generic min and max accumulator class. The parameter `V` is the type of values.

Definition at line 61 of file `accu/stat/min_max.hh`.

### 10.70.2 Member Function Documentation

**10.70.2.1** `min< V >::result mln::accu::pair< min< V >, max< V >, mln_argument(min< V > )>::first ( ) const [inherited]`

Return the result of the first accumulator.

**10.70.2.2** `min< V > mln::accu::pair< min< V >, max< V >, mln_argument(min< V > )>::first_accu ( ) const [inherited]`

Return the first accumulator.

**10.70.2.3** `void mln::accu::pair< min< V >, max< V >, mln_argument(min< V > )>::init ( ) [inherited]`

Manipulators.

**10.70.2.4** `bool mln::accu::pair< min< V >, max< V >, mln_argument(min< V > )>::is_valid ( ) const [inherited]`

Check whether this `accu` is able to return a result.

Always true here.

**10.70.2.5** `max< V >::result mln::accu::pair< min< V >, max< V >, mln_argument(min< V > )>::second ( ) const [inherited]`

Return the result of the second accumulator.

**10.70.2.6** `max< V > mln::accu::pair< min< V >, max< V >, mln_argument(min< V > )>::second_accu ( ) const [inherited]`

Return the second accumulator.

**10.70.2.7** `template<typename E > template<typename T > void mln::Accumulator< E >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Definition at line 186 of file `accumulator.hh`.

References `mln::mln_exact()`.

**10.70.2.8** `template<typename E > template<typename T > void mln::Accumulator< E >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Definition at line 213 of file `accumulator.hh`.

References `mln::mln_exact()`.

**10.70.2.9** `std::pair<typename min< V > ::result, typename max< V > ::result> mln::accu::pair< min< V > , max< V > , mln_argument(min< V > ) >::to_result ( ) const [inherited]`

Get the value of the accumulator.

## 10.71 mln::accu::stat::rank< T > Struct Template Reference

Generic rank accumulator class.

```
#include <rank.hh>
```

Inherits `base< const T &, rank< T > >`.

### Public Member Functions

- bool `is_valid ()` const  
*Check whether this `accu` is able to return a result.*
- unsigned `k ()` const  
*Give the rank.*
- void `take_as_init (const T &t)`  
*Take as initialization the value  $t$ .*
- void `take_n_times (unsigned n, const T &t)`  
*Take  $n$  times the value  $t$ .*
- const T & `to_result ()` const  
*Get the value of the accumulator.*



- void `init()`  
*Manipulators.*

### 10.71.1 Detailed Description

**template<typename T> struct mln::accu::stat::rank< T >**

Generic rank accumulator class. The parameter T is the type of values.

Definition at line 60 of file rank.hh.

### 10.71.2 Member Function Documentation

**10.71.2.1 template<typename T> void mln::accu::stat::rank< T >::init( ) [inline]**

Manipulators.

Definition at line 319 of file rank.hh.

**10.71.2.2 template<typename T> bool mln::accu::stat::rank< T >::is\_valid( ) const [inline]**

Check whether this accu is able to return a result.

Always true here.

Definition at line 343 of file rank.hh.

**10.71.2.3 template<typename T> unsigned mln::accu::stat::rank< T >::k( ) const [inline]**

Give the rank.

Definition at line 178 of file rank.hh.

**10.71.2.4 void mln::Accumulator< rank< T > >::take\_as\_init( const T & t ) [inherited]**

Take as initialization the value t.

Dev note: this is a final method; override if needed by take\_as\_init\_ (ending with '\_').

**10.71.2.5 void mln::Accumulator< rank< T > >::take\_n\_times( unsigned n, const T & t ) [inherited]**

Take n times the value t.

Dev note: this is a final method; override if needed by take\_as\_init\_ (ending with '\_').

### 10.71.2.6 `template<typename T> const T & mln::accu::stat::rank< T >::to_result ( ) const [inline]`

Get the value of the accumulator.

Definition at line 333 of file rank.hh.

## 10.72 `mln::accu::stat::rank< bool >` Struct Template Reference

rank accumulator class for Boolean.

```
#include <rank_bool.hh>
```

Inherits base< bool, rank< bool > >.

### Public Member Functions

- bool `is_valid` () const  
*Check whether this accu is able to return a result.*
- void `take_as_init` (const T &t)  
*Take as initialization the value t.*
- void `take_n_times` (unsigned n, const T &t)  
*Take n times the value t.*
- bool `to_result` () const  
*Get the value of the accumulator.*
- void `init` ()  
*Manipulators.*

### 10.72.1 Detailed Description

```
template<> struct mln::accu::stat::rank< bool >
```

rank accumulator class for Boolean.

Definition at line 58 of file rank\_bool.hh.

### 10.72.2 Member Function Documentation

#### 10.72.2.1 `void mln::accu::stat::rank< bool >::init ( ) [inline]`

Manipulators.

Definition at line 105 of file rank\_bool.hh.

**10.72.2.2** `bool mln::accu::stat::rank< bool >::is_valid ( ) const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 157 of file rank\_bool.hh.

**10.72.2.3** `void mln::Accumulator< rank< bool > >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.72.2.4** `void mln::Accumulator< rank< bool > >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.72.2.5** `bool mln::accu::stat::rank< bool >::to_result ( ) const [inline]`

Get the value of the accumulator.

Definition at line 150 of file rank\_bool.hh.

**10.73 mln::accu::stat::rank\_high\_quant< T > Struct Template Reference**

Generic rank accumulator class.

```
#include <rank_high_quant.hh>
```

Inherits `base< const T &, rank_high_quant< T > >`.

**Public Member Functions**

- `bool is_valid () const`  
*Check whether this accu is able to return a result.*
- `void take_as_init (const T &t)`  
*Take as initialization the value  $t$ .*
- `void take_n_times (unsigned n, const T &t)`  
*Take  $n$  times the value  $t$ .*
- `const T & to_result () const`  
*Get the value of the accumulator.*

- void `init()`  
*Manipulators.*

### 10.73.1 Detailed Description

`template<typename T> struct mln::accu::stat::rank_high_quant< T >`

Generic rank accumulator class. The parameter `T` is the type of values.

Definition at line 57 of file `rank_high_quant.hh`.

### 10.73.2 Member Function Documentation

**10.73.2.1** `template<typename T > void mln::accu::stat::rank_high_quant< T >::init ( )`  
`[inline]`

Manipulators.

Definition at line 148 of file `rank_high_quant.hh`.

**10.73.2.2** `template<typename T > bool mln::accu::stat::rank_high_quant< T >::is_valid ( )`  
`const [inline]`

Check whether this `accu` is able to return a result.

Always true here.

Definition at line 197 of file `rank_high_quant.hh`.

**10.73.2.3** `void mln::Accumulator< rank_high_quant< T > >::take_as_init ( const T & t )`  
`[inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.73.2.4** `void mln::Accumulator< rank_high_quant< T > >::take_n_times ( unsigned n, const T & t )`  
`[inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.73.2.5** `template<typename T > const T & mln::accu::stat::rank_high_quant< T >::to_result ( )`  
`const [inline]`

Get the value of the accumulator.

Definition at line 183 of file `rank_high_quant.hh`.

## 10.74 mln::accu::stat::var< T > Struct Template Reference

Var accumulator class.

```
#include <var.hh>
```

Inherits base< algebra::mat< T::dim, T::dim, float >, var< T > >.

### Public Types

- typedef algebra::vec< dim, float > [mean\\_t](#)  
*Type equipment.*

### Public Member Functions

- bool [is\\_valid](#) () const  
*Check whether this accu returns a valid result.*
- [mean\\_t](#) [mean](#) () const  
*Get the mean vector.*
- unsigned [n\\_items](#) () const  
*Get the number of items.*
- void [take\\_as\\_init](#) (const T &t)  
*Take as initialization the value  $\tau$ .*
- void [take\\_n\\_times](#) (unsigned n, const T &t)  
*Take  $n$  times the value  $\tau$ .*
- result [to\\_result](#) () const  
*Get the accumulator result (the var value).*
- result [variance](#) () const  
*Get the variance matrix.*
  
- void [init](#) ()  
*Manipulators.*

### 10.74.1 Detailed Description

```
template<typename T> struct mln::accu::stat::var< T >
```

Var accumulator class. Parameter T is the type of vectors

Definition at line 58 of file accu/stat/var.hh.

## 10.74.2 Member Typedef Documentation

### 10.74.2.1 `template<typename T> typedef algebra::vec<dim,float> mln::accu::stat::var< T >::mean_t`

Type equipment.

Definition at line 88 of file `accu/stat/var.hh`.

## 10.74.3 Member Function Documentation

### 10.74.3.1 `template<typename T> void mln::accu::stat::var< T >::init ( ) [inline]`

Manipulators.

Definition at line 118 of file `accu/stat/var.hh`.

### 10.74.3.2 `template<typename T> bool mln::accu::stat::var< T >::is_valid ( ) const [inline]`

Check whether this accu returns a valid result.

Definition at line 213 of file `accu/stat/var.hh`.

### 10.74.3.3 `template<typename T> var< T >::mean_t mln::accu::stat::var< T >::mean ( ) const [inline]`

Get the mean vector.

Definition at line 200 of file `accu/stat/var.hh`.

References `mln::literal::zero`.

### 10.74.3.4 `template<typename T> unsigned mln::accu::stat::var< T >::n_items ( ) const [inline]`

Get the number of items.

Definition at line 192 of file `accu/stat/var.hh`.

### 10.74.3.5 `void mln::Accumulator< var< T > >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

### 10.74.3.6 `void mln::Accumulator< var< T > >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.74.3.7** `template<typename T > var< T >::result mln::accu::stat::var< T >::to_result ( )`  
`const [inline]`

Get the accumulator result (the var value).

Definition at line 168 of file `accu/stat/var.hh`.

References `mln::literal::zero`.

**10.74.3.8** `template<typename T > var< T >::result mln::accu::stat::var< T >::variance ( )`  
`const [inline]`

Get the variance matrix.

Definition at line 184 of file `accu/stat/var.hh`.

## 10.75 `mln::accu::stat::variance< T, S, R >` Struct Template Reference

Variance accumulator class.

`#include <variance.hh>`

Inherits `base< R, variance< T, S, R > >`.

### Public Member Functions

- `bool is_valid () const`  
*Check whether this accu is able to return a result.*
- `R mean () const`  
*Get the mean value.*
- `unsigned n_items () const`  
*Get the number of items.*
- `R standard_deviation () const`  
*Get the standard deviation value.*
- `S sum () const`  
*Get the sum value.*
- `void take_as_init (const T &t)`  
*Take as initialization the value  $t$ .*
- `void take_n_times (unsigned n, const T &t)`  
*Take  $n$  times the value  $t$ .*
- `R to_result () const`  
*Get the accumulator result (the variance value).*

- R `var ()` const  
*Get the variance value.*
- void `init ()`  
*Manipulators.*

### 10.75.1 Detailed Description

**template<typename T, typename S = typename mln::value::props< T >::sum, typename R = S>  
struct mln::accu::stat::variance< T, S, R >**

Variance accumulator class. Parameter T is the type of values that we sum. Parameter S is the type to store the value sum and the sum of value \* value; the default type of S is the summation type (property) of T. Parameter R is the type of the mean and variance values; the default type of R is S.

Definition at line 61 of file variance.hh.

### 10.75.2 Member Function Documentation

**10.75.2.1 template<typename T, typename S, typename R > void mln::accu::stat::variance< T, S, R >::init ( ) [inline]**

Manipulators.

Definition at line 125 of file variance.hh.

References mln::literal::zero.

**10.75.2.2 template<typename T, typename S, typename R > bool mln::accu::stat::variance< T, S, R >::is\_valid ( ) const [inline]**

Check whether this accu is able to return a result.

Always true here.

Definition at line 229 of file variance.hh.

**10.75.2.3 template<typename T, typename S, typename R > R mln::accu::stat::variance< T, S, R >::mean ( ) const [inline]**

Get the mean value.

Definition at line 187 of file variance.hh.

References mln::literal::zero.

**10.75.2.4 template<typename T, typename S, typename R > unsigned mln::accu::stat::variance< T, S, R >::n\_items ( ) const [inline]**

Get the number of items.

Definition at line 205 of file variance.hh.



**10.75.2.5** `template<typename T, typename S, typename R > R mln::accu::stat::variance< T, S, R >::standard_deviation ( ) const [inline]`

Get the standard deviation value.

Definition at line 221 of file `variance.hh`.

References `mln::accu::stat::variance< T, S, R >::to_result()`.

**10.75.2.6** `template<typename T, typename S, typename R > S mln::accu::stat::variance< T, S, R >::sum ( ) const [inline]`

Get the sum value.

Definition at line 197 of file `variance.hh`.

**10.75.2.7** `void mln::Accumulator< variance< T, S, R > >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.75.2.8** `void mln::Accumulator< variance< T, S, R > >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.75.2.9** `template<typename T, typename S, typename R > R mln::accu::stat::variance< T, S, R >::to_result ( ) const [inline]`

Get the accumulator result (the variance value).

Definition at line 176 of file `variance.hh`.

Referenced by `mln::accu::stat::variance< T, S, R >::standard_deviation()`, and `mln::accu::stat::variance< T, S, R >::var()`.

**10.75.2.10** `template<typename T, typename S, typename R > R mln::accu::stat::variance< T, S, R >::var ( ) const [inline]`

Get the variance value.

Definition at line 213 of file `variance.hh`.

References `mln::accu::stat::variance< T, S, R >::to_result()`.

## 10.76 `mln::accu::tuple< A, n, >` Struct Template Reference

Generic tuple of accumulators.

```
#include <tuple.hh>
```

Inherits base< boost::tuple< BOOST\_PP\_REPEAT(10, RESULT\_ACCU, Le Ricard ya que ca de vrai!) >, tuple< A, n, BOOST\_PP\_ENUM\_PARAMS(10, T)> >.

## Public Member Functions

- bool [is\\_valid](#) () const  
*Check whether this accu is able to return a result.*
- void [take\\_as\\_init](#) (const T &t)  
*Take as initialization the value t.*
- void [take\\_n\\_times](#) (unsigned n, const T &t)  
*Take n times the value t.*
- res [to\\_result](#) () const  
*Get the value of the accumulator.*
- void [init](#) ()  
*Manipulators.*

### 10.76.1 Detailed Description

**template<typename A, unsigned n, BOOST\_PP\_ENUM\_PARAMS\_WITH\_A\_DEFAULT(10, typename T, boost::tuples::null\_type)> struct mln::accu::tuple< A, n, >**

Generic tuple of accumulators. The parameter T is the type of values.

Definition at line 74 of file tuple.hh.

### 10.76.2 Member Function Documentation

**10.76.2.1 template<typename A , unsigned n, BOOST\_PP\_ENUM\_PARAMS(10, typename T) > void mln::accu::tuple< A, n, >::init ( ) [inline]**

Manipulators.

Definition at line 197 of file tuple.hh.

**10.76.2.2 template<typename A , unsigned n, BOOST\_PP\_ENUM\_PARAMS(10, typename T) > bool mln::accu::tuple< A, n, >::is\_valid ( ) const [inline]**

Check whether this accu is able to return a result.

Always true here.

Definition at line 239 of file tuple.hh.

**10.76.2.3** void mln::Accumulator< tuple< A, n, BOOST\_PP\_ENUM\_PARAMS(10, T)>  
>::take\_as\_init ( const T & t ) [inherited]

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by take\_as\_init\_ (ending with '\_').

**10.76.2.4** void mln::Accumulator< tuple< A, n, BOOST\_PP\_ENUM\_PARAMS(10, T)>  
>::take\_n\_times ( unsigned n, const T & t ) [inherited]

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by take\_as\_init\_ (ending with '\_').

**10.76.2.5** template<typename A , unsigned n, BOOST\_PP\_ENUM\_PARAMS(10, typename T) >  
tuple< A, n, BOOST\_PP\_ENUM\_PARAMS(10, T) >::res mln::accu::tuple< A, n,  
>::to\_result ( ) const [inline]

Get the value of the accumulator.

Definition at line 229 of file tuple.hh.

## 10.77 mln::accu::val< A > Struct Template Reference

Generic val of accumulators.

```
#include <v.hh>
```

Inherits base< const A::result &, val< A > >.

### Public Member Functions

- bool [is\\_valid](#) () const  
*Check whether this accu is able to return a result.*
- void [take\\_as\\_init](#) (const T &t)  
*Take as initialization the value  $t$ .*
- void [take\\_n\\_times](#) (unsigned n, const T &t)  
*Take  $n$  times the value  $t$ .*
- const A::result & [to\\_result](#) () const  
*Get the value of the accumulator.*
- void [init](#) ()  
*Manipulators.*

### 10.77.1 Detailed Description

**template<typename A> struct mln::accu::val< A >**

Generic val of accumulators.

Definition at line 50 of file v.hh.

### 10.77.2 Member Function Documentation

**10.77.2.1 template<typename A > void mln::accu::val< A >::init ( ) [inline]**

Manipulators.

Definition at line 119 of file v.hh.

**10.77.2.2 template<typename A > bool mln::accu::val< A >::is\_valid ( ) const [inline]**

Check whether this accu is able to return a result.

Always true here.

Definition at line 177 of file v.hh.

**10.77.2.3 void mln::Accumulator< val< A > >::take\_as\_init ( const T & t ) [inherited]**

Take as initialization the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '\_').

**10.77.2.4 void mln::Accumulator< val< A > >::take\_n\_times ( unsigned n, const T & t ) [inherited]**

Take *n* times the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '\_').

**10.77.2.5 template<typename A > const A::result & mln::accu::val< A >::to\_result ( ) const [inline]**

Get the value of the accumulator.

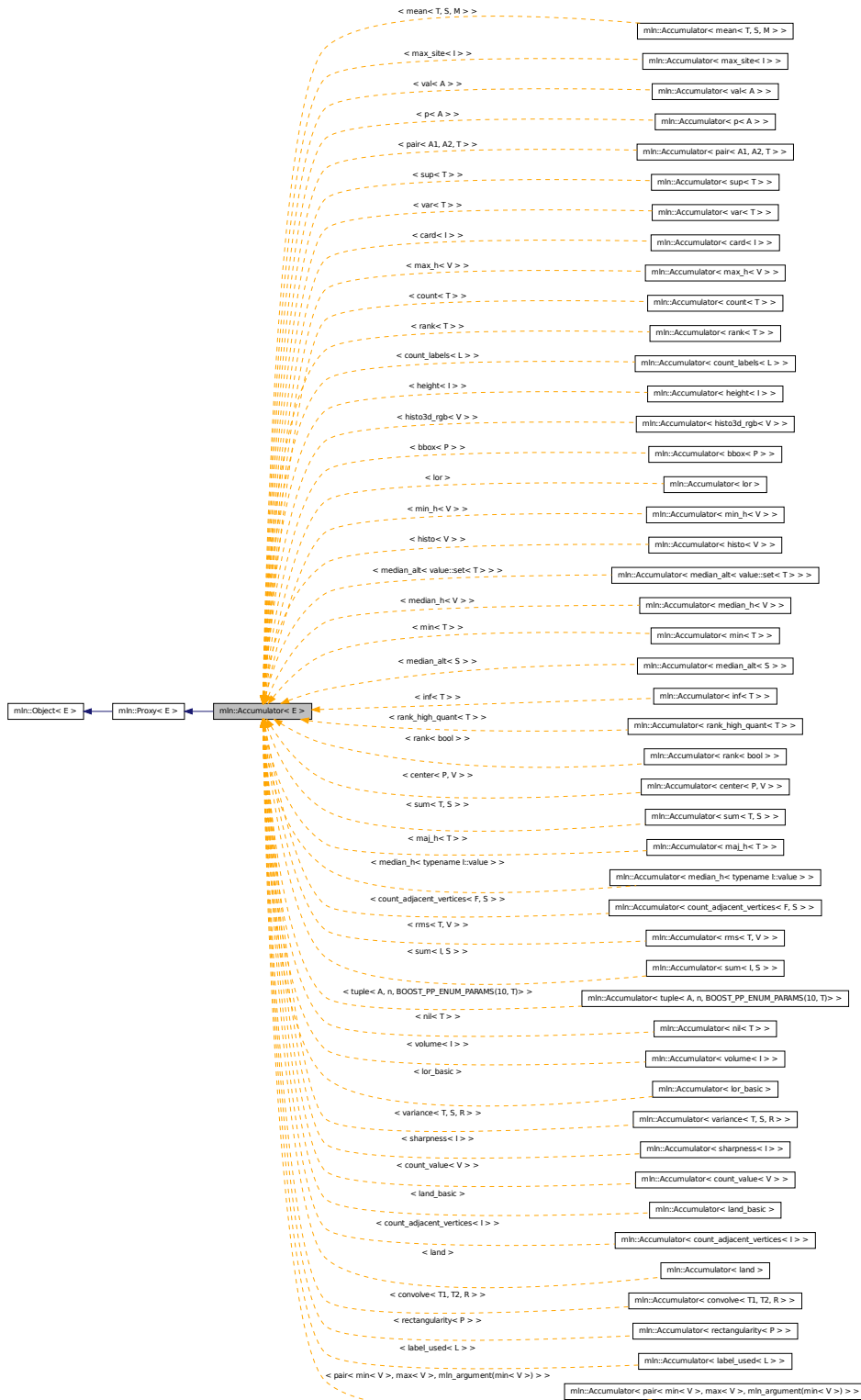
Definition at line 169 of file v.hh.

## 10.78 mln::Accumulator< E > Struct Template Reference

Base class for implementation of accumulators.

```
#include <accumulator.hh>
```

Inheritance diagram for mln::Accumulator< E > :



## Public Member Functions

- `template<typename T >`  
`void take_as_init (const T &t)`  
*Take as initialization the value  $t$ .*
- `template<typename T >`  
`void take_n_times (unsigned n, const T &t)`  
*Take  $n$  times the value  $t$ .*

### 10.78.1 Detailed Description

`template<typename E> struct mln::Accumulator< E >`

Base class for implementation of accumulators. The parameter  $E$  is the exact type.

#### See also

[mln::doc::Accumulator](#) for a complete documentation of this class contents.

Definition at line 78 of file accumulator.hh.

### 10.78.2 Member Function Documentation

**10.78.2.1** `template<typename E > template<typename T > void mln::Accumulator< E >::take_as_init ( const T & t )`

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Definition at line 186 of file accumulator.hh.

References `mln::mln_exact()`.

**10.78.2.2** `template<typename E > template<typename T > void mln::Accumulator< E >::take_n_times ( unsigned n, const T & t )`

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Definition at line 213 of file accumulator.hh.

References `mln::mln_exact()`.

## 10.79 mln::algebra::h\_mat< d, T > Struct Template Reference

N-Dimensional matrix with homogeneous coordinates.

```
#include <h_mat.hh>
```

Inherits `mln::algebra::mat< d+1, d+1, T >`.

## Public Types

- enum

*Dimension is the 'natural' one (3 for 3D), not the one of the vector (dim + 1).*

## Public Member Functions

- `mat< n, m, T > _1 () const`  
*Return the inverse of the matrix.*
- `h_mat ()`  
*Constructor without argument.*
- `h_mat (const mat< d+1, d+1, T > &x)`  
*Constructor with the underlying matrix.*
- `mat< m, n, T > t () const`  
*Return the transpose of the matrix.*

### 10.79.1 Detailed Description

`template<unsigned d, typename T> struct mln::algebra::h_mat< d, T >`

N-Dimensional matrix with homogeneous coordinates.

Definition at line 49 of file algebra/h\_mat.hh.

### 10.79.2 Member Enumeration Documentation

#### 10.79.2.1 `template<unsigned d, typename T> anonymous enum`

Dimension is the 'natural' one (3 for 3D), not the one of the vector (dim + 1).

Definition at line 52 of file algebra/h\_mat.hh.

### 10.79.3 Constructor & Destructor Documentation

#### 10.79.3.1 `template<unsigned d, typename T > mln::algebra::h_mat< d, T >::h_mat ( )` `[inline]`

Constructor without argument.

Definition at line 67 of file algebra/h\_mat.hh.

#### 10.79.3.2 `template<unsigned d, typename T > mln::algebra::h_mat< d, T >::h_mat ( const mat< d+1, d+1, T > & x )` `[inline]`

Constructor with the underlying matrix.

Definition at line 74 of file algebra/h\_mat.hh.

## 10.79.4 Member Function Documentation

**10.79.4.1** `template<unsigned n, unsigned m, typename T > mat< n, m, T > mln::algebra::mat< n, m, T >::_1 ( ) const [inline, inherited]`

Return the inverse of the matrix.

Only compile on square matrix.

Definition at line 604 of file algebra/mat.hh.

**10.79.4.2** `template<unsigned n, unsigned m, typename T > mat< m, n, T > mln::algebra::mat< n, m, T >::t ( ) const [inline, inherited]`

Return the transpose of the matrix.

Definition at line 538 of file algebra/mat.hh.

## 10.80 mln::algebra::h\_vec< d, C > Class Template Reference

N-Dimensional vector with homogeneous coordinates.

```
#include <h_vec.hh>
```

Inherits mln::algebra::vec< d+1, C >.

### Public Types

- enum

*Dimension is the 'natural' one (3 for 3D), not the one of the vector (dim + 1).*

### Public Member Functions

- [h\\_vec](#) ()

*Constructor without argument.*

- [h\\_vec](#) (const vec< d+1, C > &other)

*Constructor with the underlying vector.*

- template<typename U >  
[operator mat< n, 1, U > \(\) const](#)

*Conversion to a matrix.*

- mat< 1, n, T > [t](#) () const

*Transposition.*

- vec< d, C > [to\\_vec](#) () const

*Back to the natural (non-homogeneous) space.*



## Static Public Attributes

- static const vec< n, T > **origin** = all\_to(0)  
*Origin value.*
- static const vec< n, T > **zero** = all\_to(0)  
*Zero value.*

### 10.80.1 Detailed Description

`template<unsigned d, typename C> class mln::algebra::h_vec< d, C >`

N-Dimensional vector with homogeneous coordinates.

Definition at line 94 of file h\_vec.hh.

### 10.80.2 Member Enumeration Documentation

**10.80.2.1** `template<unsigned d, typename C> anonymous enum`

Dimension is the 'natural' one (3 for 3D), not the one of the vector (dim + 1).

Definition at line 98 of file h\_vec.hh.

### 10.80.3 Constructor & Destructor Documentation

**10.80.3.1** `template<unsigned d, typename C > mln::algebra::h_vec< d, C >::h_vec ( )`  
`[inline]`

Constructor without argument.

Definition at line 117 of file h\_vec.hh.

References mln::literal::one.

**10.80.3.2** `template<unsigned d, typename C > mln::algebra::h_vec< d, C >::h_vec ( const vec<`  
`d+1, C > & other ) [inline]`

Constructor with the underlying vector.

Definition at line 128 of file h\_vec.hh.

### 10.80.4 Member Function Documentation

**10.80.4.1** `template<unsigned n, typename T > template<typename U > mln::algebra::vec< n, T`  
`>::operator mat< n, 1, U > ( ) const [inline, inherited]`

Conversion to a matrix.

Definition at line 368 of file algebra/mat.hh.

**10.80.4.2** `template<unsigned n, typename T > mat< 1, n, T > mln::algebra::vec< n, T >::t ( )`  
`const [inline, inherited]`

Transposition.

Definition at line 858 of file algebra/mat.hh.

**10.80.4.3** `template<unsigned d, typename C > vec< d, C > mln::algebra::h_vec< d, C >::to_vec`  
`( ) const [inline]`

Back to the natural (non-homogeneous) space.

Definition at line 145 of file h\_vec.hh.

## 10.80.5 Member Data Documentation

**10.80.5.1** `template<unsigned n, typename T> const vec< n, T > mln::algebra::vec< n, T`  
`>::origin = all_to(0) [static, inherited]`

Origin value.

Definition at line 247 of file algebra/vec.hh.

**10.80.5.2** `template<unsigned n, typename T> const vec< n, T > mln::algebra::vec< n, T >::zero`  
`= all_to(0) [static, inherited]`

Zero value.

Definition at line 244 of file algebra/vec.hh.

## 10.81 mln::bkd\_pixter1d< I > Class Template Reference

Backward pixel iterator on a 1-D image with border.

```
#include <pixter1d.hh>
```

Inherits `backward_pixel_iterator_base_< I, bkd_pixter1d< I > >`.

### Public Types

- typedef [I image](#)  
*Image type.*

### Public Member Functions

- [bkd\\_pixter1d](#) (I &image)  
*Constructor.*
- void [next](#) ()  
*Go to the next element.*

### 10.81.1 Detailed Description

`template<typename I> class mln::bkd_pixter1d< I >`

Backward pixel iterator on a 1-D image with border.

Definition at line 69 of file `pixter1d.hh`.

### 10.81.2 Member Typedef Documentation

**10.81.2.1** `template<typename I > typedef I mln::bkd_pixter1d< I >::image`

[Image](#) type.

Definition at line 76 of file `pixter1d.hh`.

### 10.81.3 Constructor & Destructor Documentation

**10.81.3.1** `template<typename I > mln::bkd_pixter1d< I >::bkd_pixter1d ( I & image )`  
`[inline]`

Constructor.

#### Parameters

`[in]` *image* The image this pixel iterator is bound to.

Definition at line 117 of file `pixter1d.hh`.

### 10.81.4 Member Function Documentation

**10.81.4.1** `void mln::Iterator< bkd_pixter1d< I > >::next ( )` `[inherited]`

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

#### Precondition

The iterator is valid.

## 10.82 mln::bkd\_pixter2d< I > Class Template Reference

Backward pixel iterator on a 2-D image with border.

```
#include <pixter2d.hh>
```

Inherits `backward_pixel_iterator_base_< I, bkd_pixter2d< I > >`.

## Public Types

- typedef I [image](#)  
*Image type.*

## Public Member Functions

- [bkd\\_pixter2d](#) (I &image)  
*Constructor.*
- void [next](#) ()  
*Go to the next element.*

### 10.82.1 Detailed Description

`template<typename I> class mln::bkd_pixter2d< I >`

Backward pixel iterator on a 2-D image with border.

Definition at line 87 of file `pixter2d.hh`.

### 10.82.2 Member Typedef Documentation

**10.82.2.1** `template<typename I > typedef I mln::bkd_pixter2d< I >::image`

[Image](#) type.

Definition at line 94 of file `pixter2d.hh`.

### 10.82.3 Constructor & Destructor Documentation

**10.82.3.1** `template<typename I > mln::bkd_pixter2d< I >::bkd_pixter2d ( I & image )`  
`[inline]`

Constructor.

#### Parameters

`[in]` *image* The image this pixel iterator is bound to.

Definition at line 169 of file `pixter2d.hh`.

### 10.82.4 Member Function Documentation

**10.82.4.1** `void mln::Iterator< bkd_pixter2d< I > >::next ( )` `[inherited]`

Go to the next element.

**Warning**

This is a final method; iterator classes should not re-define this method. The actual "next" operation has to be defined through the *next\_* method.

**Precondition**

The iterator is valid.

## 10.83 mln::bkd\_pixter3d< I > Class Template Reference

Backward pixel iterator on a 3-D image with border.

```
#include <pixter3d.hh>
```

Inherits backward\_pixel\_iterator\_base\_< I, bkd\_pixter3d< I > >.

**Public Types**

- typedef I [image](#)  
*Image type.*

**Public Member Functions**

- [bkd\\_pixter3d](#) (I &[image](#))  
*Constructor.*
- void [next](#) ()  
*Go to the next element.*

### 10.83.1 Detailed Description

```
template<typename I> class mln::bkd_pixter3d< I >
```

Backward pixel iterator on a 3-D image with border.

Definition at line 100 of file pixter3d.hh.

### 10.83.2 Member Typedef Documentation

**10.83.2.1** `template<typename I > typedef I mln::bkd_pixter3d< I >::image`

[Image](#) type.

Definition at line 107 of file pixter3d.hh.

### 10.83.3 Constructor & Destructor Documentation

**10.83.3.1** `template<typename I > mln::bkd_pixter3d< I >::bkd_pixter3d ( I & image )`  
`[inline]`

Constructor.

#### Parameters

[in] *image* The image this pixel iterator is bound to.

Definition at line 207 of file `pixter3d.hh`.

### 10.83.4 Member Function Documentation

**10.83.4.1** `void mln::Iterator< bkd_pixter3d< I > >::next ( )` `[inherited]`

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

#### Precondition

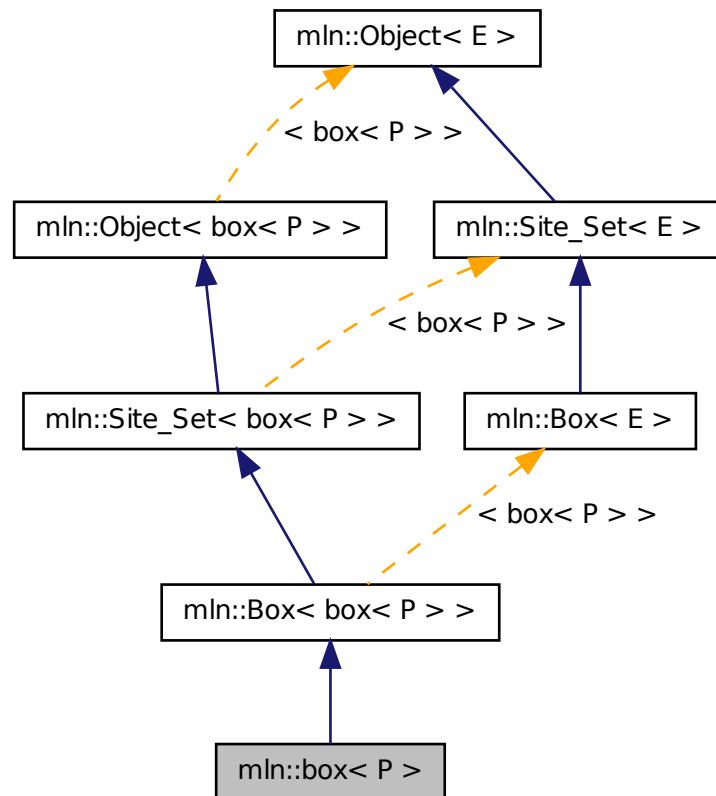
The iterator is valid.

## 10.84 mln::box< P > Class Template Reference

Generic box class: site set containing points of a regular grid.

```
#include <box.hh>
```

Inheritance diagram for mln::box< P >:



## Public Types

- enum `Dimension`  
*Dimension.*
- typedef `box_bkd_piter_< P >` `bkd_piter`  
*Backward [Site\\_Iterator](#) associated type.*
- typedef `P` `element`  
*Element associated type.*
- typedef `box_fwd_piter_< P >` `fwd_piter`  
*Forward [Site\\_Iterator](#) associated type.*
- typedef `fwd_piter` `piter`  
*[Site\\_Iterator](#) associated type.*

- typedef P [psite](#)  
*Psite associated type.*
- typedef P [site](#)  
*Site associated type.*

## Public Member Functions

- const [box](#)< P > & [bbox](#) () const  
*Give the bounding box of this site set.*
- [box](#) ()  
*Constructor without argument.*
- [box](#) (const [site](#) &pmin, const [site](#) &pmax)  
*Constructor of a box going from pmin to pmax.*
- void [crop\\_wrt](#) (const [box](#)< P > &b)  
*Crop this bbox in order to fit in the reference box b.*
- void [enlarge](#) (unsigned b)  
*Enlarge the box with a border b.*
- void [enlarge](#) (unsigned dim, unsigned b)  
*Enlarge the box with a border b for dimension dim.*
- bool [has](#) (const P &p) const  
*Test if p belongs to the box.*
- bool [is\\_empty](#) () const  
*Test if this box is empty.*
- bool [is\\_valid](#) () const  
*Test that the box owns valid data, i.e., is initialized and with pmin being 'less-than' pmax.*
- unsigned [len](#) (unsigned i) const  
*Give the length of the i-th side of the box.*
- std::size\_t [memory\\_size](#) () const  
*Return the size of this site set in memory.*
- void [merge](#) (const [box](#)< P > &b)  
*Merge inplace with another box.*
- unsigned [nsites](#) () const  
*Give the number of sites of this box.*



- P [pcenter](#) () const  
*Return the approximated central site of this box.*
- P & [pmax](#) ()  
*Reference to the maximum point.*
- P [pmax](#) () const  
*Maximum point.*
- P [pmin](#) () const  
*Minimum point.*
- P & [pmin](#) ()  
*Reference to the minimum point.*
- [box](#)< P > [to\\_larger](#) (unsigned b) const  
*Give a larger box.*
- [box](#) (typename P::coord ninds)

## Related Functions

(Note that these are not member functions.)

- `template<typename P > std::ostream & operator<< (std::ostream &ostr, const box< P > &b)`  
*Print a generic box b into the output stream `ostr`.*

### 10.84.1 Detailed Description

`template<typename P> class mln::box< P >`

Generic box class: site set containing points of a regular grid. Parameter P is the corresponding type of point.

Definition at line 81 of file `core/site_set/box.hh`.

### 10.84.2 Member Typedef Documentation

**10.84.2.1** `template<typename P> typedef box_bkd_piter_<P> mln::box< P >::bkd_piter`

Backward [Site\\_Iterator](#) associated type.

Definition at line 105 of file `core/site_set/box.hh`.

**10.84.2.2** `template<typename P> typedef P mln::box< P >::element`

Element associated type.

Definition at line 90 of file `core/site_set/box.hh`.

**10.84.2.3** `template<typename P> typedef box_fwd_piter_<P> mln::box< P >::fwd_piter`

Forward [Site\\_Iterator](#) associated type.

Definition at line 99 of file core/site\_set/box.hh.

**10.84.2.4** `template<typename P> typedef fwd_piter mln::box< P >::piter`

[Site\\_Iterator](#) associated type.

Definition at line 102 of file core/site\_set/box.hh.

**10.84.2.5** `template<typename P> typedef P mln::box< P >::psite`

Psite associated type.

Definition at line 93 of file core/site\_set/box.hh.

**10.84.2.6** `template<typename P> typedef P mln::box< P >::site`

[Site](#) associated type.

Definition at line 96 of file core/site\_set/box.hh.

**10.84.3 Member Enumeration Documentation****10.84.3.1** `template<typename P> anonymous enum`

Dimension.

Definition at line 87 of file core/site\_set/box.hh.

**10.84.4 Constructor & Destructor Documentation****10.84.4.1** `template<typename P> mln::box< P >::box ( ) [inline]`

Constructor without argument.

Definition at line 275 of file core/site\_set/box.hh.

**10.84.4.2** `template<typename P> mln::box< P >::box ( const site & pmin, const site & pmax ) [inline]`

Constructor of a box going from pmin to pmax.

Definition at line 284 of file core/site\_set/box.hh.

References `mln::box< P >::is_valid()`.

**10.84.4.3** `template<typename P> mln::box< P >::box ( typename P::coord ninds ) [inline, explicit]`

Constructors with different numbers of arguments (sizes) w.r.t. the dimension.

Definition at line 293 of file core/site\_set/box.hh.

References mln::literal::origin.

## 10.84.5 Member Function Documentation

### 10.84.5.1 `const box< P > & mln::Box< box< P > >::bbox ( ) const` [inherited]

Give the bounding box of this site set.

Return the bounding box of this site set, so that is itself. This method is declared by the [mln::Site\\_Set](#) concept.

#### Warning

This method is final for all box classes.

### 10.84.5.2 `template<typename P > void mln::box< P >::crop_wrt ( const box< P > & b )` [inline]

Crop this bbox in order to fit in the reference box *b*.

Definition at line 205 of file core/site\_set/box.hh.

References mln::box< P >::pmax(), and mln::box< P >::pmin().

Referenced by mln::debug::draw\_graph(), and mln::make\_debug\_graph\_image().

### 10.84.5.3 `template<typename P > void mln::box< P >::enlarge ( unsigned b )` [inline]

Enlarge the box with a border *b*.

Definition at line 337 of file core/site\_set/box.hh.

References mln::box< P >::is\_valid().

Referenced by mln::registration::icp().

### 10.84.5.4 `template<typename P > void mln::box< P >::enlarge ( unsigned dim, unsigned b )` [inline]

Enlarge the box with a border *b* for dimension *dim*.

Definition at line 351 of file core/site\_set/box.hh.

References mln::box< P >::is\_valid().

### 10.84.5.5 `template<typename P > bool mln::box< P >::has ( const P & p ) const` [inline]

Test if *p* belongs to the box.

#### Parameters

[in] *p* A point site.

Definition at line 325 of file core/site\_set/box.hh.

References `mln::box< P >::is_valid()`.

Referenced by `mln::morpho::line_gradient()`.

#### 10.84.5.6 `bool mln::Box< box< P > >::is_empty ( ) const [inherited]`

Test if this box is empty.

#### 10.84.5.7 `template<typename P > bool mln::box< P >::is_valid ( ) const [inline]`

Test that the box owns valid data, i.e., is initialized and with `pmin` being 'less-than' `pmax`.

Definition at line 195 of file core/site\_set/box.hh.

References `mln::util::ord_weak()`.

Referenced by `mln::box< P >::box()`, `mln::transform::distance_and_closest_point_geodesic()`, `mln::box< P >::enlarge()`, `mln::box< P >::has()`, `mln::box< P >::merge()`, `mln::box< P >::pcenter()`, `mln::box< P >::pmax()`, `mln::box< P >::pmin()`, and `mln::box< P >::to_larger()`.

#### 10.84.5.8 `unsigned mln::Box< box< P > >::len ( unsigned i ) const [inherited]`

Give the length of the `i`-th side of the box.

#### Precondition

`i < site::dim`

#### Warning

This method is final for all box classes.

#### 10.84.5.9 `template<typename P > std::size_t mln::box< P >::memory_size ( ) const [inline]`

Return the size of this site set in memory.

Definition at line 407 of file core/site\_set/box.hh.

#### 10.84.5.10 `template<typename P > void mln::box< P >::merge ( const box< P > & b ) [inline]`

Merge inplace with another box.

Definition at line 221 of file core/site\_set/box.hh.

References `mln::box< P >::is_valid()`, `mln::box< P >::pmax()`, and `mln::box< P >::pmin()`.

#### 10.84.5.11 `unsigned mln::Box< box< P > >::nsites ( ) const [inherited]`

Give the number of sites of this box.

Return the number of sites of this box. This method is declared by the `mln::Site_Set` concept.

**Warning**

This method is final for all box classes.

**10.84.5.12** `template<typename P> P mln::box< P >::pcenter ( ) const [inline]`

Return the approximated central site of this box.

Definition at line 395 of file core/site\_set/box.hh.

References mln::box< P >::is\_valid().

**10.84.5.13** `template<typename P> P mln::box< P >::pmax ( ) const [inline]`

Maximum point.

Definition at line 259 of file core/site\_set/box.hh.

References mln::box< P >::is\_valid().

Referenced by mln::box< P >::crop\_wrt(), mln::make::image3d(), mln::larger\_than(), mln::io::fld::load(), and mln::box< P >::merge().

**10.84.5.14** `template<typename P> P & mln::box< P >::pmax ( ) [inline]`

Reference to the maximum point.

Definition at line 268 of file core/site\_set/box.hh.

**10.84.5.15** `template<typename P> P & mln::box< P >::pmin ( ) [inline]`

Reference to the minimum point.

Definition at line 251 of file core/site\_set/box.hh.

**10.84.5.16** `template<typename P> P mln::box< P >::pmin ( ) const [inline]`

Minimum point.

Definition at line 242 of file core/site\_set/box.hh.

References mln::box< P >::is\_valid().

Referenced by mln::box< P >::crop\_wrt(), mln::make::image3d(), mln::larger\_than(), mln::io::fld::load(), and mln::box< P >::merge().

**10.84.5.17** `template<typename P> box< P > mln::box< P >::to_larger ( unsigned b ) const [inline]`

Give a larger box.

Definition at line 378 of file core/site\_set/box.hh.

References mln::box< P >::is\_valid().

## 10.84.6 Friends And Related Function Documentation

**10.84.6.1** `template<typename P> std::ostream & operator<< ( std::ostream & ostr, const box<P> & b )` [[related](#)]

Print a generic box *b* into the output stream *ostr*.

### Parameters

[in, out] *ostr* An output stream.

[in] *b* A generic box.

### Returns

The modified output stream *ostr*.

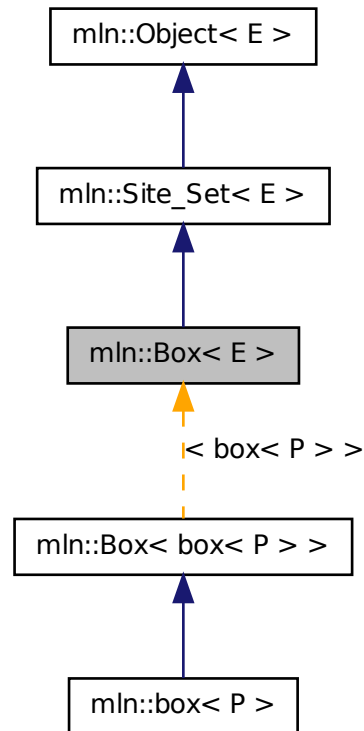
Definition at line 414 of file core/site\_set/box.hh.

## 10.85 mln::Box< E > Struct Template Reference

Base class for implementation classes of boxes.

```
#include <box.hh>
```

Inheritance diagram for mln::Box< E >:



## Public Member Functions

- `const E & bbox () const`  
*Give the bounding box of this site set.*
- `bool is_empty () const`  
*Test if this box is empty.*
- `unsigned len (unsigned i) const`  
*Give the length of the *i*-th side of the box.*
- `unsigned nsites () const`  
*Give the number of sites of this box.*

## Related Functions

(Note that these are not member functions.)

- `template<typename SI , typename Sr >`  
`p_set< typename SI::site > diff (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`  
*Set theoretic difference of lhs and rhs.*
- `template<typename SI , typename Sr >`  
`p_set< typename SI::site > inter (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`  
*Intersection between a couple of point sets.*
- `template<typename SI , typename Sr >`  
`bool operator< (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`  
*Strict inclusion test between site sets lhs and rhs.*
- `template<typename BI , typename Br >`  
`bool operator< (const Box< BI > &lhs, const Box< Br > &rhs)`  
*Strict inclusion test between boxes lhs and rhs.*
- `template<typename S >`  
`std::ostream & operator<< (std::ostream &ostr, const Site_Set< S > &set)`  
*Print a site set set into the output stream ostr.*
- `template<typename SI , typename Sr >`  
`bool operator<= (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`  
*Inclusion test between site sets lhs and rhs.*
- `template<typename BI , typename Br >`  
`bool operator<= (const Box< BI > &lhs, const Box< Br > &rhs)`  
*Inclusion test between boxes lhs and rhs.*
- `template<typename SI , typename Sr >`  
`bool operator== (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`  
*Equality test between site sets lhs and rhs.*
- `template<typename SI , typename Sr >`  
`p_set< typename SI::site > sym_diff (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`  
*Set theoretic symmetrical difference of lhs and rhs.*
- `template<typename SI , typename Sr >`  
`p_set< typename SI::site > uni (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`  
*Union of a couple of point sets.*
- `template<typename S >`  
`p_set< typename S::site > unique (const Site_Set< S > &s)`  
*Give the unique set of s.*

### 10.85.1 Detailed Description

**template<typename E> struct mln::Box< E >**

Base class for implementation classes of boxes. Boxes are particular site sets useful to bound any set of sites defined on a regular grid.



**See also**

[mln::doc::Box](#) for a complete documentation of this class contents.

Definition at line 48 of file core/concept/box.hh.

**10.85.2 Member Function Documentation****10.85.2.1 `template<typename E > const E & mln::Box< E >::bbox ( ) const [inline]`**

Give the bounding box of this site set.

Return the bounding box of this site set, so that is itself. This method is declared by the [mln::Site\\_Set](#) concept.

**Warning**

This method is final for all box classes.

Definition at line 124 of file core/concept/box.hh.

**10.85.2.2 `template<typename E > bool mln::Box< E >::is_empty ( ) const [inline]`**

Test if this box is empty.

Definition at line 167 of file core/concept/box.hh.

**10.85.2.3 `template<typename E > unsigned mln::Box< E >::len ( unsigned i ) const [inline]`**

Give the length of the *i*-th side of the box.

**Precondition**

$i < \text{site::dim}$

**Warning**

This method is final for all box classes.

Definition at line 131 of file core/concept/box.hh.

**10.85.2.4 `template<typename E > unsigned mln::Box< E >::nsites ( ) const [inline]`**

Give the number of sites of this box.

Return the number of sites of this box. This method is declared by the [mln::Site\\_Set](#) concept.

**Warning**

This method is final for all box classes.

Definition at line 153 of file core/concept/box.hh.

Referenced by `mln::morpho::line_gradient()`.

### 10.85.3 Friends And Related Function Documentation

**10.85.3.1** `template<typename SI , typename Sr > p_set< typename SI::site > diff ( const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs ) [related, inherited]`

Set theoretic difference of `lhs` and `rhs`.

Definition at line 66 of file `set/diff.hh`.

**10.85.3.2** `template<typename SI , typename Sr > p_set< typename SI::site > inter ( const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs ) [related, inherited]`

Intersection between a couple of point sets.

Definition at line 62 of file `set/inter.hh`.

**10.85.3.3** `template<typename SI , typename Sr > bool operator< ( const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs ) [related, inherited]`

Strict inclusion test between site sets `lhs` and `rhs`.

#### Parameters

[in] *lhs* A site set (strictly included?).

[in] *rhs* Another site set (includer?).

Definition at line 479 of file `operators.hh`.

**10.85.3.4** `template<typename BI , typename Br > bool operator< ( const Box< BI > & lhs, const Box< Br > & rhs ) [related]`

Strict inclusion test between boxes `lhs` and `rhs`.

#### Parameters

[in] *lhs* A box (strictly included?).

[in] *rhs* Another box (includer?).

Definition at line 193 of file `core/concept/box.hh`.

**10.85.3.5** `template<typename S > std::ostream & operator<< ( std::ostream & ostr, const Site_Set< S > & set ) [related, inherited]`

Print a site set `set` into the output stream `ostr`.

#### Parameters

[in, out] *ostr* An output stream.

[in] *set* A site set.

#### Returns

The modified output stream `ostr`.

Definition at line 505 of file `operators.hh`.

**10.85.3.6** `template<typename Sl, typename Sr > bool operator<= ( const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs )` [**related, inherited**]

Inclusion test between site sets `lhs` and `rhs`.

#### Parameters

[in] *lhs* A site set (included?).

[in] *rhs* Another site set (includer?).

Definition at line 491 of file operators.hh.

**10.85.3.7** `template<typename Bl, typename Br > bool operator<= ( const Box< Bl > & lhs, const Box< Br > & rhs )` [**related**]

Inclusion test between boxes `lhs` and `rhs`.

#### Parameters

[in] *lhs* A box (included?).

[in] *rhs* Another box (includer?).

Definition at line 178 of file core/concept/box.hh.

**10.85.3.8** `template<typename Sl, typename Sr > bool operator== ( const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs )` [**related, inherited**]

Equality test between site sets `lhs` and `rhs`.

#### Parameters

[in] *lhs* A site set.

[in] *rhs* Another site set.

Definition at line 467 of file operators.hh.

**10.85.3.9** `template<typename Sl, typename Sr > p_set< typename Sl::site > sym_diff ( const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs )` [**related, inherited**]

Set theoretic symmetrical difference of `lhs` and `rhs`.

Definition at line 65 of file sym\_diff.hh.

**10.85.3.10** `template<typename Sl, typename Sr > p_set< typename Sl::site > uni ( const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs )` [**related, inherited**]

Union of a couple of point sets.

Definition at line 61 of file uni.hh.

**10.85.3.11** `template<typename S> p_set< typename S::site > unique ( const Site_Set< S > & s )` [**related**, **inherited**]

Give the unique set of *s*.

Definition at line 61 of file unique.hh.

## 10.86 mln::box\_runend\_piter< P > Class Template Reference

A generic backward iterator on points by lines.

```
#include <box_runend_piter.hh>
```

Inherits `site_set_iterator_base< box< P >, box_runend_piter< P > >`.

### Public Member Functions

- `box_runend_piter` (const `box< P > &b`)

*Constructor.*

- void `next` ()

*Go to the next element.*

- unsigned `run_length` () const

*Give the length of the run.*

### 10.86.1 Detailed Description

```
template<typename P> class mln::box_runend_piter< P >
```

A generic backward iterator on points by lines. The parameter *P* is the type of points.

Definition at line 49 of file box\_runend\_piter.hh.

### 10.86.2 Constructor & Destructor Documentation

**10.86.2.1** `template<typename P> mln::box_runend_piter< P >::box_runend_piter ( const box< P > & b )` [**inline**]

Constructor.

#### Parameters

[in] *b* A box.

Definition at line 105 of file box\_runend\_piter.hh.

### 10.86.3 Member Function Documentation

#### 10.86.3.1 void mln::Site\_Iterator< box\_runend\_piter< P > >::next ( ) [inherited]

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

#### Precondition

The iterator is valid.

#### 10.86.3.2 template<typename P > unsigned mln::box\_runend\_piter< P >::run\_length ( ) const [inline]

Give the length of the run.

Definition at line 167 of file box\_runend\_piter.hh.

## 10.87 mln::box\_runstart\_piter< P > Class Template Reference

A generic forward iterator on points by lines.

```
#include <box_runstart_piter.hh>
```

Inherits site\_set\_iterator\_base< box< P >, box\_runstart\_piter< P > >.

### Public Member Functions

- [box\\_runstart\\_piter](#) (const [box](#)< P > &b)

*Constructor.*

- void [next](#) ()

*Go to the next element.*

- unsigned [run\\_length](#) () const

*Give the length of the run.*

#### 10.87.1 Detailed Description

```
template<typename P> class mln::box_runstart_piter< P >
```

A generic forward iterator on points by lines. The parameter P is the type of points.

Definition at line 49 of file box\_runstart\_piter.hh.

## 10.87.2 Constructor & Destructor Documentation

**10.87.2.1** `template<typename P> mln::box_runstart_piter< P >::box_runstart_piter ( const box< P > & b ) [inline]`

Constructor.

### Parameters

[in] *b* A box.

Definition at line 105 of file box\_runstart\_piter.hh.

## 10.87.3 Member Function Documentation

**10.87.3.1** `void mln::Site_Iterator< box_runstart_piter< P > >::next ( ) [inherited]`

Go to the next element.

### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

### Precondition

The iterator is valid.

**10.87.3.2** `template<typename P> unsigned mln::box_runstart_piter< P >::run_length ( ) const [inline]`

Give the lenght of the run.

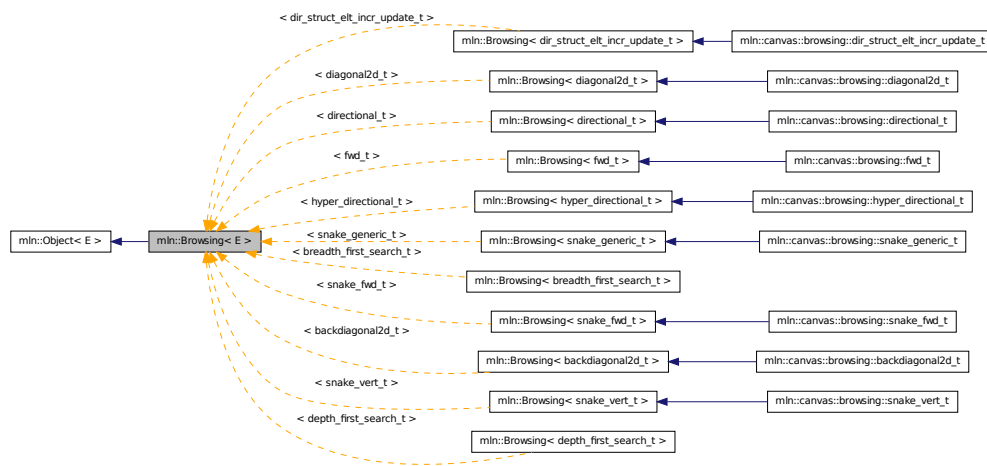
Definition at line 167 of file box\_runstart\_piter.hh.

## 10.88 mln::Browsing< E > Struct Template Reference

Base class for implementation classes that are browsings.

```
#include <browsing.hh>
```

Inheritance diagram for mln::Browsing< E >:



### 10.88.1 Detailed Description

**template<typename E> struct mln::Browsing< E >**

Base class for implementation classes that are browsings.

#### See also

mln::doc::Browsing for a complete documentation of this class contents.

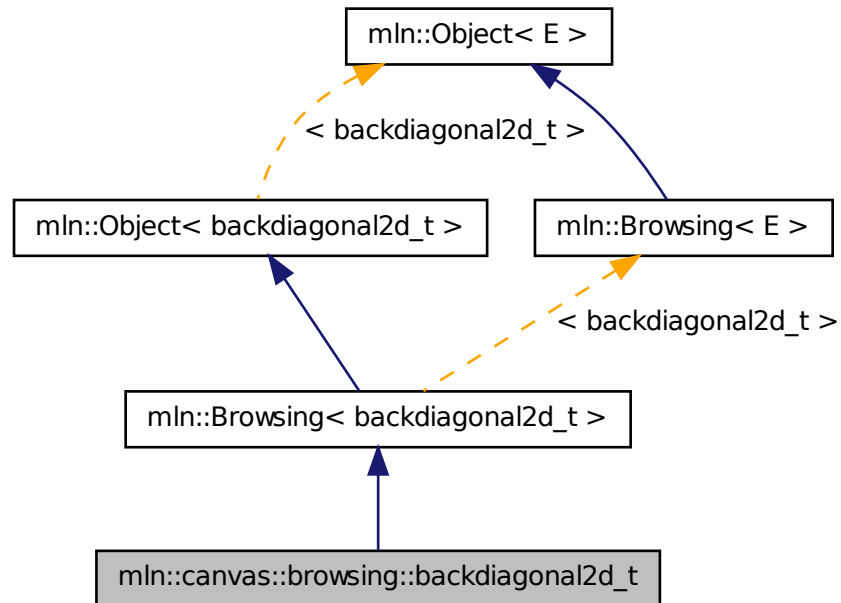
Definition at line 56 of file browsing.hh.

## 10.89 mln::canvas::browsing::backdiagonal2d\_t Struct Reference

[Browsing](#) in a certain direction.

```
#include <backdiagonal2d.hh>
```

Inheritance diagram for `mln::canvas::browsing::backdiagonal2d_t`:



### 10.89.1 Detailed Description

**Browsing** in a certain direction. This canvas browse all the point of an image 'input' of type 'I' and of dimension 'dim' in the direction 'dir'.

The functor should provide (In addition to 'input', 'I', 'dim' and 'dir') three methods :

- `init()` : Will be called at the beginning.
- `next()` : Will be called at each point 'p' (also provided by the functor).
- `final()`: Will be called at the end.

F shall features :

```

{
--- as types:
I;
--- as attributes:
dim;
dir; // and test dir < dim
input;

```



```
p;  
--- as methods:  
void init();  
void next();  
void final();  
}
```

Example :

```
-----> | 4 7 9 | 2 5 8 | 1 3 6
```

Definition at line 83 of file backdiagonal2d.hh.

## 10.90 mln::canvas::browsing::breadth\_first\_search\_t Struct Reference

Breadth-first search algorithm for graph, on vertices.

```
#include <breadth_first_search.hh>
```

Inherits graph\_first\_search\_t< breadth\_first\_search\_t, std::queue >.

### 10.90.1 Detailed Description

Breadth-first search algorithm for graph, on vertices.

Definition at line 79 of file breadth\_first\_search.hh.

## 10.91 mln::canvas::browsing::depth\_first\_search\_t Struct Reference

Breadth-first search algorithm for graph, on vertices.

```
#include <depth_first_search.hh>
```

Inherits graph\_first\_search\_t< depth\_first\_search\_t, std::stack >.

### 10.91.1 Detailed Description

Breadth-first search algorithm for graph, on vertices.

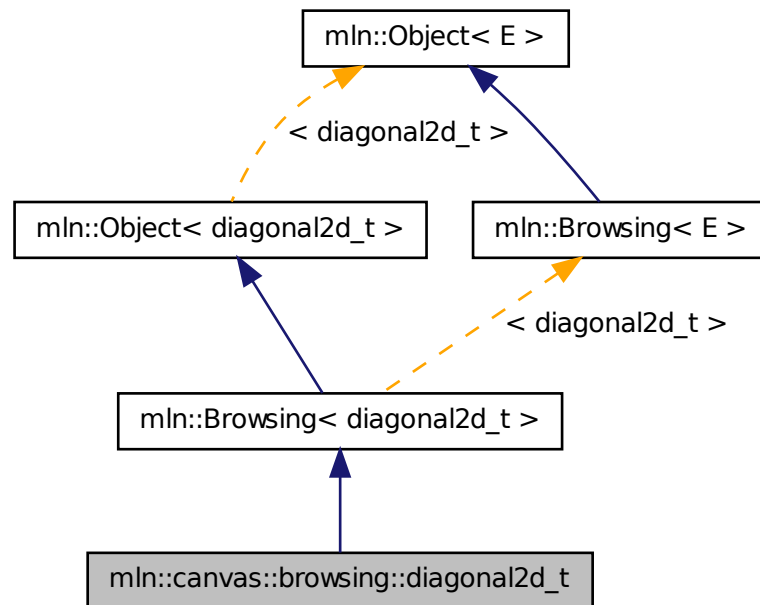
Definition at line 80 of file depth\_first\_search.hh.

## 10.92 mln::canvas::browsing::diagonal2d\_t Struct Reference

[Browsing](#) in a certain direction.

```
#include <diagonal2d.hh>
```

Inheritance diagram for mln::canvas::browsing::diagonal2d\_t:



### 10.92.1 Detailed Description

**Browsing** in a certain direction. This canvas browse all the point of an image 'input' of type 'I' and of dimension 'dim' in the direction 'dir'.

The functor should provide (In addition to 'input', 'I', 'dim' and 'dir') three methods :

- `init()` : Will be called at the beginning.
- `next()` : Will be called at each point 'p' (also provided by the functor).
- `final()` : Will be called at the end.

F shall features :

```

{
--- as types:
I;
--- as attributes:
dim;
dir; // and test dir < dim
input;

```

```

p;
--- as methods:
void init();
void next();
void final();
}

```

Example :

```
| 1 3 6 | 2 5 8 | 4 7 9 L----->
```

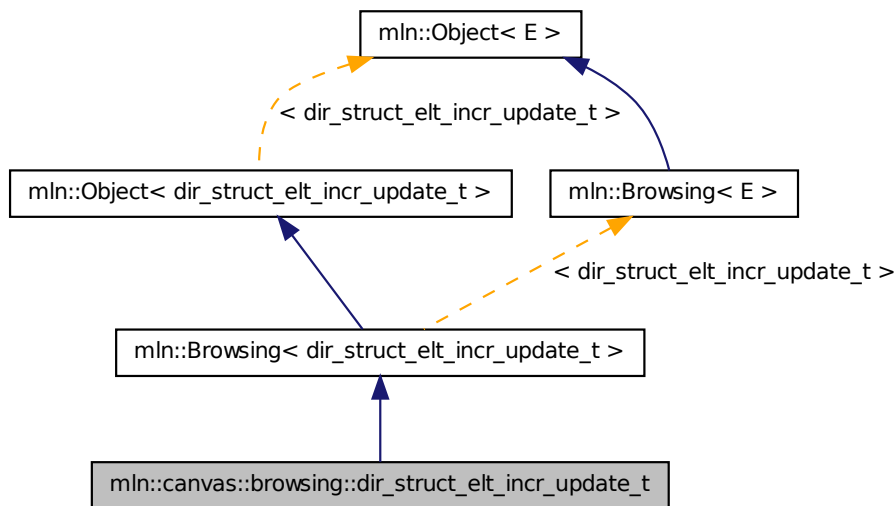
Definition at line 82 of file diagonal2d.hh.

## 10.93 mln::canvas::browsing::dir\_struct\_elt\_incr\_update\_t Struct Reference

**Browsing** in a certain direction with a segment.

```
#include <dir_struct_elt_incr_update.hh>
```

Inheritance diagram for mln::canvas::browsing::dir\_struct\_elt\_incr\_update\_t:



### 10.93.1 Detailed Description

**Browsing** in a certain direction with a segment. This canvas browse all the point of an image 'input' of type 'I', of dimension 'dim' in the direction 'dir' with considering weigh the 'length' nearest points.

The functor should provide (In addition to 'input', 'I', 'dim', 'dir' and 'length') six methods :

- `init()` : Will be called at the beginning.
- `init_line()` : Will be called at the beginning of each line.
- `add_point(q)` : Will be called for taking the new point 'q' into account.
- `remove_point(q)` : Will be called for untaking the new point 'q' into account.
- `next()` : Will be called at each point 'p' (also provided by the functor).
- `final()` : Will be called at the end.

F shall features :

```
{
--- as types:
I;
--- as attributes:
dim;
dir; // and test dir < dim
input;
p;
length;
--- as methods:
void init();
void init_line();
void add_point(q)
void remove_point(q)
void next();
void final();
}
```

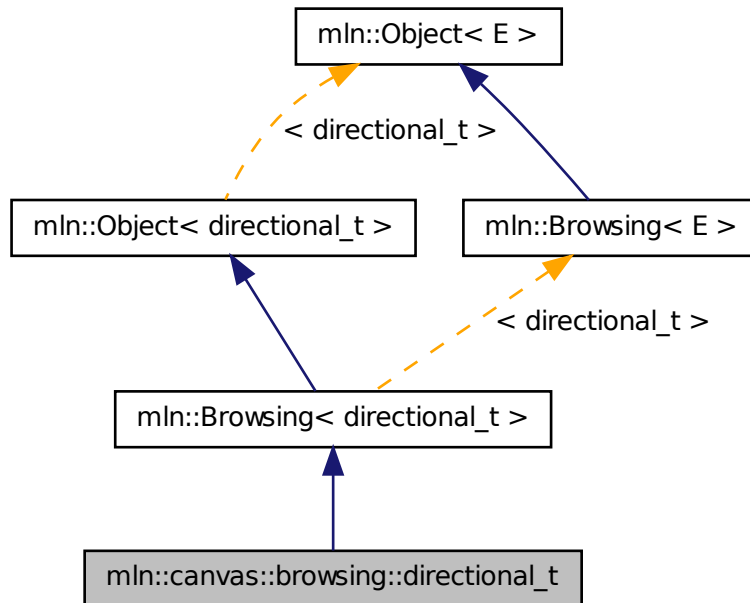
Definition at line 86 of file `dir_struct_elt_incr_update.hh`.

## 10.94 `mln::canvas::browsing::directional_t` Struct Reference

[Browsing](#) in a certain direction.

```
#include <directional.hh>
```

Inheritance diagram for mln::canvas::browsing::directional\_t:



### 10.94.1 Detailed Description

**Browsing** in a certain direction. This canvas browse all the point of an image 'input' of type 'I' and of dimension 'dim' in the direction 'dir'.

The functor should provide (In addition to 'input', 'I', 'dim' and 'dir') three methods :

- `init()` : Will be called at the beginning.
- `next()` : Will be called at each point 'p' (also provided by the functor).
- `final()` : Will be called at the end.

F shall features :

```

{
--- as types:
I;
--- as attributes:
dim;
dir; // and test dir < dim
input;

```

```

p;
--- as methods:
void init();
void next();
void final();
}

```

Example :

```

1 0 0 2 0 0 3 0 0
4 0 0 5 0 0 6 0 0
7 0 0 8 0 0 9 0 0

```

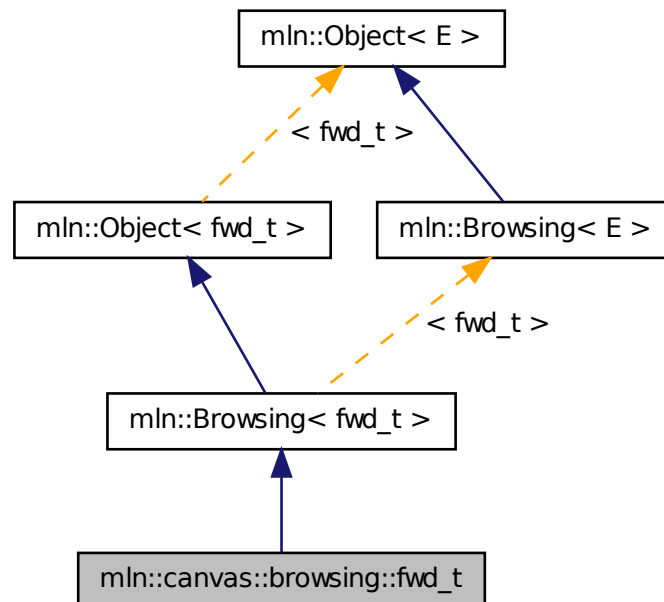
Definition at line 90 of file directional.hh.

## 10.95 mln::canvas::browsing::fwd\_t Struct Reference

Canvas for forward browsing.

```
#include <fwd.hh>
```

Inheritance diagram for mln::canvas::browsing::fwd\_t:



### 10.95.1 Detailed Description

Canvas for forward browsing. This canvas browse all the points of an image 'input' of type 'I' from left to right and from top to bottom

The functor should provide (In addition of 'I' and 'input') three methods :

- `init()` : Will be called at the beginning.
- `next()` : Will be called at each point 'p' (also provided by the functor).
- `final()` : Will be called at the end.

F shall feature:

```
{  
--- as typedef:  
I;  
--as attributes:  
input;  
p;  
--- as method:  
void init();  
void next();  
void final();  
}
```

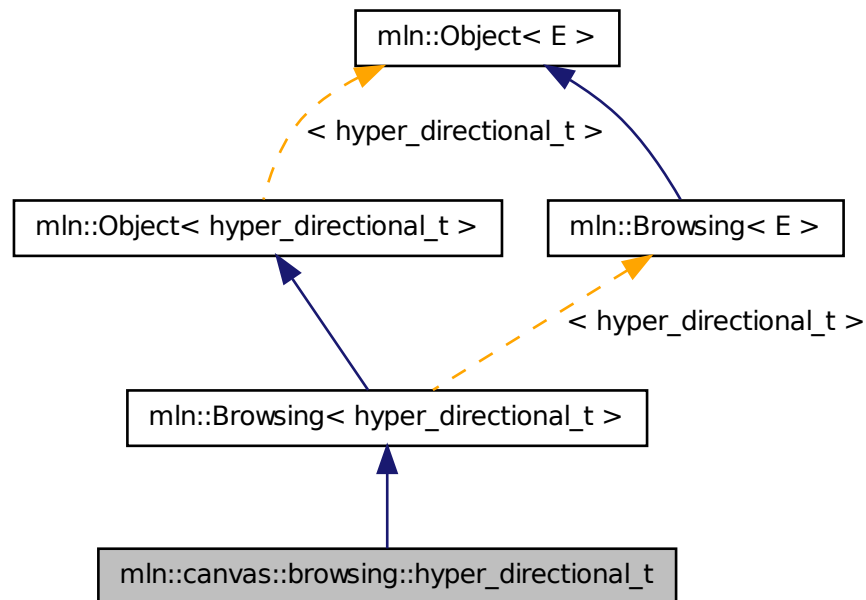
Definition at line 72 of file fwd.hh.

## 10.96 mln::canvas::browsing::hyper\_directional\_t Struct Reference

[Browsing](#) in a certain direction.

```
#include <hyper_directional.hh>
```

Inheritance diagram for mln::canvas::browsing::hyper\_directional\_t:



### 10.96.1 Detailed Description

**Browsing** in a certain direction. This canvas browse all the point of an image 'input' of type 'I' and of dimension 'dim' in the direction 'dir'.

The functor should provide (In addition to 'input', 'I', 'dim' and 'dir') three methods :

- `init()` : Will be called at the beginning.
- `next()` : Will be called at each point 'p' (also provided by the functor).
- `final()` : Will be called at the end.

F shall features :

```

{
--- as types:
I;
--- as attributes:
dim;
dir; // and test dir < dim
input;

```



```

p;
--- as methods:
void init();
void next();
void final();
}

```

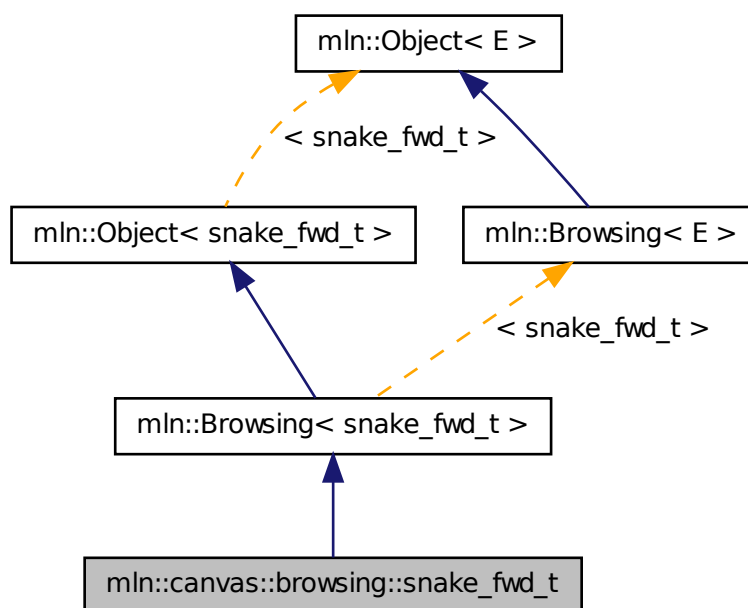
Definition at line 74 of file hyper\_directional.hh.

## 10.97 mln::canvas::browsing::snake\_fwd\_t Struct Reference

[Browsing](#) in a snake-way, forward.

```
#include <snake_fwd.hh>
```

Inheritance diagram for mln::canvas::browsing::snake\_fwd\_t:



### 10.97.1 Detailed Description

[Browsing](#) in a snake-way, forward. This canvas browse all the point of an image 'input' like this :

```
-----> <-----' '----->
```

The functor should provide (In addition to 'input') four methods :

- `init()` : Will be called at the beginning.
- `down()` : Will be called after each moving down. (will also be called once at the first point).
- `fwd()` : Will be called after each moving right.
- `bwd()` : Will be called after each moving left.

This methods should access to the current working point 'p' also provided by the functor.

Warning: This canvas works only on 2D.

F shall feature:

```
{  
--- as attributes:  
input;  
p;  
--- as methods:  
void init();  
void down();  
void fwd();  
void bkd();  
}
```

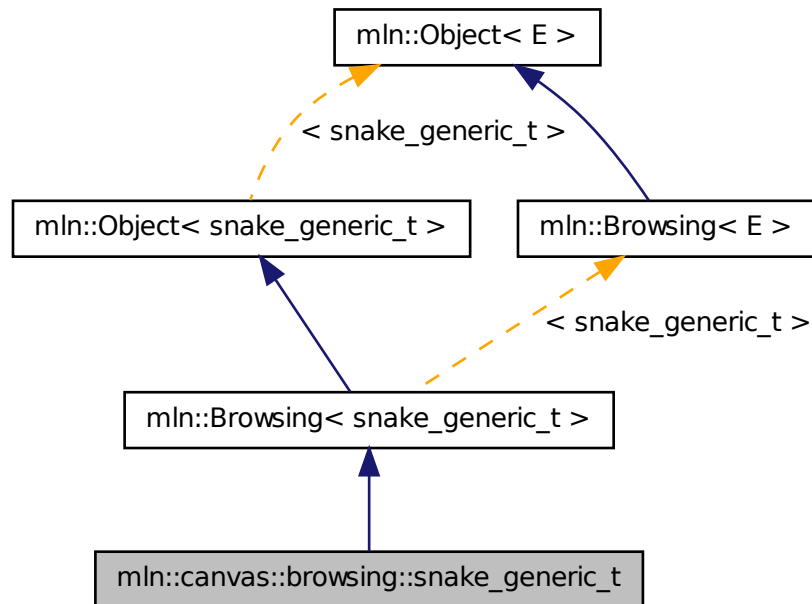
Definition at line 84 of file `snake_fwd.hh`.

## 10.98 `mln::canvas::browsing::snake_generic_t` Struct Reference

Multidimensional [Browsing](#) in a given-way.

```
#include <snake_generic.hh>
```

Inheritance diagram for mln::canvas::browsing::snake\_generic\_t:



### 10.98.1 Detailed Description

Multidimensional [Browsing](#) in a given-way. F shall feature:

```

{
--- as attributes:
input;
p;
--- as methods:
void init();
void *() moves[];
dpsite dps[];
}

```

init is called before browsing

The snake follow dimension using the delta point site of dps. dps[0] = delta psite following the global dimension (forward) dps[1] = delta psite following the 2nd dimension to follow (forward). dps[2] = delta psite following the 2nd dimension to follow (backward). dps[3] = delta psite following the 3rd dimension to follow (forward). dps[3] = delta psite following the 3rd dimension to follow (backward).

moves contains pointer to f's members. These members will be call in each time the snake progress in the

correct dimension :

moves[i] is called at each move following the delta psite dps[i]

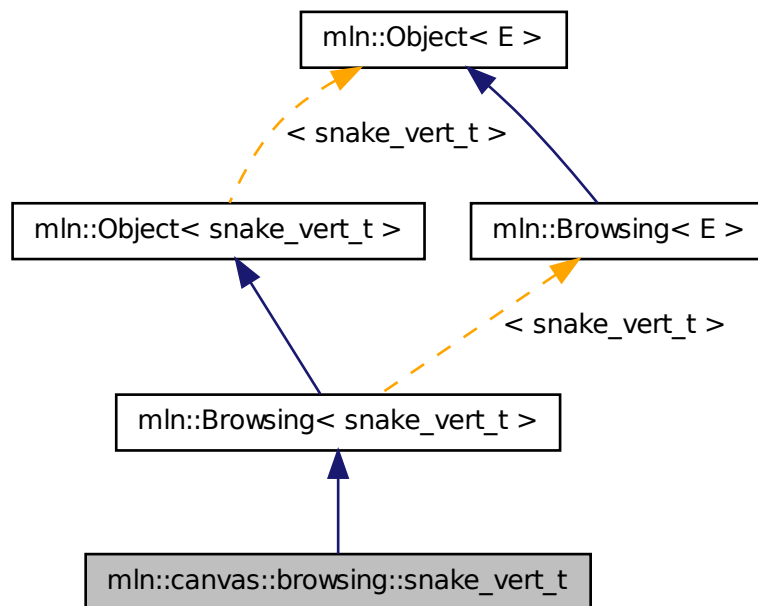
Definition at line 76 of file snake\_generic.hh.

## 10.99 mln::canvas::browsing::snake\_vert\_t Struct Reference

**Browsing** in a snake-way, forward.

```
#include <snake_vert.hh>
```

Inheritance diagram for mln::canvas::browsing::snake\_vert\_t:



### 10.99.1 Detailed Description

**Browsing** in a snake-way, forward. This canvas browse all the point of an image 'input' like this :

```
| ^ | | | | \ | \
```

The functor should provide (In addition to 'input') four methods :

- `init()` : Will be called at the beginning.
- `down()` : Will be called after each moving down.
- `up()` : Will be called after each moving up.

- fwd() : Will be called after each moving right. (will also be called once at the first point).

This methods should acces to the current working point 'p' also provided by the functor.

Warning: This canvas works only on 2D.

F shall feature:

```
{
--- as attributes:
input;
p;
--- as methods:
void init();
void down();
void up();
void fwd();
}
```

Definition at line 83 of file snake\_vert.hh.

## 10.100 mln::canvas::chamfer< F > Struct Template Reference

Compute chamfer distance.

```
#include <chamfer.hh>
```

### 10.100.1 Detailed Description

```
template<typename F> struct mln::canvas::chamfer< F >
```

Compute chamfer distance.

Definition at line 47 of file canvas/chamfer.hh.

## 10.101 mln::category< R(\*) (A) > Struct Template Reference

Category declaration for a unary C function.

```
#include <c.hh>
```

### 10.101.1 Detailed Description

```
template<typename R, typename A> struct mln::category< R(*) (A) >
```

Category declaration for a unary C function.

Definition at line 51 of file c.hh.

## 10.102 mln::complex\_image< D, G, V > Class Template Reference

[Image](#) based on a complex.

```
#include <complex_image.hh>
```

Inherits [image\\_primary](#)< V, [p\\_complex](#)< D, G >, [complex\\_image](#)< D, G, V > >.

### Public Types

- typedef G [geom](#)  
*The geometry type of the complex.*
- typedef V & [lvalue](#)  
*Return type of read-write access.*
- typedef const V & [rvalue](#)  
*Return type of read-only access.*
- typedef [complex\\_image](#)< D, tag::psite\_< G >, tag::value\_< V > > [skeleton](#)  
*Skeleton.*
- typedef V [value](#)  
*Value associated type.*

### Public Member Functions

- [rvalue operator](#)() (const [complex\\_psite](#)< D, G > &p) const  
*Read-only access of face value at point site p.*
- [lvalue operator](#)() (const [complex\\_psite](#)< D, G > &p)  
*Read-write access of face value at point site p.*
- [complex\\_image](#) ()  
*Constructors.*
- const [p\\_complex](#)< D, G > & [domain](#) () const  
*Accessors.*
- const [metal::vec](#)< D+1, [std::vector](#)< [mlc\\_unbool](#)(V) > > & [values](#) () const  
*Return the array of values associated to the faces.*

### Static Public Attributes

- static const unsigned [dim](#) = D  
*The dimension of the complex.*

### 10.102.1 Detailed Description

`template<unsigned D, typename G, typename V> class mln::complex_image< D, G, V >`

[Image](#) based on a complex. Values attached to each face of the complex.

#### Template Parameters

- D* The dimension of the complex.
- G* The geometry type of the complex.
- V* The value type of the image.

Definition at line 164 of file mln/core/image/complex\_image.hh.

### 10.102.2 Member Typedef Documentation

**10.102.2.1** `template<unsigned D, typename G, typename V> typedef G mln::complex_image< D, G, V >::geom`

The geometry type of the complex.

Definition at line 172 of file mln/core/image/complex\_image.hh.

**10.102.2.2** `template<unsigned D, typename G, typename V> typedef V& mln::complex_image< D, G, V >::lvalue`

Return type of read-write access.

Definition at line 177 of file mln/core/image/complex\_image.hh.

**10.102.2.3** `template<unsigned D, typename G, typename V> typedef const V& mln::complex_image< D, G, V >::rvalue`

Return type of read-only access.

Definition at line 180 of file mln/core/image/complex\_image.hh.

**10.102.2.4** `template<unsigned D, typename G, typename V> typedef complex_image< D, tag::psite_<G>, tag::value_<V> > mln::complex_image< D, G, V >::skeleton`

Skeleton.

Definition at line 183 of file mln/core/image/complex\_image.hh.

**10.102.2.5** `template<unsigned D, typename G, typename V> typedef V mln::complex_image< D, G, V >::value`

[Value](#) associated type.

Definition at line 174 of file mln/core/image/complex\_image.hh.

### 10.102.3 Constructor & Destructor Documentation

**10.102.3.1** `template<unsigned D, typename G, typename V > mln::complex_image< D, G, V >::complex_image ( ) [inline]`

Constructors.

Definition at line 276 of file mln/core/image/complex\_image.hh.

### 10.102.4 Member Function Documentation

**10.102.4.1** `template<unsigned D, typename G, typename V > const p_complex< D, G > & mln::complex_image< D, G, V >::domain ( ) const [inline]`

Accessors.

Return the domain of psites of the image.

Definition at line 343 of file mln/core/image/complex\_image.hh.

**10.102.4.2** `template<unsigned D, typename G, typename V > complex_image< D, G, V >::lvalue mln::complex_image< D, G, V >::operator() ( const complex_psite< D, G > & p ) [inline]`

Read-write access of face value at point site p.

Definition at line 326 of file mln/core/image/complex\_image.hh.

References mln::complex\_psite< D, G >::face\_id(), and mln::complex\_psite< D, G >::n().

**10.102.4.3** `template<unsigned D, typename G, typename V > complex_image< D, G, V >::rvalue mln::complex_image< D, G, V >::operator() ( const complex_psite< D, G > & p ) const [inline]`

Read-only access of face value at point site p.

Definition at line 317 of file mln/core/image/complex\_image.hh.

References mln::complex\_psite< D, G >::face\_id(), and mln::complex\_psite< D, G >::n().

**10.102.4.4** `template<unsigned D, typename G, typename V > const metal::vec< D+1, std::vector< mlc_unbool(V) > > & mln::complex_image< D, G, V >::values ( ) const [inline]`

Return the array of values associated to the faces.

Definition at line 335 of file mln/core/image/complex\_image.hh.

### 10.102.5 Member Data Documentation

**10.102.5.1** `template<unsigned D, typename G, typename V> const unsigned mln::complex_image< D, G, V >::dim = D [static]`

The dimension of the complex.



Definition at line 170 of file mln/core/image/complex\_image.hh.

## 10.103 mln::complex\_neighborhood\_bkd\_piter< I, G, N > Class Template Reference

Backward iterator on complex neighborhood.

```
#include <complex_neighborhood_piter.hh>
```

Inherits site\_relative\_iterator\_base< N, complex\_neighborhood\_bkd\_piter< I, G, N > >.

### Public Types

- typedef N::complex\_bkd\_iter [iter\\_type](#)  
*The type of the underlying complex iterator.*
- typedef N::psite [psite](#)  
*The [Pseudo\\_Site](#) type.*

### Public Member Functions

- void [next](#) ()  
*Go to the next element.*
- [complex\\_neighborhood\\_bkd\\_piter](#) ()  
*Construction.*
- const [iter\\_type](#) & [iter](#) () const  
*Accessors.*

#### 10.103.1 Detailed Description

```
template<typename I, typename G, typename N> class mln::complex_neighborhood_bkd_piter< I, G, N >
```

Backward iterator on complex neighborhood.

Definition at line 124 of file complex\_neighborhood\_piter.hh.

#### 10.103.2 Member Typedef Documentation

**10.103.2.1** `template<typename I, typename G, typename N> typedef N::complex_bkd_iter mln::complex_neighborhood_bkd_piter< I, G, N >::iter_type`

The type of the underlying complex iterator.

Definition at line 135 of file complex\_neighborhood\_piter.hh.

**10.103.2.2** `template<typename I, typename G, typename N> typedef N ::psite  
mln::complex_neighborhood_bkd_piter< I, G, N >::psite`

The [Pseudo\\_Site](#) type.

Definition at line 133 of file `complex_neighborhood_piter.hh`.

### 10.103.3 Constructor & Destructor Documentation

**10.103.3.1** `template<typename I , typename G , typename N > mln::complex_  
neighborhood_bkd_piter< I, G, N >::complex_neighborhood_bkd_piter ( )  
[inline]`

Construction.

Definition at line 305 of file `complex_neighborhood_piter.hh`.

### 10.103.4 Member Function Documentation

**10.103.4.1** `template<typename I , typename G , typename N > const N::complex_bkd_iter &  
mln::complex_neighborhood_bkd_piter< I, G, N >::iter ( ) const [inline]`

Accessors.

Definition at line 382 of file `complex_neighborhood_piter.hh`.

**10.103.4.2** `void mln::Site_Iterator< complex_neighborhood_bkd_piter< I, G, N > >::next ( )  
[inherited]`

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

#### Precondition

The iterator is valid.

## 10.104 mln::complex\_neighborhood\_fwd\_piter< I, G, N > Class Template Reference

Forward iterator on complex neighborhood.

```
#include <complex_neighborhood_piter.hh>
```

Inherits `site_relative_iterator_base< N, complex_neighborhood_fwd_piter< I, G, N > >`.

### Public Types

- `typedef N::complex_fwd_iter` [iter\\_type](#)

*The type of the underlying complex iterator.*

- typedef N::psite [psite](#)  
*The [Pseudo\\_Site](#) type.*

## Public Member Functions

- void [next](#) ()  
*Go to the next element.*
- [complex\\_neighborhood\\_fwd\\_piter](#) ()  
*Construction.*
- const [iter\\_type](#) & [iter](#) () const  
*Accessors.*

### 10.104.1 Detailed Description

**template<typename I, typename G, typename N> class mln::complex\_neighborhood\_fwd\_piter< I, G, N >**

Forward iterator on complex neighborhood.

Definition at line 53 of file complex\_neighborhood\_piter.hh.

### 10.104.2 Member Typedef Documentation

**10.104.2.1 template<typename I, typename G, typename N> typedef N::complex\_fwd\_iter mln::complex\_neighborhood\_fwd\_piter< I, G, N >::iter\_type**

The type of the underlying complex iterator.

Definition at line 64 of file complex\_neighborhood\_piter.hh.

**10.104.2.2 template<typename I, typename G, typename N> typedef N ::psite mln::complex\_neighborhood\_fwd\_piter< I, G, N >::psite**

The [Pseudo\\_Site](#) type.

Definition at line 62 of file complex\_neighborhood\_piter.hh.

### 10.104.3 Constructor & Destructor Documentation

**10.104.3.1 template<typename I , typename G , typename N > mln::complex\_neighborhood\_fwd\_piter< I, G, N >::complex\_neighborhood\_fwd\_piter ( ) [inline]**

Construction.

Definition at line 198 of file `complex_neighborhood_piter.hh`.

#### 10.104.4 Member Function Documentation

**10.104.4.1** `template<typename I , typename G , typename N > const N::complex_fwd_iter & mln::complex_neighborhood_fwd_piter< I, G, N >::iter ( ) const [inline]`

Accessors.

Definition at line 275 of file `complex_neighborhood_piter.hh`.

**10.104.4.2** `void mln::Site_Iterator< complex_neighborhood_fwd_piter< I, G, N > >::next ( ) [inherited]`

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

#### Precondition

The iterator is valid.

## 10.105 mln::complex\_psite< D, G > Class Template Reference

[Point](#) site associated to a [mln::p\\_complex](#).

```
#include <complex_psite.hh>
```

Inherits `pseudo_site_base_< const G::site &, complex_psite< D, G > >`.

### Public Member Functions

- [complex\\_psite](#) ()  
*Construction and assignment.*
- [complex\\_psite](#) (const [p\\_complex](#)< D, G > &pc, const [topo::face](#)< D > &face)
- bool [is\\_valid](#) () const  
*Psite manipulators.*
- void [invalidate](#) ()  
*Invalidate this psite.*
- const [target](#) & [site\\_set](#) () const  
*Site set manipulators.*
- void [change\\_target](#) (const [target](#) &new\_target)  
*Set the target site\_set.*

- const [topo::face](#)< D > & [face](#) () const  
*Face handle manipulators.*
- unsigned [n](#) () const  
*Return the dimension of the face of this psite.*
- unsigned [face\\_id](#) () const  
*Return the id of the face of this psite.*

### 10.105.1 Detailed Description

`template<unsigned D, typename G> class mln::complex_psite< D, G >`

[Point](#) site associated to a [mln::p\\_complex](#).

#### Template Parameters

- D* The dimension of the complex this psite belongs to.
- G* The geometry of the complex.

Definition at line 60 of file `complex_psite.hh`.

### 10.105.2 Constructor & Destructor Documentation

**10.105.2.1** `template<unsigned D, typename G > mln::complex_psite< D, G >::complex_psite ( ) [inline]`

Construction and assignment.

Definition at line 203 of file `complex_psite.hh`.

References `mln::complex_psite< D, G >::invalidate()`.

**10.105.2.2** `template<unsigned D, typename G > mln::complex_psite< D, G >::complex_psite ( const p_complex< D, G > & pc, const topo::face< D > & face ) [inline]`

#### Precondition

`pc.cplx() == face.cplx()`.

Definition at line 211 of file `complex_psite.hh`.

References `mln::topo::face< D >::cplx()`, `mln::p_complex< D, G >::cplx()`, and `mln::complex_psite< D, G >::is_valid()`.

### 10.105.3 Member Function Documentation

**10.105.3.1** `template<unsigned D, typename G > void mln::complex_psite< D, G >::change_target ( const target & new_target ) [inline]`

Set the target `site_set`.

Definition at line 280 of file complex\_psite.hh.

References mln::p\_complex< D, G >::cplx(), and mln::complex\_psite< D, G >::invalidate().

**10.105.3.2** `template<unsigned D, typename G > const topo::face< D > & mln::complex_psite< D, G >::face ( ) const [inline]`

Face handle manipulators.

Return the face handle of this point site.

Definition at line 301 of file complex\_psite.hh.

Referenced by mln::operator!==( ), and mln::operator==( ).

**10.105.3.3** `template<unsigned D, typename G > unsigned mln::complex_psite< D, G >::face_id ( ) const [inline]`

Return the id of the face of this psite.

Definition at line 317 of file complex\_psite.hh.

Referenced by mln::complex\_image< D, G, V >::operator()( ).

**10.105.3.4** `template<unsigned D, typename G > void mln::complex_psite< D, G >::invalidate ( ) [inline]`

Invalidate this psite.

Definition at line 251 of file complex\_psite.hh.

Referenced by mln::complex\_psite< D, G >::change\_target(), and mln::complex\_psite< D, G >::complex\_psite().

**10.105.3.5** `template<unsigned D, typename G > bool mln::complex_psite< D, G >::is_valid ( ) const [inline]`

Psite manipulators.

Is this psite valid?

Definition at line 239 of file complex\_psite.hh.

Referenced by mln::complex\_psite< D, G >::complex\_psite(), and mln::p\_complex< D, G >::has().

**10.105.3.6** `template<unsigned D, typename G > unsigned mln::complex_psite< D, G >::n ( ) const [inline]`

Return the dimension of the face of this psite.

Definition at line 309 of file complex\_psite.hh.

Referenced by mln::make::cell(), and mln::complex\_image< D, G, V >::operator()( ).

### 10.105.3.7 `template<unsigned D, typename G > const p_complex< D, G > & mln::complex_psite< D, G >::site_set( ) const [inline]`

[Site](#) set manipulators.

Return the [mln::p\\_complex](#) this site is built on. (shortcut for \*target()).

#### Precondition

Member `face_` is valid.

Definition at line 259 of file `complex_psite.hh`.

Referenced by `mln::p_complex< D, G >::has()`, `mln::operator!=( )`, and `mln::operator==( )`.

## 10.106 mln::complex\_window\_bkd\_piter< I, G, W > Class Template Reference

Backward iterator on complex window.

```
#include <complex_window_piter.hh>
```

Inherits `site_relative_iterator_base< W, complex_window_bkd_piter< I, G, W > >`.

### Public Types

- typedef `W::complex_bkd_iter` [iter\\_type](#)  
*The type of the underlying complex iterator.*
- typedef `W::psite` [psite](#)  
*The Pseudo\_Site type.*

### Public Member Functions

- void [next](#) ()  
*Go to the next element.*
- [complex\\_window\\_bkd\\_piter](#) ()  
*Construction.*
- const [iter\\_type](#) & [iter](#) () const  
*Accessors.*

### 10.106.1 Detailed Description

```
template<typename I, typename G, typename W> class mln::complex_window_bkd_piter< I, G, W >
```

Backward iterator on complex window.

Definition at line 124 of file `complex_window_piter.hh`.

## 10.106.2 Member Typedef Documentation

**10.106.2.1** `template<typename I, typename G, typename W> typedef W::complex_bkd_iter  
mln::complex_window_bkd_piter< I, G, W >::iter_type`

The type of the underlying complex iterator.

Definition at line 135 of file `complex_window_piter.hh`.

**10.106.2.2** `template<typename I, typename G, typename W> typedef W ::psite  
mln::complex_window_bkd_piter< I, G, W >::psite`

The [Pseudo\\_Site](#) type.

Definition at line 133 of file `complex_window_piter.hh`.

## 10.106.3 Constructor & Destructor Documentation

**10.106.3.1** `template<typename I , typename G , typename W > mln::complex_  
window_bkd_piter< I, G, W >::complex_window_bkd_piter ( )  
[inline]`

Construction.

Definition at line 305 of file `complex_window_piter.hh`.

## 10.106.4 Member Function Documentation

**10.106.4.1** `template<typename I , typename G , typename W > const W::complex_bkd_iter &  
mln::complex_window_bkd_piter< I, G, W >::iter ( ) const [inline]`

Accessors.

Definition at line 384 of file `complex_window_piter.hh`.

**10.106.4.2** `void mln::Site_Iterator< complex_window_bkd_piter< I, G, W > >::next ( )  
[inherited]`

Go to the next element.

### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

### Precondition

The iterator is valid.



## 10.107 mln::complex\_window\_fwd\_piter< I, G, W > Class Template Reference

Forward iterator on complex window.

```
#include <complex_window_piter.hh>
```

Inherits `site_relative_iterator_base< W, complex_window_fwd_piter< I, G, W > >`.

### Public Types

- typedef `W::complex_fwd_iter` `iter_type`  
*The type of the underlying complex iterator.*
- typedef `W::psite` `psite`  
*The `Pseudo_Site` type.*

### Public Member Functions

- void `next ()`  
*Go to the next element.*
- `complex_window_fwd_piter ()`  
*Construction.*
- const `iter_type & iter () const`  
*Accessors.*

#### 10.107.1 Detailed Description

```
template<typename I, typename G, typename W> class mln::complex_window_fwd_piter< I, G, W >
```

Forward iterator on complex window.

Definition at line 54 of file `complex_window_piter.hh`.

#### 10.107.2 Member Typedef Documentation

**10.107.2.1** `template<typename I, typename G, typename W> typedef W::complex_fwd_iter mln::complex_window_fwd_piter< I, G, W >::iter_type`

The type of the underlying complex iterator.

Definition at line 65 of file `complex_window_piter.hh`.

**10.107.2.2** `template<typename I, typename G, typename W> typedef W ::psite  
mln::complex_window_fwd_piter< I, G, W >::psite`

The [Pseudo\\_Site](#) type.

Definition at line 63 of file `complex_window_piter.hh`.

### 10.107.3 Constructor & Destructor Documentation

**10.107.3.1** `template<typename I , typename G , typename W > mln::complex_  
window_fwd_piter< I, G, W >::complex_window_fwd_piter ( )  
[inline]`

Construction.

Definition at line 197 of file `complex_window_piter.hh`.

### 10.107.4 Member Function Documentation

**10.107.4.1** `template<typename I , typename G , typename W > const W::complex_fwd_iter &  
mln::complex_window_fwd_piter< I, G, W >::iter ( ) const [inline]`

Accessors.

Definition at line 275 of file `complex_window_piter.hh`.

**10.107.4.2** `void mln::Site_Iterator< complex_window_fwd_piter< I, G, W > >::next ( )  
[inherited]`

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

#### Precondition

The iterator is valid.

## 10.108 mln::decorated\_image< I, D > Struct Template Reference

[Image](#) that can have additional features.

```
#include <decorated_image.hh>
```

Inherits `decorated_image_impl< I, decorated_image< I, D > >`, and `image_identity< I, I::domain_t, decorated_image< I, D > >`.

### Public Types

- typedef `impl_::lvalue lvalue`

*Return type of read-write access.*

- typedef I::psite [psite](#)  
*Type of the psite.*
- typedef I::rvalue [rvalue](#)  
*Return type of read-only access.*
- typedef [decorated\\_image](#)< tag::image\_< I >, tag::data\_< D > > [skeleton](#)  
*Skeleton.*

## Public Member Functions

- [decorated\\_image](#) ()  
*Ctors.*
- D & [decoration](#) ()  
*Give the decoration.*
- const D & [decoration](#) () const  
*Give the decoration.*
- [operator decorated\\_image](#)< const I, D > () const  
*Const promotion via conversion.*
- [rvalue operator](#)() (const [psite](#) &p) const  
*Read-only access of pixel value at point site p.*
- [lvalue operator](#)() (const [psite](#) &p)  
*Read-write access of pixel value at point site p.*
- [~decorated\\_image](#) ()  
*Dtor.*

### 10.108.1 Detailed Description

`template<typename I, typename D> struct mln::decorated_image< I, D >`

[Image](#) that can have additional features.

Definition at line 81 of file `decorated_image.hh`.

### 10.108.2 Member Typedef Documentation

**10.108.2.1** `template<typename I, typename D> typedef impl_::lvalue mln::decorated_image< I, D >::lvalue`

Return type of read-write access.

Definition at line 95 of file decorated\_image.hh.

**10.108.2.2** `template<typename I, typename D> typedef I ::psite mln::decorated_image< I, D >::psite`

Type of the psite.

Definition at line 90 of file decorated\_image.hh.

**10.108.2.3** `template<typename I, typename D> typedef I ::rvalue mln::decorated_image< I, D >::rvalue`

Return type of read-only access.

Definition at line 93 of file decorated\_image.hh.

**10.108.2.4** `template<typename I, typename D> typedef decorated_image< tag::image_<I>, tag::data_<D> > mln::decorated_image< I, D >::skeleton`

Skeleton.

Definition at line 108 of file decorated\_image.hh.

### 10.108.3 Constructor & Destructor Documentation

**10.108.3.1** `template<typename I , typename D > mln::decorated_image< I, D >::decorated_image ( ) [inline]`

Ctors.

Definition at line 161 of file decorated\_image.hh.

**10.108.3.2** `template<typename I , typename D > mln::decorated_image< I, D >::~~decorated_image ( ) [inline]`

Dtor.

Definition at line 184 of file decorated\_image.hh.

### 10.108.4 Member Function Documentation

**10.108.4.1** `template<typename I , typename D > const D & mln::decorated_image< I, D >::decoration ( ) const [inline]`

Give the decoration.

Definition at line 249 of file decorated\_image.hh.

**10.108.4.2** `template<typename I , typename D > D & mln::decorated_image< I, D >::decoration ( ) [inline]`

Give the decoration.

Definition at line 257 of file decorated\_image.hh.

**10.108.4.3** `template<typename I , typename D > mln::decorated_image< I, D >::operator decorated_image< const I, D > ( ) const [inline]`

Const promotion via conversion.

Definition at line 239 of file decorated\_image.hh.

**10.108.4.4** `template<typename I , typename D > decorated_image< I, D >::rvalue mln::decorated_image< I, D >::operator() ( const psite & p ) const [inline]`

Read-only access of pixel value at point site p.

Definition at line 197 of file decorated\_image.hh.

**10.108.4.5** `template<typename I , typename D > decorated_image< I, D >::lvalue mln::decorated_image< I, D >::operator() ( const psite & p ) [inline]`

Read-write access of pixel value at point site p.

Definition at line 206 of file decorated\_image.hh.

## 10.109 mln::Delta\_Point\_Site< void > Struct Template Reference

Delta point site category flag type.

```
#include <delta_point_site.hh>
```

### 10.109.1 Detailed Description

```
template<> struct mln::Delta_Point_Site< void >
```

Delta point site category flag type.

Definition at line 70 of file delta\_point\_site.hh.

## 10.110 mln::doc::Accumulator< E > Struct Template Reference

Documentation class for [mln::Accumulator](#).

```
#include <accumulator.hh>
```

### Public Types

- typedef void [argument](#)

*The argument type of elements to accumulate.*

## Public Member Functions

- void [init](#) ()  
*Initialize the accumulator.*
- void [take](#) (const E &other)  
*Take into account another accumulator `other`.*
- void [take](#) (const [argument](#) &t)  
*Take into account a argument `t` (an element).*

### 10.110.1 Detailed Description

**template<typename E> struct mln::doc::Accumulator< E >**

Documentation class for [mln::Accumulator](#).

See also

[mln::Accumulator](#)

Definition at line 36 of file `doc/accumulator.hh`.

### 10.110.2 Member Typedef Documentation

**10.110.2.1 template<typename E > typedef void mln::doc::Accumulator< E >::argument**

The argument type of elements to accumulate.

Definition at line 39 of file `doc/accumulator.hh`.

### 10.110.3 Member Function Documentation

**10.110.3.1 template<typename E > void mln::doc::Accumulator< E >::init ( )**

Initialize the accumulator.

**10.110.3.2 template<typename E > void mln::doc::Accumulator< E >::take ( const E & other )**

Take into account another accumulator `other`.

**10.110.3.3 template<typename E > void mln::doc::Accumulator< E >::take ( const argument & t )**

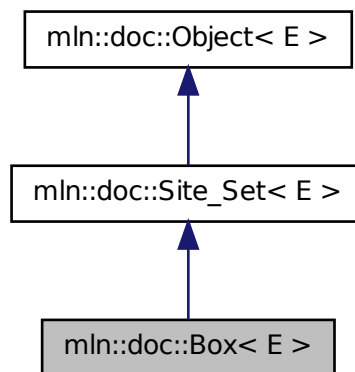
Take into account a argument `t` (an element).

## 10.111 mln::doc::Box< E > Struct Template Reference

Documentation class for [mln::Box](#).

```
#include <box.hh>
```

Inheritance diagram for mln::doc::Box< E >:



### Public Types

- typedef void [bkd\\_piter](#)  
*Backward [Site\\_Iterator](#) associated type.*
- typedef void [fwd\\_piter](#)  
*Forward [Site\\_Iterator](#) associated type.*
- typedef void [psite](#)  
*PSite associated type.*
- typedef void [site](#)  
*Site associated type.*

### Public Member Functions

- const E & [bbox](#) () const  
*Return the bounding box of this point set.*
- bool [has](#) (const [psite](#) &p) const  
*Test if *p* belongs to this site set.*

- unsigned [nsites](#) () const  
*Return the number of points of this box.*
- const [site](#) & [pmax](#) () const  
*Give the box "maximum" point.*
- const [site](#) & [pmin](#) () const  
*Give the box "minimum" point.*

### 10.111.1 Detailed Description

**template<typename E> struct mln::doc::Box< E >**

Documentation class for [mln::Box](#).

See also

[mln::Box](#)

Definition at line 36 of file core/concept/doc/box.hh.

### 10.111.2 Member Typedef Documentation

**10.111.2.1 template<typename E > typedef void mln::doc::Site\_Set< E >::bkd\_piter  
[inherited]**

Backward [Site\\_Iterator](#) associated type.

Definition at line 53 of file mln/core/concept/doc/site\_set.hh.

**10.111.2.2 template<typename E > typedef void mln::doc::Site\_Set< E >::fwd\_piter  
[inherited]**

Forward [Site\\_Iterator](#) associated type.

Definition at line 49 of file mln/core/concept/doc/site\_set.hh.

**10.111.2.3 template<typename E > typedef void mln::doc::Site\_Set< E >::psite [inherited]**

PSite associated type.

Definition at line 45 of file mln/core/concept/doc/site\_set.hh.

**10.111.2.4 template<typename E > typedef void mln::doc::Site\_Set< E >::site [inherited]**

[Site](#) associated type.

Definition at line 41 of file mln/core/concept/doc/site\_set.hh.



### 10.111.3 Member Function Documentation

#### 10.111.3.1 `template<typename E> const E& mln::doc::Box< E >::bbox ( ) const`

Return the bounding box of this point set.

Return the bounding box of this point set, so that is itself. This method is declared by the [mln::Site\\_Set](#) concept.

#### Warning

This method is final for all box classes.

#### 10.111.3.2 `template<typename E> bool mln::doc::Site_Set< E >::has ( const psite & p ) const [inherited]`

Test if *p* belongs to this site set.

#### Parameters

[in] *p* A psite.

#### Returns

True if *p* is an element of the site set.

#### 10.111.3.3 `template<typename E> unsigned mln::doc::Box< E >::nsites ( ) const`

Return the number of points of this box.

Return the number of points of this box. This method is declared by the [mln::Site\\_Set](#) concept.

#### Warning

This method is final for all box classes.

#### 10.111.3.4 `template<typename E> const site& mln::doc::Box< E >::pmax ( ) const`

Give the box "maximum" point.

Return the "maximum" point w.r.t. the ordering between points. For instance, with [mln::box2d](#), this maximum is the bottom right point of the box.

#### 10.111.3.5 `template<typename E> const site& mln::doc::Box< E >::pmin ( ) const`

Give the box "minimum" point.

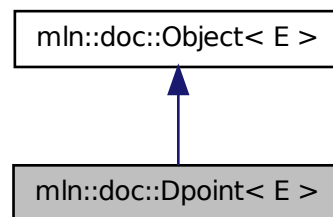
Return the "minimum" point w.r.t. the ordering between points. For instance, with [mln::box2d](#), this minimum is the top left point of the box.

## 10.112 mln::doc::Dpoint< E > Struct Template Reference

Documentation class for [mln::Dpoint](#).

```
#include <dpoint.hh>
```

Inheritance diagram for mln::doc::Dpoint< E >:



### Public Types

- enum { [dim](#) }
- typedef void [coord](#)
- typedef void [dpoint](#)  
*Dpsite associated type.*
- typedef void [point](#)  
*Site associated type.*

### Public Member Functions

- [coord operator\[ \]](#) (unsigned i) const  
*Read-only access to the *i*-th coordinate value.*

#### 10.112.1 Detailed Description

```
template<typename E> struct mln::doc::Dpoint< E >
```

Documentation class for [mln::Dpoint](#).

See also

[mln::Dpoint](#)

Definition at line 36 of file concept/doc/dpoint.hh.

## 10.112.2 Member Typedef Documentation

### 10.112.2.1 `template<typename E > typedef void mln::doc::Dpoint< E >::coord`

Coordinate associated type.

Definition at line 56 of file concept/doc/dpoint.hh.

### 10.112.2.2 `template<typename E > typedef void mln::doc::Dpoint< E >::dpoint`

Dpsite associated type.

#### Invariant

This type has to derive from [mln::Dpoint](#).

Definition at line 52 of file concept/doc/dpoint.hh.

### 10.112.2.3 `template<typename E > typedef void mln::doc::Dpoint< E >::point`

[Site](#) associated type.

#### Invariant

This type has to derive from [mln::Point](#).

Definition at line 47 of file concept/doc/dpoint.hh.

## 10.112.3 Member Enumeration Documentation

### 10.112.3.1 `template<typename E > anonymous enum`

#### Enumerator:

*dim* Dimension of the space.

#### Invariant

`dim > 0`

Definition at line 42 of file concept/doc/dpoint.hh.

## 10.112.4 Member Function Documentation

### 10.112.4.1 `template<typename E > coord mln::doc::Dpoint< E >::operator[] ( unsigned i ) const`

Read-only access to the `i`-th coordinate value.

#### Parameters

`[in]` *i* The coordinate index.

**Precondition**

$i < \text{dim}$

**Returns**

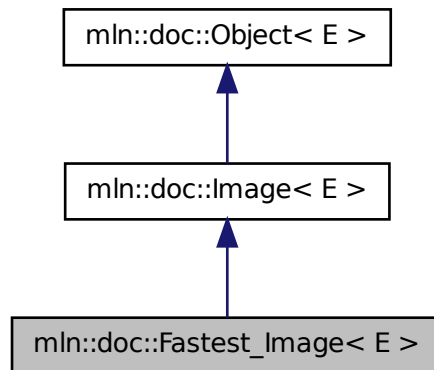
The value of the  $i$ -th coordinate.

## 10.113 `mln::doc::Fastest_Image< E >` Struct Template Reference

Documentation class for the concept of images that have the speed property set to "fastest".

```
#include <image_fastest.hh>
```

Inheritance diagram for `mln::doc::Fastest_Image< E >`:

**Public Types**

- typedef void `bkd_piter`  
*Backward point iterator associated type.*
- typedef void `coord`  
*Coordinate associated type.*
- typedef void `dpoint`  
*Dpsite associated type.*
- typedef void `fwd_piter`  
*Forward point iterator associated type.*
- typedef void `lvalue`  
*Type returned by the read-write pixel value operator.*

- typedef void [point](#)  
*Site associated type.*
- typedef void [pset](#)  
*Point set associated type.*
- typedef void [psite](#)  
*Point\_Site associated type.*
- typedef void [rvalue](#)  
*Type returned by the read pixel value operator.*
- typedef void [skeleton](#)  
*Associate type that describes how this type of image is constructed.*
- typedef void [value](#)  
*Value associated type.*
- typedef void [vset](#)  
*Value set associated type.*

## Public Member Functions

- const [box](#)< [point](#) > & [bbox](#) () const  
*Give a bounding box of the image domain.*
- unsigned [border](#) ()  
*Give the border thickness.*
- const [value](#) \* [buffer](#) () const  
*Give a hook to the value buffer.*
- int [delta\\_index](#) (const [dpoint](#) &dp)  
*Give the offset corresponding to the delta-point dp.*
- const [pset](#) & [domain](#) () const  
*Give the definition domain of the image.*
- bool [has](#) (const [psite](#) &p) const  
*Test if the image owns the point site p.*
- bool [has](#) (const [psite](#) &p) const  
*Test if p belongs to the image domain.*
- bool [is\\_valid](#) () const  
*Test if the image have been initialized.*

- unsigned `nelements` () const  
*Give the number of pixels of the image including those of the virtual border.*
- unsigned `nsites` () const  
*Give the number of points of the image domain.*
- `lvalue operator()` (const `psite` &p)  
*Read-write access to the image value located at p.*
- `rvalue operator()` (const `psite` &p) const  
*Read-only access to the image value located at p.*
- `rvalue operator[]` (unsigned o) const  
*Read-only access to the image value at offset o.*
- `lvalue operator[]` (unsigned o)  
*Read-write access to the image value at offset o.*
- `point point_at_index` (unsigned o) const  
*Give the point at offset o.*
- const `vset` & `values` () const  
*Give the set of values of the image.*

### 10.113.1 Detailed Description

`template<typename E> struct mln::doc::Fastest_Image< E >`

Documentation class for the concept of images that have the speed property set to "fastest".

Definition at line 36 of file `concept/doc/image_fastest.hh`.

### 10.113.2 Member Typedef Documentation

**10.113.2.1** `template<typename E > typedef void mln::doc::Image< E >::bkd_piter [inherited]`

Backward point iterator associated type.

#### Invariant

This type has to derive from `mln::Site_Iterator`.

Definition at line 147 of file `core/concept/doc/image.hh`.

**10.113.2.2** `template<typename E > typedef void mln::doc::Image< E >::coord [inherited]`

Coordinate associated type.

Definition at line 131 of file `core/concept/doc/image.hh`.

**10.113.2.3** `template<typename E > typedef void mln::doc::Image< E >::dpoint` `[inherited]`

Dpsite associated type.

**Invariant**

This type has to derive from [mln::Dpoint](#).

Definition at line 136 of file core/concept/doc/image.hh.

**10.113.2.4** `template<typename E > typedef void mln::doc::Image< E >::fwd_piter` `[inherited]`

Forward point iterator associated type.

**Invariant**

This type has to derive from [mln::Site\\_Iterator](#).

Definition at line 142 of file core/concept/doc/image.hh.

**10.113.2.5** `template<typename E > typedef void mln::doc::Image< E >::lvalue` `[inherited]`

Type returned by the read-write pixel value operator.

Definition at line 52 of file core/concept/doc/image.hh.

**10.113.2.6** `template<typename E > typedef void mln::doc::Image< E >::point` `[inherited]`

[Site](#) associated type.

**Invariant**

This type has to derive from [mln::Point](#).

Definition at line 121 of file core/concept/doc/image.hh.

**10.113.2.7** `template<typename E > typedef void mln::doc::Image< E >::pset` `[inherited]`

[Point](#) set associated type.

**Invariant**

This type has to derive from [mln::Site\\_Set](#).

Definition at line 116 of file core/concept/doc/image.hh.

**10.113.2.8** `template<typename E > typedef void mln::doc::Image< E >::psite` `[inherited]`

[Point\\_Site](#) associated type.

**Invariant**

This type has to derive from [mln::Point\\_Site](#).

Definition at line 126 of file core/concept/doc/image.hh.

**10.113.2.9 `template<typename E > typedef void mln::doc::Image< E >::rvalue` [inherited]**

Type returned by the read pixel value operator.

Definition at line 48 of file core/concept/doc/image.hh.

**10.113.2.10 `template<typename E > typedef void mln::doc::Image< E >::skeleton` [inherited]**

Associate type that describes how this type of image is constructed.

Definition at line 64 of file core/concept/doc/image.hh.

**10.113.2.11 `template<typename E > typedef void mln::doc::Image< E >::value` [inherited]**

[Value](#) associated type.

**Invariant**

This type is neither qualified by const, nor by reference.

Definition at line 44 of file core/concept/doc/image.hh.

**10.113.2.12 `template<typename E > typedef void mln::doc::Image< E >::vset` [inherited]**

[Value](#) set associated type.

**Invariant**

This type has to derive from [mln::Value\\_Set](#).

Definition at line 57 of file core/concept/doc/image.hh.

**10.113.3 Member Function Documentation****10.113.3.1 `template<typename E > const box<point>& mln::doc::Image< E >::bbox ( ) const` [inherited]**

Give a bounding box of the image domain.

This bounding box may be larger than the smallest bounding box (the optimal one). Practically an image type is not obliged to update its bounding box so that it is always optimal.

**Returns**

A bounding box of the image domain.



**10.113.3.2** `template<typename E> unsigned mln::doc::Fastest_Image< E >::border ( )`

Give the border thickness.

**Precondition**

The image has to be initialized.

**10.113.3.3** `template<typename E> const value* mln::doc::Fastest_Image< E >::buffer ( ) const`

Give a hook to the value buffer.

**Precondition**

The image has to be initialized.

**10.113.3.4** `template<typename E> int mln::doc::Fastest_Image< E >::delta_index ( const dpoint & dp )`

Give the offset corresponding to the delta-point dp.

**Parameters**

[in] *dp* A delta-point.

**Precondition**

The image has to be initialized.

**10.113.3.5** `template<typename E> const pset& mln::doc::Image< E >::domain ( ) const [inherited]`

Give the definition domain of the image.

**Returns**

A reference to the domain point set.

**10.113.3.6** `template<typename E> bool mln::doc::Image< E >::has ( const psite & p ) const [inherited]`

Test if the image owns the point site p.

**Returns**

True if accessing the image value at p is possible, that is, does not abort the execution.

**10.113.3.7** `template<typename E> bool mln::doc::Image< E >::has ( const psite & p ) const`  
`[inherited]`

Test if *p* belongs to the image domain.

#### Parameters

[in] *p* A point site.

#### Returns

True if *p* belongs to the image domain.

#### Invariant

has(*p*) is true => has(*p*) is also true.

**10.113.3.8** `template<typename E> bool mln::doc::Image< E >::is_valid ( ) const`  
`[inherited]`

Test if the image have been initialized.

**10.113.3.9** `template<typename E> unsigned mln::doc::Fastest_Image< E >::nelements ( )`  
`const`

Give the number of pixels of the image including those of the virtual border.

#### Precondition

The image has to be initialized.

**10.113.3.10** `template<typename E> unsigned mln::doc::Image< E >::nsites ( ) const`  
`[inherited]`

Give the number of points of the image domain.

**10.113.3.11** `template<typename E> lvalue mln::doc::Image< E >::operator() ( const psite & p`  
`) [inherited]`

Read-write access to the image value located at *p*.

#### Parameters

[in] *p* A point site.

#### Precondition

The image has to own the site *p*.

#### Returns

The value at *p* (assignable).

**10.113.3.12** `template<typename E > rvalue mln::doc::Image< E >::operator() ( const psite & p ) const [inherited]`

Read-only access to the image value located at *p*.

#### Parameters

[in] *p* A point site.

#### Precondition

The image has to own the site *p*.

#### Returns

The value at *p* (not assignable).

**10.113.3.13** `template<typename E > rvalue mln::doc::Fastest_Image< E >::operator[] ( unsigned o ) const`

Read-only access to the image value at offset *o*.

#### Parameters

[in] *o* An offset.

#### Precondition

*o* < [nelements\(\)](#)

#### Returns

The value at *o* (not assignable).

**10.113.3.14** `template<typename E > lvalue mln::doc::Fastest_Image< E >::operator[] ( unsigned o )`

Read-write access to the image value at offset *o*.

#### Parameters

[in] *o* An offset.

#### Precondition

*o* < [nelements\(\)](#)

#### Returns

The value at *o* (assignable).

**10.113.3.15** `template<typename E > point mln::doc::Fastest_Image< E >::point_at_index ( unsigned o ) const`

Give the point at offset `o`.

#### Parameters

[in] `o` An offset.

#### Precondition

The image has to be initialized.  
`o < nelements()`

**10.113.3.16** `template<typename E > const vset& mln::doc::Image< E >::values ( ) const`  
**[inherited]**

Give the set of values of the image.

#### Returns

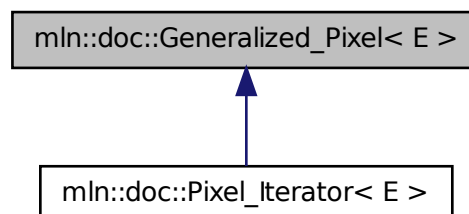
A reference to the value set.

## 10.114 mln::doc::Generalized\_Pixel< E > Struct Template Reference

Documentation class for `mln::Generalized_Pixel`.

```
#include <generalized_pixel.hh>
```

Inheritance diagram for `mln::doc::Generalized_Pixel< E >`:



### Public Types

- typedef void `image`  
*Image associated type (with possible const qualification).*

- typedef void [rvalue](#)  
*Read-only value associated type.*
- typedef void [value](#)  
*Value associated type.*

## Public Member Functions

- [image](#) & [ima](#) () const  
*Give the image of this generalized pixel.*
- [rvalue](#) [val](#) () const  
*Give the value of this generalized pixel.*

### 10.114.1 Detailed Description

`template<typename E> struct mln::doc::Generalized_Pixel< E >`

Documentation class for mln::Generalized\_Pixel.

#### See also

`mln::Generalized_Pixel`

Definition at line 45 of file doc/generalized\_pixel.hh.

### 10.114.2 Member Typedef Documentation

**10.114.2.1** `template<typename E > typedef void mln::doc::Generalized_Pixel< E >::image`

[Image](#) associated type (with possible const qualification).

Definition at line 49 of file doc/generalized\_pixel.hh.

**10.114.2.2** `template<typename E > typedef void mln::doc::Generalized_Pixel< E >::rvalue`

Read-only value associated type.

Definition at line 55 of file doc/generalized\_pixel.hh.

**10.114.2.3** `template<typename E > typedef void mln::doc::Generalized_Pixel< E >::value`

[Value](#) associated type.

Definition at line 52 of file doc/generalized\_pixel.hh.

### 10.114.3 Member Function Documentation

#### 10.114.3.1 `template<typename E> image& mln::doc::Generalized_Pixel< E >::ima ( ) const`

Give the image of this generalized pixel.

The constness of a pixel object is not transmitted to the underlying image.

#### 10.114.3.2 `template<typename E> rvalue mln::doc::Generalized_Pixel< E >::val ( ) const`

Give the value of this generalized pixel.

#### Returns

A read-only value.

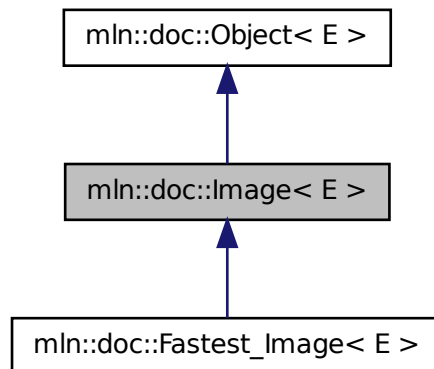
Reimplemented in [mln::doc::Pixel\\_Iterator< E >](#).

## 10.115 `mln::doc::Image< E >` Struct Template Reference

Documentation class for [mln::Image](#).

```
#include <image.hh>
```

Inheritance diagram for `mln::doc::Image< E >`:



### Public Types

- typedef void [bkd\\_piter](#)  
*Backward point iterator associated type.*
- typedef void [coord](#)

*Coordinate associated type.*

- typedef void [dpoint](#)  
*Dpsite associated type.*
- typedef void [fwd\\_piter](#)  
*Forward point iterator associated type.*
- typedef void [lvalue](#)  
*Type returned by the read-write pixel value operator.*
- typedef void [point](#)  
*Site associated type.*
- typedef void [pset](#)  
*Point set associated type.*
- typedef void [psite](#)  
*Point\_Site associated type.*
- typedef void [rvalue](#)  
*Type returned by the read pixel value operator.*
- typedef void [skeleton](#)  
*Associate type that describes how this type of image is constructed.*
- typedef void [value](#)  
*Value associated type.*
- typedef void [vset](#)  
*Value set associated type.*

## Public Member Functions

- const [box](#)< [point](#) > & [bbox](#) () const  
*Give a bounding box of the image domain.*
- const [pset](#) & [domain](#) () const  
*Give the definition domain of the image.*
- bool [has](#) (const [psite](#) &p) const  
*Test if the image owns the point site p.*
- bool [has](#) (const [psite](#) &p) const  
*Test if p belongs to the image domain.*
- bool [is\\_valid](#) () const  
*Test if the image have been initialized.*

- unsigned `nsites` () const  
*Give the number of points of the image domain.*
- `rvalue operator()` (const `psite` &p) const  
*Read-only access to the image value located at p.*
- `lvalue operator()` (const `psite` &p)  
*Read-write access to the image value located at p.*
- const `vset` & `values` () const  
*Give the set of values of the image.*

### 10.115.1 Detailed Description

`template<typename E> struct mln::doc::Image< E >`

Documentation class for `mln::Image`.

See also

[mln::Image](#)

Definition at line 36 of file `core/concept/doc/image.hh`.

### 10.115.2 Member Typedef Documentation

**10.115.2.1** `template<typename E > typedef void mln::doc::Image< E >::bkd_piter`

Backward point iterator associated type.

**Invariant**

This type has to derive from `mln::Site_Iterator`.

Definition at line 147 of file `core/concept/doc/image.hh`.

**10.115.2.2** `template<typename E > typedef void mln::doc::Image< E >::coord`

Coordinate associated type.

Definition at line 131 of file `core/concept/doc/image.hh`.

**10.115.2.3** `template<typename E > typedef void mln::doc::Image< E >::dpoint`

Dpsite associated type.

**Invariant**

This type has to derive from `mln::Dpoint`.

Definition at line 136 of file `core/concept/doc/image.hh`.



**10.115.2.4** `template<typename E > typedef void mln::doc::Image< E >::fwd_piter`

Forward point iterator associated type.

**Invariant**

This type has to derive from [mln::Site\\_Iterator](#).

Definition at line 142 of file core/concept/doc/image.hh.

**10.115.2.5** `template<typename E > typedef void mln::doc::Image< E >::lvalue`

Type returned by the read-write pixel value operator.

Definition at line 52 of file core/concept/doc/image.hh.

**10.115.2.6** `template<typename E > typedef void mln::doc::Image< E >::point`

[Site](#) associated type.

**Invariant**

This type has to derive from [mln::Point](#).

Definition at line 121 of file core/concept/doc/image.hh.

**10.115.2.7** `template<typename E > typedef void mln::doc::Image< E >::pset`

[Point](#) set associated type.

**Invariant**

This type has to derive from [mln::Site\\_Set](#).

Definition at line 116 of file core/concept/doc/image.hh.

**10.115.2.8** `template<typename E > typedef void mln::doc::Image< E >::psite`

[Point\\_Site](#) associated type.

**Invariant**

This type has to derive from [mln::Point\\_Site](#).

Definition at line 126 of file core/concept/doc/image.hh.

**10.115.2.9** `template<typename E > typedef void mln::doc::Image< E >::rvalue`

Type returned by the read pixel value operator.

Definition at line 48 of file core/concept/doc/image.hh.

**10.115.2.10** `template<typename E> typedef void mln::doc::Image< E >::skeleton`

Associate type that describes how this type of image is constructed.

Definition at line 64 of file `core/concept/doc/image.hh`.

**10.115.2.11** `template<typename E> typedef void mln::doc::Image< E >::value`

[Value](#) associated type.

**Invariant**

This type is neither qualified by `const`, nor by reference.

Definition at line 44 of file `core/concept/doc/image.hh`.

**10.115.2.12** `template<typename E> typedef void mln::doc::Image< E >::vset`

[Value](#) set associated type.

**Invariant**

This type has to derive from [mln::Value\\_Set](#).

Definition at line 57 of file `core/concept/doc/image.hh`.

**10.115.3 Member Function Documentation****10.115.3.1** `template<typename E> const box<point>& mln::doc::Image< E >::bbox ( ) const`

Give a bounding box of the image domain.

This bounding box may be larger than the smallest bounding box (the optimal one). Practically an image type is not obliged to update its bounding box so that it is always optimal.

**Returns**

A bounding box of the image domain.

**10.115.3.2** `template<typename E> const pset& mln::doc::Image< E >::domain ( ) const`

Give the definition domain of the image.

**Returns**

A reference to the domain point set.

**10.115.3.3** `template<typename E> bool mln::doc::Image< E >::has ( const psite & p ) const`

Test if the image owns the point site `p`.

**Returns**

True if accessing the image value at `p` is possible, that is, does not abort the execution.

**10.115.3.4** `template<typename E> bool mln::doc::Image< E >::has ( const psite & p ) const`

Test if *p* belongs to the image domain.

**Parameters**

[in] *p* A point site.

**Returns**

True if *p* belongs to the image domain.

**Invariant**

has(*p*) is true => has(*p*) is also true.

**10.115.3.5** `template<typename E> bool mln::doc::Image< E >::is_valid ( ) const`

Test if the image have been initialized.

**10.115.3.6** `template<typename E> unsigned mln::doc::Image< E >::nsites ( ) const`

Give the number of points of the image domain.

**10.115.3.7** `template<typename E> rvalue mln::doc::Image< E >::operator() ( const psite & p ) const`

Read-only access to the image value located at *p*.

**Parameters**

[in] *p* A point site.

**Precondition**

The image has to own the site *p*.

**Returns**

The value at *p* (not assignable).

**10.115.3.8** `template<typename E> lvalue mln::doc::Image< E >::operator() ( const psite & p )`

Read-write access to the image value located at *p*.

**Parameters**

[in] *p* A point site.

**Precondition**

The image has to own the site *p*.

**Returns**

The value at *p* (assignable).

### 10.115.3.9 `template<typename E> const vset& mln::doc::Image< E >::values ( ) const`

Give the set of values of the image.

#### Returns

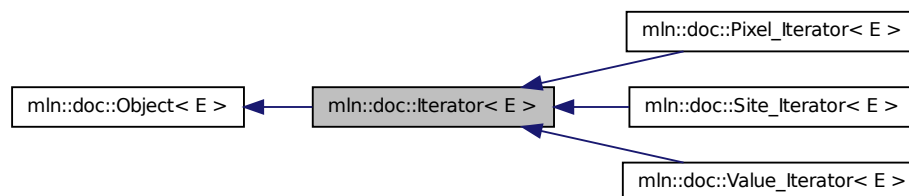
A reference to the value set.

## 10.116 `mln::doc::Iterator< E >` Struct Template Reference

Documentation class for [mln::Iterator](#).

```
#include <iterator.hh>
```

Inheritance diagram for `mln::doc::Iterator< E >`:



### Public Member Functions

- void `invalidate ( )`  
*Invalidate the iterator.*
- bool `is_valid ( ) const`  
*Returns true if the iterator is valid, that is, designates an element.*
- void `start ( )`  
*Start an iteration.*

### 10.116.1 Detailed Description

```
template<typename E> struct mln::doc::Iterator< E >
```

Documentation class for [mln::Iterator](#).

#### See also

[mln::Iterator](#)

Definition at line 36 of file `doc/iterator.hh`.

## 10.116.2 Member Function Documentation

### 10.116.2.1 `template<typename E> void mln::doc::Iterator< E >::invalidate ( )`

Invalidate the iterator.

### 10.116.2.2 `template<typename E> bool mln::doc::Iterator< E >::is_valid ( ) const`

Returns true if the iterator is valid, that is, designates an element.

### 10.116.2.3 `template<typename E> void mln::doc::Iterator< E >::start ( )`

Start an iteration.

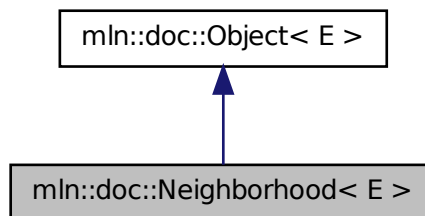
Make the iterator designate the first element if it exists. If this first element does not exist, the iterator is not valid.

## 10.117 mln::doc::Neighborhood< E > Struct Template Reference

Documentation class for [mln::Neighborhood](#).

```
#include <neighborhood.hh>
```

Inheritance diagram for mln::doc::Neighborhood< E >:



## Public Types

- typedef void [bkd\\_niter](#)  
*Site\_Iterator* type associated to this neighborhood to browse neighbors in a backward way.
- typedef void [dpoint](#)  
*Dpsite* associated type.
- typedef void [fwd\\_niter](#)  
*Site\_Iterator* type associated to this neighborhood to browse neighbors in a forward way.

- typedef void [niter](#)  
*Site\_Iterator* type associated to this neighborhood to browse neighbors.
- typedef void [point](#)  
*Site* associated type.

### 10.117.1 Detailed Description

**template<typename E> struct mln::doc::Neighborhood< E >**

Documentation class for [mln::Neighborhood](#).

See also

[mln::Neighborhood](#)

Definition at line 37 of file core/concept/doc/neighborhood.hh.

### 10.117.2 Member Typedef Documentation

**10.117.2.1 template<typename E > typedef void mln::doc::Neighborhood< E >::bkd\_niter**

[Site\\_Iterator](#) type associated to this neighborhood to browse neighbors in a backward way.

Definition at line 52 of file core/concept/doc/neighborhood.hh.

**10.117.2.2 template<typename E > typedef void mln::doc::Neighborhood< E >::dpoint**

Dpsite associated type.

Definition at line 55 of file core/concept/doc/neighborhood.hh.

**10.117.2.3 template<typename E > typedef void mln::doc::Neighborhood< E >::fwd\_niter**

[Site\\_Iterator](#) type associated to this neighborhood to browse neighbors in a forward way.

Definition at line 47 of file core/concept/doc/neighborhood.hh.

**10.117.2.4 template<typename E > typedef void mln::doc::Neighborhood< E >::niter**

[Site\\_Iterator](#) type associated to this neighborhood to browse neighbors.

Definition at line 42 of file core/concept/doc/neighborhood.hh.

**10.117.2.5 template<typename E > typedef void mln::doc::Neighborhood< E >::point**

[Site](#) associated type.

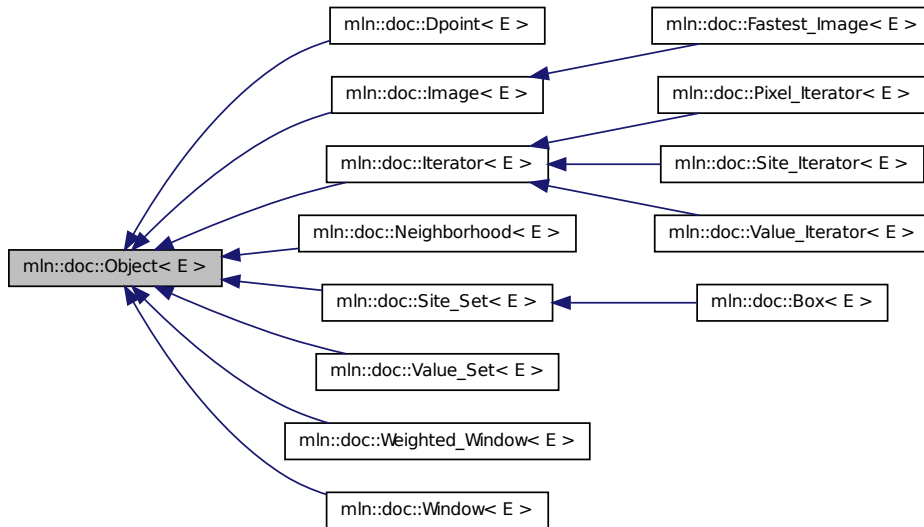
Definition at line 58 of file core/concept/doc/neighborhood.hh.

## 10.118 mln::doc::Object< E > Struct Template Reference

Documentation class for [mln::Object](#).

```
#include <object.hh>
```

Inheritance diagram for mln::doc::Object< E >:



### 10.118.1 Detailed Description

```
template<typename E> struct mln::doc::Object< E >
```

Documentation class for [mln::Object](#).

See also

[mln::Object](#)

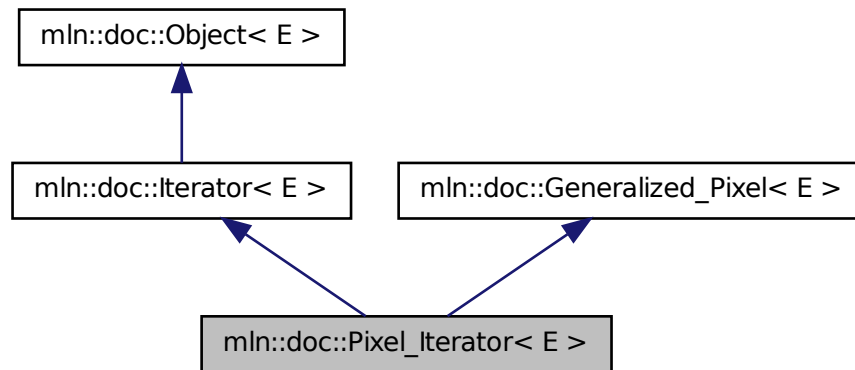
Definition at line 46 of file doc/object.hh.

## 10.119 mln::doc::Pixel\_Iterator< E > Struct Template Reference

Documentation class for [mln::Iterator](#).

```
#include <pixel_iterator.hh>
```

Inheritance diagram for `mln::doc::Pixel_Iterator< E >`:



## Public Types

- typedef void `image`  
*Image associated type (with possible const qualification).*
- typedef void `lvalue`  
*Type returned by the read-write dereference operator.*
- typedef void `rvalue`  
*Read-only value associated type.*
- typedef void `value`  
*Value associated type.*

## Public Member Functions

- `image & ima () const`  
*Give the image of this generalized pixel.*
- void `invalidate ()`  
*Invalidate the iterator.*
- bool `is_valid () const`  
*Returns true if the iterator is valid, that is, designates an element.*
- void `start ()`  
*Start an iteration.*



- [lvalue val \(\) const](#)  
*Give the pixel value.*

### 10.119.1 Detailed Description

**template<typename E> struct mln::doc::Pixel\_Iterator< E >**

Documentation class for [mln::Iterator](#).

See also

[mln::Pixel\\_Iterator](#)

Definition at line 36 of file doc/pixel\_iterator.hh.

### 10.119.2 Member Typedef Documentation

**10.119.2.1 template<typename E > typedef void mln::doc::Generalized\_Pixel< E >::image  
[inherited]**

[Image](#) associated type (with possible const qualification).

Definition at line 49 of file doc/generalized\_pixel.hh.

**10.119.2.2 template<typename E > typedef void mln::doc::Pixel\_Iterator< E >::lvalue**

Type returned by the read-write dereference operator.

Definition at line 41 of file doc/pixel\_iterator.hh.

**10.119.2.3 template<typename E > typedef void mln::doc::Generalized\_Pixel< E >::rvalue  
[inherited]**

Read-only value associated type.

Definition at line 55 of file doc/generalized\_pixel.hh.

**10.119.2.4 template<typename E > typedef void mln::doc::Generalized\_Pixel< E >::value  
[inherited]**

[Value](#) associated type.

Definition at line 52 of file doc/generalized\_pixel.hh.

### 10.119.3 Member Function Documentation

**10.119.3.1 template<typename E > image& mln::doc::Generalized\_Pixel< E >::ima ( ) const  
[inherited]**

Give the image of this generalized pixel.

The constness of a pixel object is not transmitted to the underlying image.

**10.119.3.2** `template<typename E > void mln::doc::Iterator< E >::invalidate ( )`  
**[inherited]**

Invalidate the iterator.

**10.119.3.3** `template<typename E > bool mln::doc::Iterator< E >::is_valid ( ) const`  
**[inherited]**

Returns true if the iterator is valid, that is, designates an element.

**10.119.3.4** `template<typename E > void mln::doc::Iterator< E >::start ( )` **[inherited]**

Start an iteration.

Make the iterator designate the first element if it exists. If this first element does not exist, the iterator is not valid.

**10.119.3.5** `template<typename E > lvalue mln::doc::Pixel_Iterator< E >::val ( ) const`

Give the pixel value.

#### Returns

The current pixel value; this value cannot be modified.

Reimplemented from [mln::doc::Generalized\\_Pixel< E >](#).

## 10.120 mln::doc::Point\_Site< E > Struct Template Reference

Documentation class for [mln::Point\\_Site](#).

```
#include <point_site.hh>
```

### Public Types

- enum { [dim](#) }
- typedef void [coord](#)
- typedef void [dpoint](#)  
*Dpsite associated type.*
- typedef void [mesh](#)  
*Mesh associated type.*
- typedef void [point](#)  
*Site associated type.*

## Public Member Functions

- `coord operator[]` (unsigned i) const  
*Read-only access to the  $i$ -th coordinate value.*
- `const point & to_point` () const  
*Give a reference to the corresponding point.*

### 10.120.1 Detailed Description

`template<typename E> struct mln::doc::Point_Site< E >`

Documentation class for `mln::Point_Site`.

#### See also

[mln::Point\\_Site](#)

Definition at line 37 of file `doc/point_site.hh`.

### 10.120.2 Member Typedef Documentation

**10.120.2.1** `template<typename E > typedef void mln::doc::Point_Site< E >::coord`

Coordinate associated type.

Definition at line 62 of file `doc/point_site.hh`.

**10.120.2.2** `template<typename E > typedef void mln::doc::Point_Site< E >::dpoint`

Dpsite associated type.

#### Invariant

This type has to derive from [mln::Dpoint](#).

Definition at line 58 of file `doc/point_site.hh`.

**10.120.2.3** `template<typename E > typedef void mln::doc::Point_Site< E >::mesh`

[Mesh](#) associated type.

#### Invariant

This type has to derive from [mln::Mesh](#).

Definition at line 48 of file `doc/point_site.hh`.

#### 10.120.2.4 `template<typename E > typedef void mln::doc::Point_Site< E >::point`

[Site](#) associated type.

##### Invariant

This type has to derive from [mln::Point](#).

Definition at line 53 of file doc/point\_site.hh.

### 10.120.3 Member Enumeration Documentation

#### 10.120.3.1 `template<typename E > anonymous enum`

##### Enumerator:

*dim* Dimension of the space.

##### Invariant

$dim > 0$

Definition at line 43 of file doc/point\_site.hh.

### 10.120.4 Member Function Documentation

#### 10.120.4.1 `template<typename E > coord mln::doc::Point_Site< E >::operator[] ( unsigned i ) const`

Read-only access to the *i*-th coordinate value.

##### Parameters

[in] *i* The coordinate index.

##### Precondition

$i < dim$

##### Returns

The value of the *i*-th coordinate.

#### 10.120.4.2 `template<typename E > const point& mln::doc::Point_Site< E >::to_point ( ) const`

Give a reference to the corresponding point.

This method allows for iterators to refer to a point.

##### Returns

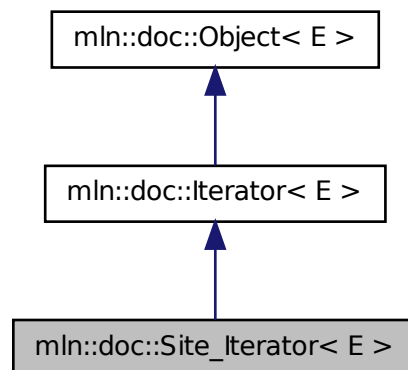
A point constant reference.

## 10.121 mln::doc::Site\_Iterator< E > Struct Template Reference

Documentation class for [mln::Site\\_Iterator](#).

```
#include <point_iterator.hh>
```

Inheritance diagram for mln::doc::Site\_Iterator< E >:



### Public Types

- typedef void [psite](#)  
*Point\_Site associated type.*

### Public Member Functions

- void [invalidate](#) ()  
*Invalidate the iterator.*
- bool [is\\_valid](#) () const  
*Returns true if the iterator is valid, that is, designates an element.*
- [operator psite](#) () const  
*Conversion into a point-site.*
- void [start](#) ()  
*Start an iteration.*

### 10.121.1 Detailed Description

**template<typename E> struct mln::doc::Site\_Iterator< E >**

Documentation class for [mln::Site\\_Iterator](#).

See also

[mln::Site\\_Iterator](#)

Definition at line 37 of file point\_iterator.hh.

### 10.121.2 Member Typedef Documentation

**10.121.2.1 template<typename E > typedef void mln::doc::Site\_Iterator< E >::psite**

[Point\\_Site](#) associated type.

**Invariant**

This type has to derive from [mln::Point\\_Site](#).

Definition at line 43 of file point\_iterator.hh.

### 10.121.3 Member Function Documentation

**10.121.3.1 template<typename E > void mln::doc::Iterator< E >::invalidate ( )**  
**[inherited]**

Invalidate the iterator.

**10.121.3.2 template<typename E > bool mln::doc::Iterator< E >::is\_valid ( ) const**  
**[inherited]**

Returns true if the iterator is valid, that is, designates an element.

**10.121.3.3 template<typename E > mln::doc::Site\_Iterator< E >::operator psite ( ) const**

Conversion into a point-site.

**Returns**

A point site.

**10.121.3.4 template<typename E > void mln::doc::Iterator< E >::start ( ) [inherited]**

Start an iteration.

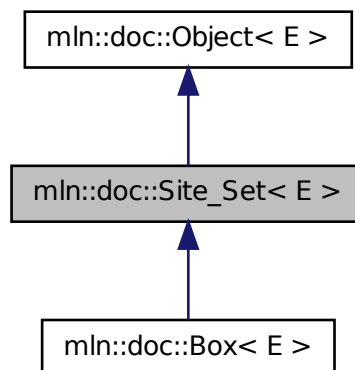
Make the iterator designate the first element if it exists. If this first element does not exist, the iterator is not valid.

## 10.122 mln::doc::Site\_Set< E > Struct Template Reference

Documentation class for [mln::Site\\_Set](#).

```
#include <site_set.hh>
```

Inheritance diagram for mln::doc::Site\_Set< E >:



### Public Types

- typedef void [bkd\\_piter](#)  
*Backward [Site\\_Iterator](#) associated type.*
- typedef void [fwd\\_piter](#)  
*Forward [Site\\_Iterator](#) associated type.*
- typedef void [psite](#)  
*PSite associated type.*
- typedef void [site](#)  
*Site associated type.*

### Public Member Functions

- bool [has](#) (const [psite](#) &p) const  
*Test if *p* belongs to this site set.*

### 10.122.1 Detailed Description

`template<typename E> struct mln::doc::Site_Set< E >`

Documentation class for [mln::Site\\_Set](#).

See also

[mln::Site\\_Set](#)

Definition at line 37 of file `mln/core/concept/doc/site_set.hh`.

### 10.122.2 Member Typedef Documentation

**10.122.2.1** `template<typename E > typedef void mln::doc::Site_Set< E >::bkd_piter`

Backward [Site\\_Iterator](#) associated type.

Definition at line 53 of file `mln/core/concept/doc/site_set.hh`.

**10.122.2.2** `template<typename E > typedef void mln::doc::Site_Set< E >::fwd_piter`

Forward [Site\\_Iterator](#) associated type.

Definition at line 49 of file `mln/core/concept/doc/site_set.hh`.

**10.122.2.3** `template<typename E > typedef void mln::doc::Site_Set< E >::psite`

PSite associated type.

Definition at line 45 of file `mln/core/concept/doc/site_set.hh`.

**10.122.2.4** `template<typename E > typedef void mln::doc::Site_Set< E >::site`

[Site](#) associated type.

Definition at line 41 of file `mln/core/concept/doc/site_set.hh`.

### 10.122.3 Member Function Documentation

**10.122.3.1** `template<typename E > bool mln::doc::Site_Set< E >::has ( const psite & p ) const`

Test if `p` belongs to this site set.

**Parameters**

[in] `p` A psite.

**Returns**

True if `p` is an element of the site set.

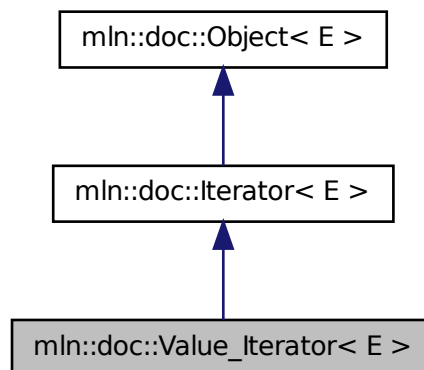


## 10.123 mln::doc::Value\_Iterator< E > Struct Template Reference

Documentation class for [mln::Value\\_Iterator](#).

```
#include <value_iterator.hh>
```

Inheritance diagram for mln::doc::Value\_Iterator< E >:



### Public Types

- typedef void [value](#)  
*Value associated type.*

### Public Member Functions

- void [invalidate](#) ()  
*Invalidate the iterator.*
- bool [is\\_valid](#) () const  
*Returns true if the iterator is valid, that is, designates an element.*
- [operator value](#) () const  
*Conversion into a value.*
- void [start](#) ()  
*Start an iteration.*

### 10.123.1 Detailed Description

**template<typename E> struct mln::doc::Value\_Iterator< E >**

Documentation class for [mln::Value\\_Iterator](#).

See also

[mln::Value\\_Iterator](#)

Definition at line 37 of file doc/value\_iterator.hh.

### 10.123.2 Member Typedef Documentation

**10.123.2.1 template<typename E > typedef void mln::doc::Value\_Iterator< E >::value**

[Value](#) associated type.

Definition at line 41 of file doc/value\_iterator.hh.

### 10.123.3 Member Function Documentation

**10.123.3.1 template<typename E > void mln::doc::Iterator< E >::invalidate ( )  
[inherited]**

Invalidate the iterator.

**10.123.3.2 template<typename E > bool mln::doc::Iterator< E >::is\_valid ( ) const  
[inherited]**

Returns true if the iterator is valid, that is, designates an element.

**10.123.3.3 template<typename E > mln::doc::Value\_Iterator< E >::operator value ( ) const**

Conversion into a value.

**Returns**

A value.

**10.123.3.4 template<typename E > void mln::doc::Iterator< E >::start ( ) [inherited]**

Start an iteration.

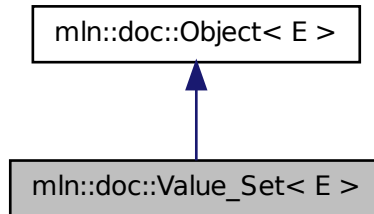
Make the iterator designate the first element if it exists. If this first element does not exist, the iterator is not valid.

## 10.124 mln::doc::Value\_Set< E > Struct Template Reference

Documentation class for [mln::Value\\_Set](#).

```
#include <value_set.hh>
```

Inheritance diagram for mln::doc::Value\_Set< E >:



## Public Types

- typedef void [bkd\\_viter](#)  
*Backward [Value\\_Iterator](#) associated type.*
- typedef void [fwd\\_viter](#)  
*Forward [Value\\_Iterator](#) associated type.*
- typedef void [value](#)  
*[Value](#) associated type.*

## Public Member Functions

- bool [has](#) (const [value](#) &v) const  
*Test if v belongs to this set of values.*
- unsigned [index\\_of](#) (const [value](#) &v) const  
*Give the index of value v in this set.*
- unsigned [nvalues](#) () const  
*Give the number of values in this set.*
- [value operator\[\]](#) (unsigned i) const  
*Give the i-th value of this set.*

### 10.124.1 Detailed Description

```
template<typename E> struct mln::doc::Value_Set< E >
```

Documentation class for [mln::Value\\_Set](#).

See also

[mln::Value\\_Set](#)

Definition at line 37 of file doc/value\_set.hh.

## 10.124.2 Member Typedef Documentation

### 10.124.2.1 `template<typename E> typedef void mln::doc::Value_Set< E >::bkd_viter`

Backward [Value\\_Iterator](#) associated type.

Definition at line 49 of file doc/value\_set.hh.

### 10.124.2.2 `template<typename E> typedef void mln::doc::Value_Set< E >::fwd_viter`

Forward [Value\\_Iterator](#) associated type.

Definition at line 45 of file doc/value\_set.hh.

### 10.124.2.3 `template<typename E> typedef void mln::doc::Value_Set< E >::value`

[Value](#) associated type.

Definition at line 41 of file doc/value\_set.hh.

## 10.124.3 Member Function Documentation

### 10.124.3.1 `template<typename E> bool mln::doc::Value_Set< E >::has ( const value & v ) const`

Test if  $v$  belongs to this set of values.

#### Parameters

[in]  $v$  A value.

#### Returns

True if  $v$  is an element of the set of values.

### 10.124.3.2 `template<typename E> unsigned mln::doc::Value_Set< E >::index_of ( const value & v ) const`

Give the index of value  $v$  in this set.

### 10.124.3.3 `template<typename E> unsigned mln::doc::Value_Set< E >::nvalues ( ) const`

Give the number of values in this set.

#### 10.124.3.4 `template<typename E> value mln::doc::Value_Set< E >::operator[] ( unsigned i ) const`

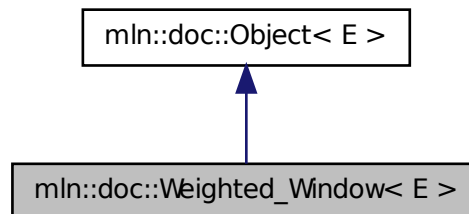
Give the `i`-th value of this set.

### 10.125 `mln::doc::Weighted_Window< E > Struct Template Reference`

Documentation class for [mln::Weighted\\_Window](#).

```
#include <weighted_window.hh>
```

Inheritance diagram for `mln::doc::Weighted_Window< E >`:



#### Public Types

- typedef void [bkd\\_qiter](#)  
*Site\_Iterator* type associated to this *weighted\_window* to browse its points in a backward way.
- typedef void [dpoint](#)  
*Dpsite* associated type.
- typedef void [fwd\\_qiter](#)  
*Site\_Iterator* type associated to this *weighted\_window* to browse its points in a forward way.
- typedef void [point](#)  
*Site* associated type.
- typedef void [weight](#)  
*Weight* associated type.
- typedef void [window](#)  
*Window* associated type.

## Public Member Functions

- unsigned `delta` () const  
*Give the maximum coordinate gap between the window center and a window point.*
- bool `is_centered` () const  
*Test if the `weighted_window` is centered.*
- bool `is_empty` () const  
*Test if the `weighted window` is empty.*
- E & `sym` ()  
*Apply a central symmetry to the target `weighted window`.*
- const `window` & `win` () const  
*Give the corresponding window.*

### 10.125.1 Detailed Description

`template<typename E> struct mln::doc::Weighted_Window< E >`

Documentation class for `mln::Weighted_Window`. A `weighted_window` is the definition of a set of points located around a central point, with a weight associated to each point.

#### See also

[mln::Weighted\\_Window](#)

Definition at line 40 of file `doc/weighted_window.hh`.

### 10.125.2 Member Typedef Documentation

**10.125.2.1** `template<typename E > typedef void mln::doc::Weighted_Window< E >::bkd_qiter`

[Site\\_Iterator](#) type associated to this `weighted_window` to browse its points in a backward way.

Definition at line 51 of file `doc/weighted_window.hh`.

**10.125.2.2** `template<typename E > typedef void mln::doc::Weighted_Window< E >::dpoint`

Dpsite associated type.

Definition at line 57 of file `doc/weighted_window.hh`.

**10.125.2.3** `template<typename E > typedef void mln::doc::Weighted_Window< E >::fwd_qiter`

[Site\\_Iterator](#) type associated to this `weighted_window` to browse its points in a forward way.

Definition at line 46 of file `doc/weighted_window.hh`.

**10.125.2.4** `template<typename E > typedef void mln::doc::Weighted_Window< E >::point`

[Site](#) associated type.

Definition at line 54 of file doc/weighted\_window.hh.

**10.125.2.5** `template<typename E > typedef void mln::doc::Weighted_Window< E >::weight`

Weight associated type.

Definition at line 60 of file doc/weighted\_window.hh.

**10.125.2.6** `template<typename E > typedef void mln::doc::Weighted_Window< E >::window`

[Window](#) associated type.

Definition at line 63 of file doc/weighted\_window.hh.

**10.125.3 Member Function Documentation****10.125.3.1** `template<typename E > unsigned mln::doc::Weighted_Window< E >::delta ( ) const`

Give the maximum coordinate gap between the window center and a window point.

**10.125.3.2** `template<typename E > bool mln::doc::Weighted_Window< E >::is_centered ( ) const`

Test if the `weighted_window` is centered.

A weighted window is centered is the origin belongs to it.

**10.125.3.3** `template<typename E > bool mln::doc::Weighted_Window< E >::is_empty ( ) const`

Test if the weighted window is empty.

A `weighted_window` of null size is empty.

**10.125.3.4** `template<typename E > E& mln::doc::Weighted_Window< E >::sym ( )`

Apply a central symmetry to the target weighted window.

**10.125.3.5** `template<typename E > const window& mln::doc::Weighted_Window< E >::win ( ) const`

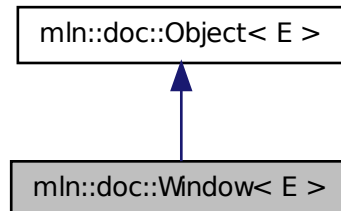
Give the corresponding window.

**10.126 mln::doc::Window< E > Struct Template Reference**

Documentation class for [mln::Window](#).

```
#include <window.hh>
```

Inheritance diagram for `mln::doc::Window< E >`:



## Public Types

- typedef void [bkd\\_qiter](#)  
*Site\_Iterator* type associated to this window to browse its points in a backward way.
- typedef void [fwd\\_qiter](#)  
*Site\_Iterator* type associated to this window to browse its points in a forward way.
- typedef void [qiter](#)  
*Site\_Iterator* type associated to this window to browse its points.

### 10.126.1 Detailed Description

```
template<typename E> struct mln::doc::Window< E >
```

Documentation class for [mln::Window](#). A window is the definition of a set of points located around a central point.

#### See also

[mln::Window](#)

Definition at line 40 of file `concept/doc/window.hh`.

### 10.126.2 Member Typedef Documentation

**10.126.2.1** `template<typename E > typedef void mln::doc::Window< E >::bkd_qiter`

[Site\\_Iterator](#) type associated to this window to browse its points in a backward way.

Definition at line 55 of file `concept/doc/window.hh`.



**10.126.2.2** `template<typename E > typedef void mln::doc::Window< E >::fwd_qiter`

[Site\\_Iterator](#) type associated to this window to browse its points in a forward way.

Definition at line 50 of file concept/doc/window.hh.

**10.126.2.3** `template<typename E > typedef void mln::doc::Window< E >::qiter`

[Site\\_Iterator](#) type associated to this window to browse its points.

Definition at line 45 of file concept/doc/window.hh.

**10.127 mln::Dpoint< E > Struct Template Reference**

Base class for implementation of delta-point classes.

```
#include <dpoint.hh>
```

Inherits mln::Delta\_Point\_Site< E >.

**Public Member Functions**

- `const E & to_dpoint () const`

*It is a [Dpoint](#) so it returns itself.*

**10.127.1 Detailed Description**

```
template<typename E> struct mln::Dpoint< E >
```

Base class for implementation of delta-point classes. A delta-point is a vector defined by a couple of points.

Given two points, A and B, the vector AB is mapped into the delta-point  $D = AB$ . Practically one can write:  $D = B - A$ .

See also

[mln::doc::Dpoint](#) for a complete documentation of this class contents.

Definition at line 63 of file concept/dpoint.hh.

**10.127.2 Member Function Documentation****10.127.2.1** `template<typename E > const E & mln::Dpoint< E >::to_dpoint ( ) const`  
[inline]

It is a [Dpoint](#) so it returns itself.

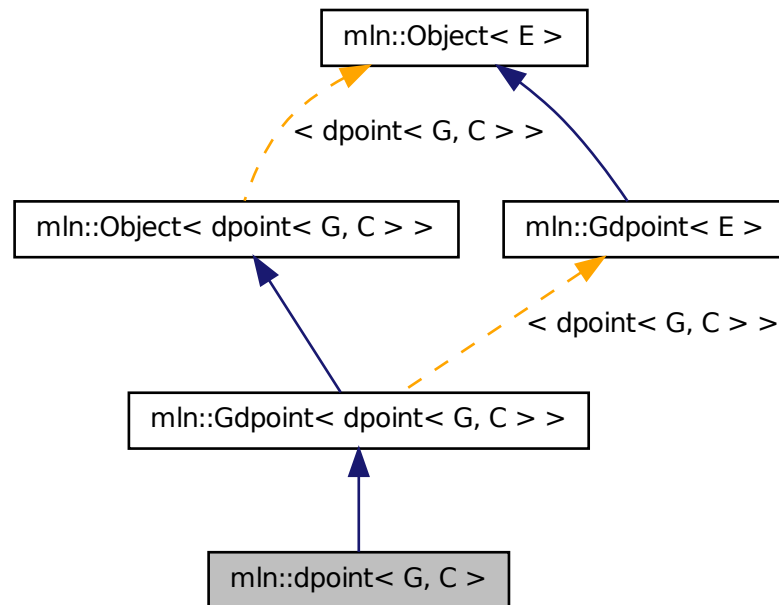
Definition at line 88 of file concept/dpoint.hh.

## 10.128 mln::dpoint< G, C > Struct Template Reference

Generic delta-point class.

```
#include <dpoint.hh>
```

Inheritance diagram for mln::dpoint< G, C >:



### Public Types

- enum { `dim` = `G::dim` }
- typedef `C` `coord`  
*Coordinate associated type.*
- typedef `G` `grid`  
*Grid associated type.*
- typedef `point< G, C >` `psite`  
*Psite associated type.*
- typedef `point< G, C >` `site`  
*Site associated type.*
- typedef `algebra::vec< G::dim, C >` `vec`  
*Algebra vector (vec) associated type.*

## Public Member Functions

- `dpoint ()`  
*Constructor without argument.*
- `template<typename C2 >`  
`dpoint (const algebra::vec< dim, C2 > &v)`  
*Constructor from an algebra vector.*
- `template<typename F >`  
`dpoint (const Function_v2v< F > &f)`  
*Constructor; coordinates are set by function  $f$ .*
- `template<typename Q >`  
`operator mln::algebra::vec< dpoint< G, C >::dim, Q > () const`  
*Conversion towards a algebra::vec.*
- `C operator[] (unsigned i) const`  
*Read-only access to the  $i$ -th coordinate value.*
- `C & operator[] (unsigned i)`  
*Read-write access to the  $i$ -th coordinate value.*
- `void set_all (C c)`  
*Set all coordinates to the value  $c$ .*
- `vec to_vec () const`  
*Explicit conversion.*
- `dpoint (C ind)`
- `dpoint (const literal::zero_t &)`  
*Constructors/assignments with literals.*

### 10.128.1 Detailed Description

`template<typename G, typename C> struct mln::dpoint< G, C >`

Generic delta-point class. Parameters are  $G$  the dimension of the space and  $C$  the coordinate type in this space.

Definition at line 58 of file `dpoint.hh`.

### 10.128.2 Member Typedef Documentation

**10.128.2.1** `template<typename G, typename C> typedef C mln::dpoint< G, C >::coord`

Coordinate associated type.

Definition at line 76 of file `dpoint.hh`.

**10.128.2.2** `template<typename G, typename C> typedef G mln::dpoint< G, C >::grid`

Grid associated type.

Definition at line 67 of file dpoint.hh.

**10.128.2.3** `template<typename G, typename C> typedef point<G,C> mln::dpoint< G, C >::psite`

Psite associated type.

Definition at line 70 of file dpoint.hh.

**10.128.2.4** `template<typename G, typename C> typedef point<G,C> mln::dpoint< G, C >::site`

[Site](#) associated type.

Definition at line 73 of file dpoint.hh.

**10.128.2.5** `template<typename G, typename C> typedef algebra::vec<G::dim, C> mln::dpoint< G, C >::vec`

Algebra vector (vec) associated type.

Definition at line 79 of file dpoint.hh.

**10.128.3 Member Enumeration Documentation****10.128.3.1** `template<typename G, typename C> anonymous enum`

**Enumerator:**

*dim* Dimension of the space.

**Invariant**

`dim > 0`

Definition at line 64 of file dpoint.hh.

**10.128.4 Constructor & Destructor Documentation****10.128.4.1** `template<typename G , typename C > mln::dpoint< G, C >::dpoint ( ) [inline]`

Constructor without argument.

Definition at line 152 of file dpoint.hh.

**10.128.4.2** `template<typename G , typename C > template<typename C2 > mln::dpoint< G, C >::dpoint ( const algebra::vec< dim, C2 > & v ) [inline]`

Constructor from an algebra vector.

Definition at line 159 of file dpoint.hh.

References `mln::dpoint< G, C >::dim`.

**10.128.4.3** `template<typename G , typename C> mln::dpoint< G, C >::dpoint ( C ind ) [inline]`

Constructors with different numbers of arguments (coordinates) w.r.t. the dimension.

Definition at line 176 of file dpoint.hh.

**10.128.4.4** `template<typename G , typename C> mln::dpoint< G, C >::dpoint ( const literal::zero_t & ) [inline]`

Constructors/assignments with literals.

Definition at line 203 of file dpoint.hh.

**10.128.4.5** `template<typename G , typename C > template<typename F > mln::dpoint< G, C >::dpoint ( const Function_v2v< F > & f ) [inline]`

Constructor; coordinates are set by function *f*.

Definition at line 238 of file dpoint.hh.

**10.128.5 Member Function Documentation****10.128.5.1** `template<typename G , typename C > template<typename Q > mln::dpoint< G, C >::operator mln::algebra::vec< dpoint< G, C >::dim, Q > ( ) const [inline]`

Conversion towards a algebra::vec.

Definition at line 257 of file dpoint.hh.

References mln::dpoint< G, C >::to\_vec().

**10.128.5.2** `template<typename G , typename C > C & mln::dpoint< G, C >::operator[] ( unsigned i ) [inline]`

Read-write access to the *i*-th coordinate value.

**Parameters**

[in] *i* The coordinate index.

**Precondition**

$i < \text{dim}$

Definition at line 144 of file dpoint.hh.

References mln::dpoint< G, C >::dim.

**10.128.5.3** `template<typename G , typename C > C mln::dpoint< G, C >::operator[] ( unsigned i ) const [inline]`

Read-only access to the *i*-th coordinate value.

**Parameters**

[in] *i* The coordinate index.

**Precondition**

$i < \text{dim}$

Definition at line 136 of file dpoint.hh.

References `mln::dpoint< G, C >::dim`.

#### 10.128.5.4 `template<typename G , typename C> void mln::dpoint< G, C >::set_all ( C c )` **[inline]**

Set all coordinates to the value `c`.

Definition at line 248 of file dpoint.hh.

Referenced by `mln::win::line< M, i, C >::line()`.

#### 10.128.5.5 `template<typename G , typename C > dpoint< G, C >::vec mln::dpoint< G, C >::to_vec ( ) const` **[inline]**

Explicit conversion.

Definition at line 265 of file dpoint.hh.

References `mln::dpoint< G, C >::dim`.

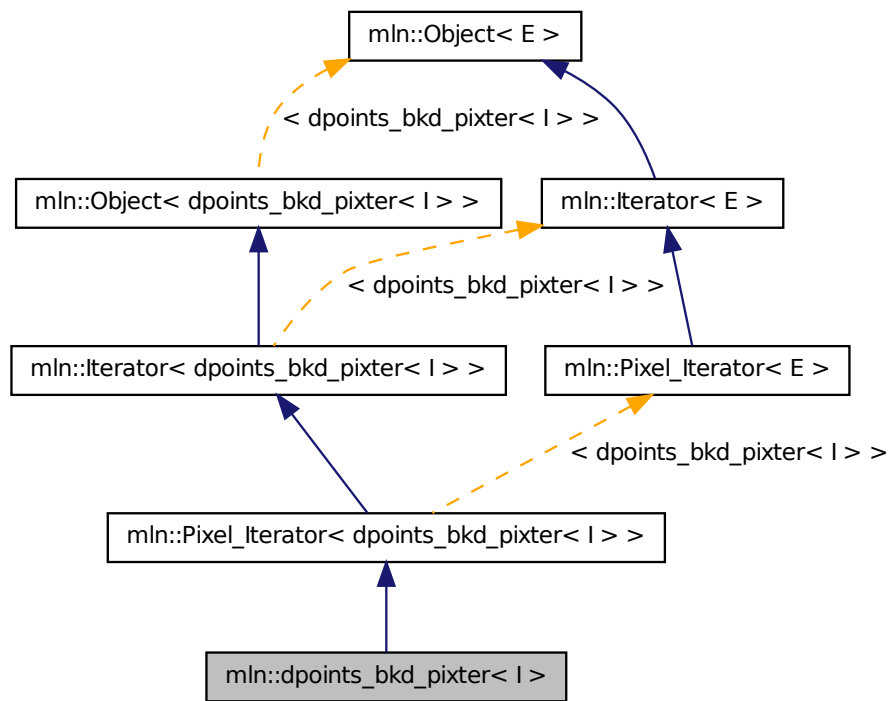
Referenced by `mln::dpoint< G, C >::operator mln::algebra::vec< dpoint< G, C >::dim, Q >()`.

## 10.129 `mln::dpoints_bkd_pixter< I >` Class Template Reference

A generic backward iterator on the pixels of a dpoint-based window or neighborhood.

```
#include <dpoints_pixter.hh>
```

Inheritance diagram for mln::dpoints\_bkd\_pixter< I >:



## Public Member Functions

- `const I::value & center_val () const`  
*The value around which this iterator moves.*
- `template<typename Dps , typename Pref >`  
`dpoints_bkd_pixter (const Generalized_Pixel< Pref > &pxl_ref, const Dps &dps)`  
*Constructor (using a generalized pixel).*
- `template<typename Dps , typename Pref >`  
`dpoints_bkd_pixter (I &image, const Dps &dps, const Pref &p_ref)`  
*Constructor (using an image).*
- `void next ()`  
*Go to the next element.*
- `void start ()`  
*Manipulation.*

- void `invalidate ()`  
*Invalidate the iterator.*
- bool `is_valid () const`  
*Test the iterator validity.*
- void `update ()`  
*Force this iterator to update its location to take into account that its center point may have moved.*

### 10.129.1 Detailed Description

`template<typename I> class mln::dpoints_bkd_pixter< I >`

A generic backward iterator on the pixels of a dpoint-based window or neighborhood. Parameter `I` is the image type.

Definition at line 140 of file `dpoints_pixter.hh`.

### 10.129.2 Constructor & Destructor Documentation

**10.129.2.1** `template<typename I > template<typename Dps , typename Pref > mln::dpoints_bkd_pixter< I >::dpoints_bkd_pixter ( I & image, const Dps & dps, const Pref & p_ref ) [inline]`

Constructor (using an image).

#### Parameters

- `[in]` *image* The image to iterate over.
- `[in]` *dps* An object (neighborhood or window) that can provide a set of delta-points.
- `[in]` *p\_ref* Center (resp. reference) point of the neighborhood (resp. window).

Definition at line 338 of file `dpoints_pixter.hh`.

**10.129.2.2** `template<typename I > template<typename Dps , typename Pref > mln::dpoints_bkd_pixter< I >::dpoints_bkd_pixter ( const Generalized_Pixel< Pref > & pxl_ref, const Dps & dps ) [inline]`

Constructor (using a generalized pixel).

#### Parameters

- `[in]` *pxl\_ref* Center (generalized) pixel to iterate around.
- `[in]` *dps* An object (neighborhood or window) that can provide a set of delta-points.

Definition at line 352 of file `dpoints_pixter.hh`.



### 10.129.3 Member Function Documentation

**10.129.3.1** `template<typename I> const I::value & mln::dpoints_bkd_pixter< I >::center_val ( ) const [inline]`

The value around which this iterator moves.

Definition at line 367 of file dpoints\_pixter.hh.

**10.129.3.2** `template<typename I> void mln::dpoints_bkd_pixter< I >::invalidate ( ) [inline]`

Invalidate the iterator.

Definition at line 436 of file dpoints\_pixter.hh.

**10.129.3.3** `template<typename I> bool mln::dpoints_bkd_pixter< I >::is_valid ( ) const [inline]`

Test the iterator validity.

Definition at line 428 of file dpoints\_pixter.hh.

Referenced by mln::dpoints\_bkd\_pixter< I >::update().

**10.129.3.4** `void mln::Iterator< dpoints_bkd_pixter< I > >::next ( ) [inherited]`

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

#### Precondition

The iterator is valid.

**10.129.3.5** `template<typename I> void mln::dpoints_bkd_pixter< I >::start ( ) [inline]`

Manipulation.

Start an iteration.

Definition at line 409 of file dpoints\_pixter.hh.

References mln::dpoints\_bkd\_pixter< I >::update().

**10.129.3.6** `template<typename I> void mln::dpoints_bkd_pixter< I >::update ( ) [inline]`

Force this iterator to update its location to take into account that its center point may have moved.

Definition at line 395 of file dpoints\_pixter.hh.

References mln::dpoints\_bkd\_pixter< I >::is\_valid().

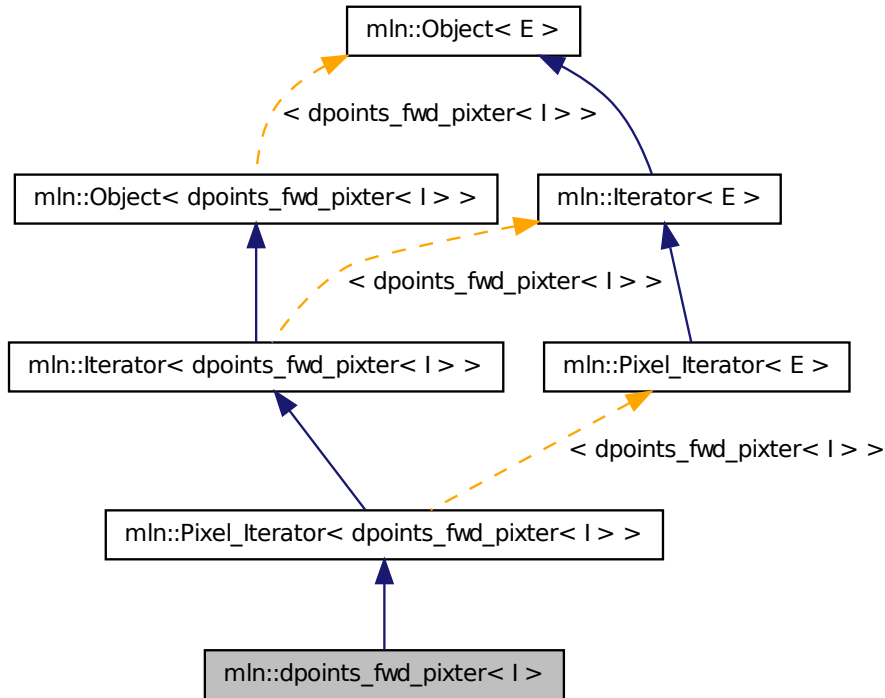
Referenced by mln::dpoints\_bkd\_pixter< I >::start().

## 10.130 mln::dpoints\_fwd\_pixter< I > Class Template Reference

A generic forward iterator on the pixels of a dpoint-based window or neighborhood.

```
#include <dpoints_pixter.hh>
```

Inheritance diagram for mln::dpoints\_fwd\_pixter< I >:



### Public Member Functions

- `const I::value & center_val () const`  
*The value around which this iterator moves.*
- `template<typename Dps , typename Pref >`  
`dpoints_fwd_pixter (const Generalized_Pixel< Pref > &pxl_ref, const Dps &dps)`  
*Constructor (using a generalized pixel).*
- `template<typename Dps , typename Pref >`  
`dpoints_fwd_pixter (I &image, const Dps &dps, const Pref &p_ref)`  
*Constructor (using an image).*
- `void next ()`  
*Go to the next element.*

- void [start](#) ()  
*Manipulation.*
- void [invalidate](#) ()  
*Invalidate the iterator.*
- bool [is\\_valid](#) () const  
*Test the iterator validity.*
- void [update](#) ()  
*Force this iterator to update its location to take into account that its center point may have moved.*

### 10.130.1 Detailed Description

**template<typename I> class mln::dpoints\_fwd\_pixter< I >**

A generic forward iterator on the pixels of a dpoint-based window or neighborhood. Parameter `I` is the image type.

Definition at line 57 of file `dpoints_pixter.hh`.

### 10.130.2 Constructor & Destructor Documentation

**10.130.2.1** `template<typename I > template<typename Dps , typename Pref > mln::dpoints_fwd_pixter< I >::dpoints_fwd_pixter ( I & image, const Dps & dps, const Pref & p_ref ) [inline]`

Constructor (using an image).

#### Parameters

- [in] *image* The image to iterate over.
- [in] *dps* An object (neighborhood or window) that can provide a set of delta-points.
- [in] *p\_ref* Center (resp. reference) point of the neighborhood (resp. window).

Definition at line 224 of file `dpoints_pixter.hh`.

**10.130.2.2** `template<typename I > template<typename Dps , typename Pref > mln::dpoints_fwd_pixter< I >::dpoints_fwd_pixter ( const Generalized_Pixel< Pref > & pxl_ref, const Dps & dps ) [inline]`

Constructor (using a generalized pixel).

#### Parameters

- [in] *pxl\_ref* Center (generalized) pixel to iterate around.
- [in] *dps* An object (neighborhood or window) that can provide a set of delta-points.

Definition at line 241 of file `dpoints_pixter.hh`.

### 10.130.3 Member Function Documentation

**10.130.3.1** `template<typename I> const I::value & mln::dpoints_fwd_pixter< I >::center_val ( ) const [inline]`

The value around which this iterator moves.

Definition at line 256 of file `dpoints_pixter.hh`.

**10.130.3.2** `template<typename I> void mln::dpoints_fwd_pixter< I >::invalidate ( ) [inline]`

Invalidate the iterator.

Definition at line 325 of file `dpoints_pixter.hh`.

**10.130.3.3** `template<typename I> bool mln::dpoints_fwd_pixter< I >::is_valid ( ) const [inline]`

Test the iterator validity.

Definition at line 317 of file `dpoints_pixter.hh`.

Referenced by `mln::dpoints_fwd_pixter< I >::update()`.

**10.130.3.4** `void mln::Iterator< dpoints_fwd_pixter< I > >::next ( ) [inherited]`

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

#### Precondition

The iterator is valid.

**10.130.3.5** `template<typename I> void mln::dpoints_fwd_pixter< I >::start ( ) [inline]`

Manipulation.

Start an iteration.

Definition at line 298 of file `dpoints_pixter.hh`.

References `mln::dpoints_fwd_pixter< I >::update()`.

**10.130.3.6** `template<typename I> void mln::dpoints_fwd_pixter< I >::update ( ) [inline]`

Force this iterator to update its location to take into account that its center point may have moved.

Definition at line 284 of file `dpoints_pixter.hh`.

References `mln::dpoints_fwd_pixter< I >::is_valid()`.

Referenced by `mln::dpoints_fwd_pixter< I >::start()`.

## 10.131 mln::Edge< E > Struct Template Reference

edge category flag type.

```
#include <edge.hh>
```

### 10.131.1 Detailed Description

**template<typename E> struct mln::Edge< E >**

edge category flag type.

Definition at line 52 of file edge.hh.

## 10.132 mln::edge\_image< P, V, G > Class Template Reference

[Image](#) based on graph edges.

```
#include <edge_image.hh>
```

Inherits [image\\_base< fun::i2v::array< V >, p\\_edges< G, internal::efsite\\_selector< P, G >::site\\_function\\_t>, edge\\_image< P, V, G >>](#).

### Public Types

- typedef [graph\\_elt\\_neighborhood< G, p\\_edges< G, site\\_function\\_t >> edge\\_nbh\\_t](#)  
*Neighborhood type.*
- typedef [graph\\_elt\\_window< G, p\\_edges< G, site\\_function\\_t >> edge\\_win\\_t](#)  
*Edge Window type.*
- typedef G [graph\\_t](#)  
*The type of the underlying graph.*
- typedef [edge\\_nbh\\_t nbh\\_t](#)  
*Default Neighborhood type.*
- typedef [internal::efsite\\_selector< P, G >::site\\_function\\_t site\\_function\\_t](#)  
*Function mapping graph elements to sites.*
- typedef [edge\\_image< tag::psite\\_< P >, tag::value\\_< V >, tag::graph\\_< G >> skeleton](#)  
*Skeleton type.*
- typedef [edge\\_win\\_t win\\_t](#)  
*Default Window type.*

## Public Member Functions

- [edge\\_image](#) ()  
*Constructors.*
- rvalue [operator](#)() (unsigned e\_id) const  
*Value accessors/operators overloads.*

### 10.132.1 Detailed Description

`template<typename P, typename V, typename G = util::graph> class mln::edge_image< P, V, G >`

[Image](#) based on graph edges.

Definition at line 123 of file core/image/edge\_image.hh.

### 10.132.2 Member Typedef Documentation

**10.132.2.1** `template<typename P, typename V, typename G = util::graph> typedef graph_elt_neighborhood<G,p_edges<G,site_function_t> > mln::edge_image< P, V, G >::edge_nbh_t`

[Neighborhood](#) type.

Definition at line 153 of file core/image/edge\_image.hh.

**10.132.2.2** `template<typename P, typename V, typename G = util::graph> typedef graph_elt_window<G,p_edges<G,site_function_t> > mln::edge_image< P, V, G >::edge_win_t`

[Edge Window](#) type.

Definition at line 151 of file core/image/edge\_image.hh.

**10.132.2.3** `template<typename P, typename V, typename G = util::graph> typedef G mln::edge_image< P, V, G >::graph_t`

The type of the underlying graph.

Definition at line 138 of file core/image/edge\_image.hh.

**10.132.2.4** `template<typename P, typename V, typename G = util::graph> typedef edge_nbh_t mln::edge_image< P, V, G >::nbh_t`

Default [Neighborhood](#) type.

Definition at line 159 of file core/image/edge\_image.hh.

**10.132.2.5** `template<typename P, typename V, typename G = util::graph> typedef internal::efsite_selector<P,G>::site_function_t mln::edge_image< P, V, G >::site_function_t`

[Function](#) mapping graph elements to sites.

Definition at line 147 of file core/image/edge\_image.hh.

**10.132.2.6** `template<typename P, typename V, typename G = util::graph> typedef edge_image< tag::psite_<P>, tag::value_<V>, tag::graph_<G> > mln::edge_image< P, V, G >::skeleton`

Skeleton type.

Definition at line 143 of file core/image/edge\_image.hh.

**10.132.2.7** `template<typename P, typename V, typename G = util::graph> typedef edge_win_t mln::edge_image< P, V, G >::win_t`

Default [Window](#) type.

Definition at line 156 of file core/image/edge\_image.hh.

### 10.132.3 Constructor & Destructor Documentation

**10.132.3.1** `template<typename P, typename V, typename G > mln::edge_image< P, V, G >::edge_image( ) [inline]`

Constructors.

Definition at line 248 of file core/image/edge\_image.hh.

### 10.132.4 Member Function Documentation

**10.132.4.1** `template<typename P, typename V, typename G > edge_image< P, V, G >::rvalue mln::edge_image< P, V, G >::operator() ( unsigned e_id ) const`

[Value](#) accessors/operators overloads.

Definition at line 302 of file core/image/edge\_image.hh.

## 10.133 mln::extended< I > Struct Template Reference

Makes an image become restricted by a point set.

```
#include <extended.hh>
```

Inherits `image_domain_morpher< I, box< I::site >, extended< I > >`.

### Public Types

- `typedef tag::image_< I > skeleton`

*Skeleton.*

- typedef I::value [value](#)  
*Value type.*

## Public Member Functions

- const [box](#)< typename I::site > & [domain](#) () const  
*Give the definition domain.*
- [extended](#) ()  
*Constructor without argument.*
- [extended](#) (I &ima, const [box](#)< typename I::site > &b)  
*Constructor.*

### 10.133.1 Detailed Description

**template<typename I> struct mln::extended< I >**

Makes an image become restricted by a point set.

Definition at line 92 of file extended.hh.

### 10.133.2 Member Typedef Documentation

**10.133.2.1 template<typename I> typedef tag::image\_<I> mln::extended< I >::skeleton**

Skeleton.

Definition at line 102 of file extended.hh.

**10.133.2.2 template<typename I> typedef I ::value mln::extended< I >::value**

[Value](#) type.

Definition at line 99 of file extended.hh.

### 10.133.3 Constructor & Destructor Documentation

**10.133.3.1 template<typename I > mln::extended< I >::extended ( ) [inline]**

Constructor without argument.

Definition at line 169 of file extended.hh.



**10.133.3.2** `template<typename I > mln::extended< I >::extended ( I & ima, const box< typename I::site > & b ) [inline]`

Constructor.

Definition at line 175 of file extended.hh.

### 10.133.4 Member Function Documentation

**10.133.4.1** `template<typename I > const box< typename I::site > & mln::extended< I >::domain ( ) const [inline]`

Give the definition domain.

Definition at line 192 of file extended.hh.

## 10.134 mln::extension\_fun< I, F > Class Template Reference

Extends the domain of an image with a function.

```
#include <extension_fun.hh>
```

Inherits image\_identity< I, I::domain\_t, extension\_fun< I, F > >.

### Public Types

- typedef I::value [rvalue](#)  
*Return type of read-only access.*
- typedef [extension\\_fun](#)< tag::image\_< I >, tag::function\_< F > > [skeleton](#)  
*Skeleton.*
- typedef I::value [value](#)  
*Image value type.*

### Public Member Functions

- const F & [extension](#) () const  
*Give the extension function.*
- [extension\\_fun](#) (I &ima, const F &fun)  
*Constructor from an image *ima* and a function *fun*.*
- [extension\\_fun](#) ()  
*Constructor without argument.*
- template<typename P >  
bool [has](#) (const P &p) const  
*Test if *p* is valid.*

- `internal::morpher_lvalue_< I >::ret operator()` (const typename I::psite &p)  
*Read-write access to the image value located at site p.*
- `I::value operator()` (const typename I::psite &p) const  
*Read-only access to the image value located at site p.*

### 10.134.1 Detailed Description

`template<typename I, typename F> class mln::extension_fun< I, F >`

Extends the domain of an image with a function.

Definition at line 99 of file extension\_fun.hh.

### 10.134.2 Member Typedef Documentation

**10.134.2.1** `template<typename I, typename F> typedef I ::value mln::extension_fun< I, F >::rvalue`

Return type of read-only access.

Definition at line 112 of file extension\_fun.hh.

**10.134.2.2** `template<typename I, typename F> typedef extension_fun< tag::image_<I>, tag::function_<F> > mln::extension_fun< I, F >::skeleton`

Skeleton.

Definition at line 106 of file extension\_fun.hh.

**10.134.2.3** `template<typename I, typename F> typedef I ::value mln::extension_fun< I, F >::value`

[Image](#) value type.

Definition at line 109 of file extension\_fun.hh.

### 10.134.3 Constructor & Destructor Documentation

**10.134.3.1** `template<typename I, typename F > mln::extension_fun< I, F >::extension_fun ( )`  
`[inline]`

Constructor without argument.

Definition at line 178 of file extension\_fun.hh.

**10.134.3.2** `template<typename I, typename F > mln::extension_fun< I, F >::extension_fun ( I & ima, const F & fun ) [inline]`

Constructor from an image `ima` and a function `fun`.

Definition at line 184 of file `extension_fun.hh`.

## 10.134.4 Member Function Documentation

**10.134.4.1** `template<typename I, typename F > const F & mln::extension_fun< I, F >::extension ( ) const [inline]`

Give the extension function.

Definition at line 243 of file `extension_fun.hh`.

**10.134.4.2** `template<typename I, typename F > template<typename P > bool mln::extension_fun< I, F >::has ( const P & p ) const [inline]`

Test if `p` is valid.

It returns always true, assuming that the function is valid for any `p`.

Definition at line 201 of file `extension_fun.hh`.

**10.134.4.3** `template<typename I, typename F > internal::morpher_lvalue_< I >::ret mln::extension_fun< I, F >::operator() ( const typename I::psite & p ) [inline]`

Read-write access to the image value located at site `p`.

Definition at line 223 of file `extension_fun.hh`.

**10.134.4.4** `template<typename I, typename F > I::value mln::extension_fun< I, F >::operator() ( const typename I::psite & p ) const [inline]`

Read-only access to the image value located at site `p`;

Definition at line 209 of file `extension_fun.hh`.

## 10.135 mln::extension\_ima< I, J > Class Template Reference

Extends the domain of an image with an image.

```
#include <extension_ima.hh>
```

Inherits `image_identity< I, I::domain_t, extension_ima< I, J > >`.

### Public Types

- typedef `I::value` [rvalue](#)

*Return type of read-only access.*

- typedef `extension_ima`< tag::image\_< I >, tag::ext\_< J > > `skeleton`  
*Skeleton.*
- typedef I::value `value`  
*Image value type.*

## Public Member Functions

- const J & `extension` () const  
*Read-only access to the extension domain (image).*
- `extension_ima` (I &ima, const J &ext)  
*Constructor from an image `ima` and a function `ext`.*
- `extension_ima` ()  
*Constructor without argument.*
- template<typename P >  
bool `has` (const P &p) const  
*Test if `p` is valid.*
- internal::morpher\_ivalue\_< I >::ret `operator`() (const typename I::psite &p)  
*Read-write access to the image value located at site `p`.*
- I::value `operator`() (const typename I::psite &p) const  
*Read-only access to the image value located at site `p`.*

### 10.135.1 Detailed Description

`template<typename I, typename J> class mln::extension_ima< I, J >`

Extends the domain of an image with an image.

Definition at line 97 of file `extension_ima.hh`.

### 10.135.2 Member Typedef Documentation

**10.135.2.1** `template<typename I, typename J> typedef I ::value mln::extension_ima< I, J >::rvalue`

Return type of read-only access.

Definition at line 111 of file `extension_ima.hh`.

**10.135.2.2** `template<typename I, typename J> typedef extension_ima< tag::image_<I>, tag::ext_<J> > mln::extension_ima< I, J >::skeleton`

Skeleton.

Definition at line 105 of file extension\_ima.hh.

**10.135.2.3** `template<typename I, typename J> typedef I ::value mln::extension_ima< I, J >::value`

[Image](#) value type.

Definition at line 108 of file extension\_ima.hh.

### 10.135.3 Constructor & Destructor Documentation

**10.135.3.1** `template<typename I, typename J > mln::extension_ima< I, J >::extension_ima ( ) [inline]`

Constructor without argument.

Definition at line 173 of file extension\_ima.hh.

**10.135.3.2** `template<typename I, typename J > mln::extension_ima< I, J >::extension_ima ( I & ima, const J & ext ) [inline]`

Constructor from an image `ima` and a function `ext`.

Definition at line 179 of file extension\_ima.hh.

### 10.135.4 Member Function Documentation

**10.135.4.1** `template<typename I, typename J > const J & mln::extension_ima< I, J >::extension ( ) const [inline]`

Read-only access to the extension domain (image).

Definition at line 244 of file extension\_ima.hh.

**10.135.4.2** `template<typename I, typename J > template<typename P > bool mln::extension_ima< I, J >::has ( const P & p ) const [inline]`

Test if `p` is valid.

Definition at line 196 of file extension\_ima.hh.

**10.135.4.3** `template<typename I, typename J > internal::morpher_lvalue_< I >::ret mln::extension_ima< I, J >::operator() ( const typename I::psite & p ) [inline]`

Read-write access to the image value located at site `p`.

Definition at line 223 of file extension\_ima.hh.

#### 10.135.4.4 `template<typename I, typename J > I::value mln::extension_ima< I, J >::operator()` `( const typename I::psite & p ) const [inline]`

Read-only access to the image value located at site `p`;

Definition at line 208 of file `extension_ima.hh`.

### 10.136 `mln::extension_val< I >` Class Template Reference

Extends the domain of an image with a value.

```
#include <extension_val.hh>
```

Inherits `image_identity< I, I::domain_t, extension_val< I > >`.

#### Public Types

- typedef `I::value` `rvalue`  
*Return type of read-only access.*
- typedef `extension_val< tag::image_< I > >` `skeleton`  
*Skeleton.*
- typedef `I::value` `value`  
*Image value type.*

#### Public Member Functions

- void `change_extension` (const typename `I::value` &val)  
*Change the value of the extension domain.*
- const `I::value` & `extension` () const  
*Read-only access to the value of the extension domain.*
- `extension_val` (I &ima, const typename `I::value` &val)  
*Constructor from an image `ima` and a value `val`.*
- `extension_val` ()  
*Constructor without argument.*
- template<typename `P` >  
bool `has` (const `P` &p) const  
*Test if `p` is valid. It returns always true.*
- internal::`morpher_lvalue_< I >::ret operator()` (const typename `I::psite` &p)  
*Read-write access to the image value located at site `p`.*
- `I::value operator()` (const typename `I::psite` &p) const  
*Read-only access to the image value located at site `p`;*

## 10.136.1 Detailed Description

**template<typename I> class mln::extension\_val< I >**

Extends the domain of an image with a value.

Definition at line 98 of file extension\_val.hh.

## 10.136.2 Member Typedef Documentation

**10.136.2.1 template<typename I> typedef I ::value mln::extension\_val< I >::rvalue**

Return type of read-only access.

Definition at line 110 of file extension\_val.hh.

**10.136.2.2 template<typename I> typedef extension\_val< tag::image\_<I> > mln::extension\_val< I >::skeleton**

Skeleton.

Definition at line 104 of file extension\_val.hh.

**10.136.2.3 template<typename I> typedef I ::value mln::extension\_val< I >::value**

[Image](#) value type.

Definition at line 107 of file extension\_val.hh.

## 10.136.3 Constructor & Destructor Documentation

**10.136.3.1 template<typename I> mln::extension\_val< I >::extension\_val ( ) [inline]**

Constructor without argument.

Definition at line 176 of file extension\_val.hh.

**10.136.3.2 template<typename I> mln::extension\_val< I >::extension\_val ( I & ima, const typename I::value & val ) [inline]**

Constructor from an image *ima* and a value *val*.

Definition at line 182 of file extension\_val.hh.

## 10.136.4 Member Function Documentation

**10.136.4.1 template<typename I> void mln::extension\_val< I >::change\_extension ( const typename I::value & val ) [inline]**

Change the value of the extension domain.

Definition at line 247 of file extension\_val.hh.

**10.136.4.2** `template<typename I> const I::value & mln::extension_val< I >::extension ( ) const [inline]`

Read-only access to the value of the extension domain.

Definition at line 238 of file extension\_val.hh.

**10.136.4.3** `template<typename I> template<typename P> bool mln::extension_val< I >::has ( const P & p ) const [inline]`

Test if `p` is valid. It returns always true.

Definition at line 199 of file extension\_val.hh.

**10.136.4.4** `template<typename I> internal::morpher_lvalue_< I >::ret mln::extension_val< I >::operator() ( const typename I::psite & p ) [inline]`

Read-write access to the image value located at site `p`.

Definition at line 220 of file extension\_val.hh.

**10.136.4.5** `template<typename I> I::value mln::extension_val< I >::operator() ( const typename I::psite & p ) const [inline]`

Read-only access to the image value located at site `p`;

Definition at line 207 of file extension\_val.hh.

## 10.137 mln::faces\_psite< N, D, P > Class Template Reference

[Point](#) site associated to a [mln::p\\_faces](#).

```
#include <faces_psite.hh>
```

Inherits `pseudo_site_base_< const P &, faces_psite< N, D, P > >`.

### Public Member Functions

- [faces\\_psite](#) ()  
*Construction and assignment.*
- [faces\\_psite](#) (const [p\\_faces](#)< N, D, P > &pf, const [topo::n\\_face](#)< N, D > &face)
- bool [is\\_valid](#) () const  
*Psite manipulators.*
- void [invalidate](#) ()  
*Invalidate this psite.*
- const [target](#) & [site\\_set](#) () const  
*Site set manipulators.*



- void `change_target` (const `target` &new\_target)  
*Set the target site\_set.*
- `topo::n_face`< N, D > `face` () const  
*Face handle manipulators.*
- unsigned `n` () const  
*Return the dimension of the face of this psite.*
- unsigned `face_id` () const  
*Return the id of the face of this psite.*

### 10.137.1 Detailed Description

`template<unsigned N, unsigned D, typename P> class mln::faces_psite< N, D, P >`

`Point` site associated to a `mln::p_faces`.

#### Template Parameters

- N* The dimension of the face associated to this psite.
- D* The dimension of the complex this psite belongs to.
- P* The type of point associated to this psite.

Definition at line 56 of file `faces_psite.hh`.

### 10.137.2 Constructor & Destructor Documentation

**10.137.2.1** `template<unsigned N, unsigned D, typename P > mln::faces_psite< N, D, P >::faces_psite ( ) [inline]`

Construction and assignment.

Definition at line 190 of file `faces_psite.hh`.

References `mln::faces_psite< N, D, P >::invalidate()`.

**10.137.2.2** `template<unsigned N, unsigned D, typename P > mln::faces_psite< N, D, P >::faces_psite ( const p_faces< N, D, P > & pf, const topo::n_face< N, D > & face ) [inline]`

#### Precondition

`pf.cplx() == face.cplx()`.

Definition at line 201 of file `faces_psite.hh`.

### 10.137.3 Member Function Documentation

**10.137.3.1** `template<unsigned N, unsigned D, typename P > void mln::faces_psite< N, D, P >::change_target ( const target & new_target ) [inline]`

Set the target site\_set.

Definition at line 265 of file faces\_psite.hh.

References mln::p\_faces< N, D, P >::cplx(), and mln::faces\_psite< N, D, P >::invalidate().

**10.137.3.2** `template<unsigned N, unsigned D, typename P > topo::n_face< N, D > mln::faces_psite< N, D, P >::face ( ) const [inline]`

Face handle manipulators.

Return the face handle of this point site.

Definition at line 289 of file faces\_psite.hh.

Referenced by mln::operator!==( ), and mln::operator==( ).

**10.137.3.3** `template<unsigned N, unsigned D, typename P > unsigned mln::faces_psite< N, D, P >::face_id ( ) const [inline]`

Return the id of the face of this psite.

Definition at line 305 of file faces\_psite.hh.

**10.137.3.4** `template<unsigned N, unsigned D, typename P > void mln::faces_psite< N, D, P >::invalidate ( ) [inline]`

Invalidate this psite.

Definition at line 239 of file faces\_psite.hh.

Referenced by mln::faces\_psite< N, D, P >::change\_target(), and mln::faces\_psite< N, D, P >::faces\_psite().

**10.137.3.5** `template<unsigned N, unsigned D, typename P > bool mln::faces_psite< N, D, P >::is_valid ( ) const [inline]`

Psite manipulators.

Is this psite valid?

Definition at line 230 of file faces\_psite.hh.

**10.137.3.6** `template<unsigned N, unsigned D, typename P > unsigned mln::faces_psite< N, D, P >::n ( ) const [inline]`

Return the dimension of the face of this psite.

Definition at line 297 of file faces\_psite.hh.

**10.137.3.7** `template<unsigned N, unsigned D, typename P > const p_faces< N, D, P > & mln::faces_psite< N, D, P >::site_set ( ) const [inline]`

[Site](#) set manipulators.

Return the [p\\_faces](#) this site is built on. (shortcut for \*target()).

#### Precondition

Member `face_` is valid.

Definition at line 247 of file `faces_psite.hh`.

Referenced by `mln::operator!=()`, and `mln::operator==()`.

## 10.138 mln::flat\_image< T, S > Struct Template Reference

[Image](#) with a single value.

```
#include <flat_image.hh>
```

Inherits `image_primary< T, S, flat_image< T, S > >`.

### Public Types

- typedef T & [lvalue](#)  
*Return type of read-write access.*
- typedef const T & [rvalue](#)  
*Return type of read-only access.*
- typedef [flat\\_image](#)< tag::value\_< T >, tag::domain\_< S > > [skeleton](#)  
*Skeleton.*
- typedef T [value](#)  
*Value associated type.*

### Public Member Functions

- const S & [domain](#) () const  
*Give the definition domain.*
- [flat\\_image](#) (const T &val, const S &pset)  
*Constructor.*
- [flat\\_image](#) ()  
*Constructor without argument.*
- bool [has](#) (const typename S::psite &p) const  
*Test if p is valid: always return true.*

- `const T & operator()` (`const typename S::psite &p`) `const`  
*Read-only access to the image value located at point  $p$ .*
- `T & operator()` (`const typename S::psite &p`)  
*Read-write access to the image value located at point  $p$ .*

### 10.138.1 Detailed Description

`template<typename T, typename S> struct mln::flat_image< T, S >`

[Image](#) with a single value.

Definition at line 105 of file `flat_image.hh`.

### 10.138.2 Member Typedef Documentation

**10.138.2.1** `template<typename T, typename S> typedef T& mln::flat_image< T, S >::lvalue`

Return type of read-write access.

Definition at line 118 of file `flat_image.hh`.

**10.138.2.2** `template<typename T, typename S> typedef const T& mln::flat_image< T, S >::rvalue`

Return type of read-only access.

Definition at line 115 of file `flat_image.hh`.

**10.138.2.3** `template<typename T, typename S> typedef flat_image< tag::value_<T>, tag::domain_<S> > mln::flat_image< T, S >::skeleton`

Skeleton.

Definition at line 108 of file `flat_image.hh`.

**10.138.2.4** `template<typename T, typename S> typedef T mln::flat_image< T, S >::value`

[Value](#) associated type.

Definition at line 112 of file `flat_image.hh`.

### 10.138.3 Constructor & Destructor Documentation

**10.138.3.1** `template<typename T, typename S > mln::flat_image< T, S >::flat_image ( )`  
`[inline]`

Constructor without argument.

Definition at line 191 of file `flat_image.hh`.

**10.138.3.2** `template<typename T , typename S > mln::flat_image< T, S >::flat_image ( const T & val, const S & pset ) [inline]`

Constructor.

Definition at line 197 of file flat\_image.hh.

## 10.138.4 Member Function Documentation

**10.138.4.1** `template<typename T , typename S > const S & mln::flat_image< T, S >::domain ( ) const [inline]`

Give the definition domain.

Definition at line 214 of file flat\_image.hh.

**10.138.4.2** `template<typename T , typename S > bool mln::flat_image< T, S >::has ( const typename S::psite & p ) const [inline]`

Test if p is valid: always return true.

Definition at line 222 of file flat\_image.hh.

**10.138.4.3** `template<typename T , typename S > const T & mln::flat_image< T, S >::operator() ( const typename S::psite & p ) const [inline]`

Read-only access to the image value located at point p.

Definition at line 230 of file flat\_image.hh.

**10.138.4.4** `template<typename T , typename S > T & mln::flat_image< T, S >::operator() ( const typename S::psite & p ) [inline]`

Read-write access to the image value located at point p.

Definition at line 239 of file flat\_image.hh.

## 10.139 mln::fun::from\_accu< A > Struct Template Reference

Wrap an accumulator into a function.

```
#include <from_accu.hh>
```

Inherits mln::fun::unary\_param< from\_accu< A >, A \* >.

### 10.139.1 Detailed Description

`template<typename A> struct mln::fun::from_accu< A >`

Wrap an accumulator into a function.

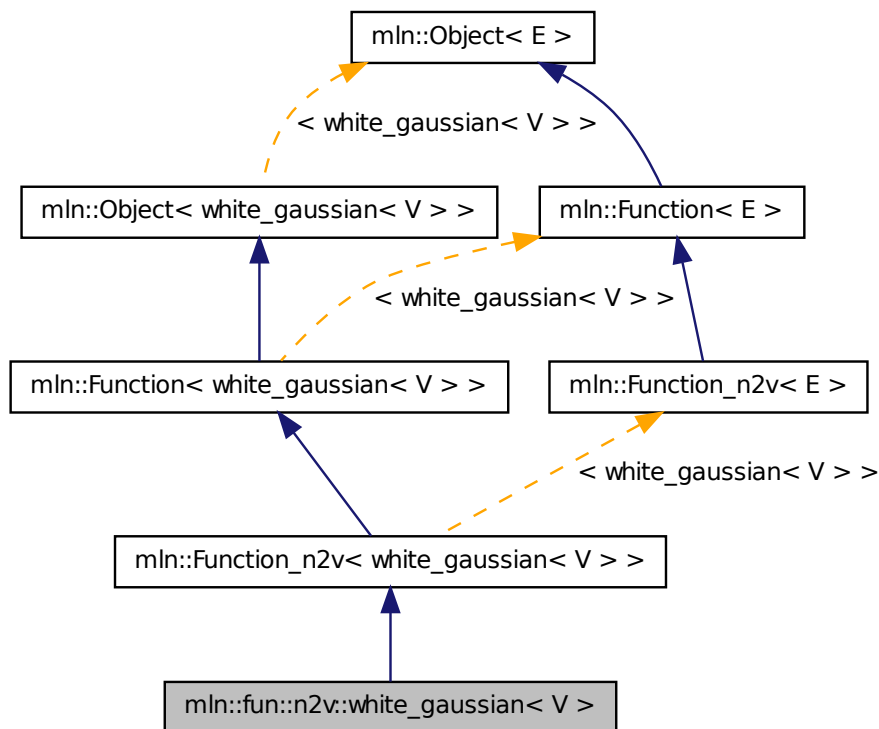
Definition at line 49 of file from\_accu.hh.

## 10.140 mln::fun::n2v::white\_gaussian< V > Struct Template Reference

Generate a White Gaussian Noise.

```
#include <white_gaussian.hh>
```

Inheritance diagram for mln::fun::n2v::white\_gaussian< V >:



### 10.140.1 Detailed Description

```
template<typename V> struct mln::fun::n2v::white_gaussian< V >
```

Generate a White Gaussian Noise. Reference: <http://www.dspguru.com/dsp/howtos/how-to-generate-white-gaussian-noise>

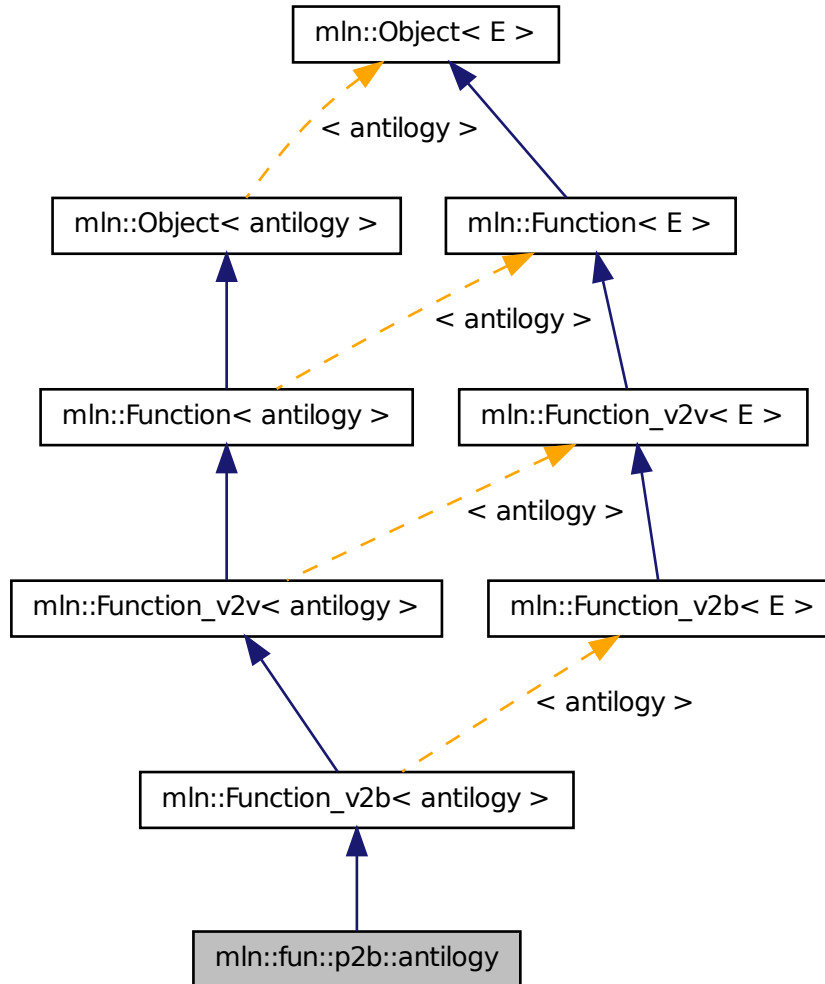
Definition at line 56 of file `white_gaussian.hh`.

## 10.141 mln::fun::p2b::antilogy Struct Reference

A `p2b` function always returning `false`.

```
#include <antilogy.hh>
```

Inheritance diagram for mln::fun::p2b::antilogy:



### 10.141.1 Detailed Description

A [p2b](#) function always returning `false`. A simpler name would be `false`, but this is not a valid C++ identifier, as `false` is a keyword of the language.

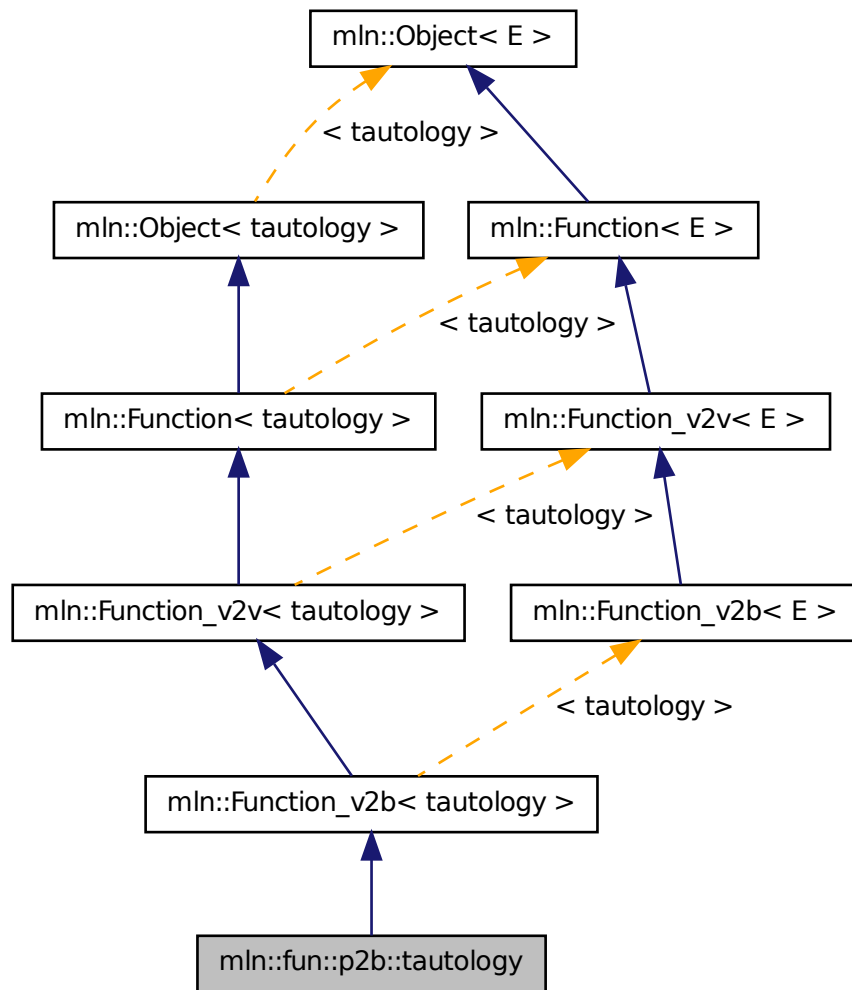
Definition at line 50 of file `antilogy.hh`.

## 10.142 mln::fun::p2b::tautology Struct Reference

A [p2b](#) function always returning `true`.

```
#include <tautology.hh>
```

Inheritance diagram for `mln::fun::p2b::tautology`:



### 10.142.1 Detailed Description

A [p2b](#) function always returning `true`. A simpler name would be 'true', but this is not a valid C++ identifier, as `true` is a keyword of the language.

Definition at line 50 of file `tautology.hh`.

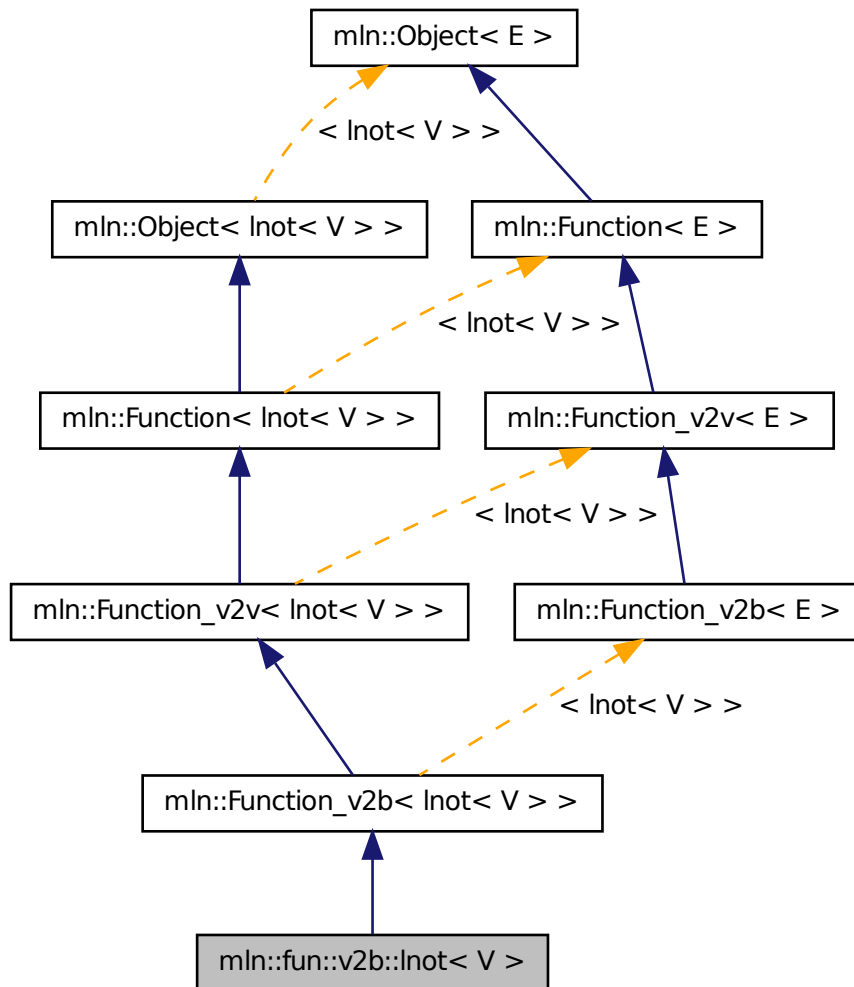


## 10.143 mln::fun::v2b::lnot< V > Struct Template Reference

Functor computing logical-not on a value.

```
#include <lnot.hh>
```

Inheritance diagram for mln::fun::v2b::lnot< V >:



### 10.143.1 Detailed Description

```
template<typename V> struct mln::fun::v2b::lnot< V >
```

Functor computing logical-not on a value.

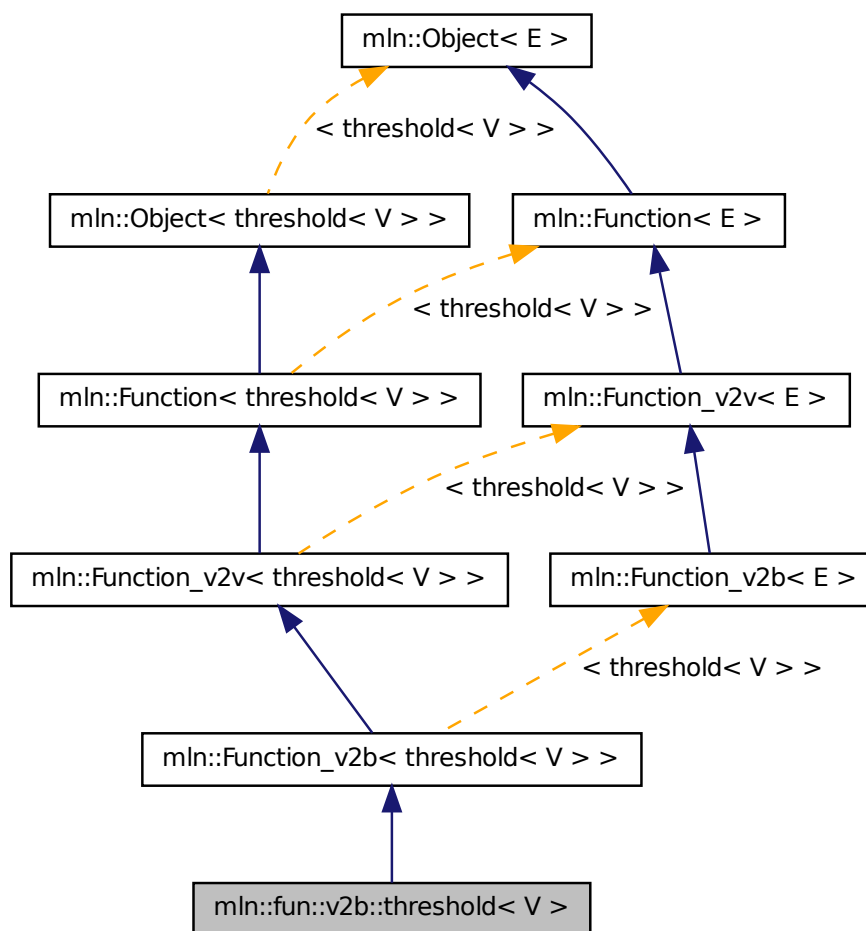
Definition at line 47 of file lnot.hh.

## 10.144 mln::fun::v2b::threshold< V > Struct Template Reference

Threshold function.

```
#include <threshold.hh>
```

Inheritance diagram for mln::fun::v2b::threshold< V >:



### 10.144.1 Detailed Description

```
template<typename V> struct mln::fun::v2b::threshold< V >
```

Threshold function.  $f(v) = (v \geq \text{threshold})$ .

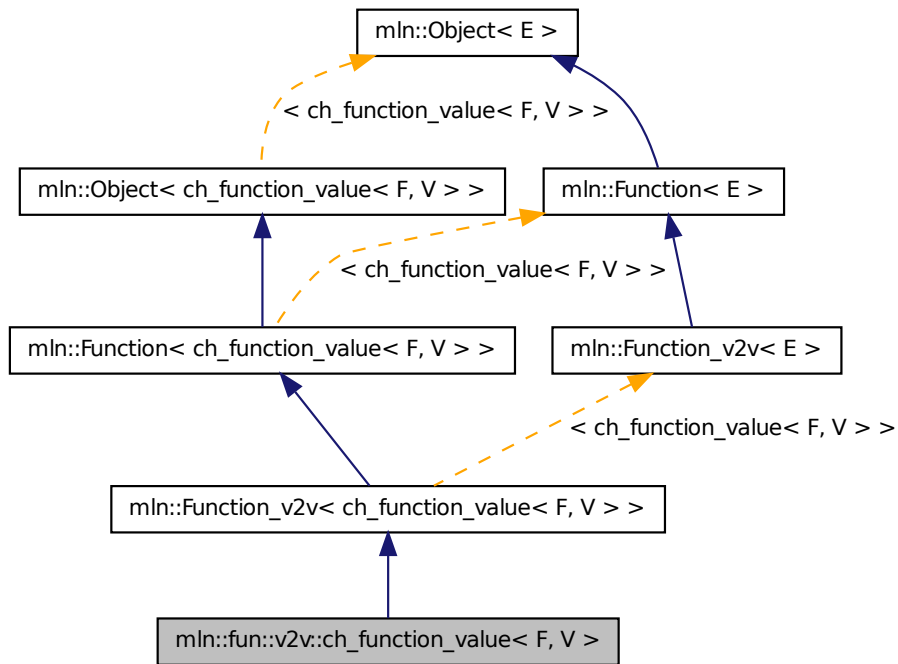
Definition at line 48 of file fun/v2b/threshold.hh.

## 10.145 mln::fun::v2v::ch\_function\_value< F, V > Class Template Reference

Wrap a function `v2v` and convert its result to another type.

```
#include <ch_function_value.hh>
```

Inheritance diagram for `mln::fun::v2v::ch_function_value< F, V >`:



### 10.145.1 Detailed Description

```
template<typename F, typename V> class mln::fun::v2v::ch_function_value< F, V >
```

Wrap a function `v2v` and convert its result to another type.

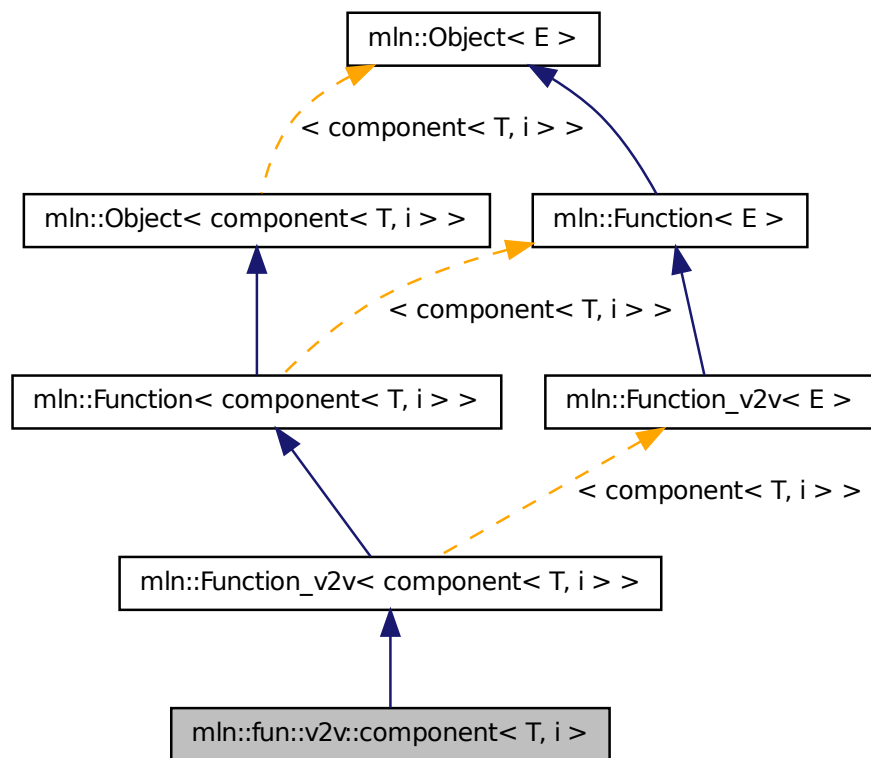
Definition at line 52 of file fun/v2v/ch\_function\_value.hh.

## 10.146 mln::fun::v2v::component< T, i > Struct Template Reference

Functor that accesses the i-th component of a value.

```
#include <component.hh>
```

Inheritance diagram for mln::fun::v2v::component< T, i >:



### 10.146.1 Detailed Description

```
template<typename T, unsigned i> struct mln::fun::v2v::component< T, i >
```

Functor that accesses the i-th component of a value.

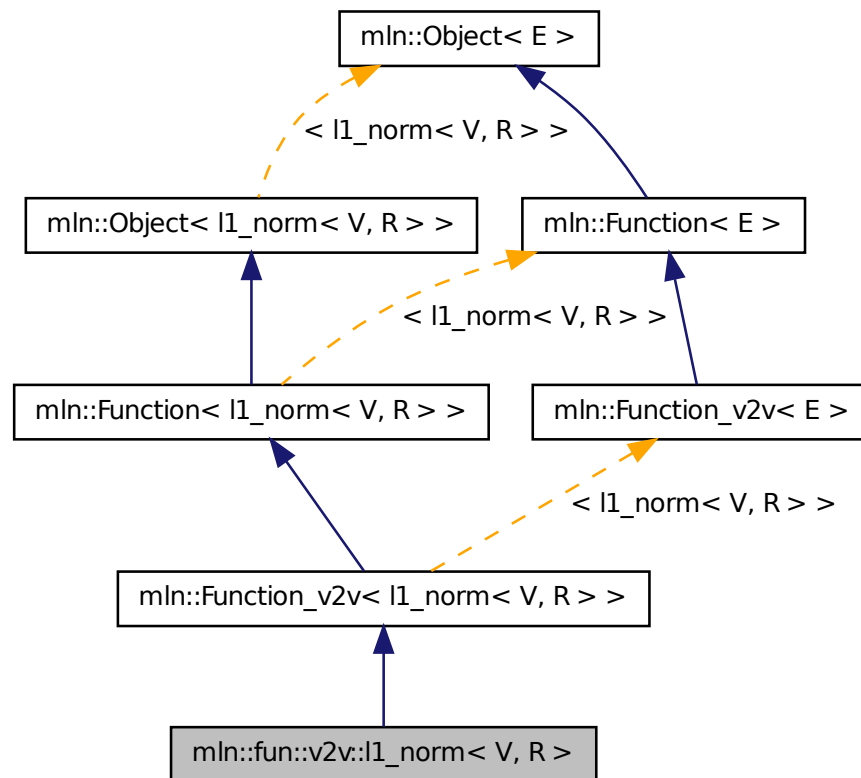
Definition at line 51 of file component.hh.

## 10.147 mln::fun::v2v::l1\_norm< V, R > Struct Template Reference

L1-norm.

```
#include <norm.hh>
```

Inheritance diagram for mln::fun::v2v::l1\_norm< V, R >:



### 10.147.1 Detailed Description

```
template<typename V, typename R> struct mln::fun::v2v::l1_norm< V, R >
```

L1-norm. V is the type of input values; R is the result type.

See also

[mln::norm::l1](#).

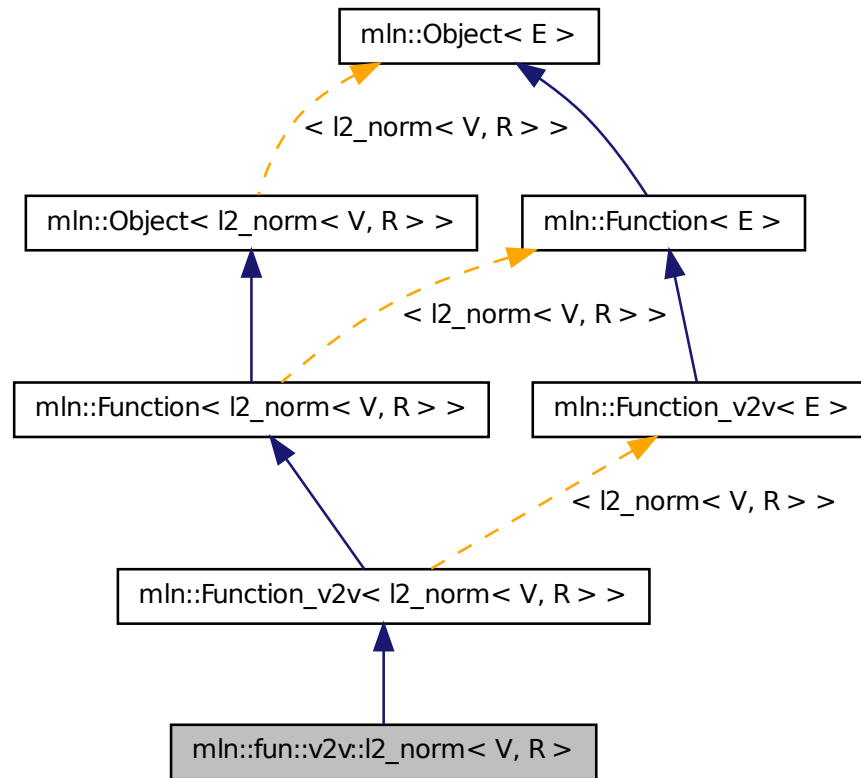
Definition at line 57 of file `v2v/norm.hh`.

## 10.148 mln::fun::v2v::l2\_norm< V, R > Struct Template Reference

L2-norm.

```
#include <norm.hh>
```

Inheritance diagram for `mln::fun::v2v::l2_norm< V, R >`:



### 10.148.1 Detailed Description

```
template<typename V, typename R> struct mln::fun::v2v::l2_norm< V, R >
```

L2-norm. `V` is the type of input values; `R` is the result type.

**See also**

`mln::norm::l2`.

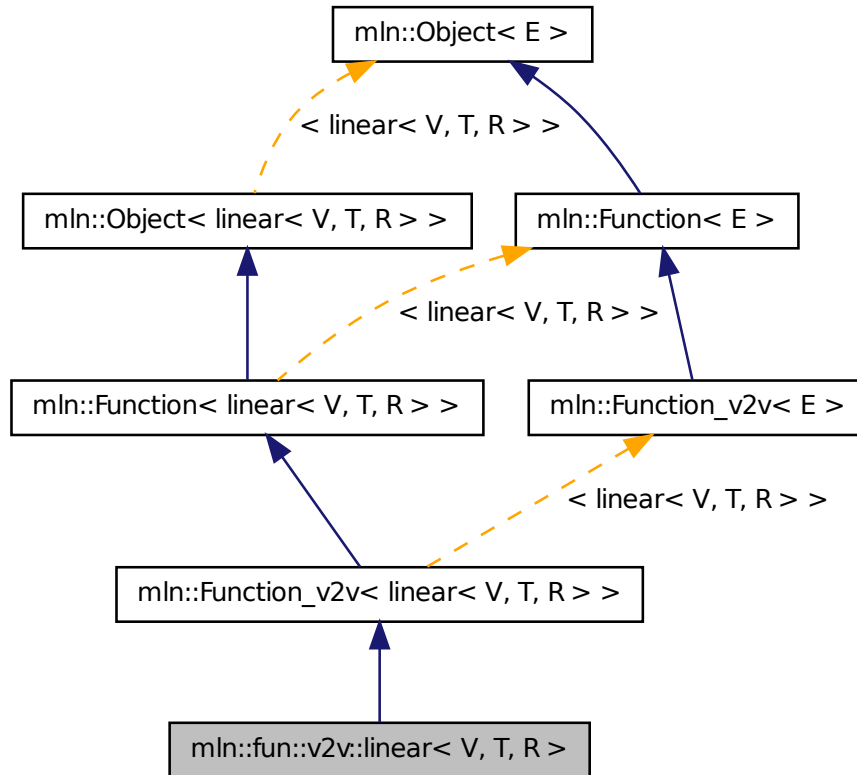
Definition at line 69 of file `v2v/norm.hh`.

## 10.149 `mln::fun::v2v::linear< V, T, R >` Struct Template Reference

Linear function.  $f(v) = a * v + b$ . `V` is the type of input values; `T` is the type used to compute the result; `R` is the result type.

```
#include <linear.hh>
```

Inheritance diagram for mln::fun::v2v::linear< V, T, R >:



### 10.149.1 Detailed Description

```
template<typename V, typename T = V, typename R = T> struct mln::fun::v2v::linear< V, T, R >
```

Linear function.  $f(v) = a * v + b$ .  $V$  is the type of input values;  $T$  is the type used to compute the result;  $R$  is the result type. By default,  $T$  is  $V$  and  $R$  is  $T$ .

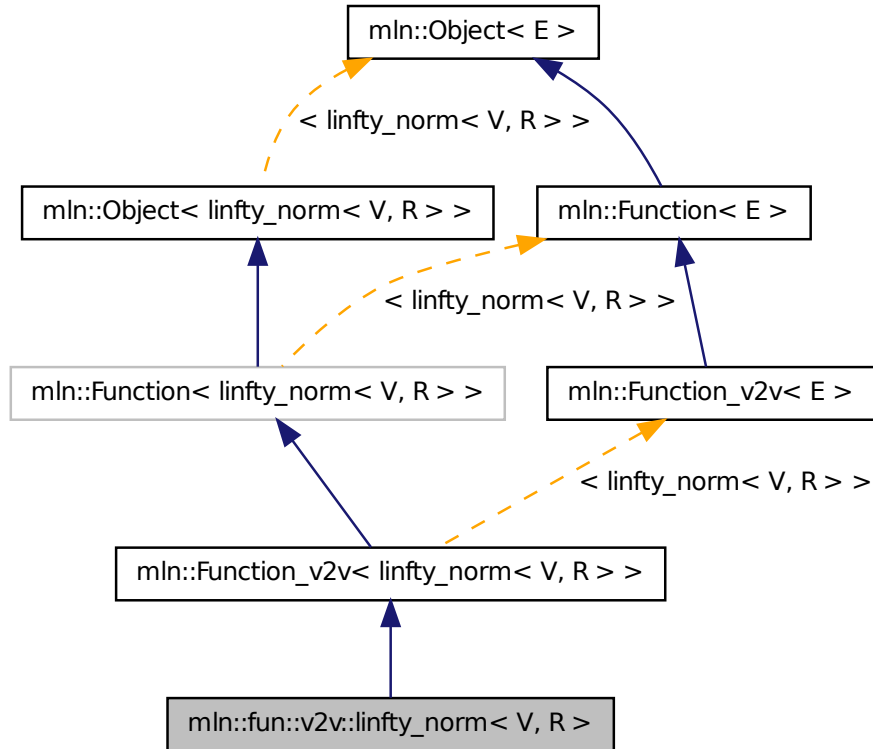
Definition at line 55 of file v2v/linear.hh.

## 10.150 mln::fun::v2v::linfty\_norm< V, R > Struct Template Reference

L-infty norm.

```
#include <norm.hh>
```

Inheritance diagram for `mln::fun::v2v::linfty_norm< V, R >`:



### 10.150.1 Detailed Description

```
template<typename V, typename R> struct mln::fun::v2v::linfty_norm< V, R >
```

L-infty norm. `V` is the type of input values; `R` is the result type.

See also

[mln::norm::linfty](#).

Definition at line 81 of file `v2v/norm.hh`.

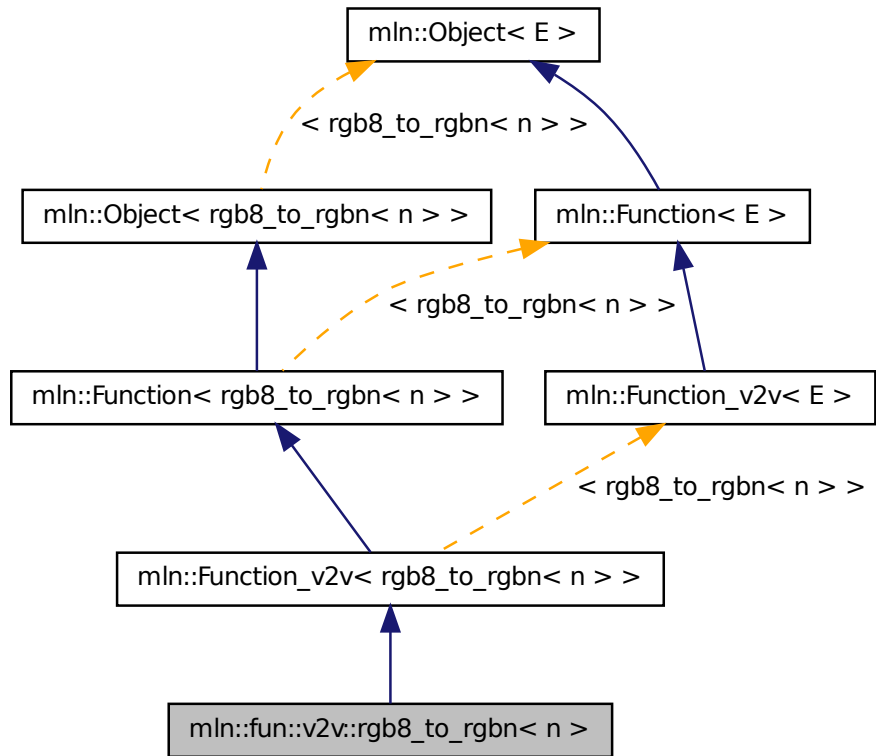
### 10.151 mln::fun::v2v::rgb8\_to\_rgn< n > Struct Template Reference

Convert a `rgb8` value to a `rgn`, `n < 8`.

```
#include <rgb8_to_rgn.hh>
```



Inheritance diagram for mln::fun::v2v::rgb8\_to\_rgn< n >:



## Public Member Functions

- `result operator()` (const `argument` &c) const  
Convert a `rgb8` value to a `rgn`,  $n < 8$ .

### 10.151.1 Detailed Description

```
template<unsigned n> struct mln::fun::v2v::rgb8_to_rgn< n >
```

Convert a `rgb8` value to a `rgn`,  $n < 8$ .

#### Parameters

`n` defines the output quantification used for the transformation.

Definition at line 56 of file `rgb8_to_rgn.hh`.

## 10.151.2 Member Function Documentation

**10.151.2.1** `template<unsigned n> rgb8_to_rgn< n >::result mln::fun::v2v::rgb8_to_rgn< n >::operator() ( const argument & c ) const`

Convert a rgb8 value to a rgn,  $n < 8$ .

### Parameters

[in] `v` the rgb8 value to convert.

Conversion is done by computing the size by which we divide each rgb component.

### Parameters

`n` defines the output quantification used for the transformation.

Definition at line 83 of file `rgb8_to_rgn.hh`.

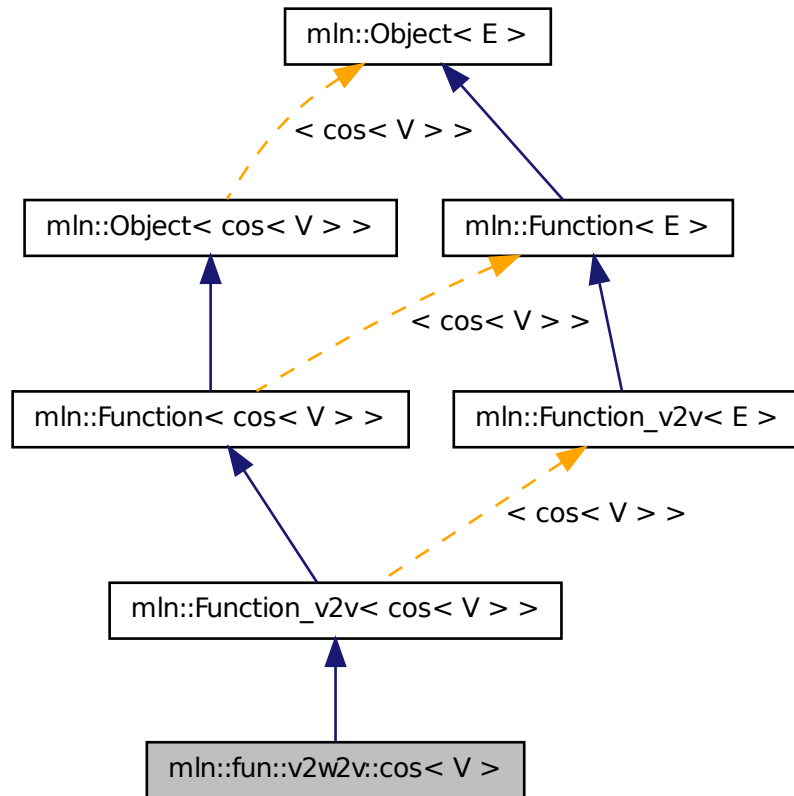
References `mln::value::rgb< n >::red()`.

## 10.152 mln::fun::v2w2v::cos< V > Struct Template Reference

Cosinus bijective functor.

```
#include <cos.hh>
```

Inheritance diagram for mln::fun::v2w2v::cos< V >:



### 10.152.1 Detailed Description

```
template<typename V> struct mln::fun::v2w2v::cos< V >
```

Cosinus bijective functor. `V` is the type of input values and the result type.

See also

`mln::math::cos`.

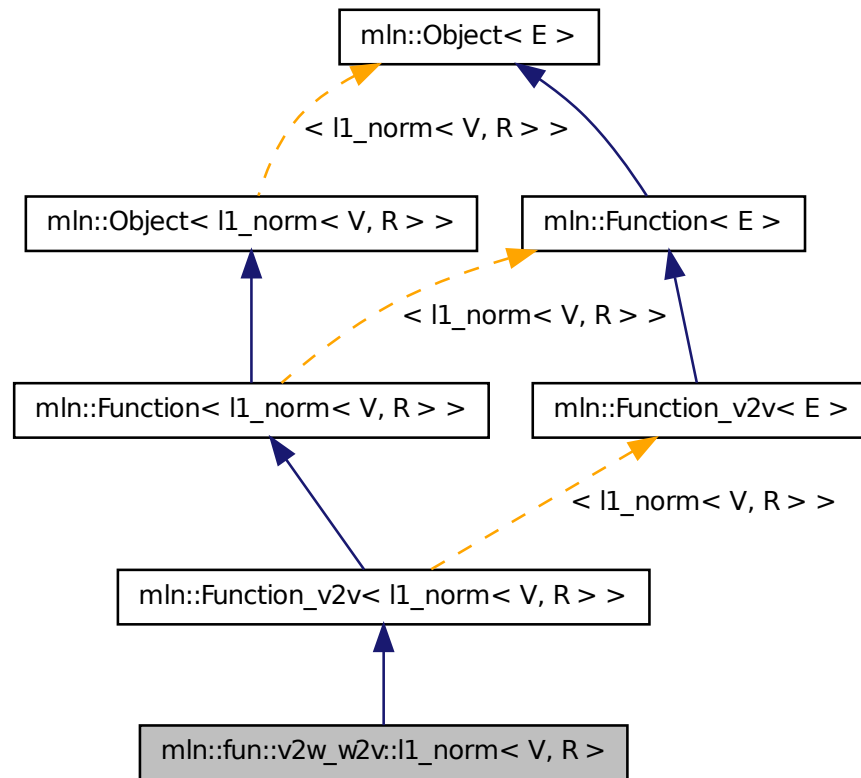
Definition at line 59 of file `fun/v2w2v/cos.hh`.

## 10.153 mln::fun::v2w\_w2v::l1\_norm< V, R > Struct Template Reference

L1-norm.

```
#include <norm.hh>
```

Inheritance diagram for `mln::fun::v2w_w2v::l1_norm< V, R >`:



### 10.153.1 Detailed Description

```
template<typename V, typename R> struct mln::fun::v2w_w2v::l1_norm< V, R >
```

L1-norm. `V` is the type of input values; `R` is the result type.

#### See also

[mln::norm::l1](#).

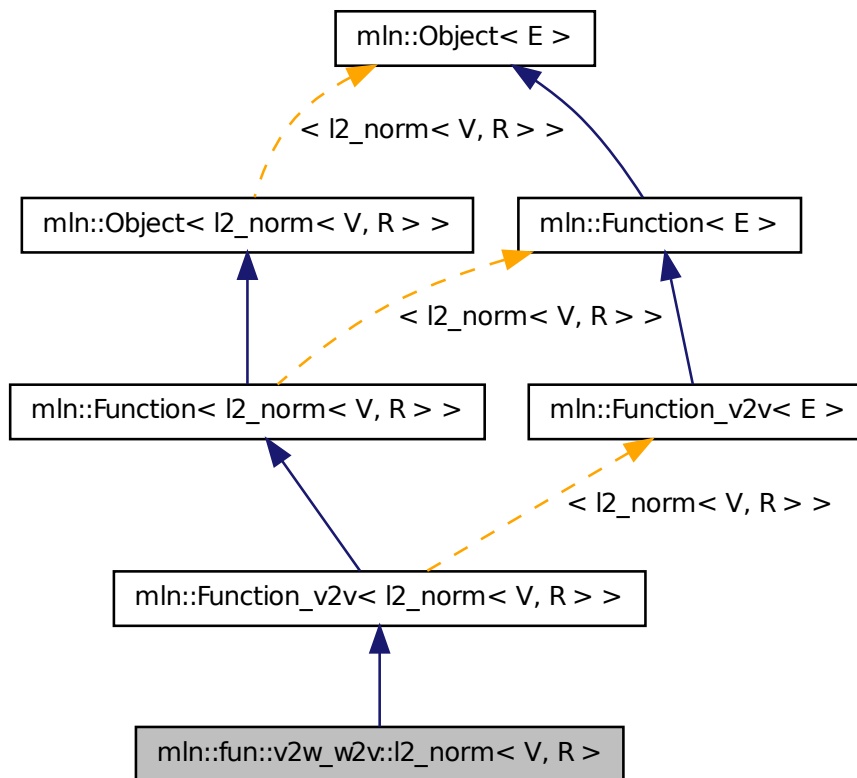
Definition at line 56 of file `v2w_w2v/norm.hh`.

## 10.154 mln::fun::v2w\_w2v::l2\_norm< V, R > Struct Template Reference

L2-norm.

```
#include <norm.hh>
```

Inheritance diagram for mln::fun::v2w\_w2v::l2\_norm< V, R >:



### 10.154.1 Detailed Description

```
template<typename V, typename R> struct mln::fun::v2w_w2v::l2_norm< V, R >
```

L2-norm. V is the type of input values; R is the result type.

See also

`mln::norm::l2`.

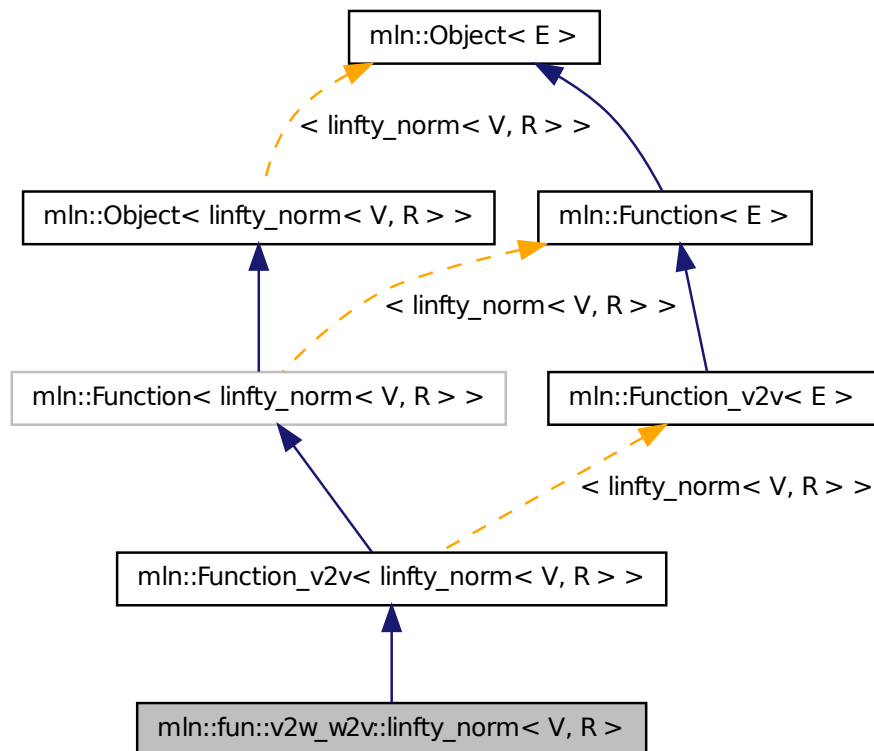
Definition at line 69 of file `v2w_w2v/norm.hh`.

## 10.155 mln::fun::v2w\_w2v::linfty\_norm< V, R > Struct Template Reference

L-infty norm.

```
#include <norm.hh>
```

Inheritance diagram for mln::fun::v2w\_w2v::linfty\_norm< V, R >:



### 10.155.1 Detailed Description

```
template<typename V, typename R> struct mln::fun::v2w_w2v::linfty_norm< V, R >
```

L-infty norm. V is the type of input values; R is the result type.

See also

[mln::norm::linfty](#).

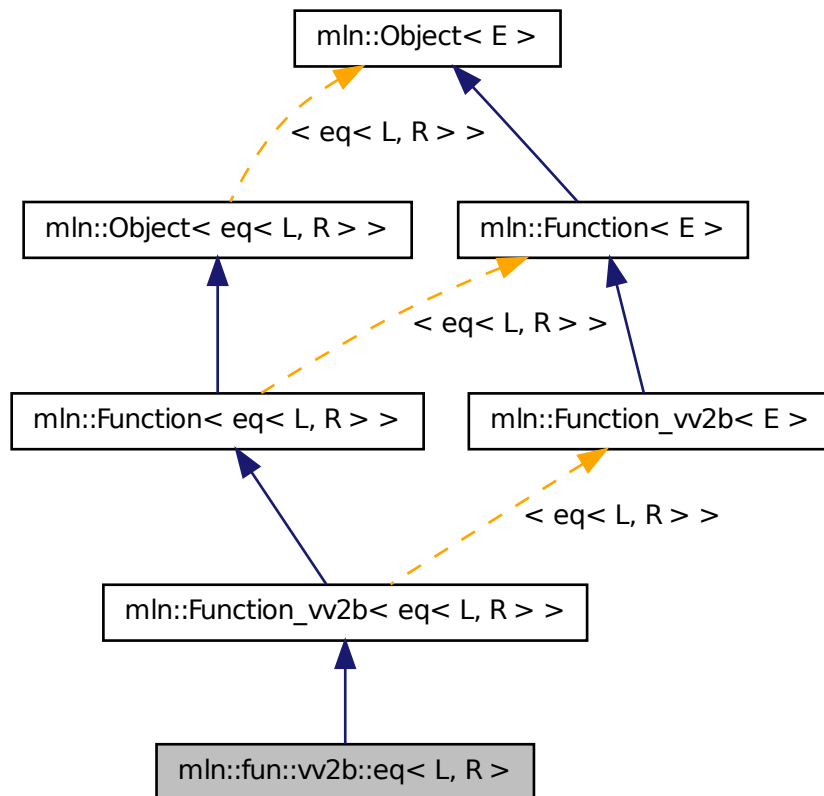
Definition at line 82 of file `v2w_w2v/norm.hh`.

## 10.156 mln::fun::vv2b::eq< L, R > Struct Template Reference

Functor computing equal between two values.

```
#include <eq.hh>
```

Inheritance diagram for mln::fun::vv2b::eq< L, R >:



### 10.156.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2b::eq< L, R >
```

Functor computing equal between two values.

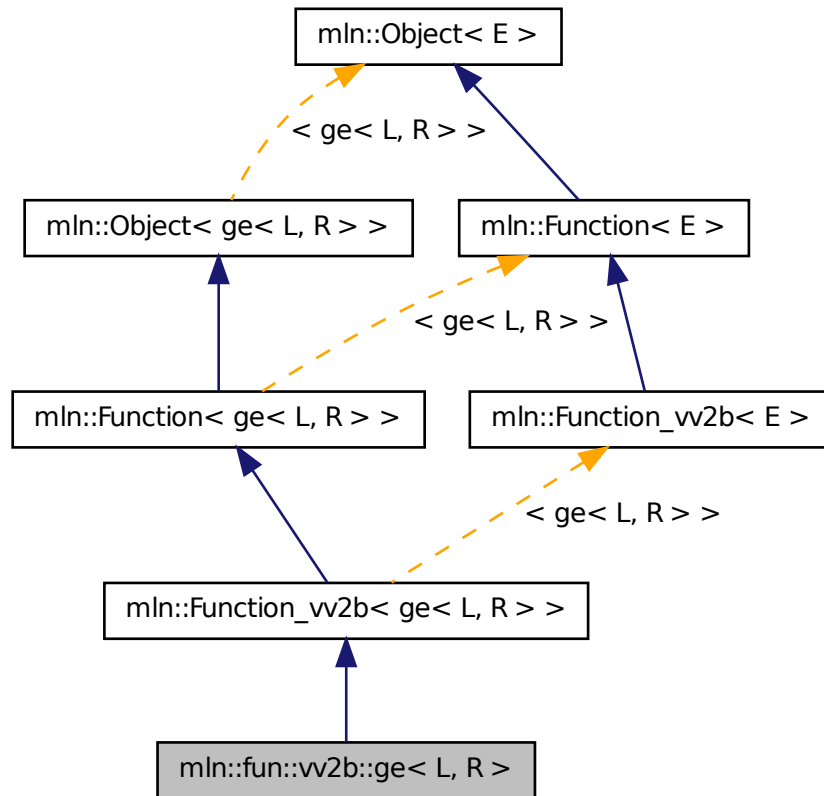
Definition at line 48 of file fun/vv2b/eq.hh.

## 10.157 mln::fun::vv2b::ge< L, R > Struct Template Reference

Functor computing "greater or equal than" between two values.

```
#include <ge.hh>
```

Inheritance diagram for `mln::fun::vv2b::ge< L, R >`:



### 10.157.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2b::ge< L, R >
```

Functor computing "greater or equal than" between two values.

Definition at line 48 of file `ge.hh`.

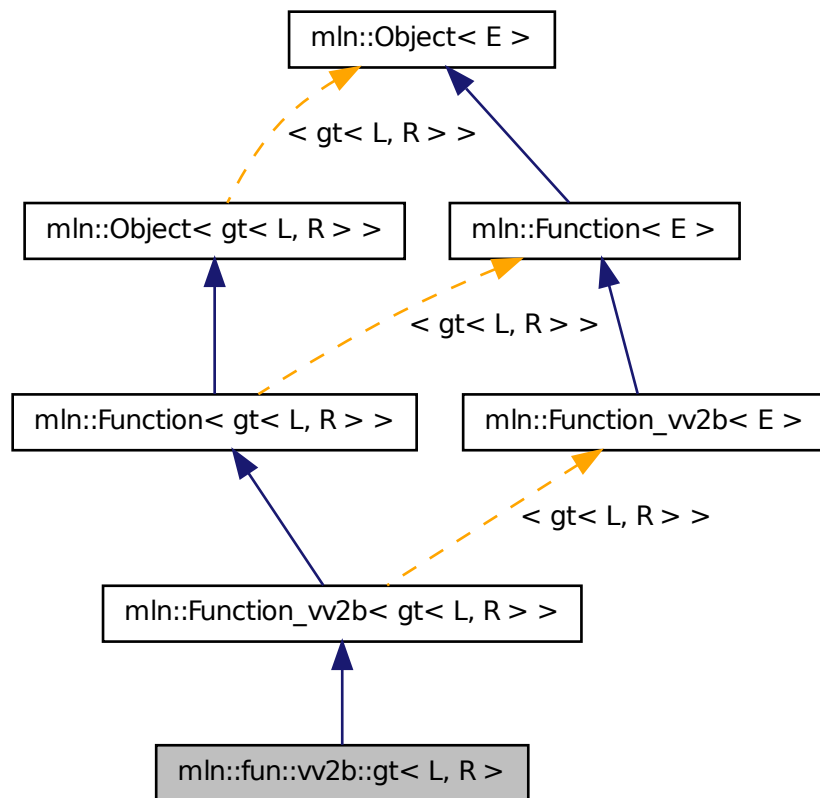
## 10.158 `mln::fun::vv2b::gt< L, R >` Struct Template Reference

Functor computing "greater than" between two values.

```
#include <gt.hh>
```



Inheritance diagram for mln::fun::vv2b::gt< L, R >:



### 10.158.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2b::gt< L, R >
```

Functor computing "greater than" between two values.

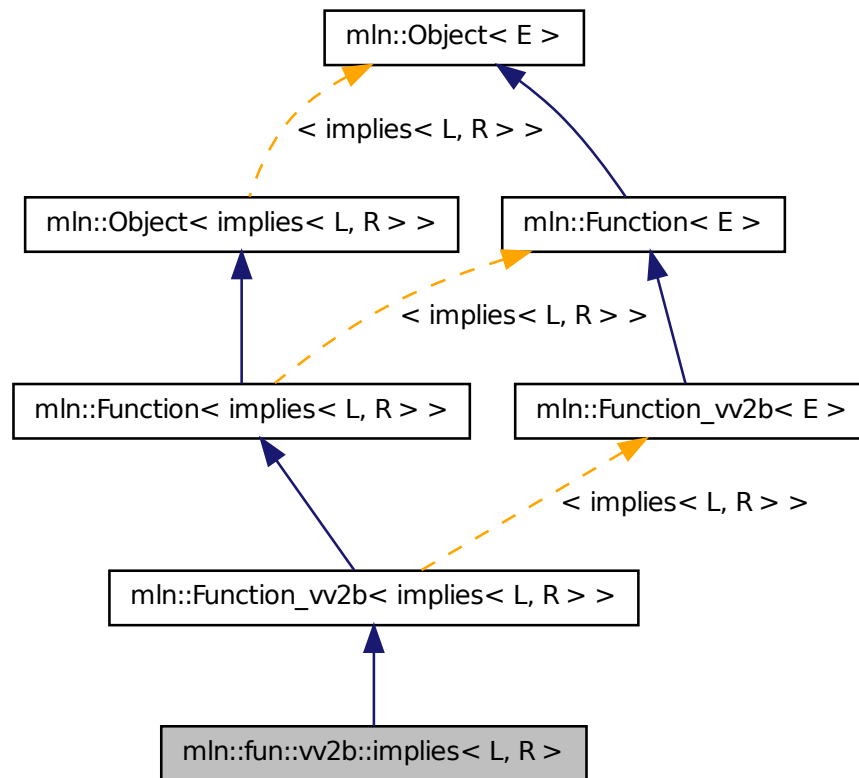
Definition at line 48 of file gt.hh.

## 10.159 mln::fun::vv2b::implies< L, R > Struct Template Reference

Functor computing logical-implies between two values.

```
#include <implies.hh>
```

Inheritance diagram for `mln::fun::vv2b::implies< L, R >`:



### 10.159.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2b::implies< L, R >
```

Functor computing logical-implies between two values.

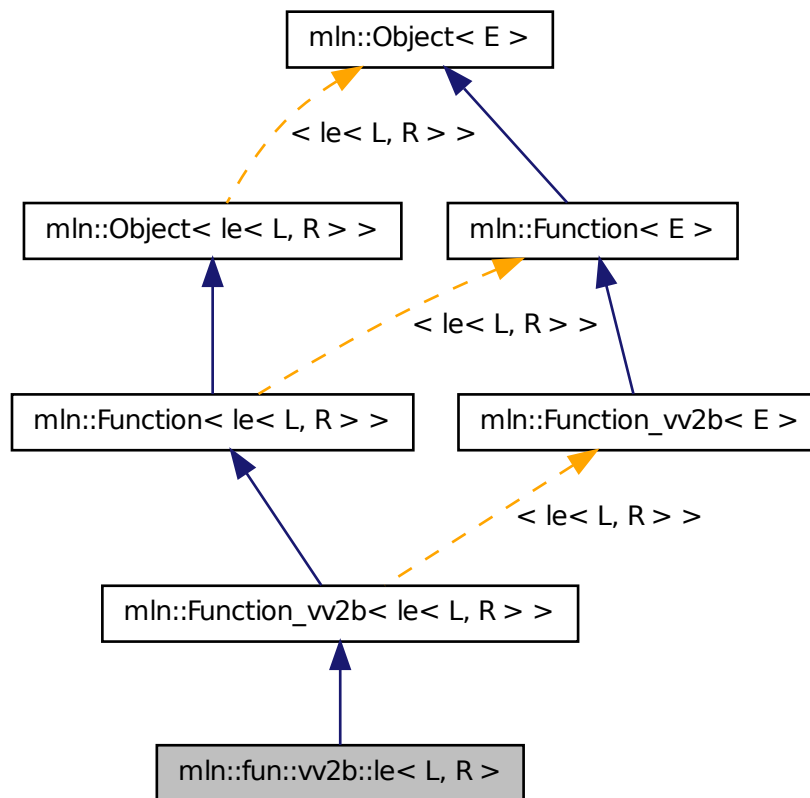
Definition at line 48 of file `implies.hh`.

### 10.160 `mln::fun::vv2b::le< L, R >` Struct Template Reference

Functor computing "lower or equal than" between two values.

```
#include <le.hh>
```

Inheritance diagram for mln::fun::vv2b::le< L, R >:



### 10.160.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2b::le< L, R >
```

Functor computing "lower or equal than" between two values.

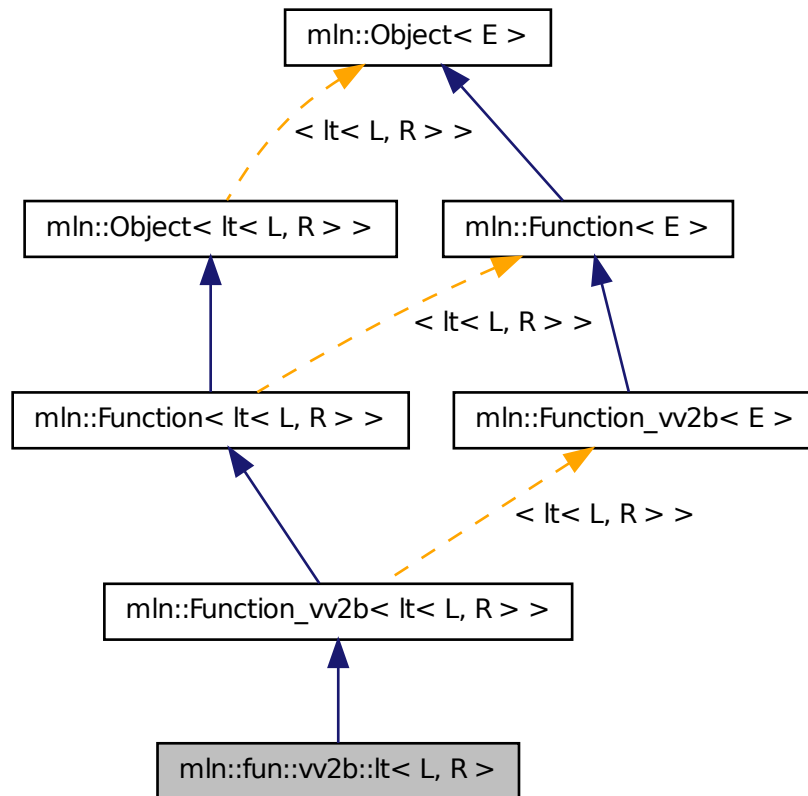
Definition at line 48 of file le.hh.

## 10.161 mln::fun::vv2b::lt< L, R > Struct Template Reference

Functor computing "lower than" between two values.

```
#include <lt.hh>
```

Inheritance diagram for `mln::fun::vv2b::lt< L, R >`:



### 10.161.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2b::lt< L, R >
```

Functor computing "lower than" between two values.

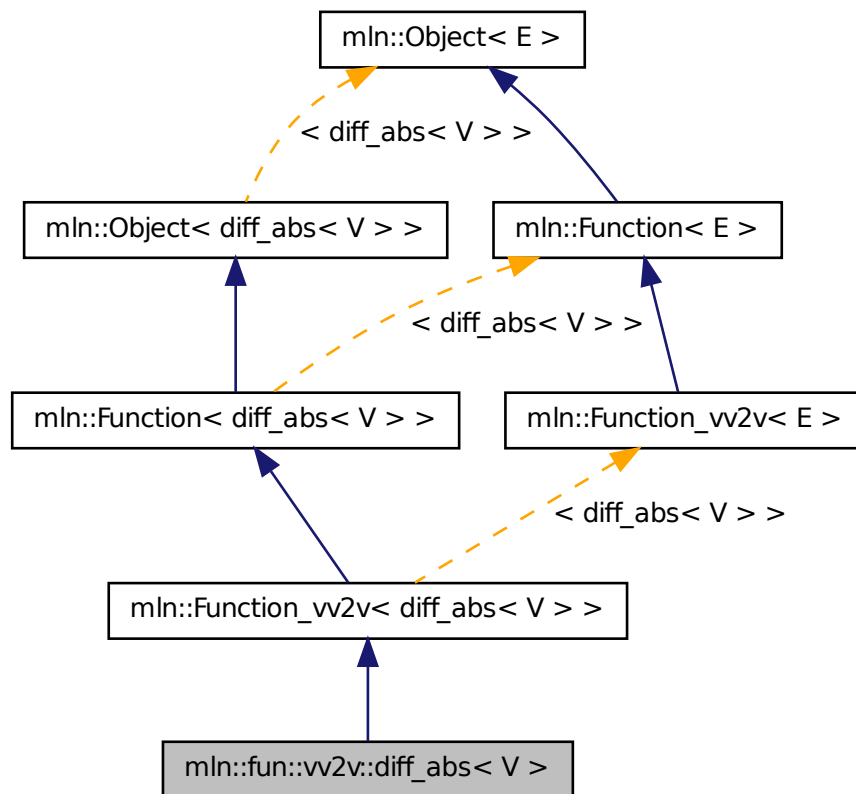
Definition at line 48 of file `lt.hh`.

## 10.162 `mln::fun::vv2v::diff_abs< V >` Struct Template Reference

A functor computing the `diff_abs`imum of two values.

```
#include <diff_abs.hh>
```

Inheritance diagram for mln::fun::vv2v::diff\_abs< V >:



### 10.162.1 Detailed Description

```
template<typename V> struct mln::fun::vv2v::diff_abs< V >
```

A functor computing the `diff_absimum` of two values.

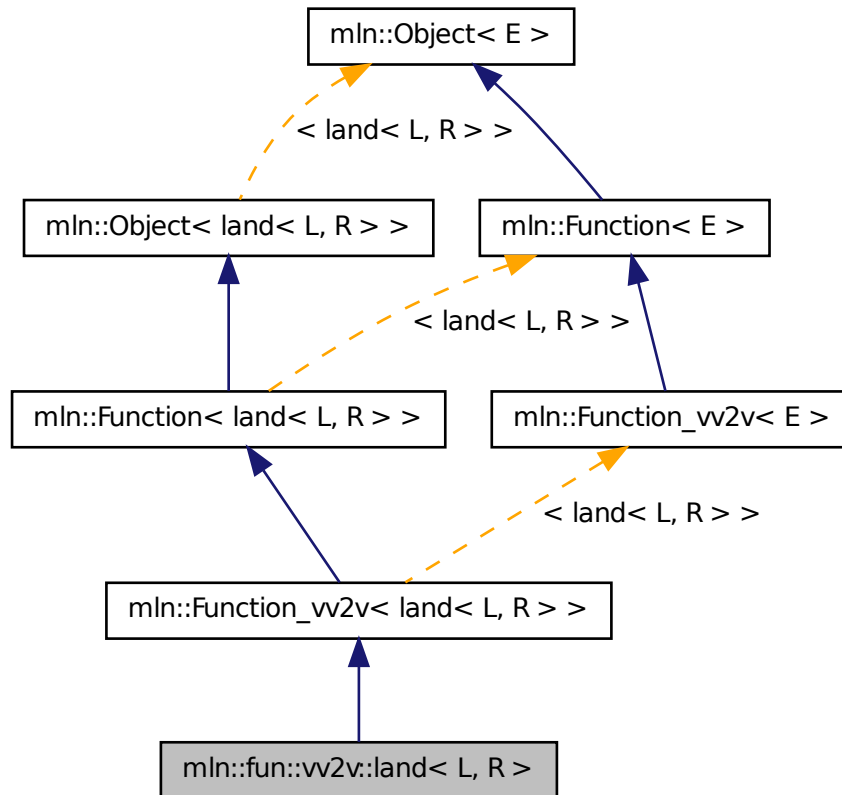
Definition at line 50 of file `fun/vv2v/diff_abs.hh`.

## 10.163 mln::fun::vv2v::land< L, R > Struct Template Reference

Functor computing logical-and between two values.

```
#include <land.hh>
```

Inheritance diagram for `mln::fun::vv2v::land< L, R >`:



### 10.163.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2v::land< L, R >
```

Functor computing logical-and between two values.

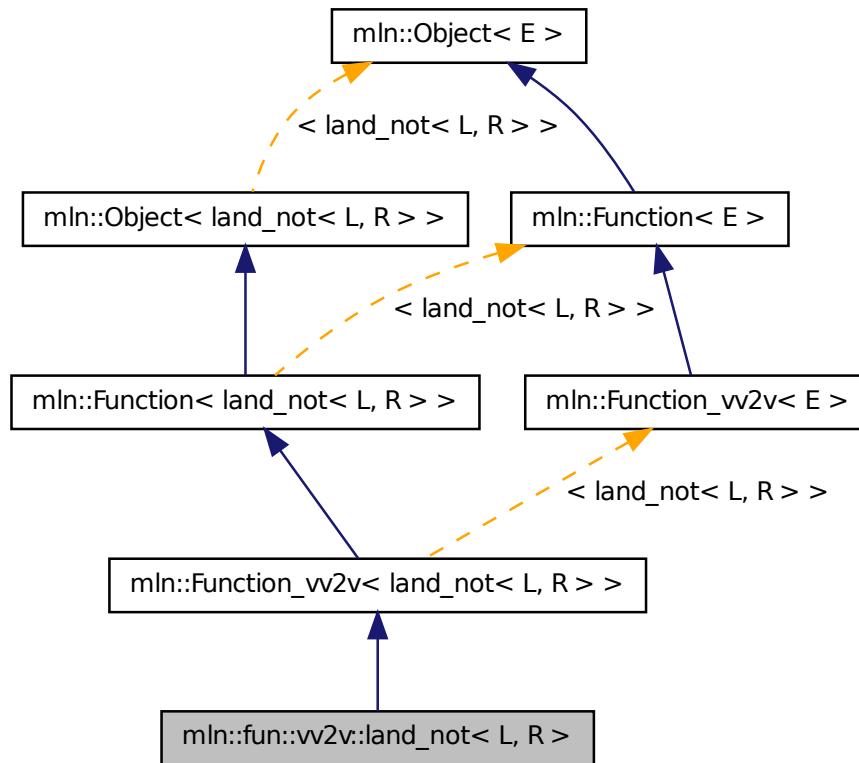
Definition at line 48 of file `fun/vv2v/land.hh`.

## 10.164 `mln::fun::vv2v::land_not< L, R >` Struct Template Reference

Functor computing logical and-not between two values.

```
#include <land_not.hh>
```

Inheritance diagram for mln::fun::vv2v::land\_not< L, R >:



### 10.164.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2v::land_not< L, R >
```

Functor computing logical and-not between two values.

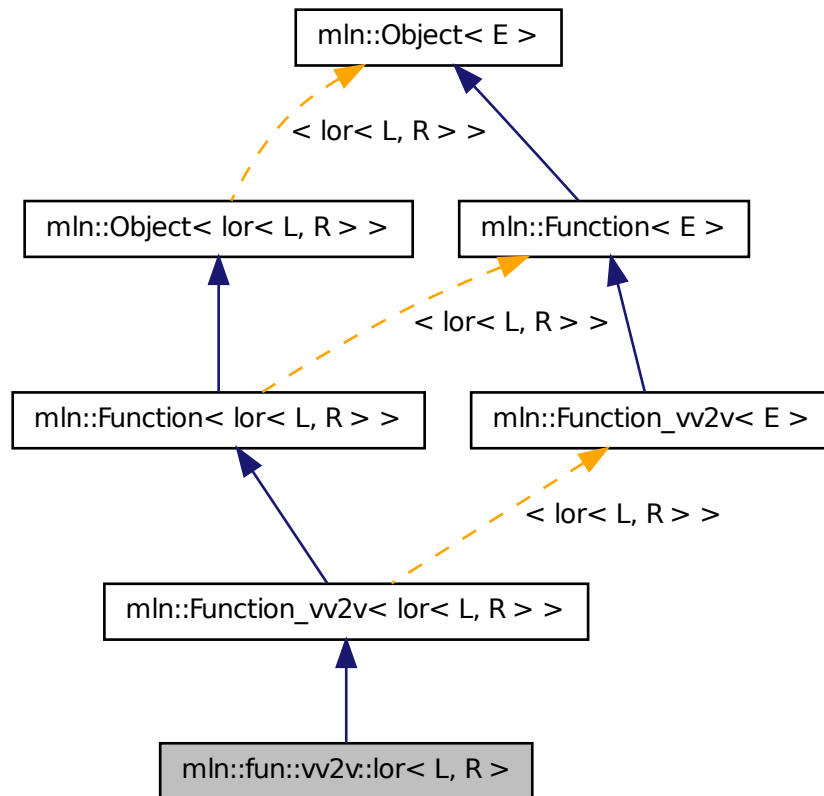
Definition at line 49 of file land\_not.hh.

## 10.165 mln::fun::vv2v::lor< L, R > Struct Template Reference

Functor computing logical-or between two values.

```
#include <lor.hh>
```

Inheritance diagram for `mln::fun::vv2v::lor< L, R >`:



### 10.165.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2v::lor< L, R >
```

Functor computing logical-or between two values.

Definition at line 48 of file `fun/vv2v/lor.hh`.

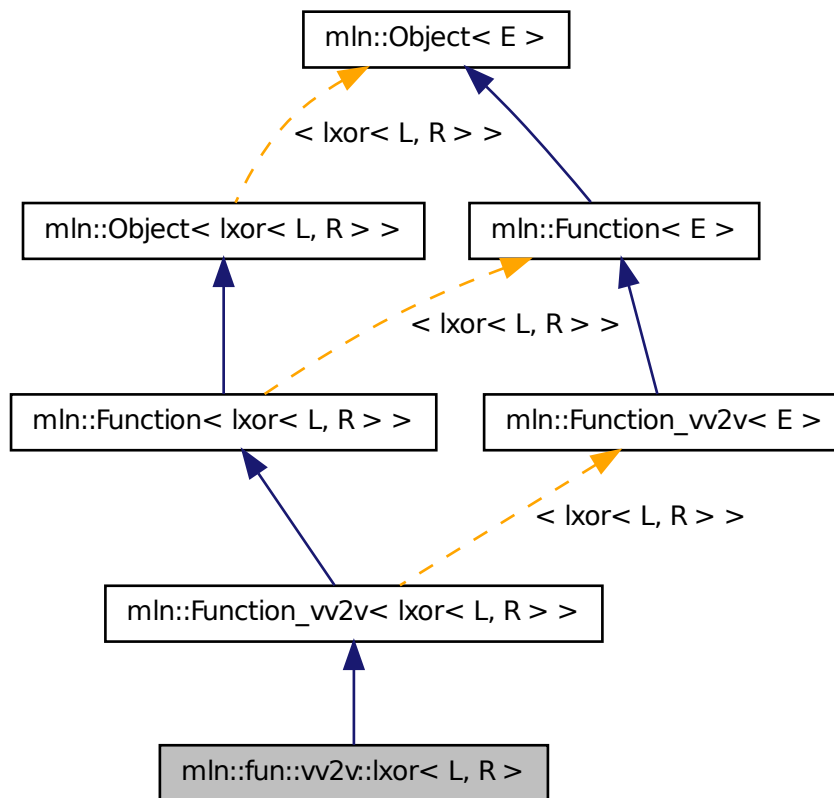
## 10.166 `mln::fun::vv2v::lxor< L, R >` Struct Template Reference

Functor computing logical-xor between two values.

```
#include <lxor.hh>
```



Inheritance diagram for mln::fun::vv2v::lxor< L, R >:



### 10.166.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2v::lxor< L, R >
```

Functor computing logical-xor between two values.

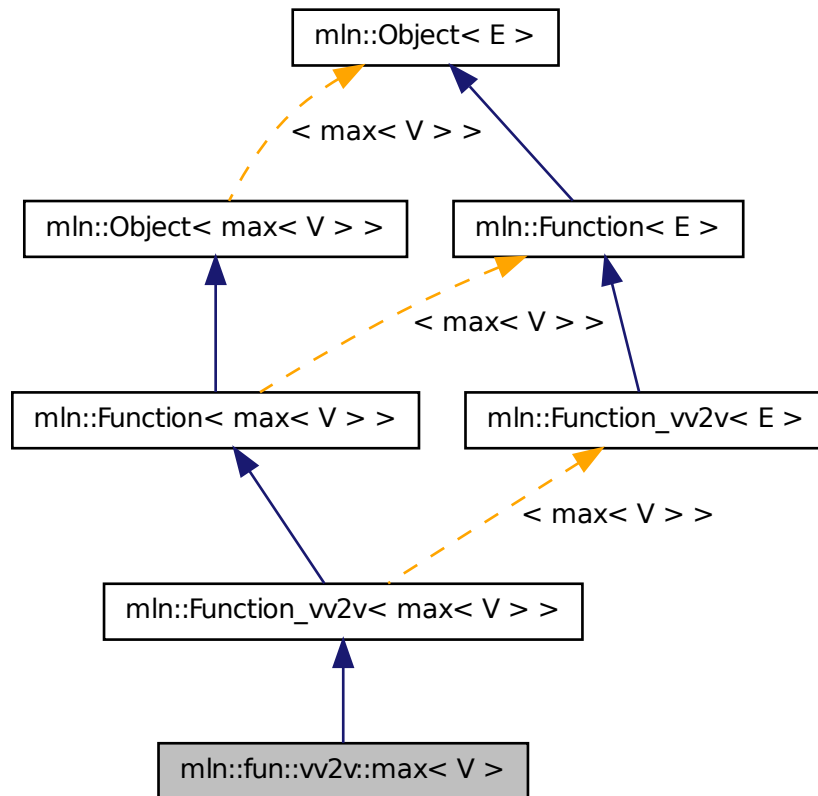
Definition at line 48 of file lxor.hh.

## 10.167 mln::fun::vv2v::max< V > Struct Template Reference

A functor computing the maximum of two values.

```
#include <max.hh>
```

Inheritance diagram for `mln::fun::vv2v::max< V >`:



### 10.167.1 Detailed Description

```
template<typename V> struct mln::fun::vv2v::max< V >
```

A functor computing the maximum of two values.

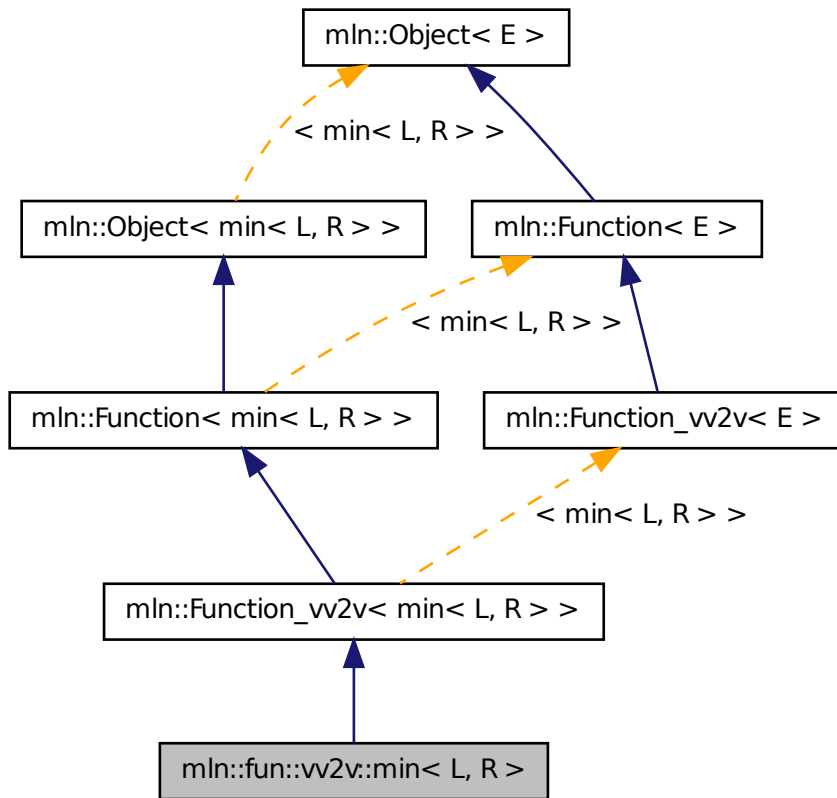
Definition at line 49 of file `fun/vv2v/max.hh`.

## 10.168 `mln::fun::vv2v::min< L, R >` Struct Template Reference

A functor computing the minimum of two values.

```
#include <min.hh>
```

Inheritance diagram for mln::fun::vv2v::min< L, R >:



### 10.168.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2v::min< L, R >
```

A functor computing the minimum of two values.

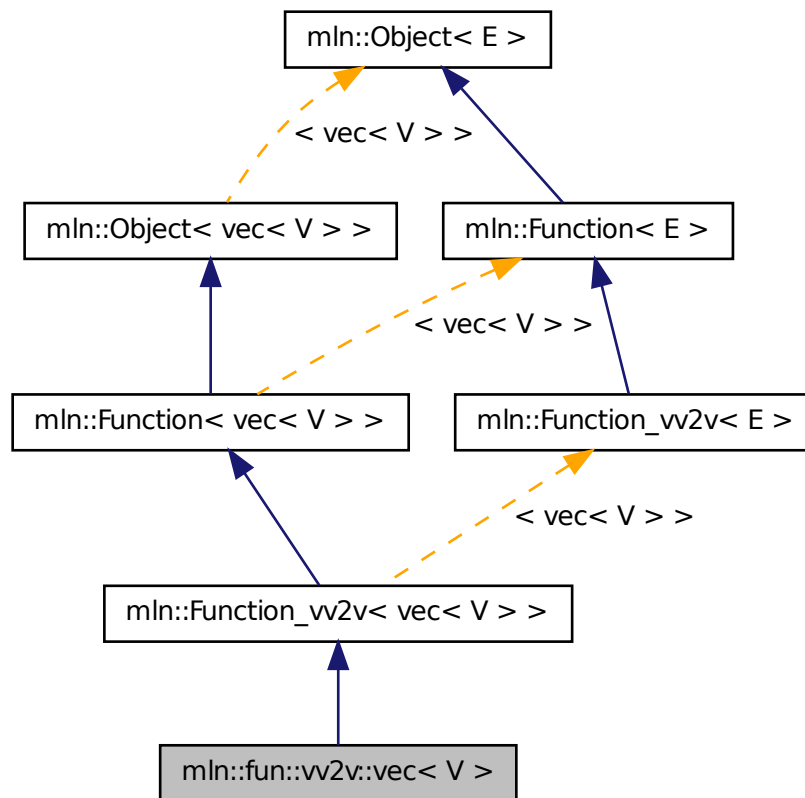
Definition at line 50 of file `fun/vv2v/min.hh`.

## 10.169 mln::fun::vv2v::vec< V > Struct Template Reference

A functor computing the vecimum of two values.

```
#include <vec.hh>
```

Inheritance diagram for `mln::fun::vv2v::vec< V >`:



### 10.169.1 Detailed Description

```
template<typename V> struct mln::fun::vv2v::vec< V >
```

A functor computing the vecimum of two values.

Definition at line 50 of file `fun/vv2v/vec.hh`.

## 10.170 `mln::fun::x2p::closest_point< P >` Struct Template Reference

FIXME: doxygen + concept checking.

```
#include <closest_point.hh>
```

### 10.170.1 Detailed Description

`template<typename P> struct mln::fun::x2p::closest_point< P >`

FIXME: doxygen + concept checking.

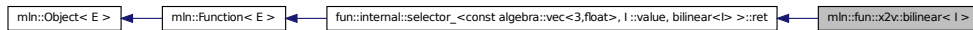
Definition at line 45 of file closest\_point.hh.

## 10.171 mln::fun::x2v::bilinear< I > Struct Template Reference

Represent a bilinear interolation of values from an underlying image.

```
#include <bilinear.hh>
```

Inheritance diagram for mln::fun::x2v::bilinear< I >:



### Public Member Functions

- `template<typename T >`  
`I::value operator() (const algebra::vec< 3, T > &v) const`  
*Bilinear filtering on 3d images. Work on slices.*
- `template<typename T >`  
`I::value operator() (const algebra::vec< 2, T > &v) const`  
*Bilinear filtering on 2d images.*

### 10.171.1 Detailed Description

`template<typename I> struct mln::fun::x2v::bilinear< I >`

Represent a bilinear interolation of values from an underlying image.

Definition at line 52 of file bilinear.hh.

### 10.171.2 Member Function Documentation

**10.171.2.1** `template<typename I > template<typename T > I::value mln::fun::x2v::bilinear< I >::operator() ( const algebra::vec< 2, T > & v ) const`

Bilinear filtering on 2d images.

Definition at line 86 of file bilinear.hh.

### 10.171.2.2 `template<typename I> template<typename T> I::value mln::fun::x2v::bilinear< I >::operator() ( const algebra::vec< 3, T > & v ) const`

Bilinear filtering on 3d images. Work on slices.

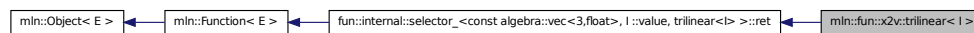
Definition at line 132 of file bilinear.hh.

## 10.172 `mln::fun::x2v::trilinear< I >` Struct Template Reference

Represent a trilinear interolation of values from an underlying image.

```
#include <trilinear.hh>
```

Inheritance diagram for `mln::fun::x2v::trilinear< I >`:



### 10.172.1 Detailed Description

```
template<typename I> struct mln::fun::x2v::trilinear< I >
```

Represent a trilinear interolation of values from an underlying image.

Definition at line 53 of file trilinear.hh.

## 10.173 `mln::fun::x2x::composed< T2, T1 >` Struct Template Reference

Represent a composition of two transformations.

```
#include <composed.hh>
```

### Public Member Functions

- `composed ()`  
*Constructor without argument.*
- `composed (const T2 &f, const T1 &g)`  
*Constructor with the two transformation to be composed.*

### 10.173.1 Detailed Description

```
template<typename T2, typename T1> struct mln::fun::x2x::composed< T2, T1 >
```

Represent a composition of two transformations.

Definition at line 144 of file composed.hh.

## 10.173.2 Constructor & Destructor Documentation

### 10.173.2.1 `template<typename T2, typename T1> mln::fun::x2x::composed< T2, T1 >::composed ( ) [inline]`

Constructor without argument.

Definition at line 153 of file composed.hh.

### 10.173.2.2 `template<typename T2, typename T1> mln::fun::x2x::composed< T2, T1 >::composed ( const T2 & f, const T1 & g ) [inline]`

Constructor with the two transformation to be composed.

Definition at line 156 of file composed.hh.

## 10.174 mln::fun::x2x::linear< I > Struct Template Reference

Represent a linear interolation of values from an underlying image.

```
#include <linear.hh>
```

Inheritance diagram for mln::fun::x2x::linear< I >:



## Public Member Functions

- `linear` (const I & *ima*)

*Constructor with the underlying image.*

- `template<typename C > I::value operator() (const algebra::vec< 1, C > &v) const`

*Return the interpolated value in the underlying image at the given 'point' v.*

## Public Attributes

- const I & *ima*

*Underlying image.*

### 10.174.1 Detailed Description

`template<typename I> struct mln::fun::x2x::linear< I >`

Represent a linear interpolation of values from an underlying image.

Definition at line 53 of file x2v/linear.hh.

### 10.174.2 Constructor & Destructor Documentation

**10.174.2.1** `template<typename I > mln::fun::x2x::linear< I >::linear ( const I & ima )`

Constructor with the underlying image.

Definition at line 77 of file x2v/linear.hh.

### 10.174.3 Member Function Documentation

**10.174.3.1** `template<typename I > template<typename C > I::value mln::fun::x2x::linear< I >::operator() ( const algebra::vec< 1, C > & v ) const`

Return the interpolated value in the underlying image at the given 'point' *v*.

Definition at line 85 of file x2v/linear.hh.

### 10.174.4 Member Data Documentation

**10.174.4.1** `template<typename I > const I& mln::fun::x2x::linear< I >::ima`

Underlying image.

Definition at line 70 of file x2v/linear.hh.

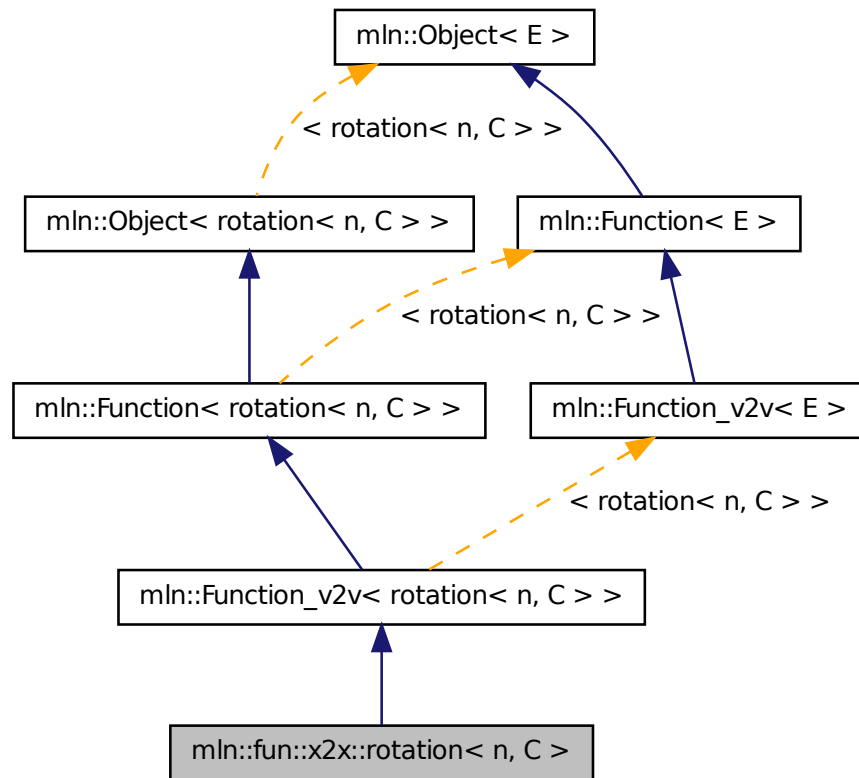
## 10.175 mln::fun::x2x::rotation< n, C > Struct Template Reference

Represent a rotation function.

```
#include <rotation.hh>
```



Inheritance diagram for mln::fun::x2x::rotation< n, C >:



## Public Types

- typedef C [data\\_t](#)  
*Type of the underlying data stored in vectors and matrices.*
- typedef [rotation< n, C >](#) [invert](#)  
*Type of the inverse function.*

## Public Member Functions

- [invert inv \(\)](#) const  
*Return the inverse function.*
- [algebra::vec< n, C > operator\(\)](#) (const algebra::vec< n, C > &v) const  
*Perform the rotation of the given vector.*

- `rotation ()`  
*Constructor without argument.*
- `rotation (const algebra::quat &q)`  
*Constructor with quaternion.*
- `rotation (const algebra::h_mat< n, C > &m)`  
*Constructor with h\_mat.*
- `rotation (C alpha, const algebra::vec< n, C > &axis)`  
*Constructor with radian alpha and a facultative direction (rotation axis).*
- `void set_alpha (C alpha)`  
*Set a new grade alpha.*
- `void set_axis (const algebra::vec< n, C > &axis)`  
*Set a new rotation axis.*

### 10.175.1 Detailed Description

`template<unsigned n, typename C> struct mln::fun::x2x::rotation< n, C >`

Represent a rotation function.

Definition at line 147 of file rotation.hh.

### 10.175.2 Member Typedef Documentation

**10.175.2.1** `template<unsigned n, typename C > typedef C mln::fun::x2x::rotation< n, C >::data_t`

Type of the underlying data stored in vectors and matrices.

Definition at line 152 of file rotation.hh.

**10.175.2.2** `template<unsigned n, typename C > typedef rotation<n,C> mln::fun::x2x::rotation< n, C >::invert`

Type of the inverse function.

Definition at line 155 of file rotation.hh.

### 10.175.3 Constructor & Destructor Documentation

**10.175.3.1** `template<unsigned n, typename C > mln::fun::x2x::rotation< n, C >::rotation ( )`  
`[inline]`

Constructor without argument.

Definition at line 192 of file rotation.hh.

**10.175.3.2** `template<unsigned n, typename C > mln::fun::x2x::rotation< n, C >::rotation ( C alpha, const algebra::vec< n, C > & axis ) [inline]`

Constructor with radian alpha and a facultative direction (rotation axis).

Definition at line 198 of file rotation.hh.

**10.175.3.3** `template<unsigned n, typename C > mln::fun::x2x::rotation< n, C >::rotation ( const algebra::quat & q ) [inline]`

Constructor with quaternion.

Definition at line 208 of file rotation.hh.

References mln::make::h\_mat().

**10.175.3.4** `template<unsigned n, typename C > mln::fun::x2x::rotation< n, C >::rotation ( const algebra::h_mat< n, C > & m ) [inline]`

Constructor with h\_mat.

Definition at line 238 of file rotation.hh.

## 10.175.4 Member Function Documentation

**10.175.4.1** `template<unsigned n, typename C > rotation< n, C > mln::fun::x2x::rotation< n, C >::inv ( ) const [inline]`

Return the inverse function.

Definition at line 265 of file rotation.hh.

**10.175.4.2** `template<unsigned n, typename C > algebra::vec< n, C > mln::fun::x2x::rotation< n, C >::operator() ( const algebra::vec< n, C > & v ) const [inline]`

Perform the rotation of the given vector.

Definition at line 247 of file rotation.hh.

**10.175.4.3** `template<unsigned n, typename C > void mln::fun::x2x::rotation< n, C >::set_alpha ( C alpha ) [inline]`

Set a new grade alpha.

Definition at line 274 of file rotation.hh.

**10.175.4.4** `template<unsigned n, typename C > void mln::fun::x2x::rotation< n, C >::set_axis ( const algebra::vec< n, C > & axis ) [inline]`

Set a new rotation axis.

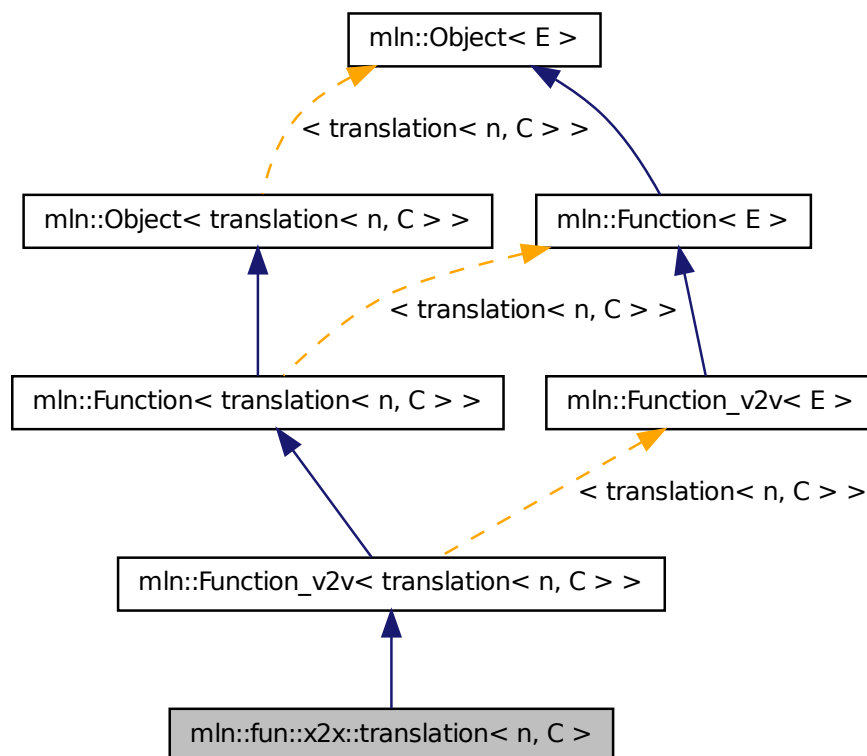
Definition at line 283 of file rotation.hh.

## 10.176 mln::fun::x2x::translation< n, C > Struct Template Reference

Translation function-object.

```
#include <translation.hh>
```

Inheritance diagram for mln::fun::x2x::translation< n, C >:



### Public Types

- typedef C `data_t`  
*Type of the underlying data stored in vectors and matrices.*
- typedef `translation< n, C >` `invert`  
*Type of the inverse function.*

### Public Member Functions

- `invert inv () const`

*Return the inverse function.*

- algebra::vec< n, C > [operator\(\)](#) (const algebra::vec< n, C > &v) const  
*Perform the translation of the given vector.*
- void [set\\_t](#) (const algebra::vec< n, C > &t)  
*Set a net translation vector.*
- const algebra::vec< n, C > & [t](#) () const  
*Return the translation vector.*
- [translation](#) (const algebra::vec< n, C > &t)  
*Constructor with the translation vector.*
- [translation](#) ()  
*Constructor without argument.*

### 10.176.1 Detailed Description

**template<unsigned n, typename C> struct mln::fun::x2x::translation< n, C >**

Translation function-object.

Definition at line 52 of file x2x/translation.hh.

### 10.176.2 Member Typedef Documentation

**10.176.2.1 template<unsigned n, typename C > typedef C mln::fun::x2x::translation< n, C >::data\_t**

Type of the underlying data stored in vectors and matrices.

Definition at line 59 of file x2x/translation.hh.

**10.176.2.2 template<unsigned n, typename C > typedef translation<n,C> mln::fun::x2x::translation< n, C >::invert**

Type of the inverse function.

Definition at line 62 of file x2x/translation.hh.

### 10.176.3 Constructor & Destructor Documentation

**10.176.3.1 template<unsigned n, typename C > mln::fun::x2x::translation< n, C >::translation ( ) [inline]**

Constructor without argument.

Definition at line 93 of file x2x/translation.hh.

**10.176.3.2** `template<unsigned n, typename C > mln::fun::x2x::translation< n, C >::translation ( const algebra::vec< n, C > & t ) [inline]`

Constructor with the translation vector.

Definition at line 99 of file x2x/translation.hh.

## 10.176.4 Member Function Documentation

**10.176.4.1** `template<unsigned n, typename C > translation< n, C > mln::fun::x2x::translation< n, C >::inv ( ) const [inline]`

Return the inverse function.

Definition at line 124 of file x2x/translation.hh.

**10.176.4.2** `template<unsigned n, typename C > algebra::vec< n, C > mln::fun::x2x::translation< n, C >::operator() ( const algebra::vec< n, C > & v ) const [inline]`

Perform the translation of the given vector.

Definition at line 108 of file x2x/translation.hh.

**10.176.4.3** `template<unsigned n, typename C > void mln::fun::x2x::translation< n, C >::set_t ( const algebra::vec< n, C > & t ) [inline]`

Set a net translation vector.

Definition at line 134 of file x2x/translation.hh.

**10.176.4.4** `template<unsigned n, typename C > const algebra::vec< n, C > & mln::fun::x2x::translation< n, C >::t ( ) const [inline]`

Return the translation vector.

Definition at line 143 of file x2x/translation.hh.

## 10.177 mln::fun\_image< F, I > Struct Template Reference

[Image](#) read through a function.

```
#include <fun_image.hh>
```

Inherits `image_value_morpher< I, F::result, fun_image< F, I > >`.

### Public Types

- typedef `F::result` [lvalue](#)  
*Return type of read-write access.*
- typedef `F::result` [rvalue](#)

*Return type of read-only access.*

- typedef [fun\\_image](#)< tag::value\_< typename F::result >, tag::image\_< I > > [skeleton](#)  
*Skeleton.*
- typedef F::result [value](#)  
*Value associated type.*

## Public Member Functions

- [fun\\_image](#) ()  
*Constructor.*
- [fun\\_image](#) (const [Function\\_v2v](#)< F > &f, const [Image](#)< I > &ima)  
*Constructor.*
- [fun\\_image](#) (const [Image](#)< I > &ima)  
*Constructor.*
- F::result [operator](#)() (const typename I::psite &p) const  
*Read-only access of pixel value at point site p.*
- F::result [operator](#)() (const typename I::psite &p)  
*Mutable access is for reading only.*

### 10.177.1 Detailed Description

**template<typename F, typename I> struct mln::fun\_image< F, I >**

[Image](#) read through a function.

Definition at line 101 of file fun\_image.hh.

### 10.177.2 Member Typedef Documentation

**10.177.2.1 template<typename F, typename I> typedef F ::result mln::fun\_image< F, I >::lvalue**

Return type of read-write access.

Definition at line 111 of file fun\_image.hh.

**10.177.2.2 template<typename F, typename I> typedef F ::result mln::fun\_image< F, I >::rvalue**

Return type of read-only access.

Definition at line 108 of file fun\_image.hh.

**10.177.2.3** `template<typename F, typename I> typedef fun_image< tag::value_<typename F>::result>, tag::image_<I> > mln::fun_image< F, I >::skeleton`

Skeleton.

Definition at line 115 of file fun\_image.hh.

**10.177.2.4** `template<typename F, typename I> typedef F::result mln::fun_image< F, I >::value`

[Value](#) associated type.

Definition at line 105 of file fun\_image.hh.

### 10.177.3 Constructor & Destructor Documentation

**10.177.3.1** `template<typename F , typename I > mln::fun_image< F, I >::fun_image ( ) [inline]`

Constructor.

Definition at line 177 of file fun\_image.hh.

**10.177.3.2** `template<typename F , typename I > mln::fun_image< F, I >::fun_image ( const Function_v2v< F > & f, const Image< I > & ima ) [inline]`

Constructor.

Definition at line 184 of file fun\_image.hh.

**10.177.3.3** `template<typename F , typename I > mln::fun_image< F, I >::fun_image ( const Image< I > & ima ) [inline]`

Constructor.

Definition at line 191 of file fun\_image.hh.

### 10.177.4 Member Function Documentation

**10.177.4.1** `template<typename F , typename I > F::result mln::fun_image< F, I >::operator() ( const typename I::psite & p ) const [inline]`

Read-only access of pixel value at point site p.

Definition at line 209 of file fun\_image.hh.

**10.177.4.2** `template<typename F , typename I > F::result mln::fun_image< F, I >::operator() ( const typename I::psite & p ) [inline]`

Mutable access is for reading only.

Definition at line 218 of file fun\_image.hh.



## 10.178 `mln::Function< E >` Struct Template Reference

Base class for implementation of function-objects.

```
#include <function.hh>
```

Inherits [mln::Object< E >](#).

Inherited by [mln::Function\\_n2v< E >](#), [mln::Function\\_v2v< E >](#), [mln::Function\\_vv2b< E >](#), and [mln::Function\\_vv2v< E >](#).

### Protected Member Functions

- [Function](#) ()

*An operator() has to be provided.*

#### 10.178.1 Detailed Description

```
template<typename E> struct mln::Function< E >
```

Base class for implementation of function-objects. The parameter *E* is the exact type.

Definition at line 64 of file `function.hh`.

#### 10.178.2 Constructor & Destructor Documentation

**10.178.2.1** `template<typename E > mln::Function< E >::Function ( ) [inline, protected]`

An operator() has to be provided.

Its signature depends on the particular function-object one considers.

Definition at line 219 of file `function.hh`.

## 10.179 `mln::Function< void >` Struct Template Reference

[Function](#) category flag type.

```
#include <function.hh>
```

### 10.179.1 Detailed Description

```
template<> struct mln::Function< void >
```

[Function](#) category flag type.

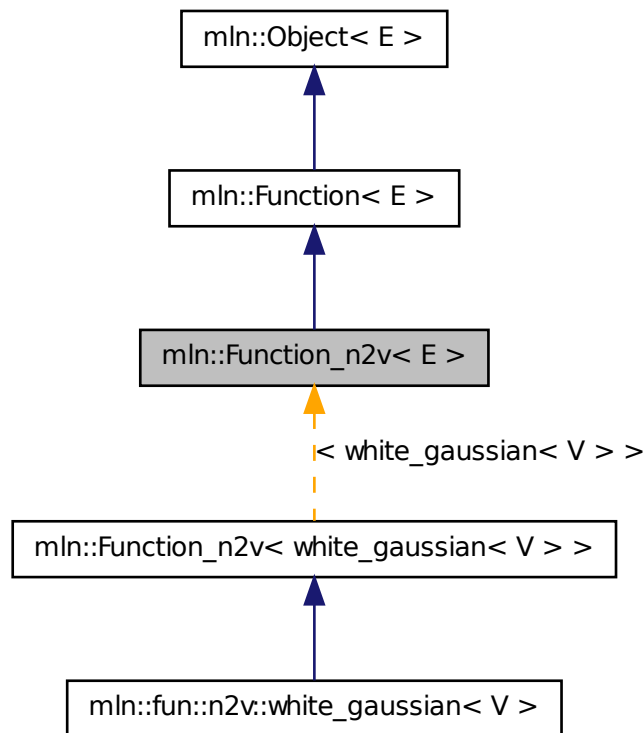
Definition at line 51 of file `function.hh`.

## 10.180 mln::Function\_n2v< E > Struct Template Reference

Base class for implementation of function-objects from Nil to value.

```
#include <function.hh>
```

Inheritance diagram for mln::Function\_n2v< E >:



### 10.180.1 Detailed Description

```
template<typename E> struct mln::Function_n2v< E >
```

Base class for implementation of function-objects from Nil to value. The parameter *E* is the exact type.

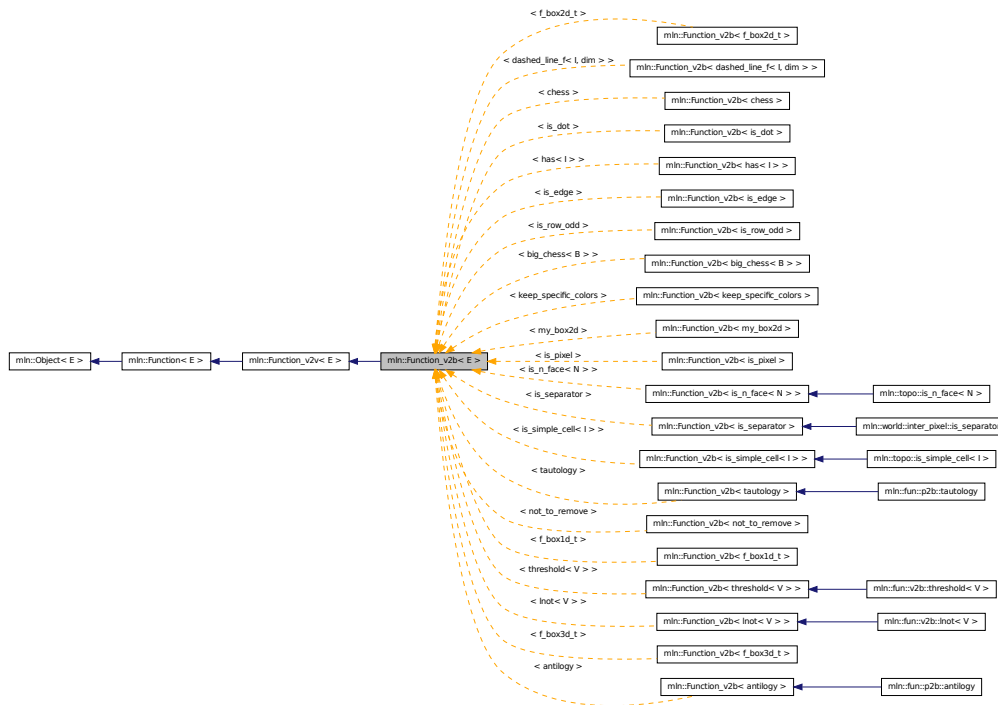
Definition at line 99 of file `function.hh`.

## 10.181 mln::Function\_v2b< E > Struct Template Reference

Base class for implementation of function-objects from a value to a Boolean.

```
#include <function.hh>
```

Inheritance diagram for mln::Function\_v2b< E >:



### 10.181.1 Detailed Description

`template<typename E> struct mln::Function_v2b< E >`

Base class for implementation of function-objects from a value to a Boolean. The parameter *E* is the exact type.

Definition at line 150 of file function.hh.

## 10.182 mln::Function\_v2v< E > Struct Template Reference

Base class for implementation of function-objects from value to value.

```
#include <function.hh>
```

Inherits [mln::Function< E >](#).

Inherited by [mln::fun::C< R\(\\*\) \(A\) >](#), [mln::fun::v2v::dec< T >](#), [mln::fun::v2v::id< T >](#), [mln::fun::v2v::inc< T >](#), [mln::fun::x2v::bilinear< I >](#), [mln::fun::x2v::trilinear< I >](#), [mln::fun::x2x::internal::helper\\_composed< T2, T1, E, false >](#), [mln::fun::x2x::internal::helper\\_composed< T2, T1, E, true >](#), [mln::fun::x2x::linear< I >](#), [mln::fun::x2x::neighbor< I >](#), and [mln::Function\\_v2b< E >](#) [virtual].

### 10.182.1 Detailed Description

```
template<typename E> struct mln::Function_v2v< E >
```

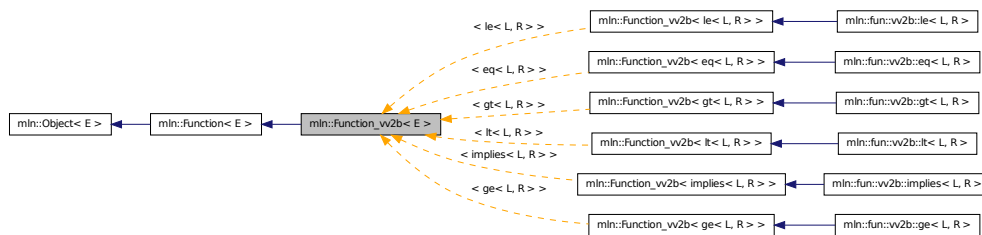
Base class for implementation of function-objects from value to value. The parameter *E* is the exact type. Definition at line 124 of file function.hh.

### 10.183 mln::Function\_vv2b< E > Struct Template Reference

Base class for implementation of function-objects from a couple of values to a Boolean.

```
#include <function.hh>
```

Inheritance diagram for mln::Function\_vv2b< E >:



### 10.183.1 Detailed Description

```
template<typename E> struct mln::Function_vv2b< E >
```

Base class for implementation of function-objects from a couple of values to a Boolean. The parameter *E* is the exact type.

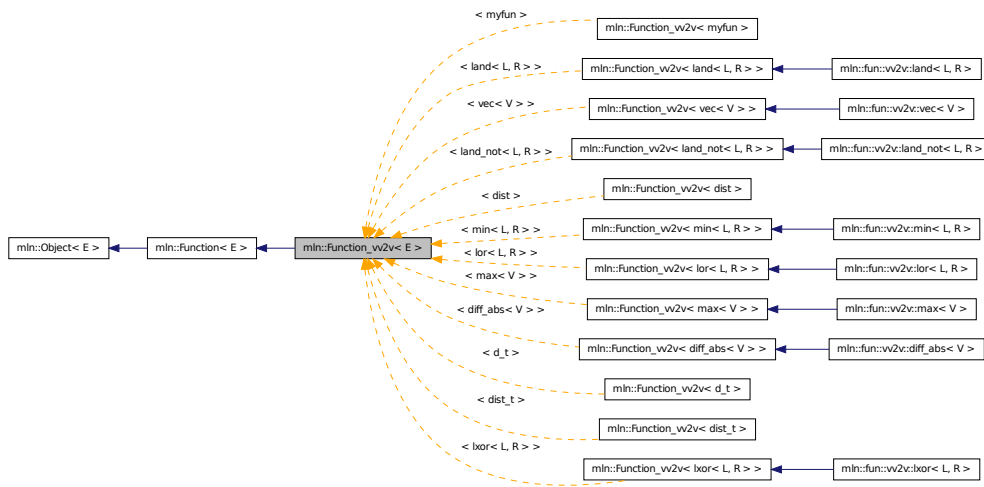
Definition at line 202 of file function.hh.

### 10.184 mln::Function\_vv2v< E > Struct Template Reference

Base class for implementation of function-objects from a couple of values to a value.

```
#include <function.hh>
```

Inheritance diagram for mln::Function\_vv2v< E >:



### 10.184.1 Detailed Description

```
template<typename E> struct mln::Function_vv2v< E >
```

Base class for implementation of function-objects from a couple of values to a value. The parameter *E* is the exact type.

Definition at line 177 of file function.hh.

## 10.185 mln::fwd\_pixter1d< I > Class Template Reference

Forward pixel iterator on a 1-D image with border.

```
#include <pixter1d.hh>
```

Inherits forward\_pixel\_iterator\_base\_< I, fwd\_pixter1d< I > >.

### Public Types

- typedef [I image](#)  
*Image type.*

### Public Member Functions

- [fwd\\_pixter1d](#) (I &image)  
*Constructor.*
- void [next](#) ()

*Go to the next element.*

### 10.185.1 Detailed Description

`template<typename I> class mln::fwd_pixter1d< I >`

Forward pixel iterator on a 1-D image with border.

Definition at line 45 of file `pixter1d.hh`.

### 10.185.2 Member Typedef Documentation

**10.185.2.1** `template<typename I > typedef I mln::fwd_pixter1d< I >::image`

[Image](#) type.

Definition at line 52 of file `pixter1d.hh`.

### 10.185.3 Constructor & Destructor Documentation

**10.185.3.1** `template<typename I > mln::fwd_pixter1d< I >::fwd_pixter1d ( I & image )`  
[`inline`]

Constructor.

#### Parameters

[`in`] *image* The image this pixel iterator is bound to.

Definition at line 96 of file `pixter1d.hh`.

### 10.185.4 Member Function Documentation

**10.185.4.1** `void mln::Iterator< fwd_pixter1d< I > >::next ( )` [`inherited`]

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

#### Precondition

The iterator is valid.

## 10.186 `mln::fwd_pixter2d< I >` Class Template Reference

Forward pixel iterator on a 2-D image with border.

```
#include <pixter2d.hh>
```

Inherits `forward_pixel_iterator_base_< I, fwd_pixter2d< I > >`.

## Public Types

- typedef `I` `image`  
*Image type.*

## Public Member Functions

- `fwd_pixter2d` (`I &image`)  
*Constructor.*
- void `next` ()  
*Go to the next element.*

### 10.186.1 Detailed Description

```
template<typename I> class mln::fwd_pixter2d< I >
```

Forward pixel iterator on a 2-D image with border.

Definition at line 47 of file `pixter2d.hh`.

### 10.186.2 Member Typedef Documentation

**10.186.2.1** `template<typename I > typedef I mln::fwd_pixter2d< I >::image`

`Image` type.

Definition at line 54 of file `pixter2d.hh`.

### 10.186.3 Constructor & Destructor Documentation

**10.186.3.1** `template<typename I > mln::fwd_pixter2d< I >::fwd_pixter2d ( I & image )`  
`[inline]`

Constructor.

#### Parameters

`[in]` `image` The image this pixel iterator is bound to.

Definition at line 130 of file `pixter2d.hh`.

## 10.186.4 Member Function Documentation

### 10.186.4.1 void mln::Iterator< fwd\_pixter2d< I > >::next ( ) [inherited]

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

#### Precondition

The iterator is valid.

## 10.187 mln::fwd\_pixter3d< I > Class Template Reference

Forward pixel iterator on a 3-D image with border.

```
#include <pixter3d.hh>
```

Inherits forward\_pixel\_iterator\_base\_< I, fwd\_pixter3d< I > >.

### Public Types

- typedef I [image](#)  
*Image type.*

### Public Member Functions

- [fwd\\_pixter3d](#) (I &image)  
*Constructor.*
- void [next](#) ()  
*Go to the next element.*

### 10.187.1 Detailed Description

```
template<typename I> class mln::fwd_pixter3d< I >
```

Forward pixel iterator on a 3-D image with border.

Definition at line 48 of file pixter3d.hh.

### 10.187.2 Member Typedef Documentation

#### 10.187.2.1 template<typename I > typedef I mln::fwd\_pixter3d< I >::image

[Image](#) type.



Definition at line 55 of file pixter3d.hh.

### 10.187.3 Constructor & Destructor Documentation

**10.187.3.1** `template<typename I > mln::fwd_pixter3d< I >::fwd_pixter3d ( I & image )`  
[inline]

Constructor.

#### Parameters

[in] *image* The image this pixel iterator is bound to.

Definition at line 154 of file pixter3d.hh.

### 10.187.4 Member Function Documentation

**10.187.4.1** `void mln::Iterator< fwd_pixter3d< I > >::next ( )` [inherited]

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

#### Precondition

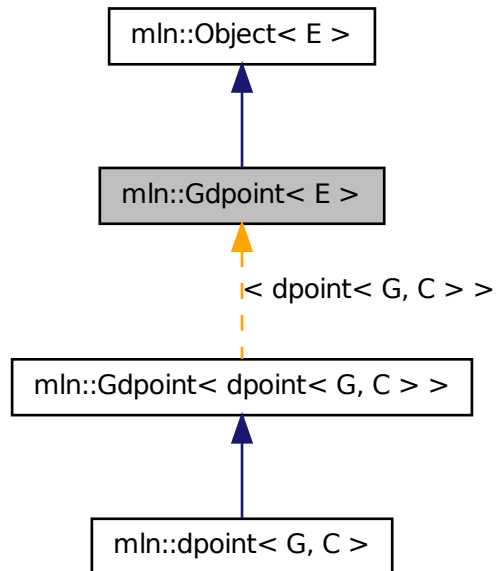
The iterator is valid.

## 10.188 mln::Gdpoint< E > Struct Template Reference

FIXME: Doc!

```
#include <gdpoint.hh>
```

Inheritance diagram for `mln::Gdpoint< E >`:



### 10.188.1 Detailed Description

```
template<typename E> struct mln::Gdpoint< E >
```

FIXME: Doc!

Definition at line 95 of file `gdpoint.hh`.

## 10.189 mln::Gdpoint< void > Struct Template Reference

Delta point site category flag type.

```
#include <gdpoint.hh>
```

### 10.189.1 Detailed Description

```
template<> struct mln::Gdpoint< void >
```

Delta point site category flag type.

Definition at line 87 of file `gdpoint.hh`.

## 10.190 mln::geom::complex\_geometry< D, P > Class Template Reference

A functor returning the sites of the faces of a complex where the locations of each 0-face is stored.

```
#include <complex_geometry.hh>
```

### Public Member Functions

- unsigned [add\\_location](#) (const P &p)  
*Populate the set of locations.*
- [complex\\_geometry](#) ()  
*Build a complex geometry object.*
- site [operator\(\)](#) (const [mln::topo::face](#)< D > &f) const  
*Retrieve the site associated to f.*

### 10.190.1 Detailed Description

```
template<unsigned D, typename P> class mln::geom::complex_geometry< D, P >
```

A functor returning the sites of the faces of a complex where the locations of each 0-face is stored. Faces of higher dimensions are computed.

#### Template Parameters

- D* The dimension of the complex.
- P* The type of the location of a 0-face.

Locations of 0-face are usually points (hence the *P* above), but can possibly be any (default-constructible) values.

The functor returns a `std::vector` of locations: 0-faces are singletons, 1-faces are (usually) pairs, faces of higher dimensions are arrays of locations.

Note that for consistency reasons w.r.t. the return type of `operator()`, returned sites are always *arrays* of locations attached to 0-faces; hence the returned singletons (of locations) for 0-faces.

Definition at line 88 of file `geom/complex_geometry.hh`.

### 10.190.2 Constructor & Destructor Documentation

```
10.190.2.1 template<unsigned D, typename P > mln::geom::complex_geometry< D, P  
>::complex_geometry ( ) [inline]
```

Build a complex geometry object.

Definition at line 132 of file `geom/complex_geometry.hh`.

### 10.190.3 Member Function Documentation

**10.190.3.1** `template<unsigned D, typename P > unsigned mln::geom::complex_geometry< D, P >::add_location ( const P & p ) [inline]`

Populate the set of locations.

Append a new location *p*. Return the index of the newly created location (which should semantically match the id of the corresponding 0-face in the complex).

Definition at line 140 of file geom/complex\_geometry.hh.

**10.190.3.2** `template<unsigned D, typename P > util::multi_site< P > mln::geom::complex_geometry< D, P >::operator() ( const mln::topo::face< D > & f ) const [inline]`

Retrieve the site associated to *f*.

Definition at line 151 of file geom/complex\_geometry.hh.

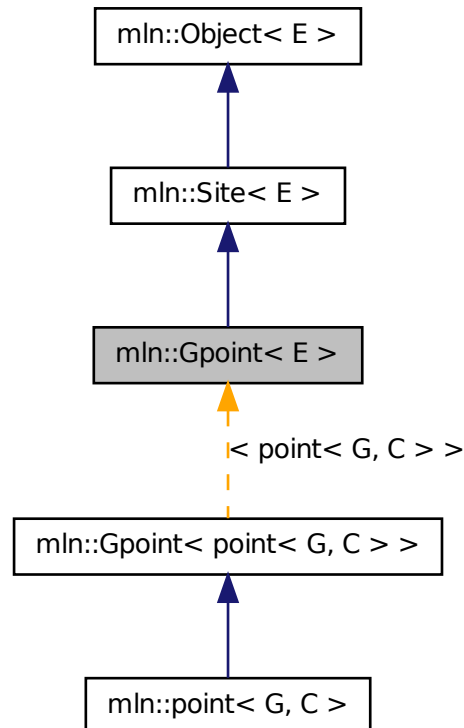
References `mln::topo::face< D >::face_id()`, and `mln::topo::face< D >::n()`.

## 10.191 mln::Gpoint< E > Struct Template Reference

Base class for implementation of point classes.

```
#include <gpoint.hh>
```

Inheritance diagram for mln::Gpoint< E >:



## Related Functions

(Note that these are not member functions.)

- `template<typename P, typename D >`  
`P & operator+ (const Gpoint< P > &p, const Gdpoint< D > &dp)`  
*Add a delta-point rhs to a grid point lhs.*
- `template<typename P, typename D >`  
`P & operator+= (Gpoint< P > &p, const Gdpoint< D > &dp)`  
*Shift a point by a delta-point dp.*
- `template<typename L, typename R >`  
`L::delta operator- (const Gpoint< L > &lhs, const Gpoint< R > &rhs)`  
*Difference between a couple of grid point lhs and rhs.*
- `template<typename P, typename D >`  
`P & operator-= (Gpoint< P > &p, const Gdpoint< D > &dp)`

*Shift a point by the negate of a delta-point  $dp$ .*

- `template<typename P, typename D >`  
`P operator/ (const Gpoint< P > &p, const value::scalar_< D > &dp)`  
*Divise a point by a scalar  $s$ .*
- `template<typename P >`  
`std::ostream & operator<< (std::ostream &ostr, const Gpoint< P > &p)`  
*Print a grid point  $p$  into the output stream  $ostr$ .*
- `template<typename L, typename R >`  
`bool operator== (const Gpoint< L > &lhs, const Gpoint< R > &rhs)`  
*Equality comparison between a couple of grid point  $lhs$  and  $rhs$ .*

### 10.191.1 Detailed Description

`template<typename E> struct mln::Gpoint< E >`

Base class for implementation of point classes. A point is an element of a space.

For instance, `mln::point2d` is the type of elements defined on the discrete square grid of the 2D plane.

Definition at line 115 of file `gpoint.hh`.

### 10.191.2 Friends And Related Function Documentation

**10.191.2.1** `template<typename P, typename D > P operator+ ( const Gpoint< P > & p, const Gdpoint< D > & dp ) [related]`

Add a delta-point `rhs` to a grid point `lhs`.

#### Parameters

[in] `p` A grid point.

[in] `dp` A delta-point.

The type of `dp` has to compatible with the type of `p`.

#### Returns

A point (temporary object).

#### See also

[mln::Gdpoint](#)

Definition at line 385 of file `gpoint.hh`.

**10.191.2.2** `template<typename P , typename D > P & operator+=( Gpoint< P > & p, const Gdpoint< D > & dp ) [related]`

Shift a point by a delta-point `dp`.

**Parameters**

[in, out] *p* The targeted point.

[in] *dp* A delta-point.

**Returns**

A reference to the point `p` once translated by `dp`.

**Precondition**

The type of `dp` has to be compatible with the type of `p`.

Definition at line 428 of file `gpoint.hh`.

**10.191.2.3** `template<typename L , typename R > L::delta operator-( const Gpoint< L > & lhs, const Gpoint< R > & rhs ) [related]`

Difference between a couple of grid point `lhs` and `rhs`.

**Parameters**

[in] *lhs* A first grid point.

[in] *rhs* A second grid point.

**Warning**

There is no type promotion in Milena so the client has to make sure that both points are defined with the same type of coordinates.

**Precondition**

Both `lhs` and `rhs` have to be defined on the same topology and with the same type of coordinates; otherwise this test does not compile.

**Postcondition**

The result, `dp`, is such as `lhs == rhs + dp`.

**Returns**

A delta point (temporary object).

**See also**

[mln::Gdpoint](#)

Definition at line 374 of file `gpoint.hh`.

**10.191.2.4** `template<typename P, typename D> P & operator==( Gpoint< P > & p, const Gdpoint< D > & dp )` [**related**]

Shift a point by the negate of a delta-point `dp`.

**Parameters**

[in, out] *p* The targeted point.

[in] *dp* A delta-point.

**Returns**

A reference to the point `p` once translated by `- dp`.

**Precondition**

The type of `dp` has to be compatible with the type of `p`.

Definition at line 436 of file `gpoint.hh`.

**10.191.2.5** `template<typename P, typename D> P operator/( const Gpoint< P > & p, const value::scalar_< D > & dp )` [**related**]

Divide a point by a scalar `s`.

**Parameters**

[in, out] *p* The targeted point.

[in] *dp* A scalar.

**Returns**

A reference to the point `p` once divided by `s`.

**10.191.2.6** `template<typename P> std::ostream & operator<<( std::ostream & ostr, const Gpoint< P > & p )` [**related**]

Print a grid point `p` into the output stream `ostr`.

**Parameters**

[in, out] *ostr* An output stream.

[in] *p* A grid point.

**Returns**

The modified output stream `ostr`.

Definition at line 417 of file `gpoint.hh`.

References `mln::debug::format()`.



**10.191.2.7** `template<typename L , typename R > bool operator==( const Gpoint< L > & lhs, const Gpoint< R > & rhs )` [related]

Equality comparison between a couple of grid point `lhs` and `rhs`.

#### Parameters

- [in] *lhs* A first grid point.
- [in] *rhs* A second grid point.

#### Precondition

Both `lhs` and `rhs` have to be defined on the same topology; otherwise this test does not compile.

#### Returns

True if both grid points have the same coordinates, otherwise false.

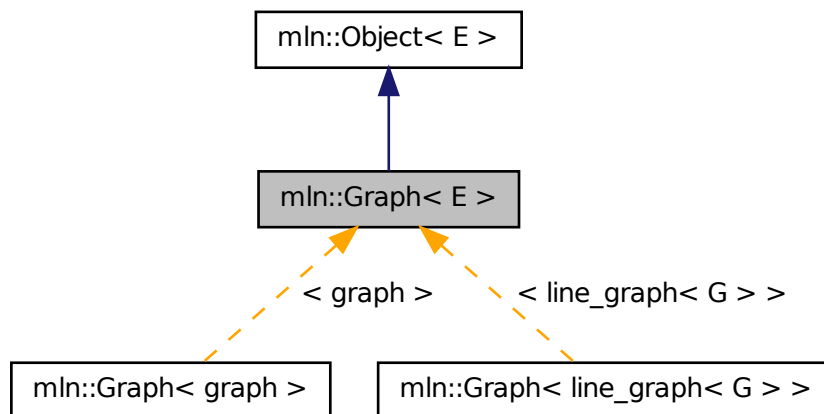
Definition at line 365 of file `gpoint.hh`.

## 10.192 mln::Graph< E > Struct Template Reference

Base class for implementation of graph classes.

```
#include <graph.hh>
```

Inheritance diagram for `mln::Graph< E >`:



### 10.192.1 Detailed Description

```
template<typename E> struct mln::Graph< E >
```

Base class for implementation of graph classes.

**See also**

`mln::doc::Graph` for a complete documentation of this class contents.

Definition at line 57 of file `mln/core/concept/graph.hh`.

## 10.193 `mln::graph::attribute::card_t` Struct Reference

Compute the cardinality of every component in a graph.

```
#include <card.hh>
```

### Public Types

- typedef `util::array< unsigned > result`

*Type of the computed value.*

#### 10.193.1 Detailed Description

Compute the cardinality of every component in a graph.

#### Returns

An array with the cardinality for each component. Components are labeled from 0.

Definition at line 61 of file `graph/attribute/card.hh`.

#### 10.193.2 Member Typedef Documentation

##### 10.193.2.1 `typedef util::array<unsigned> mln::graph::attribute::card_t::result`

Type of the computed value.

Definition at line 64 of file `graph/attribute/card.hh`.

## 10.194 `mln::graph::attribute::representative_t` Struct Reference

Compute the representative vertex of every component in a graph.

```
#include <representative.hh>
```

### Public Types

- typedef `util::array< unsigned > result`

*Type of the computed value.*

### 10.194.1 Detailed Description

Compute the representative vertex of every component in a graph.

#### Returns

An array with the representative for each component. Components are labeled from 0.

Definition at line 63 of file representative.hh.

### 10.194.2 Member Typedef Documentation

#### 10.194.2.1 typedef util::array<unsigned> mln::graph::attribute::representative\_t::result

Type of the computed value.

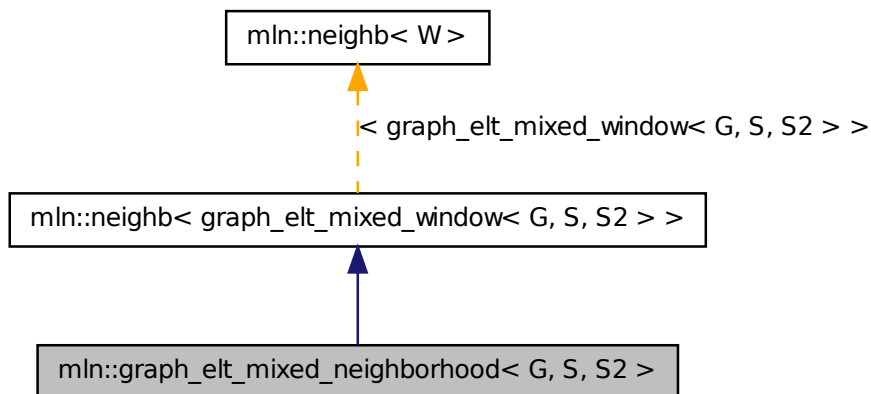
Definition at line 66 of file representative.hh.

## 10.195 mln::graph\_elt\_mixed\_neighborhood< G, S, S2 > Struct Template Reference

Elementary neighborhood on graph class.

```
#include <graph_elt_mixed_neighborhood.hh>
```

Inheritance diagram for mln::graph\_elt\_mixed\_neighborhood< G, S, S2 >:



### Public Types

- typedef `neighb_bkd_niter< graph\_elt\_mixed\_window< G, S, S2 > > bkd_niter`  
*Backward site iterator associated type.*

- typedef `neighb_fwd_niter`< `graph_elt_mixed_window`< `G`, `S`, `S2` > > `fwd_niter`  
*Forward site iterator associated type.*
- typedef `fwd_niter` `niter`  
*Site iterator associated type.*

### 10.195.1 Detailed Description

```
template<typename G, typename S, typename S2> struct mln::graph_elt_mixed_neighborhood<
G, S, S2 >
```

Elementary neighborhood on graph class.

#### Template Parameters

- `G` is a graph type.
- `S` is a site set type.
- `S2` is the site set type of the neighbors.

Definition at line 48 of file `graph_elt_mixed_neighborhood.hh`.

### 10.195.2 Member Typedef Documentation

**10.195.2.1** typedef `neighb_bkd_niter`<`graph_elt_mixed_window`< `G`, `S`, `S2` > > `mln::neighb`<`graph_elt_mixed_window`< `G`, `S`, `S2` > >::`bkd_niter` [`inherited`]

Backward site iterator associated type.

Definition at line 87 of file `mln/core/neighb.hh`.

**10.195.2.2** typedef `neighb_fwd_niter`<`graph_elt_mixed_window`< `G`, `S`, `S2` > > `mln::neighb`<`graph_elt_mixed_window`< `G`, `S`, `S2` > >::`fwd_niter` [`inherited`]

Forward site iterator associated type.

Definition at line 84 of file `mln/core/neighb.hh`.

**10.195.2.3** typedef `fwd_niter` `mln::neighb`< `graph_elt_mixed_window`< `G`, `S`, `S2` > >::`niter` [`inherited`]

Site iterator associated type.

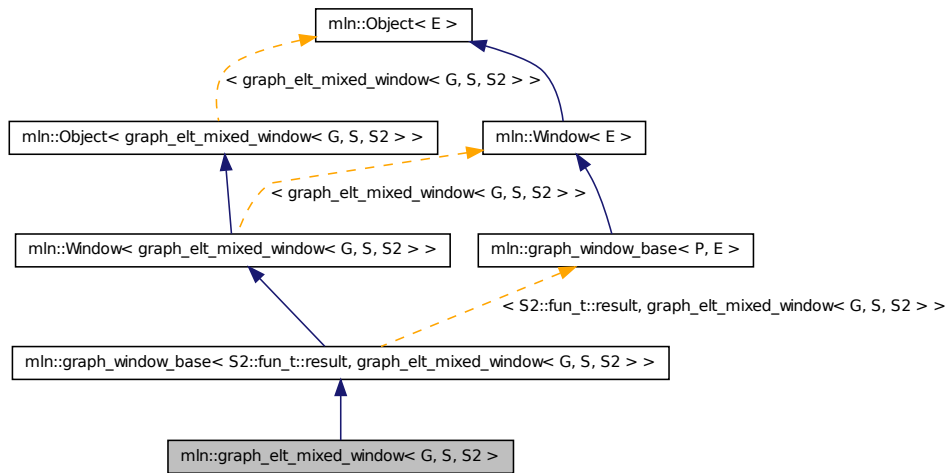
Definition at line 90 of file `mln/core/neighb.hh`.

## 10.196 `mln::graph_elt_mixed_window`< `G`, `S`, `S2` > Class Template Reference

Elementary window on graph class.

```
#include <graph_elt_mixed_window.hh>
```

Inheritance diagram for mln::graph\_elt\_mixed\_window< G, S, S2 >:



## Public Types

- typedef `super_::target` [target](#)  
*Associated types.*
- typedef `target::psite` [psite](#)  
*The type of psite corresponding to the window.*
- typedef `S::psite` [center\\_t](#)  
*Type of the window center element.*
- typedef `target::graph_element` [graph\\_element](#)  
*Type of the graph element pointed by this iterator.*
- typedef `graph_window_piter< target, self_, nbh_fwd_iter_ >` [fwd\\_qiter](#)  
*Site\_Iterator type to browse the psites of the window w.r.t.*
- typedef `graph_window_piter< target, self_, nbh_bkd_iter_ >` [bkd\\_qiter](#)  
*Site\_Iterator type to browse the psites of the window w.r.t.*
- typedef `fwd_qiter` [qiter](#)  
*The default qiter type.*
- typedef `S2::fun_t::result` [site](#)  
*Associated types.*

## Public Member Functions

- bool [is\\_valid](#) () const  
*Return true by default.*
- bool [is\\_empty](#) () const  
*Interface of the concept Window.*
- bool [is\\_centered](#) () const  
*Is the window centered?*
- bool [is\\_symmetric](#) () const  
*Is the window symmetric?*
- unsigned [delta](#) () const  
*Return the maximum coordinate gap between the window center and a window point.*
- [self\\_ & sym](#) ()  
*Apply a central symmetry to the target window.*

### 10.196.1 Detailed Description

```
template<typename G, typename S, typename S2> class mln::graph_elt_mixed_window< G, S, S2
>
```

Elementary window on graph class. *G* is the graph type. *S* is an image site set from where the center is extracted. *S2* is an image site set from where the neighbors are extracted.

Definition at line 109 of file `graph_elt_mixed_window.hh`.

### 10.196.2 Member Typedef Documentation

**10.196.2.1** `template<typename G , typename S , typename S2 > typedef graph_window_piter<target,self,nbh_bkd_iter_> mln::graph_elt_mixed_window< G, S, S2 >::bkd_qiter`

[Site\\_Iterator](#) type to browse the psites of the window w.r.t.

the reverse ordering of vertices.

Definition at line 139 of file `graph_elt_mixed_window.hh`.

**10.196.2.2** `template<typename G , typename S , typename S2 > typedef S ::psite mln::graph_elt_mixed_window< G, S, S2 >::center_t`

Type of the window center element.

Definition at line 128 of file `graph_elt_mixed_window.hh`.

**10.196.2.3** `template<typename G , typename S , typename S2 > typedef graph_window_piter<target,self,nbh_fwd_iter_> mln::graph_elt_mixed_window< G, S, S2 >::fwd_qiter`

[Site\\_Iterator](#) type to browse the psites of the window w.r.t. the ordering of vertices.

Definition at line 135 of file graph\_elt\_mixed\_window.hh.

**10.196.2.4** `template<typename G , typename S , typename S2 > typedef target ::graph_element mln::graph_elt_mixed_window< G, S, S2 >::graph_element`

Type of the graph element pointed by this iterator.

Definition at line 131 of file graph\_elt\_mixed\_window.hh.

**10.196.2.5** `template<typename G , typename S , typename S2 > typedef target ::psite mln::graph_elt_mixed_window< G, S, S2 >::psite`

The type of psite corresponding to the window.

Definition at line 125 of file graph\_elt\_mixed\_window.hh.

**10.196.2.6** `template<typename G , typename S , typename S2 > typedef fwd_qiter mln::graph_elt_mixed_window< G, S, S2 >::qiter`

The default qiter type.

Definition at line 142 of file graph\_elt\_mixed\_window.hh.

**10.196.2.7** `typedef S2::fun_t::result mln::graph_window_base< S2::fun_t::result , graph_elt_mixed_window< G, S, S2 > >::site [inherited]`

Associated types.

The type of site corresponding to the window.

Definition at line 48 of file graph\_window\_base.hh.

**10.196.2.8** `template<typename G , typename S , typename S2 > typedef super_::target mln::graph_elt_mixed_window< G, S, S2 >::target`

Associated types.

Definition at line 123 of file graph\_elt\_mixed\_window.hh.

### 10.196.3 Member Function Documentation

**10.196.3.1** `unsigned mln::graph_window_base< S2::fun_t::result , graph_elt_mixed_window< G, S, S2 > >::delta ( ) const [inherited]`

Return the maximum coordinate gap between the window center and a window point.

**10.196.3.2** `bool mln::graph_window_base< S2::fun_t::result , graph_elt_mixed_window< G, S, S2 > >::is_centered ( ) const [inherited]`

Is the window centered?

**10.196.3.3** `bool mln::graph_window_base< S2::fun_t::result , graph_elt_mixed_window< G, S, S2 > >::is_empty ( ) const [inherited]`

Interface of the concept Window.

Is the window is empty?

**10.196.3.4** `bool mln::graph_window_base< S2::fun_t::result , graph_elt_mixed_window< G, S, S2 > >::is_symmetric ( ) const [inherited]`

Is the window symmetric?

**10.196.3.5** `bool mln::graph_window_base< S2::fun_t::result , graph_elt_mixed_window< G, S, S2 > >::is_valid ( ) const [inherited]`

Return true by default.

**10.196.3.6** `self_& mln::graph_window_base< S2::fun_t::result , graph_elt_mixed_window< G, S, S2 > >::sym ( ) [inherited]`

Apply a central symmetry to the target window.

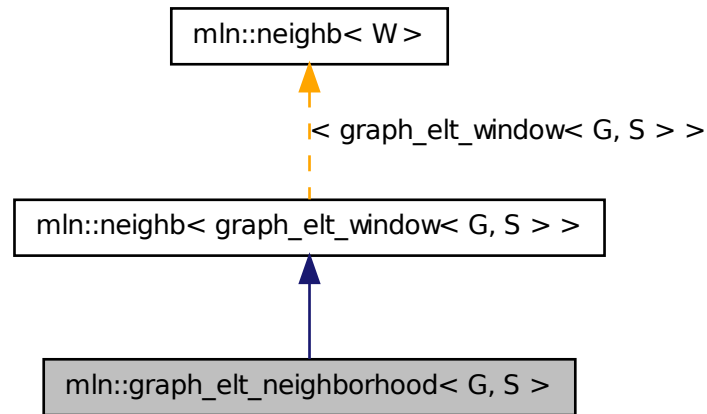
## **10.197 mln::graph\_elt\_neighborhood< G, S > Struct Template Reference**

Elementary neighborhood on graph class.

```
#include <graph_elt_neighborhood.hh>
```



Inheritance diagram for mln::graph\_elt\_neighborhood< G, S >:



## Public Types

- typedef `neighb_bkd_niter< graph_elt_window< G, S > > bkd_niter`  
*Backward site iterator associated type.*
- typedef `neighb_fwd_niter< graph_elt_window< G, S > > fwd_niter`  
*Forward site iterator associated type.*
- typedef `fwd_niter niter`  
*Site iterator associated type.*

### 10.197.1 Detailed Description

```
template<typename G, typename S> struct mln::graph_elt_neighborhood< G, S >
```

Elementary neighborhood on graph class.

#### Template Parameters

*G* is a graph type.

*S* is a site set type.

Definition at line 47 of file `graph_elt_neighborhood.hh`.

## 10.197.2 Member Typedef Documentation

**10.197.2.1** `typedef neighb_bkd_niter<graph_elt_window< G, S > > mln::neighb<graph_elt_window< G, S > >::bkd_niter` `[inherited]`

Backward site iterator associated type.

Definition at line 87 of file mln/core/neighb.hh.

**10.197.2.2** `typedef neighb_fwd_niter<graph_elt_window< G, S > > mln::neighb<graph_elt_window< G, S > >::fwd_niter` `[inherited]`

Forward site iterator associated type.

Definition at line 84 of file mln/core/neighb.hh.

**10.197.2.3** `typedef fwd_niter mln::neighb< graph_elt_window< G, S > >::niter` `[inherited]`

Site iterator associated type.

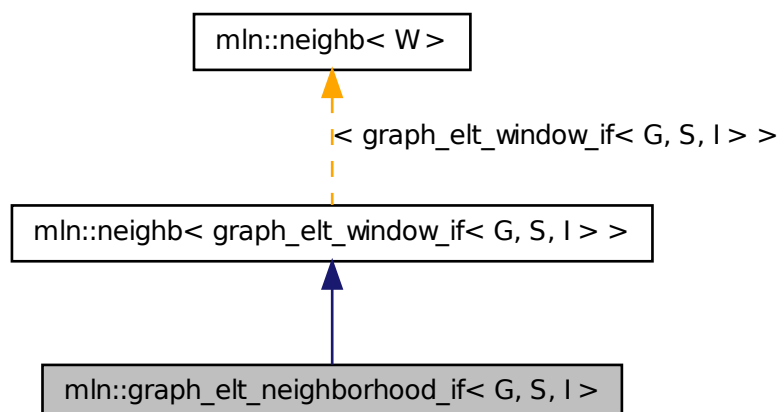
Definition at line 90 of file mln/core/neighb.hh.

## 10.198 mln::graph\_elt\_neighborhood\_if< G, S, I > Struct Template Reference

Elementary neighborhood\_if on graph class.

```
#include <graph_elt_neighborhood_if.hh>
```

Inheritance diagram for mln::graph\_elt\_neighborhood\_if< G, S, I >:



## Public Types

- typedef neighb\_bkd\_niter< graph\_elt\_window\_if< G, S, I > > bkd\_niter  
*Backward site iterator associated type.*
- typedef neighb\_fwd\_niter< graph\_elt\_window\_if< G, S, I > > fwd\_niter  
*Forward site iterator associated type.*
- typedef fwd\_niter niter  
*Site iterator associated type.*

## Public Member Functions

- graph\_elt\_neighborhood\_if ()  
*Constructors @ { Construct an invalid neighborhood.*
- graph\_elt\_neighborhood\_if (const Image< I > &mask)
- const I & mask () const  
*@ }*

### 10.198.1 Detailed Description

template<typename G, typename S, typename I> struct mln::graph\_elt\_neighborhood\_if< G, S, I >

Elementary neighborhood\_if on graph class.

Definition at line 43 of file graph\_elt\_neighborhood\_if.hh.

### 10.198.2 Member Typedef Documentation

**10.198.2.1** typedef neighb\_bkd\_niter<graph\_elt\_window\_if< G, S, I > > mln::neighb<graph\_elt\_window\_if< G, S, I > >::bkd\_niter **[inherited]**

Backward site iterator associated type.

Definition at line 87 of file mln/core/neighb.hh.

**10.198.2.2** typedef neighb\_fwd\_niter<graph\_elt\_window\_if< G, S, I > > mln::neighb<graph\_elt\_window\_if< G, S, I > >::fwd\_niter **[inherited]**

Forward site iterator associated type.

Definition at line 84 of file mln/core/neighb.hh.

**10.198.2.3** `typedef fwd_niter mln::neighb< graph_elt_window_if< G, S, I > >::niter`  
`[inherited]`

Site iterator associated type.

Definition at line 90 of file mln/core/neighb.hh.

### 10.198.3 Constructor & Destructor Documentation

**10.198.3.1** `template<typename G , typename S , typename I > mln::graph_elt_neighborhood_if<`  
`G, S, I >::graph_elt_neighborhood_if( ) [inline]`

Constructors @ { Construct an invalid neighborhood.

Definition at line 67 of file graph\_elt\_neighborhood\_if.hh.

**10.198.3.2** `template<typename G , typename S , typename I > mln::graph_elt_neighborhood_if<`  
`G, S, I >::graph_elt_neighborhood_if( const Image< I > & mask ) [inline]`

#### Parameters

[in] *mask* A graph image of Boolean.

Definition at line 74 of file graph\_elt\_neighborhood\_if.hh.

### 10.198.4 Member Function Documentation

**10.198.4.1** `template<typename G , typename S , typename I > const I &`  
`mln::graph_elt_neighborhood_if< G, S, I >::mask( ) const [inline]`

@ }

Return the graph image used as mask.

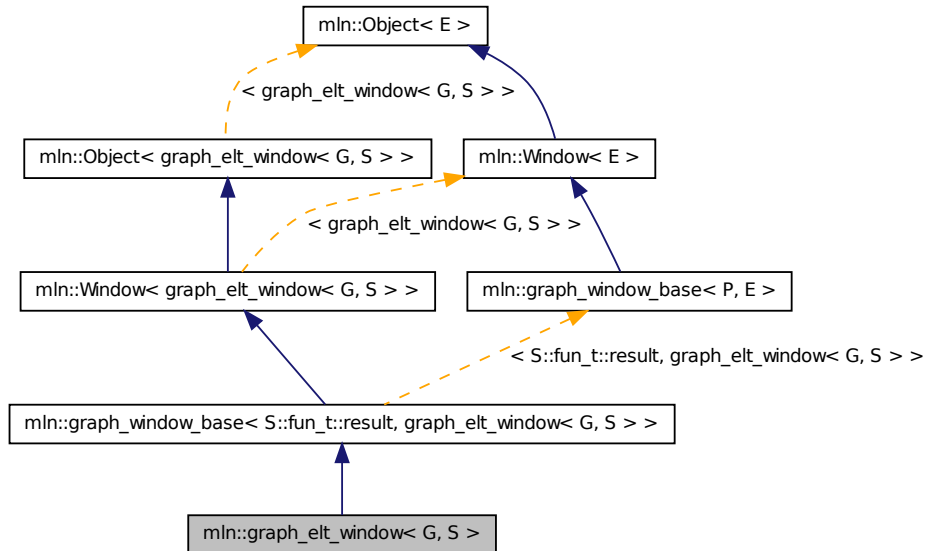
Definition at line 83 of file graph\_elt\_neighborhood\_if.hh.

## 10.199 mln::graph\_elt\_window< G, S > Class Template Reference

Elementary window on graph class.

```
#include <graph_elt_window.hh>
```

Inheritance diagram for mln::graph\_elt\_window< G, S >:



## Public Types

- typedef `S target`  
*Associated types.*
- typedef `S::psite psite`  
*The type of psite corresponding to the window.*
- typedef `S::psite center_t`  
*Type of the window center element.*
- typedef `S::graph_element graph_element`  
*Type of the graph element pointed by this iterator.*
- typedef `graph_window_piter< S, self_, nbh_fwd_iter_ > fwd_qiter`  
*Site\_Iterator type to browse the psites of the window w.r.t.*
- typedef `graph_window_piter< S, self_, nbh_bkd_iter_ > bkd_qiter`  
*Site\_Iterator type to browse the psites of the window w.r.t.*
- typedef `fwd_qiter qiter`  
*The default qiter type.*
- typedef `S::fun_t::result site`  
*Associated types.*

## Public Member Functions

- bool [is\\_valid](#) () const  
*Return true by default.*
- bool [is\\_empty](#) () const  
*Interface of the concept Window.*
- bool [is\\_centered](#) () const  
*Is the window centered?*
- bool [is\\_symmetric](#) () const  
*Is the window symmetric?*
- unsigned [delta](#) () const  
*Return the maximum coordinate gap between the window center and a window point.*
- [self\\_ & sym](#) ()  
*Apply a central symmetry to the target window.*

### 10.199.1 Detailed Description

**template<typename G, typename S> class mln::graph\_elt\_window< G, S >**

Elementary window on graph class.  $G$  is the graph type.  $S$  is an image site set from where the center is extracted.  $S_2$  is an image site set from where the neighbors are extracted.

Definition at line 111 of file graph\_elt\_window.hh.

### 10.199.2 Member Typedef Documentation

**10.199.2.1 template<typename G , typename S > typedef graph\_window\_ -  
piter<S,self\_nbh\_bkd\_iter\_> mln::graph\_elt\_window< G, S  
>::bkd\_qiter**

[Site\\_Iterator](#) type to browse the psites of the window w.r.t.

the reverse ordering of vertices.

Definition at line 142 of file graph\_elt\_window.hh.

**10.199.2.2 template<typename G , typename S > typedef S ::psite mln::graph\_elt\_window< G, S  
>::center\_t**

Type of the window center element.

Definition at line 131 of file graph\_elt\_window.hh.

**10.199.2.3** `template<typename G , typename S > typedef graph_window_riter<S,self_,nbh_fwd_iter_> mln::graph_elt_window< G, S >::fwd_qiter`

[Site\\_Iterator](#) type to browse the psites of the window w.r.t. the ordering of vertices.

Definition at line 138 of file graph\_elt\_window.hh.

**10.199.2.4** `template<typename G , typename S > typedef S ::graph_element mln::graph_elt_window< G, S >::graph_element`

Type of the graph element pointed by this iterator.

Definition at line 134 of file graph\_elt\_window.hh.

**10.199.2.5** `template<typename G , typename S > typedef S ::psite mln::graph_elt_window< G, S >::psite`

The type of psite corresponding to the window.

Definition at line 128 of file graph\_elt\_window.hh.

**10.199.2.6** `template<typename G , typename S > typedef fwd_qiter mln::graph_elt_window< G, S >::qiter`

The default qiter type.

Definition at line 145 of file graph\_elt\_window.hh.

**10.199.2.7** `typedef S::fun_t::result mln::graph_window_base< S::fun_t::result , graph_elt_window< G, S > >::site [inherited]`

Associated types.

The type of site corresponding to the window.

Definition at line 48 of file graph\_window\_base.hh.

**10.199.2.8** `template<typename G , typename S > typedef S mln::graph_elt_window< G, S >::target`

Associated types.

Definition at line 125 of file graph\_elt\_window.hh.

### 10.199.3 Member Function Documentation

**10.199.3.1** `unsigned mln::graph_window_base< S::fun_t::result , graph_elt_window< G, S > >::delta( ) const [inherited]`

Return the maximum coordinate gap between the window center and a window point.

**10.199.3.2** `bool mln::graph_window_base< S::fun_t::result , graph_elt_window< G, S >  
>::is_centered ( ) const [inherited]`

Is the window centered?

**10.199.3.3** `bool mln::graph_window_base< S::fun_t::result , graph_elt_window< G, S >  
>::is_empty ( ) const [inherited]`

Interface of the concept Window.

Is the window is empty?

**10.199.3.4** `bool mln::graph_window_base< S::fun_t::result , graph_elt_window< G, S >  
>::is_symmetric ( ) const [inherited]`

Is the window symmetric?

**10.199.3.5** `bool mln::graph_window_base< S::fun_t::result , graph_elt_window< G, S >  
>::is_valid ( ) const [inherited]`

Return true by default.

**10.199.3.6** `self_& mln::graph_window_base< S::fun_t::result , graph_elt_window< G, S >  
>::sym ( ) [inherited]`

Apply a central symmetry to the target window.

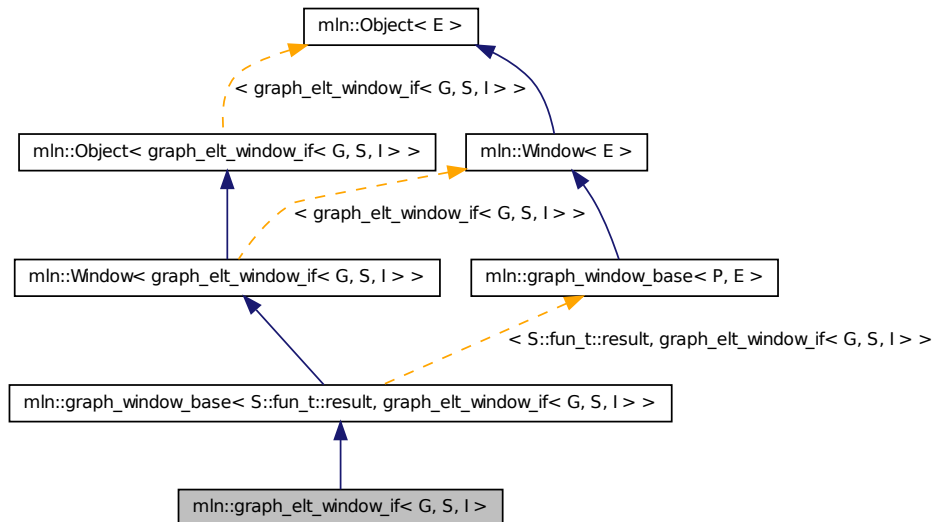
## 10.200 mln::graph\_elt\_window\_if< G, S, I > Class Template Reference

Custom window on graph class.

```
#include <graph_elt_window_if.hh>
```



Inheritance diagram for mln::graph\_elt\_window\_if< G, S, I >:



## Public Types

- typedef I [mask\\_t](#)  
*The type of the image used as mask.*
- typedef S [target](#)  
@}
- typedef target::psite [psite](#)  
*The type of psite corresponding to the window.*
- typedef [graph\\_window\\_if\\_piter](#)< [target](#), [self\\_](#), [nbh\\_fwd\\_iter\\_](#) > [fwd\\_qiter](#)  
*Site\_Iterator type to browse the psites of the window w.r.t.*
- typedef [graph\\_window\\_if\\_piter](#)< [target](#), [self\\_](#), [nbh\\_bkd\\_iter\\_](#) > [bkd\\_qiter](#)  
*Site\_Iterator type to browse the psites of the window w.r.t.*
- typedef [fwd\\_qiter](#) [qiter](#)  
*The default qiter type.*
- typedef S::fun\_t::result [site](#)  
*Associated types.*

## Public Member Functions

- void [change\\_mask](#) (const [Image](#)< I > &mask)  
*Change mask image.*
- [graph\\_elt\\_window\\_if](#) ()  
*Constructor.*
- [graph\\_elt\\_window\\_if](#) (const [Image](#)< I > &mask)
- bool [is\\_valid](#) () const  
*Return true by default.*
- const I & [mask](#) () const  
*Return the graph image used as mask.*
- bool [is\\_empty](#) () const  
*Interface of the concept Window.*
- bool [is\\_centered](#) () const  
*Is the window centered?*
- bool [is\\_symmetric](#) () const  
*Is the window symmetric?*
- unsigned [delta](#) () const  
*Return the maximum coordinate gap between the window center and a window point.*
- [self\\_ & sym](#) ()  
*Apply a central symmetry to the target window.*

### 10.200.1 Detailed Description

**template<typename G, typename S, typename I> class mln::graph\_elt\_window\_if< G, S, I >**

Custom window on graph class. It is defined thanks to a mask.

G is the graph type. S is the image site set. I is the graph image the type used as mask.

Definition at line 105 of file graph\_elt\_window\_if.hh.

### 10.200.2 Member Typedef Documentation

**10.200.2.1 template<typename G , typename S , typename I > typedef graph\_window\_if\_piter<target,self\_nbh\_bkd\_iter\_> mln::graph\_elt\_window\_if< G, S, I >::bkd\_qiter**

[Site\\_Iterator](#) type to browse the psites of the window w.r.t.

the reverse ordering of vertices.

Definition at line 147 of file graph\_elt\_window\_if.hh.

**10.200.2.2** `template<typename G , typename S , typename I > typedef graph_window_if_piter<target,self,nbh_fwd_iter_> mln::graph_elt_window_if< G, S, I >::fwd_qiter`

[Site\\_Iterator](#) type to browse the psites of the window w.r.t. the ordering of vertices.

Definition at line 143 of file graph\_elt\_window\_if.hh.

**10.200.2.3** `template<typename G , typename S , typename I > typedef I mln::graph_elt_window_if< G, S, I >::mask_t`

The type of the image used as mask.

Definition at line 119 of file graph\_elt\_window\_if.hh.

**10.200.2.4** `template<typename G , typename S , typename I > typedef target ::psite mln::graph_elt_window_if< G, S, I >::psite`

The type of psite corresponding to the window.

Definition at line 139 of file graph\_elt\_window\_if.hh.

**10.200.2.5** `template<typename G , typename S , typename I > typedef fwd_qiter mln::graph_elt_window_if< G, S, I >::qiter`

The default qiter type.

Definition at line 150 of file graph\_elt\_window\_if.hh.

**10.200.2.6** `typedef S::fun_t::result mln::graph_window_base< S::fun_t::result , graph_elt_window_if< G, S, I > >::site [inherited]`

Associated types.

The type of site corresponding to the window.

Definition at line 48 of file graph\_window\_base.hh.

**10.200.2.7** `template<typename G , typename S , typename I > typedef S mln::graph_elt_window_if< G, S, I >::target`

@ }

Associated types. The image domain on which this window iterates on.

Definition at line 136 of file graph\_elt\_window\_if.hh.

### 10.200.3 Constructor & Destructor Documentation

**10.200.3.1** `template<typename G , typename S , typename I > mln::graph_elt_window_if< G, S, I >::graph_elt_window_if( ) [inline]`

Constructor.

@{ Default. Construct an invalid window.

Definition at line 174 of file `graph_elt_window_if.hh`.

**10.200.3.2** `template<typename G , typename S , typename I > mln::graph_elt_window_if< G, S, I >::graph_elt_window_if( const Image< I > & mask ) [inline]`

#### Parameters

[in] *mask* A graph image of bool.

#### See also

[vertex\\_image](#), [edge\\_image](#).

Definition at line 181 of file `graph_elt_window_if.hh`.

### 10.200.4 Member Function Documentation

**10.200.4.1** `template<typename G , typename S , typename I > void mln::graph_elt_window_if< G, S, I >::change_mask( const Image< I > & mask ) [inline]`

Change mask image.

Definition at line 199 of file `graph_elt_window_if.hh`.

References `mln::graph_elt_window_if< G, S, I >::is_valid()`.

**10.200.4.2** `unsigned mln::graph_window_base< S::fun_t::result , graph_elt_window_if< G, S, I > >::delta( ) const [inherited]`

Return the maximum coordinate gap between the window center and a window point.

**10.200.4.3** `bool mln::graph_window_base< S::fun_t::result , graph_elt_window_if< G, S, I > >::is_centered( ) const [inherited]`

Is the window centered?

**10.200.4.4** `bool mln::graph_window_base< S::fun_t::result , graph_elt_window_if< G, S, I > >::is_empty( ) const [inherited]`

Interface of the concept Window.

Is the window is empty?

**10.200.4.5** `bool mln::graph_window_base< S::fun_t::result , graph_elt_window_if< G, S, I > >::is_symmetric ( ) const` [*inherited*]

Is the window symmetric?

**10.200.4.6** `template<typename G , typename S , typename I > bool mln::graph_elt_window_if< G, S, I >::is_valid ( ) const` [*inline*]

Return true by default.

Reimplemented from `mln::graph_window_base< S::fun_t::result, graph_elt_window_if< G, S, I > >`.

Definition at line 208 of file `graph_elt_window_if.hh`.

Referenced by `mln::graph_elt_window_if< G, S, I >::change_mask()`.

**10.200.4.7** `template<typename G , typename S , typename I > const I & mln::graph_elt_window_if< G, S, I >::mask ( ) const` [*inline*]

Return the graph image used as mask.

Definition at line 190 of file `graph_elt_window_if.hh`.

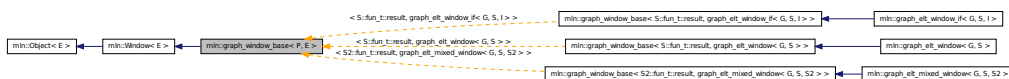
**10.200.4.8** `self_& mln::graph_window_base< S::fun_t::result , graph_elt_window_if< G, S, I > >::sym ( )` [*inherited*]

Apply a central symmetry to the target window.

## 10.201 mln::graph\_window\_base< P, E > Class Template Reference

```
#include <graph_window_base.hh>
```

Inheritance diagram for `mln::graph_window_base< P, E >`:



### Public Types

- typedef `P` [site](#)  
*Associated types.*

### Public Member Functions

- bool `is_valid () const`  
*Return true by default.*

- bool `is_empty` () const  
*Interface of the concept [Window](#).*
- bool `is_centered` () const  
*Is the window centered?*
- bool `is_symmetric` () const  
*Is the window symmetric?*
- unsigned `delta` () const  
*Return the maximum coordinate gap between the window center and a window point.*
- `self_ & sym` ()  
*Apply a central symmetry to the target window.*

### 10.201.1 Detailed Description

**template<typename P, typename E> class mln::graph\_window\_base< P, E >**

#### Template Parameters

*P* [Site](#) type.

Definition at line 40 of file `graph_window_base.hh`.

### 10.201.2 Member Typedef Documentation

**10.201.2.1 template<typename P, typename E> typedef P mln::graph\_window\_base< P, E >::site**

Associated types.

The type of site corresponding to the window.

Definition at line 48 of file `graph_window_base.hh`.

### 10.201.3 Member Function Documentation

**10.201.3.1 template<typename P , typename E > unsigned mln::graph\_window\_base< P, E >::delta ( ) const [inline]**

Return the maximum coordinate gap between the window center and a window point.

Definition at line 128 of file `graph_window_base.hh`.

**10.201.3.2 template<typename P , typename E > bool mln::graph\_window\_base< P, E >::is\_centered ( ) const [inline]**

Is the window centered?

Definition at line 112 of file `graph_window_base.hh`.

**10.201.3.3** `template<typename P , typename E > bool mln::graph_window_base< P, E >::is_empty ( ) const [inline]`

Interface of the concept [Window](#).

Is the window is empty?

Definition at line 104 of file graph\_window\_base.hh.

**10.201.3.4** `template<typename P , typename E > bool mln::graph_window_base< P, E >::is_symmetric ( ) const [inline]`

Is the window symmetric?

Definition at line 120 of file graph\_window\_base.hh.

**10.201.3.5** `template<typename P , typename E > bool mln::graph_window_base< P, E >::is_valid ( ) const [inline]`

Return true by default.

Reimplemented in [mln::graph\\_elt\\_window\\_if< G, S, I >](#).

Definition at line 153 of file graph\_window\_base.hh.

**10.201.3.6** `template<typename P , typename E > graph_window_base< P, E > & mln::graph_window_base< P, E >::sym ( ) [inline]`

Apply a central symmetry to the target window.

Definition at line 137 of file graph\_window\_base.hh.

## 10.202 mln::graph\_window\_if\_piter< S, W, I > Class Template Reference

Forward iterator on line graph window.

```
#include <graph_window_if_piter.hh>
```

Inherits [site\\_relative\\_iterator\\_base< W, graph\\_window\\_if\\_piter< S, W, I > >](#), and [is\\_masked\\_impl\\_selector< S, W::mask\\_t::domain\\_t, graph\\_window\\_if\\_piter< S, W, I > >](#).

### Public Types

- typedef `S::fun_t::result P`  
*Associated types.*

### Public Member Functions

- void [next](#) ()  
*Go to the next element.*

- [graph\\_window\\_if\\_piter](#) ()  
*Construction.*
- `const S::graph_element & element () const`  
*Return the graph element pointed by this iterator.*
- `unsigned id () const`  
*Return the graph element id.*

### 10.202.1 Detailed Description

`template<typename S, typename W, typename I> class mln::graph_window_if_piter< S, W, I >`

Forward iterator on line graph window.

Definition at line 47 of file `graph_window_if_piter.hh`.

### 10.202.2 Member Typedef Documentation

**10.202.2.1** `template<typename S , typename W , typename I > typedef S::fun_t ::result  
mln::graph_window_if_piter< S, W, I >::P`

Associated types.

Definition at line 60 of file `graph_window_if_piter.hh`.

### 10.202.3 Constructor & Destructor Documentation

**10.202.3.1** `template<typename S , typename W , typename I > mln::graph_window_if_piter< S,  
W, I >::graph_window_if_piter ( ) [inline]`

Construction.

Definition at line 122 of file `graph_window_if_piter.hh`.

### 10.202.4 Member Function Documentation

**10.202.4.1** `template<typename S , typename W , typename I > const S::graph_element &  
mln::graph_window_if_piter< S, W, I >::element ( ) const [inline]`

Return the graph element pointed by this iterator.

Definition at line 213 of file `graph_window_if_piter.hh`.

**10.202.4.2** `template<typename S , typename W , typename I > unsigned  
mln::graph_window_if_piter< S, W, I >::id ( ) const [inline]`

Return the graph element id.



FIXME: we do not want to have this member since there is an automatic conversion to the graph element. C++ does not seem to use this conversion operator.

Definition at line 221 of file graph\_window\_if\_piter.hh.

**10.202.4.3** void mln::Site\_Iterator< graph\_window\_if\_piter< S, W, I > >::next ( )  
[inherited]

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

#### Precondition

The iterator is valid.

## 10.203 mln::graph\_window\_piter< S, W, I > Class Template Reference

Forward iterator on line graph window.

```
#include <graph_window_piter.hh>
```

Inherits site\_relative\_iterator\_base< W, graph\_window\_piter< S, W, I >, W::center\_t >, and impl\_selector< W::center\_t, W::psite, graph\_window\_piter< S, W, I > >.

### Public Types

- typedef S::fun\_t::result [P](#)  
*Associated types*  
*Type of the window elements.*
- typedef W::center\_t [center\\_t](#)  
*Type of the window center.*
- typedef W::graph\_element [graph\\_element](#)  
*Type of the graph element pointed by this iterator.*

### Public Member Functions

- void [change\\_target\\_site\\_set](#) (const S &s)  
*Change the target site set.*
- void [next](#) ()  
*Go to the next element.*
- const S & [target\\_site\\_set](#) () const

*Return the target site set.*

- [graph\\_window\\_piter](#) ()  
*Construction.*
- `template<typename Pref >`  
`graph\_window\_piter (const Window< W > &win, const Pref &p_ref)`  
*To be used in case the center and neighbor sites have the same type and belong to the same site set.*
- `template<typename Pref >`  
`graph\_window\_piter (const Window< W > &win, const Site\_Set< S > &target_site_set, const Pref &p_ref)`  
*To be used in case center and neighbors sites do not have the same type and do not belong to the same site set.*
- `const graph\_element & element () const`  
*Return the graph element pointed by this iterator.*
- `unsigned id () const`  
*Return the graph element id.*

### 10.203.1 Detailed Description

`template<typename S, typename W, typename I> class mln::graph_window_piter< S, W, I >`

Forward iterator on line graph window.

#### Template Parameters

*S* is the site set type.

*W* is the window type.

*I* is the underlying iterator type.

Definition at line 99 of file `graph_window_piter.hh`.

### 10.203.2 Member Typedef Documentation

**10.203.2.1** `template<typename S , typename W , typename I > typedef W::center_t mln::graph_window_piter< S, W, I >::center_t`

Type of the window center.

Definition at line 120 of file `graph_window_piter.hh`.

**10.203.2.2** `template<typename S , typename W , typename I > typedef W::graph_element mln::graph_window_piter< S, W, I >::graph_element`

Type of the graph element pointed by this iterator.

Definition at line 122 of file `graph_window_piter.hh`.

**10.203.2.3** `template<typename S , typename W , typename I > typedef S::fun_t ::result  
mln::graph_window_piter< S, W, I >::P`

Associated types

Type of the window elements.

Definition at line 118 of file graph\_window\_piter.hh.

### 10.203.3 Constructor & Destructor Documentation

**10.203.3.1** `template<typename S , typename W , typename I > mln::graph_window_piter< S, W,  
I >::graph_window_piter ( ) [inline]`

Construction.

Definition at line 226 of file graph\_window\_piter.hh.

**10.203.3.2** `template<typename S , typename W , typename I > template<typename Pref >  
mln::graph_window_piter< S, W, I >::graph_window_piter ( const Window< W > &  
win, const Pref & p_ref ) [inline]`

To be used in case the center and neighbor sites have the same type and belong to the same site set.

#### Parameters

*win* The underlying window.

*p\_ref* [Window](#) center.

Definition at line 235 of file graph\_window\_piter.hh.

**10.203.3.3** `template<typename S , typename W , typename I > template<typename Pref >  
mln::graph_window_piter< S, W, I >::graph_window_piter ( const Window< W > &  
win, const Site_Set< S > & target_site_set, const Pref & p_ref ) [inline]`

To be used in case center and neighbors sites do not have the same type and do not belong to the same site set.

#### Parameters

*win* The underlying window.

*target\_site\_set* [Site](#) set in which neighbor sites are extracted.

*p\_ref* [Window](#) center.

Definition at line 249 of file graph\_window\_piter.hh.

### 10.203.4 Member Function Documentation

**10.203.4.1** `template<typename S , typename W , typename I > void mln::graph_window_piter<  
S, W, I >::change_target_site_set ( const S & s ) [inline]`

Change the target site set.

[Window](#) elements different from the center come from the target site set.

Definition at line 357 of file graph\_window\_piter.hh.

**10.203.4.2** `template<typename S , typename W , typename I > const graph_window_piter< S, W, I >::graph_element & mln::graph_window_piter< S, W, I >::element ( ) const [inline]`

Return the graph element pointed by this iterator.

Definition at line 341 of file graph\_window\_piter.hh.

**10.203.4.3** `template<typename S , typename W , typename I > unsigned mln::graph_window_piter< S, W, I >::id ( ) const [inline]`

Return the graph element id.

FIXME: we do not want to have this member since there is an automatic conversion to the graph element. C++ does not seem to use this conversion operator.

Definition at line 349 of file graph\_window\_piter.hh.

**10.203.4.4** `void mln::Site_Iterator< graph_window_piter< S, W, I > >::next ( ) [inherited]`

Go to the next element.

### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

### Precondition

The iterator is valid.

**10.203.4.5** `template<typename S , typename W , typename I > const S & mln::graph_window_piter< S, W, I >::target_site_set ( ) const [inline]`

Return the target site set.

[Window](#) elements different from the center come from the target site set.

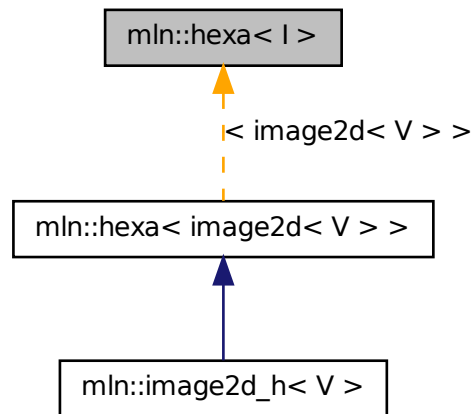
Definition at line 366 of file graph\_window\_piter.hh.

## 10.204 mln::hexa< I > Struct Template Reference

hexagonal image class.

```
#include <hexa.hh>
```

Inheritance diagram for mln::hexa< I >:



## Public Types

- typedef hexa\_bkd\_piter\_< box2d > bkd\_piter  
*FIXME : should it be in box2d\_h? Backward Site\_Iterator associated type.*
- typedef hexa\_fwd\_piter\_< box2d > fwd\_piter  
*FIXME : should it be in box2d\_h? Forward Site\_Iterator associated type.*
- typedef I::lvalue lvalue  
*Lvalue associated type.*
- typedef point2d\_h psite  
*Point site type.*
- typedef I::rvalue rvalue  
*Return type of read-only access.*
- typedef hexa< tag::image\_< I > > skeleton  
*Skeleton.*
- typedef I::value value  
*Value associated type.*

## Public Member Functions

- const box2d\_h & domain () const

*Give the definition domain.*

- `bool has (const psite &p) const`  
*Test if  $p$  belongs to the image domain.*
- `hexa (I &ima)`  
*Constructor with an base image.*
- `hexa ()`  
*Constructor without argument.*
- `lvalue operator() (const point2d\_h &p)`  
*Read-write access of pixel value at hexa point site  $p$ .*
- `rvalue operator() (const point2d\_h &p) const`  
*Read-only access of pixel value at hexa point site  $p$ .*

### 10.204.1 Detailed Description

`template<typename I> struct mln::hexa< I >`

hexagonal image class. The parameter `I` is the type of the base image. This image class which handles hexagonal grid.

```
Ex : 1 3 5 7 9 11 0 2 4 6 8 10 ----- 0 XX| | | | |XX ----- 2 XX| | | | |XX
----- 4 XX| | | | |XX ----- 6 XX| | | | |XX ----- 8 XX| | | | |
|XX -----
```

Definition at line 116 of file `hexa.hh`.

### 10.204.2 Member Typedef Documentation

**10.204.2.1** `template<typename I> typedef hexa_bkd_piter_<box2d> mln::hexa< I >::bkd_piter`

FIXME : should it be in `box2d_h`? Backward [Site\\_Iterator](#) associated type.

Definition at line 140 of file `hexa.hh`.

**10.204.2.2** `template<typename I> typedef hexa_fwd_piter_<box2d> mln::hexa< I >::fwd_piter`

FIXME : should it be in `box2d_h`? Forward [Site\\_Iterator](#) associated type.

Definition at line 136 of file `hexa.hh`.

**10.204.2.3** `template<typename I> typedef I ::lvalue mln::hexa< I >::lvalue`

Lvalue associated type.

Definition at line 126 of file `hexa.hh`.

**10.204.2.4** `template<typename I> typedef point2d_h mln::hexa< I >::psite`

[Point](#) site type.

Reimplemented in [mln::image2d\\_h< V >](#).

Definition at line 132 of file hexa.hh.

**10.204.2.5** `template<typename I> typedef I ::rvalue mln::hexa< I >::rvalue`

Return type of read-only access.

Definition at line 129 of file hexa.hh.

**10.204.2.6** `template<typename I> typedef hexa< tag::image_<I> > mln::hexa< I >::skeleton`

Skeleton.

Definition at line 120 of file hexa.hh.

**10.204.2.7** `template<typename I> typedef I ::value mln::hexa< I >::value`

[Value](#) associated type.

Definition at line 123 of file hexa.hh.

**10.204.3 Constructor & Destructor Documentation****10.204.3.1** `template<typename I> mln::hexa< I >::hexa ( ) [inline]`

Constructor without argument.

Definition at line 215 of file hexa.hh.

**10.204.3.2** `template<typename I> mln::hexa< I >::hexa ( I & ima ) [inline]`

Constructor with an base image.

Definition at line 222 of file hexa.hh.

**10.204.4 Member Function Documentation****10.204.4.1** `template<typename I> const box2d_h & mln::hexa< I >::domain ( ) const [inline]`

Give the definition domain.

Definition at line 250 of file hexa.hh.

**10.204.4.2** `template<typename I> bool mln::hexa< I >::has ( const psite & p ) const [inline]`

Test if *p* belongs to the image domain.

Definition at line 259 of file hexa.hh.

Referenced by `mln::hexa< I >::operator()`.

#### 10.204.4.3 `template<typename I> hexa< I >::rvalue mln::hexa< I >::operator() ( const point2d_h & p ) const [inline]`

Read-only access of pixel value at hexa point site `p`.

Definition at line 230 of file hexa.hh.

References `mln::hexa< I >::has()`.

#### 10.204.4.4 `template<typename I> hexa< I >::lvalue mln::hexa< I >::operator() ( const point2d_h & p ) [inline]`

Read-write access of pixel value at hexa point site `p`.

Definition at line 240 of file hexa.hh.

References `mln::hexa< I >::has()`.

## 10.205 `mln::histo::array< T >` Struct Template Reference

Generic histogram class over a value set with type `T`.

```
#include <array.hh>
```

### 10.205.1 Detailed Description

```
template<typename T> struct mln::histo::array< T >
```

Generic histogram class over a value set with type `T`.

Definition at line 48 of file `histo/array.hh`.

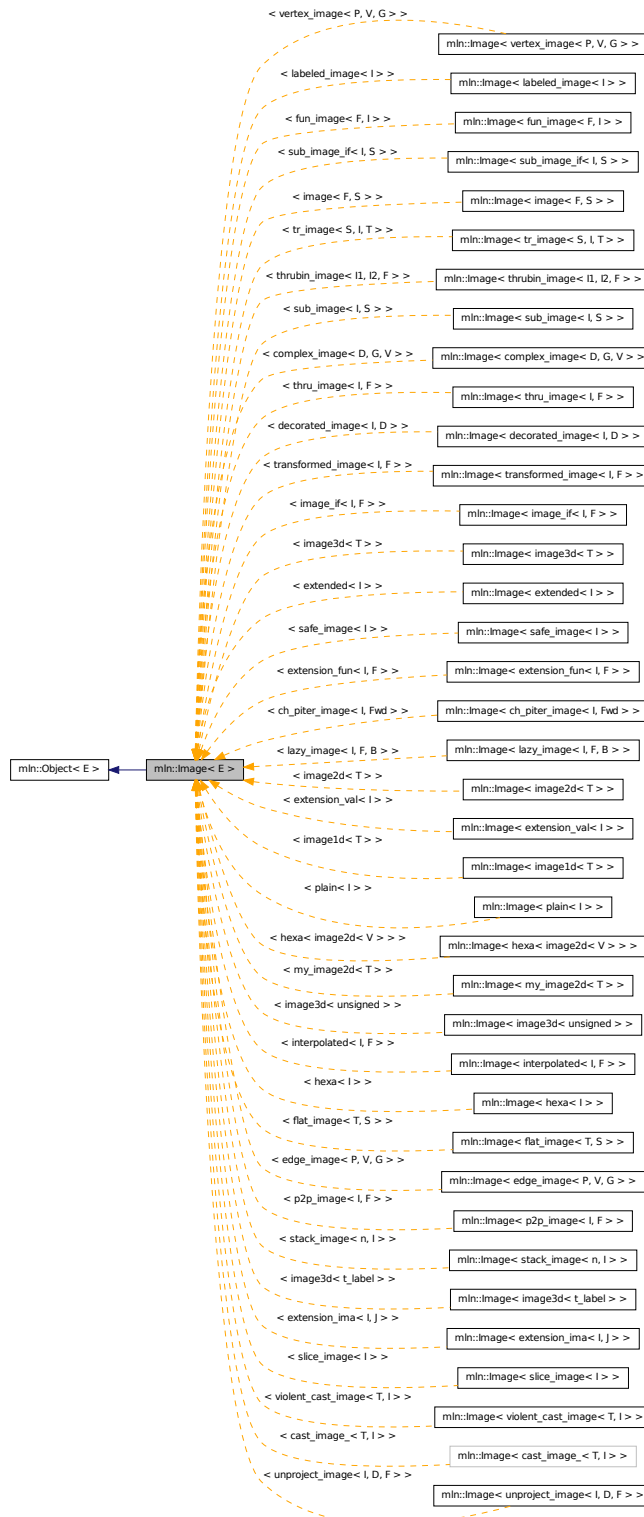
## 10.206 `mln::Image< E >` Struct Template Reference

Base class for implementation of image classes.

```
#include <image.hh>
```



Inheritance diagram for mln::Image< E >:



### 10.206.1 Detailed Description

`template<typename E> struct mln::Image< E >`

Base class for implementation of image classes.

#### See also

[mln::doc::Image](#) for a complete documentation of this class contents.

Definition at line 73 of file core/concept/image.hh.

## 10.207 mln::image1d< T > Struct Template Reference

Basic 1D image class.

```
#include <image1d.hh>
```

Inherits `image_primary< T, box1d, image1d< T > >`.

### Public Types

- typedef T & [lvalue](#)  
*Return type of read-write access.*
- typedef const T & [rvalue](#)  
*Return type of read-only access.*
- typedef [image1d< tag::value\\_< T > >](#) [skeleton](#)  
*Skeleton.*
- typedef T [value](#)  
*Value associated type.*

### Public Member Functions

- const [box1d](#) & [bbox](#) () const  
*Give the bounding box domain.*
- unsigned [border](#) () const  
*Give the border thickness.*
- const T \* [buffer](#) () const  
*Give a hook to the value buffer.*
- T \* [buffer](#) ()  
*Give a hook to the value buffer.*
- int [delta\\_index](#) (const [dpoint1d](#) &dp) const

Give the offset corresponding to the delta-point  $\Delta p$ .

- const `box1d` & `domain` () const  
Give the definition domain.
- const T & `element` (unsigned i) const  
Read-only access to the  $i$ -th image value (including the border).
- T & `element` (unsigned i)  
Read-write access to the  $i$ -th image value (including the border).
- bool `has` (const `point1d` &p) const  
Test if  $p$  is valid.
- `image1d` ()  
Constructor without argument.
- `image1d` (const `box1d` &b, unsigned bdr=border::thickness)  
Constructor with a box and the border thickness.
- `image1d` (unsigned ninds, unsigned bdr=border::thickness)  
Constructor with the number of indices and the border thickness.
- unsigned `nelements` () const  
Give the number of cells (points including border ones).
- unsigned `ninds` () const  
Give the number of indexes.
- T & `operator()` (const `point1d` &p)  
Read-write access to the image value located at point  $p$ .
- const T & `operator()` (const `point1d` &p) const  
Read-only access to the image value located at point  $p$ .
- `point1d` `point_at_index` (unsigned i) const  
Give the point corresponding to the offset  $o$ .
- const `box1d` & `vbbox` () const  
virtual box, i.e., box including the virtual border

### 10.207.1 Detailed Description

`template<typename T> struct mln::image1d< T >`

Basic 1D image class. The parameter  $T$  is the type of pixel values. This image class stores data in memory and has a virtual border with constant thickness before and after data.

Definition at line 155 of file `image1d.hh`.

## 10.207.2 Member Typedef Documentation

### 10.207.2.1 `template<typename T> typedef T& mln::image1d< T >::lvalue`

Return type of read-write access.

Definition at line 167 of file image1d.hh.

### 10.207.2.2 `template<typename T> typedef const T& mln::image1d< T >::rvalue`

Return type of read-only access.

Definition at line 164 of file image1d.hh.

### 10.207.2.3 `template<typename T> typedef image1d< tag::value_<T> > mln::image1d< T >::skeleton`

Skeleton.

Definition at line 170 of file image1d.hh.

### 10.207.2.4 `template<typename T> typedef T mln::image1d< T >::value`

[Value](#) associated type.

Definition at line 161 of file image1d.hh.

## 10.207.3 Constructor & Destructor Documentation

### 10.207.3.1 `template<typename T > mln::image1d< T >::image1d ( ) [inline]`

Constructor without argument.

Definition at line 371 of file image1d.hh.

### 10.207.3.2 `template<typename T > mln::image1d< T >::image1d ( unsigned ninds, unsigned bdr = border::thickness ) [inline]`

Constructor with the number of indices and the border thickness.

Definition at line 384 of file image1d.hh.

References `mln::make::box1d()`.

### 10.207.3.3 `template<typename T > mln::image1d< T >::image1d ( const box1d & b, unsigned bdr = border::thickness ) [inline]`

Constructor with a box and the border thickness.

Definition at line 377 of file image1d.hh.

## 10.207.4 Member Function Documentation

**10.207.4.1** `template<typename T > const box1d & mln::image1d< T >::bbox ( ) const [inline]`

Give the bounding box domain.

Definition at line 411 of file image1d.hh.

**10.207.4.2** `template<typename T > unsigned mln::image1d< T >::border ( ) const [inline]`

Give the border thickness.

Definition at line 429 of file image1d.hh.

**10.207.4.3** `template<typename T > const T * mln::image1d< T >::buffer ( ) const [inline]`

Give a hook to the value buffer.

Definition at line 520 of file image1d.hh.

**10.207.4.4** `template<typename T > T * mln::image1d< T >::buffer ( ) [inline]`

Give a hook to the value buffer.

Definition at line 529 of file image1d.hh.

**10.207.4.5** `template<typename T > int mln::image1d< T >::delta_index ( const dpoint1d & dp ) const [inline]`

Give the offset corresponding to the delta-point *dp*.

Definition at line 538 of file image1d.hh.

**10.207.4.6** `template<typename T > const box1d & mln::image1d< T >::domain ( ) const [inline]`

Give the definition domain.

Definition at line 402 of file image1d.hh.

**10.207.4.7** `template<typename T > const T & mln::image1d< T >::element ( unsigned i ) const [inline]`

Read-only access to the *i*-th image value (including the border).

Definition at line 502 of file image1d.hh.

References `mln::image1d< T >::nelements()`.

**10.207.4.8** `template<typename T> T & mln::image1d< T >::element ( unsigned i )`  
`[inline]`

Read-write access to the *i*-th image value (including the border).

Definition at line 511 of file image1d.hh.

References mln::image1d< T >::nelements().

**10.207.4.9** `template<typename T> bool mln::image1d< T >::has ( const point1d & p ) const`  
`[inline]`

Test if *p* is valid.

Definition at line 447 of file image1d.hh.

Referenced by mln::image1d< T >::operator()().

**10.207.4.10** `template<typename T> unsigned mln::image1d< T >::nelements ( ) const`  
`[inline]`

Give the number of cells (points including border ones).

Definition at line 438 of file image1d.hh.

Referenced by mln::image1d< T >::element(), and mln::image1d< T >::point\_at\_index().

**10.207.4.11** `template<typename T> unsigned mln::image1d< T >::ninds ( ) const` `[inline]`

Give the number of indexes.

Definition at line 483 of file image1d.hh.

**10.207.4.12** `template<typename T> const T & mln::image1d< T >::operator() ( const point1d`  
`& p ) const` `[inline]`

Read-only access to the image value located at point *p*.

Definition at line 456 of file image1d.hh.

References mln::image1d< T >::has().

**10.207.4.13** `template<typename T> T & mln::image1d< T >::operator() ( const point1d & p )`  
`[inline]`

Read-write access to the image value located at point *p*.

Definition at line 465 of file image1d.hh.

References mln::image1d< T >::has().

**10.207.4.14** `template<typename T> point1d mln::image1d< T >::point_at_index ( unsigned i )`  
`const` `[inline]`

Give the point corresponding to the offset *o*.

Definition at line 548 of file image1d.hh.

References mln::image1d< T >::nelements().

**10.207.4.15** `template<typename T > const box1d & mln::image1d< T >::vbox ( ) const`  
**[inline]**

virtual box, i.e., box including the virtual border

Definition at line 420 of file image1d.hh.

## 10.208 mln::image2d< T > Class Template Reference

Basic 2D image class.

```
#include <image2d.hh>
```

Inherits image\_primary< T, mln::box2d, image2d< T > >.

### Public Types

- typedef T & [lvalue](#)  
*Return type of read-write access.*
- typedef const T & [rvalue](#)  
*Return type of read-only access.*
- typedef [image2d](#)< tag::value\_< T > > [skeleton](#)  
*Skeleton.*
- typedef T [value](#)  
*Value associated type.*

### Public Member Functions

- const [box2d](#) & [bbox](#) () const  
*Give the bounding box domain.*
- unsigned [border](#) () const  
*Give the border thickness.*
- const T \* [buffer](#) () const  
*Give a hook to the value buffer.*
- T \* [buffer](#) ()  
*Give a hook to the value buffer.*
- int [delta\\_index](#) (const [dpoint2d](#) &dp) const  
*Give the delta-index corresponding to the delta-point dp.*

- const `box2d` & `domain` () const  
*Give the definition domain.*
- T & `element` (unsigned i)  
*Read-write access to the image value located at index *i*.*
- const T & `element` (unsigned i) const  
*Read-only access to the image value located at index *i*.*
- bool `has` (const `point2d` &p) const  
*Test if *p* is valid.*
- `image2d` (int nrows, int ncols, unsigned bdr=`border::thickness`)  
*Constructor with the numbers of rows and columns and the border thickness.*
- `image2d` ()  
*Constructor without argument.*
- `image2d` (const `box2d` &b, unsigned bdr=`border::thickness`)  
*Constructor with a box and the border thickness (default is 3).*
- unsigned `ncols` () const  
*Give the number of columns.*
- unsigned `nelements` () const  
*Give the number of elements (points including border ones).*
- unsigned `nrows` () const  
*Give the number of rows.*
- const T & `operator()` (const `point2d` &p) const  
*Read-only access to the image value located at point *p*.*
- T & `operator()` (const `point2d` &p)  
*Read-write access to the image value located at point *p*.*
- `point2d point_at_index` (unsigned i) const  
*Give the point corresponding to the index *i*.*

### 10.208.1 Detailed Description

```
template<typename T> class mln::image2d< T >
```

Basic 2D image class. The parameter T is the type of pixel values. This image class stores data in memory and has a virtual border with constant thickness around data.

Definition at line 136 of file core/image/image2d.hh.



## 10.208.2 Member Typedef Documentation

### 10.208.2.1 `template<typename T> typedef T& mln::image2d< T >::lvalue`

Return type of read-write access.

Definition at line 148 of file core/image/image2d.hh.

### 10.208.2.2 `template<typename T> typedef const T& mln::image2d< T >::rvalue`

Return type of read-only access.

Definition at line 145 of file core/image/image2d.hh.

### 10.208.2.3 `template<typename T> typedef image2d< tag::value_<T> > mln::image2d< T >::skeleton`

Skeleton.

Definition at line 152 of file core/image/image2d.hh.

### 10.208.2.4 `template<typename T> typedef T mln::image2d< T >::value`

[Value](#) associated type.

Definition at line 142 of file core/image/image2d.hh.

## 10.208.3 Constructor & Destructor Documentation

### 10.208.3.1 `template<typename T > mln::image2d< T >::image2d ( ) [inline]`

Constructor without argument.

Definition at line 394 of file core/image/image2d.hh.

### 10.208.3.2 `template<typename T > mln::image2d< T >::image2d ( int nrows, int ncols, unsigned bdr = border::thickness ) [inline]`

Constructor with the numbers of rows and columns and the border thickness.

Definition at line 400 of file core/image/image2d.hh.

References mln::make::box2d().

### 10.208.3.3 `template<typename T > mln::image2d< T >::image2d ( const box2d & b, unsigned bdr = border::thickness ) [inline]`

Constructor with a box and the border thickness (default is 3).

Definition at line 407 of file core/image/image2d.hh.

## 10.208.4 Member Function Documentation

**10.208.4.1** `template<typename T> const box2d & mln::image2d< T >::bbox ( ) const [inline]`

Give the bounding box domain.

Definition at line 433 of file core/image/image2d.hh.

**10.208.4.2** `template<typename T> unsigned mln::image2d< T >::border ( ) const [inline]`

Give the border thickness.

Definition at line 520 of file core/image/image2d.hh.

**10.208.4.3** `template<typename T> const T * mln::image2d< T >::buffer ( ) const [inline]`

Give a hook to the value buffer.

Definition at line 556 of file core/image/image2d.hh.

**10.208.4.4** `template<typename T> T * mln::image2d< T >::buffer ( ) [inline]`

Give a hook to the value buffer.

Definition at line 565 of file core/image/image2d.hh.

**10.208.4.5** `template<typename T> int mln::image2d< T >::delta_index ( const dpoint2d & dp ) const [inline]`

Give the delta-index corresponding to the delta-point dp.

Definition at line 574 of file core/image/image2d.hh.

**10.208.4.6** `template<typename T> const box2d & mln::image2d< T >::domain ( ) const [inline]`

Give the definition domain.

Definition at line 424 of file core/image/image2d.hh.

Referenced by mln::morpho::line\_gradient(), mln::make\_debug\_graph\_image(), and mln::io::txt::save().

**10.208.4.7** `template<typename T> const T & mln::image2d< T >::element ( unsigned i ) const [inline]`

Read-only access to the image value located at index i.

Definition at line 538 of file core/image/image2d.hh.

References mln::image2d< T >::nelements().

**10.208.4.8** `template<typename T> T & mln::image2d< T >::element ( unsigned i )`  
`[inline]`

Read-write access to the image value located at index *i*.

Definition at line 547 of file core/image/image2d.hh.

References mln::image2d< T >::nelements().

**10.208.4.9** `template<typename T> bool mln::image2d< T >::has ( const point2d & p ) const`  
`[inline]`

Test if *p* is valid.

Definition at line 451 of file core/image/image2d.hh.

Referenced by mln::image2d< T >::operator>(), and mln::debug::put\_word().

**10.208.4.10** `template<typename T> unsigned mln::image2d< T >::ncols ( ) const` `[inline]`

Give the number of columns.

Definition at line 508 of file core/image/image2d.hh.

**10.208.4.11** `template<typename T> unsigned mln::image2d< T >::nelements ( ) const`  
`[inline]`

Give the number of elements (points including border ones).

Definition at line 529 of file core/image/image2d.hh.

Referenced by mln::image2d< T >::element(), and mln::image2d< T >::point\_at\_index().

**10.208.4.12** `template<typename T> unsigned mln::image2d< T >::nrows ( ) const` `[inline]`

Give the number of rows.

Definition at line 499 of file core/image/image2d.hh.

**10.208.4.13** `template<typename T> T & mln::image2d< T >::operator() ( const point2d & p )`  
`[inline]`

Read-write access to the image value located at point *p*.

Definition at line 469 of file core/image/image2d.hh.

References mln::image2d< T >::has().

**10.208.4.14** `template<typename T> const T & mln::image2d< T >::operator() ( const point2d`  
`& p ) const` `[inline]`

Read-only access to the image value located at point *p*.

Definition at line 460 of file core/image/image2d.hh.

References mln::image2d< T >::has().

**10.208.4.15** `template<typename T> point2d mln::image2d< T >::point_at_index ( unsigned i )  
const [inline]`

Give the point corresponding to the index *i*.

Definition at line 584 of file core/image/image2d.hh.

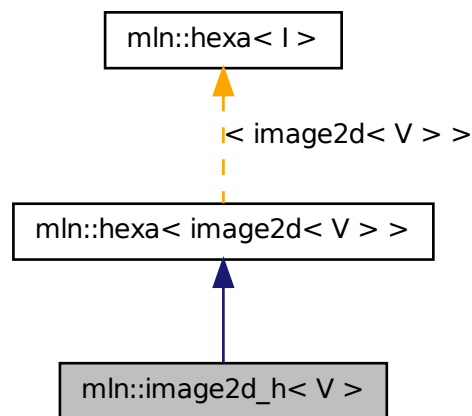
References `mln::image2d< T >::nelements()`.

## 10.209 mln::image2d\_h< V > Struct Template Reference

2d image based on an hexagonal mesh.

```
#include <image2d_h.hh>
```

Inheritance diagram for `mln::image2d_h< V >`:



### Public Types

- typedef `hexa_bkd_piter_< box2d > bkd_piter`  
*FIXME : should it be in box2d\_h? Backward Site\_Iterator associated type.*
- typedef `hexa_fwd_piter_< box2d > fwd_piter`  
*FIXME : should it be in box2d\_h? Forward Site\_Iterator associated type.*
- typedef `image2d< V >::lvalue lvalue`  
*Lvalue associated type.*
- typedef `point2d_h psite`  
*Point site type.*

- typedef `image2d< V >::rvalue rvalue`  
*Return type of read-only access.*
- typedef `hexa< tag::image_< image2d< V > > > skeleton`  
*Skeleton.*
- typedef `image2d< V >::value value`  
*Value associated type.*

## Public Member Functions

- const `box2d_h & domain () const`  
*Give the definition domain.*
- bool `has (const psite &p) const`  
*Test if  $p$  belongs to the image domain.*
- `image2d_h (int nrows, int ncols, unsigned bdr=border::thickness)`  
*Constructor with the numbers of rows and columns border thickness.*
- `lvalue operator() (const point2d_h &p)`  
*Read-write access of pixel value at hexa point site  $p$ .*
- `rvalue operator() (const point2d_h &p) const`  
*Read-only access of pixel value at hexa point site  $p$ .*

### 10.209.1 Detailed Description

`template<typename V> struct mln::image2d_h< V >`

2d image based on an hexagonal mesh.

Definition at line 50 of file image2d\_h.hh.

### 10.209.2 Member Typedef Documentation

**10.209.2.1** `typedef hexa_bkd_piter_<box2d> mln::hexa< image2d< V > >::bkd_piter`  
`[inherited]`

FIXME : should it be in box2d\_h? Backward Site\_Iterator associated type.

Definition at line 140 of file hexa.hh.

**10.209.2.2** `typedef hexa_fwd_piter_<box2d> mln::hexa< image2d< V > >::fwd_piter`  
`[inherited]`

FIXME : should it be in box2d\_h? Forward Site\_Iterator associated type.

Definition at line 136 of file hexa.hh.

**10.209.2.3 typedef image2d< V > ::lvalue mln::hexa< image2d< V > >::lvalue [inherited]**

Lvalue associated type.

Definition at line 126 of file hexa.hh.

**10.209.2.4 template<typename V > typedef point2d\_h mln::image2d\_h< V >::psite**

[Point](#) site type.

Reimplemented from [mln::hexa< image2d< V > >](#).

Definition at line 56 of file image2d\_h.hh.

**10.209.2.5 typedef image2d< V > ::rvalue mln::hexa< image2d< V > >::rvalue [inherited]**

Return type of read-only access.

Definition at line 129 of file hexa.hh.

**10.209.2.6 typedef hexa< tag::image\_<image2d< V > > > mln::hexa< image2d< V > >::skeleton [inherited]**

Skeleton.

Definition at line 120 of file hexa.hh.

**10.209.2.7 typedef image2d< V > ::value mln::hexa< image2d< V > >::value [inherited]**

Value associated type.

Definition at line 123 of file hexa.hh.

**10.209.3 Constructor & Destructor Documentation****10.209.3.1 template<typename V > mln::image2d\_h< V >::image2d\_h ( int *nrows*, int *ncols*, unsigned *bdr = border::thickness* ) [inline]**

Constructor with the numbers of rows and columns border thickness.

image2d\_h(3,6) will build this hexa image :

```
1 3 5 0 2 4 ----- 0| x x x | 2| x x x | 4| x x x
```

Definition at line 82 of file image2d\_h.hh.

**10.209.4 Member Function Documentation****10.209.4.1 const box2d\_h& mln::hexa< image2d< V > >::domain ( ) const [inherited]**

Give the definition domain.

**10.209.4.2** `bool mln::hexa< image2d< V > >::has ( const psite & p ) const` [inherited]

Test if *p* belongs to the image domain.

**10.209.4.3** `rvalue mln::hexa< image2d< V > >::operator() ( const point2d_h & p ) const` [inherited]

Read-only access of pixel value at hexa point site *p*.

**10.209.4.4** `lvalue mln::hexa< image2d< V > >::operator() ( const point2d_h & p )` [inherited]

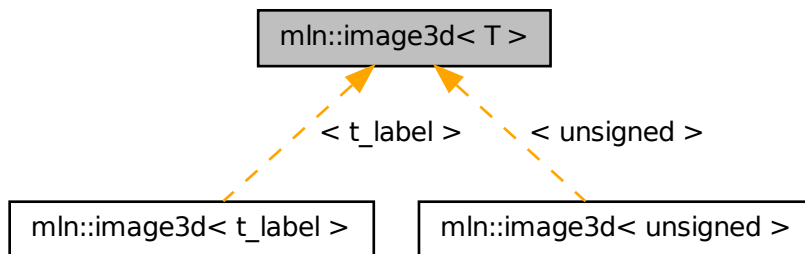
Read-write access of pixel value at hexa point site *p*.

**10.210 mln::image3d< T > Struct Template Reference**

Basic 3D image class.

```
#include <image3d.hh>
```

Inheritance diagram for mln::image3d< T >:

**Public Types**

- typedef *T* & [lvalue](#)  
*Return type of read-write access.*
- typedef const *T* & [rvalue](#)  
*Return type of read-only access.*
- typedef [image3d](#)< tag::value\_< *T* > > [skeleton](#)  
*Skeleton.*
- typedef *T* [value](#)

*Value associated type.*

## Public Member Functions

- const `box3d` & `bbox` () const  
*Give the bounding box domain.*
- unsigned `border` () const  
*Give the border thickness.*
- const T \* `buffer` () const  
*Give a hook to the value buffer.*
- T \* `buffer` ()  
*Give a hook to the value buffer.*
- int `delta_index` (const `dpoint3d` &dp) const  
*Fast Image method.*
- const `box3d` & `domain` () const  
*Give the definition domain.*
- const T & `element` (unsigned i) const  
*Read-only access to the image value located at index *i*.*
- T & `element` (unsigned i)  
*Read-write access to the image value located at index *i*.*
- bool `has` (const `point3d` &p) const  
*Test if *p* is valid.*
- `image3d` (const `box3d` &b, unsigned bdr=`border::thickness`)  
*Constructor with a box and the border thickness (default is 3).*
- `image3d` ()  
*Constructor without argument.*
- `image3d` (int nslis, int nrows, int ncols, unsigned bdr=`border::thickness`)  
*Constructor with the numbers of indexes and the border thickness.*
- unsigned `ncols` () const  
*Give the number of columns.*
- unsigned `nelements` () const  
*Give the number of cells (points including border ones).*
- unsigned `nrows` () const  
*Give the number of rows.*



- unsigned `nslis` () const  
*Give the number of slices.*
- const T & `operator()` (const `point3d` &p) const  
*Read-only access to the image value located at point p.*
- T & `operator()` (const `point3d` &p)  
*Read-write access to the image value located at point p.*
- `point3d point_at_index` (unsigned o) const  
*Give the point corresponding to the offset o.*
- const `box3d` & `vbbox` () const  
*virtual box, i.e., box including the virtual border*

### 10.210.1 Detailed Description

**template<typename T> struct mln::image3d< T >**

Basic 3D image class. The parameter T is the type of pixel values. This image class stores data in memory and has a virtual border with constant thickness around data.

Definition at line 130 of file core/image/image3d.hh.

### 10.210.2 Member Typedef Documentation

#### 10.210.2.1 template<typename T> typedef T& mln::image3d< T >::lvalue

Return type of read-write access.

Definition at line 153 of file core/image/image3d.hh.

#### 10.210.2.2 template<typename T> typedef const T& mln::image3d< T >::rvalue

Return type of read-only access.

Definition at line 150 of file core/image/image3d.hh.

#### 10.210.2.3 template<typename T> typedef image3d< tag::value\_<T> > mln::image3d< T >::skeleton

Skeleton.

Definition at line 157 of file core/image/image3d.hh.

#### 10.210.2.4 template<typename T> typedef T mln::image3d< T >::value

[Value](#) associated type.

Definition at line 147 of file core/image/image3d.hh.

### 10.210.3 Constructor & Destructor Documentation

#### 10.210.3.1 `template<typename T> mln::image3d< T >::image3d ( ) [inline]`

Constructor without argument.

Definition at line 389 of file core/image/image3d.hh.

#### 10.210.3.2 `template<typename T> mln::image3d< T >::image3d ( const box3d & b, unsigned bdr = border::thickness ) [inline]`

Constructor with a box and the border thickness (default is 3).

Definition at line 395 of file core/image/image3d.hh.

#### 10.210.3.3 `template<typename T> mln::image3d< T >::image3d ( int nslis, int nrows, int ncols, unsigned bdr = border::thickness ) [inline]`

Constructor with the numbers of indexes and the border thickness.

Definition at line 402 of file core/image/image3d.hh.

References `mln::make::box3d()`.

### 10.210.4 Member Function Documentation

#### 10.210.4.1 `template<typename T> const box3d & mln::image3d< T >::bbox ( ) const [inline]`

Give the bounding box domain.

Definition at line 428 of file core/image/image3d.hh.

#### 10.210.4.2 `template<typename T> unsigned mln::image3d< T >::border ( ) const [inline]`

Give the border thickness.

Definition at line 446 of file core/image/image3d.hh.

#### 10.210.4.3 `template<typename T> const T * mln::image3d< T >::buffer ( ) const [inline]`

Give a hook to the value buffer.

Definition at line 554 of file core/image/image3d.hh.

#### 10.210.4.4 `template<typename T> T * mln::image3d< T >::buffer ( ) [inline]`

Give a hook to the value buffer.

Definition at line 563 of file core/image/image3d.hh.

**10.210.4.5** `template<typename T> int mln::image3d< T >::delta_index ( const dpoint3d & dp ) const [inline]`

Fast [Image](#) method.

Give the offset corresponding to the delta-point `dp`.

Definition at line 572 of file `core/image/image3d.hh`.

**10.210.4.6** `template<typename T> const box3d & mln::image3d< T >::domain ( ) const [inline]`

Give the definition domain.

Definition at line 419 of file `core/image/image3d.hh`.

Referenced by `mln::accu::stat::operator==()`.

**10.210.4.7** `template<typename T> const T & mln::image3d< T >::element ( unsigned i ) const [inline]`

Read-only access to the image value located at index `i`.

Definition at line 491 of file `core/image/image3d.hh`.

References `mln::image3d< T >::nelements()`.

**10.210.4.8** `template<typename T> T & mln::image3d< T >::element ( unsigned i ) [inline]`

Read-write access to the image value located at index `i`.

Definition at line 500 of file `core/image/image3d.hh`.

References `mln::image3d< T >::nelements()`.

**10.210.4.9** `template<typename T> bool mln::image3d< T >::has ( const point3d & p ) const [inline]`

Test if `p` is valid.

Definition at line 464 of file `core/image/image3d.hh`.

Referenced by `mln::image3d< T >::operator()()`.

**10.210.4.10** `template<typename T> unsigned mln::image3d< T >::ncols ( ) const [inline]`

Give the number of columns.

Definition at line 545 of file `core/image/image3d.hh`.

**10.210.4.11** `template<typename T> unsigned mln::image3d< T >::nelements ( ) const [inline]`

Give the number of cells (points including border ones).

Definition at line 455 of file core/image/image3d.hh.

Referenced by `mln::image3d< T >::element()`, and `mln::image3d< T >::point_at_index()`.

**10.210.4.12** `template<typename T> unsigned mln::image3d< T >::nrows ( ) const [inline]`

Give the number of rows.

Definition at line 536 of file core/image/image3d.hh.

**10.210.4.13** `template<typename T> unsigned mln::image3d< T >::nslis ( ) const [inline]`

Give the number of slices.

Definition at line 527 of file core/image/image3d.hh.

**10.210.4.14** `template<typename T> T & mln::image3d< T >::operator() ( const point3d & p ) [inline]`

Read-write access to the image value located at point `p`.

Definition at line 482 of file core/image/image3d.hh.

References `mln::image3d< T >::has()`.

**10.210.4.15** `template<typename T> const T & mln::image3d< T >::operator() ( const point3d & p ) const [inline]`

Read-only access to the image value located at point `p`.

Definition at line 473 of file core/image/image3d.hh.

References `mln::image3d< T >::has()`.

**10.210.4.16** `template<typename T> point3d mln::image3d< T >::point_at_index ( unsigned o ) const [inline]`

Give the point corresponding to the offset `o`.

Definition at line 583 of file core/image/image3d.hh.

References `mln::image3d< T >::nelements()`.

**10.210.4.17** `template<typename T> const box3d & mln::image3d< T >::vbbox ( ) const [inline]`

virtual box, i.e., box including the virtual border

Definition at line 437 of file core/image/image3d.hh.

## 10.211 mln::image\_if< I, F > Struct Template Reference

[Image](#) which domain is restricted by a function 'site -> Boolean'.

```
#include <image_if.hh>
```

Inherits image\_domain\_morpher< I, p\_if< I::domain\_t, F >, image\_if< I, F > >.

## Public Types

- typedef [image\\_if](#)< tag::image\_< I >, tag::function\_< F > > [skeleton](#)  
*Skeleton.*

## Public Member Functions

- const [p\\_if](#)< typename I::domain\_t, F > & [domain](#) () const  
*Give the definition domain.*
- [image\\_if](#) (I &ima, const F &f)  
*Constructor from an image *ima* and a predicate *f*.*
- [image\\_if](#) ()  
*Constructor without argument.*
- operator [image\\_if](#)< const I, F > () const  
*Const promotion via conversion.*

### 10.211.1 Detailed Description

```
template<typename I, typename F> struct mln::image_if< I, F >
```

[Image](#) which domain is restricted by a function 'site -> Boolean'.

Definition at line 93 of file image\_if.hh.

### 10.211.2 Member Typedef Documentation

**10.211.2.1** `template<typename I, typename F> typedef image_if< tag::image_<I>, tag::function_<F> > mln::image_if< I, F >::skeleton`

Skeleton.

Definition at line 98 of file image\_if.hh.

### 10.211.3 Constructor & Destructor Documentation

**10.211.3.1** `template<typename I, typename F > mln::image_if< I, F >::image_if ( )`  
`[inline]`

Constructor without argument.

Definition at line 188 of file image\_if.hh.

**10.211.3.2** `template<typename I, typename F > mln::image_if< I, F >::image_if ( I & ima, const F & f ) [inline]`

Constructor from an image *ima* and a predicate *f*.

Definition at line 194 of file `image_if.hh`.

## 10.211.4 Member Function Documentation

**10.211.4.1** `template<typename I, typename F > const p_if< typename I::domain_t, F > & mln::image_if< I, F >::domain ( ) const [inline]`

Give the definition domain.

Definition at line 211 of file `image_if.hh`.

**10.211.4.2** `template<typename I, typename F > mln::image_if< I, F >::operator image_if< const I, F > ( ) const [inline]`

Const promotion via conversion.

Definition at line 219 of file `image_if.hh`.

## 10.212 mln::interpolated< I, F > Struct Template Reference

Makes the underlying image being accessed with floating coordinates.

```
#include <interpolated.hh>
```

Inherits `image_identity< I, I::domain_t, interpolated< I, F > >`.

### Public Types

- typedef `I::lvalue` [lvalue](#)  
*Return type of read-write access.*
- typedef `I::psite` [psite](#)  
*Point\_Site associated type.*
- typedef `I::rvalue` [rvalue](#)  
*Return type of read-only access.*
- typedef `interpolated< tag::image_< I >, F >` [skeleton](#)  
*Skeleton.*
- typedef `I::value` [value](#)  
*Value associated type.*

## Public Member Functions

- `template<typename C >`  
`bool has (const mln::algebra::vec< I::psite::dim, C > &v) const`  
*Test if a pixel value is accessible at v.*
- `interpolated (I &ima)`  
*Constructors.*
- `bool is_valid () const`  
*Test if this image has been initialized.*

### 10.212.1 Detailed Description

`template<typename I, template< class > class F> struct mln::interpolated< I, F >`

Makes the underlying image being accessed with floating coordinates.

Definition at line 84 of file interpolated.hh.

### 10.212.2 Member Typedef Documentation

**10.212.2.1** `template<typename I , template< class > class F> typedef I ::lvalue`  
`mln::interpolated< I, F >::lvalue`

Return type of read-write access.

Definition at line 98 of file interpolated.hh.

**10.212.2.2** `template<typename I , template< class > class F> typedef I ::psite mln::interpolated<`  
`I, F >::psite`

[Point\\_Site](#) associated type.

Definition at line 92 of file interpolated.hh.

**10.212.2.3** `template<typename I , template< class > class F> typedef I ::rvalue`  
`mln::interpolated< I, F >::rvalue`

Return type of read-only access.

Definition at line 101 of file interpolated.hh.

**10.212.2.4** `template<typename I , template< class > class F> typedef interpolated<`  
`tag::image_<I>, F > mln::interpolated< I, F >::skeleton`

Skeleton.

Definition at line 104 of file interpolated.hh.

**10.212.2.5** `template<typename I , template< class > class F> typedef I ::value mln::interpolated< I, F >::value`

[Value](#) associated type.

Definition at line 95 of file interpolated.hh.

### 10.212.3 Constructor & Destructor Documentation

**10.212.3.1** `template<typename I , template< class > class F> mln::interpolated< I, F >::interpolated ( I & ima ) [inline]`

Constructors.

FIXME: don't we want a 'const' here?

Definition at line 156 of file interpolated.hh.

### 10.212.4 Member Function Documentation

**10.212.4.1** `template<typename I , template< class > class F> template<typename C > bool mln::interpolated< I, F >::has ( const mln::algebra::vec< I::psite::dim, C > & v ) const [inline]`

Test if a pixel value is accessible at v.

Definition at line 189 of file interpolated.hh.

**10.212.4.2** `template<typename I , template< class > class F> bool mln::interpolated< I, F >::is_valid ( ) const [inline]`

Test if this image has been initialized.

Definition at line 180 of file interpolated.hh.

## 10.213 mln::io::dicom::dicom\_header Struct Reference

Store dicom file header.

```
#include <get_header.hh>
```

### 10.213.1 Detailed Description

Store dicom file header.

Definition at line 59 of file dicom/get\_header.hh.

## 10.214 mln::io::dump::dump\_header Struct Reference

Store dump file header.



```
#include <get_header.hh>
```

### 10.214.1 Detailed Description

Store dump file header.

Definition at line 53 of file dump/get\_header.hh.

## 10.215 mln::io::fld::fld\_header Struct Reference

Define the header structure of an AVS field data file.

```
#include <header.hh>
```

### 10.215.1 Detailed Description

Define the header structure of an AVS field data file.

Definition at line 45 of file header.hh.

## 10.216 mln::io::raw::raw\_header Struct Reference

Store raw file header.

```
#include <get_header.hh>
```

### 10.216.1 Detailed Description

Store raw file header.

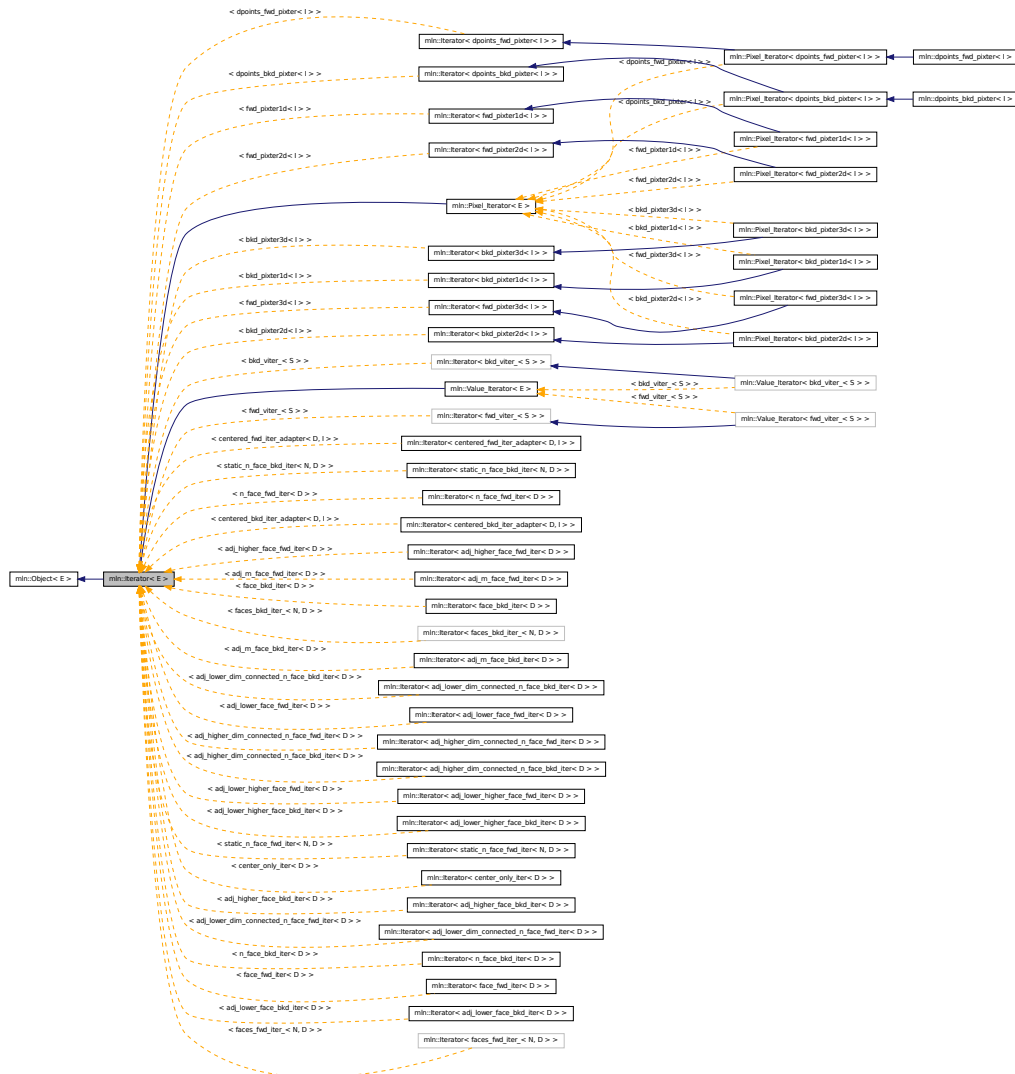
Definition at line 53 of file raw/get\_header.hh.

## 10.217 mln::Iterator< E > Struct Template Reference

Base class for implementation classes that are iterators.

```
#include <iterator.hh>
```

Inheritance diagram for `mln::Iterator< E >`:



## Public Member Functions

- `void next ()`

*Go to the next element.*

### 10.217.1 Detailed Description

`template<typename E> struct mln::Iterator< E >`

Base class for implementation classes that are iterators.

**See also**

[mln::doc::Iterator](#) for a complete documentation of this class contents.

Definition at line 75 of file iterator.hh.

**10.217.2 Member Function Documentation****10.217.2.1** `template<typename E > void mln::Iterator< E >::next ( )`

Go to the next element.

**Warning**

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

**Precondition**

The iterator is valid.

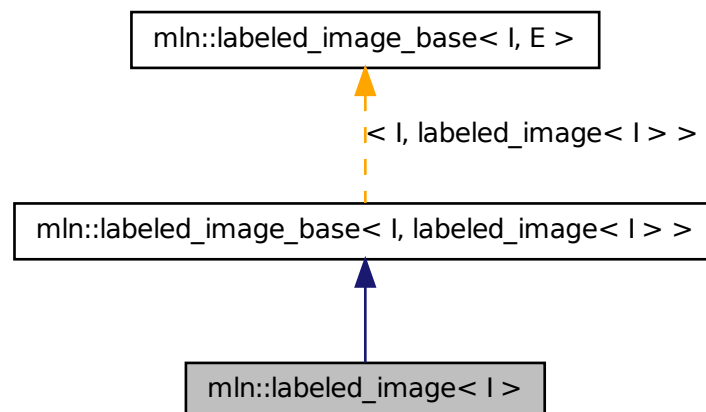
Definition at line 102 of file iterator.hh.

**10.218 mln::labeled\_image< I > Class Template Reference**

Morpher providing an improved interface for labeled image.

```
#include <labeled_image.hh>
```

Inheritance diagram for mln::labeled\_image< I >:



## Public Types

- typedef `accu::shape::bbox`< typename I::psite >::result `bbox_t`  
*Type of the bounding component bounding boxes.*
- typedef `labeled_image`< tag::image\_< I > > `skeleton`  
*Skeleton.*

## Public Member Functions

- const `bbox_t` & `bbox` (const typename I::value &label) const  
*Return the bounding box of the component `label`.*
- const `util::array`< `bbox_t` > & `bboxes` () const  
*Return the component bounding boxes.*
- I::value `nlabels` () const  
*Return the number of labels;.*
- `p_if`< mln\_box(I), fun::eq\_v2b\_expr\_< pw::value\_< I >, pw::cst\_< typename I::value > > > `subdomain` (const typename I::value &label) const  
*Return the domain of the component with label `label`.*
- `labeled_image` ()  
*Constructors*  
*Constructor without argument.*
- `labeled_image` (const I &ima, const typename I::value &nlabels)  
*Constructor from an image `ima` and the number of labels `nlabels`.*
- `labeled_image` (const I &ima, const typename I::value &nlabels, const `util::array`< mln\_box(I) > &bboxes)  
*Constructor from an image `ima`, the number of labels `nlabels` and the object bounding boxes.*
- void `relabel` (const `Function_v2v`< F > &f)  
*Relabel according to a function.*
- void `relabel` (const `Function_v2b`< F > &f)  
*Labels may be removed.*

## Protected Member Functions

- void `update_data` (const fun::i2v::array< typename I::value > &relabel\_fun)  
*Update bounding boxes information.*

## 10.218.1 Detailed Description

**template<typename I> class mln::labeled\_image< I >**

Morpher providing an improved interface for labeled image.

### Template Parameters

*I* The label image type.

This image type allows to access every site set at a given label.

This image type guaranties that labels are contiguous (from 1 to n).

Definition at line 105 of file labeled\_image.hh.

## 10.218.2 Member Typedef Documentation

**10.218.2.1 typedef accu::shape::bbox<typename I ::psite>::result mln::labeled\_image\_base< I, labeled\_image< I > >::bbox\_t [inherited]**

Type of the bounding component bounding boxes.

Definition at line 124 of file labeled\_image\_base.hh.

**10.218.2.2 template<typename I> typedef labeled\_image< tag::image\_<I> > mln::labeled\_image< I >::skeleton**

Skeleton.

Definition at line 113 of file labeled\_image.hh.

## 10.218.3 Constructor & Destructor Documentation

**10.218.3.1 template<typename I> mln::labeled\_image< I >::labeled\_image ( ) [inline]**

Constructors

Constructor without argument.

Definition at line 193 of file labeled\_image.hh.

**10.218.3.2 template<typename I> mln::labeled\_image< I >::labeled\_image ( const I & ima, const typename I::value & nlabels ) [inline]**

Constructor from an image *ima* and the number of labels *nlabels*.

Definition at line 199 of file labeled\_image.hh.

**10.218.3.3 template<typename I> mln::labeled\_image< I >::labeled\_image ( const I & ima, const typename I::value & nlabels, const util::array< mln\_box(I)> & bboxes ) [inline]**

Constructor from an image *ima*, the number of labels *nlabels* and the object bounding boxes.

Definition at line 206 of file labeled\_image.hh.

References mln::data::compute().

## 10.218.4 Member Function Documentation

**10.218.4.1** `const bbox_t& mln::labeled_image_base< I, labeled_image< I > >::bbox ( const typename I::value & label ) const` `[inherited]`

Return the bounding box of the component `label`.

**10.218.4.2** `const util::array< bbox_t > & mln::labeled_image_base< I, labeled_image< I > >::bboxes ( ) const` `[inherited]`

Return the component bounding boxes.

**10.218.4.3** `I::value mln::labeled_image_base< I, labeled_image< I > >::nlabels ( ) const` `[inherited]`

Return the number of labels;

**10.218.4.4** `void mln::labeled_image_base< I, labeled_image< I > >::relabel ( const Function_v2b< F > & f )` `[inherited]`

Labels may be removed.

This overload make sure the labeling is still contiguous.

**10.218.4.5** `void mln::labeled_image_base< I, labeled_image< I > >::relabel ( const Function_v2v< F > & f )` `[inherited]`

Relabel according to a function.

Merge or delete labels according to the given function. This method ensures that the labeling remains contiguous.

**10.218.4.6** `p_if< mln_box(I), fun::eq_v2b_expr_< pw::value_< I >, pw::cst_< typename I::value > > > mln::labeled_image_base< I, labeled_image< I > >::subdomain ( const typename I::value & label ) const` `[inherited]`

Return the domain of the component with label `label`.

**10.218.4.7** `void mln::labeled_image_base< I, labeled_image< I > >::update_data ( const fun::i2v::array< typename I::value > & relabel_fun )` `[protected, inherited]`

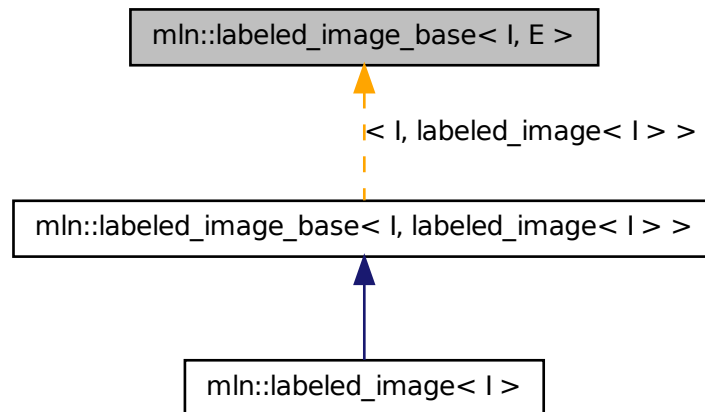
Update bounding boxes information.

## 10.219 mln::labeled\_image\_base< I, E > Class Template Reference

Base class Morpher providing an improved interface for labeled image.

```
#include <labeled_image_base.hh>
```

Inheritance diagram for mln::labeled\_image\_base< I, E >:



### Public Types

- typedef [accu::shape::bbox< typename I::psite >::result](#) [bbox\\_t](#)  
*Type of the bounding component bounding boxes.*

### Public Member Functions

- const [bbox\\_t](#) & [bbox](#) (const typename I::value &[label](#)) const  
*Return the bounding box of the component [label](#).*
- const [util::array< bbox\\_t > & bboxes](#) () const  
*Return the component bounding boxes.*
- I::value [nlabels](#) () const  
*Return the number of labels;.*
- [p\\_if< mln\\_box\(I\), fun::eq\\_v2b\\_expr\\_< pw::value\\_< I >, pw::cst\\_< typename I::value > > >](#) [subdomain](#) (const typename I::value &[label](#)) const  
*Return the domain of the component with label [label](#).*
- [labeled\\_image\\_base](#) ()

*Constructors**Constructor without argument.*

- `template<typename F >`  
`void relabel (const Function_v2v< F > &f)`  
*Relabel according to a function.*
- `template<typename F >`  
`void relabel (const Function_v2b< F > &f)`  
*Labels may be removed.*

**Protected Member Functions**

- `void update_data (const fun::i2v::array< typename I::value > &relabel_fun)`  
*Update bounding boxes information.*

**10.219.1 Detailed Description**

`template<typename I, typename E> class mln::labeled_image_base< I, E >`

Base class Morpher providing an improved interface for labeled image.

**Template Parameters**

*I* The label image type.

This image type allows to access every site set at a given label.

This image type guaranties that labels are contiguous (from 1 to n).

Definition at line 116 of file labeled\_image\_base.hh.

**10.219.2 Member Typedef Documentation**

**10.219.2.1** `template<typename I, typename E> typedef accu::shape::bbox<typename I  
::psite>::result mln::labeled_image_base< I, E >::bbox_t`

Type of the bounding component bounding boxes.

Definition at line 124 of file labeled\_image\_base.hh.

**10.219.3 Constructor & Destructor Documentation**

**10.219.3.1** `template<typename I, typename E > mln::labeled_image_base< I, E  
>::labeled_image_base ( ) [inline]`

Constructors

Constructor without argument.

Definition at line 217 of file labeled\_image\_base.hh.



### 10.219.4 Member Function Documentation

**10.219.4.1** `template<typename I, typename E > const labeled_image_base< I, E >::bbox_t & mln::labeled_image_base< I, E >::bbox ( const typename I::value & label ) const`

Return the bounding box of the component `label`.

Definition at line 305 of file `labeled_image_base.hh`.

Referenced by `mln::labeled_image_base< I, E >::subdomain()`.

**10.219.4.2** `template<typename I , typename E > const util::array< typename labeled_image_base< I, E >::bbox_t > & mln::labeled_image_base< I, E >::bboxes ( ) const`

Return the component bounding boxes.

Definition at line 313 of file `labeled_image_base.hh`.

**10.219.4.3** `template<typename I , typename E > I::value mln::labeled_image_base< I, E >::nlabels ( ) const [inline]`

Return the number of labels;

Definition at line 273 of file `labeled_image_base.hh`.

**10.219.4.4** `template<typename I , typename E > template<typename F > void mln::labeled_image_base< I, E >::relabel ( const Function_v2b< F > & f ) [inline]`

Labels may be removed.

This overload make sure the labeling is still contiguous.

Definition at line 252 of file `labeled_image_base.hh`.

References `mln::labeling::relabel_inplace()`, `mln::make::relabelfun()`, and `mln::labeled_image_base< I, E >::update_data()`.

**10.219.4.5** `template<typename I , typename E > template<typename F > void mln::labeled_image_base< I, E >::relabel ( const Function_v2v< F > & f ) [inline]`

Relabel according to a function.

Merge or delete labels according to the given function. This method ensures that the labeling remains contiguous.

Definition at line 226 of file `labeled_image_base.hh`.

References `mln::labeling::relabel_inplace()`, `mln::make::relabelfun()`, and `mln::labeled_image_base< I, E >::update_data()`.

**10.219.4.6** `template<typename I, typename E > p_if< mln_box(I), fun::eq_v2b_expr_< pw::value_< I >, pw::cst_< typename I::value > > > mln::labeled_image_base< I, E >::subdomain ( const typename I::value & label ) const`

Return the domain of the component with label `label`.

Definition at line 322 of file `labeled_image_base.hh`.

References `mln::labeled_image_base< I, E >::bbox()`.

**10.219.4.7** `template<typename I, typename E > void mln::labeled_image_base< I, E >::update_data ( const fun::i2v::array< typename I::value > & relabel_fun ) [protected]`

Update bounding boxes information.

Definition at line 281 of file `labeled_image_base.hh`.

References `mln::util::array< T >::size()`.

Referenced by `mln::labeled_image_base< I, E >::relabel()`.

## 10.220 `mln::lazy_image< I, F, B >` Struct Template Reference

[Image](#) values are computed on the fly.

```
#include <lazy_image.hh>
```

Inherits `image_identity< mln::trait::ch_value< I, F::result >::ret, I::domain_t, lazy_image< I, F, B > >`.

### Public Types

- typedef `F::result` [lvalue](#)  
*Return type of read-write access.*
- typedef `F::result` [rvalue](#)  
*Return type of read access.*
- typedef `lazy_image< tag::image_< I >, F, B >` [skeleton](#)  
*Skeleton.*

### Public Member Functions

- const [box](#)< typename `I::psite` > & [domain](#) () const  
*Return domain of `lazyd_image`.*
- bool [has](#) (const typename `I::psite` &) const  
*Test if a pixel value is accessible at `p`.*
- [lazy\\_image](#) (const `F` &`fun`, const `B` &[box](#))  
*Constructors.*

- [lazy\\_image](#) ()  
*Constructors.*
- `F::result operator()` (const typename F::input &x) const  
*Read-only access of pixel value at F::input x.*
- `lvalue operator()` (const typename I::psite &p)  
*Read and "write if possible" access of pixel value at point site p.*
- `F::result operator()` (const typename F::input &x)  
*Read and "write if possible" access of pixel value at F::input x.*
- `rvalue operator()` (const typename I::psite &p) const  
*Read-only access of pixel value at point site p.*

### 10.220.1 Detailed Description

**template<typename I, typename F, typename B> struct mln::lazy\_image< I, F, B >**

[Image](#) values are computed on the fly. The parameter `I` is the type of image. The parameter `F` is the type of function. The parameter `B` is the type of box.

This image class take a functor `fun` and a box `box`. Access to `ima(p)` where `p` include `box` return `fun(b)` lazily.

Definition at line 92 of file `lazy_image.hh`.

### 10.220.2 Member Typedef Documentation

**10.220.2.1 template<typename I, typename F, typename B> typedef F ::result mln::lazy\_image< I, F, B >::lvalue**

Return type of read-write access.

Definition at line 104 of file `lazy_image.hh`.

**10.220.2.2 template<typename I, typename F, typename B> typedef F ::result mln::lazy\_image< I, F, B >::rvalue**

Return type of read access.

Definition at line 101 of file `lazy_image.hh`.

**10.220.2.3 template<typename I, typename F, typename B> typedef lazy\_image< tag::image\_<I>, F, B > mln::lazy\_image< I, F, B >::skeleton**

Skeleton.

Definition at line 107 of file `lazy_image.hh`.

### 10.220.3 Constructor & Destructor Documentation

**10.220.3.1** `template<typename I, typename F, typename B> mln::lazy_image< I, F, B >::lazy_image ( )`

Constructors.

**10.220.3.2** `template<typename I, typename F, typename B> mln::lazy_image< I, F, B >::lazy_image ( const F & fun, const B & box ) [inline]`

Constructors.

Definition at line 161 of file lazy\_image.hh.

### 10.220.4 Member Function Documentation

**10.220.4.1** `template<typename I, typename F, typename B > const box< typename I::psite > & mln::lazy_image< I, F, B >::domain ( ) const [inline]`

Return domain of lazyd\_image.

Definition at line 226 of file lazy\_image.hh.

**10.220.4.2** `template<typename I, typename F, typename B > bool mln::lazy_image< I, F, B >::has ( const typename I::psite & p ) const [inline]`

Test if a pixel value is accessible at p.

Definition at line 175 of file lazy\_image.hh.

**10.220.4.3** `template<typename I, typename F, typename B > lazy_image< I, F, B >::rvalue mln::lazy_image< I, F, B >::operator() ( const typename I::psite & p ) const [inline]`

Read-only access of pixel value at point site p.

Definition at line 210 of file lazy\_image.hh.

**10.220.4.4** `template<typename I, typename F, typename B > F::result mln::lazy_image< I, F, B >::operator() ( const typename F::input & x ) [inline]`

Read and "write if possible" access of pixel value at F::input x.

Definition at line 197 of file lazy\_image.hh.

**10.220.4.5** `template<typename I, typename F, typename B > F::result mln::lazy_image< I, F, B >::operator() ( const typename F::input & x ) const [inline]`

Read-only access of pixel value at F::input x.

Definition at line 183 of file lazy\_image.hh.

**10.220.4.6** `template<typename I, typename F, typename B > lazy_image< I, F, B >::lvalue  
mln::lazy_image< I, F, B >::operator() ( const typename I::psite & p ) [inline]`

Read and "write if possible" access of pixel value at point site p.

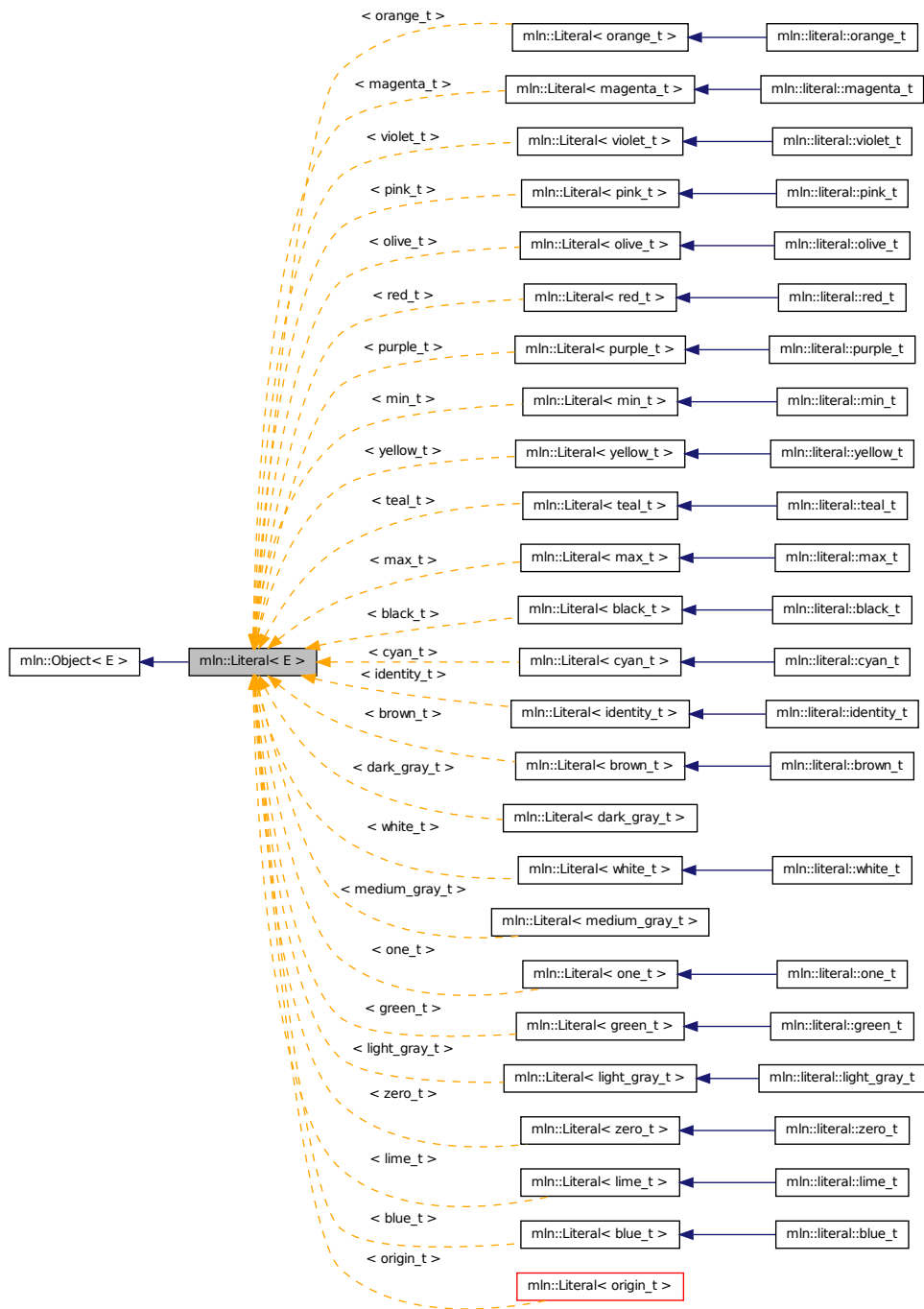
Definition at line 218 of file lazy\_image.hh.

## 10.221 mln::Literal< E > Struct Template Reference

Base class for implementation classes of literals.

```
#include <literal.hh>
```

Inheritance diagram for mln::Literal< E >:



### 10.221.1 Detailed Description

```
template<typename E> struct mln::Literal< E >
```

Base class for implementation classes of literals.

#### See also

`mln::doc::Literal` for a complete documentation of this class contents.

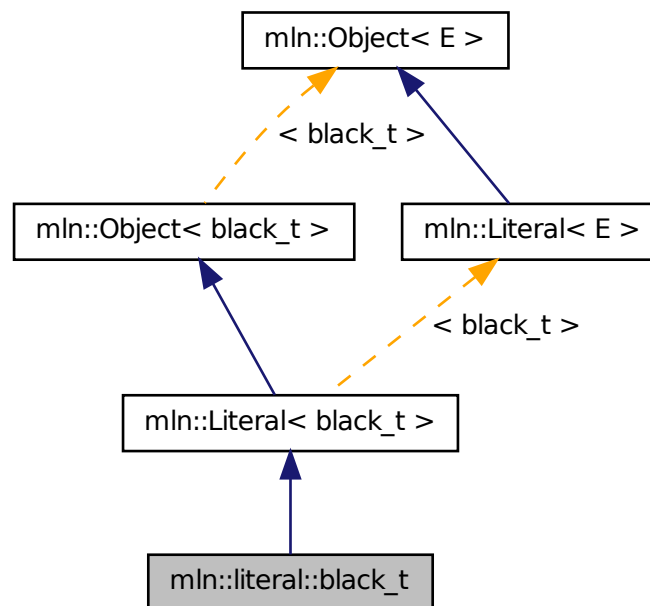
Definition at line 56 of file `literal.hh`.

## 10.222 mln::literal::black\_t Struct Reference

Type of literal black.

```
#include <black.hh>
```

Inheritance diagram for `mln::literal::black_t`:



### 10.222.1 Detailed Description

Type of literal black.

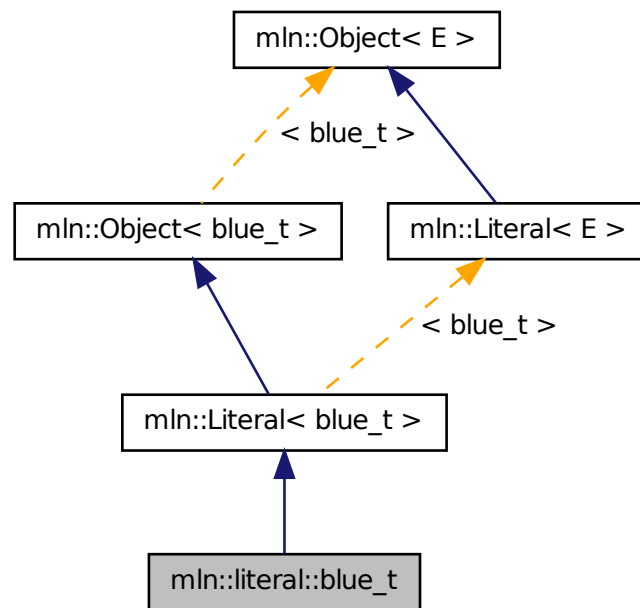
Definition at line 43 of file `black.hh`.

## 10.223 mln::literal::blue\_t Struct Reference

Type of literal blue.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::blue\_t:



### 10.223.1 Detailed Description

Type of literal blue.

Definition at line 53 of file colors.hh.

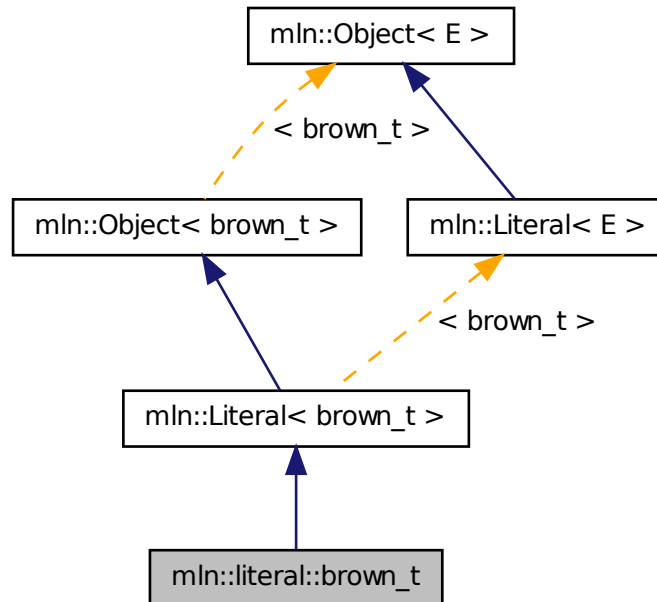
## 10.224 mln::literal::brown\_t Struct Reference

Type of literal brown.

```
#include <colors.hh>
```



Inheritance diagram for mln::literal::brown\_t:



### 10.224.1 Detailed Description

Type of literal brown.

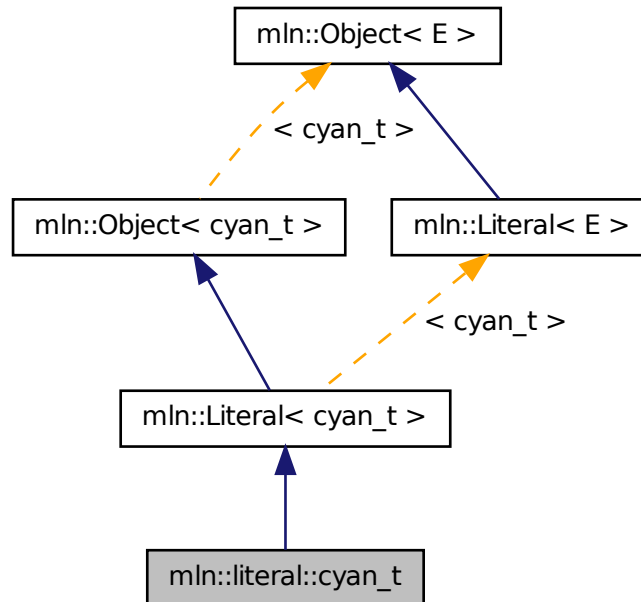
Definition at line 58 of file colors.hh.

## 10.225 mln::literal::cyan\_t Struct Reference

Type of literal cyan.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::cyan\_t:



### 10.225.1 Detailed Description

Type of literal cyan.

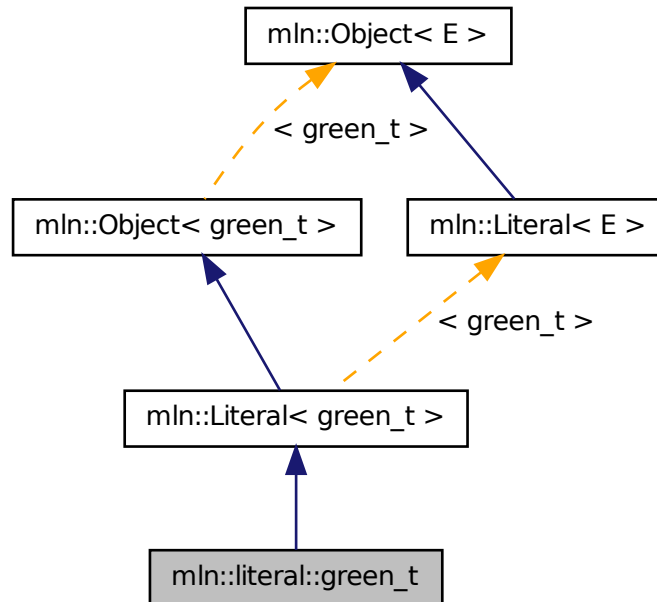
Definition at line 93 of file colors.hh.

### 10.226 mln::literal::green\_t Struct Reference

Type of literal green.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::green\_t:



### 10.226.1 Detailed Description

Type of literal green.

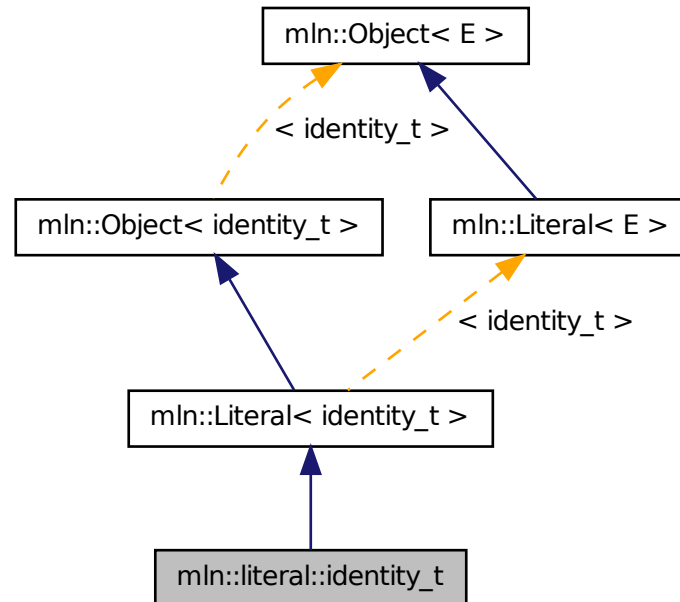
Definition at line 48 of file colors.hh.

## 10.227 mln::literal::identity\_t Struct Reference

Type of literal identity.

```
#include <identity.hh>
```

Inheritance diagram for mln::literal::identity\_t:



### 10.227.1 Detailed Description

Type of literal identity.

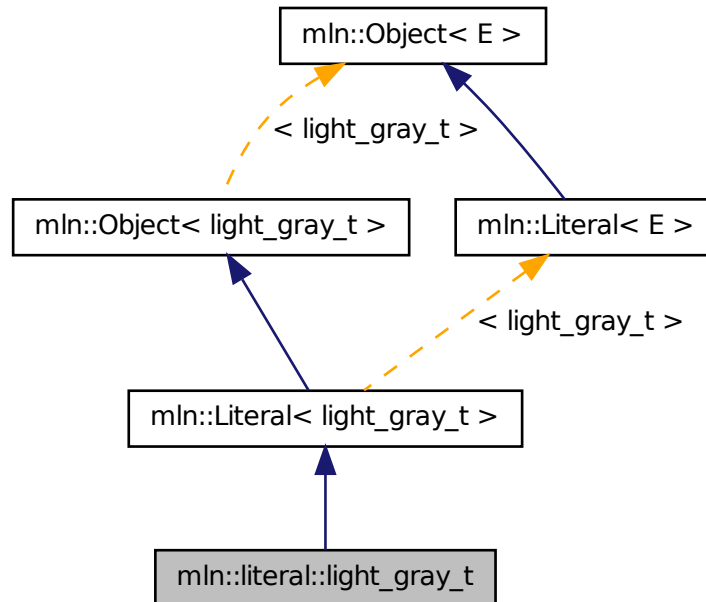
Definition at line 43 of file identity.hh.

### 10.228 mln::literal::light\_gray\_t Struct Reference

Type of literal grays.

```
#include <grays.hh>
```

Inheritance diagram for mln::literal::light\_gray\_t:



### 10.228.1 Detailed Description

Type of literal grays.

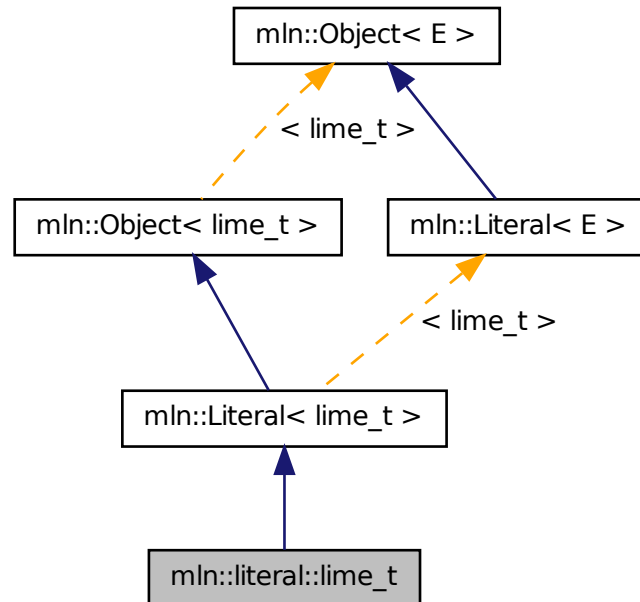
Definition at line 43 of file `grays.hh`.

## 10.229 mln::literal::lime\_t Struct Reference

Type of literal lime.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::lime\_t:



### 10.229.1 Detailed Description

Type of literal lime.

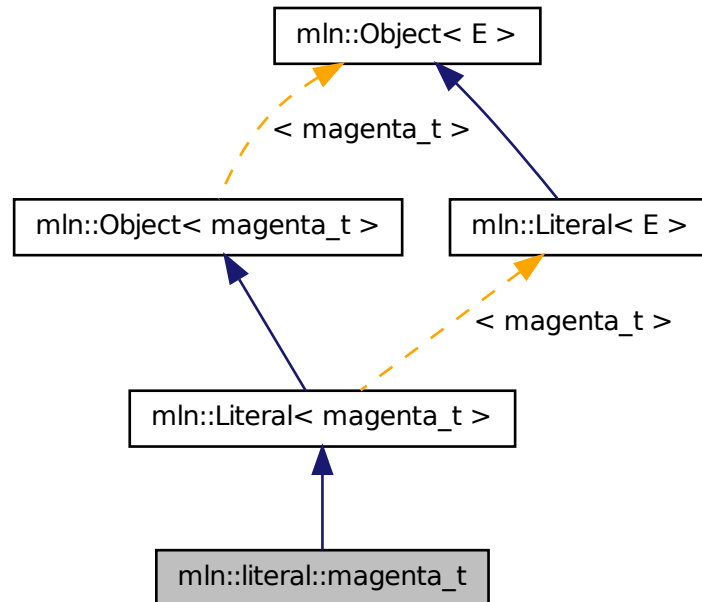
Definition at line 63 of file colors.hh.

### 10.230 mln::literal::magenta\_t Struct Reference

Type of literal magenta.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::magenta\_t:



### 10.230.1 Detailed Description

Type of literal magenta.

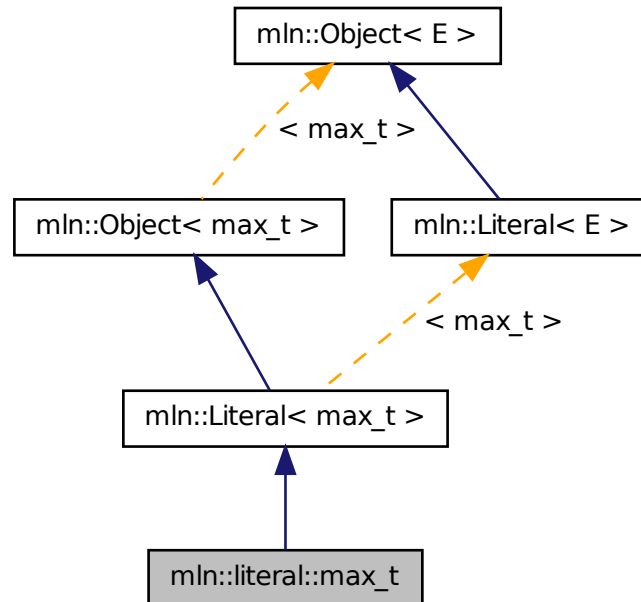
Definition at line 98 of file colors.hh.

## 10.231 mln::literal::max\_t Struct Reference

Type of literal max.

```
#include <max.hh>
```

Inheritance diagram for mln::literal::max\_t:



### 10.231.1 Detailed Description

Type of literal max.

Definition at line 44 of file literal/max.hh.

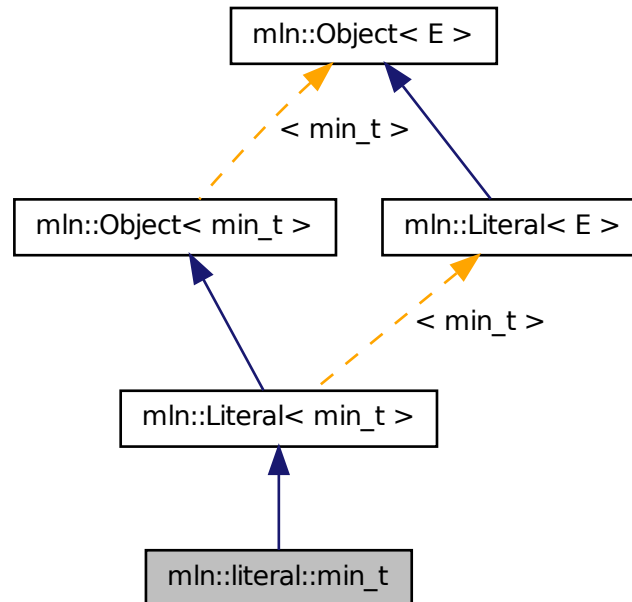
### 10.232 mln::literal::min\_t Struct Reference

Type of literal min.

```
#include <min.hh>
```



Inheritance diagram for mln::literal::min\_t:



### 10.232.1 Detailed Description

Type of literal min.

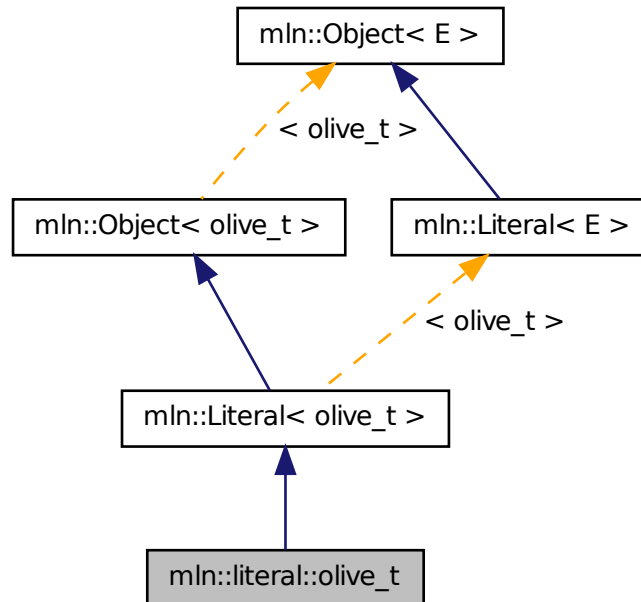
Definition at line 44 of file literal/min.hh.

## 10.233 mln::literal::olive\_t Struct Reference

Type of literal olive.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::olive\_t:



### 10.233.1 Detailed Description

Type of literal olive.

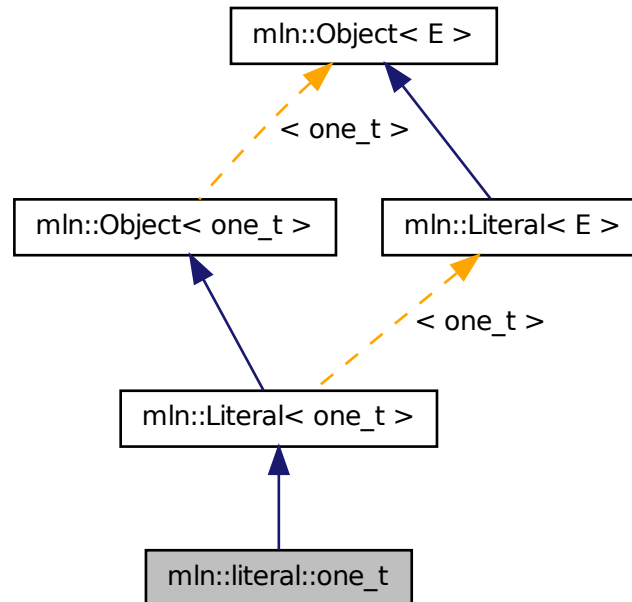
Definition at line 108 of file colors.hh.

### 10.234 mln::literal::one\_t Struct Reference

Type of literal one.

```
#include <one.hh>
```

Inheritance diagram for mln::literal::one\_t:



### 10.234.1 Detailed Description

Type of literal one.

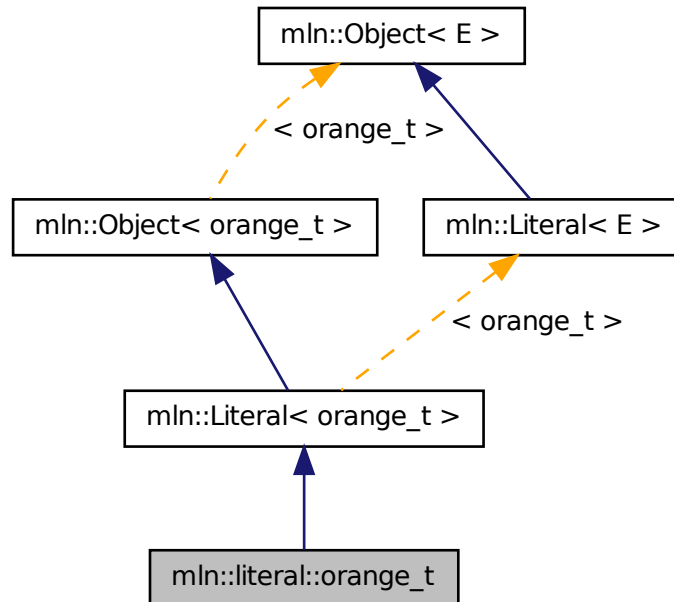
Definition at line 44 of file one.hh.

## 10.235 mln::literal::orange\_t Struct Reference

Type of literal orange.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::orange\_t:



### 10.235.1 Detailed Description

Type of literal orange.

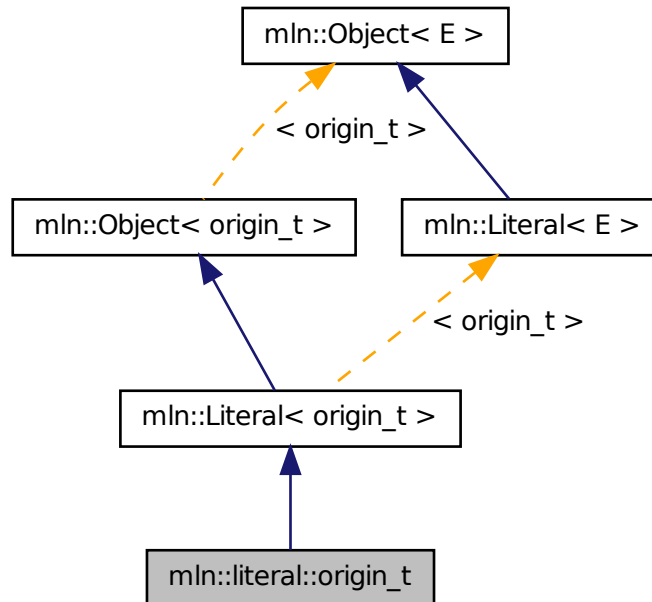
Definition at line 68 of file colors.hh.

### 10.236 mln::literal::origin\_t Struct Reference

Type of literal origin.

```
#include <origin.hh>
```

Inheritance diagram for mln::literal::origin\_t:



### 10.236.1 Detailed Description

Type of literal origin.

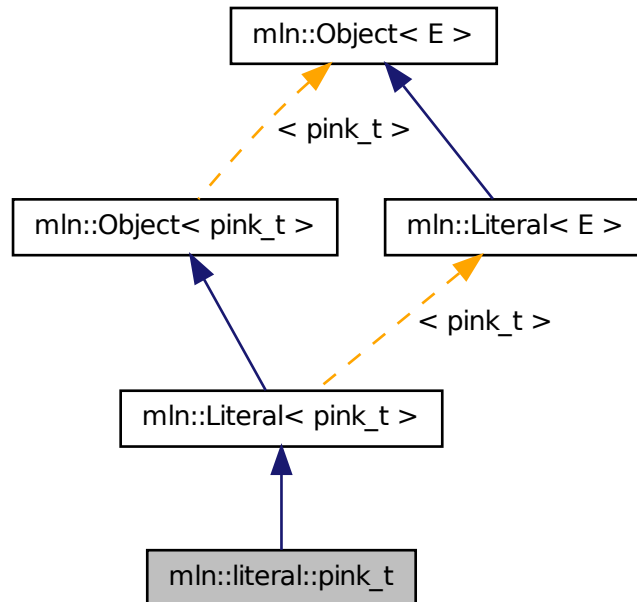
Definition at line 44 of file origin.hh.

## 10.237 mln::literal::pink\_t Struct Reference

Type of literal pink.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::pink\_t:



### 10.237.1 Detailed Description

Type of literal pink.

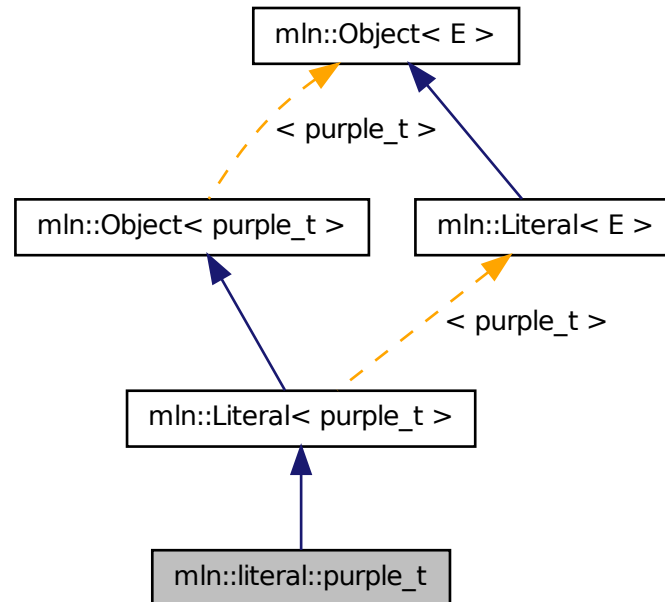
Definition at line 73 of file colors.hh.

### 10.238 mln::literal::purple\_t Struct Reference

Type of literal purple.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::purple\_t:



### 10.238.1 Detailed Description

Type of literal purple.

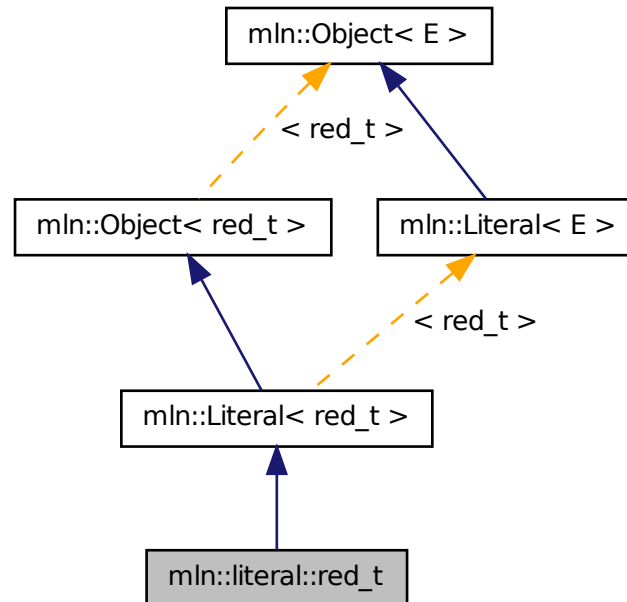
Definition at line 78 of file colors.hh.

## 10.239 mln::literal::red\_t Struct Reference

Type of literal red.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::red\_t:



### 10.239.1 Detailed Description

Type of literal red.

Definition at line 43 of file colors.hh.

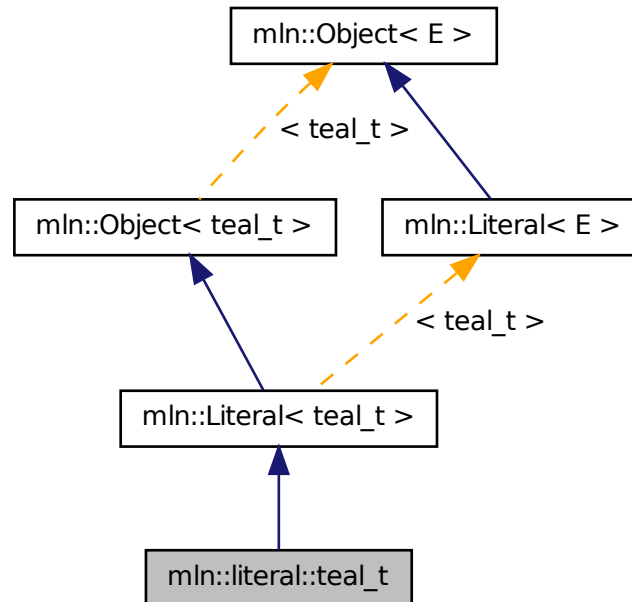
### 10.240 mln::literal::teal\_t Struct Reference

Type of literal teal.

```
#include <colors.hh>
```



Inheritance diagram for mln::literal::teal\_t:



### 10.240.1 Detailed Description

Type of literal teal.

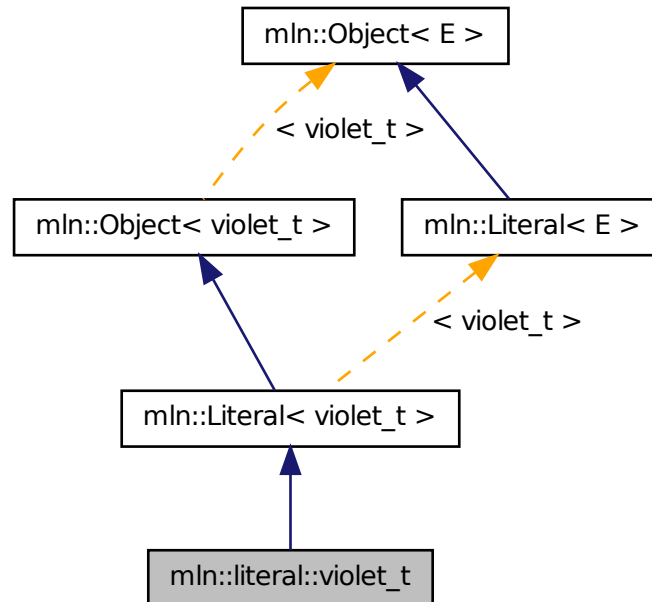
Definition at line 83 of file colors.hh.

## 10.241 mln::literal::violet\_t Struct Reference

Type of literal violet.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::violet\_t:



### 10.241.1 Detailed Description

Type of literal violet.

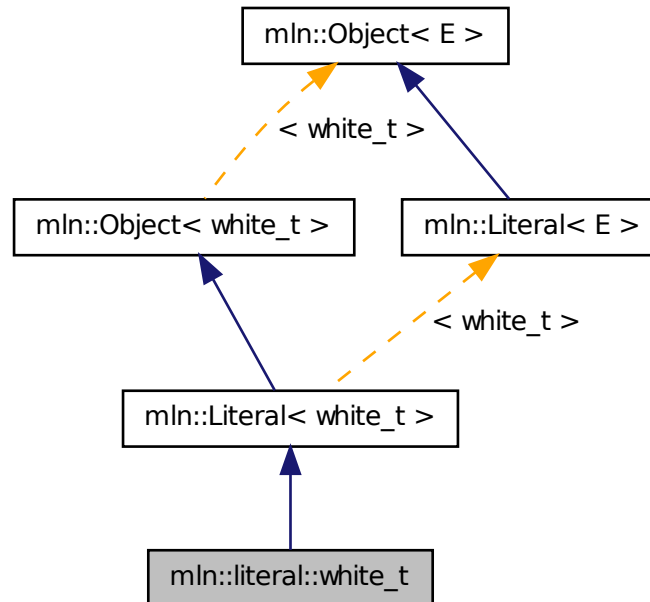
Definition at line 88 of file colors.hh.

### 10.242 mln::literal::white\_t Struct Reference

Type of literal white.

```
#include <white.hh>
```

Inheritance diagram for mln::literal::white\_t:



### 10.242.1 Detailed Description

Type of literal white.

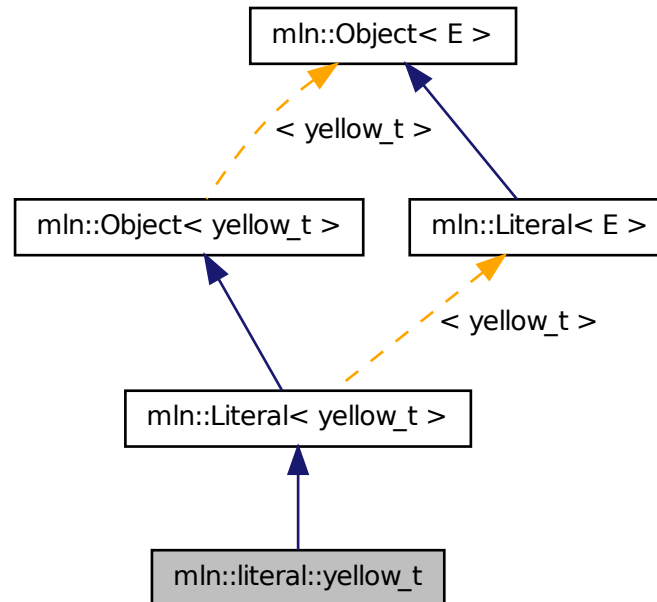
Definition at line 43 of file `white.hh`.

## 10.243 mln::literal::yellow\_t Struct Reference

Type of literal yellow.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::yellow\_t:



### 10.243.1 Detailed Description

Type of literal yellow.

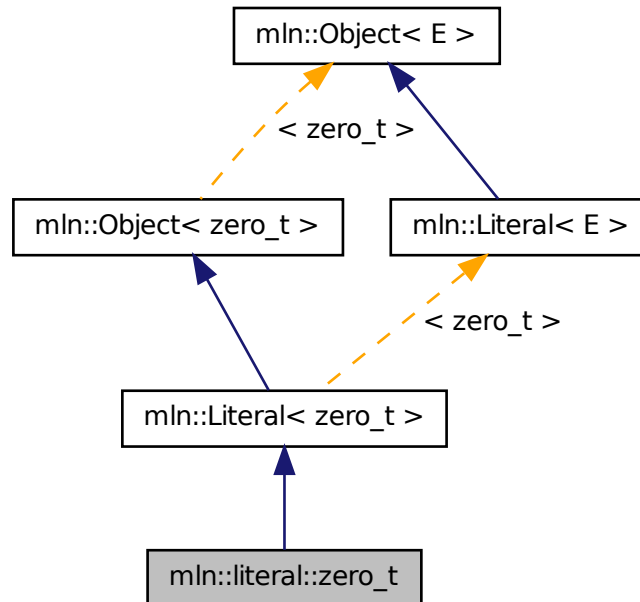
Definition at line 103 of file colors.hh.

### 10.244 mln::literal::zero\_t Struct Reference

Type of literal zero.

```
#include <zero.hh>
```

Inheritance diagram for mln::literal::zero\_t:



### 10.244.1 Detailed Description

Type of literal zero.

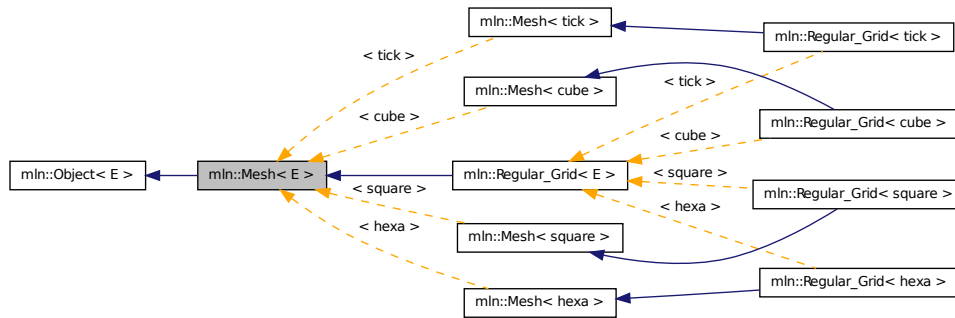
Definition at line 46 of file zero.hh.

## 10.245 mln::Mesh< E > Struct Template Reference

Base class for implementation classes of meshes.

```
#include <mesh.hh>
```

Inheritance diagram for `mln::Mesh< E >`:



### 10.245.1 Detailed Description

**template<typename E> struct mln::Mesh< E >**

Base class for implementation classes of meshes.

#### See also

`mln::doc::Mesh` for a complete documentation of this class contents.

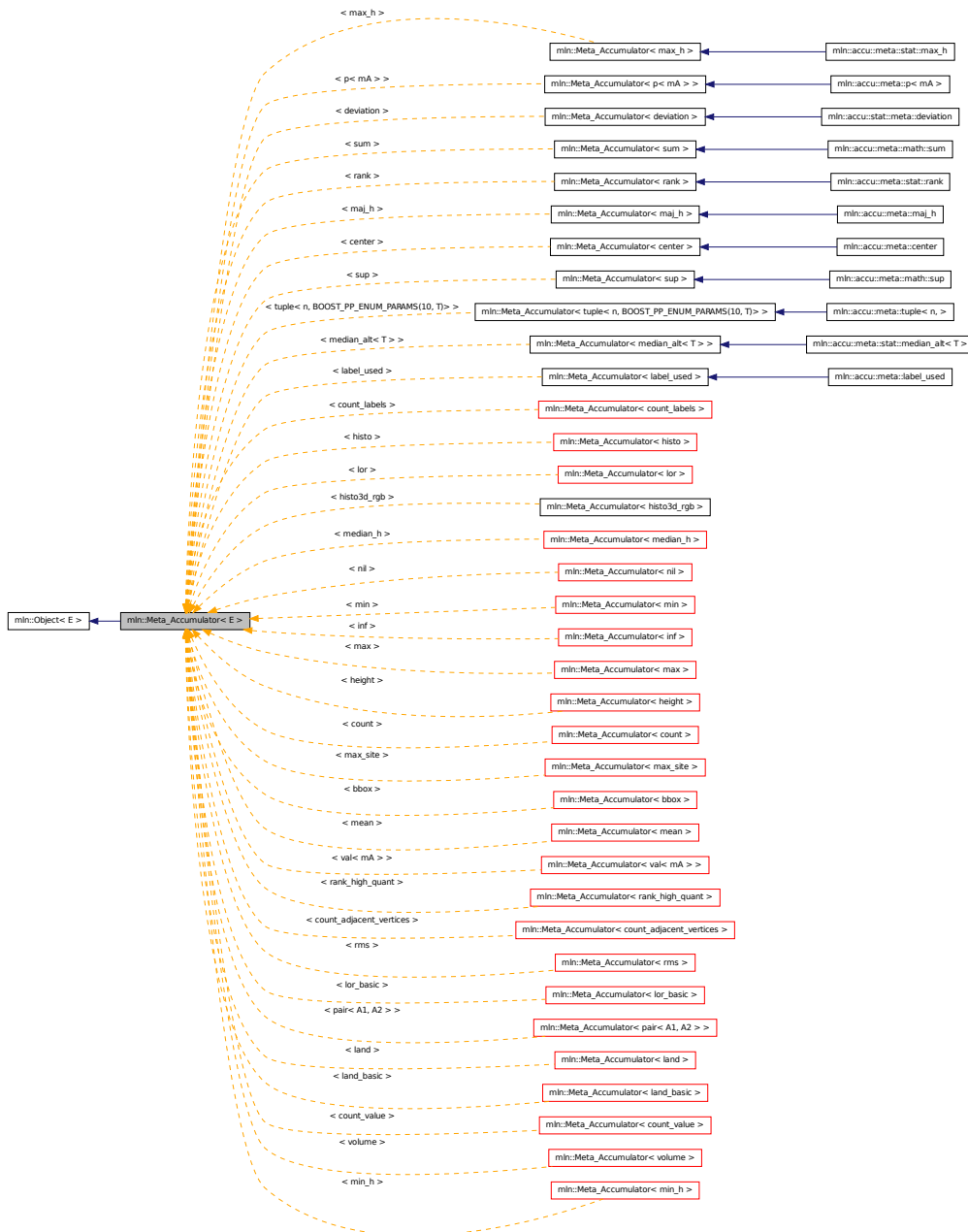
Definition at line 45 of file `mesh.hh`.

### 10.246 mln::Meta\_Accumulator< E > Struct Template Reference

Base class for implementation of meta accumulators.

```
#include <meta_accumulator.hh>
```

Inheritance diagram for mln::Meta\_Accumulator< E >:



### 10.246.1 Detailed Description

`template<typename E> struct mln::Meta_Accumulator< E >`

Base class for implementation of meta accumulators. The parameter *E* is the exact type.

**See also**

`mln::doc::Meta_Accumulator` for a complete documentation of this class contents.

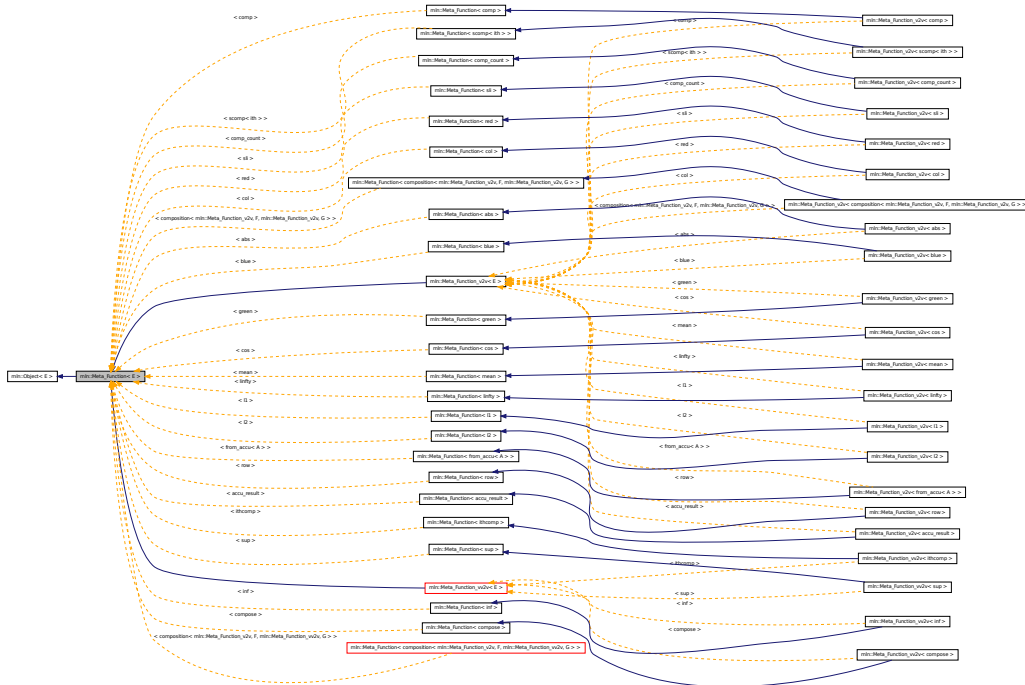
Definition at line 103 of file `meta_accumulator.hh`.

## 10.247 mln::Meta\_Function< E > Struct Template Reference

Base class for implementation of meta functions.

```
#include <meta_function.hh>
```

Inheritance diagram for `mln::Meta_Function< E >`:



### 10.247.1 Detailed Description

```
template<typename E> struct mln::Meta_Function< E >
```

Base class for implementation of meta functions. The parameter *E* is the exact type.

**See also**

`mln::doc::Meta_Function` for a complete documentation of this class contents.

Definition at line 78 of file `meta_function.hh`.

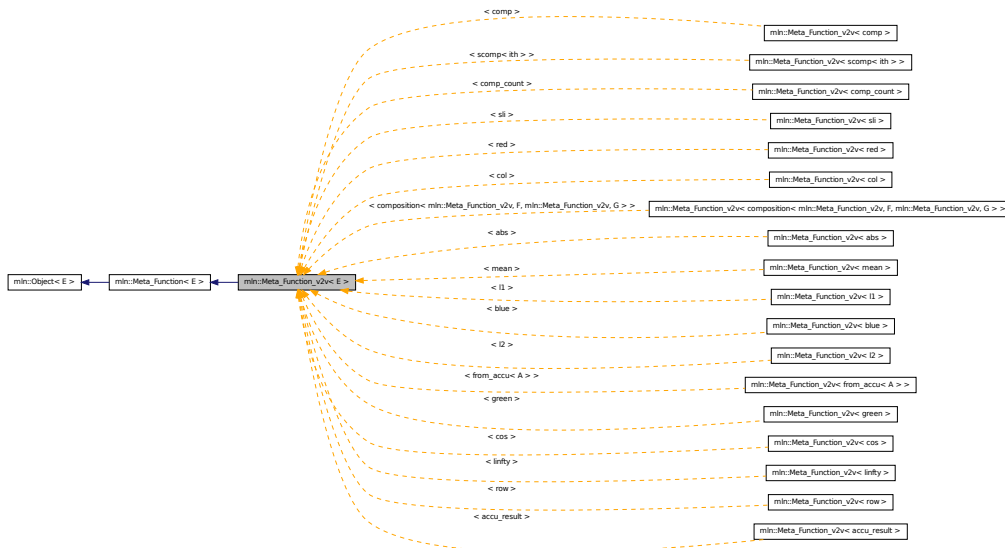


## 10.248 mln::Meta\_Function\_v2v< E > Struct Template Reference

Base class for implementation of function-objects from value to value.

```
#include <meta_function.hh>
```

Inheritance diagram for mln::Meta\_Function\_v2v< E >:



### 10.248.1 Detailed Description

```
template<typename E> struct mln::Meta_Function_v2v< E >
```

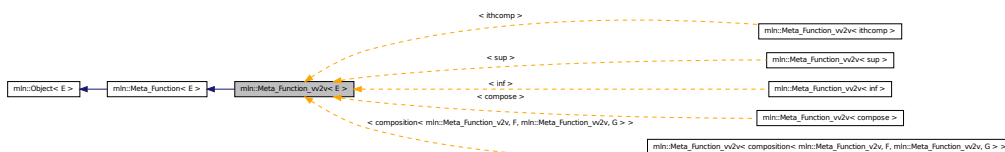
Base class for implementation of function-objects from value to value. The parameter *E* is the exact type. Definition at line 98 of file meta\_function.hh.

## 10.249 mln::Meta\_Function\_vv2v< E > Struct Template Reference

Base class for implementation of function-objects from value to value.

```
#include <meta_function.hh>
```

Inheritance diagram for mln::Meta\_Function\_vv2v< E >:



### 10.249.1 Detailed Description

```
template<typename E> struct mln::Meta_Function_vv2v< E >
```

Base class for implementation of function-objects from value to value. The parameter *E* is the exact type.  
Definition at line 120 of file meta\_function.hh.

## 10.250 mln::metal::ands< E1, E2, E3, E4, E5, E6, E7, E8 > Struct Template Reference

Ands type.

```
#include <ands.hh>
```

### 10.250.1 Detailed Description

```
template<typename E1, typename E2, typename E3, typename E4 = true_, typename E5 = true_,  
typename E6 = true_, typename E7 = true_, typename E8 = true_> struct mln::metal::ands< E1,  
E2, E3, E4, E5, E6, E7, E8 >
```

Ands type.

Definition at line 51 of file ands.hh.

## 10.251 mln::metal::converts\_to< T, U > Struct Template Reference

"converts-to" check.

```
#include <converts_to.hh>
```

Inherited by mln::metal::converts\_to< T \*, U \* >.

### 10.251.1 Detailed Description

```
template<typename T, typename U> struct mln::metal::converts_to< T, U >
```

"converts-to" check.

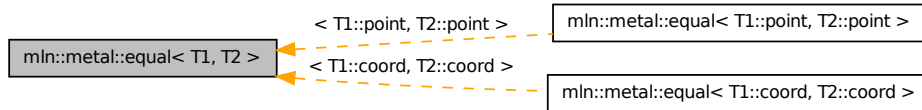
Definition at line 78 of file converts\_to.hh.

## 10.252 mln::metal::equal< T1, T2 > Struct Template Reference

Definition of a static 'equal' test.

```
#include <equal.hh>
```

Inheritance diagram for mln::metal::equal< T1, T2 >:



### 10.252.1 Detailed Description

```
template<typename T1, typename T2> struct mln::metal::equal< T1, T2 >
```

Definition of a static 'equal' test. Check whether type T1 is exactly type T2.

Definition at line 49 of file equal.hh.

## 10.253 mln::metal::goes\_to< T, U > Struct Template Reference

"goes-to" check.

```
#include <goes_to.hh>
```

### 10.253.1 Detailed Description

```
template<typename T, typename U> struct mln::metal::goes_to< T, U >
```

"goes-to" check. FIXME: Doc!

Definition at line 53 of file goes\_to.hh.

## 10.254 mln::metal::is< T, U > Struct Template Reference

"is" check.

```
#include <is.hh>
```

### 10.254.1 Detailed Description

```
template<typename T, typename U> struct mln::metal::is< T, U >
```

"is" check. Check whether T inherits from U.

Definition at line 64 of file is.hh.

## 10.255 `mln::metal::is_a< T, M >` Struct Template Reference

"is\_a" check.

```
#include <is_a.hh>
```

### 10.255.1 Detailed Description

`template<typename T, template< class > class M> struct mln::metal::is_a< T, M >`

"is\_a" check. Check whether T inherits from `_CONCEPT_M`.

Definition at line 95 of file `is_a.hh`.

## 10.256 `mln::metal::is_not< T, U >` Struct Template Reference

"is\_not" check.

```
#include <is_not.hh>
```

### 10.256.1 Detailed Description

`template<typename T, typename U> struct mln::metal::is_not< T, U >`

"is\_not" check. FIXME: Doc!

Definition at line 52 of file `is_not.hh`.

## 10.257 `mln::metal::is_not_a< T, M >` Struct Template Reference

"is\_not\_a" static Boolean expression.

```
#include <is_not_a.hh>
```

### 10.257.1 Detailed Description

`template<typename T, template< class > class M> struct mln::metal::is_not_a< T, M >`

"is\_not\_a" static Boolean expression.

Definition at line 48 of file `is_not_a.hh`.

## 10.258 `mln::mixed_neighb< W >` Class Template Reference

Adapter class from window to neighborhood.

```
#include <mixed_neighb.hh>
```

Inherits `neighb_base< W, mixed_neighb< W > >`, and `mlc_is_aW`.

## Public Types

- typedef `mixed_neighb_bkd_niter< W >` [bkd\\_niter](#)  
*Backward site iterator associated type.*
- typedef `mixed_neighb_fwd_niter< W >` [fwd\\_niter](#)  
*Forward site iterator associated type.*
- typedef [fwd\\_niter](#) `niter`  
*Site iterator associated type.*

## Public Member Functions

- [mixed\\_neighb](#) ()  
*Constructor without argument.*
- [mixed\\_neighb](#) (const W &win)  
*Constructor from a window win.*

### 10.258.1 Detailed Description

`template<typename W> class mln::mixed_neighb< W >`

Adapter class from window to neighborhood.

Definition at line 79 of file `mixed_neighb.hh`.

### 10.258.2 Member Typedef Documentation

**10.258.2.1** `template<typename W> typedef mixed_neighb_bkd_niter<W> mln::mixed_neighb< W >::bkd_niter`

Backward site iterator associated type.

Definition at line 91 of file `mixed_neighb.hh`.

**10.258.2.2** `template<typename W> typedef mixed_neighb_fwd_niter<W> mln::mixed_neighb< W >::fwd_niter`

Forward site iterator associated type.

Definition at line 88 of file `mixed_neighb.hh`.

**10.258.2.3** `template<typename W> typedef fwd_niter mln::mixed_neighb< W >::niter`

[Site](#) iterator associated type.

Definition at line 94 of file `mixed_neighb.hh`.

### 10.258.3 Constructor & Destructor Documentation

#### 10.258.3.1 `template<typename W > mln::mixed_neigh< W >::mixed_neigh ( ) [inline]`

Constructor without argument.

Definition at line 158 of file `mixed_neighb.hh`.

#### 10.258.3.2 `template<typename W > mln::mixed_neigh< W >::mixed_neigh ( const W & win ) [inline]`

Constructor from a window `win`.

Definition at line 164 of file `mixed_neighb.hh`.

## 10.259 `mln::morpho::attribute::card< I >` Class Template Reference

Cardinality accumulator class.

```
#include <card.hh>
```

Inherits `base< unsigned, card< I > >`.

### Public Member Functions

- `bool is_valid () const`  
*Check whether this `accu` is able to return a result.*
- `void take_as_init (const T &t)`  
*Take as initialization the value `t`.*
- `void take_n_times (unsigned n, const T &t)`  
*Take `n` times the value `t`.*
- `unsigned to_result () const`  
*Get the value of the accumulator.*
- `void init ()`  
*Manipulators.*

### 10.259.1 Detailed Description

```
template<typename I> class mln::morpho::attribute::card< I >
```

Cardinality accumulator class.

Definition at line 80 of file `morpho/attribute/card.hh`.

## 10.259.2 Member Function Documentation

**10.259.2.1** `template<typename I> void mln::morpho::attribute::card< I >::init ( )`  
**[inline]**

Manipulators.

Definition at line 128 of file morpho/attribute/card.hh.

**10.259.2.2** `template<typename I> bool mln::morpho::attribute::card< I >::is_valid ( ) const`  
**[inline]**

Check whether this accu is able to return a result.

Always true here.

Definition at line 197 of file morpho/attribute/card.hh.

**10.259.2.3** `void mln::Accumulator< card< I > >::take_as_init ( const T & t )` **[inherited]**

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.259.2.4** `void mln::Accumulator< card< I > >::take_n_times ( unsigned n, const T & t )`  
**[inherited]**

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.259.2.5** `template<typename I> unsigned mln::morpho::attribute::card< I >::to_result ( )`  
**const [inline]**

Get the value of the accumulator.

Definition at line 189 of file morpho/attribute/card.hh.

## 10.260 mln::morpho::attribute::count\_adjacent\_vertices< I > Struct Template Reference

Count\_Adjacent\_Vertices accumulator class.

```
#include <count_adjacent_vertices.hh>
```

Inherits base< unsigned, count\_adjacent\_vertices< I > >.

### Public Member Functions

- bool `is_valid` () const

*Check whether this accu is able to return a result.*

- void `take_as_init` (const T &t)  
*Take as initialization the value  $t$ .*
- void `take_n_times` (unsigned n, const T &t)  
*Take  $n$  times the value  $t$ .*
- unsigned `to_result` () const  
*Get the value of the accumulator.*
- void `init` ()  
*Manipulators.*

### 10.260.1 Detailed Description

**template<typename I> struct mln::morpho::attribute::count\_adjacent\_vertices< I >**

Count\_Adjacent\_Vertices accumulator class. The parameter `I` is the image type on which the accumulator of pixels is built.

Definition at line 83 of file `morpho/attribute/count_adjacent_vertices.hh`.

### 10.260.2 Member Function Documentation

**10.260.2.1** `template<typename I > void mln::morpho::attribute::count_adjacent_vertices< I >::init ( ) [inline]`

Manipulators.

Definition at line 132 of file `morpho/attribute/count_adjacent_vertices.hh`.

**10.260.2.2** `template<typename I > bool mln::morpho::attribute::count_adjacent_vertices< I >::is_valid ( ) const [inline]`

Check whether this accu is able to return a result.

Definition at line 185 of file `morpho/attribute/count_adjacent_vertices.hh`.

**10.260.2.3** `void mln::Accumulator< count_adjacent_vertices< I > >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.260.2.4** `void mln::Accumulator< count_adjacent_vertices< I > >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).



**10.260.2.5** `template<typename I > unsigned mln::morpho::attribute::count_adjacent_vertices< I >::to_result ( ) const [inline]`

Get the value of the accumulator.

Definition at line 169 of file morpho/attribute/count\_adjacent\_vertices.hh.

**10.261 mln::morpho::attribute::height< I > Struct Template Reference**

Height accumulator class.

```
#include <height.hh>
```

Inherits base< unsigned, height< I > >.

**Public Member Functions**

- unsigned `base_level` () const  
*Get base & current level of the accumulator.*
- bool `is_valid` () const  
*Check whether this accu is able to return a result.*
- void `take_as_init` (const T &t)  
*Take as initialization the value  $\tau$ .*
- void `take_n_times` (unsigned n, const T &t)  
*Take  $n$  times the value  $\tau$ .*
- unsigned `to_result` () const  
*Get the value of the accumulator.*
  
- void `init` ()  
*Manipulators.*

**10.261.1 Detailed Description**

`template<typename I> struct mln::morpho::attribute::height< I >`

Height accumulator class. The parameter  $I$  is the image type on which the accumulator of pixels is built.

Definition at line 80 of file morpho/attribute/height.hh.

## 10.261.2 Member Function Documentation

**10.261.2.1** `template<typename I> unsigned mln::morpho::attribute::height< I >::base_level ( ) const [inline]`

Get base & current level of the accumulator.

Definition at line 214 of file morpho/attribute/height.hh.

**10.261.2.2** `template<typename I> void mln::morpho::attribute::height< I >::init ( ) [inline]`

Manipulators.

Definition at line 131 of file morpho/attribute/height.hh.

**10.261.2.3** `template<typename I> bool mln::morpho::attribute::height< I >::is_valid ( ) const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 231 of file morpho/attribute/height.hh.

Referenced by `mln::morpho::attribute::height< I >::to_result()`.

**10.261.2.4** `void mln::Accumulator< height< I > >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.261.2.5** `void mln::Accumulator< height< I > >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.261.2.6** `template<typename I> unsigned mln::morpho::attribute::height< I >::to_result ( ) const [inline]`

Get the value of the accumulator.

Definition at line 204 of file morpho/attribute/height.hh.

References `mln::morpho::attribute::height< I >::is_valid()`.

## 10.262 mln::morpho::attribute::sharpness< I > Struct Template Reference

Sharpness accumulator class.

```
#include <sharpness.hh>
```

Inherits base< double, sharpness< I > >.

## Public Member Functions

- unsigned [area](#) () const  
*Give the area of the component.*
- unsigned [height](#) () const  
*Give the height.*
- bool [is\\_valid](#) () const  
*Check whether this accu is able to return a result.*
- void [take\\_as\\_init](#) (const T &t)  
*Take as initialization the value  $t$ .*
- void [take\\_n\\_times](#) (unsigned n, const T &t)  
*Take  $n$  times the value  $t$ .*
- double [to\\_result](#) () const  
*Get the value of the accumulator.*
- unsigned [volume](#) () const  
*Give the volume of the component.*
  
- void [init](#) ()  
*Manipulators.*

### 10.262.1 Detailed Description

**template<typename I> struct mln::morpho::attribute::sharpness< I >**

Sharpness accumulator class. The parameter  $I$  is the image type on which the accumulator of pixels is built.

Definition at line 80 of file sharpness.hh.

### 10.262.2 Member Function Documentation

**10.262.2.1 template<typename I > unsigned mln::morpho::attribute::sharpness< I >::area ( )  
const [inline]**

Give the area of the component.

Definition at line 190 of file sharpness.hh.

**10.262.2.2** `template<typename I > unsigned mln::morpho::attribute::sharpness< I >::height ( ) const [inline]`

Give the height.

Definition at line 206 of file sharpness.hh.

**10.262.2.3** `template<typename I > void mln::morpho::attribute::sharpness< I >::init ( ) [inline]`

Manipulators.

Definition at line 134 of file sharpness.hh.

**10.262.2.4** `template<typename I > bool mln::morpho::attribute::sharpness< I >::is_valid ( ) const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 214 of file sharpness.hh.

**10.262.2.5** `void mln::Accumulator< sharpness< I > >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.262.2.6** `void mln::Accumulator< sharpness< I > >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.262.2.7** `template<typename I > double mln::morpho::attribute::sharpness< I >::to_result ( ) const [inline]`

Get the value of the accumulator.

Definition at line 175 of file sharpness.hh.

**10.262.2.8** `template<typename I > unsigned mln::morpho::attribute::sharpness< I >::volume ( ) const [inline]`

Give the volume of the component.

Definition at line 198 of file sharpness.hh.

## 10.263 mln::morpho::attribute::sum< I, S > Class Template Reference

Suminality accumulator class.

```
#include <sum.hh>
```

Inherits base< S, sum< I, S > >.

### Public Member Functions

- bool `is_valid` () const  
*Check whether this accu is able to return a result.*
- void `set_value` (const argument &v)  
*Set the return value of the accumulator.*
- void `take_as_init` (const T &t)  
*Take as initialization the value t.*
- void `take_n_times` (unsigned n, const T &t)  
*Take n times the value t.*
- S `to_result` () const  
*Get the value of the accumulator.*
- void `untake` (const argument &v)  
*Untake a value from the accumulator.*
- void `init` ()  
*Manipulators.*

### 10.263.1 Detailed Description

```
template<typename I, typename S = typename mln::value::props< typename I ::value >::sum>  
class mln::morpho::attribute::sum< I, S >
```

Suminality accumulator class.

Definition at line 80 of file morpho/attribute/sum.hh.

### 10.263.2 Member Function Documentation

**10.263.2.1** `template<typename I, typename S > void mln::morpho::attribute::sum< I, S >::init ( ) [inline]`

Manipulators.

Definition at line 137 of file morpho/attribute/sum.hh.

References mln::literal::zero.

**10.263.2.2** `template<typename I , typename S > bool mln::morpho::attribute::sum< I, S >::is_valid ( ) const [inline]`

Check whether this accu is able to return a result.

Return always true.

Definition at line 229 of file morpho/attribute/sum.hh.

**10.263.2.3** `template<typename I , typename S > void mln::morpho::attribute::sum< I, S >::set_value ( const argument & v ) [inline]`

Set the return value of the accumulator.

Definition at line 205 of file morpho/attribute/sum.hh.

**10.263.2.4** `void mln::Accumulator< sum< I, S > >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.263.2.5** `void mln::Accumulator< sum< I, S > >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.263.2.6** `template<typename I , typename S > S mln::morpho::attribute::sum< I, S >::to_result ( ) const [inline]`

Get the value of the accumulator.

Definition at line 221 of file morpho/attribute/sum.hh.

**10.263.2.7** `template<typename I , typename S > void mln::morpho::attribute::sum< I, S >::untake ( const argument & v ) [inline]`

Untake a value from the accumulator.

Definition at line 189 of file morpho/attribute/sum.hh.

## 10.264 mln::morpho::attribute::volume< I > Struct Template Reference

Volume accumulator class.

```
#include <volume.hh>
```

Inherits base< unsigned, volume< I > >.

## Public Member Functions

- unsigned [area](#) () const  
*Give the area.*
- bool [is\\_valid](#) () const  
*Check whether this accu is able to return a result.*
- void [take\\_as\\_init](#) (const T &t)  
*Take as initialization the value  $\tau$ .*
- void [take\\_n\\_times](#) (unsigned n, const T &t)  
*Take  $n$  times the value  $\tau$ .*
- unsigned [to\\_result](#) () const  
*Get the value of the accumulator.*
  
- void [init](#) ()  
*Manipulators.*

### 10.264.1 Detailed Description

`template<typename I> struct mln::morpho::attribute::volume< I >`

Volume accumulator class. The parameter  $I$  is the image type on which the accumulator of pixels is built. Definition at line 78 of file morpho/attribute/volume.hh.

### 10.264.2 Member Function Documentation

**10.264.2.1** `template<typename I> unsigned mln::morpho::attribute::volume< I >::area ( )  
const [inline]`

Give the area.

Definition at line 202 of file morpho/attribute/volume.hh.

**10.264.2.2** `template<typename I> void mln::morpho::attribute::volume< I >::init ( )  
[inline]`

Manipulators.

Definition at line 130 of file morpho/attribute/volume.hh.

**10.264.2.3** `template<typename I> bool mln::morpho::attribute::volume< I >::is_valid ( )  
const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 210 of file morpho/attribute/volume.hh.

**10.264.2.4** `void mln::Accumulator< volume< I > >::take_as_init ( const T & t )`  
**[inherited]**

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.264.2.5** `void mln::Accumulator< volume< I > >::take_n_times ( unsigned n, const T & t )`  
**[inherited]**

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.264.2.6** `template<typename I > unsigned mln::morpho::attribute::volume< I >::to_result ( ) const` **[inline]**

Get the value of the accumulator.

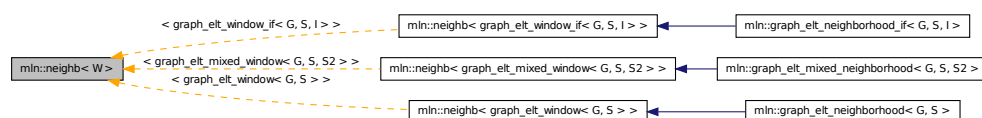
Definition at line 194 of file morpho/attribute/volume.hh.

## 10.265 mln::neighb< W > Class Template Reference

Adapter class from window to neighborhood.

```
#include <neighb.hh>
```

Inheritance diagram for `mln::neighb< W >`:



### Public Types

- typedef `neighb_bkd_niter< W >` [bkd\\_niter](#)  
*Backward site iterator associated type.*
- typedef `neighb_fwd_niter< W >` [fwd\\_niter](#)  
*Forward site iterator associated type.*
- typedef `fwd_niter` [niter](#)  
*Site iterator associated type.*



## Public Member Functions

- [neighb](#) ()  
*Constructor without argument.*
- [neighb](#) (const W &win)  
*Constructor from a window win.*

### 10.265.1 Detailed Description

`template<typename W> class mln::neighb< W >`

Adapter class from window to neighborhood.

Definition at line 76 of file mln/core/neighb.hh.

### 10.265.2 Member Typedef Documentation

**10.265.2.1** `template<typename W> typedef neighb_bkd_niter<W> mln::neighb< W >::bkd_niter`

Backward site iterator associated type.

Definition at line 87 of file mln/core/neighb.hh.

**10.265.2.2** `template<typename W> typedef neighb_fwd_niter<W> mln::neighb< W >::fwd_niter`

Forward site iterator associated type.

Definition at line 84 of file mln/core/neighb.hh.

**10.265.2.3** `template<typename W> typedef fwd_niter mln::neighb< W >::niter`

[Site](#) iterator associated type.

Definition at line 90 of file mln/core/neighb.hh.

### 10.265.3 Constructor & Destructor Documentation

**10.265.3.1** `template<typename W > mln::neighb< W >::neighb ( ) [inline]`

Constructor without argument.

Definition at line 150 of file mln/core/neighb.hh.

**10.265.3.2** `template<typename W> mln::neighb< W >::neighb ( const W & win ) [inline]`

Constructor from a window win.

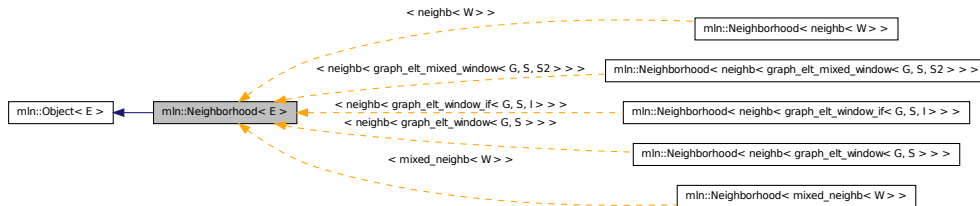
Definition at line 156 of file mln/core/neighb.hh.

## 10.266 mln::Neighborhood< E > Struct Template Reference

Base class for implementation classes that are neighborhoods.

```
#include <neighborhood.hh>
```

Inheritance diagram for mln::Neighborhood< E >:



### 10.266.1 Detailed Description

```
template<typename E> struct mln::Neighborhood< E >
```

Base class for implementation classes that are neighborhoods.

See also

[mln::doc::Neighborhood](#) for a complete documentation of this class contents.

Definition at line 66 of file core/concept/neighborhood.hh.

## 10.267 mln::Neighborhood< void > Struct Template Reference

[Neighborhood](#) category flag type.

```
#include <neighborhood.hh>
```

### 10.267.1 Detailed Description

```
template<> struct mln::Neighborhood< void >
```

[Neighborhood](#) category flag type.

Definition at line 54 of file core/concept/neighborhood.hh.

## 10.268 mln::Object< E > Struct Template Reference

Base class for almost every class defined in Milena.

```
#include <object.hh>
```

Inherited by mln::Base< E >, mln::Browsing< E >, mln::Delta\_Point\_Site< E >, mln::Function< E >, mln::Gdpoint< E >, mln::Graph< E >, mln::Image< E >, mln::io::off::internal::off\_loader< I, E >, mln::io::off::internal::off\_saver< I, E >, mln::Iterator< E >, mln::Literal< E >, mln::Mesh< E >, mln::Meta\_Accumulator< E >, mln::Meta\_Function< E >, mln::Neighborhood< E >, mln::Point\_Site< E >, mln::Proxy< E >, mln::Site< E >, mln::Site\_Set< E >, mln::Value< E >, mln::value::HSL< E >, mln::Value\_Set< E >, mln::Weighted\_Window< E >, and mln::Window< E >.

### 10.268.1 Detailed Description

`template<typename E> struct mln::Object< E >`

Base class for almost every class defined in Milena. The parameter *E* is the exact type.

Definition at line 171 of file object.hh.

## 10.269 mln::p2p\_image< I, F > Struct Template Reference

FIXME: Doc!

```
#include <p2p_image.hh>
```

Inherits image\_domain\_morpher< I, I::domain\_t, p2p\_image< I, F > >.

### Public Types

- typedef `p2p_image< tag::image_< I >, tag::function_< F > >` `skeleton`  
*Skeleton.*

### Public Member Functions

- `const I::domain_t & domain () const`  
*Give the definition domain.*
- `const F & fun () const`  
*Give the p2p function.*
- `I::rvalue operator() (const typename I::psite &p) const`  
*Read-only access to the image value located at point p.*
- `internal::morpher_lvalue_< I >::ret operator() (const typename I::psite &p)`  
*Read-write access to the image value located at point p.*
- `p2p_image (I &ima, const F &f)`  
*Constructor from an image *ima* and a predicate *f*.*
- `p2p_image ()`  
*Constructor without argument.*

### 10.269.1 Detailed Description

`template<typename I, typename F> struct mln::p2p_image< I, F >`

FIXME: Doc!

Definition at line 90 of file p2p\_image.hh.

### 10.269.2 Member Typedef Documentation

**10.269.2.1** `template<typename I, typename F> typedef p2p_image< tag::image_<I>, tag::function_<F> > mln::p2p_image< I, F >::skeleton`

Skeleton.

Definition at line 95 of file p2p\_image.hh.

### 10.269.3 Constructor & Destructor Documentation

**10.269.3.1** `template<typename I, typename F > mln::p2p_image< I, F >::p2p_image ( )`  
`[inline]`

Constructor without argument.

Definition at line 178 of file p2p\_image.hh.

**10.269.3.2** `template<typename I, typename F > mln::p2p_image< I, F >::p2p_image ( I & ima, const F & f )` `[inline]`

Constructor from an image *ima* and a predicate *f*.

Definition at line 184 of file p2p\_image.hh.

### 10.269.4 Member Function Documentation

**10.269.4.1** `template<typename I, typename F > const I::domain_t & mln::p2p_image< I, F >::domain ( ) const` `[inline]`

Give the definition domain.

Definition at line 201 of file p2p\_image.hh.

**10.269.4.2** `template<typename I, typename F > const F & mln::p2p_image< I, F >::fun ( ) const` `[inline]`

Give the p2p function.

Definition at line 210 of file p2p\_image.hh.

#### 10.269.4.3 `template<typename I, typename F > I::rvalue mln::p2p_image< I, F >::operator() ( const typename I::psite & p ) const [inline]`

Read-only access to the image value located at point `p`.

Definition at line 219 of file `p2p_image.hh`.

#### 10.269.4.4 `template<typename I, typename F > internal::morpher_lvalue< I >::ret mln::p2p_image< I, F >::operator() ( const typename I::psite & p ) [inline]`

Read-write access to the image value located at point `p`.

Definition at line 229 of file `p2p_image.hh`.

## 10.270 mln::p\_array< P > Class Template Reference

Multi-set of sites.

```
#include <p_array.hh>
```

Inherits `site_set_base< P, p_array< P > >`.

### Public Types

- typedef `p_indexed_bkd_piter< self_ > bkd_piter`  
*Backward Site\_Iterator associated type.*
- typedef `P element`  
*Element associated type.*
- typedef `p_indexed_fwd_piter< self_ > fwd_piter`  
*Forward Site\_Iterator associated type.*
- typedef `P i_element`  
*Insertion element associated type.*
- typedef `fwd_piter piter`  
*Site\_Iterator associated type.*
- typedef `p_indexed_psite< self_ > psite`  
*Psite associated type.*

### Public Member Functions

- `p_array< P > & append (const P &p)`  
*Append a point `p`.*
- `p_array< P > & append (const p_array< P > &other)`  
*Append an array `other` of points.*

- void **change** (const **psite** &p, const P &new\_p)  
*Change site p into new\_p.*
- void **clear** ()  
*Clear this set.*
- bool **has** (const util::index &i) const  
*Test is index i belongs to this site set.*
- bool **has** (const **psite** &p) const  
*Test is p belongs to this site set.*
- void **insert** (const P &p)  
*Insert a point p (equivalent as 'append').*
- bool **is\_valid** () const  
*Test this set validity so returns always true.*
- std::size\_t **memory\_size** () const  
*Return the size of this site set in memory.*
- unsigned **nsites** () const  
*Give the number of sites.*
- const P & **operator[]** (const util::index &i) const  
*Return the i-th element.*
- P & **operator[]** (unsigned i)  
*Return the i-th site (mutable).*
- const P & **operator[]** (unsigned i) const  
*Return the i-th site (constant).*
- **p\_array** ()  
*Constructor.*
- **p\_array** (const std::vector< P > &vect)  
*Constructor from a vector vect.*
- void **reserve** (size\_type n)  
*Reserve n cells.*
- void **resize** (size\_t size)  
*Update the size of this array.*
- const std::vector< P > & **std\_vector** () const  
*Return the corresponding std::vector of points.*

## 10.270.1 Detailed Description

`template<typename P> class mln::p_array< P >`

Multi-set of sites. [Site](#) set class based on `std::vector`.

Definition at line 84 of file `p_array.hh`.

## 10.270.2 Member Typedef Documentation

**10.270.2.1** `template<typename P> typedef p_indexed_bkd_piter<self_> mln::p_array< P >::bkd_piter`

Backward [Site\\_Iterator](#) associated type.

Definition at line 100 of file `p_array.hh`.

**10.270.2.2** `template<typename P> typedef P mln::p_array< P >::element`

Element associated type.

Definition at line 91 of file `p_array.hh`.

**10.270.2.3** `template<typename P> typedef p_indexed_fwd_piter<self_> mln::p_array< P >::fwd_piter`

Forward [Site\\_Iterator](#) associated type.

Definition at line 97 of file `p_array.hh`.

**10.270.2.4** `template<typename P> typedef P mln::p_array< P >::i_element`

Insertion element associated type.

Definition at line 141 of file `p_array.hh`.

**10.270.2.5** `template<typename P> typedef fwd_piter mln::p_array< P >::piter`

[Site\\_Iterator](#) associated type.

Definition at line 103 of file `p_array.hh`.

**10.270.2.6** `template<typename P> typedef p_indexed_psite<self_> mln::p_array< P >::psite`

Psite associated type.

Definition at line 94 of file `p_array.hh`.

### 10.270.3 Constructor & Destructor Documentation

#### 10.270.3.1 `template<typename P> mln::p_array<P>::p_array( ) [inline]`

Constructor.

Definition at line 340 of file p\_array.hh.

#### 10.270.3.2 `template<typename P> mln::p_array<P>::p_array( const std::vector<P> & vect ) [inline]`

Constructor from a vector `vect`.

Definition at line 346 of file p\_array.hh.

### 10.270.4 Member Function Documentation

#### 10.270.4.1 `template<typename P> p_array<P> & mln::p_array<P>::append( const P & p ) [inline]`

Append a point `p`.

Definition at line 408 of file p\_array.hh.

Referenced by `mln::convert::to_p_array()`.

#### 10.270.4.2 `template<typename P> p_array<P> & mln::p_array<P>::append( const p_array<P> & other ) [inline]`

Append an array `other` of points.

Definition at line 425 of file p\_array.hh.

References `mln::p_array<P>::std_vector()`.

#### 10.270.4.3 `template<typename P> void mln::p_array<P>::change( const psite & p, const P & new_p ) [inline]`

Change site `p` into `new_p`.

Definition at line 472 of file p\_array.hh.

References `mln::p_array<P>::has()`.

#### 10.270.4.4 `template<typename P> void mln::p_array<P>::clear( ) [inline]`

Clear this set.

Definition at line 436 of file p\_array.hh.

#### 10.270.4.5 `template<typename P> bool mln::p_array<P>::has( const psite & p ) const [inline]`

Test if `p` belongs to this site set.



Definition at line 362 of file p\_array.hh.

Referenced by mln::p\_array< P >::change(), and mln::p\_array< P >::operator[ ]().

#### 10.270.4.6 `template<typename P > bool mln::p_array< P >::has ( const util::index & i ) const [inline]`

Test is index *i* belongs to this site set.

Definition at line 375 of file p\_array.hh.

References mln::p\_array< P >::nsites().

#### 10.270.4.7 `template<typename P > void mln::p_array< P >::insert ( const P & p ) [inline]`

Insert a point *p* (equivalent as 'append').

Definition at line 417 of file p\_array.hh.

#### 10.270.4.8 `template<typename P > bool mln::p_array< P >::is_valid ( ) const [inline]`

Test this set validity so returns always true.

Definition at line 383 of file p\_array.hh.

#### 10.270.4.9 `template<typename P > std::size_t mln::p_array< P >::memory_size ( ) const [inline]`

Return the size of this site set in memory.

Definition at line 481 of file p\_array.hh.

References mln::p\_array< P >::nsites().

#### 10.270.4.10 `template<typename P > unsigned mln::p_array< P >::nsites ( ) const [inline]`

Give the number of sites.

Definition at line 400 of file p\_array.hh.

Referenced by mln::registration::get\_rot(), mln::p\_array< P >::has(), mln::p\_array< P >::memory\_size(), and mln::p\_array< P >::operator[ ]().

#### 10.270.4.11 `template<typename P > P & mln::p_array< P >::operator[ ] ( unsigned i ) [inline]`

Return the *i*-th site (mutable).

Definition at line 463 of file p\_array.hh.

References mln::p\_array< P >::nsites().

**10.270.4.12** `template<typename P> const P & mln::p_array< P >::operator[] ( const util::index & i ) const [inline]`

Return the *i*-th element.

Definition at line 391 of file `p_array.hh`.

References `mln::p_array< P >::has()`.

**10.270.4.13** `template<typename P> const P & mln::p_array< P >::operator[] ( unsigned i ) const [inline]`

Return the *i*-th site (constant).

Definition at line 454 of file `p_array.hh`.

References `mln::p_array< P >::nsites()`.

**10.270.4.14** `template<typename P> void mln::p_array< P >::reserve ( size_type n ) [inline]`

Reserve *n* cells.

Definition at line 354 of file `p_array.hh`.

Referenced by `mln::convert::to_p_array()`.

**10.270.4.15** `template<typename P> void mln::p_array< P >::resize ( size_t size ) [inline]`

Update the size of this array.

Definition at line 445 of file `p_array.hh`.

**10.270.4.16** `template<typename P> const std::vector< P > & mln::p_array< P >::std_vector ( ) const [inline]`

Return the corresponding `std::vector` of points.

Definition at line 489 of file `p_array.hh`.

Referenced by `mln::p_array< P >::append()`.

## 10.271 `mln::p_centered< W >` Class Template Reference

[Site](#) set corresponding to a window centered on a site.

```
#include <p_centered.hh>
```

Inherits `site_set_base< W::psite, p_centered< W >>`, and `mlc_is_aW`.

### Public Types

- `typedef p_centered_piter< W > bkd_piter`  
*Backward [Site Iterator](#) associated type.*

- typedef [psite element](#)  
*Element associated type.*
- typedef [p\\_centered\\_piter](#)< W > [fwd\\_piter](#)  
*Forward [Site\\_Iterator](#) associated type.*
- typedef [fwd\\_piter piter](#)  
*[Site\\_Iterator](#) associated type.*
- typedef W::psite [psite](#)  
*Psite associated type.*
- typedef W::site [site](#)  
*Site associated type.*

## Public Member Functions

- const W::psite & [center](#) () const  
*Give the center of this site set.*
- template<typename P >  
bool [has](#) (const P &p) const  
*Test if *p* belongs to the box.*
- bool [is\\_valid](#) () const  
*Test if this site set is initialized.*
- std::size\_t [memory\\_size](#) () const  
*Return the size of this site set in memory.*
- [p\\_centered](#) (const W &win, const typename W::psite &c)  
*Constructor from a window *win* and a center *c*.*
- [p\\_centered](#) ()  
*Constructor without argument.*
- const W & [window](#) () const  
*Give the window this site set is defined upon.*

### 10.271.1 Detailed Description

template<typename W> class mln::p\_centered< W >

[Site](#) set corresponding to a window centered on a site.

Definition at line 77 of file p\_centered.hh.

## 10.271.2 Member Typedef Documentation

### 10.271.2.1 `template<typename W> typedef p_centered_piter<W> mln::p_centered< W >::bkd_piter`

Backward [Site\\_Iterator](#) associated type.

Definition at line 97 of file `p_centered.hh`.

### 10.271.2.2 `template<typename W> typedef psite mln::p_centered< W >::element`

Element associated type.

Definition at line 90 of file `p_centered.hh`.

### 10.271.2.3 `template<typename W> typedef p_centered_piter<W> mln::p_centered< W >::fwd_piter`

Forward [Site\\_Iterator](#) associated type.

Definition at line 94 of file `p_centered.hh`.

### 10.271.2.4 `template<typename W> typedef fwd_piter mln::p_centered< W >::piter`

[Site\\_Iterator](#) associated type.

Definition at line 100 of file `p_centered.hh`.

### 10.271.2.5 `template<typename W> typedef W ::psite mln::p_centered< W >::psite`

Psite associated type.

Definition at line 83 of file `p_centered.hh`.

### 10.271.2.6 `template<typename W> typedef W ::site mln::p_centered< W >::site`

[Site](#) associated type.

Definition at line 86 of file `p_centered.hh`.

## 10.271.3 Constructor & Destructor Documentation

### 10.271.3.1 `template<typename W > mln::p_centered< W >::p_centered ( ) [inline]`

Constructor without argument.

Definition at line 182 of file `p_centered.hh`.

### 10.271.3.2 `template<typename W > mln::p_centered< W >::p_centered ( const W & win, const typename W::psite & c ) [inline]`

Constructor from a window `win` and a center `c`.

Definition at line 188 of file p\_centered.hh.

References mln::p\_centered< W >::is\_valid().

### 10.271.4 Member Function Documentation

**10.271.4.1** `template<typename W > const W::psite & mln::p_centered< W >::center ( ) const [inline]`

Give the center of this site set.

Definition at line 216 of file p\_centered.hh.

**10.271.4.2** `template<typename W > template<typename P > bool mln::p_centered< W >::has ( const P & p ) const [inline]`

Test if *p* belongs to the box.

Definition at line 199 of file p\_centered.hh.

References mln::p\_centered< W >::is\_valid().

**10.271.4.3** `template<typename W > bool mln::p_centered< W >::is_valid ( ) const [inline]`

Test if this site set is initialized.

Definition at line 175 of file p\_centered.hh.

Referenced by mln::p\_centered< W >::has(), and mln::p\_centered< W >::p\_centered().

**10.271.4.4** `template<typename W > std::size_t mln::p_centered< W >::memory_size ( ) const [inline]`

Return the size of this site set in memory.

Definition at line 208 of file p\_centered.hh.

**10.271.4.5** `template<typename W > const W & mln::p_centered< W >::window ( ) const [inline]`

Give the window this site set is defined upon.

Definition at line 224 of file p\_centered.hh.

## 10.272 mln::p\_complex< D, G > Class Template Reference

A complex psite set based on the *N*-faces of a complex of dimension *D* (a *D*-complex).

```
#include <p_complex.hh>
```

Inherits site\_set\_base\_< complex\_psite< D, G >, p\_complex< D, G > >.

## Public Types

- typedef `super_::site element`  
*Associated types.*
- typedef `complex_psite< D, G > psite`  
*Point\_Site associated type.*
- typedef `p_complex_fwd_piter_< D, G > fwd_piter`  
*Forward Site\_Iterator associated type.*
- typedef `p_complex_bkd_piter_< D, G > bkd_piter`  
*Backward Site\_Iterator associated type.*
- typedef `fwd_piter piter`  
*Site\_Iterator associated type.*

## Public Member Functions

- bool `has (const psite &p) const`  
*Does this site set has p?*
- bool `is_valid () const`  
*Is this site set valid?*
- unsigned `nfaces () const`  
*Return the number of faces in the complex.*
- unsigned `nfaces_of_dim (unsigned n) const`  
*Return the number of n-faces in the complex.*
- unsigned `nsites () const`  
*Return The number of sites of the set, i.e., the number of faces.*
- `p_complex (const topo::complex< D > &cplx, const G &geom)`  
*Construct a complex psite set from a complex.*
- `topo::complex< D > & cplx () const`  
*Accessors.*
- `topo::complex< D > & cplx ()`  
*Return the complex associated to the p\_complex domain (mutable version).*
- `const G & geom () const`  
*Return the geometry of the complex.*

### 10.272.1 Detailed Description

**template<unsigned D, typename G> class mln::p\_complex< D, G >**

A complex psite set based on the N-faces of a complex of dimension D (a `D-complex`).

#### Template Parameters

*D* The dimension of the complex.

*G* A function object type, associating localization information (geometry) to each face of the complex.

#### See also

[mln::geom::complex\\_geometry](#). A complex [psite set](#) based on the N-faces of a complex.

Definition at line 116 of file `p_complex.hh`.

### 10.272.2 Member Typedef Documentation

**10.272.2.1 template<unsigned D, typename G> typedef p\_complex\_bkd\_piter\_<D, G>  
mln::p\_complex< D, G >::bkd\_piter**

Backward [Site\\_Iterator](#) associated type.

Definition at line 141 of file `p_complex.hh`.

**10.272.2.2 template<unsigned D, typename G> typedef super\_::site mln::p\_complex< D, G  
>::element**

Associated types.

Element associated type.

Definition at line 132 of file `p_complex.hh`.

**10.272.2.3 template<unsigned D, typename G> typedef p\_complex\_fwd\_piter\_<D, G>  
mln::p\_complex< D, G >::fwd\_piter**

Forward [Site\\_Iterator](#) associated type.

Definition at line 138 of file `p_complex.hh`.

**10.272.2.4 template<unsigned D, typename G> typedef fwd\_piter mln::p\_complex< D, G  
>::piter**

[Site\\_Iterator](#) associated type.

Definition at line 144 of file `p_complex.hh`.

**10.272.2.5 template<unsigned D, typename G> typedef complex\_psite<D, G> mln::p\_complex<  
D, G >::psite**

[Point\\_Site](#) associated type.

Definition at line 135 of file p\_complex.hh.

### 10.272.3 Constructor & Destructor Documentation

**10.272.3.1** `template<unsigned D, typename G > mln::p_complex< D, G >::p_complex ( const topo::complex< D > & cplx, const G & geom ) [inline]`

Construct a complex psite set from a complex.

#### Parameters

*cplx* The complex upon which the complex psite set is built.  
*geom* FIXME

Definition at line 231 of file p\_complex.hh.

### 10.272.4 Member Function Documentation

**10.272.4.1** `template<unsigned D, typename G > topo::complex< D > & mln::p_complex< D, G >::cplx ( ) const`

Accessors.

Return the complex associated to the [p\\_complex](#) domain (const version)

Definition at line 293 of file p\_complex.hh.

References `mln::p_complex< D, G >::is_valid()`.

Referenced by `mln::complex_psite< D, G >::change_target()`, `mln::complex_psite< D, G >::complex_psite()`, and `mln::operator==( )`.

**10.272.4.2** `template<unsigned D, typename G > topo::complex< D > & mln::p_complex< D, G >::cplx ( )`

Return the complex associated to the [p\\_complex](#) domain (mutable version).

Definition at line 301 of file p\_complex.hh.

References `mln::p_complex< D, G >::is_valid()`.

**10.272.4.3** `template<unsigned D, typename G > const G & mln::p_complex< D, G >::geom ( ) const`

Return the geometry of the complex.

Definition at line 309 of file p\_complex.hh.

**10.272.4.4** `template<unsigned D, typename G > bool mln::p_complex< D, G >::has ( const psite & p ) const [inline]`

Does this site set has *p*?

Definition at line 271 of file p\_complex.hh.



References `mln::complex_psite< D, G >::is_valid()`, `mln::p_complex< D, G >::is_valid()`, and `mln::complex_psite< D, G >::site_set()`.

**10.272.4.5** `template<unsigned D, typename G > bool mln::p_complex< D, G >::is_valid ( )`  
`const [inline]`

Is this site set valid?

Definition at line 263 of file `p_complex.hh`.

Referenced by `mln::p_complex< D, G >::cplx()`, and `mln::p_complex< D, G >::has()`.

**10.272.4.6** `template<unsigned D, typename G > unsigned mln::p_complex< D, G >::nfaces ( )`  
`const [inline]`

Return the number of faces in the complex.

Definition at line 247 of file `p_complex.hh`.

Referenced by `mln::p_complex< D, G >::nsites()`.

**10.272.4.7** `template<unsigned D, typename G > unsigned mln::p_complex< D, G >::nfaces_of_dim ( unsigned n )`  
`const [inline]`

Return the number of *n-faces* in the complex.

Definition at line 255 of file `p_complex.hh`.

**10.272.4.8** `template<unsigned D, typename G > unsigned mln::p_complex< D, G >::nsites ( )`  
`const [inline]`

Return The number of sites of the set, i.e., the number of *faces*.

(Required by the `mln::Site_Set` concept, since the property `trait::site_set::nsites::known` of this site set is set to 'known'.)

Definition at line 239 of file `p_complex.hh`.

References `mln::p_complex< D, G >::nfaces()`.

## 10.273 `mln::p_edges< G, F >` Class Template Reference

[Site](#) set mapping graph edges and image sites.

```
#include <p_edges.hh>
```

Inherits `site_set_base< F::result, p_edges< G, F > >`.

### Public Types

- typedef `util::edge< G > edge`

*Type of graph edge.*

- typedef F `fun_t`  
*Function associated type.*
- typedef `util::edge< G >` `graph_element`  
*Type of graph element this site set focuses on.*
- typedef G `graph_t`  
*Graph associated type.*
- typedef `super_::site` `element`  
*Associated types.*
- typedef `p_edges_psite< G, F >` `psite`  
*Point\_Site associated type.*
- typedef `p_graph_piter< self_, mln_edge_fwd_iter(G) >` `fwd_piter`  
*Forward Site\_Iterator associated type.*
- typedef `p_graph_piter< self_, mln_edge_bkd_iter(G) >` `bkd_piter`  
*Backward Site\_Iterator associated type.*
- typedef `fwd_piter` `piter`  
*Site\_Iterator associated type.*

## Public Member Functions

- bool `has` (const `psite` &p) const  
*Does this site set has site p?*
- template<typename G2 >  
bool `has` (const `util::edge< G2 >` &e) const  
*Does this site set has edge e?*
- void `invalidate` ()  
*Invalidate this site set.*
- bool `is_valid` () const  
*Is this site set valid?*
- `std::size_t` `memory_size` () const  
*Does this site set has vertex\_id? FIXME: causes ambiguities while calling has(mln::neighb\_fwd\_niter<>); bool has(unsigned vertex\_id) const;.*
- unsigned `nedges` () const  
*Return The number of edges in the graph.*
- unsigned `nsites` () const  
*Return The number of points (sites) of the set, i.e., the number of edges.*

- [p\\_edges](#) ()  
*Constructors*  
*Default constructor.*
- [p\\_edges](#) (const [Graph](#)< G > &gr)  
*Construct a graph edge psite set from a graph.*
- [p\\_edges](#) (const [Graph](#)< G > &gr, const [Function](#)< F > &f)  
*Construct a graph edge psite set from a graph and a function.*
- `template<typename F2 >`  
[p\\_edges](#) (const [Graph](#)< G > &gr, const [Function](#)< F2 > &f)  
*Construct a graph edge psite set from a graph and a function.*
- const G & [graph](#) () const  
*Accessors.*
- const F & [function](#) () const  
*Return the mapping function.*

### 10.273.1 Detailed Description

`template<typename G, typename F = util::internal::id2element<G,util::edge<G> >> class mln::p_edges< G, F >`

[Site](#) set mapping graph edges and image sites.

Definition at line 70 of file p\_edges.hh.

### 10.273.2 Member Typedef Documentation

**10.273.2.1** `template<typename G, typename F = util::internal::id2element<G,util::edge<G> >> typedef p_graph_piter< self_, mln_edge_bkd_iter(G) > mln::p_edges< G, F >::bkd_piter`

Backward [Site\\_Iterator](#) associated type.

Definition at line 128 of file p\_edges.hh.

**10.273.2.2** `template<typename G, typename F = util::internal::id2element<G,util::edge<G> >> typedef util::edge<G> mln::p_edges< G, F >::edge`

Type of graph edge.

Definition at line 86 of file p\_edges.hh.

**10.273.2.3** `template<typename G, typename F = util::internal::id2element<G,util::edge<G> >> typedef super_::site mln::p_edges< G, F >::element`

Associated types.

Element associated type.

Definition at line 119 of file p\_edges.hh.

```
10.273.2.4 template<typename G, typename F = util::internal::id2element<G,util::edge<G> >>
    typedef F mln::p_edges< G, F >::fun_t
```

[Function](#) associated type.

Definition at line 83 of file p\_edges.hh.

```
10.273.2.5 template<typename G, typename F = util::internal::id2element<G,util::edge<G> >>
    typedef p_graph_piter< self_, mln_edge_fwd_iter(G) > mln::p_edges< G, F >::fwd_piter
```

Forward [Site\\_Iterator](#) associated type.

Definition at line 125 of file p\_edges.hh.

```
10.273.2.6 template<typename G, typename F = util::internal::id2element<G,util::edge<G> >>
    typedef util::edge<G> mln::p_edges< G, F >::graph_element
```

Type of graph element this site set focuses on.

Definition at line 89 of file p\_edges.hh.

```
10.273.2.7 template<typename G, typename F = util::internal::id2element<G,util::edge<G> >>
    typedef G mln::p_edges< G, F >::graph_t
```

[Graph](#) associated type.

Definition at line 80 of file p\_edges.hh.

```
10.273.2.8 template<typename G, typename F = util::internal::id2element<G,util::edge<G> >>
    typedef fwd_piter mln::p_edges< G, F >::piter
```

[Site\\_Iterator](#) associated type.

Definition at line 131 of file p\_edges.hh.

```
10.273.2.9 template<typename G, typename F = util::internal::id2element<G,util::edge<G> >>
    typedef p_edges_psite<G, F> mln::p_edges< G, F >::psite
```

[Point\\_Site](#) associated type.

Definition at line 122 of file p\_edges.hh.

### 10.273.3 Constructor & Destructor Documentation

**10.273.3.1** `template<typename G , typename F > mln::p_edges< G, F >::p_edges ( )`  
**[inline]**

Constructors

Default constructor.

Definition at line 203 of file p\_edges.hh.

**10.273.3.2** `template<typename G , typename F > mln::p_edges< G, F >::p_edges ( const`  
**Graph< G > & gr )** **[inline]**

Construct a graph edge psite set from a graph.

#### Parameters

*gr* The graph upon which the graph edge psite set is built.

Definition at line 209 of file p\_edges.hh.

References mln::p\_edges< G, F >::is\_valid().

**10.273.3.3** `template<typename G , typename F > mln::p_edges< G, F >::p_edges ( const`  
**Graph< G > & gr, const Function< F > & f )** **[inline]**

Construct a graph edge psite set from a graph and a function.

#### Parameters

*gr* The graph upon which the graph edge psite set is built.

*f* the function mapping edges and sites.

Definition at line 221 of file p\_edges.hh.

References mln::p\_edges< G, F >::is\_valid().

**10.273.3.4** `template<typename G , typename F > template<typename F2 > mln::p_edges< G, F`  
**>::p\_edges ( const Graph< G > & gr, const Function< F2 > & f )** **[inline]**

Construct a graph edge psite set from a graph and a function.

#### Parameters

*gr* The graph upon which the graph edge psite set is built.

*f* the function mapping edges and sites. It must be convertible towards the function type F.

Definition at line 231 of file p\_edges.hh.

References mln::p\_edges< G, F >::is\_valid().

### 10.273.4 Member Function Documentation

**10.273.4.1** `template<typename G , typename F > const F & mln::p_edges< G, F >::function ( ) const [inline]`

Return the mapping function.

Definition at line 324 of file p\_edges.hh.

**10.273.4.2** `template<typename G , typename F > const G & mln::p_edges< G, F >::graph ( ) const [inline]`

Accessors.

Return the graph associated to this site set

Definition at line 315 of file p\_edges.hh.

References `mln::p_edges< G, F >::is_valid()`.

Referenced by `mln::operator==()`.

**10.273.4.3** `template<typename G , typename F > bool mln::p_edges< G, F >::has ( const psite & p ) const [inline]`

Does this site set has site *p*?

Definition at line 276 of file p\_edges.hh.

References `mln::p_edges< G, F >::is_valid()`.

**10.273.4.4** `template<typename G , typename F > template<typename G2 > bool mln::p_edges< G, F >::has ( const util::edge< G2 > & e ) const [inline]`

Does this site set has edge *e*?

Definition at line 286 of file p\_edges.hh.

References `mln::util::edge< G >::graph()`, `mln::util::edge< G >::is_valid()`, and `mln::p_edges< G, F >::is_valid()`.

**10.273.4.5** `template<typename G , typename F > void mln::p_edges< G, F >::invalidate ( ) [inline]`

Invalidate this site set.

Definition at line 268 of file p\_edges.hh.

**10.273.4.6** `template<typename G , typename F > bool mln::p_edges< G, F >::is_valid ( ) const [inline]`

Is this site set valid?

Definition at line 260 of file p\_edges.hh.

Referenced by `mln::p_edges< G, F >::graph()`, `mln::p_edges< G, F >::has()`, and `mln::p_edges< G, F >::p_edges()`.

**10.273.4.7** `template<typename G, typename F> std::size_t mln::p_edges< G, F >::memory_size ( ) const [inline]`

Does this site set has *vertex\_id*? FIXME: causes ambiguities while calling `has(mln::neighb_fwd_niter<>)`; `bool has(unsigned vertex_id) const;`

Definition at line 305 of file `p_edges.hh`.

**10.273.4.8** `template<typename G, typename F> unsigned mln::p_edges< G, F >::nedges ( ) const [inline]`

Return The number of edges in the graph.

Definition at line 252 of file `p_edges.hh`.

Referenced by `mln::p_edges< G, F >::nsites()`.

**10.273.4.9** `template<typename G, typename F> unsigned mln::p_edges< G, F >::nsites ( ) const [inline]`

Return The number of points (sites) of the set, i.e., the number of *edges*.

Definition at line 244 of file `p_edges.hh`.

References `mln::p_edges< G, F >::nedges()`.

## 10.274 `mln::p_faces< N, D, P >` Struct Template Reference

A complex psite set based on a the N-faces of a complex of dimension D (a D-complex).

```
#include <p_faces.hh>
```

Inherits `site_set_base_< faces_psite< N, D, P >, p_faces< N, D, P > >`.

### Public Types

- typedef `super_::site element`  
*Associated types.*
- typedef `faces_psite< N, D, P > psite`  
*Point\_Site associated type.*
- typedef `p_faces_fwd_piter_< N, D, P > fwd_piter`  
*Forward Site\_Iterator associated type.*
- typedef `p_faces_bkd_piter_< N, D, P > bkd_piter`  
*Backward Site\_Iterator associated type.*
- typedef `fwd_piter piter`  
*Site\_Iterator associated type.*

## Public Member Functions

- `bool is_valid () const`  
*Is this site set valid?*
- `unsigned nfaces () const`  
*Return The number of faces in the complex.*
- `unsigned nsites () const`  
*Return The number of sites of the set, i.e., the number of faces.*
- `p_faces (const topo::complex< D > &cplx)`  
*Construct a faces psite set from an mln::complex.*
- `p_faces (const p_complex< D, P > &pc)`  
*Construct a faces psite set from an mln::p\_complex.*
- `topo::complex< D > & cplx () const`  
*Accessors.*
- `topo::complex< D > & cplx ()`  
*Return the complex associated to the p\_faces domain (mutable version).*

### 10.274.1 Detailed Description

`template<unsigned N, unsigned D, typename P> struct mln::p_faces< N, D, P >`

A complex psite set based on a the N-faces of a complex of dimension D (a D-complex).

Definition at line 77 of file p\_faces.hh.

### 10.274.2 Member Typedef Documentation

**10.274.2.1** `template<unsigned N, unsigned D, typename P> typedef p_faces_bkd_piter_<N, D, P> mln::p_faces< N, D, P >::bkd_piter`

Backward [Site\\_Iterator](#) associated type.

Definition at line 110 of file p\_faces.hh.

**10.274.2.2** `template<unsigned N, unsigned D, typename P> typedef super_::site mln::p_faces< N, D, P >::element`

Associated types.

Element associated type.

Definition at line 99 of file p\_faces.hh.



**10.274.2.3** `template<unsigned N, unsigned D, typename P> typedef p_faces_fwd_piter_<N, D, P> mln::p_faces< N, D, P >::fwd_piter`

Forward [Site\\_Iterator](#) associated type.

Definition at line 106 of file p\_faces.hh.

**10.274.2.4** `template<unsigned N, unsigned D, typename P> typedef fwd_piter mln::p_faces< N, D, P >::piter`

[Site\\_Iterator](#) associated type.

Definition at line 113 of file p\_faces.hh.

**10.274.2.5** `template<unsigned N, unsigned D, typename P> typedef faces_psite<N, D, P> mln::p_faces< N, D, P >::psite`

[Point\\_Site](#) associated type.

Definition at line 102 of file p\_faces.hh.

### 10.274.3 Constructor & Destructor Documentation

**10.274.3.1** `template<unsigned N, unsigned D, typename P > mln::p_faces< N, D, P >::p_faces ( const topo::complex< D > & cplx ) [inline]`

Construct a faces psite set from an mln::complex.

#### Parameters

*cplx* The complex upon which the complex psite set is built.

Definition at line 192 of file p\_faces.hh.

**10.274.3.2** `template<unsigned N, unsigned D, typename P > mln::p_faces< N, D, P >::p_faces ( const p_complex< D, P > & pc ) [inline]`

Construct a faces psite set from an mln::p\_complex.

#### Parameters

*pc* The complex upon which the complex psite set is built.

Definition at line 201 of file p\_faces.hh.

### 10.274.4 Member Function Documentation

**10.274.4.1** `template<unsigned N, unsigned D, typename P > topo::complex< D > & mln::p_faces< N, D, P >::cplx ( ) const`

Accessors.

Return the complex associated to the [p\\_faces](#) domain (const version).

Definition at line 257 of file `p_faces.hh`.

References `mln::p_faces< N, D, P >::is_valid()`.

Referenced by `mln::faces_psite< N, D, P >::change_target()`, and `mln::operator==( )`.

#### 10.274.4.2 `template<unsigned N, unsigned D, typename P > topo::complex< D > & mln::p_faces< N, D, P >::cplx ( )`

Return the complex associated to the [p\\_faces](#) domain (mutable version).

Definition at line 265 of file `p_faces.hh`.

References `mln::p_faces< N, D, P >::is_valid()`.

#### 10.274.4.3 `template<unsigned N, unsigned D, typename P > bool mln::p_faces< N, D, P >::is_valid ( ) const [inline]`

Is this site set valid?

Definition at line 227 of file `p_faces.hh`.

Referenced by `mln::p_faces< N, D, P >::cplx()`.

#### 10.274.4.4 `template<unsigned N, unsigned D, typename P > unsigned mln::p_faces< N, D, P >::nfaces ( ) const [inline]`

Return The number of faces in the complex.

Definition at line 219 of file `p_faces.hh`.

Referenced by `mln::p_faces< N, D, P >::nsites()`.

#### 10.274.4.5 `template<unsigned N, unsigned D, typename P > unsigned mln::p_faces< N, D, P >::nsites ( ) const [inline]`

Return The number of sites of the set, i.e., the number of *faces*.

(Required by the [mln::Site\\_Set](#) concept, since the property trait::site\_set::nsites::known of this site set is set to 'known'.)

Definition at line 211 of file `p_faces.hh`.

References `mln::p_faces< N, D, P >::nfaces()`.

## 10.275 `mln::p_graph_piter< S, I > Class Template Reference`

Generic iterator on point sites of a `mln::S`.

```
#include <p_graph_piter.hh>
```

Inherits `site_set_iterator_base< S, p_graph_piter< S, I > >`.

## Public Member Functions

- `const S::graph_t & graph () const`  
*Return the graph associated to the target S.*
  - `unsigned id () const`  
*Return the graph element id.*
  - `mln_q_subject (iter) element()`  
*Return the underlying graph element.*
  - `void next ()`  
*Go to the next element.*
- 
- `p_graph_piter ()`  
*Constructors.*

### 10.275.1 Detailed Description

`template<typename S, typename I> class mln::p_graph_piter< S, I >`

Generic iterator on point sites of a mln::S.

Definition at line 55 of file p\_graph\_piter.hh.

### 10.275.2 Constructor & Destructor Documentation

**10.275.2.1** `template<typename S, typename I> mln::p_graph_piter< S, I >::p_graph_piter ( )`  
`[inline]`

Constructors.

Definition at line 151 of file p\_graph\_piter.hh.

### 10.275.3 Member Function Documentation

**10.275.3.1** `template<typename S, typename I> const S::graph_t & mln::p_graph_piter< S, I >::graph ( ) const` `[inline]`

Return the graph associated to the target S.

Definition at line 212 of file p\_graph\_piter.hh.

**10.275.3.2** `template<typename S, typename I> unsigned mln::p_graph_piter< S, I >::id ( )`  
`const [inline]`

Return the graph element id.

Definition at line 228 of file p\_graph\_piter.hh.

### 10.275.3.3 `template<typename S , typename I > mln::p_graph_piter< S, I >::mln_q_subject ( iter )`

Return the underlying graph element.

### 10.275.3.4 `void mln::Site_Iterator< p_graph_piter< S, I > >::next ( ) [inherited]`

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

#### Precondition

The iterator is valid.

## 10.276 `mln::p_if< S, F > Class Template Reference`

[Site](#) set restricted w.r.t.

```
#include <p_if.hh>
```

Inherits `site_set_base_< S::psite, p_if< S, F > >`.

### Public Types

- `typedef p_if_piter_< typename S::bkd_piter, S, F > bkd_piter`  
*Backward [Site\\_Iterator](#) associated type.*
- `typedef S::element element`  
*Element associated type.*
- `typedef p_if_piter_< typename S::fwd_piter, S, F > fwd_piter`  
*Forward [Site\\_Iterator](#) associated type.*
- `typedef fwd_piter piter`  
*[Site\\_Iterator](#) associated type.*
- `typedef S::psite psite`  
*Psite associated type.*

### Public Member Functions

- `bool has (const psite &p) const`  
*Test if *p* belongs to the subset.*
- `bool is_valid () const`

*Test if this site set is valid.*

- `std::size_t memory_size () const`  
*Return the size of this site set in memory.*
- `const S & overset () const`  
*Give the primary overset.*
- `p_if ()`  
*Constructor without argument.*
- `p_if (const S &s, const F &f)`  
*Constructor with a site set *s* and a predicate *f*.*
- `bool pred (const psite &p) const`  
*Test predicate on point site *p*.*
- `const F & predicate () const`  
*Give the predicate function.*

### 10.276.1 Detailed Description

`template<typename S, typename F> class mln::p_if< S, F >`

[Site](#) set restricted w.r.t. a predicate.

Parameter *S* is a site set type; parameter *F* is a function from point to Boolean.

Definition at line 83 of file `p_if.hh`.

### 10.276.2 Member Typedef Documentation

**10.276.2.1** `template<typename S, typename F> typedef p_if_piter_<typename S ::bkd_piter, S, F> mln::p_if< S, F >::bkd_piter`

Backward [Site\\_Iterator](#) associated type.

Definition at line 100 of file `p_if.hh`.

**10.276.2.2** `template<typename S, typename F> typedef S ::element mln::p_if< S, F >::element`

Element associated type.

Definition at line 90 of file `p_if.hh`.

**10.276.2.3** `template<typename S, typename F> typedef p_if_piter_<typename S ::fwd_piter, S, F> mln::p_if< S, F >::fwd_piter`

Forward [Site\\_Iterator](#) associated type.

Definition at line 97 of file `p_if.hh`.

**10.276.2.4** `template<typename S, typename F> typedef fwd_piter mln::p_if< S, F >::piter`

[Site\\_Iterator](#) associated type.

Definition at line 103 of file p\_if.hh.

**10.276.2.5** `template<typename S, typename F> typedef S ::psite mln::p_if< S, F >::psite`

Psite associated type.

Definition at line 94 of file p\_if.hh.

**10.276.3 Constructor & Destructor Documentation****10.276.3.1** `template<typename S, typename F > mln::p_if< S, F >::p_if ( const S & s, const F & f ) [inline]`

Constructor with a site set *s* and a predicate *f*.

Definition at line 190 of file p\_if.hh.

**10.276.3.2** `template<typename S, typename F > mln::p_if< S, F >::p_if ( ) [inline]`

Constructor without argument.

Definition at line 198 of file p\_if.hh.

**10.276.4 Member Function Documentation****10.276.4.1** `template<typename S, typename F > bool mln::p_if< S, F >::has ( const psite & p ) const [inline]`

Test if *p* belongs to the subset.

Definition at line 159 of file p\_if.hh.

References `mln::p_if< S, F >::has()`.

Referenced by `mln::p_if< S, F >::has()`.

**10.276.4.2** `template<typename S, typename F > bool mln::p_if< S, F >::is_valid ( ) const [inline]`

Test if this site set is valid.

Definition at line 167 of file p\_if.hh.

**10.276.4.3** `template<typename S, typename F > std::size_t mln::p_if< S, F >::memory_size ( ) const [inline]`

Return the size of this site set in memory.

Definition at line 213 of file p\_if.hh.

**10.276.4.4** `template<typename S, typename F > const S & mln::p_if< S, F >::overset ( ) const [inline]`

Give the primary overset.

Definition at line 175 of file p\_if.hh.

**10.276.4.5** `template<typename S, typename F > bool mln::p_if< S, F >::pred ( const psite & p ) const [inline]`

Test predicate on point site p.

Definition at line 183 of file p\_if.hh.

**10.276.4.6** `template<typename S, typename F > const F & mln::p_if< S, F >::predicate ( ) const [inline]`

Give the predicate function.

Definition at line 205 of file p\_if.hh.

## 10.277 mln::p\_image< I > Class Template Reference

[Site](#) set based on an image of Booleans.

```
#include <p_image.hh>
```

Inherits `site_set_base_< I::psite, p_image< I > >`.

### Public Types

- `typedef S::bkd_piter bkd_piter`  
*Backward [Site\\_Iterator](#) associated type.*
- `typedef I::psite element`  
*Element associated type.*
- `typedef S::fwd_piter fwd_piter`  
*Forward [Site\\_Iterator](#) associated type.*
- `typedef psite i_element`  
*Insertion element associated type.*
- `typedef S::piter piter`  
*[Site\\_Iterator](#) associated type.*
- `typedef I::psite psite`  
*Psite associated type.*
- `typedef psite r_element`  
*Removal element associated type.*

- typedef `internal::p_image_site_set< I >::ret S`  
*Equivalent site\_set type.*

## Public Member Functions

- void `clear ()`  
*Clear this set.*
- bool `has (const psite &) const`  
*Test if the psite p belongs to this site set.*
- void `insert (const psite &p)`  
*Insert a site p.*
- bool `is_valid () const`  
*Test if this site set is valid, i.e., initialized.*
- `std::size_t memory_size () const`  
*Return the size of this site set in memory.*
- unsigned `nsites () const`  
*Give the number of sites.*
- operator `typename internal::p_image_site_set< I >::ret () const`  
*Conversion towards the equivalent site set.*
- `p_image ()`  
*Constructor without argument.*
- `p_image (const I &ima)`  
*Constructor.*
- void `remove (const psite &p)`  
*Remove a site p.*
- void `toggle (const psite &p)`  
*Change the status in/out of a site p.*

### 10.277.1 Detailed Description

`template<typename I> class mln::p_image< I >`

[Site](#) set based on an image of Booleans.

Definition at line 88 of file `p_image.hh`.



## 10.277.2 Member Typedef Documentation

### 10.277.2.1 `template<typename I > typedef S ::bkd_piter mln::p_image< I >::bkd_piter`

Backward [Site\\_Iterator](#) associated type.

Definition at line 110 of file p\_image.hh.

### 10.277.2.2 `template<typename I > typedef I ::psite mln::p_image< I >::element`

Element associated type.

Definition at line 100 of file p\_image.hh.

### 10.277.2.3 `template<typename I > typedef S ::fwd_piter mln::p_image< I >::fwd_piter`

Forward [Site\\_Iterator](#) associated type.

Definition at line 107 of file p\_image.hh.

### 10.277.2.4 `template<typename I > typedef psite mln::p_image< I >::i_element`

Insertion element associated type.

Definition at line 136 of file p\_image.hh.

### 10.277.2.5 `template<typename I > typedef S ::piter mln::p_image< I >::piter`

[Site\\_Iterator](#) associated type.

Definition at line 113 of file p\_image.hh.

### 10.277.2.6 `template<typename I > typedef I ::psite mln::p_image< I >::psite`

Psite associated type.

Definition at line 104 of file p\_image.hh.

### 10.277.2.7 `template<typename I > typedef psite mln::p_image< I >::r_element`

Removal element associated type.

Definition at line 142 of file p\_image.hh.

### 10.277.2.8 `template<typename I > typedef internal::p_image_site_set<I>::ret mln::p_image< I >::S`

Equivalent site\_set type.

Definition at line 93 of file p\_image.hh.

### 10.277.3 Constructor & Destructor Documentation

#### 10.277.3.1 `template<typename I> mln::p_image<I>::p_image( ) [inline]`

Constructor without argument.

Definition at line 182 of file `p_image.hh`.

#### 10.277.3.2 `template<typename I> mln::p_image<I>::p_image( const I & ima ) [inline]`

Constructor.

Definition at line 189 of file `p_image.hh`.

References `mln::p_image<I>::clear()`.

### 10.277.4 Member Function Documentation

#### 10.277.4.1 `template<typename I> void mln::p_image<I>::clear( ) [inline]`

Clear this set.

Definition at line 283 of file `p_image.hh`.

References `mln::data::fill_with_value()`, and `mln::p_image<I>::is_valid()`.

Referenced by `mln::p_image<I>::p_image()`.

#### 10.277.4.2 `template<typename I> bool mln::p_image<I>::has( const psite & p ) const [inline]`

Test if the `psite p` belongs to this site set.

Definition at line 199 of file `p_image.hh`.

References `mln::p_image<I>::is_valid()`.

#### 10.277.4.3 `template<typename I> void mln::p_image<I>::insert( const psite & p ) [inline]`

Insert a site `p`.

Definition at line 224 of file `p_image.hh`.

References `mln::p_image<I>::is_valid()`.

#### 10.277.4.4 `template<typename I> bool mln::p_image<I>::is_valid( ) const [inline]`

Test if this site set is valid, i.e., initialized.

Definition at line 208 of file `p_image.hh`.

Referenced by `mln::p_image<I>::clear()`, `mln::p_image<I>::has()`, `mln::p_image<I>::insert()`, `mln::p_image<I>::memory_size()`, `mln::p_image<I>::remove()`, and `mln::p_image<I>::toggle()`.

**10.277.4.5** `template<typename I> std::size_t mln::p_image< I >::memory_size ( ) const [inline]`

Return the size of this site set in memory.

Definition at line 273 of file p\_image.hh.

References mln::p\_image< I >::is\_valid().

**10.277.4.6** `template<typename I> unsigned mln::p_image< I >::nsites ( ) const [inline]`

Give the number of sites.

Definition at line 216 of file p\_image.hh.

**10.277.4.7** `template<typename I> mln::p_image< I >::operator typename internal::p_image_site_set< I >::ret ( ) const [inline]`

Conversion towards the equivalent site set.

Definition at line 174 of file p\_image.hh.

**10.277.4.8** `template<typename I> void mln::p_image< I >::remove ( const psite & p ) [inline]`

Remove a site p.

Definition at line 237 of file p\_image.hh.

References mln::p\_image< I >::is\_valid().

**10.277.4.9** `template<typename I> void mln::p_image< I >::toggle ( const psite & p ) [inline]`

Change the status in/out of a site p.

Definition at line 251 of file p\_image.hh.

References mln::p\_image< I >::is\_valid().

## 10.278 mln::p\_indexed\_bkd\_piter< S > Class Template Reference

Backward iterator on sites of an indexed site set.

```
#include <p_array.hh>
```

Inherits site\_set\_iterator\_base< S, p\_indexed\_bkd\_piter< S > >.

### Public Member Functions

- `int index ( ) const`

*Return the current index.*

- void `next()`  
*Go to the next element.*
- `p_indexed_bkd_piter` (const S &s)  
*Constructor.*
- `p_indexed_bkd_piter` ()  
*Constructor with no argument.*

### 10.278.1 Detailed Description

`template<typename S> class mln::p_indexed_bkd_piter< S >`

Backward iterator on sites of an indexed site set.

Definition at line 276 of file `p_array.hh`.

### 10.278.2 Constructor & Destructor Documentation

**10.278.2.1** `template<typename S > mln::p_indexed_bkd_piter< S >::p_indexed_bkd_piter ( )`  
[inline]

Constructor with no argument.

Definition at line 686 of file `p_array.hh`.

**10.278.2.2** `template<typename S > mln::p_indexed_bkd_piter< S >::p_indexed_bkd_piter ( const S & s )` [inline]

Constructor.

Definition at line 692 of file `p_array.hh`.

### 10.278.3 Member Function Documentation

**10.278.3.1** `template<typename S > int mln::p_indexed_bkd_piter< S >::index ( ) const`  
[inline]

Return the current index.

Definition at line 733 of file `p_array.hh`.

**10.278.3.2** `void mln::Site_Iterator< p_indexed_bkd_piter< S > >::next ( )` [inherited]

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

**Precondition**

The iterator is valid.

**10.279 mln::p\_indexed\_fwd\_piter< S > Class Template Reference**

Forward iterator on sites of an indexed site set.

```
#include <p_array.hh>
```

Inherits site\_set\_iterator\_base< S, p\_indexed\_fwd\_piter< S > >.

**Public Member Functions**

- int [index](#) () const  
*Return the current index.*
- void [next](#) ()  
*Go to the next element.*
- [p\\_indexed\\_fwd\\_piter](#) (const S &s)  
*Constructor.*
- [p\\_indexed\\_fwd\\_piter](#) ()  
*Constructor with no argument.*

**10.279.1 Detailed Description**

```
template<typename S> class mln::p_indexed_fwd_piter< S >
```

Forward iterator on sites of an indexed site set.

Definition at line 235 of file p\_array.hh.

**10.279.2 Constructor & Destructor Documentation**

**10.279.2.1** `template<typename S > mln::p_indexed_fwd_piter< S >::p_indexed_fwd_piter ( )`  
`[inline]`

Constructor with no argument.

Definition at line 629 of file p\_array.hh.

**10.279.2.2** `template<typename S > mln::p_indexed_fwd_piter< S >::p_indexed_fwd_piter (`  
`const S & s ) [inline]`

Constructor.

Definition at line 635 of file p\_array.hh.

### 10.279.3 Member Function Documentation

**10.279.3.1** `template<typename S> int mln::p_indexed_fwd_piter< S >::index ( ) const`  
`[inline]`

Return the current index.

Definition at line 676 of file p\_array.hh.

**10.279.3.2** `void mln::Site_Iterator< p_indexed_fwd_piter< S > >::next ( ) [inherited]`

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

#### Precondition

The iterator is valid.

## 10.280 mln::p\_indexed\_psite< S > Class Template Reference

Psite class for indexed site sets such as [p\\_array](#).

```
#include <p_array.hh>
```

Inherits `pseudo_site_base_< const S::element &, p_indexed_psite< S > >`.

### 10.280.1 Detailed Description

`template<typename S> class mln::p_indexed_psite< S >`

Psite class for indexed site sets such as [p\\_array](#).

Definition at line 183 of file p\_array.hh.

## 10.281 mln::p\_key< K, P > Class Template Reference

Priority queue class.

```
#include <p_key.hh>
```

Inherits `site_set_base_< P, p_key< K, P > >`.

### Public Types

- `typedef p_double_piter< self_, mln_bkd_eiter(util::set< K >), typename p_set< P >::bkd_piter >`  
`bkd_piter`

*Backward [Site\\_Iterator](#) associated type.*

- typedef P [element](#)  
*Element associated type.*
- typedef p\_double\_piter< [self\\_](#), mln\_fwd\_eiter(util::set< K >), typename p\_set< P >::fwd\_piter > [fwd\\_piter](#)  
*Forward [Site\\_Iterator](#) associated type.*
- typedef std::pair< K, P > [i\\_element](#)  
*Insertion element associated type.*
- typedef [fwd\\_piter](#) [piter](#)  
*[Site\\_Iterator](#) associated type.*
- typedef p\_double\_psite< [self\\_](#), p\_set< P > > [psite](#)  
*Psite associated type.*
- typedef P [r\\_element](#)  
*Removal element associated type.*

## Public Member Functions

- void [change\\_key](#) (const K &k, const K &new\_k)  
*Change the key *k* into a new value *new\_k*.*
- template<typename F >  
void [change\\_keys](#) (const [Function\\_v2v](#)< F > &f)  
*Change the keys by applying the function *f*.*
- void [clear](#) ()  
*Clear this site set.*
- bool [exists\\_key](#) (const K &key) const  
*Test if the *priority* exists.*
- bool [has](#) (const [psite](#) &) const  
*Test is the *psite p* belongs to this site set.*
- bool [has](#) (const P &p) const  
*Test is the *psite p* belongs to this site set.*
- void [insert](#) (const [i\\_element](#) &k\_p)  
*Insert a pair *k\_p* (key *k*, site *p*).*
- void [insert](#) (const K &k, const P &p)  
*Insert a pair (key *k*, site *p*).*
- bool [is\\_valid](#) () const  
*Test this set validity so returns always true.*

- `const K & key (const P &p) const`  
*Give the key associated with site `p`.*
- `const util::set< K > & keys () const`  
*Give the set of keys.*
- `std::size_t memory_size () const`  
*Return the size of this site set in memory.*
- `unsigned nsites () const`  
*Give the number of sites.*
- `const p_set< P > & operator() (const K &key) const`  
*Give the queue with the priority `priority`.*
- `p_key ()`  
*Constructor.*
- `void remove (const P &p)`  
*Remove a site `p`.*
- `void remove_key (const K &k)`  
*Remove all sites with key `k`.*

### 10.281.1 Detailed Description

`template<typename K, typename P> class mln::p_key< K, P >`

Priority queue class.

Definition at line 72 of file `p_key.hh`.

### 10.281.2 Member Typedef Documentation

**10.281.2.1** `template<typename K , typename P > typedef p_double_piter<self_, mln_bkd_eiter(util::set<K>), typename p_set<P>::bkd_piter> mln::p_key< K, P >::bkd_piter`

Backward [Site\\_Iterator](#) associated type.

Definition at line 93 of file `p_key.hh`.

**10.281.2.2** `template<typename K , typename P > typedef P mln::p_key< K, P >::element`

Element associated type.

Definition at line 79 of file `p_key.hh`.



**10.281.2.3** `template<typename K , typename P > typedef p_double_piter<self_, mln_fwd_eiter(util::set<K>), typename p_set<P>::fwd_piter> mln::p_key< K, P >::fwd_piter`

Forward [Site\\_Iterator](#) associated type.

Definition at line 88 of file p\_key.hh.

**10.281.2.4** `template<typename K , typename P > typedef std::pair<K,P> mln::p_key< K, P >::i_element`

Insertion element associated type.

Definition at line 118 of file p\_key.hh.

**10.281.2.5** `template<typename K , typename P > typedef fwd_piter mln::p_key< K, P >::piter`

[Site\\_Iterator](#) associated type.

Definition at line 96 of file p\_key.hh.

**10.281.2.6** `template<typename K , typename P > typedef p_double_psite< self_, p_set<P> > mln::p_key< K, P >::psite`

Psite associated type.

Definition at line 83 of file p\_key.hh.

**10.281.2.7** `template<typename K , typename P > typedef P mln::p_key< K, P >::r_element`

Removal element associated type.

Definition at line 132 of file p\_key.hh.

### 10.281.3 Constructor & Destructor Documentation

**10.281.3.1** `template<typename K , typename P > mln::p_key< K, P >::p_key ( ) [inline]`

Constructor.

Definition at line 208 of file p\_key.hh.

### 10.281.4 Member Function Documentation

**10.281.4.1** `template<typename K , typename P > void mln::p_key< K, P >::change_key ( const K & k, const K & new_k ) [inline]`

Change the key `k` into a new value `new_k`.

Definition at line 382 of file p\_key.hh.

References `mln::p_set< P >::nsites()`.

**10.281.4.2** `template<typename K , typename P > template<typename F > void mln::p_key< K, P >::change_keys ( const Function_v2v< F > & f ) [inline]`

Change the keys by applying the function `f`.

Definition at line 428 of file `p_key.hh`.

References `mln::util::set< T >::insert()`.

**10.281.4.3** `template<typename K , typename P > void mln::p_key< K, P >::clear ( ) [inline]`

Clear this site set.

Definition at line 462 of file `p_key.hh`.

**10.281.4.4** `template<typename K , typename P > bool mln::p_key< K, P >::exists_key ( const K & key ) const [inline]`

Test if the `priority` exists.

Definition at line 520 of file `p_key.hh`.

Referenced by `mln::p_key< K, P >::operator()()`.

**10.281.4.5** `template<typename K , typename P > bool mln::p_key< K, P >::has ( const P & p ) const [inline]`

Test is the psite `p` belongs to this site set.

Definition at line 227 of file `p_key.hh`.

**10.281.4.6** `template<typename K , typename P > bool mln::p_key< K, P >::has ( const psite & ) const [inline]`

Test is the psite `p` belongs to this site set.

Definition at line 217 of file `p_key.hh`.

Referenced by `mln::p_key< K, P >::insert()`.

**10.281.4.7** `template<typename K , typename P > void mln::p_key< K, P >::insert ( const i_element & k_p ) [inline]`

Insert a pair `k_p` (key `k`, site `p`).

Definition at line 301 of file `p_key.hh`.

**10.281.4.8** `template<typename K , typename P > void mln::p_key< K, P >::insert ( const K & k, const P & p ) [inline]`

Insert a pair (key `k`, site `p`).

Definition at line 268 of file `p_key.hh`.

References mln::p\_key< K, P >::has().

**10.281.4.9** `template<typename K , typename P > bool mln::p_key< K, P >::is_valid ( ) const [inline]`

Test this set validity so returns always true.

Definition at line 236 of file p\_key.hh.

**10.281.4.10** `template<typename K , typename P > const K & mln::p_key< K, P >::key ( const P & p ) const [inline]`

Give the key associated with site p.

Definition at line 501 of file p\_key.hh.

**10.281.4.11** `template<typename K , typename P > const util::set< K > & mln::p_key< K, P >::keys ( ) const [inline]`

Give the set of keys.

Definition at line 511 of file p\_key.hh.

**10.281.4.12** `template<typename K , typename P > std::size_t mln::p_key< K, P >::memory_size ( ) const [inline]`

Return the size of this site set in memory.

Definition at line 475 of file p\_key.hh.

**10.281.4.13** `template<typename K , typename P > unsigned mln::p_key< K, P >::nsites ( ) const [inline]`

Give the number of sites.

Definition at line 245 of file p\_key.hh.

**10.281.4.14** `template<typename K , typename P > const p_set< P > & mln::p_key< K, P >::operator() ( const K & key ) const [inline]`

Give the queue with the priority *priority*.

This method always works: if the priority is not in this set, an empty queue is returned.

Definition at line 489 of file p\_key.hh.

References mln::p\_key< K, P >::exists\_key().

**10.281.4.15** `template<typename K , typename P > void mln::p_key< K, P >::remove ( const P & p ) [inline]`

Remove a site p.

Definition at line 309 of file p\_key.hh.

**10.281.4.16** `template<typename K , typename P > void mln::p_key< K, P >::remove_key ( const K & k ) [inline]`

Remove all sites with key *k*.

Definition at line 351 of file `p_key.hh`.

References `mln::p_set< P >::nsites()`.

## 10.282 mln::p\_line2d Class Reference

2D discrete line of points.

```
#include <p_line2d.hh>
```

Inherits `site_set_base_< point2d, p_line2d >`.

### Public Types

- typedef `p_indexed_bkd_piter< self_ > bkd_piter`  
*Backward Site\_Iterator associated type.*
- typedef `point2d element`  
*Element associated type.*
- typedef `p_indexed_fwd_piter< self_ > fwd_piter`  
*Forward Site\_Iterator associated type.*
- typedef `p_indexed_fwd_piter< self_ > piter`  
*Site\_Iterator associated type.*
- typedef `p_indexed_psite< self_ > psite`  
*Psite associated type.*
- typedef const `box2d & q_box`  
*Box (qualified) associated type.*

### Public Member Functions

- const `box2d & bbox () const`  
*Give the exact bounding box.*
- const `point2d & begin () const`  
*Give the point that begins the line.*
- const `point2d & end () const`  
*Give the point that ends the line.*
- bool `has (const util::index &i) const`

*Test if index  $i$  belongs to this point set.*

- bool `has` (const `psite` &p) const  
*Test if  $p$  belongs to this point set.*
- bool `is_valid` () const  
*Test if this line is valid, i.e., initialized.*
- `std::size_t` `memory_size` () const  
*Return the size of this site set in memory.*
- unsigned `nsites` () const  
*Give the number of points.*
- const `point2d` & `operator[]` (unsigned i) const  
*Return the  $i$ -th point of the line.*
- `p_line2d` (const `point2d` &beg, const `point2d` &end, bool is\_end\_excluded=false)  
*Constructor from point beg to point end.*
- `p_line2d` ()  
*Constructor without argument.*
- const `std::vector< point2d >` & `std_vector` () const  
*Return the corresponding `std::vector` of points.*

### 10.282.1 Detailed Description

2D discrete line of points. It is based on [p\\_array](#).

Definition at line 79 of file `p_line2d.hh`.

### 10.282.2 Member Typedef Documentation

#### 10.282.2.1 typedef `p_indexed_bkd_piter<self_>` `mln::p_line2d::bkd_piter`

Backward [Site\\_Iterator](#) associated type.

Definition at line 97 of file `p_line2d.hh`.

#### 10.282.2.2 typedef `point2d` `mln::p_line2d::element`

Element associated type.

Definition at line 85 of file `p_line2d.hh`.

#### 10.282.2.3 typedef `p_indexed_fwd_piter<self_>` `mln::p_line2d::fwd_piter`

Forward [Site\\_Iterator](#) associated type.

Definition at line 94 of file `p_line2d.hh`.

#### 10.282.2.4 `typedef p_indexed_fwd_piter<self_> mln::p_line2d::piter`

[Site\\_Iterator](#) associated type.

Definition at line 91 of file `p_line2d.hh`.

#### 10.282.2.5 `typedef p_indexed_psite<self_> mln::p_line2d::psite`

`Psite` associated type.

Definition at line 88 of file `p_line2d.hh`.

#### 10.282.2.6 `typedef const box2d& mln::p_line2d::q_box`

`Box` (qualified) associated type.

Definition at line 132 of file `p_line2d.hh`.

### 10.282.3 Constructor & Destructor Documentation

#### 10.282.3.1 `mln::p_line2d::p_line2d ( ) [inline]`

Constructor without argument.

Definition at line 161 of file `p_line2d.hh`.

References `is_valid()`.

#### 10.282.3.2 `mln::p_line2d::p_line2d ( const point2d & beg, const point2d & end, bool is_end_excluded = false ) [inline]`

Constructor from point `beg` to point `end`.

Definition at line 167 of file `p_line2d.hh`.

References `is_valid()`.

### 10.282.4 Member Function Documentation

#### 10.282.4.1 `const box2d & mln::p_line2d::bbox ( ) const [inline]`

Give the exact bounding box.

Definition at line 273 of file `p_line2d.hh`.

References `is_valid()`.

#### 10.282.4.2 `const point2d & mln::p_line2d::begin ( ) const [inline]`

Give the point that begins the line.

Definition at line 307 of file `p_line2d.hh`.

References `is_valid()`.

Referenced by mln::debug::draw\_graph().

#### 10.282.4.3 const point2d & mln::p\_line2d::end ( ) const [inline]

Give the point that ends the line.

Definition at line 315 of file p\_line2d.hh.

References is\_valid(), and nsites().

Referenced by mln::debug::draw\_graph().

#### 10.282.4.4 bool mln::p\_line2d::has ( const psite & p ) const [inline]

Test if *p* belongs to this point set.

Definition at line 240 of file p\_line2d.hh.

#### 10.282.4.5 bool mln::p\_line2d::has ( const util::index & i ) const [inline]

Test if index *i* belongs to this point set.

Definition at line 251 of file p\_line2d.hh.

References nsites().

#### 10.282.4.6 bool mln::p\_line2d::is\_valid ( ) const [inline]

Test if this line is valid, i.e., initialized.

Definition at line 258 of file p\_line2d.hh.

References mln::implies().

Referenced by bbox(), begin(), end(), and p\_line2d().

#### 10.282.4.7 std::size\_t mln::p\_line2d::memory\_size ( ) const [inline]

Return the size of this site set in memory.

Definition at line 323 of file p\_line2d.hh.

#### 10.282.4.8 unsigned mln::p\_line2d::nsites ( ) const [inline]

Give the number of points.

Definition at line 266 of file p\_line2d.hh.

Referenced by end(), has(), and operator[ ]().

#### 10.282.4.9 const point2d & mln::p\_line2d::operator[ ] ( unsigned i ) const [inline]

Return the *i*-th point of the line.

Definition at line 299 of file p\_line2d.hh.

References `nsites()`.

#### 10.282.4.10 `const std::vector< point2d > & mln::p_line2d::std_vector ( ) const [inline]`

Return the corresponding `std::vector` of points.

Definition at line 281 of file `p_line2d.hh`.

## 10.283 `mln::p_mutable_array_of< S >` Class Template Reference

`p_mutable_array_of` is a mutable array of site sets.

```
#include <p_mutable_array_of.hh>
```

Inherits `site_set_base< S::site, p_mutable_array_of< S >>`.

### Public Types

- typedef `p_double_piter< self_, mln_bkd_eiter(array_), typename S::bkd_piter > bkd_piter`  
*Backward `Site_Iterator` associated type.*
- typedef `S element`  
*Element associated type.*
- typedef `p_double_piter< self_, mln_fwd_eiter(array_), typename S::fwd_piter > fwd_piter`  
*Forward `Site_Iterator` associated type.*
- typedef `S i_element`  
*Insertion element associated type.*
- typedef `fwd_piter piter`  
*`Site_Iterator` associated type.*
- typedef `p_double_psite< self_, element > psite`  
*Psite associated type.*

### Public Member Functions

- void `clear ()`  
*Clear this set.*
- bool `has (const psite &p) const`  
*Test if `p` belongs to this point set.*
- void `insert (const S &s)`  
*Insert a site set `s`.*
- bool `is_valid () const`



*Test this set validity so returns always true.*

- `std::size_t memory_size () const`  
*Return the size of this site set in memory.*
- `unsigned nelements () const`  
*Give the number of elements (site sets) of this composite.*
- `S & operator[] (unsigned i)`  
*Return the *i*-th site set (mutable version).*
- `const S & operator[] (unsigned i) const`  
*Return the *i*-th site set (const version).*
- `p_mutable_array_of ()`  
*Constructor without arguments.*
- `void reserve (unsigned n)`  
*Reserve memory for *n* elements.*

### 10.283.1 Detailed Description

`template<typename S> class mln::p_mutable_array_of< S >`

`p_mutable_array_of` is a mutable array of site sets. Parameter `S` is the type of the contained site sets. Definition at line 76 of file `p_mutable_array_of.hh`.

### 10.283.2 Member Typedef Documentation

**10.283.2.1** `template<typename S > typedef p_double_piter<self_, mln_bkd_eiter(array_), typename S ::bkd_piter> mln::p_mutable_array_of< S >::bkd_piter`

Backward `Site_Iterator` associated type.

Definition at line 99 of file `p_mutable_array_of.hh`.

**10.283.2.2** `template<typename S > typedef S mln::p_mutable_array_of< S >::element`

Element associated type.

Definition at line 85 of file `p_mutable_array_of.hh`.

**10.283.2.3** `template<typename S > typedef p_double_piter<self_, mln_fwd_eiter(array_), typename S ::fwd_piter> mln::p_mutable_array_of< S >::fwd_piter`

Forward `Site_Iterator` associated type.

Definition at line 94 of file `p_mutable_array_of.hh`.

**10.283.2.4** `template<typename S > typedef S mln::p_mutable_array_of< S >::i_element`

Insertion element associated type.

Definition at line 121 of file p\_mutable\_array\_of.hh.

**10.283.2.5** `template<typename S > typedef fwd_piter mln::p_mutable_array_of< S >::piter`

[Site\\_Iterator](#) associated type.

Definition at line 102 of file p\_mutable\_array\_of.hh.

**10.283.2.6** `template<typename S > typedef p_double_psite<self_, element>  
mln::p_mutable_array_of< S >::psite`

Psite associated type.

Definition at line 89 of file p\_mutable\_array\_of.hh.

**10.283.3 Constructor & Destructor Documentation****10.283.3.1** `template<typename S > mln::p_mutable_array_of< S >::p_mutable_array_of ( )  
[inline]`

Constructor without arguments.

Definition at line 175 of file p\_mutable\_array\_of.hh.

**10.283.4 Member Function Documentation****10.283.4.1** `template<typename S > void mln::p_mutable_array_of< S >::clear ( ) [inline]`

Clear this set.

Definition at line 241 of file p\_mutable\_array\_of.hh.

**10.283.4.2** `template<typename S > bool mln::p_mutable_array_of< S >::has ( const psite & p )  
const [inline]`

Test if *p* belongs to this point set.

Definition at line 190 of file p\_mutable\_array\_of.hh.

**10.283.4.3** `template<typename S > void mln::p_mutable_array_of< S >::insert ( const S & s )  
[inline]`

Insert a site set *s*.

**Precondition**

*s* is valid.

Definition at line 206 of file p\_mutable\_array\_of.hh.

**10.283.4.4** `template<typename S> bool mln::p_mutable_array_of< S >::is_valid ( ) const [inline]`

Test this set validity so returns always true.

Definition at line 198 of file p\_mutable\_array\_of.hh.

**10.283.4.5** `template<typename S> std::size_t mln::p_mutable_array_of< S >::memory_size ( ) const [inline]`

Return the size of this site set in memory.

Definition at line 258 of file p\_mutable\_array\_of.hh.

**10.283.4.6** `template<typename S> unsigned mln::p_mutable_array_of< S >::nelements ( ) const [inline]`

Give the number of elements (site sets) of this composite.

Definition at line 233 of file p\_mutable\_array\_of.hh.

**10.283.4.7** `template<typename S> S & mln::p_mutable_array_of< S >::operator[] ( unsigned i ) [inline]`

Return the *i*-th site set (mutable version).

Definition at line 224 of file p\_mutable\_array\_of.hh.

**10.283.4.8** `template<typename S> const S & mln::p_mutable_array_of< S >::operator[] ( unsigned i ) const [inline]`

Return the *i*-th site set (const version).

Definition at line 215 of file p\_mutable\_array\_of.hh.

**10.283.4.9** `template<typename S> void mln::p_mutable_array_of< S >::reserve ( unsigned n ) [inline]`

Reserve memory for *n* elements.

Definition at line 182 of file p\_mutable\_array\_of.hh.

## 10.284 mln::p\_n\_faces\_bkd\_piter< D, G > Class Template Reference

Backward iterator on the *n*-faces sites of an mln::p\_complex<D, G>.

```
#include <p_n_faces_piter.hh>
```

Inherits p\_complex\_piter\_base\_< topo::n\_face\_bkd\_iter< D >, p\_complex< D, G >, G::site, p\_n\_faces\_bkd\_piter< D, G > >.

## Public Member Functions

- void `next` ()  
*Go to the next element.*
  
- `p_n_faces_bkd_piter` ()  
*Construction and assignment.*
  
- unsigned `n` () const  
*Accessors.*

### 10.284.1 Detailed Description

`template<unsigned D, typename G> class mln::p_n_faces_bkd_piter< D, G >`

Backward iterator on the n-faces sites of an `mln::p_complex<D, G>`.

Definition at line 92 of file `p_n_faces_piter.hh`.

### 10.284.2 Constructor & Destructor Documentation

**10.284.2.1** `template<unsigned D, typename G > mln::p_n_faces_bkd_piter< D, G >::p_n_faces_bkd_piter ( ) [inline]`

Construction and assignment.

Definition at line 169 of file `p_n_faces_piter.hh`.

### 10.284.3 Member Function Documentation

**10.284.3.1** `template<unsigned D, typename G > unsigned mln::p_n_faces_bkd_piter< D, G >::n ( ) const [inline]`

Accessors.

Shortcuts to `face_`'s accessors.

Definition at line 186 of file `p_n_faces_piter.hh`.

**10.284.3.2** `void mln::Site_Iterator< p_n_faces_bkd_piter< D, G > >::next ( ) [inherited]`

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

#### Precondition

The iterator is valid.

## 10.285 mln::p\_n\_faces\_fwd\_piter< D, G > Class Template Reference

Forward iterator on the n-faces sites of an mln::p\_complex<D, G>.

```
#include <p_n_faces_piter.hh>
```

Inherits p\_complex\_piter\_base\_< topo::n\_face\_fwd\_iter< D >, p\_complex< D, G >, G::site, p\_n\_faces\_fwd\_piter< D, G > >.

### Public Member Functions

- void [next](#) ()  
*Go to the next element.*
- [p\\_n\\_faces\\_fwd\\_piter](#) ()  
*Construction and assignment.*
- unsigned [n](#) () const  
*Accessors.*

### 10.285.1 Detailed Description

```
template<unsigned D, typename G> class mln::p_n_faces_fwd_piter< D, G >
```

Forward iterator on the n-faces sites of an mln::p\_complex<D, G>.

Definition at line 56 of file p\_n\_faces\_piter.hh.

### 10.285.2 Constructor & Destructor Documentation

**10.285.2.1** `template<unsigned D, typename G > mln::p_n_faces_fwd_piter< D, G >::p_n_faces_fwd_piter ( ) [inline]`

Construction and assignment.

Definition at line 132 of file p\_n\_faces\_piter.hh.

### 10.285.3 Member Function Documentation

**10.285.3.1** `template<unsigned D, typename G > unsigned mln::p_n_faces_fwd_piter< D, G >::n ( ) const [inline]`

Accessors.

Shortcuts to face\_'s accessors.

Definition at line 149 of file p\_n\_faces\_piter.hh.

**10.285.3.2 void mln::Site\_Iterator< p\_n\_faces\_fwd\_piter< D, G >>::next( ) [inherited]**

Go to the next element.

**Warning**

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

**Precondition**

The iterator is valid.

**10.286 mln::p\_priority< P, Q > Class Template Reference**

Priority queue.

```
#include <p_priority.hh>
```

Inherits site\_set\_base\_< Q::site, p\_priority< P, Q >>.

**Public Types**

- typedef p\_double\_piter< [self\\_](#), mln\_fwd\_eiter(util::set< P >), typename Q::bkd\_piter > [bkd\\_piter](#)  
*Backward Site\_Iterator associated type.*
- typedef Q::element [element](#)  
*Element associated type.*
- typedef p\_double\_piter< [self\\_](#), mln\_bkd\_eiter(util::set< P >), typename Q::fwd\_piter > [fwd\\_piter](#)  
*Forward Site\_Iterator associated type.*
- typedef std::pair< P, [element](#) > [i\\_element](#)  
*Insertion element associated type.*
- typedef fwd\_piter [piter](#)  
*Site\_Iterator associated type.*
- typedef p\_double\_psite< [self\\_](#), Q > [psite](#)  
*Psite associated type.*

**Public Member Functions**

- void [clear](#) ()  
*Clear the queue.*
- bool [exists\\_priority](#) (const P &priority) const  
*Test if the priority exists.*

- const Q::element & **front** () const  
*Give an element with highest priority.*
- bool **has** (const psite &) const  
*Test is the psite p belongs to this site set.*
- const P **highest\_priority** () const  
*Give the highest priority.*
- void **insert** (const i\_element &p\_e)  
*Insert a pair p\_e (priority p, element e).*
- void **insert** (const p\_priority< P, Q > &other)  
*Insert elements from another priority queue.*
- bool **is\_valid** () const  
*Test this set validity so returns always true.*
- const P **lowest\_priority** () const  
*Give the lowest priority.*
- std::size\_t **memory\_size** () const  
*Return the size of this site set in memory.*
- unsigned **nsites** () const  
*Give the number of sites.*
- const Q & **operator()** (const P &priority) const  
*Give the queue with the priority priority.*
- **p\_priority** ()  
*Constructor.*
- void **pop** ()  
*Pop (remove) from the queue an element with highest priority.*
- Q::element **pop\_front** ()  
*Return an element with highest priority and remove it from the set.*
- const util::set< P > & **priorities** () const  
*Give the set of priorities.*
- void **push** (const P &priority, const element &e)  
*Push in the queue with priority the element e.*

## 10.286.1 Detailed Description

**template<typename P, typename Q> class mln::p\_priority< P, Q >**

Priority queue. The parameter P is the type of the priorities (for instance unsigned).

The parameter Q is a type of queue (for instance p\_queue<point2d>).

Definition at line 76 of file p\_priority.hh.

## 10.286.2 Member Typedef Documentation

**10.286.2.1 template<typename P, typename Q> typedef p\_double\_piter< self\_, mln\_fwd\_eiter(util::set<P>), typename Q ::bkd\_piter > mln::p\_priority< P, Q >::bkd\_piter**

Backward [Site\\_Iterator](#) associated type.

Definition at line 98 of file p\_priority.hh.

**10.286.2.2 template<typename P, typename Q> typedef Q ::element mln::p\_priority< P, Q >::element**

Element associated type.

Definition at line 84 of file p\_priority.hh.

**10.286.2.3 template<typename P, typename Q> typedef p\_double\_piter< self\_, mln\_bkd\_eiter(util::set<P>), typename Q ::fwd\_piter > mln::p\_priority< P, Q >::fwd\_piter**

Forward [Site\\_Iterator](#) associated type.

Definition at line 93 of file p\_priority.hh.

**10.286.2.4 template<typename P, typename Q> typedef std::pair<P, element> mln::p\_priority< P, Q >::i\_element**

Insertion element associated type.

Definition at line 121 of file p\_priority.hh.

**10.286.2.5 template<typename P, typename Q> typedef fwd\_piter mln::p\_priority< P, Q >::piter**

[Site\\_Iterator](#) associated type.

Definition at line 101 of file p\_priority.hh.

**10.286.2.6 template<typename P, typename Q> typedef p\_double\_psite<self\_, Q> mln::p\_priority< P, Q >::psite**

Psite associated type.

Definition at line 88 of file p\_priority.hh.



### 10.286.3 Constructor & Destructor Documentation

**10.286.3.1** `template<typename P , typename Q > mln::p_priority< P, Q >::p_priority ( )`  
`[inline]`

Constructor.

Definition at line 202 of file p\_priority.hh.

### 10.286.4 Member Function Documentation

**10.286.4.1** `template<typename P , typename Q > void mln::p_priority< P, Q >::clear ( )`  
`[inline]`

Clear the queue.

Definition at line 316 of file p\_priority.hh.

**10.286.4.2** `template<typename P , typename Q > bool mln::p_priority< P, Q >::exists_priority (`  
`const P & priority ) const [inline]`

Test if the `priority` exists.

Definition at line 366 of file p\_priority.hh.

Referenced by `mln::p_priority< P, Q >::operator()`.

**10.286.4.3** `template<typename P , typename Q > const Q::element & mln::p_priority< P, Q`  
`>::front ( ) const [inline]`

Give an element with highest priority.

If several elements have this priority, the least recently inserted is chosen.

#### Precondition

`! is_empty()`

Definition at line 294 of file p\_priority.hh.

References `mln::p_priority< P, Q >::highest_priority()`.

Referenced by `mln::morpho::meyer_wst()`, and `mln::morpho::watershed::topological()`.

**10.286.4.4** `template<typename P , typename Q > bool mln::p_priority< P, Q >::has ( const psite`  
`& ) const [inline]`

Test is the psite `p` belongs to this site set.

Definition at line 211 of file p\_priority.hh.

**10.286.4.5** `template<typename P , typename Q > const P mln::p_priority< P, Q`  
`>::highest_priority ( ) const [inline]`

Give the highest priority.

**Precondition**

! is\_empty()

Definition at line 375 of file p\_priority.hh.

Referenced by mln::p\_priority< P, Q >::front(), and mln::p\_priority< P, Q >::pop().

**10.286.4.6** `template<typename P , typename Q > void mln::p_priority< P, Q >::insert ( const i_element & p_e ) [inline]`

Insert a pair p\_e (priority p, element e).

Definition at line 251 of file p\_priority.hh.

References mln::p\_priority< P, Q >::push().

**10.286.4.7** `template<typename P , typename Q > void mln::p_priority< P, Q >::insert ( const p_priority< P, Q > & other ) [inline]`

Insert elements from another priority queue.

Definition at line 259 of file p\_priority.hh.

**10.286.4.8** `template<typename P , typename Q > bool mln::p_priority< P, Q >::is_valid ( ) const [inline]`

Test this set validity so returns always true.

Definition at line 221 of file p\_priority.hh.

**10.286.4.9** `template<typename P , typename Q > const P mln::p_priority< P, Q >::lowest_priority ( ) const [inline]`

Give the lowest priority.

**Precondition**

! is\_empty()

Definition at line 384 of file p\_priority.hh.

**10.286.4.10** `template<typename P , typename Q > std::size_t mln::p_priority< P, Q >::memory_size ( ) const [inline]`

Return the size of this site set in memory.

Definition at line 328 of file p\_priority.hh.

**10.286.4.11** `template<typename P , typename Q > unsigned mln::p_priority< P, Q >::nsites ( ) const [inline]`

Give the number of sites.

Definition at line 230 of file p\_priority.hh.

Referenced by mln::p\_priority< P, Q >::operator().

**10.286.4.12** `template<typename P , typename Q > const Q & mln::p_priority< P, Q >::operator()  
( const P & priority ) const [inline]`

Give the queue with the priority `priority`.

This method always works: if the priority is not in this set, an empty queue is returned.

Definition at line 341 of file p\_priority.hh.

References mln::p\_priority< P, Q >::exists\_priority(), and mln::p\_priority< P, Q >::nsites().

**10.286.4.13** `template<typename P , typename Q > void mln::p_priority< P, Q >::pop ( )  
[inline]`

Pop (remove) from the queue an element with highest priority.

If several elements have this priority, the least recently inserted is chosen.

#### Precondition

`! is_empty()`

Definition at line 277 of file p\_priority.hh.

References mln::p\_priority< P, Q >::highest\_priority().

Referenced by mln::morpho::meyer\_wst(), and mln::morpho::watershed::topological().

**10.286.4.14** `template<typename P , typename Q > Q::element mln::p_priority< P, Q  
>::pop_front ( ) [inline]`

Return an element with highest priority and remove it from the set.

If several elements have this priority, the least recently inserted is chosen.

#### Precondition

`! is_empty()`

Definition at line 304 of file p\_priority.hh.

**10.286.4.15** `template<typename P , typename Q > const util::set< P > & mln::p_priority< P, Q  
>::priorities ( ) const [inline]`

Give the set of priorities.

Definition at line 357 of file p\_priority.hh.

**10.286.4.16** `template<typename P , typename Q > void mln::p_priority< P, Q >::push ( const P  
& priority, const element & e ) [inline]`

Push in the queue with `priority` the element `e`.

Definition at line 239 of file p\_priority.hh.

Referenced by `mln::p_priority< P, Q >::insert()`, `mln::morpho::meyer_wst()`, and `mln::morpho::watershed::topological()`.

## 10.287 mln::p\_queue< P > Class Template Reference

Queue of sites (based on `std::deque`).

```
#include <p_queue.hh>
```

Inherits `site_set_base_< P, p_queue< P > >`.

### Public Types

- typedef `p_indexed_bkd_piter< self_ > bkd_piter`  
*Backward [Site\\_Iterator](#) associated type.*
- typedef `P element`  
*Element associated type.*
- typedef `p_indexed_fwd_piter< self_ > fwd_piter`  
*Forward [Site\\_Iterator](#) associated type.*
- typedef `P i_element`  
*Insertion element associated type.*
- typedef `fwd_piter piter`  
*[Site\\_Iterator](#) associated type.*
- typedef `p_indexed_psite< self_ > psite`  
*Psite associated type.*

### Public Member Functions

- void `clear ()`  
*Clear the queue.*
- const `P & front () const`  
*Give the front site `p` of the queue; `p` is the least recently inserted site.*
- bool `has (const psite &p) const`  
*Test if `p` belongs to this site set.*
- bool `has (const util::index &i) const`  
*Test if index `i` belongs to this site set.*
- void `insert (const P &p)`  
*Insert a site `p` (equivalent as 'push').*

- bool `is_valid ()` const  
*This set is always valid so it returns true.*
- std::size\_t `memory_size ()` const  
*Return the size of this site set in memory.*
- unsigned `nsites ()` const  
*Give the number of sites.*
- const P & `operator[]` (unsigned i) const  
*Return the *i*-th site.*
- `p_queue ()`  
*Constructor without argument.*
- void `pop ()`  
*Pop (remove) the front site *p* from the queue; *p* is the least recently inserted site.*
- P `pop_front ()`  
*Pop (remove) the front site *p* from the queue; *p* is the least recently inserted site and give the front site *p* of the queue; *p* is the least recently inserted site.*
- void `push` (const P &p)  
*Push a site *p* in the queue.*
- const std::deque< P > & `std_deque ()` const  
*Return the corresponding std::deque of sites.*

### 10.287.1 Detailed Description

**template<typename P> class mln::p\_queue< P >**

Queue of sites (based on std::deque). The parameter P shall be a site or pseudo-site type.

Definition at line 74 of file p\_queue.hh.

### 10.287.2 Member Typedef Documentation

**10.287.2.1 template<typename P> typedef p\_indexed\_bkd\_piter<self\_> mln::p\_queue< P >::bkd\_piter**

Backward [Site\\_Iterator](#) associated type.

Definition at line 90 of file p\_queue.hh.

**10.287.2.2 template<typename P> typedef P mln::p\_queue< P >::element**

Element associated type.

Definition at line 80 of file p\_queue.hh.

**10.287.2.3** `template<typename P> typedef p_indexed_fwd_piter<self_> mln::p_queue< P >::fwd_piter`

Forward [Site\\_Iterator](#) associated type.

Definition at line 87 of file p\_queue.hh.

**10.287.2.4** `template<typename P> typedef P mln::p_queue< P >::i_element`

Insertion element associated type.

Definition at line 118 of file p\_queue.hh.

**10.287.2.5** `template<typename P> typedef fwd_piter mln::p_queue< P >::piter`

[Site\\_Iterator](#) associated type.

Definition at line 93 of file p\_queue.hh.

**10.287.2.6** `template<typename P> typedef p_indexed_psite<self_> mln::p_queue< P >::psite`

Psite associated type.

Definition at line 84 of file p\_queue.hh.

### 10.287.3 Constructor & Destructor Documentation

**10.287.3.1** `template<typename P> mln::p_queue< P >::p_queue ( ) [inline]`

Constructor without argument.

Definition at line 163 of file p\_queue.hh.

### 10.287.4 Member Function Documentation

**10.287.4.1** `template<typename P> void mln::p_queue< P >::clear ( ) [inline]`

Clear the queue.

Definition at line 244 of file p\_queue.hh.

**10.287.4.2** `template<typename P> const P & mln::p_queue< P >::front ( ) const [inline]`

Give the front site *p* of the queue; *p* is the least recently inserted site.

Definition at line 224 of file p\_queue.hh.

Referenced by `mln::p_queue< P >::pop_front()`, and `mln::geom::impl::seeds2tiling()`.

**10.287.4.3** `template<typename P> bool mln::p_queue< P >::has ( const util::index & i ) const [inline]`

Test if index *i* belongs to this site set.

Definition at line 183 of file p\_queue.hh.

References mln::p\_queue< P >::nsites().

**10.287.4.4** `template<typename P > bool mln::p_queue< P >::has ( const psite & p ) const [inline]`

Test if *p* belongs to this site set.

Definition at line 170 of file p\_queue.hh.

References mln::p\_queue< P >::nsites().

**10.287.4.5** `template<typename P > void mln::p_queue< P >::insert ( const P & p ) [inline]`

Insert a site *p* (equivalent as 'push').

Definition at line 261 of file p\_queue.hh.

References mln::p\_queue< P >::push().

**10.287.4.6** `template<typename P > bool mln::p_queue< P >::is_valid ( ) const [inline]`

This set is always valid so it returns true.

Definition at line 191 of file p\_queue.hh.

**10.287.4.7** `template<typename P > std::size_t mln::p_queue< P >::memory_size ( ) const [inline]`

Return the size of this site set in memory.

Definition at line 277 of file p\_queue.hh.

References mln::p\_queue< P >::nsites().

**10.287.4.8** `template<typename P > unsigned mln::p_queue< P >::nsites ( ) const [inline]`

Give the number of sites.

Definition at line 199 of file p\_queue.hh.

Referenced by mln::p\_queue< P >::has(), mln::p\_queue< P >::memory\_size(), and mln::p\_queue< P >::operator[ ]().

**10.287.4.9** `template<typename P > const P & mln::p_queue< P >::operator[ ] ( unsigned i ) const [inline]`

Return the *i*-th site.

Definition at line 252 of file p\_queue.hh.

References mln::p\_queue< P >::nsites().

**10.287.4.10** `template<typename P> void mln::p_queue< P >::pop ( ) [inline]`

Pop (remove) the front site `p` from the queue; `p` is the least recently inserted site.

Definition at line 215 of file `p_queue.hh`.

Referenced by `mln::p_queue< P >::pop_front()`, and `mln::geom::impl::seeds2tiling()`.

**10.287.4.11** `template<typename P> P mln::p_queue< P >::pop_front ( ) [inline]`

Pop (remove) the front site `p` from the queue; `p` is the least recently inserted site and give the front site `p` of the queue; `p` is the least recently inserted site.

Definition at line 233 of file `p_queue.hh`.

References `mln::p_queue< P >::front()`, and `mln::p_queue< P >::pop()`.

**10.287.4.12** `template<typename P> void mln::p_queue< P >::push ( const P & p ) [inline]`

Push a site `p` in the queue.

Definition at line 207 of file `p_queue.hh`.

Referenced by `mln::p_queue< P >::insert()`, and `mln::geom::impl::seeds2tiling()`.

**10.287.4.13** `template<typename P> const std::deque< P > & mln::p_queue< P >::std_deque ( ) const [inline]`

Return the corresponding `std::deque` of sites.

Definition at line 269 of file `p_queue.hh`.

**10.288** `mln::p_queue_fast< P >` Class Template Reference

Queue of sites class (based on [p\\_array](#)).

```
#include <p_queue_fast.hh>
```

Inherits `site_set_base_< P, p_queue_fast< P >>`.

**Public Types**

- typedef `p_indexed_bkd_piter< self_ > bkd_piter`  
*Backward [Site\\_Iterator](#) associated type.*
- typedef `P element`  
*Element associated type.*
- typedef `p_indexed_fwd_piter< self_ > fwd_piter`  
*Forward [Site\\_Iterator](#) associated type.*
- typedef `P i_element`  
*Insertion element associated type.*



- typedef [fwd\\_piter](#) piter  
*Site\_iterator associated type.*
- typedef [p\\_indexed\\_psite](#)< self\_ > psite  
*Psite associated type.*

## Public Member Functions

- void [clear](#) ()  
*Clear the queue.*
- bool [compute\\_has](#) (const P &p) const  
*Test if p belongs to this site set.*
- bool [empty](#) () const  
*Test if the queue is empty.*
- const P & [front](#) () const  
*Give the front site p of the queue; p is the least recently inserted site.*
- bool [has](#) (const util::index &i) const  
*Test if index i belongs to this site set.*
- bool [has](#) (const psite &p) const  
*Test if p belongs to this site set.*
- void [insert](#) (const P &p)  
*Insert a site p (equivalent as 'push').*
- bool [is\\_valid](#) () const  
*This set is always valid so it returns true.*
- std::size\_t [memory\\_size](#) () const  
*Return the size of this site set in memory.*
- unsigned [nsites](#) () const  
*Give the number of sites.*
- const P & [operator\[\]](#) (unsigned i) const  
*Return the i-th site.*
- [p\\_queue\\_fast](#) ()  
*Constructor without argument.*
- void [pop](#) ()  
*Pop (remove) the front site p from the queue; p is the least recently inserted site.*

- `const P & pop_front ()`  
*Pop (remove) the front site  $p$  from the queue;  $p$  is the least recently inserted site and give the front site  $p$  of the queue;  $p$  is the least recently inserted site.*
- `void purge ()`  
*Purge the queue to save (free) some memory.*
- `void push (const P &p)`  
*Push a site  $p$  in the queue.*
- `void reserve (typename p_array< P >::size_type n)`  
*Reserve  $n$  cells.*
- `const std::vector< P > & std_vector () const`  
*Return the corresponding `std::vector` of sites.*

## 10.288.1 Detailed Description

`template<typename P> class mln::p_queue_fast< P >`

Queue of sites class (based on `p_array` ).

This container is efficient; FIXME: explain...

The parameter `P` shall be a site or pseudo-site type.

Definition at line 72 of file `p_queue_fast.hh`.

## 10.288.2 Member Typedef Documentation

**10.288.2.1** `template<typename P > typedef p_indexed_bkd_piter<self_> mln::p_queue_fast< P >::bkd_piter`

Backward `Site_Iterator` associated type.

Definition at line 87 of file `p_queue_fast.hh`.

**10.288.2.2** `template<typename P > typedef P mln::p_queue_fast< P >::element`

Element associated type.

Definition at line 78 of file `p_queue_fast.hh`.

**10.288.2.3** `template<typename P > typedef p_indexed_fwd_piter<self_> mln::p_queue_fast< P >::fwd_piter`

Forward `Site_Iterator` associated type.

Definition at line 84 of file `p_queue_fast.hh`.

**10.288.2.4** `template<typename P > typedef P mln::p_queue_fast< P >::i_element`

Insertion element associated type.

Definition at line 121 of file p\_queue\_fast.hh.

**10.288.2.5** `template<typename P > typedef fwd_piter mln::p_queue_fast< P >::piter`

[Site\\_Iterator](#) associated type.

Definition at line 90 of file p\_queue\_fast.hh.

**10.288.2.6** `template<typename P > typedef p_indexed_psite<self_> mln::p_queue_fast< P >::psite`

Psite associated type.

Definition at line 81 of file p\_queue\_fast.hh.

**10.288.3 Constructor & Destructor Documentation****10.288.3.1** `template<typename P > mln::p_queue_fast< P >::p_queue_fast ( ) [inline]`

Constructor without argument.

Definition at line 170 of file p\_queue\_fast.hh.

**10.288.4 Member Function Documentation****10.288.4.1** `template<typename P > void mln::p_queue_fast< P >::clear ( ) [inline]`

Clear the queue.

Definition at line 297 of file p\_queue\_fast.hh.

**10.288.4.2** `template<typename P > bool mln::p_queue_fast< P >::compute_has ( const P & p ) const [inline]`

Test if p belongs to this site set.

Definition at line 222 of file p\_queue\_fast.hh.

**10.288.4.3** `template<typename P > bool mln::p_queue_fast< P >::empty ( ) const [inline]`

Test if the queue is empty.

Definition at line 250 of file p\_queue\_fast.hh.

**10.288.4.4** `template<typename P > const P & mln::p_queue_fast< P >::front ( ) const [inline]`

Give the front site p of the queue; p is the least recently inserted site.

Definition at line 277 of file p\_queue\_fast.hh.

Referenced by mln::p\_queue\_fast< P >::pop\_front().

**10.288.4.5** `template<typename P> bool mln::p_queue_fast< P >::has ( const psite & p ) const [inline]`

Test if *p* belongs to this site set.

Definition at line 201 of file p\_queue\_fast.hh.

References mln::p\_queue\_fast< P >::nsites().

**10.288.4.6** `template<typename P> bool mln::p_queue_fast< P >::has ( const util::index & i ) const [inline]`

Test if index *i* belongs to this site set.

Definition at line 214 of file p\_queue\_fast.hh.

References mln::p\_queue\_fast< P >::nsites().

**10.288.4.7** `template<typename P> void mln::p_queue_fast< P >::insert ( const P & p ) [inline]`

Insert a site *p* (equivalent as 'push').

Definition at line 314 of file p\_queue\_fast.hh.

References mln::p\_queue\_fast< P >::push().

**10.288.4.8** `template<typename P> bool mln::p_queue_fast< P >::is_valid ( ) const [inline]`

This set is always valid so it returns true.

Definition at line 233 of file p\_queue\_fast.hh.

**10.288.4.9** `template<typename P> std::size_t mln::p_queue_fast< P >::memory_size ( ) const [inline]`

Return the size of this site set in memory.

Definition at line 330 of file p\_queue\_fast.hh.

**10.288.4.10** `template<typename P> unsigned mln::p_queue_fast< P >::nsites ( ) const [inline]`

Give the number of sites.

Definition at line 241 of file p\_queue\_fast.hh.

Referenced by mln::p\_queue\_fast< P >::has(), and mln::p\_queue\_fast< P >::operator[]().

**10.288.4.11** `template<typename P> const P & mln::p_queue_fast< P >::operator[] ( unsigned i ) const [inline]`

Return the *i*-th site.

Definition at line 305 of file p\_queue\_fast.hh.

References mln::p\_queue\_fast< P >::nsites().

**10.288.4.12** `template<typename P> void mln::p_queue_fast< P >::pop ( ) [inline]`

Pop (remove) the front site *p* from the queue; *p* is the least recently inserted site.

Definition at line 268 of file p\_queue\_fast.hh.

Referenced by mln::p\_queue\_fast< P >::pop\_front().

**10.288.4.13** `template<typename P> const P & mln::p_queue_fast< P >::pop_front ( ) [inline]`

Pop (remove) the front site *p* from the queue; *p* is the least recently inserted site and give the front site *p* of the queue; *p* is the least recently inserted site.

Definition at line 286 of file p\_queue\_fast.hh.

References mln::p\_queue\_fast< P >::front(), and mln::p\_queue\_fast< P >::pop().

**10.288.4.14** `template<typename P> void mln::p_queue_fast< P >::purge ( ) [inline]`

Purge the queue to save (free) some memory.

Definition at line 187 of file p\_queue\_fast.hh.

**10.288.4.15** `template<typename P> void mln::p_queue_fast< P >::push ( const P & p ) [inline]`

Push a site *p* in the queue.

Definition at line 259 of file p\_queue\_fast.hh.

Referenced by mln::p\_queue\_fast< P >::insert().

**10.288.4.16** `template<typename P> void mln::p_queue_fast< P >::reserve ( typename p_array< P >::size_type n ) [inline]`

Reserve *n* cells.

Definition at line 179 of file p\_queue\_fast.hh.

**10.288.4.17** `template<typename P> const std::vector< P > & mln::p_queue_fast< P >::std_vector ( ) const [inline]`

Return the corresponding std::vector of sites.

Definition at line 322 of file p\_queue\_fast.hh.

## 10.289 mln::p\_run< P > Class Template Reference

[Point](#) set class in run.

```
#include <p_run.hh>
```

Inherits [site\\_set\\_base\\_< P, p\\_run< P > >](#).

### Public Types

- typedef [p\\_run\\_bkd\\_piter\\_< P >](#) [bkd\\_piter](#)  
*Backward [Site\\_Iterator](#) associated type.*
- typedef P [element](#)  
*Element associated type.*
- typedef [p\\_run\\_fwd\\_piter\\_< P >](#) [fwd\\_piter](#)  
*Forward [Site\\_Iterator](#) associated type.*
- typedef [fwd\\_piter](#) [piter](#)  
*[Site\\_Iterator](#) associated type.*
- typedef [p\\_run\\_psite< P >](#) [psite](#)  
*Psite associated type.*
- typedef [mln::box< P >](#) [q\\_box](#)  
*Box associated type.*

### Public Member Functions

- [mln::box< P >](#) [bbox](#) () const  
*Give the exact bounding box.*
- P [end](#) () const  
*Return (compute) the ending point.*
- bool [has](#) (const P &p) const  
*Test if *p* belongs to this point set.*
- bool [has](#) (const [psite](#) &p) const  
*Test if *p* belongs to this point set.*
- bool [has\\_index](#) (unsigned short i) const  
*Test if index *i* belongs to this point set.*
- void [init](#) (const P &start, unsigned short len)  
*Set the starting point.*
- bool [is\\_valid](#) () const

*Test if this run is valid, i.e., with length > 0.*

- unsigned short [length](#) () const  
*Give the length of the run.*
- std::size\_t [memory\\_size](#) () const  
*Return the size of this site set in memory.*
- unsigned [nsites](#) () const  
*Give the number of sites.*
- P [operator\[\]](#) (unsigned short i) const  
*Return the *i*-th point.*
- [p\\_run](#) ()  
*Constructor without argument.*
- [p\\_run](#) (const P &start, unsigned short len)  
*Constructor.*
- [p\\_run](#) (const P &start, const P &end)  
*Constructor.*
- const P & [start](#) () const  
*Return the starting point.*

### 10.289.1 Detailed Description

`template<typename P> class mln::p_run< P >`

[Point](#) set class in run. This is a mathematical set of points (not a multi-set). The parameter P shall be a [Point](#) type.

Definition at line 86 of file p\_run.hh.

### 10.289.2 Member Typedef Documentation

**10.289.2.1** `template<typename P> typedef p_run_bkd_piter_<P> mln::p_run< P >::bkd_piter`

Backward [Site\\_Iterator](#) associated type.

Definition at line 101 of file p\_run.hh.

**10.289.2.2** `template<typename P> typedef P mln::p_run< P >::element`

Element associated type.

Definition at line 91 of file p\_run.hh.

**10.289.2.3** `template<typename P> typedef p_run_fwd_piter_<P> mln::p_run< P >::fwd_piter`

Forward [Site\\_Iterator](#) associated type.

Definition at line 98 of file p\_run.hh.

**10.289.2.4** `template<typename P> typedef fwd_piter mln::p_run< P >::piter`

[Site\\_Iterator](#) associated type.

Definition at line 104 of file p\_run.hh.

**10.289.2.5** `template<typename P> typedef p_run_psite<P> mln::p_run< P >::psite`

Psite associated type.

Definition at line 95 of file p\_run.hh.

**10.289.2.6** `template<typename P> typedef mln::box<P> mln::p_run< P >::q_box`

[Box](#) associated type.

Definition at line 149 of file p\_run.hh.

**10.289.3** **Constructor & Destructor Documentation****10.289.3.1** `template<typename P > mln::p_run< P >::p_run ( ) [inline]`

Constructor without argument.

Definition at line 223 of file p\_run.hh.

**10.289.3.2** `template<typename P > mln::p_run< P >::p_run ( const P & start, unsigned short len ) [inline]`

Constructor.

Definition at line 230 of file p\_run.hh.

References `mln::p_run< P >::init()`.

**10.289.3.3** `template<typename P > mln::p_run< P >::p_run ( const P & start, const P & end ) [inline]`

Constructor.

Definition at line 238 of file p\_run.hh.



## 10.289.4 Member Function Documentation

**10.289.4.1** `template<typename P > mln::box< P > mln::p_run< P >::bbox ( ) const [inline]`

Give the exact bounding box.

Definition at line 267 of file p\_run.hh.

References mln::p\_run< P >::end().

**10.289.4.2** `template<typename P > P mln::p_run< P >::end ( ) const [inline]`

Return (compute) the ending point.

Definition at line 348 of file p\_run.hh.

References mln::point< G, C >::last\_coord().

Referenced by mln::p\_run< P >::bbox().

**10.289.4.3** `template<typename P > bool mln::p_run< P >::has ( const psite & p ) const [inline]`

Test if *p* belongs to this point set.

Definition at line 276 of file p\_run.hh.

**10.289.4.4** `template<typename P > bool mln::p_run< P >::has ( const P & p ) const [inline]`

Test if *p* belongs to this point set.

Definition at line 289 of file p\_run.hh.

References mln::p\_run< P >::is\_valid().

**10.289.4.5** `template<typename P > bool mln::p_run< P >::has_index ( unsigned short i ) const [inline]`

Test if index *i* belongs to this point set.

Definition at line 302 of file p\_run.hh.

**10.289.4.6** `template<typename P > void mln::p_run< P >::init ( const P & start, unsigned short len ) [inline]`

Set the starting point.

Definition at line 249 of file p\_run.hh.

Referenced by mln::p\_run< P >::p\_run().

**10.289.4.7** `template<typename P> bool mln::p_run< P >::is_valid ( ) const [inline]`

Test if this run is valid, i.e., with length > 0.

Definition at line 259 of file p\_run.hh.

Referenced by `mln::p_run< P >::has()`, `mln::p_run< P >::length()`, `mln::p_run< P >::nsites()`, and `mln::p_run< P >::operator[]()`.

**10.289.4.8** `template<typename P> unsigned short mln::p_run< P >::length ( ) const [inline]`

Give the length of the run.

Definition at line 319 of file p\_run.hh.

References `mln::p_run< P >::is_valid()`.

**10.289.4.9** `template<typename P> std::size_t mln::p_run< P >::memory_size ( ) const [inline]`

Return the size of this site set in memory.

Definition at line 358 of file p\_run.hh.

**10.289.4.10** `template<typename P> unsigned mln::p_run< P >::nsites ( ) const [inline]`

Give the number of sites.

Definition at line 310 of file p\_run.hh.

References `mln::p_run< P >::is_valid()`.

**10.289.4.11** `template<typename P> P mln::p_run< P >::operator[] ( unsigned short i ) const [inline]`

Return the *i*-th point.

Definition at line 328 of file p\_run.hh.

References `mln::p_run< P >::is_valid()`, and `mln::point< G, C >::last_coord()`.

**10.289.4.12** `template<typename P> const P & mln::p_run< P >::start ( ) const [inline]`

Return the starting point.

Definition at line 340 of file p\_run.hh.

**10.290** `mln::p_set< P >` Class Template Reference

Mathematical set of sites (based on [util::set](#)).

```
#include <p_set.hh>
```

Inherits `site_set_base< P, p_set< P >>`.

## Public Types

- typedef `p_indexed_bkd_piter< self_ > bkd_piter`  
*Backward [Site\\_Iterator](#) associated type.*
- typedef `P element`  
*Element associated type.*
- typedef `p_indexed_fwd_piter< self_ > fwd_piter`  
*Forward [Site\\_Iterator](#) associated type.*
- typedef `P i_element`  
*Insertion element associated type.*
- typedef `fwd_piter piter`  
*[Site\\_Iterator](#) associated type.*
- typedef `p_indexed_psite< self_ > psite`  
*Psite associated type.*
- typedef `P r_element`  
*Removal element associated type.*

## Public Member Functions

- void `clear ()`  
*Clear this set.*
- bool `has (const psite &p) const`  
*Test if psite *p* belongs to this point set.*
- bool `has (const util::index &i) const`  
*Test if index *i* belongs to this point set.*
- bool `has (const P &p) const`  
*Test if *p* belongs to this point set.*
- void `insert (const P &p)`  
*Insert a site *p*.*
- bool `is_valid () const`  
*Test this set validity so returns always true.*
- `std::size_t memory_size () const`  
*Return the size of this site set in memory.*
- unsigned `nsites () const`  
*Give the number of sites.*

- `const P & operator[] (unsigned i) const`  
*Return the  $i$ -th site.*
- `p_set ()`  
*Constructor.*
- `void remove (const P &p)`  
*Remove a site  $p$ .*
- `const std::vector< P > & std_vector () const`  
*Return the corresponding `std::vector` of sites.*
- `const util::set< P > & util_set () const`  
*Return the corresponding `util::set` of sites.*

### 10.290.1 Detailed Description

`template<typename P> class mln::p_set< P >`

Mathematical set of sites (based on `util::set`). This is a mathematical set of sites (not a multi-set).

The parameter `P` shall be a site or pseudo-site type.

Definition at line 70 of file `p_set.hh`.

### 10.290.2 Member Typedef Documentation

**10.290.2.1** `template<typename P> typedef p_indexed_bkd_piter<self_> mln::p_set< P >::bkd_piter`

Backward `Site_Iterator` associated type.

Definition at line 85 of file `p_set.hh`.

**10.290.2.2** `template<typename P> typedef P mln::p_set< P >::element`

Element associated type.

Definition at line 76 of file `p_set.hh`.

**10.290.2.3** `template<typename P> typedef p_indexed_fwd_piter<self_> mln::p_set< P >::fwd_piter`

Forward `Site_Iterator` associated type.

Definition at line 82 of file `p_set.hh`.

**10.290.2.4** `template<typename P> typedef P mln::p_set< P >::i_element`

Insertion element associated type.

Definition at line 113 of file `p_set.hh`.

**10.290.2.5** `template<typename P> typedef fwd_piter mln::p_set< P >::piter`

[Site\\_Iterator](#) associated type.

Definition at line 88 of file p\_set.hh.

**10.290.2.6** `template<typename P> typedef p_indexed_psite<self_> mln::p_set< P >::psite`

Psite associated type.

Definition at line 79 of file p\_set.hh.

**10.290.2.7** `template<typename P> typedef P mln::p_set< P >::r_element`

Removal element associated type.

Definition at line 119 of file p\_set.hh.

**10.290.3** **Constructor & Destructor Documentation****10.290.3.1** `template<typename P> mln::p_set< P >::p_set ( ) [inline]`

Constructor.

Definition at line 152 of file p\_set.hh.

**10.290.4** **Member Function Documentation****10.290.4.1** `template<typename P> void mln::p_set< P >::clear ( ) [inline]`

Clear this set.

Definition at line 219 of file p\_set.hh.

**10.290.4.2** `template<typename P> bool mln::p_set< P >::has ( const psite & p ) const [inline]`

Test if psite *p* belongs to this point set.

Definition at line 167 of file p\_set.hh.

**10.290.4.3** `template<typename P> bool mln::p_set< P >::has ( const P & p ) const [inline]`

Test if *p* belongs to this point set.

Definition at line 159 of file p\_set.hh.

**10.290.4.4** `template<typename P> bool mln::p_set< P >::has ( const util::index & i ) const [inline]`

Test if index *i* belongs to this point set.

Definition at line 179 of file p\_set.hh.

References mln::p\_set< P >::nsites().

#### 10.290.4.5 `template<typename P> void mln::p_set< P >::insert ( const P & p ) [inline]`

Insert a site p.

Definition at line 203 of file p\_set.hh.

Referenced by mln::convert::to\_p\_set().

#### 10.290.4.6 `template<typename P> bool mln::p_set< P >::is_valid ( ) const [inline]`

Test this set validity so returns always true.

Definition at line 187 of file p\_set.hh.

#### 10.290.4.7 `template<typename P> std::size_t mln::p_set< P >::memory_size ( ) const [inline]`

Return the size of this site set in memory.

Definition at line 236 of file p\_set.hh.

#### 10.290.4.8 `template<typename P> unsigned mln::p_set< P >::nsites ( ) const [inline]`

Give the number of sites.

Definition at line 195 of file p\_set.hh.

Referenced by mln::p\_key< K, P >::change\_key(), mln::p\_set< P >::has(), mln::p\_set< P >::operator[](), and mln::p\_key< K, P >::remove\_key().

#### 10.290.4.9 `template<typename P> const P & mln::p_set< P >::operator[] ( unsigned i ) const [inline]`

Return the *i*-th site.

Definition at line 227 of file p\_set.hh.

References mln::p\_set< P >::nsites().

#### 10.290.4.10 `template<typename P> void mln::p_set< P >::remove ( const P & p ) [inline]`

Remove a site p.

Definition at line 211 of file p\_set.hh.

#### 10.290.4.11 `template<typename P> const std::vector< P > & mln::p_set< P >::std_vector ( ) const [inline]`

Return the corresponding std::vector of sites.

Definition at line 244 of file p\_set.hh.

#### 10.290.4.12 `template<typename P> const util::set< P > & mln::p_set< P >::util_set ( ) const` `[inline]`

Return the corresponding `util::set` of sites.

Definition at line 252 of file `p_set.hh`.

## 10.291 `mln::p_set_of< S >` Class Template Reference

`p_set_of` is a set of site sets.

```
#include <p_set_of.hh>
```

Inherits `site_set_base_< S::site, p_set_of< S > >`, and `site_set_impl< S >`.

### Public Types

- typedef `p_double_piter< self_, mln_bkd_eiter(set_), typename S::bkd_piter > bkd_piter`  
*Backward `Site_Iterator` associated type.*
- typedef `S element`  
*Element associated type.*
- typedef `p_double_piter< self_, mln_fwd_eiter(set_), typename S::fwd_piter > fwd_piter`  
*Forward `Site_Iterator` associated type.*
- typedef `S i_element`  
*Insertion element associated type.*
- typedef `fwd_piter piter`  
*`Site_Iterator` associated type.*
- typedef `p_double_psite< self_, element > psite`  
*Psite associated type.*

### Public Member Functions

- void `clear ()`  
*Clear this set.*
- bool `has (const psite &p) const`  
*Test if `p` belongs to this point set.*
- void `insert (const S &s)`  
*Insert a site set `s`.*
- bool `is_valid () const`  
*Test if this set of runs is valid.*

- `std::size_t memory_size () const`  
*Return the size of this site set in memory.*
- `unsigned nelements () const`  
*Give the number of elements (site sets) of this composite.*
- `const S & operator[] (unsigned i) const`  
*Return the *i*-th site set.*
- `p_set_of ()`  
*Constructor without arguments.*

### 10.291.1 Detailed Description

`template<typename S> class mln::p_set_of< S >`

`p_set_of` is a set of site sets. Parameter `S` is the type of the contained site sets.

Definition at line 71 of file `p_set_of.hh`.

### 10.291.2 Member Typedef Documentation

**10.291.2.1** `template<typename S > typedef p_double_piter<self_, mln_bkd_eiter(set_), typename S ::bkd_piter> mln::p_set_of< S >::bkd_piter`

Backward `Site_Iterator` associated type.

Definition at line 95 of file `p_set_of.hh`.

**10.291.2.2** `template<typename S > typedef S mln::p_set_of< S >::element`

Element associated type.

Definition at line 81 of file `p_set_of.hh`.

**10.291.2.3** `template<typename S > typedef p_double_piter<self_, mln_fwd_eiter(set_), typename S ::fwd_piter> mln::p_set_of< S >::fwd_piter`

Forward `Site_Iterator` associated type.

Definition at line 90 of file `p_set_of.hh`.

**10.291.2.4** `template<typename S > typedef S mln::p_set_of< S >::i_element`

Insertion element associated type.

Definition at line 113 of file `p_set_of.hh`.



**10.291.2.5** `template<typename S > typedef fwd_piter mln::p_set_of< S >::piter`

[Site\\_Iterator](#) associated type.

Definition at line 98 of file p\_set\_of.hh.

**10.291.2.6** `template<typename S > typedef p_double_psite<self_, element> mln::p_set_of< S >::psite`

Psite associated type.

Definition at line 85 of file p\_set\_of.hh.

**10.291.3** **Constructor & Destructor Documentation****10.291.3.1** `template<typename S > mln::p_set_of< S >::p_set_of ( ) [inline]`

Constructor without arguments.

Definition at line 160 of file p\_set\_of.hh.

**10.291.4** **Member Function Documentation****10.291.4.1** `template<typename S > void mln::p_set_of< S >::clear ( ) [inline]`

Clear this set.

Definition at line 211 of file p\_set\_of.hh.

**10.291.4.2** `template<typename S > bool mln::p_set_of< S >::has ( const psite & p ) const [inline]`

Test if *p* belongs to this point set.

Definition at line 167 of file p\_set\_of.hh.

**10.291.4.3** `template<typename S > void mln::p_set_of< S >::insert ( const S & s ) [inline]`

Insert a site set *s*.

Definition at line 183 of file p\_set\_of.hh.

**10.291.4.4** `template<typename S > bool mln::p_set_of< S >::is_valid ( ) const [inline]`

Test if this set of runs is valid.

Definition at line 175 of file p\_set\_of.hh.

**10.291.4.5** `template<typename S > std::size_t mln::p_set_of< S >::memory_size ( ) const [inline]`

Return the size of this site set in memory.

Definition at line 220 of file p\_set\_of.hh.

**10.291.4.6** `template<typename S > unsigned mln::p_set_of< S >::nelements ( ) const`  
**[inline]**

Give the number of elements (site sets) of this composite.

Definition at line 203 of file p\_set\_of.hh.

**10.291.4.7** `template<typename S > const S & mln::p_set_of< S >::operator[] ( unsigned i )`  
**const [inline]**

Return the *i*-th site set.

Definition at line 194 of file p\_set\_of.hh.

## 10.292 mln::p\_transformed< S, F > Class Template Reference

[Site](#) set transformed through a function.

```
#include <p_transformed.hh>
```

Inherits `site_set_base_< S::psite, p_transformed< S, F > >`.

### Public Types

- typedef `p_transformed_piter< typename S::bkd_piter, S, F > bkd_piter`  
*Backward [Site\\_Iterator](#) associated type.*
- typedef `S::element element`  
*Element associated type.*
- typedef `p_transformed_piter< typename S::fwd_piter, S, F > fwd_piter`  
*Forward [Site\\_Iterator](#) associated type.*
- typedef `fwd_piter piter`  
*[Site\\_Iterator](#) associated type.*
- typedef `S::psite psite`  
*Psite associated type.*

### Public Member Functions

- `const F & function () const`  
*Return the transformation function.*
- `bool has (const psite &p) const`  
*Test if *p* belongs to the subset.*

- bool [is\\_valid](#) () const  
*Test if this site set is valid.*
- std::size\_t [memory\\_size](#) () const  
*Return the size of this site set in memory.*
- [p\\_transformed](#) ()  
*Constructor without argument.*
- [p\\_transformed](#) (const S &s, const F &f)  
*Constructor with a site set s and a predicate f.*
- const S & [primary\\_set](#) () const  
*Return the primary set.*

### 10.292.1 Detailed Description

**template<typename S, typename F> class mln::p\_transformed< S, F >**

[Site](#) set transformed through a function. Parameter S is a site set type; parameter F is a function from site to site.

Definition at line 82 of file p\_transformed.hh.

### 10.292.2 Member Typedef Documentation

**10.292.2.1 template<typename S, typename F> typedef p\_transformed\_piter<typename S  
::bkd\_piter, S, F> mln::p\_transformed< S, F >::bkd\_piter**

Backward [Site\\_Iterator](#) associated type.

Definition at line 101 of file p\_transformed.hh.

**10.292.2.2 template<typename S, typename F> typedef S ::element mln::p\_transformed< S, F  
>::element**

Element associated type.

Definition at line 91 of file p\_transformed.hh.

**10.292.2.3 template<typename S, typename F> typedef p\_transformed\_piter<typename S  
::fwd\_piter, S, F> mln::p\_transformed< S, F >::fwd\_piter**

Forward [Site\\_Iterator](#) associated type.

Definition at line 98 of file p\_transformed.hh.

**10.292.2.4** `template<typename S, typename F> typedef fwd_piter mln::p_transformed< S, F >::piter`

[Site\\_Iterator](#) associated type.

Definition at line 104 of file p\_transformed.hh.

**10.292.2.5** `template<typename S, typename F> typedef S ::psite mln::p_transformed< S, F >::psite`

Psite associated type.

Definition at line 95 of file p\_transformed.hh.

### 10.292.3 Constructor & Destructor Documentation

**10.292.3.1** `template<typename S, typename F > mln::p_transformed< S, F >::p_transformed ( const S & s, const F & f ) [inline]`

Constructor with a site set *s* and a predicate *f*.

Definition at line 163 of file p\_transformed.hh.

**10.292.3.2** `template<typename S, typename F > mln::p_transformed< S, F >::p_transformed ( ) [inline]`

Constructor without argument.

Definition at line 157 of file p\_transformed.hh.

### 10.292.4 Member Function Documentation

**10.292.4.1** `template<typename S, typename F > const F & mln::p_transformed< S, F >::function ( ) const [inline]`

Return the transformation function.

Definition at line 206 of file p\_transformed.hh.

**10.292.4.2** `template<typename S, typename F > bool mln::p_transformed< S, F >::has ( const psite & p ) const [inline]`

Test if *p* belongs to the subset.

Definition at line 172 of file p\_transformed.hh.

**10.292.4.3** `template<typename S, typename F > bool mln::p_transformed< S, F >::is_valid ( ) const [inline]`

Test if this site set is valid.

Definition at line 182 of file p\_transformed.hh.

**10.292.4.4** `template<typename S , typename F > std::size_t mln::p_transformed< S, F >::memory_size ( ) const [inline]`

Return the size of this site set in memory.

Definition at line 190 of file p\_transformed.hh.

**10.292.4.5** `template<typename S , typename F > const S & mln::p_transformed< S, F >::primary_set ( ) const [inline]`

Return the primary set.

Definition at line 198 of file p\_transformed.hh.

Referenced by mln::p\_transformed\_piter< Pi, S, F >::change\_target().

## 10.293 mln::p\_transformed\_piter< Pi, S, F > Struct Template Reference

[Iterator](#) on p\_transformed<S,F>.

```
#include <p_transformed_piter.hh>
```

Inherits mln::internal::site\_set\_iterator\_base< p\_transformed< S, F >,p\_transformed\_piter< Pi, S, F >>.

### Public Member Functions

- void [change\\_target](#) (const [p\\_transformed](#)< S, F > &s)  
*Change the set site targeted by this iterator.*
- void [next](#) ()  
*Go to the next element.*
- [p\\_transformed\\_piter](#) (const [p\\_transformed](#)< S, F > &s)  
*Constructor from a site set.*
- [p\\_transformed\\_piter](#) ()  
*Constructor without argument.*

### 10.293.1 Detailed Description

`template<typename Pi, typename S, typename F> struct mln::p_transformed_piter< Pi, S, F >`

[Iterator](#) on p\_transformed<S,F>. Parameter S is a site set type; parameter F is a function from point to Boolean.

**See also**

[mln::p\\_transformed](#)

Definition at line 50 of file p\_transformed\_piter.hh.

## 10.293.2 Constructor & Destructor Documentation

**10.293.2.1** `template<typename Pi , typename S , typename F > mln::p_transformed_piter< Pi, S, F >::p_transformed_piter ( ) [inline]`

Constructor without argument.

Definition at line 93 of file p\_transformed\_piter.hh.

**10.293.2.2** `template<typename Pi , typename S , typename F > mln::p_transformed_piter< Pi, S, F >::p_transformed_piter ( const p_transformed< S, F > & s ) [inline]`

Constructor from a site set.

Definition at line 99 of file p\_transformed\_piter.hh.

References mln::p\_transformed\_piter< Pi, S, F >::change\_target().

## 10.293.3 Member Function Documentation

**10.293.3.1** `template<typename Pi , typename S , typename F > void mln::p_transformed_piter< Pi, S, F >::change_target ( const p_transformed< S, F > & s ) [inline]`

Change the set site targeted by this iterator.

Definition at line 143 of file p\_transformed\_piter.hh.

References mln::p\_transformed< S, F >::primary\_set().

Referenced by mln::p\_transformed\_piter< Pi, S, F >::p\_transformed\_piter().

**10.293.3.2** `template<typename E > void mln::Site_Iterator< E >::next ( ) [inline, inherited]`

Go to the next element.

### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

### Precondition

The iterator is valid.

Definition at line 92 of file site\_iterator.hh.

## 10.294 mln::p\_vaccess< V, S > Class Template Reference

[Site](#) set in which sites are grouped by their associated value.

```
#include <p_vaccess.hh>
```

Inherits site\_set\_base\_< S::site, p\_vaccess< V, S > >, and site\_set\_impl< S >.

## Public Types

- typedef p\_double\_piter< [self\\_](#), typename vset::bkd\_viter, typename S::bkd\_piter > [bkd\\_piter](#)  
*Backward [Site\\_Iterator](#) associated type.*
- typedef S::element [element](#)  
*Element associated type.*
- typedef p\_double\_piter< [self\\_](#), typename vset::fwd\_viter, typename S::fwd\_piter > [fwd\\_piter](#)  
*Forward [Site\\_Iterator](#) associated type.*
- typedef std::pair< V, [element](#) > [i\\_element](#)  
*Insertion element associated type.*
- typedef [fwd\\_piter](#) piter  
*[Site\\_Iterator](#) associated type.*
- typedef S [pset](#)  
*Inner site set associated type.*
- typedef p\_double\_psite< [self\\_](#), S > [psite](#)  
*Psite associated type.*
- typedef V [value](#)  
*Value associated type.*
- typedef [mln::value::set](#)< V > [vset](#)  
*[Value\\_Set](#) associated type.*

## Public Member Functions

- bool [has](#) (const [psite](#) &p) const  
*Test if *p* belongs to this site set.*
- bool [has](#) (const V &v, const typename S::psite &p) const  
*Test if the couple (value *v*, psite *p*) belongs to this site set.*
- void [insert](#) (const [i\\_element](#) &v\_e)  
*Insert a pair *v\_e* (value *v*, element *e*).*
- void [insert](#) (const V &v, const [element](#) &e)  
*Insert *e* at value *v*.*
- bool [is\\_valid](#) () const  
*Test if this site set is valid.*
- std::size\_t [memory\\_size](#) () const  
*Return the size of this site set in memory.*

- `const S & operator() (const V &v) const`  
*Return the site set at value v.*
- `p_vaccess ()`  
*Constructor.*
- `const mln::value::set< V > & values () const`  
*Give the set of values.*

### 10.294.1 Detailed Description

`template<typename V, typename S> class mln::p_vaccess< V, S >`

[Site](#) set in which sites are grouped by their associated value.

Definition at line 70 of file `p_vaccess.hh`.

### 10.294.2 Member Typedef Documentation

**10.294.2.1** `template<typename V , typename S > typedef p_double_piter<self_, typename vset  
::bkd_viter, typename S ::bkd_piter> mln::p_vaccess< V, S >::bkd_piter`

Backward [Site\\_Iterator](#) associated type.

Definition at line 94 of file `p_vaccess.hh`.

**10.294.2.2** `template<typename V , typename S > typedef S ::element mln::p_vaccess< V, S  
>::element`

Element associated type.

Definition at line 117 of file `p_vaccess.hh`.

**10.294.2.3** `template<typename V , typename S > typedef p_double_piter<self_, typename vset  
::fwd_viter, typename S ::fwd_piter> mln::p_vaccess< V, S >::fwd_piter`

Forward [Site\\_Iterator](#) associated type.

Definition at line 91 of file `p_vaccess.hh`.

**10.294.2.4** `template<typename V , typename S > typedef std::pair<V, element> mln::p_vaccess<  
V, S >::i_element`

Insertion element associated type.

Definition at line 120 of file `p_vaccess.hh`.

**10.294.2.5** `template<typename V , typename S > typedef fwd_piter mln::p_vaccess< V, S >::piter`

[Site\\_Iterator](#) associated type.



Definition at line 97 of file p\_vaccess.hh.

#### 10.294.2.6 `template<typename V , typename S > typedef S mln::p_vaccess< V, S >::pset`

Inner site set associated type.

Definition at line 85 of file p\_vaccess.hh.

#### 10.294.2.7 `template<typename V , typename S > typedef p_double_psite<self_, S> mln::p_vaccess< V, S >::psite`

Psite associated type.

Definition at line 88 of file p\_vaccess.hh.

#### 10.294.2.8 `template<typename V , typename S > typedef V mln::p_vaccess< V, S >::value`

[Value](#) associated type.

Definition at line 78 of file p\_vaccess.hh.

#### 10.294.2.9 `template<typename V , typename S > typedef mln::value::set<V> mln::p_vaccess< V, S >::vset`

[Value\\_Set](#) associated type.

Definition at line 81 of file p\_vaccess.hh.

### 10.294.3 Constructor & Destructor Documentation

#### 10.294.3.1 `template<typename V , typename S > mln::p_vaccess< V, S >::p_vaccess ( ) [inline]`

Constructor.

Definition at line 163 of file p\_vaccess.hh.

### 10.294.4 Member Function Documentation

#### 10.294.4.1 `template<typename V , typename S > bool mln::p_vaccess< V, S >::has ( const V & v, const typename S::psite & p ) const [inline]`

Test if the couple (value *v*, psite *p*) belongs to this site set.

Definition at line 189 of file p\_vaccess.hh.

#### 10.294.4.2 `template<typename V , typename S > bool mln::p_vaccess< V, S >::has ( const psite & p ) const [inline]`

Test if *p* belongs to this site set.

Definition at line 180 of file p\_vaccess.hh.

**10.294.4.3** `template<typename V , typename S > void mln::p_vaccess< V, S >::insert ( const i_element & v_e ) [inline]`

Insert a pair `v_e` (value `v`, element `e`).

Definition at line 216 of file `p_vaccess.hh`.

**10.294.4.4** `template<typename V , typename S > void mln::p_vaccess< V, S >::insert ( const V & v, const element & e ) [inline]`

Insert `e` at value `v`.

Definition at line 206 of file `p_vaccess.hh`.

**10.294.4.5** `template<typename V , typename S > bool mln::p_vaccess< V, S >::is_valid ( ) const [inline]`

Test if this site set is valid.

Definition at line 197 of file `p_vaccess.hh`.

**10.294.4.6** `template<typename V , typename S > std::size_t mln::p_vaccess< V, S >::memory_size ( ) const [inline]`

Return the size of this site set in memory.

Definition at line 242 of file `p_vaccess.hh`.

**10.294.4.7** `template<typename V , typename S > const S & mln::p_vaccess< V, S >::operator() ( const V & v ) const [inline]`

Return the site set at value `v`.

Definition at line 234 of file `p_vaccess.hh`.

**10.294.4.8** `template<typename V , typename S > const mln::value::set< V > & mln::p_vaccess< V, S >::values ( ) const [inline]`

Give the set of values.

Definition at line 254 of file `p_vaccess.hh`.

## 10.295 `mln::p_vertices< G, F >` Class Template Reference

[Site](#) set based mapping graph vertices to sites.

```
#include <p_vertices.hh>
```

Inherits `site_set_base_< F::result, p_vertices< G, F >>`.

## Public Types

- typedef F [fun\\_t](#)  
*Function associated type.*
- typedef [util::vertex](#)< G > [graph\\_element](#)  
*Type of graph element this site set focuses on.*
- typedef G [graph\\_t](#)  
*Graph associated type.*
- typedef [util::vertex](#)< G > [vertex](#)  
*Type of graph vertex.*
  
- typedef [super\\_::site](#) [element](#)  
*Associated types.*
- typedef [p\\_vertices\\_psite](#)< G, F > [psite](#)  
*Point\_Site associated type.*
- typedef [p\\_graph\\_piter](#)< [self\\_](#), [mln\\_vertex\\_fwd\\_iter](#)(G) > [fwd\\_piter](#)  
*Forward Site\_Iterator associated type.*
- typedef [p\\_graph\\_piter](#)< [self\\_](#), [mln\\_vertex\\_bkd\\_iter](#)(G) > [bkd\\_piter](#)  
*Backward Site\_Iterator associated type.*
- typedef [fwd\\_piter](#) [piter](#)  
*Site\_Iterator associated type.*

## Public Member Functions

- bool [has](#) (const [psite](#) &p) const  
*Does this site set has p?*
- template<typename G2 >  
bool [has](#) (const [util::vertex](#)< G2 > &v) const  
*Does this site set has v?*
- void [invalidate](#) ()  
*Invalidate this site set.*
- bool [is\\_valid](#) () const  
*Test this site set validity.*
- std::size\_t [memory\\_size](#) () const  
*Does this site set has vertex\_id? FIXME: causes ambiguities while calling has(mln::neighb\_fwd\_niter<>);  
bool has(unsigned vertex\_id) const;.*
- unsigned [nsites](#) () const

*Return The number of points (sites) of the set, i.e., the number of vertices.*

- unsigned `nvertices` () const  
*Return The number of vertices in the graph.*
- `p_vertices` (const `Graph`< G > &gr)  
*Construct a graph psite set from a graph of points.*
- `p_vertices` (const `Graph`< G > &gr, const `Function`< F > &f)  
*Construct a graph psite set from a graph of points.*
- template<typename F2 >  
`p_vertices` (const `p_vertices`< G, F2 > &other)  
*Copy constructor.*
- `p_vertices` ()  
*Constructor without argument.*
- template<typename F2 >  
`p_vertices` (const `Graph`< G > &gr, const `Function`< F2 > &f)  
*Construct a graph psite set from a graph of points.*
- `F::result operator()` (const `psite` &p) const  
*Return the value associated to an element of this site set.*
- const G & `graph` () const  
*Accessors.*
- const F & `function` () const  
*Return the association function.*

## 10.295.1 Detailed Description

```
template<typename G, typename F = util::internal::id2element<G,util::vertex<G> >> class
mln::p_vertices< G, F >
```

[Site](#) set based mapping graph vertices to sites.

Definition at line 71 of file `p_vertices.hh`.

## 10.295.2 Member Typedef Documentation

**10.295.2.1** `template<typename G, typename F = util::internal::id2element<G,util::vertex<G> >> typedef p_graph_piter< self_, mln_vertex_bkd_iter(G) > mln::p_vertices< G, F >::bkd_piter`

Backward [Site\\_Iterator](#) associated type.

Definition at line 132 of file `p_vertices.hh`.

**10.295.2.2** `template<typename G, typename F = util::internal::id2element<G,util::vertex<G>>> typedef super_::site mln::p_vertices< G, F >::element`

Associated types.

Element associated type.

Definition at line 123 of file p\_vertices.hh.

**10.295.2.3** `template<typename G, typename F = util::internal::id2element<G,util::vertex<G>>> typedef F mln::p_vertices< G, F >::fun_t`

[Function](#) associated type.

Definition at line 84 of file p\_vertices.hh.

**10.295.2.4** `template<typename G, typename F = util::internal::id2element<G,util::vertex<G>>> typedef p_graph_piter< self_, mln_vertex_fwd_iter(G) > mln::p_vertices< G, F >::fwd_piter`

Forward [Site\\_Iterator](#) associated type.

Definition at line 129 of file p\_vertices.hh.

**10.295.2.5** `template<typename G, typename F = util::internal::id2element<G,util::vertex<G>>> typedef util::vertex<G> mln::p_vertices< G, F >::graph_element`

Type of graph element this site set focuses on.

Definition at line 91 of file p\_vertices.hh.

**10.295.2.6** `template<typename G, typename F = util::internal::id2element<G,util::vertex<G>>> typedef G mln::p_vertices< G, F >::graph_t`

[Graph](#) associated type.

Definition at line 81 of file p\_vertices.hh.

**10.295.2.7** `template<typename G, typename F = util::internal::id2element<G,util::vertex<G>>> typedef fwd_piter mln::p_vertices< G, F >::piter`

[Site\\_Iterator](#) associated type.

Definition at line 135 of file p\_vertices.hh.

**10.295.2.8** `template<typename G, typename F = util::internal::id2element<G,util::vertex<G>>> typedef p_vertices_psite<G,F> mln::p_vertices< G, F >::psite`

[Point\\_Site](#) associated type.

Definition at line 126 of file p\_vertices.hh.

**10.295.2.9** `template<typename G, typename F = util::internal::id2element<G,util::vertex<G>>> typedef util::vertex<G> mln::p_vertices< G, F >::vertex`

Type of graph vertex.

Definition at line 87 of file p\_vertices.hh.

### 10.295.3 Constructor & Destructor Documentation

**10.295.3.1** `template<typename G , typename F > mln::p_vertices< G, F >::p_vertices ( ) [inline]`

Constructor without argument.

Definition at line 220 of file p\_vertices.hh.

**10.295.3.2** `template<typename G , typename F > mln::p_vertices< G, F >::p_vertices ( const Graph< G > & gr ) [inline]`

Construct a graph psite set from a graph of points.

#### Parameters

*gr* The graph upon which the graph psite set is built. The identity function is used.

Definition at line 226 of file p\_vertices.hh.

References `mln::p_vertices< G, F >::is_valid()`.

**10.295.3.3** `template<typename G , typename F > mln::p_vertices< G, F >::p_vertices ( const Graph< G > & gr, const Function< F > & f ) [inline]`

Construct a graph psite set from a graph of points.

#### Parameters

*gr* The graph upon which the graph psite set is built.

*f* the function which maps a vertex to a site.

Definition at line 238 of file p\_vertices.hh.

References `mln::p_vertices< G, F >::is_valid()`.

**10.295.3.4** `template<typename G , typename F > template<typename F2 > mln::p_vertices< G, F >::p_vertices ( const Graph< G > & gr, const Function< F2 > & f ) [inline]`

Construct a graph psite set from a graph of points.

#### Parameters

*gr* The graph upon which the graph psite set is built.

*f* the function which maps a vertex to a site. It must be convertible to the function type `F`.

Definition at line 248 of file p\_vertices.hh.

References `mln::p_vertices< G, F >::is_valid()`.

**10.295.3.5** `template<typename G , typename F > template<typename F2 > mln::p_vertices< G, F >::p_vertices ( const p_vertices< G, F2 > & other ) [inline]`

Copy constructor.

Definition at line 260 of file p\_vertices.hh.

References mln::p\_vertices< G, F >::function(), mln::p\_vertices< G, F >::graph(), and mln::p\_vertices< G, F >::is\_valid().

## 10.295.4 Member Function Documentation

**10.295.4.1** `template<typename G , typename F > const F & mln::p_vertices< G, F >::function ( ) const [inline]`

Return the association function.

Definition at line 385 of file p\_vertices.hh.

Referenced by mln::p\_vertices< G, F >::p\_vertices().

**10.295.4.2** `template<typename G , typename F > const G & mln::p_vertices< G, F >::graph ( ) const [inline]`

Accessors.

Return the graph associated to this site set (const version)

Definition at line 376 of file p\_vertices.hh.

References mln::p\_vertices< G, F >::is\_valid().

Referenced by mln::debug::draw\_graph(), mln::operator==( ), and mln::p\_vertices< G, F >::p\_vertices().

**10.295.4.3** `template<typename G , typename F > bool mln::p_vertices< G, F >::has ( const psite & p ) const [inline]`

Does this site set has *p*?

Definition at line 304 of file p\_vertices.hh.

References mln::p\_vertices< G, F >::is\_valid().

**10.295.4.4** `template<typename G , typename F > template<typename G2 > bool mln::p_vertices< G, F >::has ( const util::vertex< G2 > & v ) const [inline]`

Does this site set has *v*?

Definition at line 314 of file p\_vertices.hh.

References mln::util::vertex< G >::graph(), mln::util::vertex< G >::is\_valid(), and mln::p\_vertices< G, F >::is\_valid().

**10.295.4.5** `template<typename G , typename F > void mln::p_vertices< G, F >::invalidate ( )`  
`[inline]`

Invalidate this site set.

Definition at line 296 of file p\_vertices.hh.

**10.295.4.6** `template<typename G , typename F > bool mln::p_vertices< G, F >::is_valid ( )`  
`const [inline]`

Test this site set validity.

Definition at line 288 of file p\_vertices.hh.

Referenced by `mln::p_vertices< G, F >::graph()`, `mln::p_vertices< G, F >::has()`, and `mln::p_vertices< G, F >::p_vertices()`.

**10.295.4.7** `template<typename G , typename F > std::size_t mln::p_vertices< G, F`  
`>::memory_size ( ) const [inline]`

Does this site set has *vertex\_id*? FIXME: causes ambiguities while calling `has(mln::neighb_fwd_niter<>)`; `bool has(unsigned vertex_id) const;`.

Definition at line 339 of file p\_vertices.hh.

**10.295.4.8** `template<typename G , typename F > unsigned mln::p_vertices< G, F >::nsites ( )`  
`const [inline]`

Return The number of points (sites) of the set, i.e., the number of *vertices*.

Required by the `mln::Point_Set` concept.

Definition at line 272 of file p\_vertices.hh.

References `mln::p_vertices< G, F >::nvertices()`.

**10.295.4.9** `template<typename G , typename F > unsigned mln::p_vertices< G, F >::nvertices (`  
`) const [inline]`

Return The number of vertices in the graph.

Definition at line 280 of file p\_vertices.hh.

Referenced by `mln::p_vertices< G, F >::nsites()`.

**10.295.4.10** `template<typename G , typename F > F::result mln::p_vertices< G, F >::operator()`  
`( const psite & p ) const [inline]`

Return the value associated to an element of this site set.

Definition at line 349 of file p\_vertices.hh.

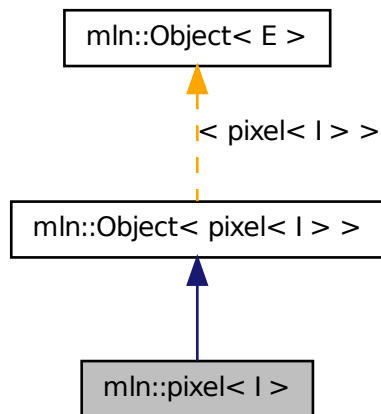


## 10.296 mln::pixel< I > Struct Template Reference

Generic pixel class.

```
#include <pixel.hh>
```

Inheritance diagram for mln::pixel< I >:



### Public Member Functions

- void `change_to` (const typename I::psite &p)  
*Change the pixel to the one at point p.*
- bool `is_valid` () const  
*Test if this pixel is valid.*
- `pixel` (I &image)  
*Constructor.*
- `pixel` (I &image, const typename I::psite &p)  
*Constructor.*

### 10.296.1 Detailed Description

```
template<typename I> struct mln::pixel< I >
```

Generic pixel class. The parameter is  $I$  the type of the image it belongs to.

Definition at line 50 of file core/pixel.hh.

## 10.296.2 Constructor & Destructor Documentation

### 10.296.2.1 `template<typename I> mln::pixel< I >::pixel ( I & image ) [inline]`

Constructor.

Definition at line 75 of file core/pixel.hh.

### 10.296.2.2 `template<typename I> mln::pixel< I >::pixel ( I & image, const typename I::psite & p ) [inline]`

Constructor.

Definition at line 82 of file core/pixel.hh.

References `mln::pixel< I >::change_to()`.

## 10.296.3 Member Function Documentation

### 10.296.3.1 `template<typename I> void mln::pixel< I >::change_to ( const typename I::psite & p ) [inline]`

Change the pixel to the one at point `p`.

Definition at line 92 of file core/pixel.hh.

Referenced by `mln::pixel< I >::pixel()`.

### 10.296.3.2 `template<typename I> bool mln::pixel< I >::is_valid ( ) const [inline]`

Test if this pixel is valid.

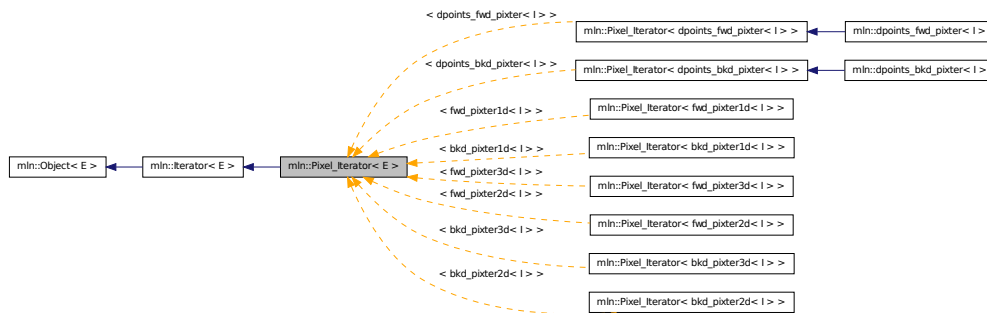
Definition at line 101 of file core/pixel.hh.

## 10.297 `mln::Pixel_Iterator< E >` Struct Template Reference

Base class for the implementation of pixel iterator classes.

```
#include <pixel_iterator.hh>
```

Inheritance diagram for mln::Pixel\_Iterator< E >:



## Public Member Functions

- void [next](#) ()  
*Go to the next element.*

### 10.297.1 Detailed Description

**template<typename E> struct mln::Pixel\_Iterator< E >**

Base class for the implementation of pixel iterator classes. An iterator on pixels is an iterator that is bound to a particular image and that browses over a set of image pixels.

See also

[mln::doc::Pixel\\_Iterator](#) for a complete documentation of this class contents.

Definition at line 51 of file pixel\_iterator.hh.

### 10.297.2 Member Function Documentation

**10.297.2.1 template<typename E > void mln::Iterator< E >::next ( ) [inherited]**

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

#### Precondition

The iterator is valid.

Definition at line 102 of file iterator.hh.

## 10.298 mln::plain< I > Class Template Reference

Prevents an image from sharing its data.

```
#include <plain.hh>
```

Inherits image\_identity< I, I::domain\_t, plain< I > >.

### Public Types

- typedef [plain](#)< tag::image\_< I > > [skeleton](#)  
*Skeleton.*

### Public Member Functions

- [operator I](#) () const  
*Conversion into an image with type I.*
- [plain](#)< I > & [operator=](#) (const I &ima)  
*Assignment operator from an image ima.*
- [plain](#)< I > & [operator=](#) (const [plain](#)< I > &rhs)  
*Assignment operator.*
- [plain](#) (const [plain](#)< I > &rhs)  
*Copy constructor.*
- [plain](#) ()  
*Constructor without argument.*
- [plain](#) (const I &ima)  
*Copy constructor from an image ima.*

#### 10.298.1 Detailed Description

```
template<typename I> class mln::plain< I >
```

Prevents an image from sharing its data. While assigned to another image, its data is duplicated.

Definition at line 82 of file plain.hh.

#### 10.298.2 Member Typedef Documentation

**10.298.2.1** `template<typename I> typedef plain< tag::image_<I> > mln::plain< I >::skeleton`

Skeleton.

Definition at line 93 of file plain.hh.

### 10.298.3 Constructor & Destructor Documentation

#### 10.298.3.1 `template<typename I > mln::plain< I >::plain ( ) [inline]`

Constructor without argument.

Definition at line 141 of file plain.hh.

#### 10.298.3.2 `template<typename I > mln::plain< I >::plain ( const plain< I > & rhs ) [inline]`

Copy constructor.

Definition at line 147 of file plain.hh.

#### 10.298.3.3 `template<typename I > mln::plain< I >::plain ( const I & ima ) [inline]`

Copy constructor from an image *ima*.

Definition at line 156 of file plain.hh.

### 10.298.4 Member Function Documentation

#### 10.298.4.1 `template<typename I > mln::plain< I >::operator I ( ) const [inline]`

Conversion into an image with type *I*.

Definition at line 197 of file plain.hh.

References `mln::duplicate()`.

#### 10.298.4.2 `template<typename I > plain< I > & mln::plain< I >::operator= ( const plain< I > & rhs ) [inline]`

Assignment operator.

Definition at line 174 of file plain.hh.

#### 10.298.4.3 `template<typename I > plain< I > & mln::plain< I >::operator= ( const I & ima ) [inline]`

Assignment operator from an image *ima*.

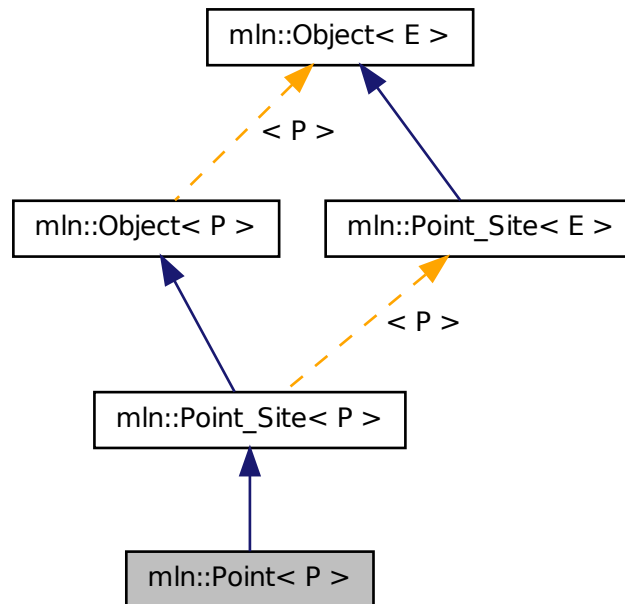
Definition at line 187 of file plain.hh.

## 10.299 mln::Point< P > Struct Template Reference

Base class for implementation of point classes.

```
#include <point.hh>
```

Inheritance diagram for `mln::Point< P >`:



## Public Types

- typedef `P` `point`

*The associated point type is itself.*

## Public Member Functions

- `const P & to_point () const`

*It is a `Point` so it returns itself.*

## Related Functions

(Note that these are not member functions.)

- `template<typename P , typename D >`  
`P & operator+= (Point< P > &p, const Dpoint< D > &dp)`

*Shift a point by a delta-point `dp`.*

- `template<typename P , typename D >`  
`P & operator-= (Point< P > &p, const Dpoint< D > &dp)`  
*Shift a point by the negate of a delta-point dp.*
- `template<typename P , typename D >`  
`P & operator/ (Point< P > &p, const value::Scalar< D > &dp)`  
*Divide a point by a scalar s.*

### 10.299.1 Detailed Description

`template<typename P> struct mln::Point< P >`

Base class for implementation of point classes. A point is an element of a space.

For instance, `mln::point2d` is the type of elements defined on the discrete square grid of the 2D plane.

Definition at line 62 of file `concept/point.hh`.

### 10.299.2 Member Typedef Documentation

**10.299.2.1** `template<typename P > typedef P mln::Point< P >::point`

The associated point type is itself.

Definition at line 66 of file `concept/point.hh`.

### 10.299.3 Member Function Documentation

**10.299.3.1** `template<typename P > const P & mln::Point< P >::to_point ( ) const [inline]`

It is a `Point` so it returns itself.

Definition at line 130 of file `concept/point.hh`.

### 10.299.4 Friends And Related Function Documentation

**10.299.4.1** `template<typename P , typename D > P & operator+= ( Point< P > & p, const Dpoint< D > & dp ) [related]`

Shift a point by a delta-point dp.

#### Parameters

[in, out] *p* The targeted point.

[in] *dp* A delta-point.

#### Returns

A reference to the point *p* once translated by *dp*.

**Precondition**

The type of `dp` has to be compatible with the type of `p`.

Definition at line 137 of file `concept/point.hh`.

#### 10.299.4.2 `template<typename P , typename D > P & operator=( Point< P > & p, const Dpoint< D > & dp ) [related]`

Shift a point by the negate of a delta-point `dp`.

**Parameters**

[in, out] `p` The targeted point.

[in] `dp` A delta-point.

**Returns**

A reference to the point `p` once translated by `- dp`.

**Precondition**

The type of `dp` has to be compatible with the type of `p`.

Definition at line 149 of file `concept/point.hh`.

#### 10.299.4.3 `template<typename P , typename D > P & operator/( Point< P > & p, const value::Scalar< D > & dp ) [related]`

Divide a point by a scalar `s`.

**Parameters**

[in, out] `p` The targeted point.

[in] `dp` A scalar.

**Returns**

A reference to the point `p` once divided by `s`.

Definition at line 163 of file `concept/point.hh`.

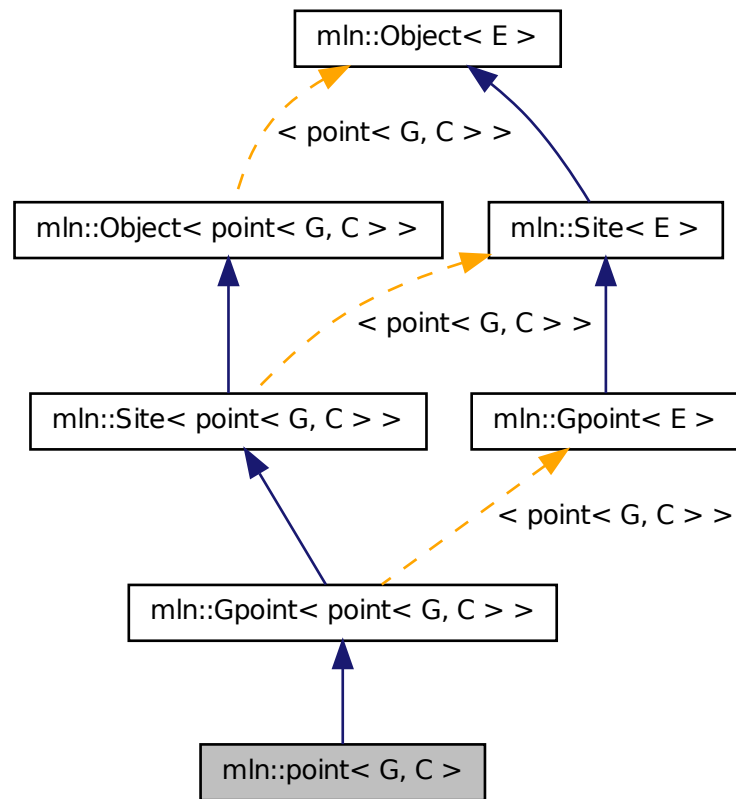
## 10.300 `mln::point< G, C >` Struct Template Reference

Generic point class.

```
#include <point.hh>
```



Inheritance diagram for mln::point< G, C >:



## Public Types

- enum { `dim = G::dim` }
- typedef `C coord`  
*Coordinate associated type.*
- typedef `dpoint< G, C > delta`  
*Delta associated type.*
- typedef `dpoint< G, C > dpsite`  
*DPsite associated type.*
- typedef `G grid`  
*Grid associated type.*
- typedef `mln::algebra::h_vec< G::dim, float > h_vec`

*Algebra hexagonal vector (hvec) associated type.*

- `typedef mln::algebra::vec< G::dim, float > vec`  
*Algebra vector (vec) associated type.*

## Public Member Functions

- `const C & last_coord () const`  
*Read-only access to the last coordinate.*
- `C & last_coord ()`  
*Read-write access to the last coordinate.*
- `point< G, C > & operator+= (const delta &dp)`  
*Shifting by dp.*
- `point< G, C > & operator-= (const delta &dp)`  
*Shifting by the inverse of dp.*
- `C & operator[] (unsigned i)`  
*Read-write access to the i-th coordinate value.*
- `const C & operator[] (unsigned i) const`  
*Read-only access to the i-th coordinate value.*
- `template<typename F >`  
`point (const Function_v2v< F > &f)`  
*Constructor; coordinates are set by function f.*
- `point ()`  
*Constructor without argument.*
- `template<typename C2 >`  
`point (const mln::algebra::vec< dim, C2 > &v)`  
*Constructor from an algebra vector.*
- `void set_all (C c)`  
*Set all coordinates to the value c.*
- `h_vec to_h_vec () const`  
*Transform to point in homogeneous coordinate system.*
- `vec to_vec () const`  
*Explicit conversion towards mln::algebra::vec.*
- `point (C ind)`
- `point (const literal::origin_t &)`  
*Constructors/assignments with literals.*

## Static Public Member Functions

- static const `point< G, C >` & `minus_infty` ()  
*Point with all coordinates set to the minimum value.*
- static const `point< G, C >` & `plus_infty` ()  
*Point with all coordinates set to the maximum value.*

## Static Public Attributes

- static const `point< G, C >` `origin` = `all_to(0)`  
*Origin point (all coordinates are 0).*

### 10.300.1 Detailed Description

`template<typename G, typename C> struct mln::point< G, C >`

Generic point class. Parameters are n the dimension of the space and C the coordinate type in this space.

Definition at line 108 of file point.hh.

### 10.300.2 Member Typedef Documentation

**10.300.2.1** `template<typename G, typename C> typedef C mln::point< G, C >::coord`

Coordinate associated type.

Definition at line 131 of file point.hh.

**10.300.2.2** `template<typename G, typename C> typedef dpoint<G,C> mln::point< G, C >::delta`

Delta associated type.

Definition at line 125 of file point.hh.

**10.300.2.3** `template<typename G, typename C> typedef dpoint<G,C> mln::point< G, C >::dpsite`

DPsite associated type.

Definition at line 128 of file point.hh.

**10.300.2.4** `template<typename G, typename C> typedef G mln::point< G, C >::grid`

Grid associated type.

Definition at line 122 of file point.hh.

**10.300.2.5** `template<typename G, typename C> typedef mln::algebra::h_vec<G::dim, float>  
mln::point< G, C >::h_vec`

Algebra hexagonal vector (hvec) associated type.

Definition at line 137 of file point.hh.

**10.300.2.6** `template<typename G, typename C> typedef mln::algebra::vec<G::dim, float>  
mln::point< G, C >::vec`

Algebra vector (vec) associated type.

Definition at line 134 of file point.hh.

### 10.300.3 Member Enumeration Documentation

**10.300.3.1** `template<typename G, typename C> anonymous enum`

**Enumerator:**

*dim* Dimension of the space.

**Invariant**

`dim > 0`

Definition at line 119 of file point.hh.

### 10.300.4 Constructor & Destructor Documentation

**10.300.4.1** `template<typename G , typename C > mln::point< G, C >::point ( ) [inline]`

Constructor without argument.

Definition at line 420 of file point.hh.

**10.300.4.2** `template<typename G , typename C > template<typename C2 > mln::point< G, C  
>::point ( const mln::algebra::vec< dim, C2 > & v ) [inline]`

Constructor from an algebra vector.

Definition at line 427 of file point.hh.

**10.300.4.3** `template<typename G , typename C> mln::point< G, C >::point ( C ind )  
[inline, explicit]`

Constructors with different numbers of arguments (coordinates) w.r.t. the dimension.

Definition at line 443 of file point.hh.

**10.300.4.4** `template<typename G , typename C> mln::point< G, C >::point ( const  
literal::origin_t & ) [inline]`

Constructors/assignments with literals.

Definition at line 481 of file point.hh.

**10.300.4.5** `template<typename G , typename C > template<typename F > mln::point< G, C >::point ( const Function_v2v< F > & f ) [inline]`

Constructor; coordinates are set by function *f*.

Definition at line 471 of file point.hh.

## 10.300.5 Member Function Documentation

**10.300.5.1** `template<typename G , typename C > const C & mln::point< G, C >::last_coord ( ) const [inline]`

Read-only access to the last coordinate.

Definition at line 402 of file point.hh.

Referenced by `mln::p_run< P >::end()`, `mln::p_run< P >::operator[]()`, and `mln::debug::put_word()`.

**10.300.5.2** `template<typename G , typename C > C & mln::point< G, C >::last_coord ( ) [inline]`

Read-write access to the last coordinate.

Definition at line 410 of file point.hh.

**10.300.5.3** `template<typename G , typename C > const point< G, C > & mln::point< G, C >::minus_infty ( ) [inline, static]`

[Point](#) with all coordinates set to the minimum value.

Definition at line 627 of file point.hh.

**10.300.5.4** `template<typename G , typename C > point< G, C > & mln::point< G, C >::operator+=( const delta & dp ) [inline]`

Shifting by *dp*.

Definition at line 544 of file point.hh.

**10.300.5.5** `template<typename G , typename C > point< G, C > & mln::point< G, C >::operator-=( const delta & dp ) [inline]`

Shifting by the inverse of *dp*.

Definition at line 554 of file point.hh.

**10.300.5.6** `template<typename G , typename C > C & mln::point< G, C >::operator[] ( unsigned i ) [inline]`

Read-write access to the *i*-th coordinate value.

**Parameters**

[in] *i* The coordinate index.

**Precondition**

$i < \text{dim}$

Definition at line 393 of file point.hh.

**10.300.5.7** `template<typename G , typename C > const C & mln::point< G, C >::operator[] ( unsigned i ) const [inline]`

Read-only access to the *i*-th coordinate value.

**Parameters**

[in] *i* The coordinate index.

**Precondition**

$i < \text{dim}$

Definition at line 385 of file point.hh.

**10.300.5.8** `template<typename G , typename C > const point< G, C > & mln::point< G, C >::plus_infty ( ) [inline, static]`

[Point](#) with all coordinates set to the maximum value.

Definition at line 618 of file point.hh.

**10.300.5.9** `template<typename G , typename C> void mln::point< G, C >::set_all ( C c ) [inline]`

Set all coordinates to the value *c*.

Definition at line 533 of file point.hh.

**10.300.5.10** `template<typename G , typename C > point< G, C >::h_vec mln::point< G, C >::to_h_vec ( ) const [inline]`

Transform to point in homogeneous coordinate system.

Definition at line 592 of file point.hh.

**10.300.5.11** `template<typename G , typename C > point< G, C >::vec mln::point< G, C >::to_vec ( ) const [inline]`

Explicit conversion towards `mln::algebra::vec`.

Definition at line 571 of file point.hh.

### 10.300.6 Member Data Documentation

**10.300.6.1** `template<typename G, typename C> const point< G, C > mln::point< G, C >::origin = all_to(0) [static]`

Origin point (all coordinates are 0).

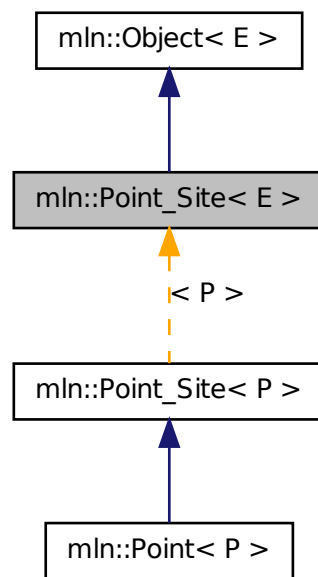
Definition at line 192 of file point.hh.

## 10.301 mln::Point\_Site< E > Struct Template Reference

Base class for implementation classes of the notion of "point site".

```
#include <point_site.hh>
```

Inheritance diagram for mln::Point\_Site< E >:



### Related Functions

(Note that these are not member functions.)

- `template<typename L, typename R >`  
`L::dpoint operator- (const Point_Site< L > &lhs, const Point_Site< R > &rhs)`

*Difference between a couple of point site lhs and rhs.*

- `template<typename P >`  
`std::ostream & operator<< (std::ostream &ostr, const Point_Site< P > &p)`  
*Print a point site `p` into the output stream `ostr`.*
- `template<typename L , typename R >`  
`bool operator== (const Point_Site< L > &lhs, const Point_Site< R > &rhs)`  
*Equality comparison between a couple of point site `lhs` and `rhs`.*
- `template<typename P , typename D >`  
`P::point operator+ (const Point_Site< P > &p, const Delta_Point_Site< D > &dp)`  
*Add a delta-point `rhs` to a point site `lhs`.*
- `template<typename P , typename D >`  
`P::point operator- (const Point_Site< P > &p, const Delta_Point_Site< D > &dp)`  
`}`

### 10.301.1 Detailed Description

`template<typename E> struct mln::Point_Site< E >`

Base class for implementation classes of the notion of "point site". A point site ("psite" for short) is an object that allows an efficient access to data associated with a point. A point site is either a point or designates a point.

When a point site is not really a point, it is automatically convertible to the point it designates.

Let us take the example of a 2D image encoded as an array of runs of values. With a point, a pair (row index, column index), retrieving the corresponding pixel value would mean to browse the array of runs to find the value location. That would not be efficient. Conversely, a point site dedicated to this image structure allows for value access in constant time; precisely the proper point site is a pair (index of run, index within the run).

Definition at line 106 of file `point_site.hh`.

### 10.301.2 Friends And Related Function Documentation

**10.301.2.1** `template<typename P , typename D > P::point operator+ ( const Point_Site< P > & p, const Delta_Point_Site< D > & dp )` [**related**]

Add a delta-point `rhs` to a point site `lhs`.

#### Parameters

- [in] `p` A point site.
- [in] `dp` A delta-point.

The type of `dp` has to be compatible with the type of `p`.

#### Returns

A point (temporary object).



**See also**

mln::Delta\_Point\_Site

Definition at line 282 of file point\_site.hh.

### 10.301.2.2 `template<typename P , typename D > P::point operator- ( const Point_Site< P > & p, const Delta_Point_Site< D > & dp ) [related]`

}

Substract a delta-point `dp` to a point site `p`.

**Parameters**

[in] *p* A point site.

[in] *dp* A delta-point.

The type of `dp` has to compatible with the type of `p`.

**Returns**

A point (temporary object).

**See also**

[mln::Dpoint](#)

mln::Delta\_Point\_Site

Definition at line 296 of file point\_site.hh.

### 10.301.2.3 `template<typename L , typename R > L::dpoint operator- ( const Point_Site< L > & lhs, const Point_Site< R > & rhs ) [related]`

Difference between a couple of point site `lhs` and `rhs`.

**Parameters**

[in] *lhs* A first point site.

[in] *rhs* A second point site.

**Warning**

There is no type promotion in Milena so the client has to make sure that both points are defined with the same type of coordinates.

**Precondition**

Both `lhs` and `rhs` have to be defined on the same topology and with the same type of coordinates; otherwise this test does not compile.

**Postcondition**

The result, `dp`, is such as `lhs == rhs + dp`.

**Returns**

A delta point (temporary object).

**See also**

`mln::Delta_Point_Site`

Definition at line 267 of file `point_site.hh`.

#### 10.301.2.4 `template<typename P > std::ostream & operator<< ( std::ostream & ostr, const Point_Site< P > & p )` **[related]**

Print a point site `p` into the output stream `ostr`.

**Parameters**

[in, out] *ostr* An output stream.

[in] *p* A point site.

**Returns**

The modified output stream `ostr`.

Definition at line 311 of file `point_site.hh`.

#### 10.301.2.5 `template<typename L , typename R > bool operator== ( const Point_Site< L > & lhs, const Point_Site< R > & rhs )` **[related]**

Equality comparison between a couple of point site `lhs` and `rhs`.

**Parameters**

[in] *lhs* A first point site.

[in] *rhs* A second point site.

**Precondition**

Both `lhs` and `rhs` have to be defined on the same topology; otherwise this test does not compile.

**Returns**

True if both point sites have the same coordinates, otherwise false.

Definition at line 251 of file `point_site.hh`.

## 10.302 `mln::Point_Site< void >` Struct Template Reference

[Point](#) site category flag type.

```
#include <point_site.hh>
```

### 10.302.1 Detailed Description

```
template<> struct mln::Point_Site< void >
```

[Point](#) site category flag type.

Definition at line 80 of file point\_site.hh.

## 10.303 mln::Proxy< E > Struct Template Reference

Base class for implementation classes of the notion of "proxy".

```
#include <proxy.hh>
```

Inherits [mln::Object< E >](#).

Inherited by [mln::Accumulator< E >](#), [mln::internal::graph\\_iter\\_base< G, Elt, E >](#), [mln::internal::nbh\\_iterator\\_base< G, C, Elt, E >](#), and [mln::Site\\_Proxy< E >](#).

### 10.303.1 Detailed Description

```
template<typename E> struct mln::Proxy< E >
```

Base class for implementation classes of the notion of "proxy".

Definition at line 232 of file core/concept/proxy.hh.

## 10.304 mln::Proxy< void > Struct Template Reference

[Proxy](#) category flag type.

```
#include <proxy.hh>
```

### 10.304.1 Detailed Description

```
template<> struct mln::Proxy< void >
```

[Proxy](#) category flag type.

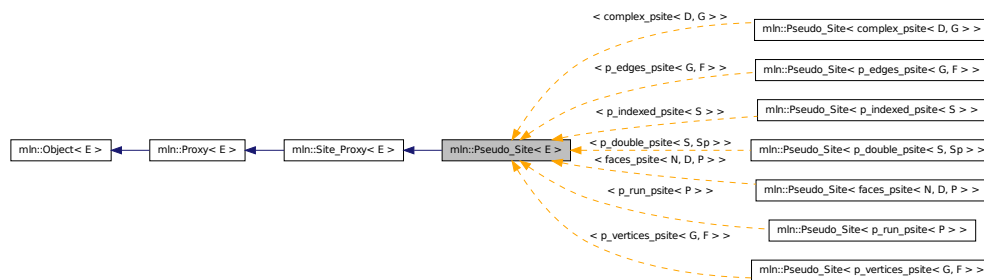
Definition at line 222 of file core/concept/proxy.hh.

## 10.305 mln::Pseudo\_Site< E > Struct Template Reference

Base class for implementation classes of the notion of "pseudo site".

```
#include <pseudo_site.hh>
```

Inheritance diagram for `mln::Pseudo_Site< E >`:



### 10.305.1 Detailed Description

```
template<typename E> struct mln::Pseudo_Site< E >
```

Base class for implementation classes of the notion of "pseudo site". FIXME: Explain...

Definition at line 64 of file `pseudo_site.hh`.

### 10.306 mln::Pseudo\_Site< void > Struct Template Reference

[Pseudo\\_Site](#) category flag type.

```
#include <pseudo_site.hh>
```

#### 10.306.1 Detailed Description

```
template<> struct mln::Pseudo_Site< void >
```

[Pseudo\\_Site](#) category flag type.

Definition at line 52 of file `pseudo_site.hh`.

### 10.307 mln::pw::image< F, S > Class Template Reference

A generic point-wise image implementation.

```
#include <image.hh>
```

Inherits `image_base< F, S, image< F, S >>`.

#### Public Types

- typedef `image< tag::function_< F >, tag::domain_< S >>` [skeleton](#)

*Skeleton.*

## Public Member Functions

- `image ()`  
*Constructor without argument.*
- `image (const Function_v2v< F > &f, const Site_Set< S > &ps)`  
*Constructor.*

### 10.307.1 Detailed Description

`template<typename F, typename S> class mln::pw::image< F, S >`

A generic point-wise image implementation. Parameter `F` is a function restricting the domain. Parameter `S` is the domain type.

Definition at line 92 of file `pw/image.hh`.

### 10.307.2 Member Typedef Documentation

**10.307.2.1** `template<typename F, typename S> typedef image< tag::function_<F>, tag::domain_<S> > mln::pw::image< F, S >::skeleton`

Skeleton.

Definition at line 99 of file `pw/image.hh`.

### 10.307.3 Constructor & Destructor Documentation

**10.307.3.1** `template<typename F , typename S > mln::pw::image< F, S >::image ( )`  
`[inline]`

Constructor without argument.

Definition at line 169 of file `pw/image.hh`.

**10.307.3.2** `template<typename F , typename S > mln::pw::image< F, S >::image ( const Function_v2v< F > & f, const Site_Set< S > & ps )`  
`[inline]`

Constructor.

Definition at line 175 of file `pw/image.hh`.

## 10.308 mln::registration::closest\_point\_basic< P > Class Template Reference

Closest point functor based on map distance.

```
#include <icp.hh>
```

### 10.308.1 Detailed Description

`template<typename P> class mln::registration::closest_point_basic< P >`

Closest point functor based on map distance.

Definition at line 240 of file icp.hh.

### 10.309 mln::registration::closest\_point\_with\_map< P > Class Template Reference

Closest point functor based on map distance.

```
#include <icp.hh>
```

#### 10.309.1 Detailed Description

`template<typename P> class mln::registration::closest_point_with_map< P >`

Closest point functor based on map distance.

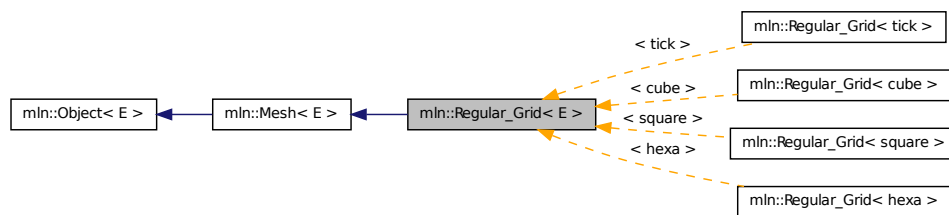
Definition at line 145 of file icp.hh.

### 10.310 mln::Regular\_Grid< E > Struct Template Reference

Base class for implementation classes of regular grids.

```
#include <regular_grid.hh>
```

Inheritance diagram for `mln::Regular_Grid< E >`:



#### 10.310.1 Detailed Description

`template<typename E> struct mln::Regular_Grid< E >`

Base class for implementation classes of regular grids.

Definition at line 42 of file regular\_grid.hh.

## 10.311 mln::safe\_image< I > Class Template Reference

Makes an image accessible at undefined location.

```
#include <safe.hh>
```

Inherits image\_identity< I, I::domain\_t, safe\_image< I > >.

### Public Types

- typedef [safe\\_image](#)< tag::image\_< I > > [skeleton](#)  
*Skeleton.*

### Public Member Functions

- [operator safe\\_image](#)< const I > () const  
*Const promotion via conversion.*

#### 10.311.1 Detailed Description

```
template<typename I> class mln::safe_image< I >
```

Makes an image accessible at undefined location.

Definition at line 83 of file safe.hh.

#### 10.311.2 Member Typedef Documentation

**10.311.2.1** `template<typename I> typedef safe_image< tag::image_<I> > mln::safe_image< I >::skeleton`

Skeleton.

Definition at line 88 of file safe.hh.

#### 10.311.3 Member Function Documentation

**10.311.3.1** `template<typename I> mln::safe_image< I >::operator safe_image< const I > ( ) const [inline]`

Const promotion via conversion.

Definition at line 195 of file safe.hh.

## 10.312 mln::select::p\_of< P > Struct Template Reference

Structure [p\\_of](#).

```
#include <pix.hh>
```

### 10.312.1 Detailed Description

`template<typename P> struct mln::select::p_of< P >`

Structure [p\\_of](#).

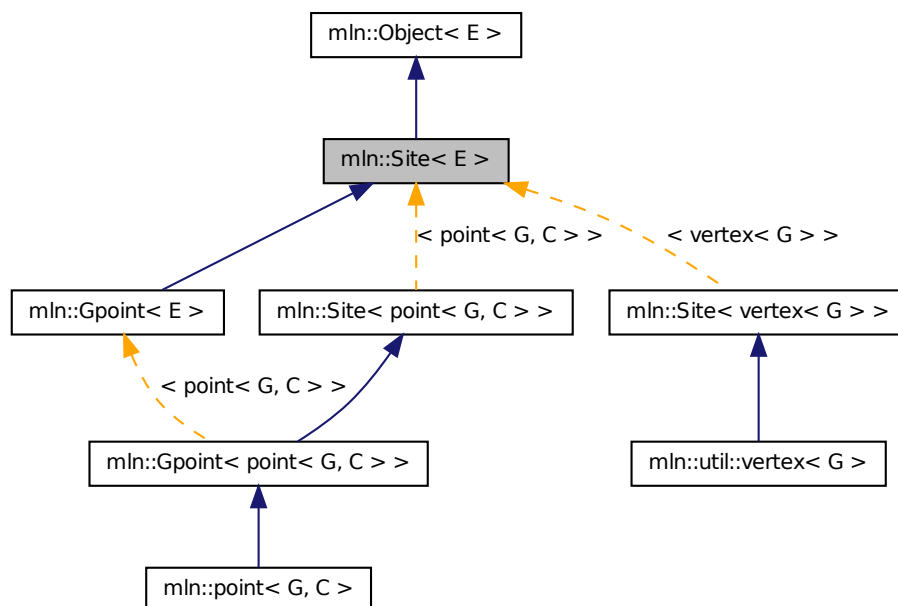
Definition at line 52 of file util/pix.hh.

### 10.313 mln::Site< E > Struct Template Reference

Base class for classes that are explicitly sites.

```
#include <site.hh>
```

Inheritance diagram for mln::Site< E >:



### 10.313.1 Detailed Description

`template<typename E> struct mln::Site< E >`

Base class for classes that are explicitly sites.

Definition at line 55 of file site.hh.



## 10.314 mln::Site< void > Struct Template Reference

[Site](#) category flag type.

```
#include <site.hh>
```

### 10.314.1 Detailed Description

**template<> struct mln::Site< void >**

[Site](#) category flag type.

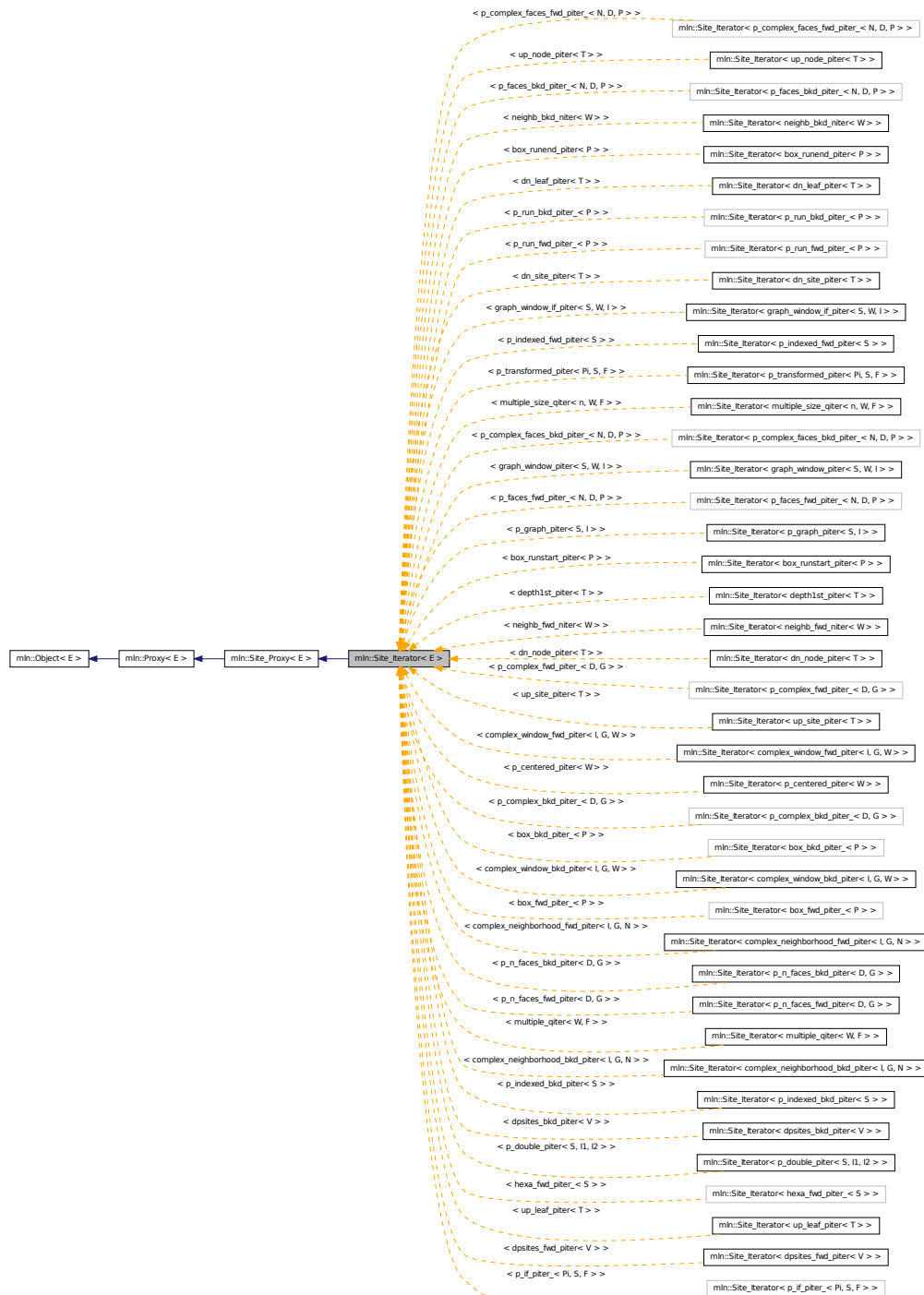
Definition at line 46 of file site.hh.

## 10.315 mln::Site\_Iterator< E > Struct Template Reference

Base class for implementation of classes of iterator on points.

```
#include <site_iterator.hh>
```

Inheritance diagram for mln::Site\_Iterator< E >:



## Public Member Functions

- void [next](#) ()  
*Go to the next element.*

### 10.315.1 Detailed Description

`template<typename E> struct mln::Site_Iterator< E >`

Base class for implementation of classes of iterator on points. An iterator on points is an iterator that browse over a set of points.

#### See also

[mln::doc::Site\\_Iterator](#) for a complete documentation of this class contents.

Definition at line 53 of file `site_iterator.hh`.

### 10.315.2 Member Function Documentation

**10.315.2.1** `template<typename E > void mln::Site_Iterator< E >::next ( ) [inline]`

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

#### Precondition

The iterator is valid.

Definition at line 92 of file `site_iterator.hh`.

## 10.316 mln::Site\_Proxy< E > Struct Template Reference

Base class for implementation classes of the notion of "site proxy".

```
#include <site_proxy.hh>
```

Inherits [mln::Proxy< E >](#).

Inherited by [mln::Pseudo\\_Site< E >](#), and [mln::Site\\_Iterator< E >](#).

### 10.316.1 Detailed Description

`template<typename E> struct mln::Site_Proxy< E >`

Base class for implementation classes of the notion of "site proxy". FIXME: Explain...

Definition at line 61 of file `site_proxy.hh`.

## 10.317 mln::Site\_Proxy< void > Struct Template Reference

[Site\\_Proxy](#) category flag type.

```
#include <site_proxy.hh>
```

### 10.317.1 Detailed Description

`template<> struct mln::Site_Proxy< void >`

[Site\\_Proxy](#) category flag type.

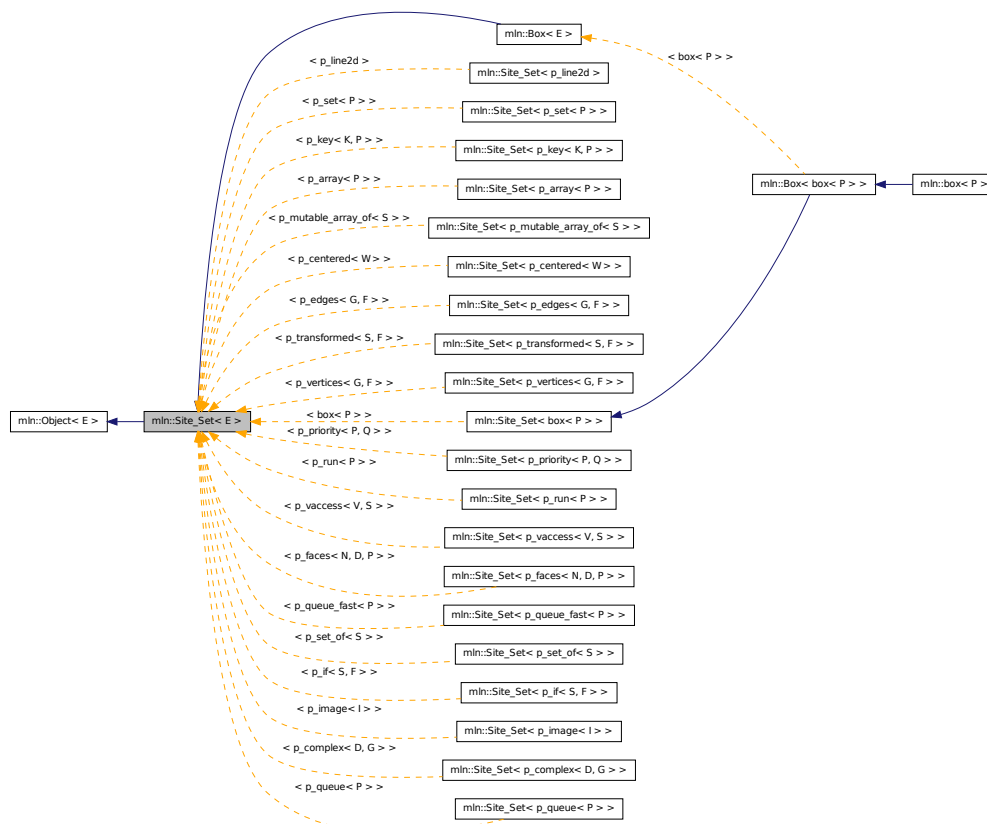
Definition at line 49 of file `site_proxy.hh`.

## 10.318 mln::Site\_Set< E > Struct Template Reference

Base class for implementation classes of site sets.

```
#include <site_set.hh>
```

Inheritance diagram for `mln::Site_Set< E >`:



## Related Functions

(Note that these are not member functions.)

- `template<typename SI , typename Sr >`  
`p_set< typename SI::site > diff (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`  
*Set theoretic difference of lhs and rhs.*
- `template<typename SI , typename Sr >`  
`p_set< typename SI::site > inter (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`  
*Intersection between a couple of point sets.*
- `template<typename SI , typename Sr >`  
`bool operator< (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`  
*Strict inclusion test between site sets lhs and rhs.*
- `template<typename S >`  
`std::ostream & operator<< (std::ostream &ostr, const Site_Set< S > &set)`  
*Print a site set set into the output stream ostr.*
- `template<typename SI , typename Sr >`  
`bool operator<= (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`  
*Inclusion test between site sets lhs and rhs.*
- `template<typename SI , typename Sr >`  
`bool operator== (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`  
*Equality test between site sets lhs and rhs.*
- `template<typename SI , typename Sr >`  
`p_set< typename SI::site > sym_diff (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`  
*Set theoretic symmetrical difference of lhs and rhs.*
- `template<typename SI , typename Sr >`  
`p_set< typename SI::site > uni (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`  
*Union of a couple of point sets.*
- `template<typename S >`  
`p_set< typename S::site > unique (const Site_Set< S > &s)`  
*Give the unique set of s.*

### 10.318.1 Detailed Description

`template<typename E> struct mln::Site_Set< E >`

Base class for implementation classes of site sets.

#### See also

[mln::doc::Site\\_Set](#) for a complete documentation of this class contents.

Definition at line 65 of file mln/core/concept/site\_set.hh.

## 10.318.2 Friends And Related Function Documentation

**10.318.2.1** `template<typename SI , typename Sr > p_set< typename SI::site > diff ( const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs ) [related]`

Set theoretic difference of `lhs` and `rhs`.

Definition at line 66 of file `set/diff.hh`.

**10.318.2.2** `template<typename SI , typename Sr > p_set< typename SI::site > inter ( const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs ) [related]`

Intersection between a couple of point sets.

Definition at line 62 of file `set/inter.hh`.

**10.318.2.3** `template<typename SI , typename Sr > bool operator< ( const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs ) [related]`

Strict inclusion test between site sets `lhs` and `rhs`.

### Parameters

[in] *lhs* A site set (strictly included?).

[in] *rhs* Another site set (includer?).

Definition at line 479 of file `operators.hh`.

**10.318.2.4** `template<typename S > std::ostream & operator<< ( std::ostream & ostr, const Site_Set< S > & set ) [related]`

Print a site set `set` into the output stream `ostr`.

### Parameters

[in, out] *ostr* An output stream.

[in] *set* A site set.

### Returns

The modified output stream `ostr`.

Definition at line 505 of file `operators.hh`.

**10.318.2.5** `template<typename SI , typename Sr > bool operator<= ( const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs ) [related]`

Inclusion test between site sets `lhs` and `rhs`.

### Parameters

[in] *lhs* A site set (included?).

[in] *rhs* Another site set (includer?).

Definition at line 491 of file `operators.hh`.

**10.318.2.6** `template<typename SI, typename Sr > bool operator==( const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs ) [related]`

Equality test between site sets `lhs` and `rhs`.

#### Parameters

[in] *lhs* A site set.

[in] *rhs* Another site set.

Definition at line 467 of file operators.hh.

**10.318.2.7** `template<typename SI, typename Sr > p_set< typename SI::site > sym_diff( const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs ) [related]`

Set theoretic symmetrical difference of `lhs` and `rhs`.

Definition at line 65 of file sym\_diff.hh.

**10.318.2.8** `template<typename SI, typename Sr > p_set< typename SI::site > uni( const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs ) [related]`

Union of a couple of point sets.

Definition at line 61 of file uni.hh.

**10.318.2.9** `template<typename S > p_set< typename S::site > unique( const Site_Set< S > & s ) [related]`

Give the unique set of `s`.

Definition at line 61 of file unique.hh.

## 10.319 mln::Site\_Set< void > Struct Template Reference

[Site\\_Set](#) category flag type.

```
#include <site_set.hh>
```

### 10.319.1 Detailed Description

`template<> struct mln::Site_Set< void >`

[Site\\_Set](#) category flag type.

Definition at line 54 of file mln/core/concept/site\_set.hh.

## 10.320 mln::slice\_image< I > Struct Template Reference

2D image extracted from a slice of a 3D image.

```
#include <slice_image.hh>
```

Inherits `image_domain_morpher< I, box2d, slice_image< I > >`.

## Public Types

- typedef `slice_image< tag::image_< I > > skeleton`  
*Skeleton.*

## Public Member Functions

- const `box2d & domain () const`  
*Give the definition domain.*
- `operator slice_image< const I > () const`  
*Const promotion via conversion.*
- `I::rvalue operator() (const point2d &p) const`  
*Read-only access to the image value located at point p.*
- `internal::morpher_lvalue_< I >::ret operator() (const point2d &p)`  
*Read-write access to the image value located at point p.*
- `def::coord sli () const`  
*Give the slice number.*
- `slice_image ()`  
*Constructor without argument.*
- `slice_image (I &ima, def::coord sli)`  
*Constructor from an image *ima* and a predicate *f*.*

### 10.320.1 Detailed Description

```
template<typename I> struct mln::slice_image< I >
```

2D image extracted from a slice of a 3D image.

Definition at line 95 of file `slice_image.hh`.

### 10.320.2 Member Typedef Documentation

**10.320.2.1** `template<typename I> typedef slice_image< tag::image_<I> > mln::slice_image< I >::skeleton`

Skeleton.

Definition at line 100 of file `slice_image.hh`.



### 10.320.3 Constructor & Destructor Documentation

#### 10.320.3.1 `template<typename I > mln::slice_image< I >::slice_image ( ) [inline]`

Constructor without argument.

Definition at line 193 of file slice\_image.hh.

#### 10.320.3.2 `template<typename I > mln::slice_image< I >::slice_image ( I & ima, def::coord sli ) [inline]`

Constructor from an image *ima* and a predicate *f*.

Definition at line 199 of file slice\_image.hh.

### 10.320.4 Member Function Documentation

#### 10.320.4.1 `template<typename I > const box2d & mln::slice_image< I >::domain ( ) const [inline]`

Give the definition domain.

Definition at line 216 of file slice\_image.hh.

#### 10.320.4.2 `template<typename I > mln::slice_image< I >::operator slice_image< const I > ( ) const [inline]`

Const promotion via conversion.

Definition at line 255 of file slice\_image.hh.

#### 10.320.4.3 `template<typename I > internal::morpher_lvalue_< I >::ret mln::slice_image< I >::operator() ( const point2d & p ) [inline]`

Read-write access to the image value located at point *p*.

Definition at line 245 of file slice\_image.hh.

#### 10.320.4.4 `template<typename I > I::rvalue mln::slice_image< I >::operator() ( const point2d & p ) const [inline]`

Read-only access to the image value located at point *p*.

Definition at line 234 of file slice\_image.hh.

#### 10.320.4.5 `template<typename I > def::coord mln::slice_image< I >::sli ( ) const [inline]`

Give the slice number.

Definition at line 225 of file slice\_image.hh.

## 10.321 mln::sub\_image< I, S > Class Template Reference

[Image](#) having its domain restricted by a site set.

```
#include <sub_image.hh>
```

Inherits [image\\_domain\\_morpher< I, S, sub\\_image< I, S > >](#).

### Public Types

- typedef [sub\\_image< tag::image\\_< I >, tag::domain\\_< S > >](#) [skeleton](#)

*Skeleton.*

### Public Member Functions

- const S & [domain](#) () const  
*Give the definition domain.*
- operator [sub\\_image< const I, S >](#) () const  
*Const promotion via conversion.*
- [sub\\_image](#) ()  
*Constructor without argument.*
- [sub\\_image](#) (const I &ima, const S &pset)  
*Constructor.*

#### 10.321.1 Detailed Description

```
template<typename I, typename S> class mln::sub_image< I, S >
```

[Image](#) having its domain restricted by a site set.

Definition at line 102 of file [sub\\_image.hh](#).

#### 10.321.2 Member Typedef Documentation

**10.321.2.1** `template<typename I, typename S> typedef sub_image< tag::image_<I>, tag::domain_<S> > mln::sub_image< I, S >::skeleton`

Skeleton.

Definition at line 108 of file [sub\\_image.hh](#).

### 10.321.3 Constructor & Destructor Documentation

**10.321.3.1** `template<typename I, typename S > mln::sub_image< I, S >::sub_image ( )`  
`[inline]`

Constructor without argument.

Definition at line 182 of file sub\_image.hh.

**10.321.3.2** `template<typename I, typename S > mln::sub_image< I, S >::sub_image ( const I & ima, const S & pset )`  
`[inline]`

Constructor.

Definition at line 188 of file sub\_image.hh.

### 10.321.4 Member Function Documentation

**10.321.4.1** `template<typename I, typename S > const S & mln::sub_image< I, S >::domain ( )`  
`const [inline]`

Give the definition domain.

Definition at line 205 of file sub\_image.hh.

**10.321.4.2** `template<typename I, typename S > mln::sub_image< I, S >::operator sub_image< const I, S > ( ) const`  
`[inline]`

Const promotion via conversion.

Definition at line 212 of file sub\_image.hh.

## 10.322 mln::sub\_image\_if< I, S > Struct Template Reference

[Image](#) having its domain restricted by a site set and a function.

```
#include <sub_image_if.hh>
```

Inherits `image_domain_morpher< I, p_if< S, fun::p2b::has< I > >, sub_image_if< I, S > >`.

### Public Types

- typedef `sub_image_if< tag::image_< I >, tag::domain_< S > >` [skeleton](#)  
*Skeleton.*

### Public Member Functions

- const `p_if< S, fun::p2b::has< I > > & domain ( ) const`  
*Give the definition domain.*

- `sub_image_if()`  
*Constructor without argument.*
- `sub_image_if(I &ima, const S &s)`  
*Constructor.*

### 10.322.1 Detailed Description

`template<typename I, typename S> struct mln::sub_image_if< I, S >`

[Image](#) having its domain restricted by a site set and a function.

Definition at line 101 of file `sub_image_if.hh`.

### 10.322.2 Member Typedef Documentation

**10.322.2.1** `template<typename I, typename S> typedef sub_image_if< tag::image_<I>, tag::domain_<S> > mln::sub_image_if< I, S >::skeleton`

Skeleton.

Definition at line 106 of file `sub_image_if.hh`.

### 10.322.3 Constructor & Destructor Documentation

**10.322.3.1** `template<typename I, typename S > mln::sub_image_if< I, S >::sub_image_if ( )`  
`[inline]`

Constructor without argument.

Definition at line 181 of file `sub_image_if.hh`.

**10.322.3.2** `template<typename I, typename S > mln::sub_image_if< I, S >::sub_image_if ( I & ima, const S & s )` `[inline]`

Constructor.

Definition at line 187 of file `sub_image_if.hh`.

### 10.322.4 Member Function Documentation

**10.322.4.1** `template<typename I, typename S > const p_if< S, fun::p2b::has< I > > & mln::sub_image_if< I, S >::domain ( ) const` `[inline]`

Give the definition domain.

Definition at line 204 of file `sub_image_if.hh`.

## 10.323 mln::thru\_image< I, F > Class Template Reference

Morph image values through a function.

```
#include <thru_image.hh>
```

### Public Member Functions

- [operator thru\\_image< const I, F > \(\) const](#)  
*Const promotion via conversion.*

#### 10.323.1 Detailed Description

```
template<typename I, typename F> class mln::thru_image< I, F >
```

Morph image values through a function.

Definition at line 156 of file thru\_image.hh.

#### 10.323.2 Member Function Documentation

**10.323.2.1** `template<typename I, typename F > mln::thru_image< I, F >::operator thru_image< const I, F > ( ) const` [[inline](#)]

Const promotion via conversion.

Definition at line 239 of file thru\_image.hh.

## 10.324 mln::thrubin\_image< I1, I2, F > Class Template Reference

Morphes values from two images through a binary function.

```
#include <thrubin_image.hh>
```

Inherits `image_value_morpher< I1, F::result, thrubin_image< I1, I2, F > >`.

### Public Types

- typedef `I1::psite` [psite](#)  
*Point\_Site associated type.*
- typedef `value` [rvalue](#)  
*Return type of read-only access.*
- typedef `thrubin_image< tag::image_< I1 >, tag::image_< I2 >, F >` [skeleton](#)  
*Skeleton.*
- typedef `F::result` [value](#)  
*Value associated type.*

## Public Member Functions

- [operator thrubin\\_image< const I1, const I2, F > \(\) const](#)  
*Const promotion via conversion.*

### 10.324.1 Detailed Description

**template<typename I1, typename I2, typename F> class mln::thrubin\_image< I1, I2, F >**

Morphes values from two images through a binary function.

Definition at line 82 of file thrubin\_image.hh.

### 10.324.2 Member Typedef Documentation

**10.324.2.1 template<typename I1, typename I2, typename F> typedef I1 ::psite  
mln::thrubin\_image< I1, I2, F >::psite**

[Point\\_Site](#) associated type.

Definition at line 94 of file thrubin\_image.hh.

**10.324.2.2 template<typename I1, typename I2, typename F> typedef value  
mln::thrubin\_image< I1, I2, F >::rvalue**

Return type of read-only access.

Definition at line 100 of file thrubin\_image.hh.

**10.324.2.3 template<typename I1, typename I2, typename F> typedef thrubin\_  
image<tag::image\_<I1>, tag::image\_<I2>, F> mln::thrubin\_image< I1, I2, F  
>::skeleton**

Skeleton.

Definition at line 91 of file thrubin\_image.hh.

**10.324.2.4 template<typename I1, typename I2, typename F> typedef F ::result  
mln::thrubin\_image< I1, I2, F >::value**

[Value](#) associated type.

Definition at line 97 of file thrubin\_image.hh.

### 10.324.3 Member Function Documentation

**10.324.3.1 template<typename I1 , typename I2 , typename F > mln::thrubin\_image< I1, I2, F  
>::operator thrubin\_image< const I1, const I2, F > ( ) const [inline]**

Const promotion via conversion.

## 10.325 mln::topo::adj\_higher\_dim\_connected\_n\_face\_bkd\_iter< D > Class Template Reference977

Definition at line 186 of file thrubin\_image.hh.

### 10.325 mln::topo::adj\_higher\_dim\_connected\_n\_face\_bkd\_iter< D > > Class Template Reference

Backward iterator on all the n-faces sharing an adjacent (n+1)-face with a (reference) n-face of an mln::complex<D>.

```
#include <adj_higher_dim_connected_n_face_iter.hh>
```

Inherits backward\_complex\_relative\_iterator\_base< topo::face< D >, algebraic\_face< D >, adj\_higher\_dim\_connected\_n\_face\_bkd\_iter< D > >, and mln::topo::internal::adj\_higher\_dim\_connected\_n\_face\_iterator< D >.

#### Public Member Functions

- void [next](#) ()  
*Go to the next element.*
- [adj\\_higher\\_dim\\_connected\\_n\\_face\\_bkd\\_iter](#) ()  
*Construction.*

#### 10.325.1 Detailed Description

```
template<unsigned D> class mln::topo::adj_higher_dim_connected_n_face_bkd_iter< D >
```

Backward iterator on all the n-faces sharing an adjacent (n+1)-face with a (reference) n-face of an mln::complex<D>.

#### Template Parameters

*D* The dimension of the complex this iterator belongs to.

Definition at line 104 of file adj\_higher\_dim\_connected\_n\_face\_iter.hh.

#### 10.325.2 Constructor & Destructor Documentation

**10.325.2.1** template<unsigned D> mln::topo::adj\_higher\_dim\_connected\_n\_face\_bkd\_iter< D >::adj\_higher\_dim\_connected\_n\_face\_bkd\_iter ( ) [inline]

Construction.

Definition at line 196 of file adj\_higher\_dim\_connected\_n\_face\_iter.hh.

### 10.325.3 Member Function Documentation

#### 10.325.3.1 void mln::Iterator< adj\_higher\_dim\_connected\_n\_face\_bkd\_iter< D > >::next ( ) [inherited]

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

#### Precondition

The iterator is valid.

## 10.326 mln::topo::adj\_higher\_dim\_connected\_n\_face\_fwd\_iter< D > > Class Template Reference

Forward iterator on all the n-faces sharing an adjacent (n+1)-face with a (reference) n-face of an mln::complex<D>.

```
#include <adj_higher_dim_connected_n_face_iter.hh>
```

Inherits forward\_complex\_relative\_iterator\_base< topo::face< D >, algebraic\_face< D >, adj\_higher\_dim\_connected\_n\_face\_fwd\_iter< D > >, and mln::topo::internal::adj\_higher\_dim\_connected\_n\_face\_iterator< D >.

### Public Member Functions

- void [next](#) ()  
*Go to the next element.*
- [adj\\_higher\\_dim\\_connected\\_n\\_face\\_fwd\\_iter](#) ()  
*Construction.*

### 10.326.1 Detailed Description

```
template<unsigned D> class mln::topo::adj_higher_dim_connected_n_face_fwd_iter< D >
```

Forward iterator on all the n-faces sharing an adjacent (n+1)-face with a (reference) n-face of an mln::complex<D>.

#### Template Parameters

*D* The dimension of the complex this iterator belongs to.

Definition at line 65 of file adj\_higher\_dim\_connected\_n\_face\_iter.hh.



## 10.326.2 Constructor & Destructor Documentation

**10.326.2.1** `template<unsigned D> mln::topo::adj_higher_dim_connected_n_face_fwd_iter< D >::adj_higher_dim_connected_n_face_fwd_iter ( ) [inline]`

Construction.

Definition at line 162 of file adj\_higher\_dim\_connected\_n\_face\_iter.hh.

## 10.326.3 Member Function Documentation

**10.326.3.1** `void mln::Iterator< adj_higher_dim_connected_n_face_fwd_iter< D > >::next ( ) [inherited]`

Go to the next element.

### Warning

This is a final method; iterator classes should not re-define this method. The actual "next" operation has to be defined through the *next\_* method.

### Precondition

The iterator is valid.

## 10.327 mln::topo::adj\_higher\_face\_bkd\_iter< D > Class Template Reference

Backward iterator on all the adjacent (n+1)-faces of the n-face of an mln::complex<D>.

```
#include <adj_higher_face_iter.hh>
```

Inherits backward\_complex\_relative\_iterator\_base< topo::face< D >, algebraic\_face< D >, adj\_higher\_face\_bkd\_iter< D > >.

## Public Member Functions

- void [next](#) ()

*Go to the next element.*

- [adj\\_higher\\_face\\_bkd\\_iter](#) ()

*Construction.*

### 10.327.1 Detailed Description

```
template<unsigned D> class mln::topo::adj_higher_face_bkd_iter< D >
```

Backward iterator on all the adjacent (n+1)-faces of the n-face of an mln::complex<D>.

### Template Parameters

*D* The dimension of the complex this iterator belongs to.

Definition at line 106 of file `adj_higher_face_iter.hh`.

## 10.327.2 Constructor & Destructor Documentation

**10.327.2.1** `template<unsigned D> mln::topo::adj_higher_face_bkd_iter< D >::adj_higher_face_bkd_iter ( ) [inline]`

Construction.

Definition at line 167 of file `adj_higher_face_iter.hh`.

## 10.327.3 Member Function Documentation

**10.327.3.1** `void mln::Iterator< adj_higher_face_bkd_iter< D > >::next ( ) [inherited]`

Go to the next element.

### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

### Precondition

The iterator is valid.

## 10.328 mln::topo::adj\_higher\_face\_fwd\_iter< D > Class Template Reference

Forward iterator on all the adjacent (n+1)-faces of the n-face of an `mln::complex<D>`.

```
#include <adj_higher_face_iter.hh>
```

Inherits `forward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_face_fwd_iter< D >>`.

### Public Member Functions

- void [next](#) ()

*Go to the next element.*

- [adj\\_higher\\_face\\_fwd\\_iter](#) ()

*Construction.*

### 10.328.1 Detailed Description

**template<unsigned D> class mln::topo::adj\_higher\_face\_fwd\_iter< D >**

Forward iterator on all the adjacent (n+1)-faces of the n-face of an mln::complex<D>.

#### Template Parameters

*D* The dimension of the complex this iterator belongs to.

Definition at line 72 of file adj\_higher\_face\_iter.hh.

### 10.328.2 Constructor & Destructor Documentation

**10.328.2.1 template<unsigned D> mln::topo::adj\_higher\_face\_fwd\_iter< D >::adj\_higher\_face\_fwd\_iter( ) [inline]**

Construction.

Definition at line 139 of file adj\_higher\_face\_iter.hh.

### 10.328.3 Member Function Documentation

**10.328.3.1 void mln::Iterator< adj\_higher\_face\_fwd\_iter< D > >::next( ) [inherited]**

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

#### Precondition

The iterator is valid.

## 10.329 mln::topo::adj\_lower\_dim\_connected\_n\_face\_bkd\_iter< D > Class Template Reference

Backward iterator on all the n-faces sharing an adjacent (n-1)-face with a (reference) n-face of an mln::complex<D>.

```
#include <adj_lower_dim_connected_n_face_iter.hh>
```

Inherits backward\_complex\_relative\_iterator\_base< topo::face< D >, algebraic\_face< D >, adj\_lower\_dim\_connected\_n\_face\_bkd\_iter< D > >, and mln::topo::internal::adj\_lower\_dim\_connected\_n\_face\_iterator< D >.

### Public Member Functions

- void [next](#) ()

Go to the next element.

- [adj\\_lower\\_dim\\_connected\\_n\\_face\\_bkd\\_iter\(\)](#)  
Construction.

### 10.329.1 Detailed Description

```
template<unsigned D> class mln::topo::adj_lower_dim_connected_n_face_bkd_iter< D >
```

Backward iterator on all the n-faces sharing an adjacent (n-1)-face with a (reference) n-face of an mln::complex<D>.

#### Template Parameters

*D* The dimension of the complex this iterator belongs to.

Definition at line 104 of file adj\_lower\_dim\_connected\_n\_face\_iter.hh.

### 10.329.2 Constructor & Destructor Documentation

**10.329.2.1** `template<unsigned D> mln::topo::adj_lower_dim_connected_n_face_bkd_iter< D >::adj_lower_dim_connected_n_face_bkd_iter( ) [inline]`

Construction.

Definition at line 196 of file adj\_lower\_dim\_connected\_n\_face\_iter.hh.

### 10.329.3 Member Function Documentation

**10.329.3.1** `void mln::Iterator< adj_lower_dim_connected_n_face_bkd_iter< D > >::next( ) [inherited]`

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

#### Precondition

The iterator is valid.

## 10.330 mln::topo::adj\_lower\_dim\_connected\_n\_face\_fwd\_iter< D > > Class Template Reference

Forward iterator on all the n-faces sharing an adjacent (n-1)-face with a (reference) n-face of an mln::complex<D>.

## 10.330 mln::topo::adj\_lower\_dim\_connected\_n\_face\_fwd\_iter< D > Class Template Reference 983

```
#include <adj_lower_dim_connected_n_face_iter.hh>
```

Inherits forward\_complex\_relative\_iterator\_base< topo::face< D >, algebraic\_face< D >, adj\_lower\_dim\_connected\_n\_face\_fwd\_iter< D > >, and mln::topo::internal::adj\_lower\_dim\_connected\_n\_face\_iterator< D >.

### Public Member Functions

- void [next](#) ()  
*Go to the next element.*
- [adj\\_lower\\_dim\\_connected\\_n\\_face\\_fwd\\_iter](#) ()  
*Construction.*

### 10.330.1 Detailed Description

```
template<unsigned D> class mln::topo::adj_lower_dim_connected_n_face_fwd_iter< D >
```

Forward iterator on all the n-faces sharing an adjacent (n-1)-face with a (reference) n-face of an mln::complex<D>.

#### Template Parameters

*D* The dimension of the complex this iterator belongs to.

Definition at line 65 of file adj\_lower\_dim\_connected\_n\_face\_iter.hh.

### 10.330.2 Constructor & Destructor Documentation

**10.330.2.1** `template<unsigned D> mln::topo::adj_lower_dim_connected_n_face_fwd_iter< D >::adj_lower_dim_connected_n_face_fwd_iter ( ) [inline]`

Construction.

Definition at line 162 of file adj\_lower\_dim\_connected\_n\_face\_iter.hh.

### 10.330.3 Member Function Documentation

**10.330.3.1** `void mln::Iterator< adj_lower_dim_connected_n_face_fwd_iter< D > >::next ( ) [inherited]`

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

#### Precondition

The iterator is valid.

## 10.331 mln::topo::adj\_lower\_face\_bkd\_iter< D > Class Template Reference

Backward iterator on all the adjacent (n-1)-faces of the n-face of an mln::complex<D>.

```
#include <adj_lower_face_iter.hh>
```

Inherits backward\_complex\_relative\_iterator\_base< topo::face< D >, algebraic\_face< D >, adj\_lower\_face\_bkd\_iter< D > >.

### Public Member Functions

- void [next](#) ()  
*Go to the next element.*
- [adj\\_lower\\_face\\_bkd\\_iter](#) ()  
*Construction.*

#### 10.331.1 Detailed Description

```
template<unsigned D> class mln::topo::adj_lower_face_bkd_iter< D >
```

Backward iterator on all the adjacent (n-1)-faces of the n-face of an mln::complex<D>.

#### Template Parameters

- D* The dimension of the complex this iterator belongs to.

Definition at line 108 of file adj\_lower\_face\_iter.hh.

#### 10.331.2 Constructor & Destructor Documentation

**10.331.2.1** `template<unsigned D> mln::topo::adj_lower_face_bkd_iter< D >::adj_lower_face_bkd_iter ( ) [inline]`

Construction.

Definition at line 169 of file adj\_lower\_face\_iter.hh.

#### 10.331.3 Member Function Documentation

**10.331.3.1** `void mln::Iterator< adj_lower_face_bkd_iter< D > >::next ( ) [inherited]`

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

**Precondition**

The iterator is valid.

## 10.332 mln::topo::adj\_lower\_face\_fwd\_iter< D > Class Template Reference

Forward iterator on all the adjacent (n-1)-faces of the n-face of an mln::complex<D>.

```
#include <adj_lower_face_iter.hh>
```

Inherits forward\_complex\_relative\_iterator\_base< topo::face< D >, algebraic\_face< D >, adj\_lower\_face\_fwd\_iter< D >>.

**Public Member Functions**

- void [next](#) ()  
*Go to the next element.*
  
- [adj\\_lower\\_face\\_fwd\\_iter](#) ()  
*Construction.*

### 10.332.1 Detailed Description

```
template<unsigned D> class mln::topo::adj_lower_face_fwd_iter< D >
```

Forward iterator on all the adjacent (n-1)-faces of the n-face of an mln::complex<D>.

**Template Parameters**

*D* The dimension of the complex this iterator belongs to.

Definition at line 73 of file adj\_lower\_face\_iter.hh.

### 10.332.2 Constructor & Destructor Documentation

**10.332.2.1** `template<unsigned D> mln::topo::adj_lower_face_fwd_iter< D >::adj_lower_face_fwd_iter ( ) [inline]`

Construction.

Definition at line 141 of file adj\_lower\_face\_iter.hh.

### 10.332.3 Member Function Documentation

**10.332.3.1** `void mln::Iterator< adj_lower_face_fwd_iter< D >>::next ( ) [inherited]`

Go to the next element.

**Warning**

This is a final method; iterator classes should not re-define this method. The actual "next" operation has to be defined through the *next\_* method.

**Precondition**

The iterator is valid.

## 10.333 mln::topo::adj\_lower\_higher\_face\_bkd\_iter< D > Class Template Reference

Forward iterator on all the adjacent (n-1)-faces and (n+1)-faces of the n-face of an mln::complex<D>.

```
#include <adj_lower_higher_face_iter.hh>
```

Inherits `complex_relative_iterator_sequence< adj_higher_face_bkd_iter< D >, adj_lower_face_bkd_iter< D >, adj_lower_higher_face_bkd_iter< D >>`.

**Public Member Functions**

- `void next ()`  
*Go to the next element.*
- `adj_lower_higher_face_bkd_iter ()`  
*Construction.*

**10.333.1 Detailed Description**

```
template<unsigned D> class mln::topo::adj_lower_higher_face_bkd_iter< D >
```

Forward iterator on all the adjacent (n-1)-faces and (n+1)-faces of the n-face of an mln::complex<D>.

**Template Parameters**

*D* The dimension of the complex this iterator belongs to.

Definition at line 102 of file `adj_lower_higher_face_iter.hh`.

**10.333.2 Constructor & Destructor Documentation**

```
10.333.2.1 template<unsigned D> mln::topo::adj_lower_higher_face_bkd_iter< D
>::adj_lower_higher_face_bkd_iter ( ) [inline]
```

Construction.

Definition at line 152 of file `adj_lower_higher_face_iter.hh`.



### 10.333.3 Member Function Documentation

**10.333.3.1** void mln::Iterator< adj\_lower\_higher\_face\_bkd\_iter< D > >::next ( )  
[inherited]

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

#### Precondition

The iterator is valid.

## 10.334 mln::topo::adj\_lower\_higher\_face\_fwd\_iter< D > Class Template Reference

Forward iterator on all the adjacent (n-1)-faces and (n+1)-faces of the n-face of an mln::complex<D>.

```
#include <adj_lower_higher_face_iter.hh>
```

Inherits complex\_relative\_iterator\_sequence< adj\_lower\_face\_fwd\_iter< D >, adj\_higher\_face\_fwd\_iter< D >, adj\_lower\_higher\_face\_fwd\_iter< D > >.

### Public Member Functions

- void [next](#) ()

*Go to the next element.*

- [adj\\_lower\\_higher\\_face\\_fwd\\_iter](#) ()

*Construction.*

### 10.334.1 Detailed Description

```
template<unsigned D> class mln::topo::adj_lower_higher_face_fwd_iter< D >
```

Forward iterator on all the adjacent (n-1)-faces and (n+1)-faces of the n-face of an mln::complex<D>.

#### Template Parameters

*D* The dimension of the complex this iterator belongs to.

Definition at line 71 of file adj\_lower\_higher\_face\_iter.hh.

### 10.334.2 Constructor & Destructor Documentation

**10.334.2.1** `template<unsigned D> mln::topo::adj_lower_higher_face_fwd_iter< D >::adj_lower_higher_face_fwd_iter ( ) [inline]`

Construction.

Definition at line 133 of file `adj_lower_higher_face_iter.hh`.

### 10.334.3 Member Function Documentation

**10.334.3.1** `void mln::Iterator< adj_lower_higher_face_fwd_iter< D > >::next ( ) [inherited]`

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

#### Precondition

The iterator is valid.

## 10.335 mln::topo::adj\_m\_face\_bkd\_iter< D > Class Template Reference

Backward iterator on all the m-faces transitively adjacent to a (reference) n-face in a complex.

```
#include <adj_m_face_iter.hh>
```

Inherits `backward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_m_face_bkd_iter< D >>`, and `mln::topo::internal::adj_m_face_iterator< D >`.

### Public Member Functions

- void [next](#) ()

*Go to the next element.*

- [adj\\_m\\_face\\_bkd\\_iter](#) ()

*Construction.*

- `template<typename Fref >`  
[adj\\_m\\_face\\_bkd\\_iter](#) (const Fref &f\_ref, unsigned m)

*Constructs an iterator, with f\_ref as reference face, and a target dimension equal to m.*

### 10.335.1 Detailed Description

**template<unsigned D> class mln::topo::adj\_m\_face\_bkd\_iter< D >**

Backward iterator on all the m-faces transitively adjacent to a (reference) n-face in a complex.

#### Template Parameters

*D* The dimension of the complex this iterator belongs to.

The dimension parameter (*m\_*) must be lower or equal to *D*.

If *m\_* is equal to the dimension of the reference face, then the iterated set is empty.

Definition at line 118 of file adj\_m\_face\_iter.hh.

### 10.335.2 Constructor & Destructor Documentation

**10.335.2.1 template<unsigned D> mln::topo::adj\_m\_face\_bkd\_iter< D >::adj\_m\_face\_bkd\_iter ( ) [inline]**

Construction.

Construct an iterator, with an invalid reference face, and a target dimension equal to 0.

Definition at line 223 of file adj\_m\_face\_iter.hh.

**10.335.2.2 template<unsigned D> template<typename Fref > mln::topo::adj\_m\_face\_bkd\_iter< D >::adj\_m\_face\_bkd\_iter ( const Fref & f\_ref, unsigned m ) [inline]**

Constructs an iterator, with *f\_ref* as reference face, and a target dimension equal to *m*.

Definition at line 230 of file adj\_m\_face\_iter.hh.

### 10.335.3 Member Function Documentation

**10.335.3.1 void mln::Iterator< adj\_m\_face\_bkd\_iter< D > >::next ( ) [inherited]**

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

#### Precondition

The iterator is valid.

## 10.336 mln::topo::adj\_m\_face\_fwd\_iter< D > Class Template Reference

Forward iterator on all the m-faces transitively adjacent to a (reference) n-face in a complex.

```
#include <adj_m_face_iter.hh>
```

Inherits `forward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_m_face_fwd_iter< D > >`, and `mln::topo::internal::adj_m_face_iterator< D >`.

## Public Member Functions

- void `next()`

*Go to the next element.*

- `adj_m_face_fwd_iter()`

*Construction.*

- `template<typename Fref >`

`adj_m_face_fwd_iter(const Fref &f_ref, unsigned m)`

*Constructs an iterator, with  $f\_ref$  as reference face, and a target dimension equal to  $m$ .*

### 10.336.1 Detailed Description

`template<unsigned D> class mln::topo::adj_m_face_fwd_iter< D >`

Forward iterator on all the  $m$ -faces transitively adjacent to a (reference)  $n$ -face in a complex.

#### Template Parameters

**$D$**  The dimension of the complex this iterator belongs to.

The dimension parameter ( $m_$ ) must be lower or equal to  $D$ .

If  $m_$  is equal to the dimension of the reference face, then the iterated set is empty.

Definition at line 70 of file `adj_m_face_iter.hh`.

### 10.336.2 Constructor & Destructor Documentation

**10.336.2.1** `template<unsigned D> mln::topo::adj_m_face_fwd_iter< D >::adj_m_face_fwd_iter( ) [inline]`

Construction.

Construct an iterator, with an invalid reference face, and a target dimension equal to 0.

Definition at line 194 of file `adj_m_face_iter.hh`.

**10.336.2.2** `template<unsigned D> template<typename Fref > mln::topo::adj_m_face_fwd_iter< D >::adj_m_face_fwd_iter( const Fref & f_ref, unsigned m ) [inline]`

Constructs an iterator, with  $f\_ref$  as reference face, and a target dimension equal to  $m$ .

Definition at line 201 of file `adj_m_face_iter.hh`.

### 10.336.3 Member Function Documentation

#### 10.336.3.1 void mln::Iterator< adj\_m\_face\_fwd\_iter< D > >::next ( ) [inherited]

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_method*.

#### Precondition

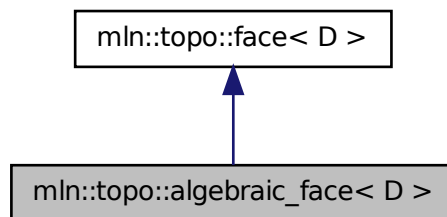
The iterator is valid.

## 10.337 mln::topo::algebraic\_face< D > Class Template Reference

Algebraic face handle in a complex; the face dimension is dynamic.

```
#include <algebraic_face.hh>
```

Inheritance diagram for mln::topo::algebraic\_face< D >:



### Public Member Functions

- [algebraic\\_face](#) ()  
*Build a non-initialized algebraic face handle.*
- [algebraic\\_face](#) ([complex](#)< D > &[complex](#), unsigned n, unsigned face\_id, bool [sign](#))  
*Build an algebraic face handle from complex and face\_id.*
- [template](#)<unsigned N>  
[algebraic\\_face](#) (const [algebraic\\_n\\_face](#)< N, D > &f)  
*Build a face handle from an mln::topo::algebraic\_n\_face.*
- [algebraic\\_face](#) (const [face](#)< D > &f, bool [sign](#))  
*Build an algebraic face handle from an mln::face.*

- void [invalidate](#) ()  
*Invalidate this handle.*
- bool [is\\_valid](#) () const  
*Is this handle valid?*
- bool [sign](#) () const  
*Accessors.*
- void [set\\_sign](#) (bool [sign](#))  
*Set the sign of this face.*
- [complex](#)< D > [cplx](#) () const  
*Accessors.*
- unsigned [n](#) () const  
*Return the dimension of the face.*
- unsigned [face\\_id](#) () const  
*Return the id of the face.*
- void [set\\_cplx](#) (const [complex](#)< D > &cplx)  
*Set the complex the face belongs to.*
- void [set\\_n](#) (unsigned n)  
*Set the dimension of the face.*
- void [inc\\_n](#) ()  
*Increment the dimension of the face.*
- void [dec\\_n](#) ()  
*Decrement the dimension of the face.*
- void [set\\_face\\_id](#) (unsigned [face\\_id](#))  
*Set the id of the face.*
- void [inc\\_face\\_id](#) ()  
*Increment the id of the face.*
- void [dec\\_face\\_id](#) ()  
*Decrement the id of the face.*
- template<unsigned N>  
[face\\_data](#)< N, D > & [data](#) () const  
*Return the `mln::topo::face_data` pointed by this handle.*
- std::vector< [algebraic\\_face](#)< D > > [lower\\_dim\\_adj\\_faces](#) () const  
*Return an array of face handles pointing to adjacent (n-1)-faces.*
- std::vector< [algebraic\\_face](#)< D > > [higher\\_dim\\_adj\\_faces](#) () const  
*Return an array of face handles pointing to adjacent (n+1)-faces.*

### 10.337.1 Detailed Description

**template<unsigned D> class mln::topo::algebraic\_face< D >**

Algebraic face handle in a complex; the face dimension is dynamic. Contrary to an [mln::topo::algebraic\\_n\\_face](#), the dimension of an [mln::topo::algebraic\\_face](#) is not fixed.

Definition at line 60 of file algebraic\_face.hh.

### 10.337.2 Constructor & Destructor Documentation

**10.337.2.1 template<unsigned D> mln::topo::algebraic\_face< D >::algebraic\_face ( ) [inline]**

Build a non-initialized algebraic face handle.

Definition at line 157 of file algebraic\_face.hh.

**10.337.2.2 template<unsigned D> mln::topo::algebraic\_face< D >::algebraic\_face ( complex< D > & complex, unsigned n, unsigned face\_id, bool sign ) [inline]**

Build an algebraic face handle from *complex* and *face\_id*.

Definition at line 164 of file algebraic\_face.hh.

**10.337.2.3 template<unsigned D> mln::topo::algebraic\_face< D >::algebraic\_face ( const face< D > & f, bool sign ) [inline]**

Build an algebraic face handle from an mln::face.

Definition at line 174 of file algebraic\_face.hh.

References mln::topo::face< D >::n().

**10.337.2.4 template<unsigned D> template<unsigned N> mln::topo::algebraic\_face< D >::algebraic\_face ( const algebraic\_n\_face< N, D > & f ) [inline]**

Build a face handle from an [mln::topo::algebraic\\_n\\_face](#).

Definition at line 184 of file algebraic\_face.hh.

### 10.337.3 Member Function Documentation

**10.337.3.1 template<unsigned D> complex< D > mln::topo::face< D >::cplx ( ) const [inline, inherited]**

Accessors.

Return the complex the face belongs to.

Definition at line 224 of file face.hh.

Referenced by mln::complex\_psite< D, G >::complex\_psite(), mln::topo::operator!==( ), and mln::topo::operator==( ).

**10.337.3.2** `template<unsigned D> template<unsigned N> face_data< N, D > & mln::topo::face< D >::data( ) const [inline, inherited]`

Return the `mln::topo::face_data` pointed by this handle.

Definition at line 305 of file `face.hh`.

References `mln::topo::face< D >::is_valid()`.

**10.337.3.3** `template<unsigned D> void mln::topo::face< D >::dec_face_id( ) [inline, inherited]`

Decrement the id of the face.

Definition at line 296 of file `face.hh`.

**10.337.3.4** `template<unsigned D> void mln::topo::face< D >::dec_n( ) [inline, inherited]`

Decrement the dimension of the face.

Definition at line 272 of file `face.hh`.

**10.337.3.5** `template<unsigned D> unsigned mln::topo::face< D >::face_id( ) const [inline, inherited]`

Return the id of the face.

Definition at line 240 of file `face.hh`.

Referenced by `mln::geom::complex_geometry< D, P >::operator()()`, and `mln::topo::operator==( )`.

**10.337.3.6** `template<unsigned D> std::vector< algebraic_face< D > > mln::topo::face< D >::higher_dim_adj_faces( ) const [inline, inherited]`

Return an array of face handles pointing to adjacent (n+1)-faces.

Definition at line 370 of file `face.hh`.

**10.337.3.7** `template<unsigned D> void mln::topo::face< D >::inc_face_id( ) [inline, inherited]`

Increment the id of the face.

Definition at line 288 of file `face.hh`.

**10.337.3.8** `template<unsigned D> void mln::topo::face< D >::inc_n( ) [inline, inherited]`

Increment the dimension of the face.

Definition at line 264 of file `face.hh`.



**10.337.3.9** `template<unsigned D> void mln::topo::face< D >::invalidate ( ) [inline, inherited]`

Invalidate this handle.

Definition at line 215 of file face.hh.

References mln::topo::face< D >::set\_face\_id(), and mln::topo::face< D >::set\_n().

**10.337.3.10** `template<unsigned D> bool mln::topo::face< D >::is_valid ( ) const [inline, inherited]`

Is this handle valid?

Definition at line 207 of file face.hh.

Referenced by mln::topo::face< D >::data().

**10.337.3.11** `template<unsigned D> std::vector< algebraic_face< D > > mln::topo::face< D >::lower_dim_adj_faces ( ) const [inline, inherited]`

Return an array of face handles pointing to adjacent (n-1)-faces.

Definition at line 357 of file face.hh.

**10.337.3.12** `template<unsigned D> unsigned mln::topo::face< D >::n ( ) const [inline, inherited]`

Return the dimension of the face.

Definition at line 232 of file face.hh.

Referenced by mln::topo::algebraic\_face< D >::algebraic\_face(), mln::geom::complex\_geometry< D, P >::operator>(), and mln::topo::operator==( ).

**10.337.3.13** `template<unsigned D> void mln::topo::face< D >::set_cplx ( const complex< D > & cplx ) [inline, inherited]`

Set the complex the face belongs to.

Definition at line 248 of file face.hh.

**10.337.3.14** `template<unsigned D> void mln::topo::face< D >::set_face_id ( unsigned face_id ) [inline, inherited]`

Set the id of the face.

Definition at line 280 of file face.hh.

Referenced by mln::topo::face< D >::invalidate().

**10.337.3.15** `template<unsigned D> void mln::topo::face< D >::set_n ( unsigned n ) [inline, inherited]`

Set the dimension of the face.

Definition at line 256 of file face.hh.

Referenced by `mln::topo::face< D >::invalidate()`.

**10.337.3.16** `template<unsigned D> void mln::topo::algebraic_face< D >::set_sign ( bool sign )`  
**[inline]**

Set the sign of this face.

Definition at line 203 of file algebraic\_face.hh.

**10.337.3.17** `template<unsigned D> bool mln::topo::algebraic_face< D >::sign ( ) const`  
**[inline]**

Accessors.

Return the sign of this face.

Definition at line 195 of file algebraic\_face.hh.

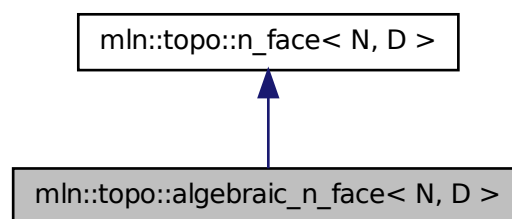
Referenced by `mln::topo::operator==( )`.

## 10.338 `mln::topo::algebraic_n_face< N, D >` Class Template Reference

Algebraic N-face handle in a complex.

```
#include <algebraic_n_face.hh>
```

Inheritance diagram for `mln::topo::algebraic_n_face< N, D >`:



### Public Member Functions

- `algebraic_n_face ( )`  
*Build a non-initialized algebraic face handle.*
- `algebraic_n_face (complex< D > &complex, unsigned face_id, bool sign)`

*Build an algebraic face handle from complex and face\_id.*

- `algebraic_n_face` (const `n_face`< N, D > &f, bool `sign`)  
*Build an algebraic face handle from an mln::n\_face.*
- void `invalidate` ()  
*Invalidate this handle.*
- bool `is_valid` () const  
*Is this handle valid?*
- bool `sign` () const  
*Accessors.*
- void `set_sign` (bool `sign`)  
*Set the sign of this face.*
- `complex`< D > `cplx` () const  
*Accessors.*
- unsigned `face_id` () const  
*Return the id of the face.*
- void `set_cplx` (const `complex`< D > &cplx)  
*Set the complex the face belongs to.*
- unsigned `n` () const  
*Return the dimension of the face.*
- void `set_face_id` (unsigned `face_id`)  
*Set the id of the face.*
- void `inc_face_id` ()  
*Increment the id of the face.*
- void `dec_face_id` ()  
*Decrement the id of the face.*
- `face_data`< N, D > & `data` () const  
*Return the mln::topo::face\_data pointed by this handle.*
- `std::vector`< `algebraic_n_face`< N-1, D > > `lower_dim_adj_faces` () const  
*Return an array of face handles pointing to adjacent (n-1)-faces.*
- `std::vector`< `algebraic_n_face`< N+1, D > > `higher_dim_adj_faces` () const  
*Return an array of face handles pointing to adjacent (n+1)-faces.*

### 10.338.1 Detailed Description

`template<unsigned N, unsigned D> class mln::topo::algebraic_n_face< N, D >`

Algebraic `N-face` handle in a complex. Contrary to an [mln::topo::algebraic\\_face](#), the dimension of an [mln::topo::algebraic\\_n\\_face](#) is fixed.

Definition at line 50 of file `algebraic_n_face.hh`.

### 10.338.2 Constructor & Destructor Documentation

**10.338.2.1** `template<unsigned N, unsigned D> mln::topo::algebraic_n_face< N, D >::algebraic_n_face ( ) [inline]`

Build a non-initialized algebraic face handle.

Definition at line 166 of file `algebraic_n_face.hh`.

References `mln::topo::n_face< N, D >::is_valid()`.

**10.338.2.2** `template<unsigned N, unsigned D> mln::topo::algebraic_n_face< N, D >::algebraic_n_face ( complex< D > & complex, unsigned face_id, bool sign ) [inline]`

Build an algebraic face handle from `complex` and `face_id`.

Definition at line 176 of file `algebraic_n_face.hh`.

**10.338.2.3** `template<unsigned N, unsigned D> mln::topo::algebraic_n_face< N, D >::algebraic_n_face ( const n_face< N, D > & f, bool sign ) [inline]`

Build an algebraic face handle from an `mln::n_face`.

Definition at line 186 of file `algebraic_n_face.hh`.

### 10.338.3 Member Function Documentation

**10.338.3.1** `template<unsigned N, unsigned D> complex< D > mln::topo::n_face< N, D >::cplx ( ) const [inline, inherited]`

Accessors.

Return the complex the face belongs to.

Definition at line 195 of file `n_face.hh`.

Referenced by `mln::topo::n_faces_set< N, D >::add()`, `mln::topo::operator!=()`, and `mln::topo::operator==()`.

**10.338.3.2** `template<unsigned N, unsigned D> face_data< N, D > & mln::topo::n_face< N, D >::data ( ) const [inline, inherited]`

Return the `mln::topo::face_data` pointed by this handle.

Definition at line 251 of file n\_face.hh.

References mln::topo::n\_face< N, D >::is\_valid().

**10.338.3.3** `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::dec_face_id ( )  
[inline, inherited]`

Decrement the id of the face.

Definition at line 243 of file n\_face.hh.

**10.338.3.4** `template<unsigned N, unsigned D> unsigned mln::topo::n_face< N, D >::face_id ( )  
const [inline, inherited]`

Return the id of the face.

Definition at line 211 of file n\_face.hh.

Referenced by mln::topo::operator==( ).

**10.338.3.5** `template<unsigned N, unsigned D> std::vector< algebraic_n_face< N+1, D >  
> mln::topo::n_face< N, D >::higher_dim_adj_faces ( ) const [inline,  
inherited]`

Return an array of face handles pointing to adjacent (n+1)-faces.

Definition at line 270 of file n\_face.hh.

References mln::topo::n\_face< N, D >::is\_valid().

Referenced by mln::topo::edge().

**10.338.3.6** `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::inc_face_id ( )  
[inline, inherited]`

Increment the id of the face.

Definition at line 235 of file n\_face.hh.

**10.338.3.7** `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::invalidate ( )  
[inline, inherited]`

Invalidate this handle.

Definition at line 187 of file n\_face.hh.

References mln::topo::n\_face< N, D >::set\_face\_id().

**10.338.3.8** `template<unsigned N, unsigned D> bool mln::topo::n_face< N, D >::is_valid ( )  
const [inline, inherited]`

Is this handle valid?

Definition at line 179 of file n\_face.hh.

Referenced by `mln::topo::algebraic_n_face< N, D >::algebraic_n_face()`, `mln::topo::n_face< N, D >::data()`, `mln::topo::n_face< N, D >::higher_dim_adj_faces()`, `mln::topo::n_face< N, D >::lower_dim_adj_faces()`, and `mln::topo::n_face< N, D >::n_face()`.

**10.338.3.9** `template<unsigned N, unsigned D> std::vector< algebraic_n_face< N-1, D >> mln::topo::n_face< N, D >::lower_dim_adj_faces ( ) const [inline, inherited]`

Return an array of face handles pointing to adjacent (n-1)-faces.

Definition at line 260 of file `n_face.hh`.

References `mln::topo::n_face< N, D >::is_valid()`.

**10.338.3.10** `template<unsigned N, unsigned D> unsigned mln::topo::n_face< N, D >::n ( ) const [inline, inherited]`

Return the dimension of the face.

Definition at line 203 of file `n_face.hh`.

**10.338.3.11** `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::set_cplx ( const complex< D > & cplx ) [inline, inherited]`

Set the complex the face belongs to.

Definition at line 219 of file `n_face.hh`.

**10.338.3.12** `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::set_face_id ( unsigned face_id ) [inline, inherited]`

Set the id of the face.

Definition at line 227 of file `n_face.hh`.

Referenced by `mln::topo::n_face< N, D >::invalidate()`.

**10.338.3.13** `template<unsigned N, unsigned D> void mln::topo::algebraic_n_face< N, D >::set_sign ( bool sign ) [inline]`

Set the sign of this face.

Definition at line 205 of file `algebraic_n_face.hh`.

**10.338.3.14** `template<unsigned N, unsigned D> bool mln::topo::algebraic_n_face< N, D >::sign ( ) const [inline]`

Accessors.

Return the sign of this face.

Definition at line 197 of file `algebraic_n_face.hh`.

Referenced by `mln::topo::operator==( )`.

## 10.339 mln::topo::center\_only\_iter< D > Class Template Reference

[Iterator](#) on all the adjacent (n-1)-faces of the n-face of an mln::complex<D>.

```
#include <center_only_iter.hh>
```

Inherits forward\_complex\_relative\_iterator\_base< topo::face< D >, algebraic\_face< D >, center\_only\_iter< D >>.

### Public Member Functions

- void [next](#) ()  
*Go to the next element.*
- [center\\_only\\_iter](#) ()  
*Construction.*

### 10.339.1 Detailed Description

```
template<unsigned D> class mln::topo::center_only_iter< D >
```

[Iterator](#) on all the adjacent (n-1)-faces of the n-face of an mln::complex<D>.

#### Template Parameters

*D* The dimension of the complex this iterator belongs to.

[mln::topo::center\\_only\\_iter](#) inherits from mln::topo::internal::forward\_complex\_relative\_iterator\_base, but it could inherit from mln::topo::internal::backward\_complex\_relative\_iterator\_base as well, since it always contains a single element, the center/reference face (and the traversal order is meaningless).

This iterator is essentially used to implement other iterators.

#### See also

```
mln::topo::centered_iter_adapter
mln::complex_lower_window
mln::complex_higher_window
mln::complex_lower_higher_window
```

Definition at line 73 of file center\_only\_iter.hh.

### 10.339.2 Constructor & Destructor Documentation

```
10.339.2.1 template<unsigned D> mln::topo::center_only_iter< D >::center_only_iter ( )
[inline]
```

Construction.

Definition at line 107 of file center\_only\_iter.hh.

### 10.339.3 Member Function Documentation

#### 10.339.3.1 void mln::Iterator< center\_only\_iter< D > >::next ( ) [inherited]

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

#### Precondition

The iterator is valid.

## 10.340 mln::topo::centered\_bkd\_iter\_adapter< D, I > Class Template Reference

Forward complex relative iterator adapters adding the central (reference) point to the set of iterated faces.

```
#include <centered_iter_adapter.hh>
```

Inherits complex\_relative\_iterator\_sequence< I, center\_only\_iter< D >, centered\_bkd\_iter\_adapter< D, I > >.

### Public Member Functions

- void [next](#) ()

*Go to the next element.*

- [centered\\_bkd\\_iter\\_adapter](#) ()

*Construction.*

#### 10.340.1 Detailed Description

```
template<unsigned D, typename I> class mln::topo::centered_bkd_iter_adapter< D, I >
```

Forward complex relative iterator adapters adding the central (reference) point to the set of iterated faces.

#### Template Parameters

*D* The dimension of the complex this iterator belongs to.

*I* The adapted complex relative iterator.

Definition at line 91 of file centered\_iter\_adapter.hh.



## 10.340.2 Constructor & Destructor Documentation

**10.340.2.1** `template<unsigned D, typename I > mln::topo::centered_bkd_iter_adapter< D, I >::centered_bkd_iter_adapter ( ) [inline]`

Construction.

Definition at line 141 of file centered\_iter\_adapter.hh.

## 10.340.3 Member Function Documentation

**10.340.3.1** `void mln::Iterator< centered_bkd_iter_adapter< D, I > >::next ( ) [inherited]`

Go to the next element.

### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

### Precondition

The iterator is valid.

## 10.341 mln::topo::centered\_fwd\_iter\_adapter< D, I > Class Template Reference

Backward complex relative iterator adapters adding the central (reference) point to the set of iterated faces.

```
#include <centered_iter_adapter.hh>
```

Inherits `complex_relative_iterator_sequence< center_only_iter< D >, I, centered_fwd_iter_adapter< D, I >>`.

## Public Member Functions

- `void next ()`  
*Go to the next element.*
- `centered_fwd_iter_adapter ()`  
*Construction.*

## 10.341.1 Detailed Description

`template<unsigned D, typename I > class mln::topo::centered_fwd_iter_adapter< D, I >`

Backward complex relative iterator adapters adding the central (reference) point to the set of iterated faces.

### Template Parameters

*D* The dimension of the complex this iterator belongs to.

*I* The adapted complex relative iterator.

Definition at line 57 of file centered\_iter\_adapter.hh.

## 10.341.2 Constructor & Destructor Documentation

**10.341.2.1** `template<unsigned D, typename I > mln::topo::centered_fwd_iter_adapter< D, I >::centered_fwd_iter_adapter ( ) [inline]`

Construction.

Definition at line 122 of file centered\_iter\_adapter.hh.

## 10.341.3 Member Function Documentation

**10.341.3.1** `void mln::Iterator< centered_fwd_iter_adapter< D, I > >::next ( ) [inherited]`

Go to the next element.

### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_method*.

### Precondition

The iterator is valid.

## 10.342 mln::topo::complex< D > Class Template Reference

General complex of dimension D.

```
#include <complex.hh>
```

### Public Types

- typedef [face\\_bkd\\_iter< D >](#) [bkd\\_citer](#)  
*Backward mln::Iterator type iterating on all faces.*
- typedef [face\\_fwd\\_iter< D >](#) [fwd\\_citer](#)  
*Forward mln::Iterator type iterating on all faces.*

### Public Member Functions

- const void \* [addr](#) () const  
*Get the address of the data of this complex.*

- [complex](#) ()  
*Complex construction.*
- [n\\_face](#)< 0u, D > [add\\_face](#) ()  
*Add a 0-face to the complex.*
- `template<unsigned N>`  
[n\\_face](#)< N+1, D > [add\\_face](#) (const [n\\_faces\\_set](#)< N, D > &adjacent\_faces)  
*Add a (N+1)-face to the complex (with N >= 0).*
- unsigned [nfaces](#) () const  
*Static manipulators.*
- `template<unsigned N>`  
unsigned [nfaces\\_of\\_static\\_dim](#) () const  
*Return the number of N-faces.*
- unsigned [nfaces\\_of\\_dim](#) (unsigned n) const  
*Dynamic manipulators.*
- void [print](#) (std::ostream &ostr) const  
*Pretty-printing.*
- `template<unsigned N>`  
void [print\\_faces](#) (std::ostream &ostr) const  
*Print the faces of dimension N.*

### 10.342.1 Detailed Description

`template<unsigned D> class mln::topo::complex< D >`

General complex of dimension D.

Definition at line 87 of file complex.hh.

### 10.342.2 Member Typedef Documentation

**10.342.2.1** `template<unsigned D> typedef face_bkd_iter<D> mln::topo::complex< D >::bkd_citer`

Backward [mln::Iterator](#) type iterating on all faces.

Definition at line 93 of file complex.hh.

### 10.342.2.2 `template<unsigned D> typedef face_fwd_iter<D> mln::topo::complex< D >::fwd_citer`

Forward [mln::Iterator](#) type iterating on all faces.

Definition at line 91 of file complex.hh.

## 10.342.3 Constructor & Destructor Documentation

### 10.342.3.1 `template<unsigned D> mln::topo::complex< D >::complex ( ) [inline]`

Complex construction.

Create a new `D-complex`.

Definition at line 471 of file complex.hh.

## 10.342.4 Member Function Documentation

### 10.342.4.1 `template<unsigned D> n_face< 0u, D > mln::topo::complex< D >::add_face ( ) [inline]`

Add a 0-face to the complex.

Definition at line 480 of file complex.hh.

### 10.342.4.2 `template<unsigned D> template<unsigned N> n_face< N+1, D > mln::topo::complex< D >::add_face ( const n_faces_set< N, D > & adjacent_faces ) [inline]`

Add a (N+1)-face to the complex (with  $N \geq 0$ ).

#### Parameters

*adjacent\_faces* The (N-1)-faces adjacent to the new N-face.

Definition at line 493 of file complex.hh.

References `mln::topo::n_faces_set< N, D >::faces()`.

### 10.342.4.3 `template<unsigned D> const void * mln::topo::complex< D >::addr ( ) const [inline]`

Get the address of the data of this complex.

This address is a concise and useful information to print and track the actual content of this complex.

Definition at line 699 of file complex.hh.

### 10.342.4.4 `template<unsigned D> unsigned mln::topo::complex< D >::nfaces ( ) const [inline]`

Static manipulators.

These methods use statically-known input.

Return the total number of faces, whatever their dimension.

Definition at line 579 of file complex.hh.

#### 10.342.4.5 `template<unsigned D> unsigned mln::topo::complex< D >::nfaces_of_dim ( unsigned n ) const [inline]`

Dynamic manipulators.

These methods use input know as run time.

Return the number of *n*-faces.

Warning, this function has a complexity linear in term of N, since each `n_faces_set` is checked (the present implementation does not provide a direct access to `n_faces_set` through a dynamic value of the dimension).

Definition at line 601 of file complex.hh.

#### 10.342.4.6 `template<unsigned D> template<unsigned N> unsigned mln::topo::complex< D >::nfaces_of_static_dim ( ) const [inline]`

Return the number of N-faces.

Definition at line 588 of file complex.hh.

#### 10.342.4.7 `template<unsigned D> void mln::topo::complex< D >::print ( std::ostream & ostr ) const [inline]`

Pretty-printing.

Print the complex.

Definition at line 679 of file complex.hh.

Referenced by `mln::topo::operator<<()`.

#### 10.342.4.8 `template<unsigned D> template<unsigned N> void mln::topo::complex< D >::print_faces ( std::ostream & ostr ) const [inline]`

Print the faces of dimension N.

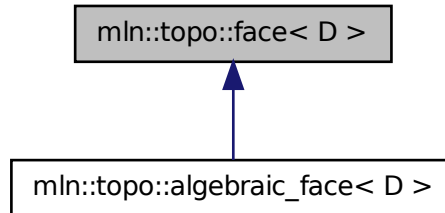
Definition at line 688 of file complex.hh.

## 10.343 mln::topo::face< D > Class Template Reference

Face handle in a complex; the face dimension is dynamic.

```
#include <face.hh>
```

Inheritance diagram for `mln::topo::face< D >`:



## Public Member Functions

- `face ()`  
*Build a non-initialized face handle.*
- `face (complex< D > &complex, unsigned n, unsigned face_id)`  
*Build a face handle from complex and face\_id.*
- `template<unsigned N>`  
`face (const n_face< N, D > &f)`  
*Build a face handle from an `mln::topo::n_face`.*
- `void invalidate ()`  
*Invalidate this handle.*
- `bool is_valid () const`  
*Is this handle valid?*
- `complex< D > cplx () const`  
*Accessors.*
- `unsigned n () const`  
*Return the dimension of the face.*
- `unsigned face_id () const`  
*Return the id of the face.*
- `void set_cplx (const complex< D > &cplx)`  
*Set the complex the face belongs to.*
- `void set_n (unsigned n)`  
*Set the dimension of the face.*
- `void inc_n ()`

*Increment the dimension of the face.*

- void `dec_n ()`  
*Decrement the dimension of the face.*
- void `set_face_id (unsigned face_id)`  
*Set the id of the face.*
- void `inc_face_id ()`  
*Increment the id of the face.*
- void `dec_face_id ()`  
*Decrement the id of the face.*
- `template<unsigned N>`  
`face_data< N, D > & data () const`  
*Return the mln::topo::face\_data pointed by this handle.*
- `std::vector< algebraic_face< D > > lower_dim_adj_faces () const`  
*Return an array of face handles pointing to adjacent (n-1)-faces.*
- `std::vector< algebraic_face< D > > higher_dim_adj_faces () const`  
*Return an array of face handles pointing to adjacent (n+1)-faces.*

### 10.343.1 Detailed Description

`template<unsigned D> class mln::topo::face< D >`

Face handle in a complex; the face dimension is dynamic. Contrary to an `mln::topo::n_face`, the dimension of an `mln::topo::face` is not fixed.

Definition at line 64 of file face.hh.

### 10.343.2 Constructor & Destructor Documentation

**10.343.2.1** `template<unsigned D> mln::topo::face< D >::face ( ) [inline]`

Build a non-initialized face handle.

Definition at line 178 of file face.hh.

**10.343.2.2** `template<unsigned D> mln::topo::face< D >::face ( complex< D > & complex, unsigned n, unsigned face_id ) [inline]`

Build a face handle from *complex* and *face\_id*.

Definition at line 187 of file face.hh.

**10.343.2.3** `template<unsigned D> template<unsigned N> mln::topo::face< D >::face ( const n_face< N, D > & f ) [inline]`

Build a face handle from an `mln::topo::n_face`.

Definition at line 197 of file face.hh.

### 10.343.3 Member Function Documentation

**10.343.3.1** `template<unsigned D> complex< D > mln::topo::face< D >::cplx ( ) const [inline]`

Accessors.

Return the complex the face belongs to.

Definition at line 224 of file face.hh.

Referenced by `mln::complex_psite< D, G >::complex_psite()`, `mln::topo::operator!=()`, and `mln::topo::operator==()`.

**10.343.3.2** `template<unsigned D> template<unsigned N> face_data< N, D > & mln::topo::face< D >::data ( ) const [inline]`

Return the `mln::topo::face_data` pointed by this handle.

Definition at line 305 of file face.hh.

References `mln::topo::face< D >::is_valid()`.

**10.343.3.3** `template<unsigned D> void mln::topo::face< D >::dec_face_id ( ) [inline]`

Decrement the id of the face.

Definition at line 296 of file face.hh.

**10.343.3.4** `template<unsigned D> void mln::topo::face< D >::dec_n ( ) [inline]`

Decrement the dimension of the face.

Definition at line 272 of file face.hh.

**10.343.3.5** `template<unsigned D> unsigned mln::topo::face< D >::face_id ( ) const [inline]`

Return the id of the face.

Definition at line 240 of file face.hh.

Referenced by `mln::geom::complex_geometry< D, P >::operator()()`, and `mln::topo::operator==()`.

**10.343.3.6** `template<unsigned D> std::vector< algebraic_face< D > > mln::topo::face< D >::higher_dim_adj_faces ( ) const [inline]`

Return an array of face handles pointing to adjacent (n+1)-faces.

Definition at line 370 of file face.hh.



**10.343.3.7** `template<unsigned D> void mln::topo::face< D >::inc_face_id ( ) [inline]`

Increment the id of the face.

Definition at line 288 of file face.hh.

**10.343.3.8** `template<unsigned D> void mln::topo::face< D >::inc_n ( ) [inline]`

Increment the dimension of the face.

Definition at line 264 of file face.hh.

**10.343.3.9** `template<unsigned D> void mln::topo::face< D >::invalidate ( ) [inline]`

Invalidate this handle.

Definition at line 215 of file face.hh.

References mln::topo::face< D >::set\_face\_id(), and mln::topo::face< D >::set\_n().

**10.343.3.10** `template<unsigned D> bool mln::topo::face< D >::is_valid ( ) const [inline]`

Is this handle valid?

Definition at line 207 of file face.hh.

Referenced by mln::topo::face< D >::data().

**10.343.3.11** `template<unsigned D> std::vector< algebraic_face< D > > mln::topo::face< D >::lower_dim_adj_faces ( ) const [inline]`

Return an array of face handles pointing to adjacent (n-1)-faces.

Definition at line 357 of file face.hh.

**10.343.3.12** `template<unsigned D> unsigned mln::topo::face< D >::n ( ) const [inline]`

Return the dimension of the face.

Definition at line 232 of file face.hh.

Referenced by mln::topo::algebraic\_face< D >::algebraic\_face(), mln::geom::complex\_geometry< D, P >::operator>(), and mln::topo::operator==().

**10.343.3.13** `template<unsigned D> void mln::topo::face< D >::set_cplx ( const complex< D > & cplx ) [inline]`

Set the complex the face belongs to.

Definition at line 248 of file face.hh.

**10.343.3.14** `template<unsigned D> void mln::topo::face< D >::set_face_id ( unsigned face_id )`  
**[inline]**

Set the id of the face.

Definition at line 280 of file face.hh.

Referenced by `mln::topo::face< D >::invalidate()`.

**10.343.3.15** `template<unsigned D> void mln::topo::face< D >::set_n ( unsigned n )`  
**[inline]**

Set the dimension of the face.

Definition at line 256 of file face.hh.

Referenced by `mln::topo::face< D >::invalidate()`.

## 10.344 mln::topo::face\_bkd\_iter< D > Class Template Reference

Backward iterator on all the faces of an `mln::complex<D>`.

```
#include <face_iter.hh>
```

Inherits `complex_set_iterator_base< topo::face< D >, face_bkd_iter< D > >`.

### Public Member Functions

- void [next](#) ()  
*Go to the next element.*
- [face\\_bkd\\_iter](#) ()  
*Construction and assignment.*
- void [start](#) ()  
*Manipulation.*

### 10.344.1 Detailed Description

`template<unsigned D> class mln::topo::face_bkd_iter< D >`

Backward iterator on all the faces of an `mln::complex<D>`.

#### Template Parameters

- D*** The dimension of the complex this iterator belongs to.

Definition at line 112 of file face\_iter.hh.

## 10.344.2 Constructor & Destructor Documentation

### 10.344.2.1 `template<unsigned D> mln::topo::face_bkd_iter< D >::face_bkd_iter ( )` [inline]

Construction and assignment.

Definition at line 207 of file face\_iter.hh.

## 10.344.3 Member Function Documentation

### 10.344.3.1 `void mln::Iterator< face_bkd_iter< D > >::next ( )` [inherited]

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_method*.

#### Precondition

The iterator is valid.

### 10.344.3.2 `template<unsigned D> void mln::topo::face_bkd_iter< D >::start ( )` [inline]

Manipulation.

Start an iteration.

Definition at line 224 of file face\_iter.hh.

## 10.345 mln::topo::face\_fwd\_iter< D > Class Template Reference

Forward iterator on all the faces of an mln::complex<D>.

```
#include <face_iter.hh>
```

Inherits complex\_set\_iterator\_base< topo::face< D >, face\_fwd\_iter< D > >.

### Public Member Functions

- void [next](#) ()  
*Go to the next element.*
- [face\\_fwd\\_iter](#) ()  
*Construction and assignment.*
- void [start](#) ()  
*Manipulation.*

### 10.345.1 Detailed Description

**template<unsigned D> class mln::topo::face\_fwd\_iter< D >**

Forward iterator on all the faces of an mln::complex<D>.

#### Template Parameters

*D* The dimension of the complex this iterator belongs to.

Definition at line 69 of file face\_iter.hh.

### 10.345.2 Constructor & Destructor Documentation

**10.345.2.1 template<unsigned D> mln::topo::face\_fwd\_iter< D >::face\_fwd\_iter ( )**  
**[inline]**

Construction and assignment.

Definition at line 155 of file face\_iter.hh.

### 10.345.3 Member Function Documentation

**10.345.3.1 void mln::Iterator< face\_fwd\_iter< D > >::next ( )** **[inherited]**

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

#### Precondition

The iterator is valid.

**10.345.3.2 template<unsigned D> void mln::topo::face\_fwd\_iter< D >::start ( )** **[inline]**

Manipulation.

Test if the iterator is valid.

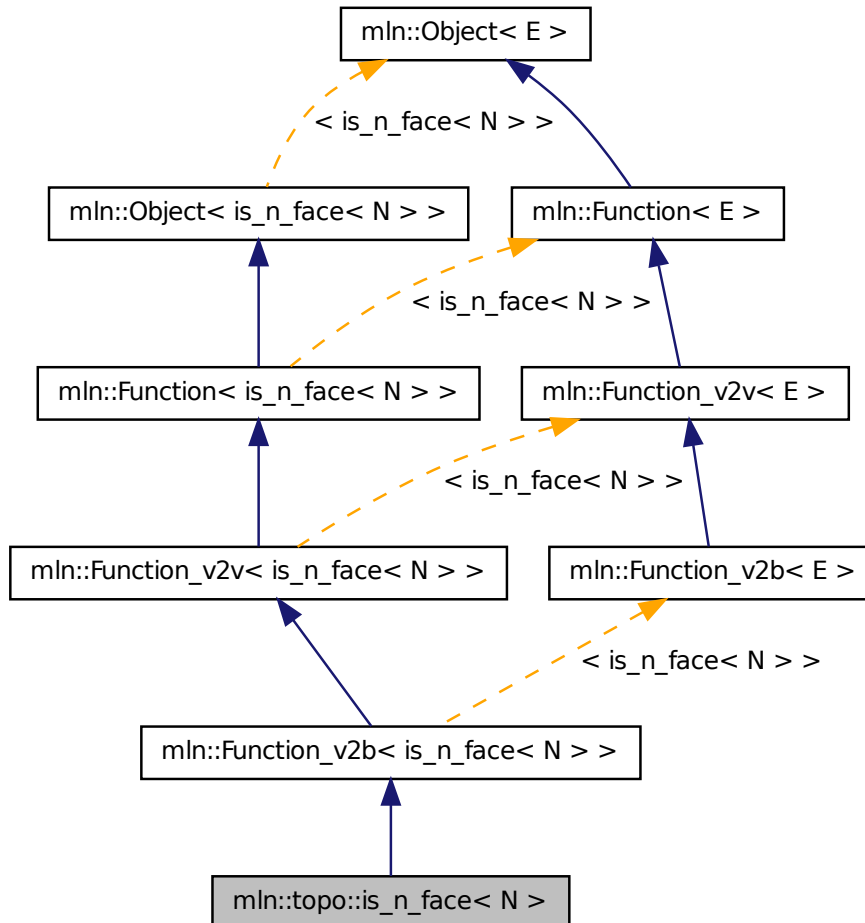
Definition at line 172 of file face\_iter.hh.

## 10.346 mln::topo::is\_n\_face< N > Struct Template Reference

A functor testing wheter a [mln::complex\\_psite](#) is an N -face.

```
#include <is_n_face.hh>
```

Inheritance diagram for mln::topo::is\_n\_face< N >:



### 10.346.1 Detailed Description

```
template<unsigned N> struct mln::topo::is_n_face< N >
```

A functor testing whether a [mln::complex\\_site](#) is an N-face.

Definition at line 48 of file `is_n_face.hh`.

## 10.347 mln::topo::is\_simple\_2d\_t< N > Struct Template Reference

Test if a point is simple or not.

```
#include <is_simple_2d.hh>
```

### 10.347.1 Detailed Description

```
template<typename N> struct mln::topo::is_simple_2d_t< N >
```

Test if a point is simple or not. A point of an object is simple if in its c8 neighborhood, there is exactly one connected component of the object, and only one connected component of the background Examples : ( | == object, - = background)

- - | | P | Here p is simple in the c4 and c8 case. | | |

- | - | P | Here p is never simple. | | |

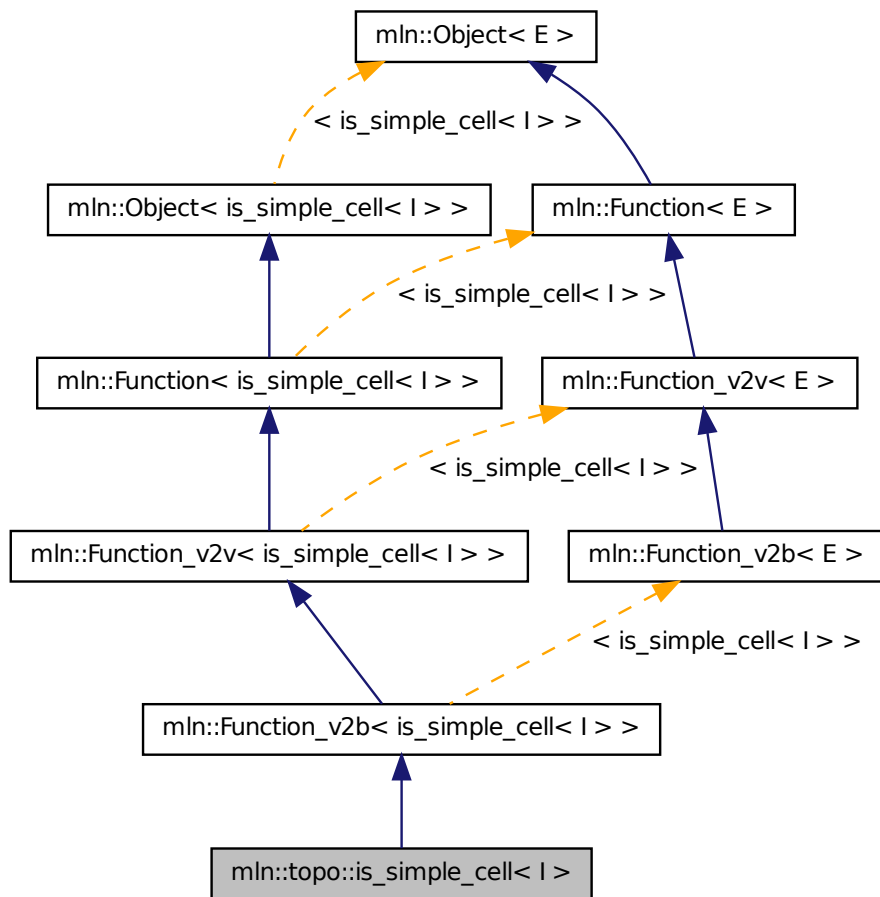
Definition at line 66 of file is\_simple\_2d.hh.

## 10.348 mln::topo::is\_simple\_cell< I > Class Template Reference

A predicate for the simplicity of a point based on the collapse property of the attachment.

```
#include <is_simple_cell.hh>
```

Inheritance diagram for mln::topo::is\_simple\_cell< I >:



## Public Types

- typedef `mln::complex_psite< D, G > psite`  
*Psite type.*
- typedef `bool result`  
*Result type of the functor.*

## Public Member Functions

- typedef `mln_geom (I) G`  
*Geometry of the image.*

- bool `operator()` (const `mln::complex_psite`< I::dim, `mln_geom`(I)> &p) const  
*Based on the algorithm A2 from couprie.08.pami.*

- void `set_image` (const `mln::Image`< I > &ima)  
*Set the underlying image.*

## Static Public Attributes

- static const unsigned `D` = I::dim  
*Dimension of the image (and therefore of the complex).*

### 10.348.1 Detailed Description

`template<typename I> class mln::topo::is_simple_cell< I >`

A predicate for the simplicity of a point based on the collapse property of the attachment. The functor does not actually take a cell as input, but a face that is expected to be a D-facet.

Definition at line 57 of file `is_simple_cell.hh`.

### 10.348.2 Member Typedef Documentation

**10.348.2.1** `template<typename I > typedef mln::complex_psite<D, G>  
mln::topo::is_simple_cell< I >::psite`

Psite type.

Definition at line 65 of file `is_simple_cell.hh`.

**10.348.2.2** `template<typename I > typedef bool mln::topo::is_simple_cell< I >::result`

Result type of the functor.

Reimplemented from `mln::Function_v2b< is_simple_cell< I > >`.

Definition at line 68 of file `is_simple_cell.hh`.

### 10.348.3 Member Function Documentation

**10.348.3.1** `template<typename I > typedef mln::topo::is_simple_cell< I >::mln_geom ( I )`

Geometry of the image.

**10.348.3.2** `template<typename I > bool mln::topo::is_simple_cell< I >::operator() ( const  
mln::complex_psite< I::dim, mln_geom(I)> & p ) const [inline]`

Based on the algorithm A2 from couprie.08.pami.

Definition at line 115 of file `is_simple_cell.hh`.



References mln::make::attachment().

**10.348.3.3** `template<typename I > void mln::topo::is_simple_cell< I >::set_image ( const mln::Image< I > & ima ) [inline]`

Set the underlying image.

Definition at line 107 of file is\_simple\_cell.hh.

## 10.348.4 Member Data Documentation

**10.348.4.1** `template<typename I > const unsigned mln::topo::is_simple_cell< I >::D = I::dim [static]`

Dimension of the image (and therefore of the complex).

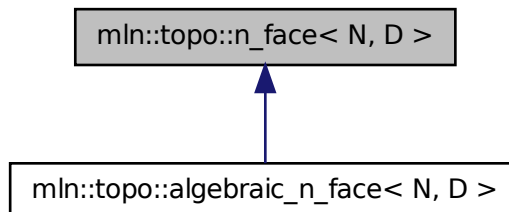
Definition at line 61 of file is\_simple\_cell.hh.

## 10.349 mln::topo::n\_face< N, D > Class Template Reference

N-face handle in a complex.

```
#include <n_face.hh>
```

Inheritance diagram for mln::topo::n\_face< N, D >:



## Public Member Functions

- void `invalidate ()`  
*Invalidate this handle.*
- bool `is_valid () const`  
*Is this handle valid?*
- `n_face ()`  
*Build a non-initialized face handle.*

- [n\\_face](#) ([complex](#)< D > &[complex](#), unsigned face\_id)  
*Build a face handle from complex and face\_id.*
- [complex](#)< D > [cplx](#) () const  
*Accessors.*
- unsigned [face\\_id](#) () const  
*Return the id of the face.*
- void [set\\_cplx](#) (const [complex](#)< D > &cplx)  
*Set the complex the face belongs to.*
- unsigned [n](#) () const  
*Return the dimension of the face.*
- void [set\\_face\\_id](#) (unsigned face\_id)  
*Set the id of the face.*
- void [inc\\_face\\_id](#) ()  
*Increment the id of the face.*
- void [dec\\_face\\_id](#) ()  
*Decrement the id of the face.*
- [face\\_data](#)< N, D > & [data](#) () const  
*Return the mln::topo::face\_data pointed by this handle.*
- std::vector< [algebraic\\_n\\_face](#)< N-1, D > > [lower\\_dim\\_adj\\_faces](#) () const  
*Return an array of face handles pointing to adjacent (n-1)-faces.*
- std::vector< [algebraic\\_n\\_face](#)< N+1, D > > [higher\\_dim\\_adj\\_faces](#) () const  
*Return an array of face handles pointing to adjacent (n+1)-faces.*

### 10.349.1 Detailed Description

**template<unsigned N, unsigned D> class mln::topo::n\_face< N, D >**

N-face handle in a complex. Contrary to an [mln::topo::face](#), the dimension of an [mln::topo::n\\_face](#) is fixed.

Definition at line 61 of file n\_face.hh.

### 10.349.2 Constructor & Destructor Documentation

**10.349.2.1 template<unsigned N, unsigned D> mln::topo::n\_face< N, D >::n\_face ( )**  
**[inline]**

Build a non-initialized face handle.

Definition at line 159 of file n\_face.hh.

References [mln::topo::n\\_face< N, D >::is\\_valid\(\)](#).

**10.349.2.2** `template<unsigned N, unsigned D> mln::topo::n_face< N, D >::n_face ( complex< D > & complex, unsigned face_id ) [inline]`

Build a face handle from *complex* and *face\_id*.

Definition at line 169 of file n\_face.hh.

### 10.349.3 Member Function Documentation

**10.349.3.1** `template<unsigned N, unsigned D> complex< D > mln::topo::n_face< N, D >::cplx ( ) const [inline]`

Accessors.

Return the complex the face belongs to.

Definition at line 195 of file n\_face.hh.

Referenced by `mln::topo::n_faces_set< N, D >::add()`, `mln::topo::operator!=()`, and `mln::topo::operator==()`.

**10.349.3.2** `template<unsigned N, unsigned D> face_data< N, D > & mln::topo::n_face< N, D >::data ( ) const [inline]`

Return the `mln::topo::face_data` pointed by this handle.

Definition at line 251 of file n\_face.hh.

References `mln::topo::n_face< N, D >::is_valid()`.

**10.349.3.3** `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::dec_face_id ( ) [inline]`

Decrement the id of the face.

Definition at line 243 of file n\_face.hh.

**10.349.3.4** `template<unsigned N, unsigned D> unsigned mln::topo::n_face< N, D >::face_id ( ) const [inline]`

Return the id of the face.

Definition at line 211 of file n\_face.hh.

Referenced by `mln::topo::operator==()`.

**10.349.3.5** `template<unsigned N, unsigned D> std::vector< algebraic_n_face< N+1, D > > mln::topo::n_face< N, D >::higher_dim_adj_faces ( ) const [inline]`

Return an array of face handles pointing to adjacent (n+1)-faces.

Definition at line 270 of file n\_face.hh.

References `mln::topo::n_face< N, D >::is_valid()`.

Referenced by `mln::topo::edge()`.

**10.349.3.6** `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::inc_face_id ( )`  
`[inline]`

Increment the id of the face.

Definition at line 235 of file n\_face.hh.

**10.349.3.7** `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::invalidate ( )`  
`[inline]`

Invalidate this handle.

Definition at line 187 of file n\_face.hh.

References `mln::topo::n_face< N, D >::set_face_id()`.

**10.349.3.8** `template<unsigned N, unsigned D> bool mln::topo::n_face< N, D >::is_valid ( )`  
`const [inline]`

Is this handle valid?

Definition at line 179 of file n\_face.hh.

Referenced by `mln::topo::algebraic_n_face< N, D >::algebraic_n_face()`, `mln::topo::n_face< N, D >::data()`, `mln::topo::n_face< N, D >::higher_dim_adj_faces()`, `mln::topo::n_face< N, D >::lower_dim_adj_faces()`, and `mln::topo::n_face< N, D >::n_face()`.

**10.349.3.9** `template<unsigned N, unsigned D> std::vector< algebraic_n_face< N-1, D > >`  
`mln::topo::n_face< N, D >::lower_dim_adj_faces ( ) const [inline]`

Return an array of face handles pointing to adjacent (n-1)-faces.

Definition at line 260 of file n\_face.hh.

References `mln::topo::n_face< N, D >::is_valid()`.

**10.349.3.10** `template<unsigned N, unsigned D> unsigned mln::topo::n_face< N, D >::n ( )`  
`const [inline]`

Return the dimension of the face.

Definition at line 203 of file n\_face.hh.

**10.349.3.11** `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::set_cplx ( const`  
`complex< D > & cplx ) [inline]`

Set the complex the face belongs to.

Definition at line 219 of file n\_face.hh.

**10.349.3.12** `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::set_face_id (`  
`unsigned face_id ) [inline]`

Set the id of the face.

Definition at line 227 of file n\_face.hh.

Referenced by mln::topo::n\_face< N, D >::invalidate().

## 10.350 mln::topo::n\_face\_bkd\_iter< D > Class Template Reference

Backward iterator on all the faces of an mln::complex<D>.

```
#include <n_face_iter.hh>
```

Inherits complex\_set\_iterator\_base< topo::face< D >, n\_face\_bkd\_iter< D > >.

### Public Member Functions

- void [next](#) ()  
*Go to the next element.*
- [n\\_face\\_bkd\\_iter](#) ()  
*Construction and assignment.*
- void [start](#) ()  
*Manipulation.*
- unsigned [n](#) () const  
*Accessors.*

### 10.350.1 Detailed Description

```
template<unsigned D> class mln::topo::n_face_bkd_iter< D >
```

Backward iterator on all the faces of an mln::complex<D>.

#### Template Parameters

*D* The dimension of the complex this iterator belongs to.

Definition at line 127 of file n\_face\_iter.hh.

### 10.350.2 Constructor & Destructor Documentation

```
10.350.2.1 template<unsigned D> mln::topo::n_face_bkd_iter< D >::n_face_bkd_iter ( )  
[inline]
```

Construction and assignment.

Definition at line 256 of file n\_face\_iter.hh.

### 10.350.3 Member Function Documentation

**10.350.3.1** `template<unsigned D> unsigned mln::topo::n_face_bkd_iter< D >::n ( ) const`  
**[inline]**

Accessors.

Shortcuts to face\_'s accessors.

Definition at line 308 of file n\_face\_iter.hh.

Referenced by `mln::topo::n_face_bkd_iter< D >::start()`.

**10.350.3.2** `void mln::Iterator< n_face_bkd_iter< D > >::next ( )` **[inherited]**

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

#### Precondition

The iterator is valid.

**10.350.3.3** `template<unsigned D> void mln::topo::n_face_bkd_iter< D >::start ( )` **[inline]**

Manipulation.

Start an iteration.

Definition at line 275 of file n\_face\_iter.hh.

References `mln::topo::n_face_bkd_iter< D >::n()`.

## 10.351 mln::topo::n\_face\_fwd\_iter< D > Class Template Reference

Forward iterator on all the faces of an `mln::complex<D>`.

```
#include <n_face_iter.hh>
```

Inherits `complex_set_iterator_base< topo::face< D >, n_face_fwd_iter< D > >`.

### Public Member Functions

- `void next ()`  
*Go to the next element.*
- `n_face_fwd_iter ()`  
*Construction and assignment.*

- void `start()`  
*Manipulation.*
- unsigned `n()` const  
*Accessors.*

### 10.351.1 Detailed Description

`template<unsigned D> class mln::topo::n_face_fwd_iter< D >`

Forward iterator on all the faces of an `mln::complex<D>`.

#### Template Parameters

*D* The dimension of the complex this iterator belongs to.

Definition at line 72 of file `n_face_iter.hh`.

### 10.351.2 Constructor & Destructor Documentation

**10.351.2.1** `template<unsigned D> mln::topo::n_face_fwd_iter< D >::n_face_fwd_iter ( )`  
`[inline]`

Construction and assignment.

Definition at line 183 of file `n_face_iter.hh`.

### 10.351.3 Member Function Documentation

**10.351.3.1** `template<unsigned D> unsigned mln::topo::n_face_fwd_iter< D >::n ( ) const`  
`[inline]`

Accessors.

Shortcuts to `face_`'s accessors.

Definition at line 235 of file `n_face_iter.hh`.

**10.351.3.2** `void mln::Iterator< n_face_fwd_iter< D > >::next ( )` `[inherited]`

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

#### Precondition

The iterator is valid.

**10.351.3.3** `template<unsigned D> void mln::topo::n_face_fwd_iter< D >::start ( ) [inline]`

Manipulation.

Test if the iterator is valid.

Definition at line 202 of file `n_face_iter.hh`.

**10.352** `mln::topo::n_faces_set< N, D >` Class Template Reference

Set of face handles of dimension N.

```
#include <n_faces_set.hh>
```

**Public Types**

- typedef `std::vector< algebraic_n_face< N, D > >` `faces_type`

*The type of the set of face handles.*

**Public Member Functions**

- void `add` (const `algebraic_n_face< N, D >` &f)

*Append an algebraic face f to the set.*

- void `reserve` (size\_t n)

*Reserve n cells in the set.*

- const `faces_type` & `faces` () const

*Accessors.*

**10.352.1** Detailed Description

```
template<unsigned N, unsigned D> class mln::topo::n_faces_set< N, D >
```

Set of face handles of dimension N.

Definition at line 56 of file `n_faces_set.hh`.

**10.352.2** Member Typedef Documentation

**10.352.2.1** `template<unsigned N, unsigned D> typedef std::vector< algebraic_n_face<N, D> > mln::topo::n_faces_set< N, D >::faces_type`

The type of the set of face handles.

Definition at line 70 of file `n_faces_set.hh`.



### 10.352.3 Member Function Documentation

**10.352.3.1** `template<unsigned N, unsigned D> void mln::topo::n_faces_set< N, D >::add ( const algebraic_n_face< N, D > & f ) [inline]`

Append an algebraic face  $f$  to the set.

Definition at line 171 of file n\_faces\_set.hh.

References mln::topo::n\_face< N, D >::cplx().

Referenced by mln::topo::operator+(), and mln::topo::operator-().

**10.352.3.2** `template<unsigned N, unsigned D> const std::vector< algebraic_n_face< N, D > > & mln::topo::n_faces_set< N, D >::faces ( ) const [inline]`

Accessors.

Return the set of handles.

Definition at line 190 of file n\_faces\_set.hh.

Referenced by mln::topo::complex< D >::add\_face().

**10.352.3.3** `template<unsigned N, unsigned D> void mln::topo::n_faces_set< N, D >::reserve ( size_t n ) [inline]`

Reserve  $n$  cells in the set.

This methods does not change the content of `faces_;` it only pre-allocate memory. Method reserve is provided for efficiency purpose, and its use is completely optional.

Definition at line 182 of file n\_faces\_set.hh.

## 10.353 mln::topo::skeleton::is\_simple\_point< N > Struct Template Reference

```
#include <is_simple_point.hh>
```

### 10.353.1 Detailed Description

`template<typename N> struct mln::topo::skeleton::is_simple_point< N >`

Tell if a point is simple or not. A point of an object is simple if in its  $c_8$  neighborhood, there is exactly one connected component of the object, and only one connected component of the background Examples : ( | == object, - = background)

```
- - |
| P | Here p is simple in the c4 and c8 case.
| | |

- | -
| P | Here p is never simple.
| | |
```

Definition at line 68 of file `is_simple_point.hh`.

## 10.354 `mln::topo::static_n_face_bkd_iter< N, D >` Class Template Reference

Backward iterator on all the `N`-faces of a `mln::complex<D>`.

```
#include <static_n_face_iter.hh>
```

Inherits `complex_set_iterator_base< topo::face< D >, static_n_face_bkd_iter< N, D > >`.

### Public Member Functions

- void `next()`  
*Go to the next element.*
- `static_n_face_bkd_iter()`  
*Construction and assignment.*
- void `start()`  
*Manipulation.*

### 10.354.1 Detailed Description

```
template<unsigned N, unsigned D> class mln::topo::static_n_face_bkd_iter< N, D >
```

Backward iterator on all the `N`-faces of a `mln::complex<D>`.

#### Template Parameters

- N* The dimension of the face associated to this iterator.
- D* The dimension of the complex this iterator belongs to.

Definition at line 101 of file `static_n_face_iter.hh`.

### 10.354.2 Constructor & Destructor Documentation

```
10.354.2.1 template<unsigned N, unsigned D> mln::topo::static_n_face_bkd_iter< N, D >::static_n_face_bkd_iter ( ) [inline]
```

Construction and assignment.

Definition at line 194 of file `static_n_face_iter.hh`.

### 10.354.3 Member Function Documentation

#### 10.354.3.1 void mln::Iterator< static\_n\_face\_bkd\_iter< N, D > >::next ( ) [inherited]

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-define this method. The actual "next" operation has to be defined through the *next\_* method.

#### Precondition

The iterator is valid.

#### 10.354.3.2 template<unsigned N, unsigned D> void mln::topo::static\_n\_face\_bkd\_iter< N, D >::start ( ) [inline]

Manipulation.

Start an iteration.

Definition at line 217 of file static\_n\_face\_iter.hh.

## 10.355 mln::topo::static\_n\_face\_fwd\_iter< N, D > Class Template Reference

Forward iterator on all the *N*-faces of a mln::complex<D>.

```
#include <static_n_face_iter.hh>
```

Inherits complex\_set\_iterator\_base< topo::face< D >, static\_n\_face\_fwd\_iter< N, D > >.

### Public Member Functions

- void [next](#) ()  
*Go to the next element.*
- [static\\_n\\_face\\_fwd\\_iter](#) ()  
*Construction and assignment.*
- void [start](#) ()  
*Manipulation.*

### 10.355.1 Detailed Description

```
template<unsigned N, unsigned D> class mln::topo::static_n_face_fwd_iter< N, D >
```

Forward iterator on all the *N*-faces of a mln::complex<D>.

**Template Parameters**

- N* The dimension of the face associated to this iterator.
- D* The dimension of the complex this iterator belongs to.

Definition at line 55 of file `static_n_face_iter.hh`.

**10.355.2 Constructor & Destructor Documentation**

**10.355.2.1** `template<unsigned N, unsigned D> mln::topo::static_n_face_fwd_iter< N, D >::static_n_face_fwd_iter ( ) [inline]`

Construction and assignment.

Definition at line 145 of file `static_n_face_iter.hh`.

**10.355.3 Member Function Documentation**

**10.355.3.1** `void mln::Iterator< static_n_face_fwd_iter< N, D > >::next ( ) [inherited]`

Go to the next element.

**Warning**

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

**Precondition**

The iterator is valid.

**10.355.3.2** `template<unsigned N, unsigned D> void mln::topo::static_n_face_fwd_iter< N, D >::start ( ) [inline]`

Manipulation.

Test if the iterator is valid.

Definition at line 168 of file `static_n_face_iter.hh`.

**10.356 mln::tr\_image< S, I, T > Struct Template Reference**

Transform an image by a given transformation.

```
#include <tr_image.hh>
```

Inherits `image_identity< I, S, tr_image< S, I, T > >`.

**Public Types**

- typedef `I::value` `lvalue`

*Return type of read-write access.*

- typedef I::psite [psite](#)  
*Point\_Site associated type.*
- typedef I::value [rvalue](#)  
*Return type of read-only access.*
- typedef I::site [site](#)  
*Site associated type.*
- typedef [tr\\_image](#)< S, tag::image\_< I >, T > [skeleton](#)  
*Skeleton.*
- typedef I::value [value](#)  
*Value associated type.*

## Public Member Functions

- const S & [domain](#) () const  
*Return the domain morpher.*
- bool [has](#) (const vec\_t &v) const  
*Test if a pixel value is accessible at v.*
- bool [is\\_valid](#) () const  
*Test if this image has been initialized.*
- I::value [operator\(\)](#) (const [psite](#) &p) const  
*Read-only access of pixel value at point site p.*
- void [set\\_tr](#) (T &tr)  
*Set the transformation.*
- const T & [tr](#) () const  
*Return the underlying transformation.*
- [tr\\_image](#) (const S &s, const I &ima, const T &tr)  
*Constructors.*

### 10.356.1 Detailed Description

**template**<typename S, typename I, typename T> **struct** mln::tr\_image< S, I, T >

Transform an image by a given transformation.

Definition at line 83 of file tr\_image.hh.

## 10.356.2 Member Typedef Documentation

### 10.356.2.1 `template<typename S, typename I, typename T> typedef I ::value mln::tr_image< S, I, T >::lvalue`

Return type of read-write access.

Definition at line 101 of file `tr_image.hh`.

### 10.356.2.2 `template<typename S, typename I, typename T> typedef I ::psite mln::tr_image< S, I, T >::psite`

[Point\\_Site](#) associated type.

Definition at line 92 of file `tr_image.hh`.

### 10.356.2.3 `template<typename S, typename I, typename T> typedef I ::value mln::tr_image< S, I, T >::rvalue`

Return type of read-only access.

Definition at line 104 of file `tr_image.hh`.

### 10.356.2.4 `template<typename S, typename I, typename T> typedef I ::site mln::tr_image< S, I, T >::site`

[Site](#) associated type.

Definition at line 95 of file `tr_image.hh`.

### 10.356.2.5 `template<typename S, typename I, typename T> typedef tr_image< S, tag::image_<I>, T> mln::tr_image< S, I, T >::skeleton`

Skeleton.

Definition at line 107 of file `tr_image.hh`.

### 10.356.2.6 `template<typename S, typename I, typename T> typedef I ::value mln::tr_image< S, I, T >::value`

[Value](#) associated type.

Definition at line 98 of file `tr_image.hh`.

## 10.356.3 Constructor & Destructor Documentation

### 10.356.3.1 `template<typename S , typename I , typename T > mln::tr_image< S, I, T >::tr_image ( const S & s, const I & ima, const T & tr ) [inline]`

Constructors.

Definition at line 174 of file `tr_image.hh`.

### 10.356.4 Member Function Documentation

**10.356.4.1** `template<typename S, typename I, typename T > const S & mln::tr_image< S, I, T >::domain( ) const [inline]`

Return the domain morpher.

Definition at line 247 of file tr\_image.hh.

**10.356.4.2** `template<typename S, typename I, typename T > bool mln::tr_image< S, I, T >::has( const vec_t & v ) const [inline]`

Test if a pixel value is accessible at v.

Definition at line 200 of file tr\_image.hh.

**10.356.4.3** `template<typename S, typename I, typename T > bool mln::tr_image< S, I, T >::is_valid( ) const [inline]`

Test if this image has been initialized.

Definition at line 191 of file tr\_image.hh.

**10.356.4.4** `template<typename S, typename I, typename T > I::value mln::tr_image< S, I, T >::operator()( const psite & p ) const [inline]`

Read-only access of pixel value at point site p.

Mutable access is only OK for reading (not writing).

Definition at line 213 of file tr\_image.hh.

**10.356.4.5** `template<typename S, typename I, typename T > void mln::tr_image< S, I, T >::set_tr( T & tr ) [inline]`

Set the transformation.

Definition at line 231 of file tr\_image.hh.

**10.356.4.6** `template<typename S, typename I, typename T > const T & mln::tr_image< S, I, T >::tr( ) const [inline]`

Return the underlying transformation.

Definition at line 239 of file tr\_image.hh.

## 10.357 mln::unproject\_image< I, D, F > Struct Template Reference

Un-projects an image.

```
#include <unproject_image.hh>
```

Inherits image\_domain\_morpher< I, D, unproject\_image< I, D, F > >.

## Public Member Functions

- `const D & domain () const`  
*Give the definition domain.*
- `internal::morpher_lvalue_< I >::ret operator() (const typename D::psite &p)`  
*Read-write access to the image value located at point  $p$ .*
- `I::rvalue operator() (const typename D::psite &p) const`  
*Read-only access to the image value located at point  $p$ .*
- `unproject_image ()`  
*Constructor without argument.*
- `unproject_image (I &ima, const D &dom, const F &f)`  
*Constructor from an image  $ima$ , a domain  $dom$ , and a function  $f$ .*

### 10.357.1 Detailed Description

`template<typename I, typename D, typename F> struct mln::unproject_image< I, D, F >`

Un-projects an image.

Definition at line 96 of file `unproject_image.hh`.

### 10.357.2 Constructor & Destructor Documentation

**10.357.2.1** `template<typename I, typename D, typename F > mln::unproject_image< I, D, F >::unproject_image ( ) [inline]`

Constructor without argument.

Definition at line 182 of file `unproject_image.hh`.

**10.357.2.2** `template<typename I, typename D, typename F > mln::unproject_image< I, D, F >::unproject_image ( I & ima, const D & dom, const F & f ) [inline]`

Constructor from an image `ima`, a domain `dom`, and a function `f`.

Definition at line 188 of file `unproject_image.hh`.

### 10.357.3 Member Function Documentation

**10.357.3.1** `template<typename I, typename D, typename F > const D & mln::unproject_image< I, D, F >::domain ( ) const [inline]`

Give the definition domain.

Definition at line 205 of file `unproject_image.hh`.



**10.357.3.2** `template<typename I, typename D, typename F > internal::morpher_lvalue_< I >::ret mln::unproject_image< I, D, F >::operator() ( const typename D::psite & p ) [inline]`

Read-write access to the image value located at point p.

Definition at line 225 of file unproject\_image.hh.

**10.357.3.3** `template<typename I, typename D, typename F > I::rvalue mln::unproject_image< I, D, F >::operator() ( const typename D::psite & p ) const [inline]`

Read-only access to the image value located at point p.

Definition at line 214 of file unproject\_image.hh.

## 10.358 mln::util::adjacency\_matrix< V > Class Template Reference

A class of adjacency matrix.

```
#include <adjacency_matrix.hh>
```

Inherits adjacency\_matrix\_impl\_selector< V, mln::metal::equal< mln::trait::value\_< V >::quant, trait::value::quant::low >::eval >.

### Public Member Functions

- [adjacency\\_matrix \(\)](#)  
*Constructors.*
- [adjacency\\_matrix \(const V &elements\)](#)  
*Construct an adjacency matrix with `n` elements elements maximum.*

### 10.358.1 Detailed Description

```
template<typename V = def::coord> class mln::util::adjacency_matrix< V >
```

A class of adjacency matrix. Support low and high quantification value types. In case of low quantification value type, it uses an [image2d](#) to store adjacency information. In case of high quantification value type, it uses a [util::set](#) to store the adjacency information.

Definition at line 136 of file adjacency\_matrix.hh.

### 10.358.2 Constructor & Destructor Documentation

**10.358.2.1** `template<typename V > mln::util::adjacency_matrix< V >::adjacency_matrix ( )`

Constructors.

```
@{
```

Default

Definition at line 308 of file adjacency\_matrix.hh.

**10.358.2.2** `template<typename V > mln::util::adjacency_matrix< V >::adjacency_matrix ( const V & nelements )`

Construct an adjacency matrix with `nelements` elements maximum.

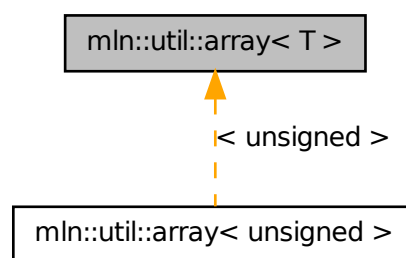
Definition at line 315 of file adjacency\_matrix.hh.

## 10.359 `mln::util::array< T >` Class Template Reference

A dynamic array class.

```
#include <array.hh>
```

Inheritance diagram for `mln::util::array< T >`:



### Public Types

- typedef T [element](#)  
*Element associated type.*
- typedef T [result](#)  
*Returned value types.*
- typedef `array_fwd_iter< T >` [fwd\\_eiter](#)  
*Iterator types*  
*Forward iterator associated type.*
- typedef `array_bkd_iter< T >` [bkd\\_eiter](#)  
*Backward iterator associated type.*

- typedef `fwd_eiter eiter`  
*Iterator associated type.*

## Public Member Functions

- `array< T > & append` (const T &elt)  
*Add the element elt at the end of this array.*
- `template<typename U > array< T > & append` (const `array< U > &other`)  
*Add the elements of other at the end of this array.*
- `void clear` ()  
*Empty the array.*
- `void fill` (const T &value)  
*Fill the whole array with value value.*
- `bool is_empty` () const  
*Test if the array is empty.*
- `ro_result last` () const  
*Return the last element.*
- `mutable_result last` ()  
*Return the last element.*
- `std::size_t memory_size` () const  
*Return the size of this array in memory.*
- `unsigned nelements` () const  
*Return the number of elements of the array.*
- `ro_result operator()` (unsigned i) const  
*Return the i-th element of the array.*
- `mutable_result operator()` (unsigned i)  
*Return the i-th element of the array.*
- `ro_result operator[ ]` (unsigned i) const  
*Return the i-th element of the array.*
- `mutable_result operator[ ]` (unsigned i)  
*Return the i-th element of the array.*
- `void reserve` (unsigned n)  
*Reserve memory for n elements.*
- `void resize` (unsigned n)

*Resize this array to  $n$  elements.*

- void [resize](#) (unsigned  $n$ , const  $T$  &value)  
*Resize this array to  $n$  elements with `value` as value.*
- unsigned [size](#) () const  
*Return the number of elements of the array.*
- const `std::vector< T >` & [std\\_vector](#) () const  
*Return the corresponding `std::vector` of elements.*
- [array](#) ()  
*Constructors  
Constructor without arguments.*
- [array](#) (unsigned  $n$ )  
*Construct a new array and resize it to elements.*
- [array](#) (unsigned  $n$ , const  $T$  &value)  
*Construct a new array, resize it to elements and fill it with `default_value`.*

### 10.359.1 Detailed Description

**template<typename T> class mln::util::array< T >**

A dynamic array class. Elements are stored by copy. Implementation is lazy.

The parameter  $T$  is the element type, which shall not be const-qualified.

Definition at line 99 of file util/array.hh.

### 10.359.2 Member Typedef Documentation

**10.359.2.1 template<typename T> typedef array\_bkd\_iter<T> mln::util::array< T >::bkd\_eiter**

Backward iterator associated type.

Definition at line 124 of file util/array.hh.

**10.359.2.2 template<typename T> typedef fwd\_eiter mln::util::array< T >::eiter**

[Iterator](#) associated type.

Definition at line 127 of file util/array.hh.

**10.359.2.3** `template<typename T> typedef T mln::util::array< T >::element`

Element associated type.

Definition at line 107 of file util/array.hh.

**10.359.2.4** `template<typename T> typedef array_fwd_iter<T> mln::util::array< T >::fwd_eiter`

[Iterator](#) types

Forward iterator associated type.

Definition at line 121 of file util/array.hh.

**10.359.2.5** `template<typename T> typedef T mln::util::array< T >::result`

Returned value types.

Related to the [Function\\_v2v](#) concept.

Definition at line 112 of file util/array.hh.

**10.359.3** **Constructor & Destructor Documentation****10.359.3.1** `template<typename T> mln::util::array< T >::array ( ) [inline]`

Constructors

Constructor without arguments.

Definition at line 427 of file util/array.hh.

**10.359.3.2** `template<typename T> mln::util::array< T >::array ( unsigned n ) [inline]`

Construct a new array and resize it to elements.

Definition at line 433 of file util/array.hh.

**10.359.3.3** `template<typename T> mln::util::array< T >::array ( unsigned n, const T & value ) [inline]`

Construct a new array, resize it to elements and fill it with `default_value`.

Definition at line 440 of file util/array.hh.

**10.359.4** **Member Function Documentation****10.359.4.1** `template<typename T> array< T > & mln::util::array< T >::append ( const T & elt ) [inline]`

Add the element `elt` at the end of this array.

Definition at line 472 of file util/array.hh.

Referenced by `mln::io::dicom::get_header()`, and `mln::io::plot::load()`.

**10.359.4.2** `template<typename T> template<typename U> array< T> & mln::util::array< T>::append ( const array< U> & other ) [inline]`

Add the elements of `other` at the end of this array.

Definition at line 482 of file util/array.hh.

References `mln::util::array< T>::is_empty()`, and `mln::util::array< T>::std_vector()`.

**10.359.4.3** `template<typename T> void mln::util::array< T>::clear ( ) [inline]`

Empty the array.

All elements contained in the array are destroyed.

#### Postcondition

`is_empty() == true`

Definition at line 495 of file util/array.hh.

References `mln::util::array< T>::is_empty()`.

Referenced by `mln::io::plot::load()`.

**10.359.4.4** `template<typename T> void mln::util::array< T>::fill ( const T & value ) [inline]`

Fill the whole array with value `value`.

Definition at line 504 of file util/array.hh.

**10.359.4.5** `template<typename T> bool mln::util::array< T>::is_empty ( ) const [inline]`

Test if the array is empty.

Definition at line 578 of file util/array.hh.

References `mln::util::array< T>::nelements()`.

Referenced by `mln::util::array< T>::append()`, `mln::util::array< T>::clear()`, `mln::make::image3d()`, and `mln::io::pnms::load()`.

**10.359.4.6** `template<typename T> array< T>::ro_result mln::util::array< T>::last ( ) const [inline]`

Return the last element.

Definition at line 562 of file util/array.hh.

References `mln::util::array< T>::nelements()`.

**10.359.4.7** `template<typename T> array< T >::mutable_result mln::util::array< T >::last ( )`  
`[inline]`

Return the last element.

Definition at line 570 of file util/array.hh.

References `mln::util::array< T >::nelements()`.

**10.359.4.8** `template<typename T> std::size_t mln::util::array< T >::memory_size ( ) const`  
`[inline]`

Return the size of this array in memory.

Definition at line 602 of file util/array.hh.

References `mln::util::array< T >::nelements()`.

**10.359.4.9** `template<typename T> unsigned mln::util::array< T >::nelements ( ) const`  
`[inline]`

Return the number of elements of the array.

Definition at line 520 of file util/array.hh.

Referenced by `mln::labeling::fill_holes()`, `mln::make::image3d()`, `mln::util::array< T >::is_empty()`, `mln::util::array< T >::last()`, `mln::io::pnms::load()`, `mln::util::array< T >::memory_size()`, `mln::util::operator<<()`, `mln::util::array< T >::operator[]()`, `mln::io::plot::save()`, and `mln::util::array< T >::size()`.

**10.359.4.10** `template<typename T> array< T >::ro_result mln::util::array< T >::operator() ( unsigned i ) const`  
`[inline]`

Return the *i*-th element of the array.

**Precondition**

`i < nelements()`

Definition at line 528 of file util/array.hh.

**10.359.4.11** `template<typename T> array< T >::mutable_result mln::util::array< T >::operator() ( unsigned i )`  
`[inline]`

Return the *i*-th element of the array.

**Precondition**

`i < nelements()`

Definition at line 536 of file util/array.hh.

**10.359.4.12** `template<typename T> array< T >::ro_result mln::util::array< T >::operator[] ( unsigned i ) const [inline]`

Return the `i`-th element of the array.

**Precondition**

`i < nelements()`

Definition at line 544 of file `util/array.hh`.

References `mln::util::array< T >::nelements()`.

**10.359.4.13** `template<typename T> array< T >::mutable_result mln::util::array< T >::operator[] ( unsigned i ) [inline]`

Return the `i`-th element of the array.

**Precondition**

`i < nelements()`

Definition at line 553 of file `util/array.hh`.

References `mln::util::array< T >::nelements()`.

**10.359.4.14** `template<typename T> void mln::util::array< T >::reserve ( unsigned n ) [inline]`

Reserve memory for `n` elements.

Definition at line 448 of file `util/array.hh`.

**10.359.4.15** `template<typename T> void mln::util::array< T >::resize ( unsigned n ) [inline]`

Resize this array to `n` elements.

Definition at line 456 of file `util/array.hh`.

Referenced by `mln::labeling::impl::generic::compute()`, `mln::labeling::impl::compute_fastest()`, `mln::io::raw::get_header()`, and `mln::io::dump::get_header()`.

**10.359.4.16** `template<typename T> void mln::util::array< T >::resize ( unsigned n, const T & value ) [inline]`

Resize this array to `n` elements with `value` as value.

Definition at line 464 of file `util/array.hh`.

**10.359.4.17** `template<typename T> unsigned mln::util::array< T >::size ( ) const [inline]`

Return the number of elements of the array.

Added for compatibility with `fun::i2v::array`.



See also

[nelements](#)

Definition at line 512 of file util/array.hh.

References mln::util::array< T >::nelements().

Referenced by mln::labeling::impl::generic::compute(), mln::labeling::impl::compute\_fastest(), mln::value::lut\_vec< S, T >::lut\_vec(), and mln::labeled\_image\_base< I, E >::update\_data().

**10.359.4.18** `template<typename T> const std::vector< T > & mln::util::array< T >::std_vector ( ) const [inline]`

Return the corresponding std::vector of elements.

Definition at line 586 of file util/array.hh.

Referenced by mln::util::array< T >::append(), mln::value::lut\_vec< S, T >::lut\_vec(), and mln::util::operator==( ).

## 10.360 mln::util::branch< T > Class Template Reference

Class of generic branch.

```
#include <tree.hh>
```

### Public Member Functions

- [tree\\_node< T > & apex \( \)](#)  
*The getter of the apex.*
- [branch \(tree< T > &tree, tree\\_node< T > &apex\)](#)  
*Constructor.*
- [tree< T > & util\\_tree \( \)](#)  
*The getter of the tree.*

### 10.360.1 Detailed Description

```
template<typename T> class mln::util::branch< T >
```

Class of generic branch.

Definition at line 249 of file tree.hh.

### 10.360.2 Constructor & Destructor Documentation

**10.360.2.1** `template<typename T> mln::util::branch< T >::branch ( util::tree< T > & tree, util::tree_node< T > & apex ) [inline]`

Constructor.

**Parameters**

- [in] *tree* The tree of the branch.
- [in] *apex* The apex of the branch.

Definition at line 537 of file tree.hh.

**10.360.3 Member Function Documentation****10.360.3.1** `template<typename T > util::tree_node< T > & mln::util::branch< T >::apex ( )`  
[inline]

The getter of the appex.

**Returns**

The [tree\\_node](#) appex of the current branch.

Definition at line 548 of file tree.hh.

**10.360.3.2** `template<typename T > mln::util::tree< T > & mln::util::branch< T >::util_tree ( )`  
[inline]

The getter of the tree.

**Returns**

The tree of the current branch.

Definition at line 556 of file tree.hh.

**10.361 mln::util::branch\_iter< T > Class Template Reference**

Basic 2D image class.

```
#include <branch_iter.hh>
```

**Public Member Functions**

- unsigned [deepness](#) () const  
*Give how deep is the iterator in the branch.*
- void [invalidate](#) ()  
*Invalidate the iterator.*
- bool [is\\_valid](#) () const  
*Test the iterator validity.*
- void [next](#) ()  
*Go to the next point.*

- `operator util::tree_node< T > & () const`

*Conversion to node.*

- `void start ()`

*Start an iteration.*

### 10.361.1 Detailed Description

`template<typename T> class mln::util::branch_iter< T >`

Basic 2D image class. The parameter T is the type of node's data. `branch_iter` is used to pre-order walk a branch.

Definition at line 52 of file `branch_iter.hh`.

### 10.361.2 Member Function Documentation

**10.361.2.1** `template<typename T > unsigned mln::util::branch_iter< T >::deepness ( ) const`  
[`inline`]

Give how deep is the iterator in the branch.

Definition at line 119 of file `branch_iter.hh`.

References `mln::util::branch_iter< T >::is_valid()`, and `mln::util::tree_node< T >::parent()`.

**10.361.2.2** `template<typename T > void mln::util::branch_iter< T >::invalidate ( )`  
[`inline`]

Invalidate the iterator.

Definition at line 143 of file `branch_iter.hh`.

Referenced by `mln::util::branch_iter< T >::next()`.

**10.361.2.3** `template<typename T > bool mln::util::branch_iter< T >::is_valid ( ) const`  
[`inline`]

Test the iterator validity.

Definition at line 135 of file `branch_iter.hh`.

Referenced by `mln::util::branch_iter< T >::deepness()`.

**10.361.2.4** `template<typename T > void mln::util::branch_iter< T >::next ( )` [`inline`]

Go to the next point.

Definition at line 162 of file `branch_iter.hh`.

References `mln::util::branch_iter< T >::invalidate()`.

**10.361.2.5** `template<typename T> mln::util::branch_iter< T >::operator util::tree_node< T > & ( ) const [inline]`

Conversion to node.

Definition at line 101 of file `branch_iter.hh`.

**10.361.2.6** `template<typename T> void mln::util::branch_iter< T >::start ( ) [inline]`

Start an iteration.

Definition at line 152 of file `branch_iter.hh`.

## 10.362 `mln::util::branch_iter_ind< T >` Class Template Reference

Basic 2D image class.

```
#include <branch_iter_ind.hh>
```

### Public Member Functions

- unsigned `deepness` () const  
*Give how deep is the iterator in the branch.*
- void `invalidate` ()  
*Invalidate the iterator.*
- bool `is_valid` () const  
*Test the iterator validity.*
- void `next` ()  
*Go to the next point.*
- `operator util::tree_node< T > & ()` const  
*Conversion to node.*
- void `start` ()  
*Start an iteration.*

### 10.362.1 Detailed Description

`template<typename T> class mln::util::branch_iter_ind< T >`

Basic 2D image class. The parameter `T` is the type of node's data. `branch_iter_ind` is used to pre-order walk a branch.

Definition at line 66 of file `branch_iter_ind.hh`.

## 10.362.2 Member Function Documentation

**10.362.2.1** `template<typename T> unsigned mln::util::branch_iter_ind< T >::deepness ( ) const [inline]`

Give how deep is the iterator in the branch.

Definition at line 131 of file branch\_iter\_ind.hh.

References mln::util::branch\_iter\_ind< T >::is\_valid(), and mln::util::tree\_node< T >::parent().

**10.362.2.2** `template<typename T> void mln::util::branch_iter_ind< T >::invalidate ( ) [inline]`

Invalidate the iterator.

Definition at line 155 of file branch\_iter\_ind.hh.

Referenced by mln::util::branch\_iter\_ind< T >::next().

**10.362.2.3** `template<typename T> bool mln::util::branch_iter_ind< T >::is_valid ( ) const [inline]`

Test the iterator validity.

Definition at line 147 of file branch\_iter\_ind.hh.

Referenced by mln::util::branch\_iter\_ind< T >::deepness().

**10.362.2.4** `template<typename T> void mln::util::branch_iter_ind< T >::next ( ) [inline]`

Go to the next point.

Definition at line 174 of file branch\_iter\_ind.hh.

References mln::util::branch\_iter\_ind< T >::invalidate().

**10.362.2.5** `template<typename T> mln::util::branch_iter_ind< T >::operator util::tree_node< T > &( ) const [inline]`

Conversion to node.

Definition at line 113 of file branch\_iter\_ind.hh.

**10.362.2.6** `template<typename T> void mln::util::branch_iter_ind< T >::start ( ) [inline]`

Start an iteration.

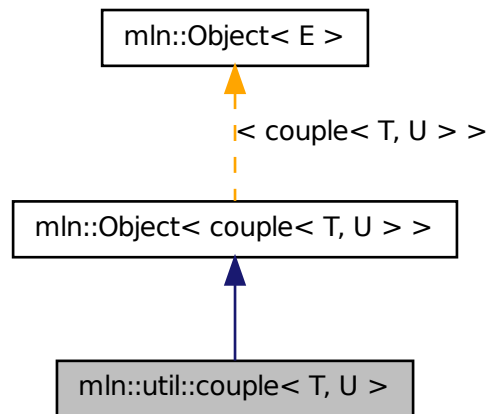
Definition at line 164 of file branch\_iter\_ind.hh.

## 10.363 mln::util::couple< T, U > Class Template Reference

Definition of a couple.

```
#include <couple.hh>
```

Inheritance diagram for `mln::util::couple< T, U >`:



## Public Member Functions

- void `change_both` (const T &first, const U &second)  
*Replace both members of the couple by val.*
- void `change_first` (const T &val)  
*Replace the first member of the couple by val.*
- void `change_second` (const U &val)  
*Replace the second member of the couple by val.*
- const T & `first` () const  
*Get the first member of the couple.*
- const U & `second` () const  
*Get the second member of the couple.*

### 10.363.1 Detailed Description

```
template<typename T, typename U> class mln::util::couple< T, U >
```

Definition of a couple.

Definition at line 48 of file util/couple.hh.

## 10.363.2 Member Function Documentation

**10.363.2.1** `template<typename T , typename U > void mln::util::couple< T, U >::change_both ( const T & first, const U & second ) [inline]`

Replace both members of the couple by *val*.

Definition at line 182 of file util/couple.hh.

**10.363.2.2** `template<typename T , typename U > void mln::util::couple< T, U >::change_first ( const T & val ) [inline]`

Replace the first member of the couple by *val*.

Definition at line 166 of file util/couple.hh.

**10.363.2.3** `template<typename T , typename U > void mln::util::couple< T, U >::change_second ( const U & val ) [inline]`

Replace the second member of the couple by *val*.

Definition at line 174 of file util/couple.hh.

**10.363.2.4** `template<typename T , typename U > const T & mln::util::couple< T, U >::first ( ) const [inline]`

Get the first member of the couple.

Definition at line 134 of file util/couple.hh.

**10.363.2.5** `template<typename T , typename U > const U & mln::util::couple< T, U >::second ( ) const [inline]`

Get the second member of the couple.

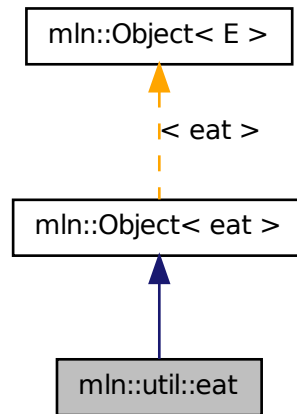
Definition at line 150 of file util/couple.hh.

## 10.364 mln::util::eat Struct Reference

Eat structure.

```
#include <eat.hh>
```

Inheritance diagram for mln::util::eat:



### 10.364.1 Detailed Description

Eat structure.

Definition at line 46 of file eat.hh.

## 10.365 mln::util::edge< G > Class Template Reference

[Edge](#) of a graph G.

```
#include <edge.hh>
```

Inherits mln::util::internal::edge\_impl\_< G >.

### Public Types

- typedef [Edge](#)< void > [category](#)  
*Object category.*
- typedef G [graph\\_t](#)  
*Graph associated type.*
- typedef [edge\\_id\\_t](#) [id\\_t](#)  
*The edge type id.*
- typedef [edge\\_id\\_t::value\\_t](#) [id\\_value\\_t](#)  
*The underlying type used to store edge ids.*



## Public Member Functions

- [edge](#) ()  
*Constructors.*
- [bool is\\_valid](#) () const  
*Misc.*
- [void invalidate](#) ()  
*Invalidate that vertex.*
- [edge\\_id\\_t id](#) () const  
*Return the edge id.*
- [void update\\_id](#) (const [edge\\_id\\_t](#) &id)  
*Set id\_ with id;*
- [operator edge\\_id\\_t](#) () const  
*Conversion to the edge id.*
- [const G & graph](#) () const  
*Return a reference to the graph holding this edge.*
- [void change\\_graph](#) (const G &g)  
*Set g\_ with g;*
- [vertex\\_id\\_t v\\_other](#) (const [vertex\\_id\\_t](#) &id\_v) const  
*Vertex and edges oriented.*
- [vertex\\_id\\_t v1](#) () const  
*Edge oriented.*
- [vertex\\_id\\_t v2](#) () const  
*Return the highest vertex id adjacent to this edge.*
- [size\\_t nmax\\_nbh\\_edges](#) () const  
*Return the number max of adjacent edges.*
- [edge\\_id\\_t ith\\_nbh\\_edge](#) (unsigned i) const  
*Return the i th adjacent edge.*

### 10.365.1 Detailed Description

`template<typename G> class mln::util::edge< G >`

[Edge](#) of a graph G.

Definition at line 69 of file edge.hh.

## 10.365.2 Member Typedef Documentation

### 10.365.2.1 `template<typename G> typedef Edge<void> mln::util::edge< G >::category`

[Object](#) category.

Definition at line 73 of file edge.hh.

### 10.365.2.2 `template<typename G> typedef G mln::util::edge< G >::graph_t`

[Graph](#) associated type.

Definition at line 82 of file edge.hh.

### 10.365.2.3 `template<typename G> typedef edge_id_t mln::util::edge< G >::id_t`

The edge type id.

Definition at line 79 of file edge.hh.

### 10.365.2.4 `template<typename G> typedef edge_id_t::value_t mln::util::edge< G >::id_value_t`

The underlying type used to store edge ids.

Definition at line 76 of file edge.hh.

## 10.365.3 Constructor & Destructor Documentation

### 10.365.3.1 `template<typename G > mln::util::edge< G >::edge ( ) [inline]`

Constructors.

Definition at line 227 of file edge.hh.

References `mln::util::edge< G >::invalidate()`.

## 10.365.4 Member Function Documentation

### 10.365.4.1 `template<typename G > void mln::util::edge< G >::change_graph ( const G & g ) [inline]`

Set `g_` with `g`;

Definition at line 290 of file edge.hh.

### 10.365.4.2 `template<typename G > const G & mln::util::edge< G >::graph ( ) const [inline]`

Return a reference to the graph holding this edge.

Definition at line 282 of file edge.hh.

Referenced by `mln::p_edges< G, F >::has()`, and `mln::util::line_graph< G >::has()`.

**10.365.4.3** `template<typename G> edge_id_t mln::util::edge< G >::id ( ) const [inline]`

Return the edge id.

Definition at line 259 of file edge.hh.

Referenced by mln::util::line\_graph< G >::has().

**10.365.4.4** `template<typename G> void mln::util::edge< G >::invalidate ( ) [inline]`

Invalidate that vertex.

Definition at line 306 of file edge.hh.

Referenced by mln::util::edge< G >::edge().

**10.365.4.5** `template<typename G> bool mln::util::edge< G >::is_valid ( ) const [inline]`

Misc.

Return whether is points to a known edge.

Definition at line 298 of file edge.hh.

Referenced by mln::p\_edges< G, F >::has().

**10.365.4.6** `template<typename G> edge_id_t mln::util::edge< G >::ith_nbh_edge ( unsigned i ) const [inline]`

Return the *i* th adjacent edge.

Definition at line 351 of file edge.hh.

**10.365.4.7** `template<typename G> size_t mln::util::edge< G >::nmax_nbh_edges ( ) const [inline]`

Return the number max of adjacent edges.

Definition at line 342 of file edge.hh.

**10.365.4.8** `template<typename G> mln::util::edge< G >::operator edge_id_t ( ) const [inline]`

Conversion to the edge id.

Definition at line 274 of file edge.hh.

**10.365.4.9** `template<typename G> void mln::util::edge< G >::update_id ( const edge_id_t & id ) [inline]`

Set `id_` with `id`;

Definition at line 267 of file edge.hh.

**10.365.4.10** `template<typename G> vertex_id_t mln::util::edge< G >::v1 ( ) const [inline]`

[Edge](#) oriented.

Return the lowest vertex id adjacent to this edge.

Definition at line 324 of file edge.hh.

Referenced by `mln::util::edge< G >::v_other()`.

**10.365.4.11** `template<typename G> vertex_id_t mln::util::edge< G >::v2 ( ) const [inline]`

Return the highest vertex id adjacent to this edge.

Definition at line 333 of file edge.hh.

Referenced by `mln::util::edge< G >::v_other()`.

**10.365.4.12** `template<typename G> vertex_id_t mln::util::edge< G >::v_other ( const vertex_id_t & id_v ) const [inline]`

[Vertex](#) and edges oriented.

Return the vertex id of this edge which is different from `id_v`.

Definition at line 315 of file edge.hh.

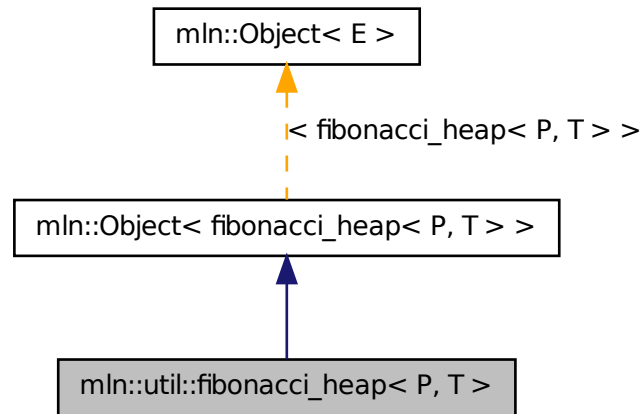
References `mln::util::edge< G >::v1()`, and `mln::util::edge< G >::v2()`.

**10.366** `mln::util::fibonacci_heap< P, T >` Class Template Reference

Fibonacci heap.

```
#include <fibonacci_heap.hh>
```

Inheritance diagram for mln::util::fibonacci\_heap< P, T >:



## Public Member Functions

- void [clear](#) ()  
*Clear all elements in the heap and make the heap empty.*
- [fibonacci\\_heap](#) ()  
*Default constructor.*
- [fibonacci\\_heap](#) (const [fibonacci\\_heap](#)< P, T > &node)  
*Copy constructor Be ware that once this heap is constructed, the argument node is cleared and all its elements are part of this new heap.*
- const T & [front](#) () const  
*Return the minimum value in the heap.*
- bool [is\\_empty](#) () const  
*Is it empty?*
- bool [is\\_valid](#) () const  
*return false if it is empty.*
- unsigned [nelements](#) () const  
*Return the number of elements.*
- [fibonacci\\_heap](#)< P, T > & [operator=](#) ([fibonacci\\_heap](#)< P, T > &rhs)  
*Assignment operator.*
- T [pop\\_front](#) ()

*Return and remove the minimum value in the heap.*

- void `push` (const P &priority, const T &value)  
*Push a new element in the heap.*
- void `push` (`fibonacci_heap`< P, T > &other\_heap)  
*Take other\_heap's elements and insert them in this heap.*

### 10.366.1 Detailed Description

`template<typename P, typename T> class mln::util::fibonacci_heap< P, T >`

Fibonacci heap.

Definition at line 117 of file fibonacci\_heap.hh.

### 10.366.2 Constructor & Destructor Documentation

**10.366.2.1** `template<typename P , typename T > mln::util::fibonacci_heap< P, T >::fibonacci_heap ( ) [inline]`

Default constructor.

Definition at line 472 of file fibonacci\_heap.hh.

**10.366.2.2** `template<typename P , typename T > mln::util::fibonacci_heap< P, T >::fibonacci_heap ( const fibonacci_heap< P, T > & node ) [inline]`

Copy constructor Be ware that once this heap is constructed, the argument `node` is cleared and all its elements are part of this new heap.

Definition at line 480 of file fibonacci\_heap.hh.

### 10.366.3 Member Function Documentation

**10.366.3.1** `template<typename P , typename T > void mln::util::fibonacci_heap< P, T >::clear ( ) [inline]`

Clear all elements in the heap and make the heap empty.

Definition at line 723 of file fibonacci\_heap.hh.

References `mln::util::fibonacci_heap< P, T >::pop_front()`.

**10.366.3.2** `template<typename P , typename T > const T & mln::util::fibonacci_heap< P, T >::front ( ) const [inline]`

Return the minimum value in the heap.

Definition at line 569 of file fibonacci\_heap.hh.

**10.366.3.3** `template<typename P , typename T > bool mln::util::fibonacci_heap< P, T >::is_empty ( ) const [inline]`

Is it empty?

Definition at line 705 of file fibonacci\_heap.hh.

Referenced by `mln::util::fibonacci_heap< P, T >::pop_front()`, and `mln::util::fibonacci_heap< P, T >::push()`.

**10.366.3.4** `template<typename P , typename T > bool mln::util::fibonacci_heap< P, T >::is_valid ( ) const [inline]`

return false if it is empty.

Definition at line 714 of file fibonacci\_heap.hh.

Referenced by `mln::util::fibonacci_heap< P, T >::pop_front()`.

**10.366.3.5** `template<typename P , typename T > unsigned mln::util::fibonacci_heap< P, T >::nelements ( ) const [inline]`

Return the number of elements.

Definition at line 745 of file fibonacci\_heap.hh.

**10.366.3.6** `template<typename P , typename T > fibonacci_heap< P, T > & mln::util::fibonacci_heap< P, T >::operator= ( fibonacci_heap< P, T > & rhs ) [inline]`

Assignment operator.

Be ware that this operator do *\*not\** copy the data from `rhs` to this heap. It moves all elements which means that afterwards, `rhs` is is cleared and all its elements are part of this new heap.

Definition at line 754 of file fibonacci\_heap.hh.

**10.366.3.7** `template<typename P , typename T > T mln::util::fibonacci_heap< P, T >::pop_front ( ) [inline]`

Return and remove the minimum value in the heap.

Definition at line 578 of file fibonacci\_heap.hh.

References `mln::util::fibonacci_heap< P, T >::is_empty()`, `mln::util::fibonacci_heap< P, T >::is_valid()`, and `mln::util::fibonacci_heap< P, T >::push()`.

Referenced by `mln::util::fibonacci_heap< P, T >::clear()`.

**10.366.3.8** `template<typename P , typename T > void mln::util::fibonacci_heap< P, T >::push ( const P & priority, const T & value ) [inline]`

Push a new element in the heap.

**See also**

insert

Definition at line 508 of file fibonacci\_heap.hh.

Referenced by mln::util::fibonacci\_heap< P, T >::pop\_front().

### 10.366.3.9 `template<typename P, typename T> void mln::util::fibonacci_heap< P, T >::push ( fibonacci_heap< P, T > & other_heap ) [inline]`

Take `other_heap`'s elements and insert them in this heap.

After this call `other_heap` is cleared.

Definition at line 520 of file fibonacci\_heap.hh.

References mln::util::fibonacci\_heap< P, T >::is\_empty().

## 10.367 mln::util::graph Class Reference

Undirected graph.

```
#include <graph.hh>
```

Inherits graph\_base< graph >.

### Public Types

- typedef std::set< edge\_data\_t > [edges\\_set\\_t](#)  
*A set to test the presence of a given edge.*
- typedef std::vector< edge\_data\_t > [edges\\_t](#)  
*The type of the set of edges.*
- typedef std::vector< vertex\_data\_t > [vertices\\_t](#)  
*The type of the set of vertices.*
- typedef mln::internal::vertex\_fwd\_iterator< [graph](#) > [vertex\\_fwd\\_iter](#)  
*Iterator types*  
*Vertex iterators.*
- typedef mln::internal::vertex\_nbh\_edge\_fwd\_iterator< [graph](#) > [vertex\\_nbh\\_edge\\_fwd\\_iter](#)  
*Vertex centered edge iterators.*
- typedef mln::internal::vertex\_nbh\_vertex\_fwd\_iterator< [graph](#) > [vertex\\_nbh\\_vertex\\_fwd\\_iter](#)  
*Vertex centered vertex iterators.*
- typedef mln::internal::edge\_fwd\_iterator< [graph](#) > [edge\\_fwd\\_iter](#)



*Edge iterators.*

- typedef mln::internal::edge\_nbh\_edge\_fwd\_iterator< graph > edge\_nbh\_edge\_fwd\_iter  
*Edge centered edge iterators.*

## Public Member Functions

- graph ()
- graph (unsigned nvertices)  
*Construct a graph with nvertices vertices.*
- bool has\_v (const vertex\_id\_t &id\_v) const  
*Check whether a vertex id id\_v exists in the graph.*
- edge\_id\_t v\_ith\_nbh\_edge (const vertex\_id\_t &id\_v, unsigned i) const  
*Returns the i th edge adjacent to the vertex id\_v.*
- vertex\_id\_t v\_ith\_nbh\_vertex (const vertex\_id\_t &id\_v, unsigned i) const  
*Returns the i th vertex adjacent to the vertex id\_v.*
- size\_t v\_nmax () const  
*Return the number of vertices in the graph.*
- size\_t v\_nmax\_nbh\_edges (const vertex\_id\_t &id\_v) const  
*Return the number of adjacent edges of vertex id\_v.*
- size\_t v\_nmax\_nbh\_vertices (const vertex\_id\_t &id\_v) const  
*Return the number of adjacent vertices of vertex id\_v.*
- unsigned add\_vertex ()  
*Vertex oriented.*
- std::pair< vertex\_id\_t, vertex\_id\_t > add\_vertices (unsigned n)  
*Add n vertices to the graph.*
- vertex\_t vertex (vertex\_id\_t id\_v) const  
*Return the vertex whose id is v.*
- edge\_id\_t add\_edge (const vertex\_id\_t &id\_v1, const vertex\_id\_t &id\_v2)  
*Edge oriented.*
- edge\_t edge (const edge\_id\_t &e) const  
*Return the edge whose id is e.*
- const std::vector< util::ord\_pair< vertex\_id\_t > > & edges () const  
*Return the list of all edges.*

- `size_t e_nmax () const`  
*Return the number of edges in the graph.*
- `bool has_e (const edge_id_t &id_e) const`  
*Return whether `id_e` is in the graph.*
- `edge_t edge (const vertex_t &v1, const vertex_t &v2) const`  
*Return the corresponding edge id if exists.*
- `vertex_id_t v1 (const edge_id_t &id_e) const`  
*Return the first vertex associated to the edge `id_e`.*
- `vertex_id_t v2 (const edge_id_t &id_e) const`  
*Return the second vertex associated to edge `id_e`.*
- `size_t e_nmax_nbh_edges (const edge_id_t &id_e) const`  
*Return the number max of adjacent edge, given an edge `id_e`.*
- `edge_id_t e_ith_nbh_edge (const edge_id_t &id_e, unsigned i) const`  
*Return the `i` th edge adjacent to the edge `id_e`.*
- `template<typename G2 >`  
`bool is_subgraph_of (const G2 &g) const`  
*Return whether this graph is a subgraph Return true if `g` and `*this` have the same `graph_id`.*

### 10.367.1 Detailed Description

Undirected graph.

Definition at line 87 of file `mln/util/graph.hh`.

### 10.367.2 Member Typedef Documentation

#### 10.367.2.1 `typedef mln::internal::edge_fwd_iterator<graph> mln::util::graph::edge_fwd_iter`

Edge iterators.

Definition at line 129 of file `mln/util/graph.hh`.

#### 10.367.2.2 `typedef mln::internal::edge_nbh_edge_fwd_iterator<graph>` `mln::util::graph::edge_nbh_edge_fwd_iter`

Edge centered edge iterators.

Definition at line 136 of file `mln/util/graph.hh`.

#### 10.367.2.3 `typedef std::set<edge_data_t> mln::util::graph::edges_set_t`

A set to test the presence of a given edge.

Definition at line 102 of file `mln/util/graph.hh`.

### 10.367.2.4 `typedef std::vector<edge_data_t> mln::util::graph::edges_t`

The type of the set of edges.

Definition at line 100 of file mln/util/graph.hh.

### 10.367.2.5 `typedef mln::internal::vertex_fwd_iterator<graph> mln::util::graph::vertex_fwd_iter`

[Iterator](#) types

[Vertex](#) iterators.

Definition at line 108 of file mln/util/graph.hh.

### 10.367.2.6 `typedef mln::internal::vertex_nbh_edge_fwd_iterator<graph> mln::util::graph::vertex_nbh_edge_fwd_iter`

[Vertex](#) centered edge iterators.

Definition at line 115 of file mln/util/graph.hh.

### 10.367.2.7 `typedef mln::internal::vertex_nbh_vertex_fwd_iterator<graph> mln::util::graph::vertex_nbh_vertex_fwd_iter`

[Vertex](#) centered vertex iterators.

Definition at line 122 of file mln/util/graph.hh.

### 10.367.2.8 `typedef std::vector<vertex_data_t> mln::util::graph::vertices_t`

The type of the set of vertices.

Definition at line 97 of file mln/util/graph.hh.

## 10.367.3 Constructor & Destructor Documentation

### 10.367.3.1 `mln::util::graph::graph ( ) [inline]`

Constructor.

Definition at line 282 of file mln/util/graph.hh.

### 10.367.3.2 `mln::util::graph::graph ( unsigned nvertices ) [inline]`

Construct a graph with `nvertices` vertices.

Definition at line 288 of file mln/util/graph.hh.

## 10.367.4 Member Function Documentation

### 10.367.4.1 `edge_id_t mln::util::graph::add_edge ( const vertex_id_t & id_v1, const vertex_id_t & id_v2 ) [inline]`

[Edge](#) oriented.

Add an edge.

#### Returns

The id of the new edge if it does not exist yet; otherwise, return `mln_max(unsigned)`.

Definition at line 386 of file `mln/util/graph.hh`.

References `edge()`, and `has_v()`.

Referenced by `mln::make::voronoi()`.

### 10.367.4.2 `unsigned mln::util::graph::add_vertex ( ) [inline]`

[Vertex](#) oriented.

Shortcuts factoring the insertion of vertices and edges. Add a vertex.

#### Returns

The id of the new vertex.

Definition at line 299 of file `mln/util/graph.hh`.

References `v_nmax()`.

Referenced by `mln::make::voronoi()`.

### 10.367.4.3 `std::pair< vertex_id_t, vertex_id_t > mln::util::graph::add_vertices ( unsigned n ) [inline]`

Add `n` vertices to the graph.

#### Returns

A range of vertex ids.

Definition at line 310 of file `mln/util/graph.hh`.

References `v_nmax()`.

### 10.367.4.4 `edge_id_t mln::util::graph::e_ith_nbh_edge ( const edge_id_t & id_e, unsigned i ) const [inline]`

Return the `i` th edge adjacent to the edge `id_e`.

Definition at line 503 of file `mln/util/graph.hh`.

References `e_nmax()`, `e_nmax_nbh_edges()`, `has_e()`, `v1()`, `v2()`, `v_ith_nbh_edge()`, and `v_nmax_nbh_edges()`.

**10.367.4.5** `size_t mln::util::graph::e_nmax ( ) const [inline]`

Return the number of edges in the graph.

Definition at line 443 of file mln/util/graph.hh.

Referenced by `e_ith_nbh_edge()`, and `edge()`.

**10.367.4.6** `size_t mln::util::graph::e_nmax_nbh_edges ( const edge_id_t & id_e ) const [inline]`

Return the number max of adjacent edge, given an edge `id_e`.

Definition at line 495 of file mln/util/graph.hh.

References `has_e()`, `v1()`, `v2()`, and `v_nmax_nbh_edges()`.

Referenced by `e_ith_nbh_edge()`.

**10.367.4.7** `graph::edge_t mln::util::graph::edge ( const edge_id_t & e ) const [inline]`

Return the edge whose id is `e`.

Definition at line 435 of file mln/util/graph.hh.

References `e_nmax()`.

Referenced by `add_edge()`.

**10.367.4.8** `graph::edge_t mln::util::graph::edge ( const vertex_t & v1, const vertex_t & v2 ) const [inline]`

Return the corresponding edge id if exists.

If it is not, returns an invalid edge.

Definition at line 457 of file mln/util/graph.hh.

References `has_v()`.

**10.367.4.9** `const std::vector< util::ord_pair< vertex_id_t > > & mln::util::graph::edges ( ) const [inline]`

Return the list of all edges.

Definition at line 428 of file mln/util/graph.hh.

**10.367.4.10** `bool mln::util::graph::has_e ( const edge_id_t & id_e ) const [inline]`

Return whether `id_e` is in the graph.

Definition at line 450 of file mln/util/graph.hh.

Referenced by `e_ith_nbh_edge()`, `e_nmax_nbh_edges()`, `v1()`, and `v2()`.

**10.367.4.11** `bool mln::util::graph::has_v ( const vertex_id_t & id_v ) const [inline]`

Check whether a vertex id `id_v` exists in the graph.

Definition at line 338 of file `mln/util/graph.hh`.

Referenced by `add_edge()`, `edge()`, `v_ith_nbh_edge()`, `v_ith_nbh_vertex()`, `v_nmax_nbh_edges()`, `v_nmax_nbh_vertices()`, and `vertex()`.

**10.367.4.12** `template<typename G2 > bool mln::util::graph::is_subgraph_of ( const G2 & g ) const [inline]`

Return whether this graph is a subgraph Return true if `g` and `*this` have the same `graph_id`.

Definition at line 519 of file `mln/util/graph.hh`.

**10.367.4.13** `vertex_id_t mln::util::graph::v1 ( const edge_id_t & id_e ) const [inline]`

Return the first vertex associated to the edge `id_e`.

Definition at line 479 of file `mln/util/graph.hh`.

References `has_e()`.

Referenced by `e_ith_nbh_edge()`, and `e_nmax_nbh_edges()`.

**10.367.4.14** `vertex_id_t mln::util::graph::v2 ( const edge_id_t & id_e ) const [inline]`

Return the second vertex associated to edge `id_e`.

Definition at line 487 of file `mln/util/graph.hh`.

References `has_e()`.

Referenced by `e_ith_nbh_edge()`, and `e_nmax_nbh_edges()`.

**10.367.4.15** `edge_id_t mln::util::graph::v_ith_nbh_edge ( const vertex_id_t & id_v, unsigned i ) const [inline]`

Returns the `i` th edge adjacent to the vertex `id_v`.

Definition at line 353 of file `mln/util/graph.hh`.

References `has_v()`, and `v_nmax_nbh_edges()`.

Referenced by `e_ith_nbh_edge()`, and `v_ith_nbh_vertex()`.

**10.367.4.16** `vertex_id_t mln::util::graph::v_ith_nbh_vertex ( const vertex_id_t & id_v, unsigned i ) const [inline]`

Returns the `i` th vertex adjacent to the vertex `id_v`.

Definition at line 371 of file `mln/util/graph.hh`.

References `has_v()`, and `v_ith_nbh_edge()`.

**10.367.4.17** `size_t mln::util::graph::v_nmax ( ) const [inline]`

Return the number of vertices in the graph.

Definition at line 331 of file mln/util/graph.hh.

Referenced by `add_vertex()`, and `add_vertices()`.

**10.367.4.18** `size_t mln::util::graph::v_nmax_nbh_edges ( const vertex_id_t & id_v ) const [inline]`

Return the number of adjacent edges of vertex `id_v`.

Definition at line 345 of file mln/util/graph.hh.

References `has_v()`.

Referenced by `e_ith_nbh_edge()`, `e_nmax_nbh_edges()`, `v_ith_nbh_edge()`, and `v_nmax_nbh_vertices()`.

**10.367.4.19** `size_t mln::util::graph::v_nmax_nbh_vertices ( const vertex_id_t & id_v ) const [inline]`

Return the number of adjacent vertices of vertex `id_v`.

Definition at line 363 of file mln/util/graph.hh.

References `has_v()`, and `v_nmax_nbh_edges()`.

**10.367.4.20** `graph::vertex_t mln::util::graph::vertex ( vertex_id_t id_v ) const [inline]`

Return the vertex whose id is `v`.

Definition at line 322 of file mln/util/graph.hh.

References `has_v()`.

**10.368 mln::util::greater\_point< I > Class Template Reference**

A “greater than” functor comparing points w.r.t.

```
#include <greater_point.hh>
```

**Public Member Functions**

- bool `operator()` (const point &x, const point &y)

*Is x greater than y?*

**10.368.1 Detailed Description**

```
template<typename I> class mln::util::greater_point< I >
```

A “greater than” functor comparing points w.r.t. the values they refer to in an image.

This functor used in useful to implement ordered queues of points.

Definition at line 42 of file greater\_point.hh.

## 10.368.2 Member Function Documentation

### 10.368.2.1 `template<typename I> bool mln::util::greater_point< I >::operator() ( const point & x, const point & y )`

Is *x* greater than *y*?

Definition at line 74 of file greater\_point.hh.

## 10.369 `mln::util::greater_psite< I >` Class Template Reference

A “greater than” functor comparing psites w.r.t.

```
#include <greater_psite.hh>
```

### Public Member Functions

- `bool operator() (const psite &x, const psite &y)`  
*Is x greater than y?*

### 10.369.1 Detailed Description

```
template<typename I> class mln::util::greater_psite< I >
```

A “greater than” functor comparing psites w.r.t. the values they refer to in an image.

This functor used in useful to implement ordered queues of psites.

Definition at line 42 of file greater\_psite.hh.

### 10.369.2 Member Function Documentation

#### 10.369.2.1 `template<typename I> bool mln::util::greater_psite< I >::operator() ( const psite & x, const psite & y )`

Is *x* greater than *y*?

Definition at line 74 of file greater\_psite.hh.

## 10.370 `mln::util::head< T, R >` Class Template Reference

Top structure of the soft heap.

```
#include <soft_heap.hh>
```



### 10.370.1 Detailed Description

```
template<typename T, typename R> class mln::util::head< T, R >
```

Top structure of the soft heap.

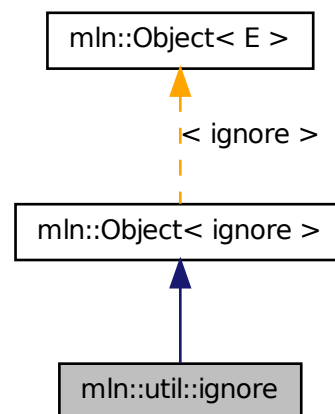
Definition at line 138 of file soft\_heap.hh.

## 10.371 mln::util::ignore Struct Reference

Ignore structure.

```
#include <ignore.hh>
```

Inheritance diagram for mln::util::ignore:



### 10.371.1 Detailed Description

Ignore structure.

Definition at line 46 of file ignore.hh.

## 10.372 mln::util::ilcell< T > Struct Template Reference

Element of an item list. Store the data (key) used in [soft\\_heap](#).

```
#include <soft_heap.hh>
```

### 10.372.1 Detailed Description

`template<typename T> struct mln::util::ilcell< T >`

Element of an item list. Store the data (key) used in [soft\\_heap](#).

Definition at line 76 of file `soft_heap.hh`.

## 10.373 mln::util::line\_graph< G > Class Template Reference

Undirected line graph of a graph of type G.

```
#include <line_graph.hh>
```

Inherits `graph_base< line_graph< G > >`.

### Public Types

- `typedef std::vector< edge_data_t > edges_t`  
*The type of the set of edges.*
- `typedef std::vector< vertex_data_t > vertices_t`  
*The type of the set of vertices.*
- `typedef mln::internal::vertex_fwd_iterator< line_graph< G > > vertex_fwd_iter`  
*Iterator types*  
*Vertex iterators.*
- `typedef mln::internal::edge_fwd_iterator< line_graph< G > > edge_fwd_iter`  
*Edge iterators.*
- `typedef mln::internal::edge_nbh_edge_fwd_iterator< line_graph< G > > edge_nbh_edge_fwd_iter`  
*Edge nbh edge iterators.*
- `typedef mln::internal::vertex_nbh_vertex_fwd_iterator< line_graph< G > > vertex_nbh_vertex_fwd_iter`  
*Vertex nbh vertex iterators.*
- `typedef mln::internal::vertex_nbh_edge_fwd_iterator< line_graph< G > > vertex_nbh_edge_fwd_iter`  
*Vertex nbh edge iterators.*

## Public Member Functions

- `template<typename G2 >`  
`bool has (const util::vertex< G2 > &v) const`  
*Check whether a vertex `v` exists in the line graph.*
- `bool has_v (const vertex_id_t &id_v) const`  
*Check whether a vertex id `id_v` exists in the line graph.*
- `edge_id_t v_ith_nbh_edge (const vertex_id_t &id_v, unsigned i) const`  
*Returns the `i` th edge adjacent to the vertex `id_v`.*
- `vertex_id_t v_ith_nbh_vertex (const vertex_id_t &id_v, unsigned i) const`  
*Returns the `i` th vertex adjacent to the vertex `id_v`.*
- `size_t v_nmax () const`  
*Return the number of vertices in the graph.*
- `size_t v_nmax_nbh_edges (const vertex_id_t &id_v) const`  
*Return the number of adjacent edges of vertex `id_v`.*
- `size_t v_nmax_nbh_vertices (const vertex_id_t &id_v) const`  
*Return the number of adjacent vertices of vertex `id_v`.*
  
- `vertex_t vertex (const vertex_id_t &id_v) const`  
*Vertex oriented.*
  
- `edge_t edge (const edge_id_t &e) const`  
*Edge oriented.*
- `size_t e_nmax () const`  
*Return the number of edges in the graph.*
- `bool has_e (const util::edge_id_t &id_e) const`  
*Return whether `id_e` is in the line graph.*
- `template<typename G2 >`  
`bool has (const util::edge< G2 > &e) const`  
*Return whether `e` is in the line graph.*
- `vertex_id_t v1 (const edge_id_t &id_e) const`  
*Return the first vertex associated to the edge `id_e`.*
- `vertex_id_t v2 (const edge_id_t &id_e) const`  
*Return the second vertex associated to edge `id_e`.*
- `size_t e_nmax_nbh_edges (const edge_id_t &id_e) const`  
*Return the number max of adjacent edge, given an edge `id_e`.*
- `edge_id_t e_ith_nbh_edge (const edge_id_t &id_e, unsigned i) const`

Return the  $i$  th edge adjacent to the edge  $id_e$ .

- `template<typename G2 >`  
`bool is_subgraph_of (const G2 &g) const`  
 Return whether this graph is a subgraph Return true if  $g$  and  $*this$  have the same  $graph\_id$ .
- `const G & graph () const`  
 Return the underlying graph.

### 10.373.1 Detailed Description

`template<typename G> class mln::util::line_graph< G >`

Undirected line graph of a graph of type  $G$ .

Definition at line 82 of file `line_graph.hh`.

### 10.373.2 Member Typedef Documentation

**10.373.2.1** `template<typename G> typedef mln::internal::edge_fwd_iterator< line_graph<G> > mln::util::line_graph< G >::edge_fwd_iter`

[Edge](#) iterators.

Definition at line 114 of file `line_graph.hh`.

**10.373.2.2** `template<typename G> typedef mln::internal::edge_nbh_edge_fwd_iterator< line_graph<G> > mln::util::line_graph< G >::edge_nbh_edge_fwd_iter`

[Edge](#) nbh edge iterators.

Definition at line 123 of file `line_graph.hh`.

**10.373.2.3** `template<typename G> typedef std::vector<edge_data_t> mln::util::line_graph< G >::edges_t`

The type of the set of edges.

Definition at line 98 of file `line_graph.hh`.

**10.373.2.4** `template<typename G> typedef mln::internal::vertex_fwd_iterator< line_graph<G> > mln::util::line_graph< G >::vertex_fwd_iter`

[Iterator](#) types

[Vertex](#) iterators.

Definition at line 105 of file `line_graph.hh`.

**10.373.2.5** `template<typename G> typedef mln::internal::vertex_nbh_edge_fwd_iterator< line_graph<G> > mln::util::line_graph< G >::vertex_nbh_edge_fwd_iter`

Vertex nbh edge iterators.

Definition at line 141 of file line\_graph.hh.

**10.373.2.6** `template<typename G> typedef mln::internal::vertex_nbh_vertex_fwd_iterator< line_graph<G> > mln::util::line_graph< G >::vertex_nbh_vertex_fwd_iter`

Vertex nbh vertex iterators.

Definition at line 132 of file line\_graph.hh.

**10.373.2.7** `template<typename G> typedef std::vector<vertex_data_t> mln::util::line_graph< G >::vertices_t`

The type of the set of vertices.

Definition at line 95 of file line\_graph.hh.

### 10.373.3 Member Function Documentation

**10.373.3.1** `template<typename G > edge_id_t mln::util::line_graph< G >::e_ith_nbh_edge ( const edge_id_t & id_e, unsigned i ) const [inline]`

Return the *i* th edge adjacent to the edge *id\_e*.

Definition at line 460 of file line\_graph.hh.

References `mln::util::line_graph< G >::e_nmax()`, `mln::util::line_graph< G >::e_nmax_nbh_edges()`, `mln::util::line_graph< G >::has_e()`, `mln::util::line_graph< G >::v1()`, `mln::util::line_graph< G >::v2()`, `mln::util::line_graph< G >::v_ith_nbh_edge()`, and `mln::util::line_graph< G >::v_nmax_nbh_edges()`.

**10.373.3.2** `template<typename G > size_t mln::util::line_graph< G >::e_nmax ( ) const [inline]`

Return the number of edges in the graph.

Definition at line 408 of file line\_graph.hh.

Referenced by `mln::util::line_graph< G >::e_ith_nbh_edge()`, and `mln::util::line_graph< G >::edge()`.

**10.373.3.3** `template<typename G > size_t mln::util::line_graph< G >::e_nmax_nbh_edges ( const edge_id_t & id_e ) const [inline]`

Return the number max of adjacent edge, given an edge *id\_e*.

Definition at line 451 of file line\_graph.hh.

References `mln::util::line_graph< G >::has_e()`, `mln::util::line_graph< G >::v1()`, `mln::util::line_graph< G >::v2()`, and `mln::util::line_graph< G >::v_nmax_nbh_edges()`.

Referenced by `mln::util::line_graph< G >::e_ith_nbh_edge()`.

**10.373.3.4** `template<typename G > line_graph< G >::edge_t mln::util::line_graph< G >::edge ( const edge_id_t & e ) const [inline]`

Edge oriented.

Return the edge whose id is *e*.

Definition at line 399 of file line\_graph.hh.

References mln::util::line\_graph< G >::e\_nmax().

**10.373.3.5** `template<typename G > const G & mln::util::line_graph< G >::graph ( ) const [inline]`

Return the underlying graph.

Definition at line 485 of file line\_graph.hh.

**10.373.3.6** `template<typename G > template<typename G2 > bool mln::util::line_graph< G >::has ( const util::vertex< G2 > & v ) const [inline]`

Check whether a vertex *v* exists in the line graph.

Definition at line 345 of file line\_graph.hh.

References mln::util::vertex< G >::graph(), mln::util::line\_graph< G >::has\_v(), and mln::util::vertex< G >::id().

**10.373.3.7** `template<typename G > template<typename G2 > bool mln::util::line_graph< G >::has ( const util::edge< G2 > & e ) const [inline]`

Return whether *e* is in the line graph.

Definition at line 425 of file line\_graph.hh.

References mln::util::edge< G >::graph(), mln::util::line\_graph< G >::has\_e(), and mln::util::edge< G >::id().

**10.373.3.8** `template<typename G > bool mln::util::line_graph< G >::has_e ( const util::edge_id_t & id_e ) const [inline]`

Return whether *id\_e* is in the line graph.

Definition at line 416 of file line\_graph.hh.

Referenced by mln::util::line\_graph< G >::e\_ith\_nbh\_edge(), mln::util::line\_graph< G >::e\_nmax\_nbh\_edges(), mln::util::line\_graph< G >::has(), mln::util::line\_graph< G >::v1(), and mln::util::line\_graph< G >::v2().

**10.373.3.9** `template<typename G > bool mln::util::line_graph< G >::has_v ( const vertex_id_t & id_v ) const [inline]`

Check whether a vertex id *id\_v* exists in the line graph.

Definition at line 336 of file line\_graph.hh.

Referenced by mln::util::line\_graph< G >::has(), mln::util::line\_graph< G >::v\_ith\_nbh\_edge(), mln::util::line\_graph< G >::v\_ith\_nbh\_vertex(), mln::util::line\_graph< G >::v\_nmax\_nbh\_edges(), mln::util::line\_graph< G >::v\_nmax\_nbh\_vertices(), and mln::util::line\_graph< G >::vertex().

**10.373.3.10** `template<typename G > template<typename G2 > bool mln::util::line_graph< G >::is_subgraph_of( const G2 & g ) const [inline]`

Return whether this graph is a subgraph Return true if g and \*this have the same graph\_id.

Definition at line 477 of file line\_graph.hh.

**10.373.3.11** `template<typename G > vertex_id_t mln::util::line_graph< G >::v1( const edge_id_t & id_e ) const [inline]`

Return the first vertex associated to the edge id\_e.

Definition at line 433 of file line\_graph.hh.

References mln::util::line\_graph< G >::has\_e().

Referenced by mln::util::line\_graph< G >::e\_ith\_nbh\_edge(), and mln::util::line\_graph< G >::e\_nmax\_nbh\_edges().

**10.373.3.12** `template<typename G > vertex_id_t mln::util::line_graph< G >::v2( const edge_id_t & id_e ) const [inline]`

Return the second vertex associated to edge id\_e.

Definition at line 442 of file line\_graph.hh.

References mln::util::line\_graph< G >::has\_e().

Referenced by mln::util::line\_graph< G >::e\_ith\_nbh\_edge(), and mln::util::line\_graph< G >::e\_nmax\_nbh\_edges().

**10.373.3.13** `template<typename G > edge_id_t mln::util::line_graph< G >::v_ith_nbh_edge( const vertex_id_t & id_v, unsigned i ) const [inline]`

Returns the i th edge adjacent to the vertex id\_v.

Definition at line 363 of file line\_graph.hh.

References mln::util::line\_graph< G >::has\_v(), mln::util::line\_graph< G >::v\_nmax(), and mln::util::line\_graph< G >::v\_nmax\_nbh\_edges().

Referenced by mln::util::line\_graph< G >::e\_ith\_nbh\_edge(), and mln::util::line\_graph< G >::v\_ith\_nbh\_vertex().

**10.373.3.14** `template<typename G > vertex_id_t mln::util::line_graph< G >::v_ith_nbh_vertex( const vertex_id_t & id_v, unsigned i ) const [inline]`

Returns the i th vertex adjacent to the vertex id\_v.

Definition at line 383 of file line\_graph.hh.

References mln::util::line\_graph< G >::has\_v(), and mln::util::line\_graph< G >::v\_ith\_nbh\_edge().

**10.373.3.15** `template<typename G> size_t mln::util::line_graph< G >::v_nmax ( ) const [inline]`

Return the number of vertices in the graph.

Definition at line 328 of file line\_graph.hh.

Referenced by `mln::util::line_graph< G >::v_ith_nbh_edge()`.

**10.373.3.16** `template<typename G> size_t mln::util::line_graph< G >::v_nmax_nbh_edges ( const vertex_id_t & id_v ) const [inline]`

Return the number of adjacent edges of vertex `id_v`.

Definition at line 354 of file line\_graph.hh.

References `mln::util::line_graph< G >::has_v()`.

Referenced by `mln::util::line_graph< G >::e_ith_nbh_edge()`, `mln::util::line_graph< G >::e_nmax_nbh_edges()`, `mln::util::line_graph< G >::v_ith_nbh_edge()`, and `mln::util::line_graph< G >::v_nmax_nbh_vertices()`.

**10.373.3.17** `template<typename G> size_t mln::util::line_graph< G >::v_nmax_nbh_vertices ( const vertex_id_t & id_v ) const [inline]`

Return the number of adjacent vertices of vertex `id_v`.

Definition at line 374 of file line\_graph.hh.

References `mln::util::line_graph< G >::has_v()`, and `mln::util::line_graph< G >::v_nmax_nbh_edges()`.

**10.373.3.18** `template<typename G> line_graph< G >::vertex_t mln::util::line_graph< G >::vertex ( const vertex_id_t & id_v ) const [inline]`

[Vertex](#) oriented.

Shortcuts factoring the insertion of vertices and edges.

Return the vertex whose id is `v`.

Definition at line 318 of file line\_graph.hh.

References `mln::util::line_graph< G >::has_v()`.

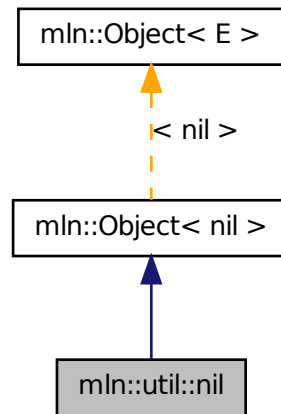
## 10.374 mln::util::nil Struct Reference

Nil structure.

```
#include <nil.hh>
```



Inheritance diagram for mln::util::nil:



### 10.374.1 Detailed Description

Nil structure.

Definition at line 46 of file util/nil.hh.

## 10.375 mln::util::node< T, R > Class Template Reference

Meta-data of an element in the heap.

```
#include <soft_heap.hh>
```

### 10.375.1 Detailed Description

```
template<typename T, typename R> class mln::util::node< T, R >
```

Meta-data of an element in the heap.

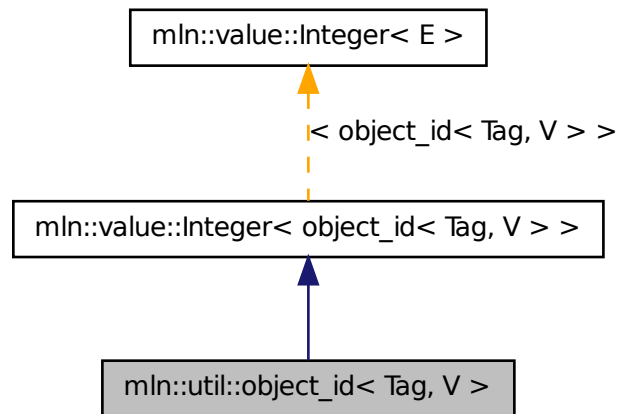
Definition at line 97 of file soft\_heap.hh.

## 10.376 mln::util::object\_id< Tag, V > Class Template Reference

Base class of an object id.

```
#include <object_id.hh>
```

Inheritance diagram for `mln::util::object_id< Tag, V >`:



## Public Types

- typedef `V` `value_t`

*The underlying type id.*

## Public Member Functions

- `object_id()`

*Constructors.*

### 10.376.1 Detailed Description

```
template<typename Tag, typename V> class mln::util::object_id< Tag, V >
```

Base class of an object id.

#### Template Parameters

*Tag* the tag type

*Equiv* the equivalent value.

Definition at line 66 of file `object_id.hh`.

## 10.376.2 Member Typedef Documentation

**10.376.2.1** `template<typename Tag, typename V> typedef V mln::util::object_id< Tag, V >::value_t`

The underlying type id.

Definition at line 70 of file object\_id.hh.

## 10.376.3 Constructor & Destructor Documentation

**10.376.3.1** `template<typename Tag , typename V > mln::util::object_id< Tag, V >::object_id ( ) [inline]`

Constructors.

Definition at line 120 of file object\_id.hh.

## 10.377 mln::util::ord< T > Struct Template Reference

Function-object that defines an ordering between objects with type T: *lhs R rhs*.

```
#include <ord.hh>
```

### 10.377.1 Detailed Description

`template<typename T> struct mln::util::ord< T >`

Function-object that defines an ordering between objects with type T: *lhs R rhs*. Its meaning is "lhs less-than rhs."

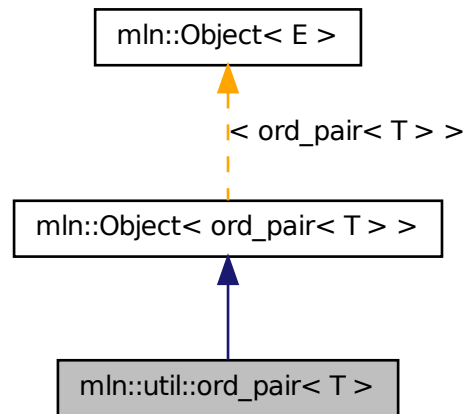
Definition at line 48 of file util/ord.hh.

## 10.378 mln::util::ord\_pair< T > Struct Template Reference

Ordered pair structure s.a.

```
#include <ord_pair.hh>
```

Inheritance diagram for `mln::util::ord_pair< T >`:



## Public Member Functions

- void `change_both` (const T &first, const T &second)  
*Replace both members of the pair by val, while keeping the relative order.*
- void `change_first` (const T &val)  
*Replace the first member of the pair by val, while keeping the relative order.*
- void `change_second` (const T &val)  
*Replace the second member of the pair by val, while keeping the relative order.*
- const T & `first` () const  
*Get the first (lowest) member of the pair.*
- const T & `second` () const  
*Get the second (highest) member of the pair.*

### 10.378.1 Detailed Description

`template<typename T> struct mln::util::ord_pair< T >`

Ordered pair structure s.a. `this->first <= this->second`; ordered pairs are partially ordered using lexicographical ordering.

Definition at line 50 of file `ord_pair.hh`.

## 10.378.2 Member Function Documentation

### 10.378.2.1 `template<typename T > void mln::util::ord_pair< T >::change_both ( const T & first, const T & second ) [inline]`

Replace both members of the pair by *val*, while keeping the relative order.

#### Postcondition

*first\_ <= second\_* (with *<=* being the [mln::util::ord\\_weak](#) relationship).

Definition at line 211 of file ord\_pair.hh.

References [mln::util::ord\\_strict\(\)](#), and [mln::util::ord\\_weak\(\)](#).

### 10.378.2.2 `template<typename T > void mln::util::ord_pair< T >::change_first ( const T & val ) [inline]`

Replace the first member of the pair by *val*, while keeping the relative order.

#### Postcondition

*first\_ <= second\_* (with *<=* being the [mln::util::ord\\_weak](#) relationship).

Definition at line 181 of file ord\_pair.hh.

References [mln::util::ord\\_strict\(\)](#), and [mln::util::ord\\_weak\(\)](#).

### 10.378.2.3 `template<typename T > void mln::util::ord_pair< T >::change_second ( const T & val ) [inline]`

Replace the second member of the pair by *val*, while keeping the relative order.

#### Postcondition

*first\_ <= second\_* (with *<=* being the [mln::util::ord\\_weak](#) relationship).

Definition at line 196 of file ord\_pair.hh.

References [mln::util::ord\\_strict\(\)](#), and [mln::util::ord\\_weak\(\)](#).

### 10.378.2.4 `template<typename T > const T & mln::util::ord_pair< T >::first ( ) const [inline]`

Get the first (lowest) member of the pair.

Definition at line 149 of file ord\_pair.hh.

### 10.378.2.5 `template<typename T > const T & mln::util::ord_pair< T >::second ( ) const [inline]`

Get the second (highest) member of the pair.

Definition at line 165 of file ord\_pair.hh.

## 10.379 mln::util::pix< I > Struct Template Reference

Structure pix.

```
#include <pix.hh>
```

### Public Types

- typedef I::psite [psite](#)  
*Point\_Site* associated type.
- typedef I::value [value](#)  
*Value* associated type.

### Public Member Functions

- const I & [ima](#) () const  
*The getter of the image associate to pix structure.*
- const I::psite & [p](#) () const  
*The getter of psite associate to pix structure.*
- [pix](#) (const [Image](#)< I > &ima, const typename I::psite &p)  
*Constructor.*
- I::rvalue [v](#) () const  
*The getter of value associate to pix structure.*

#### 10.379.1 Detailed Description

```
template<typename I> struct mln::util::pix< I >
```

Structure pix.

Definition at line 69 of file util/pix.hh.

#### 10.379.2 Member Typedef Documentation

**10.379.2.1** `template<typename I> typedef I ::psite mln::util::pix< I >::psite`

[Point\\_Site](#) associated type.

Definition at line 73 of file util/pix.hh.

**10.379.2.2** `template<typename I> typedef I ::value mln::util::pix< I >::value`

[Value](#) associated type.

Definition at line 76 of file util/pix.hh.

### 10.379.3 Constructor & Destructor Documentation

**10.379.3.1** `template<typename I> mln::util::pix< I >::pix ( const Image< I > & ima, const typename I::psite & p ) [inline]`

Constructor.

#### Parameters

[in] *ima* The image.

[in] *p* The p\_site.

Definition at line 121 of file util/pix.hh.

### 10.379.4 Member Function Documentation

**10.379.4.1** `template<typename I> const I & mln::util::pix< I >::ima ( ) const [inline]`

The getter of the image associate to pix structure.

#### Returns

The image ima\_.

Definition at line 131 of file util/pix.hh.

**10.379.4.2** `template<typename I> const I::psite & mln::util::pix< I >::p ( ) const [inline]`

The getter of psite associate to pix structure.

#### Returns

The psite p\_.

Definition at line 140 of file util/pix.hh.

**10.379.4.3** `template<typename I> I::rvalue mln::util::pix< I >::v ( ) const [inline]`

The getter of value associate to pix structure.

#### Returns

The value of pix.

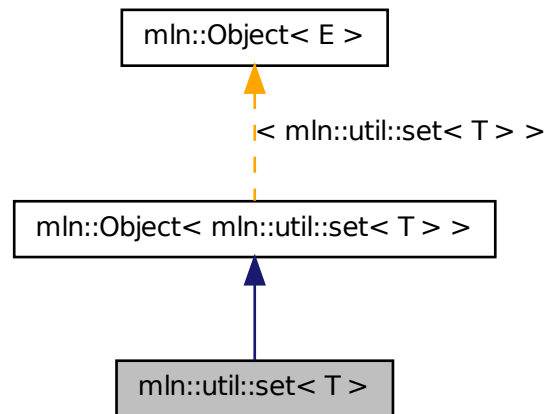
Definition at line 150 of file util/pix.hh.

## 10.380 mln::util::set< T > Class Template Reference

An "efficient" mathematical set class.

```
#include <set.hh>
```

Inheritance diagram for `mln::util::set< T >`:



## Public Types

- typedef `set_bkd_iter< T >` `bkd_eiter`  
*Backward iterator associated type.*
- typedef `fwd_eiter` `eiter`  
*Iterator associated type.*
- typedef `T` `element`  
*Element associated type.*
- typedef `set_fwd_iter< T >` `fwd_eiter`  
*Forward iterator associated type.*

## Public Member Functions

- void `clear ()`  
*Empty the set.*
- const `T` `first_element ()` const  
*Return the first element of the set.*
- bool `has (const T &elt)` const  
*Test if the object `elt` belongs to the set.*
- `set< T >` & `insert (const T &elt)`



*Insert an element  $e \in T$  into the set.*

- `template<typename U >`  
`set< T > & insert (const set< U > &other)`

*Insert the elements of `other` into the set.*

- `bool is_empty () const`  
*Test if the set is empty.*
- `const T last_element () const`  
*Return the last element of the set.*
- `std::size_t memory_size () const`  
*Return the size of this set in memory.*
- `unsigned nelements () const`  
*Return the number of elements of the set.*
- `const T & operator[] (unsigned i) const`  
*Return the  $i$ -th element of the set.*
- `set< T > & remove (const T &elt)`  
*Remove an element  $e \in T$  into the set.*
- `set ()`  
*Constructor without arguments.*
- `const std::vector< T > & std_vector () const`  
*Give access to the set elements.*

### 10.380.1 Detailed Description

**template<typename T> class mln::util::set< T >**

An "efficient" mathematical set class. This set class is designed to store a mathematical set and to present it to the user as a linear array (`std::vector`).

Elements are stored by copy. Implementation is lazy.

The set has two states: frozen or not. There is an automatic switch of state when the user modifies its contents (insert, remove, or clear) or access to its contents (`op[i]`).

The parameter `T` is the element type, which shall not be `const`-qualified.

The unicity of set elements is handled by the `mln::util::ord` mechanism.

#### See also

[mln::util::ord](#)

Definition at line 81 of file `util/set.hh`.

## 10.380.2 Member Typedef Documentation

### 10.380.2.1 `template<typename T> typedef set_bkd_iter<T> mln::util::set< T >::bkd_eiter`

Backward iterator associated type.

Definition at line 93 of file util/set.hh.

### 10.380.2.2 `template<typename T> typedef fwd_eiter mln::util::set< T >::eiter`

[Iterator](#) associated type.

Definition at line 96 of file util/set.hh.

### 10.380.2.3 `template<typename T> typedef T mln::util::set< T >::element`

Element associated type.

Definition at line 86 of file util/set.hh.

### 10.380.2.4 `template<typename T> typedef set_fwd_iter<T> mln::util::set< T >::fwd_eiter`

Forward iterator associated type.

Definition at line 90 of file util/set.hh.

## 10.380.3 Constructor & Destructor Documentation

### 10.380.3.1 `template<typename T> mln::util::set< T >::set ( ) [inline]`

Constructor without arguments.

Definition at line 348 of file util/set.hh.

## 10.380.4 Member Function Documentation

### 10.380.4.1 `template<typename T> void mln::util::set< T >::clear ( ) [inline]`

Empty the set.

All elements contained in the set are destroyed so the set is emptied.

#### Postcondition

`is_empty() == true`

Definition at line 390 of file util/set.hh.

References `mln::util::set< T >::is_empty()`.

### 10.380.4.2 `template<typename T> const T mln::util::set< T >::first_element ( ) const [inline]`

Return the first element of the set.

**Precondition**

not `is_empty()`

Definition at line 427 of file util/set.hh.

References `mln::util::set< T >::is_empty()`.

**10.380.4.3 `template<typename T > bool mln::util::set< T >::has ( const T & elt ) const [inline]`**

Test if the object `elt` belongs to the set.

**Parameters**

[in] *elt* A possible element of the set.

**Returns**

True is `elt` is in the set.

Definition at line 445 of file util/set.hh.

**10.380.4.4 `template<typename T > set< T > & mln::util::set< T >::insert ( const T & elt ) [inline]`**

Insert an element `elt` into the set.

**Parameters**

[in] *elt* The element to be inserted.

If `elt` is already in the set, this method is a no-op.

**Returns**

The set itself after insertion.

Definition at line 356 of file util/set.hh.

Referenced by `mln::p_key< K, P >::change_keys()`.

**10.380.4.5 `template<typename T > template<typename U > set< T > & mln::util::set< T >::insert ( const set< U > & other ) [inline]`**

Insert the elements of `other` into the set.

**Parameters**

[in] *other* The set containing the elements to be inserted.

**Returns**

The set itself after insertion.

Definition at line 367 of file util/set.hh.

References `mln::util::set< T >::is_empty()`, and `mln::util::set< T >::std_vector()`.

**10.380.4.6** `template<typename T> bool mln::util::set< T >::is_empty ( ) const [inline]`

Test if the set is empty.

Definition at line 453 of file util/set.hh.

References `mln::util::set< T >::nelements()`.

Referenced by `mln::util::set< T >::clear()`, `mln::util::set< T >::first_element()`, `mln::util::set< T >::insert()`, and `mln::util::set< T >::last_element()`.

**10.380.4.7** `template<typename T> const T mln::util::set< T >::last_element ( ) const [inline]`

Return the last element of the set.

**Precondition**

not `is_empty()`

Definition at line 436 of file util/set.hh.

References `mln::util::set< T >::is_empty()`.

**10.380.4.8** `template<typename T> std::size_t mln::util::set< T >::memory_size ( ) const [inline]`

Return the size of this set in memory.

Definition at line 494 of file util/set.hh.

References `mln::util::set< T >::nelements()`.

**10.380.4.9** `template<typename T> unsigned mln::util::set< T >::nelements ( ) const [inline]`

Return the number of elements of the set.

Definition at line 409 of file util/set.hh.

Referenced by `mln::util::set< T >::is_empty()`, `mln::util::set< T >::memory_size()`, and `mln::util::set< T >::operator[]()`.

**10.380.4.10** `template<typename T> const T & mln::util::set< T >::operator[] ( unsigned i ) const [inline]`

Return the *i*-th element of the set.

**Parameters**

[in] *i* Index of the element to retrieve.

**Precondition**

*i* < `nelements()`

The element is returned by reference and is constant.

Definition at line 417 of file util/set.hh.

References mln::util::set< T >::nelements().

#### 10.380.4.11 `template<typename T > set< T > & mln::util::set< T >::remove ( const T & elt )` `[inline]`

Remove an element `elt` into the set.

##### Parameters

[in] *elt* The element to be inserted.

If `elt` is already in the set, this method is a no-op.

##### Returns

The set itself after suppression.

Definition at line 380 of file util/set.hh.

#### 10.380.4.12 `template<typename T > const std::vector< T > & mln::util::set< T >::std_vector ( ) const` `[inline]`

Give access to the set elements.

The complexity of this method is O(1).

##### Postcondition

The set is frozen.

##### Returns

An array (std::vector) of elements.

Definition at line 461 of file util/set.hh.

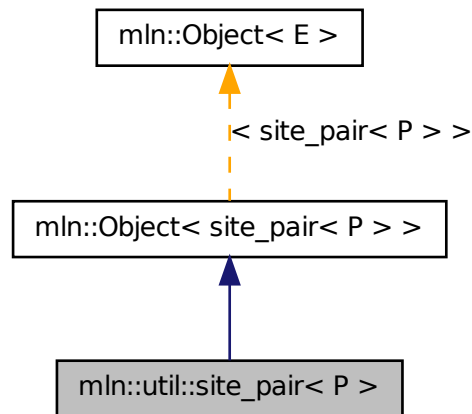
Referenced by mln::util::set< T >::insert().

## 10.381 mln::util::site\_pair< P > Class Template Reference

A pair of sites.

```
#include <site_pair.hh>
```

Inheritance diagram for `mln::util::site_pair< P >`:



## Public Member Functions

- `const P & first () const`  
*Return the first site.*
- `const util::ord_pair< P > & pair () const`  
*Return the underlying pair.*
- `const P & second () const`  
*Return the second site.*

### 10.381.1 Detailed Description

`template<typename P> class mln::util::site_pair< P >`

A pair of sites. It can be used as site.

Definition at line 52 of file `site_pair.hh`.

### 10.381.2 Member Function Documentation

**10.381.2.1** `template<typename P > const P & mln::util::site_pair< P >::first ( ) const`  
[inline]

Return the first site.

Definition at line 142 of file `site_pair.hh`.

**10.381.2.2** `template<typename P > const util::ord_pair< P > & mln::util::site_pair< P >::pair ( ) const [inline]`

Return the underlying pair.

Definition at line 158 of file site\_pair.hh.

**10.381.2.3** `template<typename P > const P & mln::util::site_pair< P >::second ( ) const [inline]`

Return the second site.

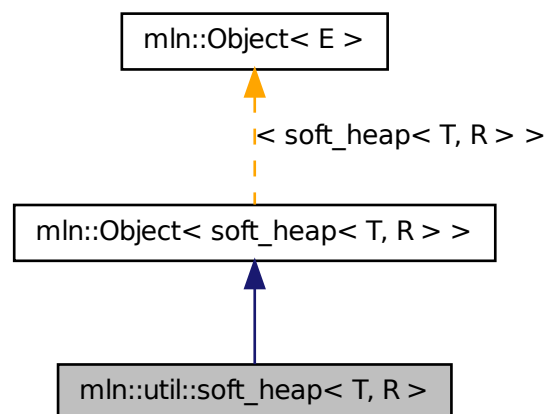
Definition at line 150 of file site\_pair.hh.

## 10.382 mln::util::soft\_heap< T, R > Class Template Reference

Soft heap.

```
#include <soft_heap.hh>
```

Inheritance diagram for mln::util::soft\_heap< T, R >:



### Public Types

- typedef T [element](#)  
*Element associated type.*

### Public Member Functions

- void [clear](#) ()

*Clear the heap.*

- `bool is_empty () const`  
*Return true if there is at least one element.*
- `bool is_valid () const`  
*Return true if there is at least one element.*
- `int nelements () const`  
*Return the number of element in the heap.*
- `T pop_front ()`  
*Returns the element with the lowest priority and remove it from the heap.*
- `void push (soft_heap< T, R > &sh)`  
*Merge sh with this heap.*
- `void push (const T &element)`  
*Add a new element element.*
- `soft_heap (unsigned r=20)`  
*Default constructor.*
- `~soft_heap ()`  
*Destructor.*

### 10.382.1 Detailed Description

`template<typename T, typename R> class mln::util::soft_heap< T, R >`

Soft heap. T key, the data to store in the heap. For instance a point 2d. R rank, for instance int\_u8

Definition at line 178 of file soft\_heap.hh.

### 10.382.2 Member Typedef Documentation

**10.382.2.1** `template<typename T, typename R> typedef T mln::util::soft_heap< T, R >::element`

Element associated type.

Definition at line 185 of file soft\_heap.hh.

### 10.382.3 Constructor & Destructor Documentation

**10.382.3.1** `template<typename T , typename R > mln::util::soft_heap< T, R >::soft_heap ( unsigned r = 20 ) [inline]`

Default constructor.



A corruption threshold  $x$  can be specified. This threshold means that if nodes have a rank higher than this threshold they can be "corrupted" and therefore their rank can be reduced.

Definition at line 619 of file soft\_heap.hh.

**10.382.3.2** `template<typename T , typename R > mln::util::soft_heap< T, R >::~~soft_heap ( )`  
`[inline]`

Destructor.

Definition at line 631 of file soft\_heap.hh.

## 10.382.4 Member Function Documentation

**10.382.4.1** `template<typename T , typename R > void mln::util::soft_heap< T, R >::clear ( )`  
`[inline]`

Clear the heap.

Definition at line 771 of file soft\_heap.hh.

**10.382.4.2** `template<typename T , typename R > bool mln::util::soft_heap< T, R >::is_empty ( ) const` `[inline]`

Return true if there is at least one element.

Definition at line 753 of file soft\_heap.hh.

**10.382.4.3** `template<typename T , typename R > bool mln::util::soft_heap< T, R >::is_valid ( ) const` `[inline]`

Return true if there is at least one element.

Definition at line 744 of file soft\_heap.hh.

Referenced by `mln::util::soft_heap< T, R >::pop_front()`.

**10.382.4.4** `template<typename T , typename R > int mln::util::soft_heap< T, R >::nelements ( ) const` `[inline]`

Return the number of element in the heap.

Definition at line 762 of file soft\_heap.hh.

Referenced by `mln::util::soft_heap< T, R >::push()`.

**10.382.4.5** `template<typename T , typename R > T mln::util::soft_heap< T, R >::pop_front ( )`  
`[inline]`

Returns the element with the lowest priority and remove it from the heap.

Definition at line 675 of file soft\_heap.hh.

References `mln::util::soft_heap< T, R >::is_valid()`.

**10.382.4.6** `template<typename T, typename R > void mln::util::soft_heap< T, R >::push ( const T & element ) [inline]`

Add a new element `element`.

Definition at line 646 of file `soft_heap.hh`.

**10.382.4.7** `template<typename T, typename R > void mln::util::soft_heap< T, R >::push ( soft_heap< T, R > & sh ) [inline]`

Merge `sh` with this heap.

Be ware that after this call, `sh` will be empty. This heap will hold the elements which were part of `sh`.

Definition at line 658 of file `soft_heap.hh`.

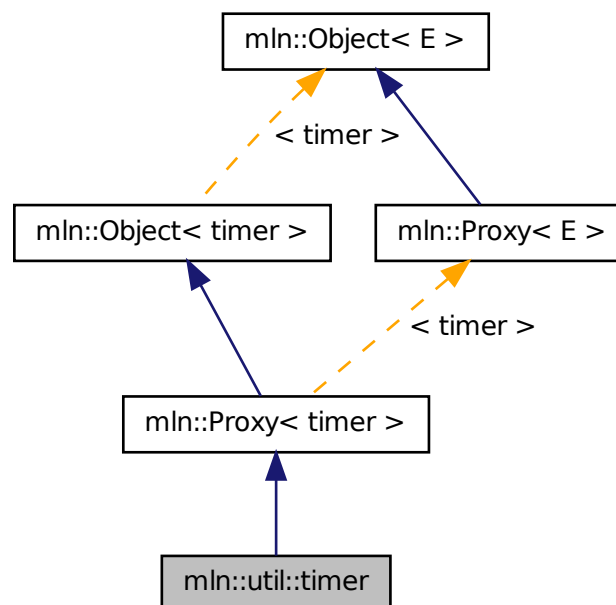
References `mln::util::soft_heap< T, R >::nelements()`.

## 10.383 mln::util::timer Class Reference

Timer structure.

```
#include <timer.hh>
```

Inheritance diagram for `mln::util::timer`:



### 10.383.1 Detailed Description

Timer structure.

Definition at line 45 of file mln/util/timer.hh.

## 10.384 mln::util::tracked\_ptr< T > Struct Template Reference

Smart pointer for shared data with tracking.

```
#include <tracked_ptr.hh>
```

### Public Member Functions

- `operator bool () const`  
*Coercion towards Boolean (for arithmetical tests).*
- `bool operator! () const`  
*Negation (for arithmetical tests).*
- `T * operator-> ()`  
*Mimics the behavior of op-> for a pointer in the mutable case.*
- `const T * operator-> () const`  
*Mimics the behavior of op-> for a pointer in the const case.*
- `tracked_ptr< T > & operator= (T *ptr)`  
*Assignment.*
- `tracked_ptr< T > & operator= (const tracked_ptr< T > &rhs)`  
*Assignment.*
- `~tracked_ptr ()`  
*Destructor.*
- `tracked_ptr ()`  
*Constructors.*
- `tracked_ptr (const tracked_ptr< T > &rhs)`  
*Copy constructor.*

### 10.384.1 Detailed Description

```
template<typename T> struct mln::util::tracked_ptr< T >
```

Smart pointer for shared data with tracking.

Definition at line 52 of file tracked\_ptr.hh.

## 10.384.2 Constructor & Destructor Documentation

### 10.384.2.1 `template<typename T> mln::util::tracked_ptr< T >::tracked_ptr ( ) [inline]`

Constructors.

Definition at line 140 of file tracked\_ptr.hh.

### 10.384.2.2 `template<typename T> mln::util::tracked_ptr< T >::tracked_ptr ( const tracked_ptr< T > & rhs ) [inline]`

Copy constructor.

Definition at line 164 of file tracked\_ptr.hh.

### 10.384.2.3 `template<typename T> mln::util::tracked_ptr< T >::~~tracked_ptr ( ) [inline]`

Destructor.

Definition at line 216 of file tracked\_ptr.hh.

## 10.384.3 Member Function Documentation

### 10.384.3.1 `template<typename T> mln::util::tracked_ptr< T >::operator bool ( ) const [inline]`

Coercion towards Boolean (for arithmetical tests).

Definition at line 106 of file tracked\_ptr.hh.

### 10.384.3.2 `template<typename T> bool mln::util::tracked_ptr< T >::operator! ( ) const [inline]`

Negation (for arithmetical tests).

Definition at line 114 of file tracked\_ptr.hh.

### 10.384.3.3 `template<typename T> T * mln::util::tracked_ptr< T >::operator-> ( ) [inline]`

Mimics the behavior of `op->` for a pointer in the mutable case.

#### Invariant

Pointer proxy exists.

Definition at line 131 of file tracked\_ptr.hh.

### 10.384.3.4 `template<typename T> const T * mln::util::tracked_ptr< T >::operator-> ( ) const [inline]`

Mimics the behavior of `op->` for a pointer in the const case.

**Invariant**

Pointer proxy exists.

Definition at line 122 of file tracked\_ptr.hh.

**10.384.3.5** `template<typename T > tracked_ptr< T > & mln::util::tracked_ptr< T >::operator=( T * ptr ) [inline]`

Assignment.

Definition at line 195 of file tracked\_ptr.hh.

**10.384.3.6** `template<typename T > tracked_ptr< T > & mln::util::tracked_ptr< T >::operator=( const tracked_ptr< T > & rhs ) [inline]`

Assignment.

Definition at line 176 of file tracked\_ptr.hh.

## 10.385 mln::util::tree< T > Class Template Reference

Class of generic tree.

```
#include <tree.hh>
```

### Public Member Functions

- void `add_tree_down` (T &elt)  
*Bind a new tree downer the current.*
- void `add_tree_up` (T &elt)  
*Bind a new tree upper the current.*
- bool `check_consistency` ()  
*Check the consistency of the tree.*
- `branch< T > main_branch` ()  
*Convert the tree into brach.*
- `tree_node< T > * root` ()  
*The getter of the root.*
- `tree` ()  
*Constructor.*
- `tree` (`tree_node< T > *root`)  
*Constructor.*

### 10.385.1 Detailed Description

**template<typename T> class mln::util::tree< T >**

Class of generic tree.

Definition at line 187 of file tree.hh.

### 10.385.2 Constructor & Destructor Documentation

**10.385.2.1 template<typename T > mln::util::tree< T >::tree ( ) [inline]**

Constructor.

Definition at line 285 of file tree.hh.

**10.385.2.2 template<typename T > mln::util::tree< T >::tree ( tree\_node< T > \* root ) [inline]**

Constructor.

#### Parameters

[in] *root* The root of the tree.

Definition at line 292 of file tree.hh.

### 10.385.3 Member Function Documentation

**10.385.3.1 template<typename T > void mln::util::tree< T >::add\_tree\_down ( T & elt ) [inline]**

Bind a new tree downer the current.

#### Parameters

[in] *elt* The new value of the new [tree\\_node](#) of the new tree add downer the current.

Definition at line 328 of file tree.hh.

**10.385.3.2 template<typename T > void mln::util::tree< T >::add\_tree\_up ( T & elt ) [inline]**

Bind a new tree upper the current.

#### Parameters

[in] *elt* The new value of the new [tree\\_node](#) of the new tree add upper the current.

Definition at line 317 of file tree.hh.

References [mln::util::tree\\_node< T >::children\(\)](#).

### 10.385.3.3 `template<typename T> bool mln::util::tree< T >::check_consistency ( )` **[inline]**

Check the consistency of the tree.

#### Returns

true if no error, else false.

Definition at line 338 of file tree.hh.

References mln::util::tree< T >::root().

### 10.385.3.4 `template<typename T> branch< T > mln::util::tree< T >::main_branch ( )` **[inline]**

Convert the tree into brach.

#### Returns

The root's [tree\\_node](#) of the the current tree.

Definition at line 309 of file tree.hh.

References mln::util::tree< T >::root().

### 10.385.3.5 `template<typename T> tree_node< T > * mln::util::tree< T >::root ( )` **[inline]**

The getter of the root.

#### Returns

The root's [tree\\_node](#) of the the current tree.

Definition at line 301 of file tree.hh.

Referenced by mln::util::tree< T >::check\_consistency(), mln::util::display\_tree(), mln::util::tree< T >::main\_branch(), and mln::util::tree\_to\_fast().

## 10.386 mln::util::tree\_node< T > Class Template Reference

Class of generic [tree\\_node](#) for tree.

```
#include <tree.hh>
```

### Public Member Functions

- [tree\\_node< T > \\* add\\_child \(T elt\)](#)  
*Create a [tree\\_node](#) with `elt` which become the child of the current [tree\\_node](#).*
- [tree\\_node< T > \\* add\\_child \(tree\\_node< T > \\*tree\\_node\)](#)

Bind `tree_node` to the current `tree_node` and become its child.

- `bool check_consistency ()`  
Check the consistency of the `tree_node`.
- `children_t & children ()`  
The getter of the children.
- `const children_t & children () const`  
The getter of the children.
- `tree_node< T > * delete_tree_node ()`  
Delete the current `tree_node`.
- `T & elt ()`  
The getter of the element.
- `const T & elt () const`  
The const getter of the element.
- `tree_node< T > * parent ()`  
The getter of the parent.
- `void print (std::ostream &ostr, int level=0)`  
Print on `ostr` the arborescence with the current `tree_node` as root.
- `tree_node< T > * search (T &elt)`  
Search the `tree_node` with value `elt` in the arborescence of the current `tree_node`.
- `int search_rec (tree_node< T > **res, T &elt)`  
The using method for method search.
- `void set_parent (tree_node< T > *parent)`  
Bind `tree_node` to the current `tree_node` and become its parent.
- `tree_node ()`  
Constructor.
- `tree_node (T elt)`  
Constructor.

### 10.386.1 Detailed Description

```
template<typename T> class mln::util::tree_node< T >
```

Class of generic `tree_node` for tree.

Definition at line 58 of file tree.hh.



## 10.386.2 Constructor & Destructor Documentation

### 10.386.2.1 `template<typename T > mln::util::tree_node< T >::tree_node ( ) [inline]`

Constructor.

Definition at line 345 of file tree.hh.

### 10.386.2.2 `template<typename T > mln::util::tree_node< T >::tree_node ( T elt ) [inline]`

Constructor.

#### Parameters

[in] *elt* The element of [tree\\_node](#).

Definition at line 352 of file tree.hh.

## 10.386.3 Member Function Documentation

### 10.386.3.1 `template<typename T > tree_node< T > * mln::util::tree_node< T >::add_child ( T elt ) [inline]`

Create a [tree\\_node](#) with *elt* which become the child of the current [tree\\_node](#).

#### Parameters

[in] *elt* The element of the new child to add.

#### Returns

The new [tree\\_node](#) created.

Definition at line 394 of file tree.hh.

### 10.386.3.2 `template<typename T > tree_node< T > * mln::util::tree_node< T >::add_child ( tree_node< T > * tree_node ) [inline]`

Bind [tree\\_node](#) to the current [tree\\_node](#) and become its child.

#### Parameters

[in] *tree\_node* The new child [tree\\_node](#).

#### Returns

The child [tree\\_node](#).

Definition at line 407 of file tree.hh.

References [mln::util::tree\\_node< T >::children\(\)](#), and [mln::util::tree\\_node< T >::parent\(\)](#).

**10.386.3.3** `template<typename T> bool mln::util::tree_node< T >::check_consistency ( )`  
`[inline]`

Check the consistency of the [tree\\_node](#).

#### Returns

true if no error, else false.

Definition at line 519 of file tree.hh.

**10.386.3.4** `template<typename T> const std::vector< tree_node< T > * > &`  
`mln::util::tree_node< T >::children ( ) const [inline]`

The getter of the children.

#### Returns

The children of the [tree\\_node](#) in const.

Definition at line 386 of file tree.hh.

**10.386.3.5** `template<typename T> std::vector< tree_node< T > * > & mln::util::tree_node< T`  
`>::children ( ) [inline]`

The getter of the children.

#### Returns

The children of the [tree\\_node](#).

Definition at line 378 of file tree.hh.

Referenced by `mln::util::tree_node< T >::add_child()`, and `mln::util::tree< T >::add_tree_up()`.

**10.386.3.6** `template<typename T> tree_node< T > * mln::util::tree_node< T`  
`>::delete_tree_node ( ) [inline]`

Delete the current [tree\\_node](#).

Definition at line 427 of file tree.hh.

**10.386.3.7** `template<typename T> const T & mln::util::tree_node< T >::elt ( ) const`  
`[inline]`

The const getter of the element.

#### Returns

The element of the [tree\\_node](#) in const.

Definition at line 361 of file tree.hh.

**10.386.3.8** `template<typename T> T & mln::util::tree_node< T >::elt ( ) [inline]`

The getter of the element.

**Returns**

The element of the [tree\\_node](#).

Definition at line 369 of file tree.hh.

Referenced by `mln::util::tree_node< T >::print()`.

**10.386.3.9** `template<typename T> tree_node< T > * mln::util::tree_node< T >::parent ( ) [inline]`

The getter of the parent.

**Returns**

The parent of the [tree\\_node](#).

Definition at line 477 of file tree.hh.

Referenced by `mln::util::tree_node< T >::add_child()`, `mln::util::branch_iter_ind< T >::deepness()`, and `mln::util::branch_iter< T >::deepness()`.

**10.386.3.10** `template<typename T> void mln::util::tree_node< T >::print ( std::ostream & ostr, int level = 0 ) [inline]`

Print on `ostr` the arborescence with the current [tree\\_node](#) as root.

**Parameters**

[in] *ostr* The output stream.

[in] *level* The deep level

Definition at line 449 of file tree.hh.

References `mln::util::tree_node< T >::elt()`.

**10.386.3.11** `template<typename T> tree_node< T > * mln::util::tree_node< T >::search ( T & elt ) [inline]`

Search the [tree\\_node](#) with value `elt` in the arborescence of the current [tree\\_node](#).

**Parameters**

[in] *elt* The value of the searched [tree\\_node](#).

**Returns**

If not found 0 else the [tree\\_node](#) with `elt` value.

Definition at line 507 of file tree.hh.

References `mln::util::tree_node< T >::search_rec()`.

**10.386.3.12** `template<typename T> int mln::util::tree_node< T >::search_rec ( tree_node< T > ** res, T & elt ) [inline]`

The using method for method search.

Definition at line 485 of file tree.hh.

Referenced by `mln::util::tree_node< T >::search()`.

**10.386.3.13** `template<typename T> void mln::util::tree_node< T >::set_parent ( tree_node< T > * parent ) [inline]`

Bind `tree_node` to the current `tree_node` and become its parent.

#### Parameters

[in] *parent* The new parent `tree_node`.

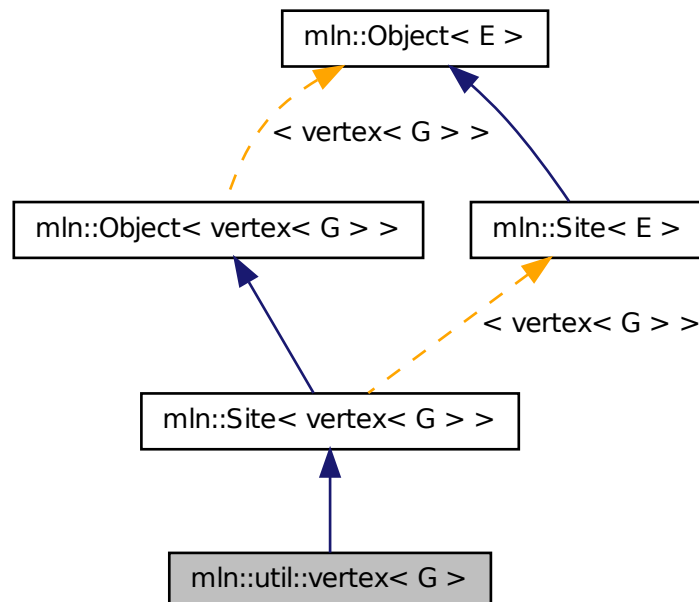
Definition at line 467 of file tree.hh.

## 10.387 mln::util::vertex< G > Class Template Reference

`Vertex` of a graph `G`.

```
#include <vertex.hh>
```

Inheritance diagram for `mln::util::vertex< G >`:



## Public Types

- typedef [Vertex](#)< void > [Category](#)  
*Object category.*
- typedef G [graph\\_t](#)  
*Graph associated type.*
- typedef [vertex\\_id\\_t](#) [id\\_t](#)  
*The vertex type id.*
- typedef [vertex\\_id\\_t::value\\_t](#) [id\\_value\\_t](#)  
*The underlying type used to store vertex ids.*

## Public Member Functions

- void [change\\_graph](#) (const G &g)  
*Change the parent graph of that vertex.*
- [edge](#)< G > [edge\\_with](#) (const [vertex](#)< G > &v\_id) const  
*Returns true if this vertex has an edge with the given vertex.*
- const G & [graph](#) () const  
*Returns the graph pointer this vertex belongs to.*
- const [vertex\\_id\\_t](#) & [id](#) () const  
*Returns the vertex id.*
- void [invalidate](#) ()  
*Invalidate that vertex.*
- bool [is\\_valid](#) () const  
*Check whether the vertex is still part of the graph.*
- [edge\\_id\\_t](#) [ith\\_nbh\\_edge](#) (unsigned i) const  
*Returns the ith edge starting from this vertex.*
- [vertex\\_id\\_t](#) [ith\\_nbh\\_vertex](#) (unsigned i) const  
*Returns the ith vertex adjacent to this vertex.*
- unsigned [nmax\\_nbh\\_edges](#) () const  
*Returns the number max of edges starting from this vertex.*
- unsigned [nmax\\_nbh\\_vertices](#) () const  
*Returns the number max of vertices adjacent to this vertex.*
- operator [vertex\\_id\\_t](#) () const  
*Conversion to the vertex id.*

- [vertex\\_id\\_t other](#) (const [edge\\_id\\_t](#) &id\_e) const  
*Returns the other vertex located on edge id\_e.*
- void [update\\_id](#) (const [vertex\\_id\\_t](#) &id)  
*Update the vertex id.*
- [vertex](#) ()  
*Constructors.*

### 10.387.1 Detailed Description

**template<typename G> class mln::util::vertex< G >**

[Vertex](#) of a graph G.

Definition at line 71 of file vertex.hh.

### 10.387.2 Member Typedef Documentation

**10.387.2.1 template<typename G> typedef Vertex<void> mln::util::vertex< G >::Category**

[Object](#) category.

Definition at line 77 of file vertex.hh.

**10.387.2.2 template<typename G> typedef G mln::util::vertex< G >::graph\_t**

[Graph](#) associated type.

Definition at line 86 of file vertex.hh.

**10.387.2.3 template<typename G> typedef vertex\_id\_t mln::util::vertex< G >::id\_t**

The vertex type id.

Definition at line 83 of file vertex.hh.

**10.387.2.4 template<typename G> typedef vertex\_id\_t::value\_t mln::util::vertex< G >::id\_value\_t**

The underlying type used to store vertex ids.

Definition at line 80 of file vertex.hh.

### 10.387.3 Constructor & Destructor Documentation

**10.387.3.1 template<typename G> mln::util::vertex< G >::vertex ( ) [inline]**

Constructors.

Definition at line 226 of file vertex.hh.

References mln::util::vertex< G >::invalidate().

## 10.387.4 Member Function Documentation

### 10.387.4.1 `template<typename G > void mln::util::vertex< G >::change_graph ( const G & g ) [inline]`

Change the parent graph of that vertex.

Definition at line 331 of file vertex.hh.

### 10.387.4.2 `template<typename G > edge< G > mln::util::vertex< G >::edge_with ( const vertex< G > & v_id ) const [inline]`

Returns true if this vertex has an edge with the given vertex.

Definition at line 321 of file vertex.hh.

### 10.387.4.3 `template<typename G > const G & mln::util::vertex< G >::graph ( ) const [inline]`

Returns the graph pointer this vertex belongs to.

Definition at line 348 of file vertex.hh.

Referenced by mln::p\_vertices< G, F >::has(), mln::util::line\_graph< G >::has(), and mln::util::operator==( ).

### 10.387.4.4 `template<typename G > const vertex_id_t & mln::util::vertex< G >::id ( ) const [inline]`

Returns the vertex id.

Definition at line 356 of file vertex.hh.

Referenced by mln::util::line\_graph< G >::has(), and mln::util::operator==( ).

### 10.387.4.5 `template<typename G > void mln::util::vertex< G >::invalidate ( ) [inline]`

Invalidate that vertex.

Definition at line 266 of file vertex.hh.

Referenced by mln::util::vertex< G >::vertex().

### 10.387.4.6 `template<typename G > bool mln::util::vertex< G >::is_valid ( ) const [inline]`

Check whether the vertex is still part of the graph.

Definition at line 258 of file vertex.hh.

Referenced by mln::p\_vertices< G, F >::has().

**10.387.4.7** `template<typename G > edge_id_t mln::util::vertex< G >::ith_nbh_edge ( unsigned i ) const [inline]`

Returns the ith edge starting from this vertex.

Definition at line 285 of file vertex.hh.

**10.387.4.8** `template<typename G > vertex_id_t mln::util::vertex< G >::ith_nbh_vertex ( unsigned i ) const [inline]`

Returns the ith vertex adjacent to this vertex.

Definition at line 303 of file vertex.hh.

**10.387.4.9** `template<typename G > unsigned mln::util::vertex< G >::nmax_nbh_edges ( ) const [inline]`

Returns the number max of edges starting from this vertex.

If `g_` is a sub graph of another graph, `nmax` will be retrieved from the initial graph.

Definition at line 294 of file vertex.hh.

**10.387.4.10** `template<typename G > unsigned mln::util::vertex< G >::nmax_nbh_vertices ( ) const [inline]`

Returns the number max of vertices adjacent to this vertex.

Definition at line 312 of file vertex.hh.

**10.387.4.11** `template<typename G > mln::util::vertex< G >::operator vertex_id_t ( ) const [inline]`

Conversion to the vertex id.

FIXME: May cause ambiguities... :(

Definition at line 363 of file vertex.hh.

**10.387.4.12** `template<typename G > vertex_id_t mln::util::vertex< G >::other ( const edge_id_t & id_e ) const [inline]`

Returns the other vertex located on edge `id_e`.

Definition at line 274 of file vertex.hh.

**10.387.4.13** `template<typename G > void mln::util::vertex< G >::update_id ( const vertex_id_t & id ) [inline]`

Update the vertex id.

Definition at line 340 of file vertex.hh.

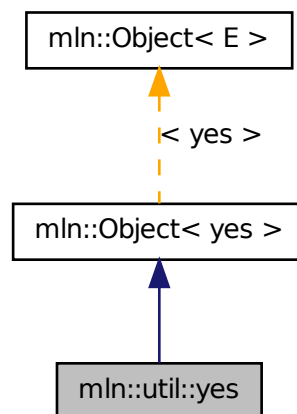


## 10.388 mln::util::yes Struct Reference

[Object](#) that always says "yes".

```
#include <yes.hh>
```

Inheritance diagram for mln::util::yes:



### 10.388.1 Detailed Description

[Object](#) that always says "yes".

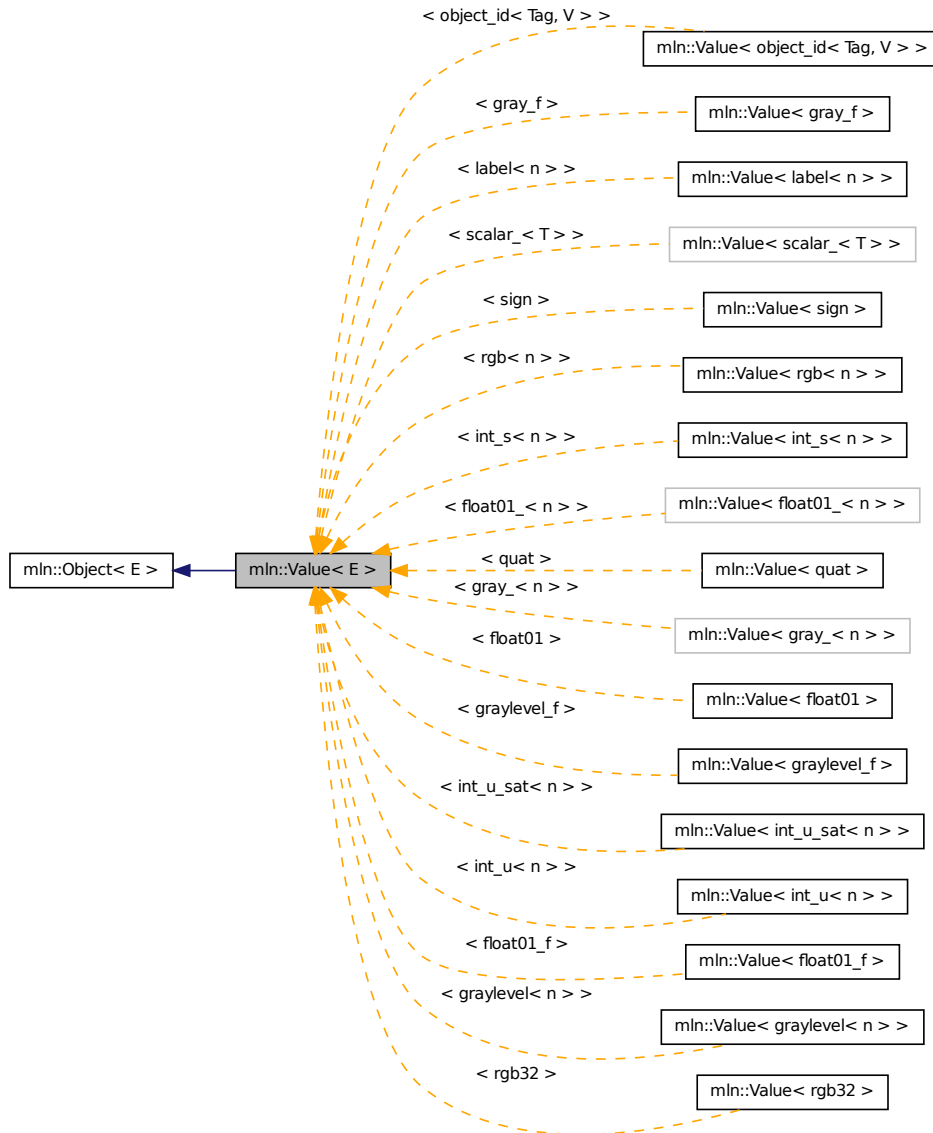
Definition at line 76 of file `yes.hh`.

## 10.389 mln::Value< E > Struct Template Reference

Base class for implementation classes of values.

```
#include <value.hh>
```

Inheritance diagram for `mln::Value< E >`:



### 10.389.1 Detailed Description

```
template<typename E> struct mln::Value< E >
```

Base class for implementation classes of values.

#### See also

`mln::doc::Value` for a complete documentation of this class contents.

Definition at line 57 of file core/concept/value.hh.

## 10.390 mln::value::float01 Class Reference

Class for floating values restricted to the interval [0..1] and discretized with n bits.

```
#include <float01.hh>
```

Inherits mln::value::Floating< float01 >.

### Public Types

- typedef std::pair< unsigned, unsigned long > [enc](#)  
*Encoding associated type.*
- typedef float [equiv](#)  
*Equivalent associated type.*

### Public Member Functions

- [float01](#) ()  
*Ctor.*
- template<unsigned n>  
[float01](#) (const float01\_< n > &val)  
*Ctor.*
- [float01](#) (unsigned nbits, float val)  
*Ctor.*
- unsigned [nbits](#) () const  
*Access to the encoding size.*
- operator float () const  
*Conversion to float.*
- [float01](#) & [set\\_nbits](#) (unsigned nbits)  
*Set the encoding size to nbits.*
- const [float01 to\\_nbits](#) (unsigned nbits) const  
*Return an equivalent gray encoded on nbits bits.*
- float [value](#) () const  
*Access to std type.*
- unsigned long [value\\_ind](#) () const  
*Access to the position in the quantized interval.*

### 10.390.1 Detailed Description

Class for floating values restricted to the interval [0..1] and discretized with n bits.

Definition at line 56 of file float01.hh.

### 10.390.2 Member Typedef Documentation

#### 10.390.2.1 `typedef std::pair<unsigned, unsigned long> mln::value::float01::enc`

Encoding associated type.

Definition at line 61 of file float01.hh.

#### 10.390.2.2 `typedef float mln::value::float01::equiv`

Equivalent associated type.

Definition at line 64 of file float01.hh.

### 10.390.3 Constructor & Destructor Documentation

#### 10.390.3.1 `mln::value::float01::float01 ( ) [inline]`

Ctor.

Definition at line 151 of file float01.hh.

#### 10.390.3.2 `template<unsigned n> mln::value::float01::float01 ( const float01_<n> & val ) [inline]`

Ctor.

Definition at line 158 of file float01.hh.

#### 10.390.3.3 `mln::value::float01::float01 ( unsigned nbits, float val ) [inline]`

Ctor.

Definition at line 165 of file float01.hh.

### 10.390.4 Member Function Documentation

#### 10.390.4.1 `unsigned mln::value::float01::nbits ( ) const [inline]`

Access to the encoding size.

Definition at line 186 of file float01.hh.

#### 10.390.4.2 `mln::value::float01::operator float ( ) const [inline]`

Conversion to float.

Definition at line 224 of file float01.hh.

#### 10.390.4.3 float01 & mln::value::float01::set\_nbits ( unsigned *nbits* ) [inline]

Set the encoding size to *nbits*.

Definition at line 193 of file float01.hh.

Referenced by to\_nbits().

#### 10.390.4.4 const float01 mln::value::float01::to\_nbits ( unsigned *nbits* ) const [inline]

Return an equivalent gray encoded on *nbits* bits.

Definition at line 214 of file float01.hh.

References set\_nbits().

#### 10.390.4.5 float mln::value::float01::value ( ) const [inline]

Access to std type.

Definition at line 172 of file float01.hh.

#### 10.390.4.6 unsigned long mln::value::float01::value\_ind ( ) const [inline]

Access to the position in the quantized interval.

Definition at line 179 of file float01.hh.

## 10.391 mln::value::float01\_f Struct Reference

Class for floating values restricted to the interval [0..1].

```
#include <float01_f.hh>
```

Inherits mln::value::Floating< float01\_f >, and mln::value::internal::value\_like\_< float,float,float,float01\_f >.

### Public Member Functions

- [float01\\_f\(\)](#)  
*Constructor without argument.*
- [float01\\_f\(float val\)](#)  
*Constructor from a float.*
- [operator float\(\)](#) const  
*Conversion to a float.*
- [float01\\_f & operator=\(const float val\)](#)  
*Assignment from a float.*

- float `value` () const  
*Access to float value.*

### 10.391.1 Detailed Description

Class for floating values restricted to the interval [0..1].

Definition at line 84 of file float01\_f.hh.

### 10.391.2 Constructor & Destructor Documentation

#### 10.391.2.1 `mln::value::float01_f::float01_f( )` [inline]

Constructor without argument.

Definition at line 115 of file float01\_f.hh.

#### 10.391.2.2 `mln::value::float01_f::float01_f( float val )` [inline]

Constructor from a float.

Definition at line 120 of file float01\_f.hh.

### 10.391.3 Member Function Documentation

#### 10.391.3.1 `mln::value::float01_f::operator float( ) const` [inline]

Conversion to a float.

Definition at line 145 of file float01\_f.hh.

#### 10.391.3.2 `float01_f & mln::value::float01_f::operator=( const float val )` [inline]

Assignment from a float.

Definition at line 136 of file float01\_f.hh.

#### 10.391.3.3 `float mln::value::float01_f::value( ) const` [inline]

Access to float value.

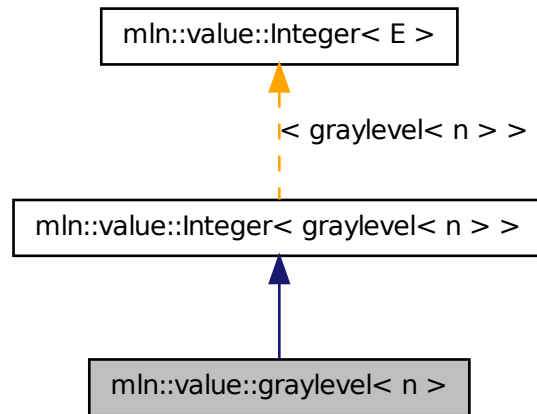
Definition at line 129 of file float01\_f.hh.

## 10.392 `mln::value::graylevel< n >` Struct Template Reference

General gray-level class on n bits.

```
#include <graylevel.hh>
```

Inheritance diagram for mln::value::graylevel< n >:



## Public Member Functions

- [graylevel \(\)](#)  
*Constructor without argument.*
- [graylevel \(const \[graylevel< n >\]\(#\) &rhs\)](#)  
*Copy constructor.*
- [graylevel \(int val\)](#)  
*Constructor from int.*
- [template<unsigned m>  
graylevel \(const \[graylevel< m >\]\(#\) &rhs\)](#)  
*Constructor from any graylevel.*
- [graylevel< n > & operator= \(const \[graylevel< n >\]\(#\) &rhs\)](#)  
*Assignment.*
- [graylevel< n > & operator= \(int val\)](#)  
*Assignment with int.*
- [template<unsigned m>  
graylevel< n > & operator= \(const \[graylevel< m >\]\(#\) &rhs\)](#)  
*Assignment with any graylevel.*
- [float \[to\\\_float \\(\\)\]\(#\) const](#)  
*Conversion to float between 0 and 1.*

- unsigned `value` () const  
*Access to std type.*
- `graylevel` (const `mln::literal::black_t` &)  
*Ctors with literals.*
- `graylevel`< n > & `operator=` (const `mln::literal::black_t` &)  
*Assigment with literals.*

### 10.392.1 Detailed Description

`template<unsigned n> struct mln::value::graylevel< n >`

General gray-level class on n bits.

Definition at line 257 of file `graylevel.hh`.

### 10.392.2 Constructor & Destructor Documentation

**10.392.2.1** `template<unsigned n> mln::value::graylevel< n >::graylevel ( ) [inline]`

Constructor without argument.

Definition at line 436 of file `graylevel.hh`.

**10.392.2.2** `template<unsigned n> mln::value::graylevel< n >::graylevel ( const graylevel< n > & rhs ) [inline]`

Copy constructor.

Definition at line 463 of file `graylevel.hh`.

**10.392.2.3** `template<unsigned n> mln::value::graylevel< n >::graylevel ( int val ) [inline]`

Constructor from int.

Definition at line 443 of file `graylevel.hh`.

**10.392.2.4** `template<unsigned n> template<unsigned m> mln::value::graylevel< n >::graylevel ( const graylevel< m > & rhs ) [inline]`

Constructor from any graylevel.

Definition at line 481 of file `graylevel.hh`.

References `mln::value::graylevel< n >::value()`.



**10.392.2.5** `template<unsigned n> mln::value::graylevel< n >::graylevel ( const mln::literal::black_t & ) [inline]`

ctors with literals.

Definition at line 499 of file graylevel.hh.

### 10.392.3 Member Function Documentation

**10.392.3.1** `template<unsigned n> graylevel< n > & mln::value::graylevel< n >::operator= ( const graylevel< n > & rhs ) [inline]`

Assignment.

Definition at line 472 of file graylevel.hh.

**10.392.3.2** `template<unsigned n> graylevel< n > & mln::value::graylevel< n >::operator= ( int val ) [inline]`

Assignment with int.

Definition at line 453 of file graylevel.hh.

**10.392.3.3** `template<unsigned n> graylevel< n > & mln::value::graylevel< n >::operator= ( const mln::literal::black_t & ) [inline]`

Assignment with literals.

Definition at line 507 of file graylevel.hh.

**10.392.3.4** `template<unsigned n> template<unsigned m> graylevel< n > & mln::value::graylevel< n >::operator= ( const graylevel< m > & rhs ) [inline]`

Assignment with any graylevel.

Definition at line 490 of file graylevel.hh.

References mln::value::graylevel< n >::value().

**10.392.3.5** `template<unsigned n> float mln::value::graylevel< n >::to_float ( ) const [inline]`

Conversion to float between 0 and 1.

Definition at line 557 of file graylevel.hh.

Referenced by mln::value::graylevel\_f::graylevel\_f(), and mln::value::graylevel\_f::operator=().

**10.392.3.6** `template<unsigned n> unsigned mln::value::graylevel< n >::value ( ) const [inline]`

Access to std type.

Definition at line 549 of file graylevel.hh.

Referenced by `mln::value::graylevel<n>::graylevel()`, and `mln::value::graylevel<n>::operator=()`.

## 10.393 mln::value::graylevel\_f Struct Reference

General gray-level class on n bits.

```
#include <graylevel_f.hh>
```

Inherits `mln::value::Floating<graylevel_f>`, and `mln::value::internal::value_like_<float01_f, float01_f::enc, internal::gray_f, graylevel_f>`.

### Public Member Functions

- [graylevel\\_f\(\)](#)  
*Constructor without argument.*
- [graylevel\\_f\(const graylevel\\_f &rhs\)](#)  
*Copy constructor.*
- [graylevel\\_f\(float val\)](#)  
*Constructor from float.*
- `template<unsigned n>`  
[graylevel\\_f\(const graylevel<n> &rhs\)](#)  
*Constructor from graylevel.*
- `template<unsigned n>`  
[operator graylevel<n>\(\) const](#)  
*Conversion to graylevel<n>.*
- [graylevel\\_f & operator=\(float val\)](#)  
*Assignment with float.*
- [graylevel\\_f & operator=\(const graylevel\\_f &rhs\)](#)  
*Assignment.*
- `template<unsigned n>`  
[graylevel\\_f & operator=\(const graylevel<n> &rhs\)](#)  
*Assignment with graylevel.*
- `float value() const`  
*Access to std type.*
- [graylevel\\_f\(const mln::literal::black\\_t &\)](#)  
*Ctors with literals.*
- [graylevel\\_f & operator=\(const mln::literal::black\\_t &\)](#)  
*Assignment with literals.*

### 10.393.1 Detailed Description

General gray-level class on n bits.

Definition at line 193 of file graylevel\_f.hh.

### 10.393.2 Constructor & Destructor Documentation

#### 10.393.2.1 mln::value::graylevel\_f::graylevel\_f ( ) [inline]

Constructor without argument.

Definition at line 341 of file graylevel\_f.hh.

#### 10.393.2.2 mln::value::graylevel\_f::graylevel\_f ( const graylevel\_f & rhs ) [inline]

Copy constructor.

Definition at line 383 of file graylevel\_f.hh.

#### 10.393.2.3 mln::value::graylevel\_f::graylevel\_f ( float val ) [inline]

Constructor from float.

Definition at line 347 of file graylevel\_f.hh.

#### 10.393.2.4 template<unsigned n> mln::value::graylevel\_f::graylevel\_f ( const graylevel< n > & rhs )

Constructor from graylevel.

Definition at line 365 of file graylevel\_f.hh.

References mln::value::graylevel< n >::to\_float().

#### 10.393.2.5 mln::value::graylevel\_f::graylevel\_f ( const mln::literal::black\_t & ) [inline]

Ctors with literals.

Definition at line 400 of file graylevel\_f.hh.

### 10.393.3 Member Function Documentation

#### 10.393.3.1 template<unsigned n> mln::value::graylevel\_f::operator graylevel< n > ( ) const [inline]

Conversion to graylevel<n>.

Definition at line 443 of file graylevel\_f.hh.

#### 10.393.3.2 graylevel\_f & mln::value::graylevel\_f::operator= ( float val ) [inline]

Assignment with float.

Definition at line 356 of file graylevel\_f.hh.

**10.393.3.3** `template<unsigned n> graylevel_f & mln::value::graylevel_f::operator= ( const graylevel< n > & rhs )`

Assignment with graylevel.

Definition at line 374 of file graylevel\_f.hh.

References `mln::value::graylevel< n >::to_float()`.

**10.393.3.4** `graylevel_f & mln::value::graylevel_f::operator= ( const mln::literal::black_t & ) [inline]`

Assignment with literals.

Definition at line 407 of file graylevel\_f.hh.

**10.393.3.5** `graylevel_f & mln::value::graylevel_f::operator= ( const graylevel_f & rhs ) [inline]`

Assignment.

Definition at line 391 of file graylevel\_f.hh.

**10.393.3.6** `float mln::value::graylevel_f::value ( ) const [inline]`

Access to std type.

Definition at line 450 of file graylevel\_f.hh.

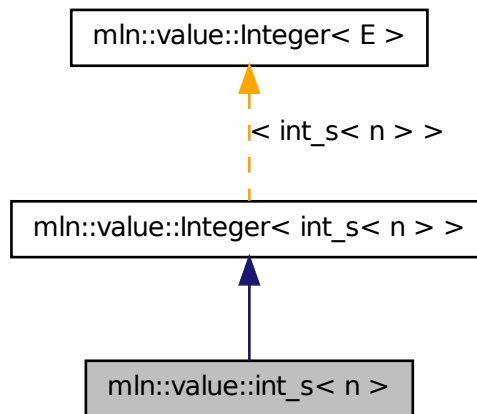
Referenced by `mln::value::operator<<()`.

## 10.394 `mln::value::int_s< n >` Struct Template Reference

Signed integer value class.

```
#include <int_s.hh>
```

Inheritance diagram for mln::value::int\_s< n >:



## Public Member Functions

- `int_s ()`  
*Constructor without argument.*
- `int_s (int i)`  
*Constructor from an integer.*
- `operator int () const`  
*Conversion to an integer.*
- `int_s< n > & operator= (int i)`  
*Assignment from an integer.*
- `int_s (const mln::literal::zero_t &)`  
*Constructors/assignments with literals.*

## Static Public Attributes

- static const `int_s< n > one = 1`  
*Unit value.*
- static const `int_s< n > zero = 0`  
*Zero value.*

### 10.394.1 Detailed Description

**template<unsigned n> struct mln::value::int\_s< n >**

Signed integer value class. The parameter is n the number of encoding bits.

Definition at line 115 of file int\_s.hh.

### 10.394.2 Constructor & Destructor Documentation

**10.394.2.1 template<unsigned n> mln::value::int\_s< n >::int\_s ( ) [inline]**

Constructor without argument.

Definition at line 179 of file int\_s.hh.

**10.394.2.2 template<unsigned n> mln::value::int\_s< n >::int\_s ( int i ) [inline]**

Constructor from an integer.

Definition at line 192 of file int\_s.hh.

**10.394.2.3 template<unsigned n> mln::value::int\_s< n >::int\_s ( const mln::literal::zero\_t & ) [inline]**

Constructors/assignments with literals.

Definition at line 222 of file int\_s.hh.

### 10.394.3 Member Function Documentation

**10.394.3.1 template<unsigned n> mln::value::int\_s< n >::operator int ( ) const [inline]**

Conversion to an integer.

Definition at line 185 of file int\_s.hh.

**10.394.3.2 template<unsigned n> int\_s< n > & mln::value::int\_s< n >::operator= ( int i ) [inline]**

Assignment from an integer.

Definition at line 207 of file int\_s.hh.

### 10.394.4 Member Data Documentation

**10.394.4.1 template<unsigned n> const int\_s< n > mln::value::int\_s< n >::one = 1 [static]**

Unit value.

Definition at line 149 of file int\_s.hh.

**10.394.4.2** `template<unsigned n> const int_s< n > mln::value::int_s< n >::zero = 0 [static]`

Zero value.

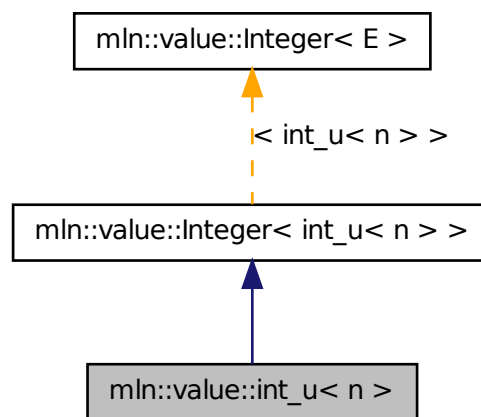
Definition at line 146 of file int\_s.hh.

## 10.395 mln::value::int\_u< n > Struct Template Reference

Unsigned integer value class.

```
#include <int_u.hh>
```

Inheritance diagram for mln::value::int\_u< n >:



### Public Member Functions

- `int_u ()`  
*Constructor without argument.*
- `int_u (int i)`  
*Constructor from an integer.*
- `int_u< n > next () const`  
*Give the next value (i.e., i + 1).*
- `operator unsigned () const`  
*Conversion to an unsigned integer.*
- `int operator- () const`  
*Unary operator minus.*

- `int_u<n> & operator= (int i)`  
*Assignment from an integer.*
- `int_u (const mln::literal::zero_t &)`  
*Constructors/assignments with literals.*

### 10.395.1 Detailed Description

`template<unsigned n> struct mln::value::int_u<n>`

Unsigned integer value class. The parameter is `n` the number of encoding bits.

Definition at line 156 of file `int_u.hh`.

### 10.395.2 Constructor & Destructor Documentation

**10.395.2.1** `template<unsigned n> mln::value::int_u<n>::int_u ( ) [inline]`

Constructor without argument.

Definition at line 276 of file `int_u.hh`.

**10.395.2.2** `template<unsigned n> mln::value::int_u<n>::int_u ( int i ) [inline]`

Constructor from an integer.

Definition at line 282 of file `int_u.hh`.

**10.395.2.3** `template<unsigned n> mln::value::int_u<n>::int_u ( const mln::literal::zero_t & ) [inline]`

Constructors/assignments with literals.

Definition at line 291 of file `int_u.hh`.

### 10.395.3 Member Function Documentation

**10.395.3.1** `template<unsigned n> int_u<n> mln::value::int_u<n>::next ( ) const [inline]`

Give the next value (i.e.,  $i + 1$ ).

Definition at line 350 of file `int_u.hh`.

**10.395.3.2** `template<unsigned n> mln::value::int_u<n>::operator unsigned ( ) const [inline]`

Conversion to an unsigned integer.



Definition at line 323 of file int\_u.hh.

**10.395.3.3** `template<unsigned n> int mln::value::int_u< n >::operator-( ) const [inline]`

Unary operator minus.

Definition at line 331 of file int\_u.hh.

**10.395.3.4** `template<unsigned n> int_u< n > & mln::value::int_u< n >::operator=( int i ) [inline]`

Assignment from an integer.

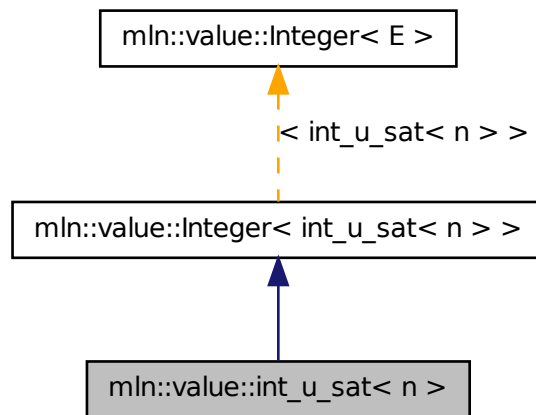
Definition at line 339 of file int\_u.hh.

## 10.396 mln::value::int\_u\_sat< n > Struct Template Reference

Unsigned integer value class with saturation behavior.

```
#include <int_u_sat.hh>
```

Inheritance diagram for mln::value::int\_u\_sat< n >:



### Public Member Functions

- `int_u_sat ()`  
*Constructor without argument.*
- `int_u_sat (int i)`  
*Constructor from an integer.*

- `operator int () const`  
*Conversion to an integer.*
- `int_u_sat< n > & operator+= (int i)`  
*Self addition.*
- `int_u_sat< n > & operator-= (int i)`  
*Self subtraction.*
- `int_u_sat< n > & operator= (int i)`  
*Assignment from an integer.*

### Static Public Attributes

- static const `int_u_sat< n > one = 1`  
*Unit value.*
- static const `int_u_sat< n > zero = 0`  
*Zero value.*

## 10.396.1 Detailed Description

`template<unsigned n> struct mln::value::int_u_sat< n >`

Unsigned integer value class with saturation behavior. The parameter is `n` the number of encoding bits.  
Definition at line 90 of file `int_u_sat.hh`.

## 10.396.2 Constructor & Destructor Documentation

**10.396.2.1** `template<unsigned n> mln::value::int_u_sat< n >::int_u_sat ( ) [inline]`

Constructor without argument.

Definition at line 149 of file `int_u_sat.hh`.

**10.396.2.2** `template<unsigned n> mln::value::int_u_sat< n >::int_u_sat ( int i ) [inline]`

Constructor from an integer.

Definition at line 155 of file `int_u_sat.hh`.

## 10.396.3 Member Function Documentation

**10.396.3.1** `template<unsigned n> mln::value::int_u_sat< n >::operator int ( ) const [inline]`

Conversion to an integer.

Definition at line 170 of file int\_u\_sat.hh.

**10.396.3.2** `template<unsigned n> int_u_sat< n > & mln::value::int_u_sat< n >::operator+=( int i ) [inline]`

Self addition.

Definition at line 195 of file int\_u\_sat.hh.

**10.396.3.3** `template<unsigned n> int_u_sat< n > & mln::value::int_u_sat< n >::operator-= ( int i ) [inline]`

Self subtraction.

Definition at line 205 of file int\_u\_sat.hh.

**10.396.3.4** `template<unsigned n> int_u_sat< n > & mln::value::int_u_sat< n >::operator= ( int i ) [inline]`

Assignment from an integer.

Definition at line 178 of file int\_u\_sat.hh.

## 10.396.4 Member Data Documentation

**10.396.4.1** `template<unsigned n> const int_u_sat< n > mln::value::int_u_sat< n >::one = 1 [static]`

Unit value.

Definition at line 115 of file int\_u\_sat.hh.

**10.396.4.2** `template<unsigned n> const int_u_sat< n > mln::value::int_u_sat< n >::zero = 0 [static]`

Zero value.

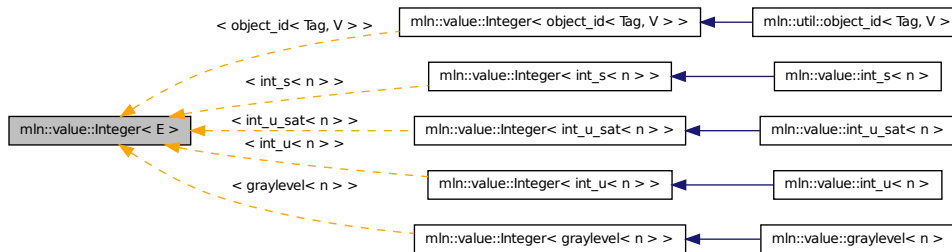
Definition at line 112 of file int\_u\_sat.hh.

## 10.397 mln::value::Integer< E > Struct Template Reference

Concept of integer.

```
#include <integer.hh>
```

Inheritance diagram for `mln::value::Integer< E >`:



### 10.397.1 Detailed Description

`template<typename E> struct mln::value::Integer< E >`

Concept of integer.

Definition at line 58 of file `concept/integer.hh`.

## 10.398 `mln::value::Integer< void >` Struct Template Reference

Category flag type.

```
#include <integer.hh>
```

### 10.398.1 Detailed Description

`template<> struct mln::value::Integer< void >`

Category flag type.

Definition at line 50 of file `concept/integer.hh`.

## 10.399 `mln::value::label< n >` Struct Template Reference

Label value class.

```
#include <label.hh>
```

Inherits `mln::value::Symbolic< label< n > >`, and `mln::value::internal::value_like< unsigned,internal::encoding_unsigned< n >::ret,int,label< n > >`.

### Public Types

- `typedef internal::encoding_unsigned< n >::ret enc`  
*Encoding associated type.*

## Public Member Functions

- `label ()`  
*Constructor without argument.*
- `label (unsigned i)`  
*Constructor from an (unsigned) integer.*
- `label (const literal::zero_t &v)`  
*Constructor from `literal::zero`.*
- `label< n > next () const`  
*Return the next value.*
- `operator unsigned () const`  
*Conversion to an unsigned integer.*
- `label< n > & operator++ ()`  
*Self increment.*
- `label< n > & operator-- ()`  
*Self decrement.*
- `label< n > & operator= (unsigned i)`  
*Assignment from an (unsigned) integer.*
- `label< n > & operator= (const literal::zero_t &v)`  
*Assignment from `literal::zero`.*
- `label< n > prev () const`  
*Return the previous value.*

### 10.399.1 Detailed Description

`template<unsigned n> struct mln::value::label< n >`

Label value class. The parameter `n` is the number of encoding bits.

Definition at line 140 of file `label.hh`.

### 10.399.2 Member Typedef Documentation

**10.399.2.1** `template<unsigned n> typedef internal::encoding_unsigned_<n>::ret mln::value::label< n >::enc`

Encoding associated type.

Definition at line 150 of file `label.hh`.

### 10.399.3 Constructor & Destructor Documentation

#### 10.399.3.1 `template<unsigned n> mln::value::label< n >::label ( ) [inline]`

Constructor without argument.

Definition at line 271 of file label.hh.

#### 10.399.3.2 `template<unsigned n> mln::value::label< n >::label ( unsigned i ) [inline]`

Constructor from an (unsigned) integer.

Definition at line 277 of file label.hh.

#### 10.399.3.3 `template<unsigned n> mln::value::label< n >::label ( const literal::zero_t & v ) [inline]`

Constructor from [literal::zero](#).

Definition at line 284 of file label.hh.

### 10.399.4 Member Function Documentation

#### 10.399.4.1 `template<unsigned n> label< n > mln::value::label< n >::next ( ) const [inline]`

Return the next value.

Definition at line 338 of file label.hh.

#### 10.399.4.2 `template<unsigned n> mln::value::label< n >::operator unsigned ( ) const [inline]`

Conversion to an unsigned integer.

Definition at line 291 of file label.hh.

#### 10.399.4.3 `template<unsigned n> label< n > & mln::value::label< n >::operator++ ( ) [inline]`

Self increment.

Definition at line 318 of file label.hh.

#### 10.399.4.4 `template<unsigned n> label< n > & mln::value::label< n >::operator-- ( ) [inline]`

Self decrement.

Definition at line 328 of file label.hh.

**10.399.4.5** `template<unsigned n> label< n > & mln::value::label< n >::operator= ( unsigned i ) [inline]`

Assignment from an (unsigned) integer.

Definition at line 299 of file label.hh.

**10.399.4.6** `template<unsigned n> label< n > & mln::value::label< n >::operator= ( const literal::zero_t & v ) [inline]`

Assignment from `literal::zero`.

Definition at line 309 of file label.hh.

**10.399.4.7** `template<unsigned n> label< n > mln::value::label< n >::prev ( ) const [inline]`

Return the previous value.

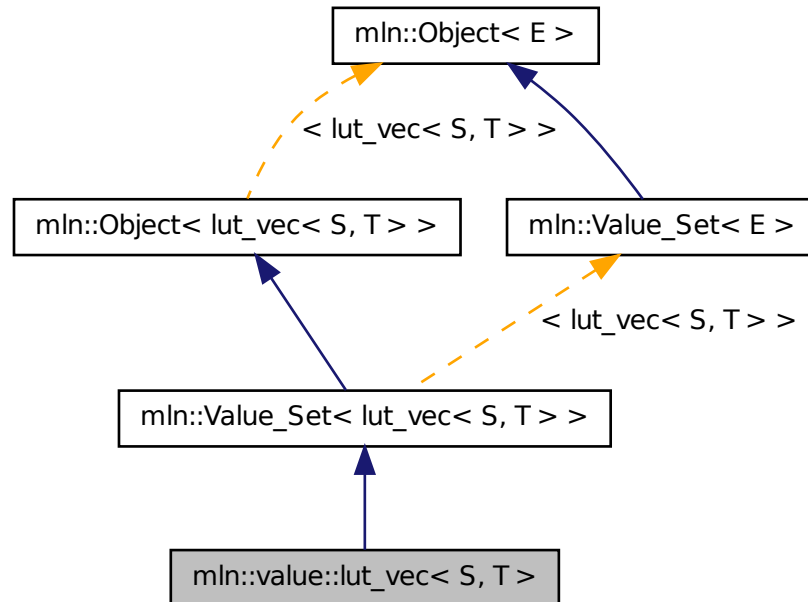
Definition at line 346 of file label.hh.

## 10.400 mln::value::lut\_vec< S, T > Struct Template Reference

Class that defines FIXME.

```
#include <lut_vec.hh>
```

Inheritance diagram for `mln::value::lut_vec< S, T >`:



## Public Types

- typedef `bkd_viter_< lut_vec< S, T > >` `bkd_viter`  
*Backward Value\_Iterator associated type.*
- typedef `fwd_viter_< lut_vec< S, T > >` `fwd_viter`  
*Forward Value\_Iterator associated type.*
- typedef `T` `value`  
*Value associated type.*

## Public Member Functions

- `bool` `has` (const `value` &`v`) const  
*Test if `v` belongs to this set.*
- `unsigned` `index_of` (const `value` &`v`) const  
*Give the index of value `v` in this set.*
- `unsigned` `nvalues` () const



*Give the number of values.*

- `T operator[]` (unsigned `i`) `const`  
*Give the  $i$ -th value.*
- `template<typename F >`  
`lut_vec` (`const S &vset`, `const Function_v2v< F > &f`)  
*Constructors*  
*Constructor from a value set and any `Function_v2v`.*
- `template<typename V >`  
`lut_vec` (`const S &vset`, `const Function_v2v< fun::i2v::array< V > > &f`)  
*Constructor from a value set and any `fun::i2v::array`.*
- `template<typename V >`  
`lut_vec` (`const S &vset`, `const Function_v2v< util::array< V > > &f`)  
*Constructor from a value set and any `util::array`.*

### 10.400.1 Detailed Description

`template<typename S, typename T> struct mln::value::lut_vec< S, T >`

Class that defines FIXME.

#### Warning

This is a multi-set!!! FIXME

Definition at line 71 of file `lut_vec.hh`.

### 10.400.2 Member Typedef Documentation

**10.400.2.1** `template<typename S , typename T > typedef bkd_viter_< lut_vec<S,T> >`  
`mln::value::lut_vec< S, T >::bkd_viter`

Backward [Value\\_Iterator](#) associated type.

Definition at line 80 of file `lut_vec.hh`.

**10.400.2.2** `template<typename S , typename T > typedef fwd_viter_< lut_vec<S,T> >`  
`mln::value::lut_vec< S, T >::fwd_viter`

Forward [Value\\_Iterator](#) associated type.

Definition at line 77 of file `lut_vec.hh`.

**10.400.2.3** `template<typename S , typename T > typedef T mln::value::lut_vec< S, T >::value`

[Value](#) associated type.

Definition at line 74 of file `lut_vec.hh`.

### 10.400.3 Constructor & Destructor Documentation

**10.400.3.1** `template<typename S , typename T > template<typename F > mln::value::lut_vec< S, T >::lut_vec ( const S & vset, const Function_v2v< F > & f ) [inline]`

Constructors

Constructor from a value set and any [Function\\_v2v](#).

Definition at line 148 of file lut\_vec.hh.

**10.400.3.2** `template<typename S , typename T > template<typename V > mln::value::lut_vec< S, T >::lut_vec ( const S & vset, const Function_v2v< fun::i2v::array< V > > & f ) [inline]`

Constructor from a value set and any [fun::i2v::array](#).

Definition at line 161 of file lut\_vec.hh.

**10.400.3.3** `template<typename S , typename T > template<typename V > mln::value::lut_vec< S, T >::lut_vec ( const S & vset, const Function_v2v< util::array< V > > & f ) [inline]`

Constructor from a value set and any [util::array](#).

Definition at line 172 of file lut\_vec.hh.

References [mln::util::array< T >::size\(\)](#), and [mln::util::array< T >::std\\_vector\(\)](#).

### 10.400.4 Member Function Documentation

**10.400.4.1** `template<typename S , typename T > bool mln::value::lut_vec< S, T >::has ( const value & v ) const`

Test if *v* belongs to this set.

**10.400.4.2** `template<typename S , typename T > unsigned mln::value::lut_vec< S, T >::index_of ( const value & v ) const`

Give the index of value *v* in this set.

**10.400.4.3** `template<typename S , typename T > unsigned mln::value::lut_vec< S, T >::nvalues ( ) const [inline]`

Give the number of values.

Definition at line 202 of file lut\_vec.hh.

Referenced by [mln::value::lut\\_vec< S, T >::operator\[\]\(\)](#).

**10.400.4.4** `template<typename S, typename T> T mln::value::lut_vec< S, T >::operator[] ( unsigned i ) const [inline]`

Give the  $i$ -th value.

Definition at line 193 of file lut\_vec.hh.

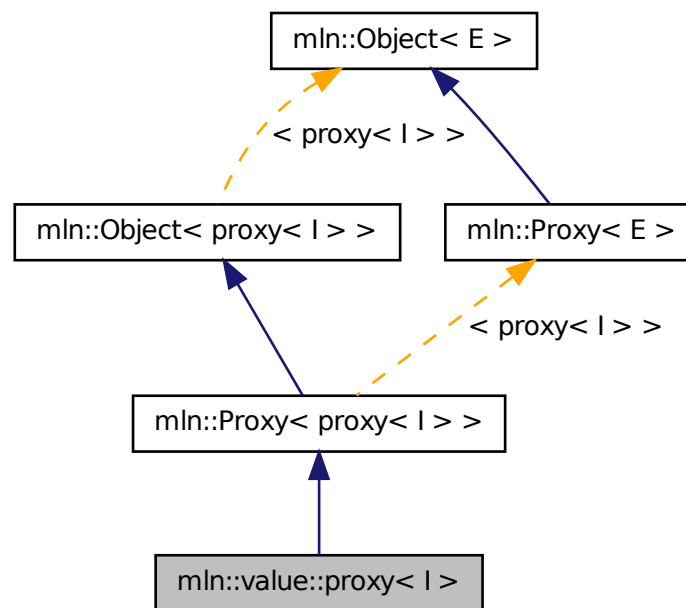
References `mln::value::lut_vec< S, T >::nvalues()`.

## 10.401 mln::value::proxy< I > Class Template Reference

Generic proxy class for an image pixel value.

```
#include <proxy.hh>
```

Inheritance diagram for `mln::value::proxy< I >`:



### Public Types

- typedef void `enc`  
*Encoding associated type.*
- typedef `I::value` `equiv`  
*Equivalent associated type.*

## Public Member Functions

- `proxy< I > & operator=` (const `proxy< I > &rhs`)  
*Assignment (write access); replacement for default op.*
- `template<typename J > proxy< I > & operator=` (const `proxy< J > &rhs`)  
*Assignment (write access); with other proxy.*
- `proxy ()`  
*Constructor.*
- `proxy (I &ima, const typename I::psite &p)`  
*Constructor.*
- `I::value to_value ()` const  
*Explicit read access.*
- `~proxy ()`  
*Destructor.*

### 10.401.1 Detailed Description

`template<typename I> class mln::value::proxy< I >`

Generic proxy class for an image pixel value. The parameter `I` is an image type.

Definition at line 85 of file `value/proxy.hh`.

### 10.401.2 Member Typedef Documentation

**10.401.2.1** `template<typename I> typedef void mln::value::proxy< I >::enc`

Encoding associated type.

Definition at line 91 of file `value/proxy.hh`.

**10.401.2.2** `template<typename I> typedef I ::value mln::value::proxy< I >::equiv`

Equivalent associated type.

Definition at line 94 of file `value/proxy.hh`.

### 10.401.3 Constructor & Destructor Documentation

**10.401.3.1** `template<typename I> mln::value::proxy< I >::proxy ( ) [inline]`

Constructor.

Definition at line 150 of file `value/proxy.hh`.

**10.401.3.2** `template<typename I > mln::value::proxy< I >::proxy ( I & ima, const typename I::psite & p ) [inline]`

Constructor.

Definition at line 157 of file value/proxy.hh.

**10.401.3.3** `template<typename I > mln::value::proxy< I >::~~proxy ( ) [inline]`

Destructor.

Definition at line 165 of file value/proxy.hh.

## 10.401.4 Member Function Documentation

**10.401.4.1** `template<typename I > proxy< I > & mln::value::proxy< I >::operator= ( const proxy< I > & rhs ) [inline]`

Assignment (write access); replacement for default op.

Definition at line 186 of file value/proxy.hh.

References mln::value::proxy< I >::to\_value().

**10.401.4.2** `template<typename I > template<typename J > proxy< I > & mln::value::proxy< I >::operator= ( const proxy< J > & rhs ) [inline]`

Assignment (write access); with other proxy.

Definition at line 199 of file value/proxy.hh.

References mln::value::proxy< I >::to\_value().

**10.401.4.3** `template<typename I > I::value mln::value::proxy< I >::to_value ( ) const [inline]`

Explicit read access.

Definition at line 226 of file value/proxy.hh.

Referenced by mln::value::proxy< I >::operator=().

## 10.402 mln::value::qt::rgb32 Struct Reference

Color class for red-green-blue where every component is n-bit encoded.

```
#include <rgb32.hh>
```

Inherits mln::value::Vectorial< rgb32 >, and mln::value::internal::value\_like\_< algebra::vec< 3, int\_u< 8 > >, algebra::vec< 3, int\_u< 8 > >, algebra::vec< 3, int >, rgb32 >.

### Public Member Functions

- [rgb32](#) & [operator=](#) (const [rgb32](#) &rhs)

*Assignment.*

- `rgb32` (`const algebra::vec< 3, int > &rhs`)  
*Constructor from a algebra::vec.*
- `rgb32` (`int r, int g, int b`)  
*Constructor from component values.*
- `rgb32` (`()`)  
*Constructor without argument.*
- `int_u< 8 > red` (`() const`)  
*Acces to red/green/blue component.*
- `rgb32` (`const mln::literal::zero_t &`)  
*Constructors with literals.*

## Static Public Attributes

- static const `rgb32 zero`  
*Zero value.*

### 10.402.1 Detailed Description

Color class for red-green-blue where every component is n-bit encoded.

Definition at line 197 of file rgb32.hh.

### 10.402.2 Constructor & Destructor Documentation

#### 10.402.2.1 `mln::value::qt::rgb32::rgb32 ( ) [inline]`

Constructor without argument.

Definition at line 385 of file rgb32.hh.

#### 10.402.2.2 `mln::value::qt::rgb32::rgb32 ( int r, int g, int b ) [inline]`

Constructor from component values.

Definition at line 427 of file rgb32.hh.

#### 10.402.2.3 `mln::value::qt::rgb32::rgb32 ( const algebra::vec< 3, int > & rhs ) [inline]`

Constructor from a algebra::vec.

Definition at line 391 of file rgb32.hh.

**10.402.2.4** `mln::value::qt::rgb32::rgb32 ( const mln::literal::zero_t & ) [inline]`

Constructors with literals.

Definition at line 442 of file `rgb32.hh`.

**10.402.3 Member Function Documentation****10.402.3.1** `rgb32 & mln::value::qt::rgb32::operator= ( const rgb32 & rhs ) [inline]`

Assignment.

Definition at line 623 of file `rgb32.hh`.

**10.402.3.2** `int_u<8> mln::value::qt::rgb32::red ( ) const [inline]`

Access to red/green/blue component.

Definition at line 212 of file `rgb32.hh`.

**10.402.4 Member Data Documentation****10.402.4.1** `const rgb32 mln::value::qt::rgb32::zero [static]`

Zero value.

Definition at line 272 of file `rgb32.hh`.

**10.403** `mln::value::rgb< n >` Struct Template Reference

Color class for red-green-blue where every component is n-bit encoded.

```
#include <rgb.hh>
```

Inherits `mln::value::Vectorial< rgb< n > >`, and `mln::value::internal::value_like_< algebra::vec< 3, int_u< n > >, algebra::vec< 3, int_u< n > >, algebra::vec< 3, int >, rgb< n > >`.

**Public Member Functions**

- `rgb< n > & operator= (const rgb< n > &rhs)`  
*Assignment.*
- `rgb (const algebra::vec< 3, int > &rhs)`  
*Constructor from a algebra::vec.*
- `rgb (int r, int g, int b)`  
*Constructor from component values.*
- `rgb ()`  
*Constructor without argument.*

- `int_u<n> red () const`  
*Acces to red/green/blue component.*
- `rgb (const mln::literal::white_t &)`  
*Constructors with literals.*

### Static Public Attributes

- static const `rgb<n> zero`  
*Zero value.*

### 10.403.1 Detailed Description

`template<unsigned n> struct mln::value::rgb<n>`

Color class for red-green-blue where every component is n-bit encoded.

Definition at line 248 of file value/rgb.hh.

### 10.403.2 Constructor & Destructor Documentation

**10.403.2.1** `template<unsigned n> mln::value::rgb<n>::rgb ( ) [inline]`

Constructor without argument.

Definition at line 422 of file value/rgb.hh.

**10.403.2.2** `template<unsigned n> mln::value::rgb<n>::rgb ( int r, int g, int b ) [inline]`

Constructor from component values.

Definition at line 458 of file value/rgb.hh.

**10.403.2.3** `template<unsigned n> mln::value::rgb<n>::rgb ( const algebra::vec<3, int> & rhs ) [inline]`

Constructor from a algebra::vec.

Definition at line 428 of file value/rgb.hh.

**10.403.2.4** `template<unsigned n> mln::value::rgb<n>::rgb ( const mln::literal::white_t & ) [inline]`

Constructors with literals.

Definition at line 473 of file value/rgb.hh.



### 10.403.3 Member Function Documentation

**10.403.3.1** `template<unsigned n> rgb< n > & mln::value::rgb< n >::operator= ( const rgb< n > & rhs ) [inline]`

Assignment.

Definition at line 645 of file value/rgb.hh.

**10.403.3.2** `template<unsigned n> int_u<n> mln::value::rgb< n >::red ( ) const [inline]`

Acces to red/green/blue component.

Definition at line 264 of file value/rgb.hh.

Referenced by mln::fun::v2v::rgb8\_to\_rgbn< n >::operator().

### 10.403.4 Member Data Documentation

**10.403.4.1** `template<unsigned n> const rgb< n > mln::value::rgb< n >::zero [static]`

Zero value.

Definition at line 322 of file value/rgb.hh.

## 10.404 mln::value::set< T > Struct Template Reference

Class that defines the set of values of type T.

```
#include <set.hh>
```

Inherits `set_selector_< T, set< T >, mln::metal::equal< mln::trait::value_< T >::quant, mln::trait::value::quant::low >::value >`.

### Static Public Member Functions

- static const `set< T > & the ()`

*Return a singleton.*

### 10.404.1 Detailed Description

`template<typename T> struct mln::value::set< T >`

Class that defines the set of values of type T. This is the exhaustive set of values obtainable from type T.

Definition at line 65 of file value/set.hh.

## 10.404.2 Member Function Documentation

### 10.404.2.1 `template<typename T> const set< T> & mln::value::set< T>::the ( ) [inline, static]`

Return a singleton.

Definition at line 80 of file value/set.hh.

## 10.405 mln::value::sign Class Reference

The sign class represents the value type composed by the set (-1, 0, 1) sign value type is a subset of the int value type.

```
#include <sign.hh>
```

Inherits Integer< sign >.

### Public Types

- typedef int `enc`  
*FIXME Are these typedefs correct?*
- typedef int `equiv`  
*Define the equivalent type.*

### Public Member Functions

- `operator int () const`  
*Conversion to an integer.*
- `sign & operator= (int i)`  
*Assignment from an integer.*
- `sign ()`  
*Constructor without argument.*
- `sign (int i)`  
*Constructor from an integer.*
- `sign (const mln::literal::zero_t &)`  
*Constructors/assignments with literals.*

### Static Public Attributes

- static const `sign one = 1`  
*Unit value.*

- static const `sign zero = 0`

*Zero value.*

### 10.405.1 Detailed Description

The sign class represents the value type composed by the set (-1, 0, 1) sign value type is a subset of the int value type.

Definition at line 49 of file value/sign.hh.

### 10.405.2 Member Typedef Documentation

#### 10.405.2.1 typedef int mln::value::sign::enc

FIXME Are these typedefs correct?

Define the encoding type

Definition at line 55 of file value/sign.hh.

#### 10.405.2.2 typedef int mln::value::sign::equiv

Define the equivalent type.

Definition at line 58 of file value/sign.hh.

### 10.405.3 Constructor & Destructor Documentation

#### 10.405.3.1 mln::value::sign::sign ( ) [inline]

Constructor without argument.

Definition at line 119 of file value/sign.hh.

#### 10.405.3.2 mln::value::sign::sign ( int i ) [inline]

Constructor from an integer.

Definition at line 137 of file value/sign.hh.

#### 10.405.3.3 mln::value::sign::sign ( const mln::literal::zero\_t & ) [inline]

Constructors/assignments with literals.

Definition at line 155 of file value/sign.hh.

### 10.405.4 Member Function Documentation

#### 10.405.4.1 `mln::value::sign::operator int ( ) const [inline]`

Conversion to an integer.

Definition at line 124 of file `value/sign.hh`.

#### 10.405.4.2 `sign & mln::value::sign::operator=( int i ) [inline]`

Assignment from an integer.

Definition at line 146 of file `value/sign.hh`.

### 10.405.5 Member Data Documentation

#### 10.405.5.1 `const sign mln::value::sign::one = 1 [static]`

Unit value.

Definition at line 88 of file `value/sign.hh`.

#### 10.405.5.2 `const sign mln::value::sign::zero = 0 [static]`

Zero value.

Definition at line 85 of file `value/sign.hh`.

## 10.406 `mln::value::stack_image< n, I >` Struct Template Reference

Stack image class.

```
#include <stack.hh>
```

Inherits `image_value_morpher< I, algebra::vec< n, I::value >, stack_image< n, I >>`.

### Public Types

- typedef `I::domain_t` `domain_t`  
*Site\_Set* associated type.
- typedef `internal::helper_stack_image_lvalue_< n, I >::ret` `lvalue`  
*Return type of read-write access.*
- typedef `I::psite` `psite`  
*Point\_Site* associated type.
- typedef `value` `rvalue`  
*Return type of read-only access.*
- typedef `stack_image< n, tag::image_< I >>` `skeleton`

*Skeleton.*

- typedef algebra::vec< n, typename I::value > [value](#)  
*Value associated type.*

## Public Member Functions

- bool [is\\_valid](#) () const  
*Test if this image has been initialized.*
- [lvalue operator](#)() (const [psite](#) &)  
*Read-write access of pixel value at point site p.*
- [rvalue operator](#)() (const [psite](#) &p) const  
*Read-only access of pixel value at point site p.*
- [stack\\_image](#) (const algebra::vec< n, I > &imas)  
*Constructors.*

### 10.406.1 Detailed Description

**template<unsigned n, typename I> struct mln::value::stack\_image< n, I >**

Stack image class. [mln::value::stack\\_image](#) stores a vector of n images of the same domain.

The parameter n is the number of images, I is the type of a stack element. Acces a value will compute a vector which contains n coordinates : [stack[0](p), stack[1](p), ... , stack[n](p)]

Definition at line 145 of file stack.hh.

### 10.406.2 Member Typedef Documentation

**10.406.2.1 template<unsigned n, typename I> typedef I ::domain\_t mln::value::stack\_image< n, I >::domain\_t**

[Site\\_Set](#) associated type.

Definition at line 154 of file stack.hh.

**10.406.2.2 template<unsigned n, typename I> typedef internal::helper\_  
stack\_image\_lvalue\_<n,I>::ret mln::value::stack\_image< n, I  
>::lvalue**

Return type of read-write access.

Definition at line 166 of file stack.hh.

**10.406.2.3** `template<unsigned n, typename I> typedef I ::psite mln::value::stack_image< n, I >::psite`

[Point\\_Site](#) associated type.

Definition at line 151 of file stack.hh.

**10.406.2.4** `template<unsigned n, typename I> typedef value mln::value::stack_image< n, I >::rvalue`

Return type of read-only access.

The rvalue type is not a const reference, since the value type is built on the fly, and return by value (copy).

Definition at line 163 of file stack.hh.

**10.406.2.5** `template<unsigned n, typename I> typedef stack_image< n, tag::image_<I> > mln::value::stack_image< n, I >::skeleton`

Skeleton.

Definition at line 170 of file stack.hh.

**10.406.2.6** `template<unsigned n, typename I> typedef algebra::vec<n, typename I ::value> mln::value::stack_image< n, I >::value`

[Value](#) associated type.

Definition at line 157 of file stack.hh.

### 10.406.3 Constructor & Destructor Documentation

**10.406.3.1** `template<unsigned n, typename I> mln::value::stack_image< n, I >::stack_image ( const algebra::vec< n, I > & imas ) [inline]`

Constructors.

Definition at line 236 of file stack.hh.

### 10.406.4 Member Function Documentation

**10.406.4.1** `template<unsigned n, typename I > bool mln::value::stack_image< n, I >::is_valid ( ) const [inline]`

Test if this image has been initialized.

Definition at line 255 of file stack.hh.

**10.406.4.2** `template<unsigned n, typename I > stack_image< n, I >::lvalue mln::value::stack_image< n, I >::operator() ( const psite & p ) [inline]`

Read-write access of pixel value at point site *p*.

Definition at line 296 of file stack.hh.

**10.406.4.3** `template<unsigned n, typename I > stack_image< n, I >::rvalue  
mln::value::stack_image< n, I >::operator() ( const psite & p ) const [inline]`

Read-only access of pixel value at point site p.

Definition at line 277 of file stack.hh.

## 10.407 mln::value::super\_value< sign > Struct Template Reference

Specializations:

```
#include <super_value.hh>
```

### 10.407.1 Detailed Description

```
template<> struct mln::value::super_value< sign >
```

Specializations: Sign type is a subset of the short value type.

Definition at line 56 of file super\_value.hh.

## 10.408 mln::value::value\_array< T, V > Struct Template Reference

Generic array class over indexed by a value set with type T.

```
#include <value_array.hh>
```

### Public Member Functions

- `const V & operator() (const T &v) const`  
}
- `const V & operator[] (unsigned i) const`  
}
- `value_array ()`  
*Constructors.*
- `const mln::value::set< T > & vset () const`  
}

### 10.408.1 Detailed Description

```
template<typename T, typename V> struct mln::value::value_array< T, V >
```

Generic array class over indexed by a value set with type T.

Definition at line 45 of file value\_array.hh.

## 10.408.2 Constructor & Destructor Documentation

**10.408.2.1** `template<typename T , typename V > mln::value::value_array< T, V >::value_array ( ) [inline]`

Constructors.

```
{
```

Definition at line 89 of file value\_array.hh.

## 10.408.3 Member Function Documentation

**10.408.3.1** `template<typename T , typename V > const V & mln::value::value_array< T, V >::operator()( const T & v ) const [inline]`

```
}
```

Access elements through a value of T. {

Definition at line 128 of file value\_array.hh.

**10.408.3.2** `template<typename T , typename V > const V & mln::value::value_array< T, V >::operator[]( unsigned i ) const [inline]`

```
}
```

Access elements through array indexes. {

Definition at line 152 of file value\_array.hh.

**10.408.3.3** `template<typename T , typename V > const mln::value::set< T > & mln::value::value_array< T, V >::vset ( ) const [inline]`

```
}
```

Reference to the set of T.

Definition at line 144 of file value\_array.hh.

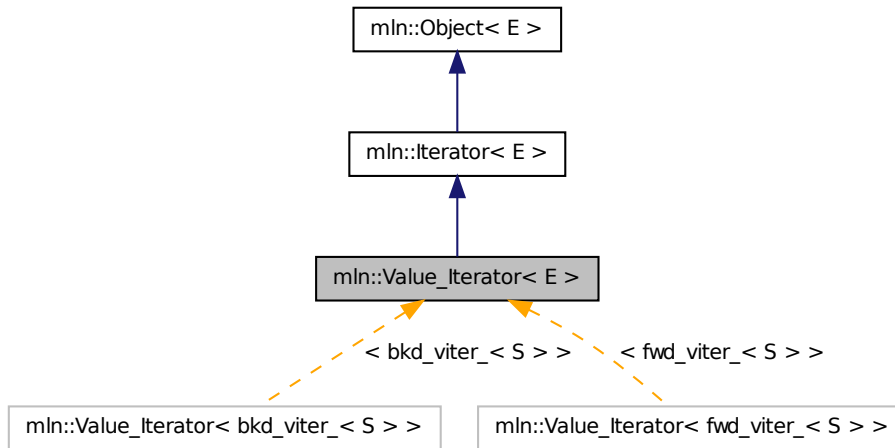
## 10.409 mln::Value\_Iterator< E > Struct Template Reference

Base class for implementation of classes of iterator on values.

```
#include <value_iterator.hh>
```



Inheritance diagram for mln::Value\_Iterator< E >:



## Public Member Functions

- void [next](#) ()  
*Go to the next element.*

## Related Functions

(Note that these are not member functions.)

- `template<typename E >`  
`std::ostream & operator<< (std::ostream &ostr, const Value\_Iterator< E > &v)`  
*Print an iterator v on value set into the output stream ostr.*

### 10.409.1 Detailed Description

`template<typename E> struct mln::Value_Iterator< E >`

Base class for implementation of classes of iterator on values. An iterator on values is an iterator that browse over a set of values.

#### See also

[mln::doc::Value\\_Iterator](#) for a complete documentation of this class contents.

Definition at line 50 of file value\_iterator.hh.

## 10.409.2 Member Function Documentation

### 10.409.2.1 `template<typename E> void mln::Iterator< E >::next ( )` `[inherited]`

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

#### Precondition

The iterator is valid.

Definition at line 102 of file iterator.hh.

## 10.409.3 Friends And Related Function Documentation

### 10.409.3.1 `template<typename E> std::ostream & operator<< ( std::ostream & ostr, const Value_Iterator< E > & v )` `[related]`

Print an iterator *v* on value set into the output stream *ostr*.

#### Parameters

`[in, out]` *ostr* An output stream.

`[in]` *v* An iterator on value set.

#### Precondition

*v* is a valid.

#### Returns

The modified output stream *ostr*.

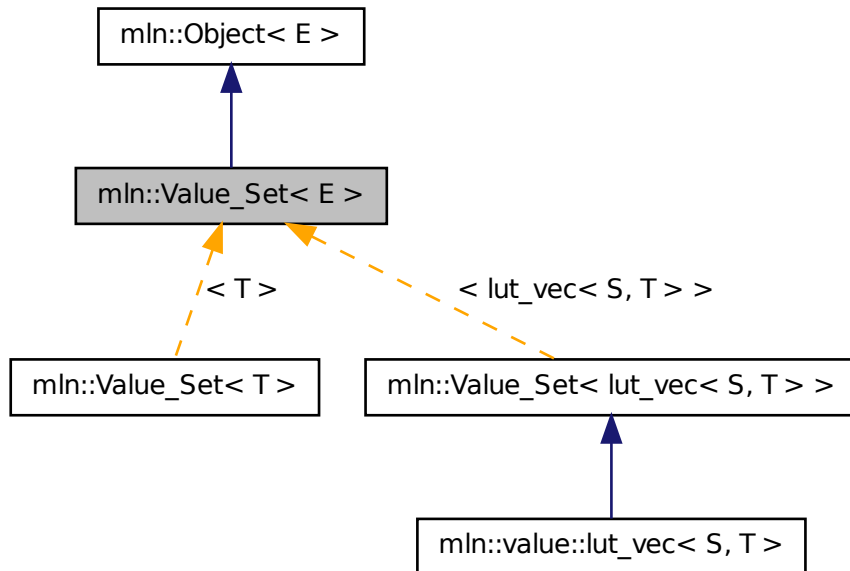
Definition at line 92 of file value\_iterator.hh.

## 10.410 mln::Value\_Set< E > Struct Template Reference

Base class for implementation classes of sets of values.

```
#include <value_set.hh>
```

Inheritance diagram for mln::Value\_Set< E >:



### 10.410.1 Detailed Description

```
template<typename E> struct mln::Value_Set< E >
```

Base class for implementation classes of sets of values.

See also

[mln::doc::Value\\_Set](#) for a complete documentation of this class contents.

Definition at line 57 of file `value_set.hh`.

## 10.411 mln::Vertex< E > Struct Template Reference

[Vertex](#) category flag type.

```
#include <vertex.hh>
```

### 10.411.1 Detailed Description

```
template<typename E> struct mln::Vertex< E >
```

[Vertex](#) category flag type.

Definition at line 53 of file vertex.hh.

## 10.412 mln::vertex\_image< P, V, G > Class Template Reference

[Image](#) based on graph vertices.

```
#include <vertex_image.hh>
```

Inherits [image\\_base< fun::i2v::array< V >, p\\_vertices< G, internal::vfsite\\_selector< P, G >::site\\_function\\_t >, vertex\\_image< P, V, G >>](#).

### Public Types

- typedef [G](#) [graph\\_t](#)  
*The type of the underlying graph.*
- typedef [vertex\\_nbh\\_t](#) [nbh\\_t](#)  
*Neighborhood type.*
- typedef [internal::vfsite\\_selector< P, G >::site\\_function\\_t](#) [site\\_function\\_t](#)  
*Function mapping graph elements to sites.*
- typedef [vertex\\_image< tag::psite\\_< P >, tag::value\\_< V >, tag::graph\\_< G >>](#) [skeleton](#)  
*Skeleton type.*
- typedef [graph\\_elt\\_neighborhood< G, S >](#) [vertex\\_nbh\\_t](#)  
*Vertex Neighborhood type.*
- typedef [graph\\_elt\\_window< G, S >](#) [vertex\\_win\\_t](#)  
*Vertex Window type.*
- typedef [vertex\\_win\\_t](#) [win\\_t](#)  
*Window type.*

### Public Member Functions

- [vertex\\_image](#) ()  
*Constructors.*
- rvalue [operator](#)() (unsigned v\_id) const  
*Value accessors/operators overloads.*

### 10.412.1 Detailed Description

`template<typename P, typename V, typename G = util::graph> class mln::vertex_image< P, V, G >`

[Image](#) based on graph vertices.

Definition at line 126 of file core/image/vertex\_image.hh.

### 10.412.2 Member Typedef Documentation

**10.412.2.1** `template<typename P, typename V, typename G = util::graph> typedef G mln::vertex_image< P, V, G >::graph_t`

The type of the underlying graph.

Definition at line 141 of file core/image/vertex\_image.hh.

**10.412.2.2** `template<typename P, typename V, typename G = util::graph> typedef vertex_nbh_t mln::vertex_image< P, V, G >::nbh_t`

[Neighborhood](#) type.

Definition at line 165 of file core/image/vertex\_image.hh.

**10.412.2.3** `template<typename P, typename V, typename G = util::graph> typedef internal::vfsite_selector<P,G>::site_function_t mln::vertex_image< P, V, G >::site_function_t`

[Function](#) mapping graph elements to sites.

Definition at line 145 of file core/image/vertex\_image.hh.

**10.412.2.4** `template<typename P, typename V, typename G = util::graph> typedef vertex_image< tag::psite_<P>, tag::value_<V>, tag::graph_<G> > mln::vertex_image< P, V, G >::skeleton`

[Skeleton](#) type.

Definition at line 152 of file core/image/vertex\_image.hh.

**10.412.2.5** `template<typename P, typename V, typename G = util::graph> typedef graph_elt_neighborhood<G,S> mln::vertex_image< P, V, G >::vertex_nbh_t`

[Vertex Neighborhood](#) type.

Definition at line 160 of file core/image/vertex\_image.hh.

**10.412.2.6** `template<typename P, typename V, typename G = util::graph> typedef graph_elt_window<G,S> mln::vertex_image< P, V, G >::vertex_win_t`

[Vertex Window](#) type.

Definition at line 157 of file core/image/vertex\_image.hh.

**10.412.2.7** `template<typename P, typename V, typename G = util::graph> typedef vertex_win_t mln::vertex_image< P, V, G >::win_t`

[Window](#) type.

Definition at line 163 of file core/image/vertex\_image.hh.

### 10.412.3 Constructor & Destructor Documentation

**10.412.3.1** `template<typename P, typename V, typename G > mln::vertex_image< P, V, G >::vertex_image( ) [inline]`

Constructors.

Definition at line 247 of file core/image/vertex\_image.hh.

### 10.412.4 Member Function Documentation

**10.412.4.1** `template<typename P, typename V, typename G > vertex_image< P, V, G >::rvalue mln::vertex_image< P, V, G >::operator() ( unsigned v_id ) const`

[Value](#) accessors/operators overloads.

Definition at line 292 of file core/image/vertex\_image.hh.

## 10.413 mln::violent\_cast\_image< T, I > Struct Template Reference

Violently cast image values to a given type.

```
#include <violent_cast_image.hh>
```

Inherits `image_value_morpher< I, T, violent_cast_image< T, I > >`.

### Public Types

- typedef T [lvalue](#)  
*Return type of read-write access.*
- typedef T [rvalue](#)  
*Return type of read-only access.*
- typedef [violent\\_cast\\_image](#)< tag::value\_< T >, tag::image\_< I > > [skeleton](#)  
*Skeleton.*
- typedef T [value](#)  
*Value associated type.*

## Public Member Functions

- [T operator\(\)](#) (const typename I::psite &p) const  
*Read-only access of pixel value at point site p.*
- [T operator\(\)](#) (const typename I::psite &p)  
*Mutable access is only OK for reading (not writing).*
- [violent\\_cast\\_image](#) (const [Image](#)< I > &ima)  
*Constructor.*

### 10.413.1 Detailed Description

**template<typename T, typename I> struct mln::violent\_cast\_image< T, I >**

Violently cast image values to a given type.

Definition at line 112 of file violent\_cast\_image.hh.

### 10.413.2 Member Typedef Documentation

**10.413.2.1 template<typename T, typename I> typedef T mln::violent\_cast\_image< T, I >::lvalue**

Return type of read-write access.

Definition at line 122 of file violent\_cast\_image.hh.

**10.413.2.2 template<typename T, typename I> typedef T mln::violent\_cast\_image< T, I >::rvalue**

Return type of read-only access.

Definition at line 119 of file violent\_cast\_image.hh.

**10.413.2.3 template<typename T, typename I> typedef violent\_cast\_image< tag::value\_<T>, tag::image\_<I> > mln::violent\_cast\_image< T, I >::skeleton**

Skeleton.

Definition at line 125 of file violent\_cast\_image.hh.

**10.413.2.4 template<typename T, typename I> typedef T mln::violent\_cast\_image< T, I >::value**

[Value](#) associated type.

Definition at line 116 of file violent\_cast\_image.hh.

### 10.413.3 Constructor & Destructor Documentation

**10.413.3.1** `template<typename T , typename I > mln::violent_cast_image< T, I >::violent_cast_image ( const Image< I > & ima ) [inline]`

Constructor.

Definition at line 172 of file violent\_cast\_image.hh.

### 10.413.4 Member Function Documentation

**10.413.4.1** `template<typename T , typename I > T mln::violent_cast_image< T, I >::operator() ( const typename I::psite & p ) const [inline]`

Read-only access of pixel value at point site p.

Definition at line 191 of file violent\_cast\_image.hh.

**10.413.4.2** `template<typename T , typename I > T mln::violent_cast_image< T, I >::operator() ( const typename I::psite & p ) [inline]`

Mutable access is only OK for reading (not writing).

Definition at line 200 of file violent\_cast\_image.hh.

## 10.414 mln::w\_window< D, W > Struct Template Reference

Generic [w\\_window](#) class.

```
#include <w_window.hh>
```

Inherits [weighted\\_window\\_base< mln::window< D >, w\\_window< D, W > >](#).

### Public Types

- typedef `with_w_< dpsites_bkd_piter< w_window< D, W > >, W >` [bkd\\_qiter](#)  
*Site\_Iterator type to browse (backward) the points of a generic w\_window.*
- typedef `D` [dpsite](#)  
*Dpsite associated type.*
- typedef `with_w_< dpsites_fwd_piter< w_window< D, W > >, W >` [fwd\\_qiter](#)  
*Site\_Iterator type to browse (forward) the points of a generic w\_window.*
- typedef `W` [weight](#)  
*Weight associated type.*

### Public Member Functions

- void [clear](#) ()



*Clear this window.*

- `w_window< D, W > & insert (const W &w, const D &d)`  
*Insert a couple of weight  $w$  and delta-point  $d$ .*
- `bool is_symmetric () const`  
*Test if the window is symmetric.*
- `const std::vector< D > & std_vector () const`  
*Give access to the vector of delta-points.*
- `void sym ()`  
*Apply a central symmetry to the window.*
- `W w (unsigned i) const`  
*Give the  $i$ -th weight.*
- `w_window ()`  
*Constructor without argument.*
- `const std::vector< W > & weights () const`  
*Give access to the vector of weights.*
- `const mln::window< D > & win () const`  
*Give the corresponding window.*

## Related Functions

(Note that these are not member functions.)

- `template<typename D, typename W > std::ostream & operator<< (std::ostream &ostr, const w_window< D, W > &w_win)`  
*Print a weighted window  $w\_win$  into an output stream  $ostr$ .*
- `template<typename D, typename Wl, typename Wr > bool operator== (const w_window< D, Wl > &lhs, const w_window< D, Wr > &rhs)`  
*Equality test between two weighted windows  $lhs$  and  $rhs$ .*

### 10.414.1 Detailed Description

`template<typename D, typename W> struct mln::w_window< D, W >`

Generic `w_window` class. This type of `w_window` is just like a set of delta-points. The parameter `D` is the type of delta-points; the parameter `W` is the type of weights.

Definition at line 100 of file `core/w_window.hh`.

### 10.414.2 Member Typedef Documentation

**10.414.2.1** `template<typename D, typename W> typedef with_w_< dpsites_bkd_piter< w_window<D, W> >, W > mln::w_window< D, W >::bkd_qiter`

[Site\\_Iterator](#) type to browse (backward) the points of a generic [w\\_window](#).

Definition at line 114 of file core/w\_window.hh.

**10.414.2.2** `template<typename D, typename W> typedef D mln::w_window< D, W >::dpsite`

Dpsite associated type.

Definition at line 104 of file core/w\_window.hh.

**10.414.2.3** `template<typename D, typename W> typedef with_w_< dpsites_fwd_piter< w_window<D, W> >, W > mln::w_window< D, W >::fwd_qiter`

[Site\\_Iterator](#) type to browse (forward) the points of a generic [w\\_window](#).

Definition at line 111 of file core/w\_window.hh.

**10.414.2.4** `template<typename D, typename W> typedef W mln::w_window< D, W >::weight`

Weight associated type.

Definition at line 107 of file core/w\_window.hh.

### 10.414.3 Constructor & Destructor Documentation

**10.414.3.1** `template<typename D , typename W > mln::w_window< D, W >::w_window ( )  
[inline]`

Constructor without argument.

Definition at line 213 of file core/w\_window.hh.

### 10.414.4 Member Function Documentation

**10.414.4.1** `template<typename D , typename W > void mln::w_window< D, W >::clear ( )  
[inline]`

Clear this window.

Definition at line 308 of file core/w\_window.hh.

**10.414.4.2** `template<typename D , typename W > w_window< D, W > & mln::w_window< D,  
W >::insert ( const W & w, const D & d ) [inline]`

Insert a couple of weight *w* and delta-point *d*.

Definition at line 254 of file core/w\_window.hh.

Referenced by mln::w\_window< D, W >::sym(), mln::make::w\_window(), mln::make::w\_window1d(), mln::make::w\_window3d(), and mln::make::w\_window\_directional().

**10.414.4.3** `template<typename D , typename W > bool mln::w_window< D, W >::is_symmetric ( ) const [inline]`

Test if the window is symmetric.

Definition at line 284 of file core/w\_window.hh.

References mln::w\_window< D, W >::sym().

**10.414.4.4** `template<typename D , typename W > const std::vector< D > & mln::w_window< D, W >::std_vector ( ) const [inline]`

Give access to the vector of delta-points.

Definition at line 228 of file core/w\_window.hh.

**10.414.4.5** `template<typename D , typename W > void mln::w_window< D, W >::sym ( ) [inline]`

Apply a central symmetry to the window.

Definition at line 296 of file core/w\_window.hh.

References mln::w\_window< D, W >::insert().

Referenced by mln::w\_window< D, W >::is\_symmetric().

**10.414.4.6** `template<typename D , typename W > W mln::w_window< D, W >::w ( unsigned i ) const [inline]`

Give the *i*-th weight.

Definition at line 244 of file core/w\_window.hh.

**10.414.4.7** `template<typename D , typename W > const std::vector< W > & mln::w_window< D, W >::weights ( ) const [inline]`

Give access to the vector of weights.

Definition at line 236 of file core/w\_window.hh.

Referenced by mln::w\_window< D, W >::operator==( ).

**10.414.4.8** `template<typename D , typename W > const mln::window< D > & mln::w_window< D, W >::win ( ) const [inline]`

Give the corresponding window.

Definition at line 220 of file core/w\_window.hh.

Referenced by mln::w\_window< D, W >::operator==( ).

### 10.414.5 Friends And Related Function Documentation

**10.414.5.1** `template<typename D , typename W > std::ostream & operator<< ( std::ostream & ostr, const w_window< D, W > & w_win ) [related]`

Print a weighted window `w_win` into an output stream `ostr`.

Definition at line 420 of file `core/w_window.hh`.

**10.414.5.2** `template<typename D , typename Wl , typename Wr > bool operator== ( const w_window< D, Wl > & lhs, const w_window< D, Wr > & rhs ) [related]`

Equality test between two weighted windows `lhs` and `rhs`.

Definition at line 430 of file `core/w_window.hh`.

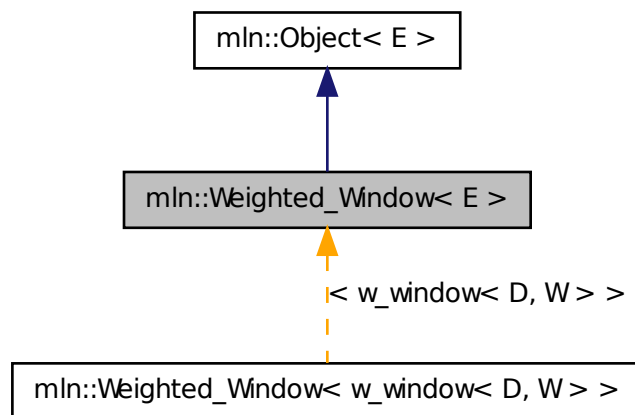
References `mln::w_window< D, W >::weights()`, and `mln::w_window< D, W >::win()`.

## 10.415 mln::Weighted\_Window< E > Struct Template Reference

Base class for implementation classes that are `weighted_windows`.

```
#include <weighted_window.hh>
```

Inheritance diagram for `mln::Weighted_Window< E >`:



### Related Functions

(Note that these are not member functions.)

- `template<typename W > W operator- (const Weighted_Window< W > &rhs)`

*Compute the symmetrical weighted window of rhs.*

### 10.415.1 Detailed Description

`template<typename E> struct mln::Weighted_Window< E >`

Base class for implementation classes that are weighted\_windows.

**See also**

[mln::doc::Weighted\\_Window](#) for a complete documentation of this class contents.

Definition at line 68 of file weighted\_window.hh.

### 10.415.2 Friends And Related Function Documentation

**10.415.2.1** `template<typename W > W operator- ( const Weighted_Window< W > & rhs )`  
[related]

Compute the symmetrical weighted window of rhs.

## 10.416 mln::win::backdiag2d Struct Reference

Diagonal line window defined on the 2D square grid.

```
#include <backdiag2d.hh>
```

Inherits `classical_window_base< dpoint2d, backdiag2d >`.

### Public Member Functions

- `backdiag2d` (unsigned length)  
*Constructor.*
- unsigned `length` () const  
*Give the diagonal length, that is, its width.*

### 10.416.1 Detailed Description

Diagonal line window defined on the 2D square grid. An `backdiag2d` is centered and symmetric. its width (length) is odd.

For instance:

```
*  ○
*   ○
*    ×
*   ○
*    ○
*   ○
*
```

is defined with `length = 5`.

Definition at line 63 of file `backdiag2d.hh`.

## 10.416.2 Constructor & Destructor Documentation

### 10.416.2.1 `mln::win::backdiag2d::backdiag2d ( unsigned length ) [inline]`

Constructor.

#### Parameters

[in] *length* Length, thus width, of the diagonal line.

#### Precondition

`length` is odd.

Definition at line 93 of file `backdiag2d.hh`.

## 10.416.3 Member Function Documentation

### 10.416.3.1 `unsigned mln::win::backdiag2d::length ( ) const [inline]`

Give the diagonal length, that is, its width.

Definition at line 105 of file `backdiag2d.hh`.

## 10.417 `mln::win::ball< G, C >` Struct Template Reference

Generic ball window defined on a given grid.

```
#include <ball.hh>
```

Inherits `classical_window_base< dpoint< G, C >, ball< G, C > >`.

### Public Member Functions

- `ball` (unsigned diameter)  
*Constructor.*
- unsigned `diameter` () const  
*Give the ball diameter.*

### 10.417.1 Detailed Description

```
template<typename G, typename C> struct mln::win::ball< G, C >
```

Generic ball window defined on a given grid. A ball is centered and symmetric; so its diameter is odd.

`G` is the given grid on which the ball is defined and `C` is the type of coordinates.

Definition at line 71 of file `ball.hh`.

## 10.417.2 Constructor & Destructor Documentation

**10.417.2.1** `template<typename G , typename C > mln::win::ball< G, C >::ball ( unsigned diameter ) [inline]`

Constructor.

### Parameters

[in] *diameter* Diameter of the ball.

### Precondition

`diameter` is odd.

Definition at line 99 of file ball.hh.

References `mln::literal::origin`.

## 10.417.3 Member Function Documentation

**10.417.3.1** `template<typename G , typename C > unsigned mln::win::ball< G, C >::diameter ( ) const [inline]`

Give the ball diameter.

Definition at line 122 of file ball.hh.

## 10.418 mln::win::cube3d Struct Reference

Cube window defined on the 3D grid.

```
#include <cube3d.hh>
```

Inherits `classical_window_base< dpoint3d, cube3d >`.

### Public Member Functions

- `cube3d` (unsigned length)  
*Constructor.*
- unsigned `length` () const  
*Give the cube length, that is, its height.*

### 10.418.1 Detailed Description

Cube window defined on the 3D grid. An `cube3d` is centered and symmetric; so its height (length) is odd.

For instance:

```
*   o o o
*   o o o
```

```

*  o  o  o
*
*  o  o  o
*  o  x  o
*  o  o  o
*
*  o  o  o
*  o  o  o
*  o  o  o
*
*

```

is defined with `length = 3`.

Definition at line 69 of file `cube3d.hh`.

## 10.418.2 Constructor & Destructor Documentation

### 10.418.2.1 `mln::win::cube3d::cube3d ( unsigned length ) [inline]`

Constructor.

#### Parameters

[in] *length* Length, thus height, of the `cube3d`.

#### Precondition

`length` is odd.

Definition at line 99 of file `cube3d.hh`.

## 10.418.3 Member Function Documentation

### 10.418.3.1 `unsigned mln::win::cube3d::length ( ) const [inline]`

Give the cube length, that is, its height.

Definition at line 113 of file `cube3d.hh`.

## 10.419 mln::win::cuboid3d Struct Reference

Cuboid defined on the 3-D square grid.

```
#include <cuboid3d.hh>
```

Inherits `classical_window_base< dpoint3d, cuboid3d >`.

### Public Member Functions

- `cuboid3d` (unsigned depth, unsigned height, unsigned width)  
*Constructor.*
- unsigned `volume` () const  
*Return the volume of the cuboid.*



- unsigned `depth` () const  
*Accessors.*
- unsigned `height` () const  
*Return the height of the cuboid.*
- unsigned `width` () const  
*Return the width of the cuboid.*

### 10.419.1 Detailed Description

Cuboid defined on the 3-D square grid. A `cuboid3d` is a 3-D window with cuboid (also known as rectangular prism or rectangular parallelepiped) shape. It is centered and symmetric.

For instance:

```

      o o o o o o o
      o o o o o o o
    o o o o o o o
    o o o o o o o
  o o o o o o o

      o o o o o o o
      o o o o o o o
    o o o x o o o
    o o o o o o o
  o o o o o o o

      o o o o o o o
      o o o o o o o
    o o o o o o o
    o o o o o o o
  o o o o o o o

```

is defined with `depth = 3`, `height = 5` and `width = 7`.

Reference: <http://en.wikipedia.org/wiki/Cuboid>

Definition at line 80 of file `cuboid3d.hh`.

### 10.419.2 Constructor & Destructor Documentation

#### 10.419.2.1 `mln::win::cuboid3d::cuboid3d ( unsigned depth, unsigned height, unsigned width )` [inline]

Constructor.

#### Parameters

- [in] *depth* The depth of the `cuboid3d`.
- [in] *height* The height of the `cuboid3d`.
- [in] *width* The width of the `cuboid3d`.

#### Precondition

Argument *depth*, *height* and *width* must be odd.

Definition at line 125 of file cuboid3d.hh.

### 10.419.3 Member Function Documentation

#### 10.419.3.1 unsigned mln::win::cuboid3d::depth ( ) const [inline]

Accessors.

Return the depth of the cuboid.

Definition at line 146 of file cuboid3d.hh.

#### 10.419.3.2 unsigned mln::win::cuboid3d::height ( ) const [inline]

Return the height of the cuboid.

Definition at line 153 of file cuboid3d.hh.

#### 10.419.3.3 unsigned mln::win::cuboid3d::volume ( ) const [inline]

Return the volume of the cuboid.

Definition at line 167 of file cuboid3d.hh.

#### 10.419.3.4 unsigned mln::win::cuboid3d::width ( ) const [inline]

Return the width of the cuboid.

Definition at line 160 of file cuboid3d.hh.

## 10.420 mln::win::diag2d Struct Reference

Diagonal line window defined on the 2D square grid.

```
#include <diag2d.hh>
```

Inherits classical\_window\_base< dpoint2d, diag2d >.

### Public Member Functions

- [diag2d](#) (unsigned length)  
*Constructor.*
- unsigned [length](#) () const  
*Give the diagonal length, that is, its width.*

#### 10.420.1 Detailed Description

Diagonal line window defined on the 2D square grid. An [diag2d](#) is centered and symmetric. its width (length) is odd.

For instance:

```

*           o
*         o
*       x
*     o
*   o
* o
*

```

is defined with length = 5.

Definition at line 63 of file diag2d.hh.

## 10.420.2 Constructor & Destructor Documentation

### 10.420.2.1 mln::win::diag2d::diag2d ( unsigned *length* ) [inline]

Constructor.

#### Parameters

[in] *length* Length, thus width, of the diagonal line.

#### Precondition

*length* is odd.

Definition at line 93 of file diag2d.hh.

## 10.420.3 Member Function Documentation

### 10.420.3.1 unsigned mln::win::diag2d::length ( ) const [inline]

Give the diagonal length, that is, its width.

Definition at line 106 of file diag2d.hh.

## 10.421 mln::win::line< M, i, C > Struct Template Reference

Generic line window defined on a given grid in the given dimension.

```
#include <line.hh>
```

Inherits classical\_window\_base< dpoint< M, C >, line< M, i, C > >.

### Public Types

- enum

*Direction*.

## Public Member Functions

- unsigned `length` () const  
*Give the line length.*
- `line` (unsigned length)  
*Constructor.*
- unsigned `size` () const  
*Give the line size, that is, its length.*

### 10.421.1 Detailed Description

`template<typename M, unsigned i, typename C> struct mln::win::line< M, i, C >`

Generic line window defined on a given grid in the given dimension. An line is centered and symmetric; so its length is odd.

M is the given grid on which the line is defined, i is the given dimension of the line end C is the type of the coordinates.

#### See also

`mln::win::hline2d` for an exemple of his use.

Definition at line 73 of file `win/line.hh`.

### 10.421.2 Member Enumeration Documentation

**10.421.2.1** `template<typename M , unsigned i, typename C > anonymous enum`

Direction.

Definition at line 76 of file `win/line.hh`.

### 10.421.3 Constructor & Destructor Documentation

**10.421.3.1** `template<typename M , unsigned i, typename C > mln::win::line< M, i, C >::line ( unsigned length ) [inline]`

Constructor.

#### Parameters

[in] *length* Length of the line.

#### Precondition

`length` is odd.

Definition at line 106 of file `win/line.hh`.

References `mln::dpoint< G, C >::set_all()`.

### 10.421.4 Member Function Documentation

**10.421.4.1** `template<typename M, unsigned i, typename C > unsigned mln::win::line< M, i, C >::length( ) const [inline]`

Give the line length.

Definition at line 125 of file win/line.hh.

**10.421.4.2** `template<typename M, unsigned i, typename C > unsigned mln::win::line< M, i, C >::size( ) const [inline]`

Give the line size, that is, its length.

Definition at line 132 of file win/line.hh.

## 10.422 mln::win::multiple< W, F > Class Template Reference

Multiple window.

```
#include <multiple.hh>
```

Inherits window\_base< W::dpsite, multiple< W, F > >.

### 10.422.1 Detailed Description

`template<typename W, typename F> class mln::win::multiple< W, F >`

Multiple window.

Definition at line 76 of file multiple.hh.

## 10.423 mln::win::multiple\_size< n, W, F > Class Template Reference

Definition of a multiple-size window.

```
#include <multiple_size.hh>
```

Inherits window\_base< W::dpsite, multiple\_size< n, W, F > >.

### 10.423.1 Detailed Description

`template<unsigned n, typename W, typename F> class mln::win::multiple_size< n, W, F >`

Definition of a multiple-size window.

Definition at line 78 of file multiple\_size.hh.

## 10.424 mln::win::octagon2d Struct Reference

Octagon window defined on the 2D square grid.

```
#include <octagon2d.hh>
```

Inherits classical\_window\_base< dpoint2d, octagon2d >.

### Public Member Functions

- unsigned [area](#) () const  
*Give the area.*
- unsigned [length](#) () const  
*Give the octagon length, that is, its width.*
- [octagon2d](#) (unsigned length)  
*Constructor.*

### 10.424.1 Detailed Description

Octagon window defined on the 2D square grid. An [octagon2d](#) is centered and symmetric.

The length  $L$  of the octagon is such as  $L = 6 * l + 1$  where  $l \geq 0$ .

For instance:

```
*      o o o
*     o o o o o
*    o o o o o o o
*   o o o x o o o
*  o o o o o o o
*   o o o o o
*    o o o
*
*
```

is defined with  $L = 7$  ( $l = 1$ ).

Definition at line 67 of file octagon2d.hh.

### 10.424.2 Constructor & Destructor Documentation

#### 10.424.2.1 mln::win::octagon2d::octagon2d ( unsigned *length* ) [inline]

Constructor.

#### Parameters

[in] *length* Length, of the octagon.

#### Precondition

*length* is such as  $length = 6 * x + 1$  where  $x \geq 0$ .

Definition at line 101 of file octagon2d.hh.

### 10.424.3 Member Function Documentation

#### 10.424.3.1 unsigned mln::win::octagon2d::area ( ) const [inline]

Give the area.

Definition at line 157 of file octagon2d.hh.

#### 10.424.3.2 unsigned mln::win::octagon2d::length ( ) const [inline]

Give the octagon length, that is, its width.

Definition at line 145 of file octagon2d.hh.

## 10.425 mln::win::rectangle2d Struct Reference

Rectangular window defined on the 2D square grid.

```
#include <rectangle2d.hh>
```

Inherits classical\_window\_base< dpoint2d, rectangle2d >.

### Public Member Functions

- unsigned [area](#) ( ) const  
*Give the rectangle area.*
- unsigned [height](#) ( ) const  
*Give the rectangle height.*
- [rectangle2d](#) (unsigned height, unsigned width)  
*Constructor.*
- const std::vector< [dpoint2d](#) > & [std\\_vector](#) ( ) const  
*Give the std vector of delta-points.*
- unsigned [width](#) ( ) const  
*Give the rectangle width.*

### 10.425.1 Detailed Description

Rectangular window defined on the 2D square grid. A [rectangle2d](#) is a 2D window with rectangular shape. It is centered and symmetric.

For instance:

```
*  o o o o o
*  o o x o o
*  o o o o o
*
```

is defined with height = 3 and width = 5.

Definition at line 64 of file rectangle2d.hh.

## 10.425.2 Constructor & Destructor Documentation

### 10.425.2.1 `mln::win::rectangle2d::rectangle2d ( unsigned height, unsigned width ) [inline]`

Constructor.

#### Parameters

[in] *height* Height of the [rectangle2d](#).

[in] *width* Width of the [rectangle2d](#).

#### Precondition

Height and width are odd.

Definition at line 106 of file rectangle2d.hh.

## 10.425.3 Member Function Documentation

### 10.425.3.1 `unsigned mln::win::rectangle2d::area ( ) const [inline]`

Give the rectangle area.

Definition at line 132 of file rectangle2d.hh.

### 10.425.3.2 `unsigned mln::win::rectangle2d::height ( ) const [inline]`

Give the rectangle height.

Definition at line 120 of file rectangle2d.hh.

### 10.425.3.3 `const std::vector< dpoint2d > & mln::win::rectangle2d::std_vector ( ) const [inline]`

Give the std vector of delta-points.

Definition at line 145 of file rectangle2d.hh.

### 10.425.3.4 `unsigned mln::win::rectangle2d::width ( ) const [inline]`

Give the rectangle width.

Definition at line 126 of file rectangle2d.hh.

## 10.426 `mln::Window< E >` Struct Template Reference

Base class for implementation classes that are windows.



```
#include <window.hh>
```

Inheritance diagram for mln::Window< E >:



### 10.426.1 Detailed Description

```
template<typename E> struct mln::Window< E >
```

Base class for implementation classes that are windows.

See also

[mln::doc::Window](#) for a complete documentation of this class contents.

Definition at line 87 of file concept/window.hh.

## 10.427 mln::window< D > Class Template Reference

Generic window class.

```
#include <window.hh>
```

Inherits `window_base< D, window< D > >`.

### Public Types

- typedef `dpsites_bkd_piter< window< D > >` `bkd_qiter`  
*Site Iterator* type to browse the points of a basic window w.r.t. the reverse ordering of delta-points.
- typedef `dpsites_fwd_piter< window< D > >` `fwd_qiter`  
*Site Iterator* type to browse the points of a basic window w.r.t. the ordering of delta-points.
- typedef `fwd_qiter` `qiter`  
*Site Iterator* type to browse the points of a basic window whatever the ordering of delta-points.
- typedef `window< D >` `regular`  
*Regular window associated type.*

## Public Member Functions

- void `clear ()`  
*Clear the window.*
- unsigned `delta () const`  
*Give the maximum coordinate gap between the window center and a window point.*
- const D & `dp (unsigned i) const`  
*Give the *i*-th delta-point.*
- bool `has (const D &dp) const`  
*Test if *dp* is in this window definition.*
- `window < D > & insert (const D &dp)`  
*Insert a delta-point *dp*.*
- template<typename W >  
`window < D > & insert (const Window < W > &win)`  
*Insert another window *win*.*
- bool `is_centered () const`  
*Test if the window is centered.*
- bool `is_empty () const`  
*Test if the window is empty (null size; no delta-point).*
- bool `is_symmetric () const`
- void `print (std::ostream &ostr) const`  
*Print the window definition into *ostr*.*
- unsigned `size () const`  
*Give the window size, i.e., the number of delta-sites.*
- const std::vector< D > & `std_vector () const`  
*Give the std vector of delta-points.*
- void `sym ()`  
*Apply a central symmetry to the target window.*
- `window ()`  
*Constructor without argument.*
- `window < D > & insert (const typename D::coord &dind)`

## Related Functions

(Note that these are not member functions.)

- `template<typename D >`  
`bool operator==(const window< D > &lhs, const window< D > &rhs)`  
*Equality comparison between windows lhs and rhs.*

### 10.427.1 Detailed Description

`template<typename D> class mln::window< D >`

Generic window class. This type of window is just like a set of delta-points. The parameter is D, type of delta-point.

Definition at line 85 of file window.hh.

### 10.427.2 Member Typedef Documentation

**10.427.2.1** `template<typename D> typedef dpsites_bkd_piter< window<D> > mln::window< D >::bkd_qiter`

[Site\\_Iterator](#) type to browse the points of a basic window w.r.t. the reverse ordering of delta-points.

Definition at line 124 of file window.hh.

**10.427.2.2** `template<typename D> typedef dpsites_fwd_piter< window<D> > mln::window< D >::fwd_qiter`

[Site\\_Iterator](#) type to browse the points of a basic window w.r.t. the ordering of delta-points.

Definition at line 119 of file window.hh.

**10.427.2.3** `template<typename D> typedef fwd_qiter mln::window< D >::qiter`

[Site\\_Iterator](#) type to browse the points of a basic window whatever the ordering of delta-points.

Definition at line 129 of file window.hh.

**10.427.2.4** `template<typename D> typedef window<D> mln::window< D >::regular`

Regular window associated type.

Definition at line 90 of file window.hh.

### 10.427.3 Constructor & Destructor Documentation

**10.427.3.1** `template<typename D > mln::window< D >::window ( ) [inline]`

Constructor without argument.

The constructed window is empty.

Definition at line 207 of file window.hh.

## 10.427.4 Member Function Documentation

### 10.427.4.1 `template<typename D> void mln::window<D>::clear ( ) [inline]`

Clear the window.

Definition at line 254 of file window.hh.

### 10.427.4.2 `template<typename D> unsigned mln::window<D>::delta ( ) const [inline]`

Give the maximum coordinate gap between the window center and a window point.

Definition at line 262 of file window.hh.

References `mln::window<D>::dp()`, and `mln::window<D>::size()`.

### 10.427.4.3 `template<typename D> const D & mln::window<D>::dp ( unsigned i ) const [inline]`

Give the *i*-th delta-point.

Definition at line 302 of file window.hh.

References `mln::window<D>::size()`.

Referenced by `mln::window<D>::delta()`, and `mln::window<D>::insert()`.

### 10.427.4.4 `template<typename D> bool mln::window<D>::has ( const D & dp ) const [inline]`

Test if `dp` is in this window definition.

Definition at line 311 of file window.hh.

### 10.427.4.5 `template<typename D> window<D> & mln::window<D>::insert ( const D & dp ) [inline]`

Insert a delta-point `dp`.

Definition at line 327 of file window.hh.

Referenced by `mln::c18()`, `mln::c26()`, `mln::c2_3d_sli()`, `mln::c4_3d()`, `mln::c6()`, `mln::window<D>::insert()`, `mln::morpho::line_gradient()`, `mln::window<D>::sym()`, `mln::convert::to_upper_window()`, `mln::convert::to_window()`, `mln::win_c4p()`, `mln::win_c4p_3d()`, `mln::win_c8p()`, and `mln::win_c8p_3d()`.

### 10.427.4.6 `template<typename D> template<typename W> window<D> & mln::window<D>::insert ( const Window<W> & win ) [inline]`

Insert another window `win`.

Definition at line 337 of file window.hh.

**10.427.4.7** `template<typename D > window< D > & mln::window< D >::insert ( const typename D::coord & dind ) [inline]`

Insertion of a delta-point with different numbers of arguments (coordinates) w.r.t. the dimension.

Definition at line 349 of file window.hh.

References mln::window< D >::dp(), and mln::window< D >::insert().

**10.427.4.8** `template<typename D > bool mln::window< D >::is_centered ( ) const [inline]`

Test if the window is centered.

#### Returns

True if the delta-point 0 belongs to the window.

Definition at line 226 of file window.hh.

References mln::literal::zero.

**10.427.4.9** `template<typename D > bool mln::window< D >::is_empty ( ) const [inline]`

Test if the window is empty (null size; no delta-point).

Definition at line 246 of file window.hh.

**10.427.4.10** `template<typename D > bool mln::window< D >::is_symmetric ( ) const [inline]`

Test if the window is symmetric.

#### Returns

True if for every dp of this window, -dp is also in this window.

Definition at line 216 of file window.hh.

References mln::window< D >::sym().

**10.427.4.11** `template<typename D > void mln::window< D >::print ( std::ostream & ostr ) const [inline]`

Print the window definition into `ostr`.

Definition at line 390 of file window.hh.

**10.427.4.12** `template<typename D > unsigned mln::window< D >::size ( ) const [inline]`

Give the window size, i.e., the number of delta-sites.

Definition at line 294 of file window.hh.

Referenced by mln::window< D >::delta(), mln::window< D >::dp(), mln::window< D >::sym(), mln::win\_c4p(), mln::win\_c4p\_3d(), mln::win\_c8p(), and mln::win\_c8p\_3d().

**10.427.4.13** `template<typename D> const std::vector< D> & mln::window< D>::std_vector ( ) const [inline]`

Give the std vector of delta-points.

Definition at line 319 of file window.hh.

**10.427.4.14** `template<typename D> void mln::window< D>::sym ( ) [inline]`

Apply a central symmetry to the target window.

Definition at line 234 of file window.hh.

References `mln::window< D>::insert()`, and `mln::window< D>::size()`.

Referenced by `mln::window< D>::is_symmetric()`.

## 10.427.5 Friends And Related Function Documentation

**10.427.5.1** `template<typename D> bool operator==( const window< D> & lhs, const window< D> & rhs ) [related]`

Equality comparison between windows `lhs` and `rhs`.

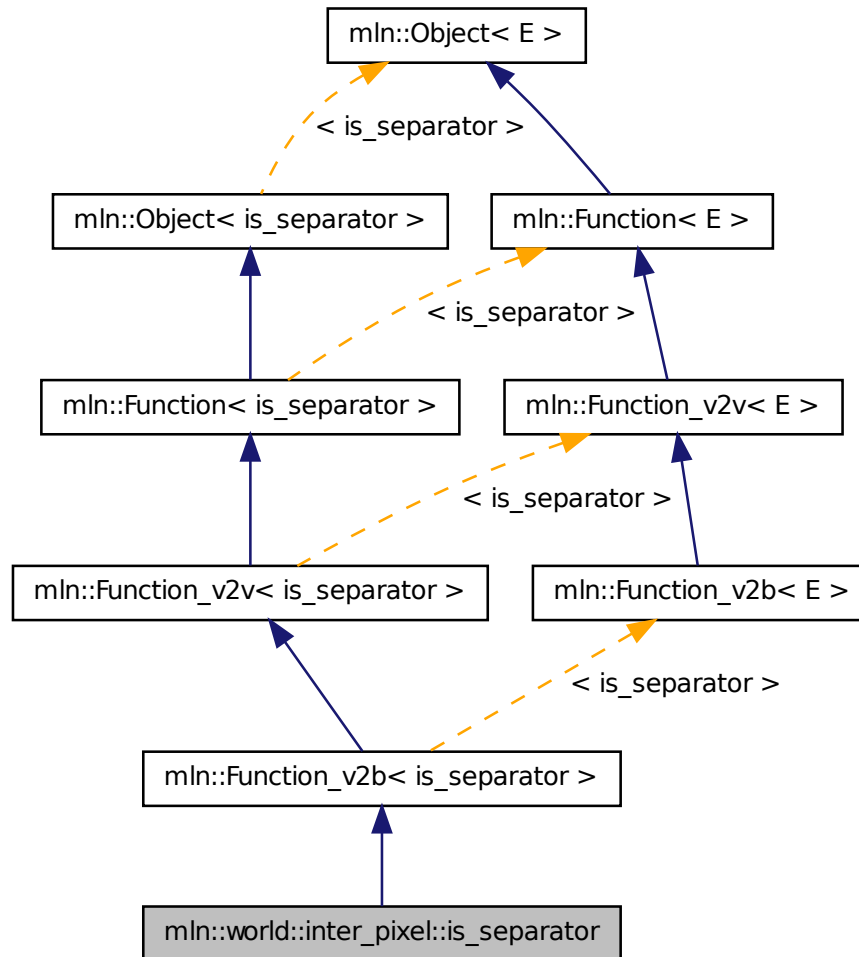
Definition at line 400 of file window.hh.

## 10.428 mln::world::inter\_pixel::is\_separator Struct Reference

Functor returning whether a site is a separator in an inter-pixel image.

```
#include <is_separator.hh>
```

Inheritance diagram for mln::world::inter\_pixel::is\_separator:



### 10.428.1 Detailed Description

Functor returning whether a site is a separator in an inter-pixel image.

Definition at line 52 of file is\_separator.hh.

## 10.429 trait::graph< I > Struct Template Reference

Graph traits.

```
#include <morpho.hh>
```

### 10.429.1 Detailed Description

```
template<typename I> struct trait::graph< I >
```

Graph traits.

Definition at line 73 of file morpho.hh.

## 10.430 `trait::graph< mln::complex_image< 1, G, V > >` Struct Template Reference

Graph traits for 1-complexes images.

```
#include <morpho.hh>
```

### 10.430.1 Detailed Description

```
template<typename G, typename V> struct trait::graph< mln::complex_image< 1, G, V > >
```

Graph traits for 1-complexes images.

Definition at line 133 of file morpho.hh.

## 10.431 `trait::graph< mln::image2d< T > >` Struct Template Reference

Graph traits for [mln::image2d](#).

```
#include <morpho.hh>
```

### 10.431.1 Detailed Description

```
template<typename T> struct trait::graph< mln::image2d< T > >
```

Graph traits for [mln::image2d](#).

Definition at line 94 of file morpho.hh.



# Index

- ~decorated\_image
  - mln::decorated\_image, 598
- ~proxy
  - mln::value::proxy, 1135
- ~soft\_heap
  - mln::util::soft\_heap, 1091
- ~tracked\_ptr
  - mln::util::tracked\_ptr, 1094
- \_1
  - mln::algebra::h\_mat, 546
- 1D neighborhoods, 101
- 1D windows, 112
- 2D neighborhoods, 102
- 2D windows, 113
- 3D neighborhoods, 104
- 3D windows, 115
- a\_point\_of
  - mln, 150
- abs
  - mln::data, 206
  - mln::math, 355
- abs\_inplace
  - mln::data, 206
- Accumulators, 99
- add
  - mln::topo::n\_faces\_set, 1027
- add\_child
  - mln::util::tree\_node, 1099
- add\_edge
  - mln::util::graph, 1062
- add\_face
  - mln::topo::complex, 1006
- add\_location
  - mln::geom::complex\_geometry, 726
- add\_tree\_down
  - mln::util::tree, 1096
- add\_tree\_up
  - mln::util::tree, 1096
- add\_vertex
  - mln::util::graph, 1062
- add\_vertices
  - mln::util::graph, 1062
- addr
  - mln::topo::complex, 1006
- adj\_higher\_dim\_connected\_n\_face\_bkd\_iter
  - mln::topo::adj\_higher\_dim\_connected\_n\_face\_bkd\_iter, 977
- adj\_higher\_dim\_connected\_n\_face\_fwd\_iter
  - mln::topo::adj\_higher\_dim\_connected\_n\_face\_fwd\_iter, 979
- adj\_higher\_face\_bkd\_iter
  - mln::topo::adj\_higher\_face\_bkd\_iter, 980
- adj\_higher\_face\_fwd\_iter
  - mln::topo::adj\_higher\_face\_fwd\_iter, 981
- adj\_lower\_dim\_connected\_n\_face\_bkd\_iter
  - mln::topo::adj\_lower\_dim\_connected\_n\_face\_bkd\_iter, 982
- adj\_lower\_dim\_connected\_n\_face\_fwd\_iter
  - mln::topo::adj\_lower\_dim\_connected\_n\_face\_fwd\_iter, 983
- adj\_lower\_face\_bkd\_iter
  - mln::topo::adj\_lower\_face\_bkd\_iter, 984
- adj\_lower\_face\_fwd\_iter
  - mln::topo::adj\_lower\_face\_fwd\_iter, 985
- adj\_lower\_higher\_face\_bkd\_iter
  - mln::topo::adj\_lower\_higher\_face\_bkd\_iter, 986
- adj\_lower\_higher\_face\_fwd\_iter
  - mln::topo::adj\_lower\_higher\_face\_fwd\_iter, 988
- adj\_m\_face\_bkd\_iter
  - mln::topo::adj\_m\_face\_bkd\_iter, 989
- adj\_m\_face\_fwd\_iter
  - mln::topo::adj\_m\_face\_fwd\_iter, 990
- adjacency\_matrix
  - mln::util::adjacency\_matrix, 1035, 1036
- adjust
  - mln::border, 190
  - mln::extension, 240
- adjust\_duplicate
  - mln::extension, 241
- adjust\_fill
  - mln::extension, 241
- algebraic\_face
  - mln::topo::algebraic\_face, 993
- algebraic\_n\_face
  - mln::topo::algebraic\_n\_face, 998
- and\_inplace
  - mln::logical, 327

- and\_not
  - mln::logical, 327
- and\_not\_inplace
  - mln::logical, 327
- antialiased
  - mln::subsampling, 395
- apex
  - mln::util::branch, 1044
- append
  - mln::p\_array, 850
  - mln::util::array, 1039, 1040
- apply
  - mln::data, 206
- apply\_p2p
  - mln, 150
- area
  - mln::accu::site\_set::rectangularity, 510
  - mln::morpho::attribute::sharpness, 837
  - mln::morpho::attribute::volume, 841
  - mln::win::octagon2d, 1169
  - mln::win::rectangle2d, 1170
- argument
  - mln::accu::shape::height, 506
  - mln::accu::shape::volume, 508
  - mln::doc::Accumulator, 600
- array
  - mln::util::array, 1039
- at
  - mln::opt, 388
- attachment
  - mln::make, 335
- backdiag2d
  - mln::win::backdiag2d, 1160
- background
  - mln::labeling, 298
- ball
  - mln::win::ball, 1161
- base\_level
  - mln::morpho::attribute::height, 836
- Basic types, 95, 108
- bbox
  - mln::accu::site\_set::rectangularity, 510
  - mln::Box, 563
  - mln::box, 557
  - mln::doc::Box, 603
  - mln::doc::Fastest\_Image, 610
  - mln::doc::Image, 620
  - mln::geom, 255
  - mln::image1d, 767
  - mln::image2d, 772
  - mln::image3d, 780
  - mln::labeled\_image, 792
  - mln::labeled\_image\_base, 795
  - mln::p\_line2d, 888
  - mln::p\_run, 915
- bbox\_t
  - mln::labeled\_image, 791
  - mln::labeled\_image\_base, 794
- bboxes
  - mln::labeled\_image, 792
  - mln::labeled\_image\_base, 795
- before
  - mln, 163
- begin
  - mln::p\_line2d, 888
- bin\_1complex\_image2d
  - mln, 145
- bin\_2complex\_image3df
  - mln, 145
- binarization
  - mln::binarization, 188
- bkd\_citer
  - mln::topo::complex, 1005
- bkd\_eiter
  - mln::util::array, 1038
  - mln::util::set, 1084
- bkd\_niter
  - mln::doc::Neighborhood, 624
  - mln::graph\_elt\_mixed\_neighborhood, 734
  - mln::graph\_elt\_neighborhood, 740
  - mln::graph\_elt\_neighborhood\_if, 741
  - mln::mixed\_neighb, 831
  - mln::neighb, 843
- bkd\_piter
  - mln::box, 555
  - mln::doc::Box, 602
  - mln::doc::Fastest\_Image, 608
  - mln::doc::Image, 618
  - mln::doc::Site\_Set, 634
  - mln::hexa, 760
  - mln::image2d\_h, 775
  - mln::p\_array, 849
  - mln::p\_centered, 854
  - mln::p\_complex, 857
  - mln::p\_edges, 861
  - mln::p\_faces, 866
  - mln::p\_if, 871
  - mln::p\_image, 875
  - mln::p\_key, 882
  - mln::p\_line2d, 887
  - mln::p\_mutable\_array\_of, 891
  - mln::p\_priority, 898
  - mln::p\_queue, 903
  - mln::p\_queue\_fast, 908
  - mln::p\_run, 913
  - mln::p\_set, 918
  - mln::p\_set\_of, 922

- mln::p\_transformed, [925](#)
- mln::p\_vaccess, [930](#)
- mln::p\_vertices, [934](#)
- bkd\_pixter1d
  - mln::bkd\_pixter1d, [549](#)
- bkd\_pixter2d
  - mln::bkd\_pixter2d, [550](#)
- bkd\_pixter3d
  - mln::bkd\_pixter3d, [552](#)
- bkd\_qiter
  - mln::doc::Weighted\_Window, [640](#)
  - mln::doc::Window, [642](#)
  - mln::graph\_elt\_mixed\_window, [736](#)
  - mln::graph\_elt\_window, [744](#)
  - mln::graph\_elt\_window\_if, [748](#)
  - mln::w\_window, [1156](#)
  - mln::window, [1173](#)
- bkd\_viter
  - mln::doc::Value\_Set, [638](#)
  - mln::value::lut\_vec, [1131](#)
- black
  - mln::literal, [323](#)
- blobs
  - mln::canvas::labeling, [196](#)
  - mln::labeling, [299](#)
- blobs\_and\_compute
  - mln::labeling, [299](#)
- blue
  - mln::literal, [323](#)
- border
  - mln::doc::Fastest\_Image, [610](#)
  - mln::image1d, [767](#)
  - mln::image2d, [772](#)
  - mln::image3d, [780](#)
- box
  - mln::box, [556](#)
  - mln::draw, [234](#)
- box1d
  - mln, [145](#)
  - mln::make, [335](#), [336](#)
- box2d
  - mln, [145](#)
  - mln::make, [336](#)
- box2d\_h
  - mln, [145](#)
  - mln::make, [337](#)
- box3d
  - mln, [146](#)
  - mln::make, [338](#)
- box\_plain
  - mln::draw, [234](#)
- box\_runend\_piter
  - mln::box\_runend\_piter, [566](#)
- box\_runstart\_piter
  - mln::box\_runstart\_piter, [568](#)
- branch
  - mln::util::branch, [1043](#)
- brown
  - mln::literal, [323](#)
- buffer
  - mln::doc::Fastest\_Image, [611](#)
  - mln::image1d, [767](#)
  - mln::image2d, [772](#)
  - mln::image3d, [780](#)
- c18
  - modneighb3d, [105](#)
- c2
  - modneighb1d, [102](#)
- c26
  - modneighb3d, [105](#)
- c2\_3d\_sli
  - modneighb3d, [106](#)
- c2\_col
  - modneighb2d, [103](#)
- c2\_row
  - modneighb2d, [103](#)
- c4
  - modneighb2d, [103](#)
- c4\_3d
  - modneighb3d, [106](#)
- c6
  - modneighb3d, [107](#)
- c8
  - modneighb2d, [104](#)
- c8\_3d
  - modneighb3d, [107](#)
- can\_stop
  - mln::accu::logic::land\_basic, [450](#)
  - mln::accu::logic::lor\_basic, [453](#)
- Canvas, [99](#)
- card
  - mln::set, [393](#)
- cast
  - mln::value, [427](#)
- Category
  - mln::util::vertex, [1104](#)
- category
  - mln::util::edge, [1052](#)
- cell
  - mln::make, [338](#)
- center
  - mln::p\_centered, [855](#)
- center\_only\_iter
  - mln::topo::center\_only\_iter, [1001](#)
- center\_t
  - mln::graph\_elt\_mixed\_window, [736](#)
  - mln::graph\_elt\_window, [744](#)

- mln::graph\_window\_piter, 756
- center\_val
  - mln::dpoints\_bkd\_pixter, 651
  - mln::dpoints\_fwd\_pixter, 654
- centered\_bkd\_iter\_adapter
  - mln::topo::centered\_bkd\_iter\_adapter, 1003
- centered\_fwd\_iter\_adapter
  - mln::topo::centered\_fwd\_iter\_adapter, 1004
- chamfer
  - mln::geom, 255
- change
  - mln::p\_array, 850
- change\_both
  - mln::util::couple, 1049
  - mln::util::ord\_pair, 1079
- change\_extension
  - mln::extension\_val, 665
- change\_first
  - mln::util::couple, 1049
  - mln::util::ord\_pair, 1079
- change\_graph
  - mln::util::edge, 1052
  - mln::util::vertex, 1105
- change\_key
  - mln::p\_key, 883
- change\_keys
  - mln::p\_key, 883
- change\_mask
  - mln::graph\_elt\_window\_if, 750
- change\_second
  - mln::util::couple, 1049
  - mln::util::ord\_pair, 1079
- change\_target
  - mln::complex\_psite, 591
  - mln::faces\_psite, 668
  - mln::p\_transformed\_piter, 928
- change\_target\_site\_set
  - mln::graph\_window\_piter, 757
- change\_to
  - mln::pixel, 940
- check\_consistency
  - mln::util::tree, 1096
  - mln::util::tree\_node, 1099
- children
  - mln::util::tree\_node, 1100
- clear
  - mln::p\_array, 850
  - mln::p\_image, 876
  - mln::p\_key, 884
  - mln::p\_mutable\_array\_of, 892
  - mln::p\_priority, 899
  - mln::p\_queue, 904
  - mln::p\_queue\_fast, 909
  - mln::p\_set, 919
- mln::p\_set\_of, 923
- mln::util::array, 1040
- mln::util::fibonacci\_heap, 1056
- mln::util::set, 1084
- mln::util::soft\_heap, 1091
- mln::w\_window, 1156
- mln::window, 1174
- closing
  - mln::morpho::elementary, 369
- colorize
  - mln::labeling, 299, 300
- complementation
  - mln::morpho, 360
- complementation\_inplace
  - mln::morpho, 360
- complex
  - mln::topo::complex, 1006
- Complex based, 109
- complex\_geometry
  - mln::geom::complex\_geometry, 725
- complex\_image
  - mln::complex\_image, 586
- complex\_neighborhood\_bkd\_piter
  - mln::complex\_neighborhood\_bkd\_piter, 588
- complex\_neighborhood\_fwd\_piter
  - mln::complex\_neighborhood\_fwd\_piter, 589
- complex\_psite
  - mln::complex\_psite, 591
- complex\_window\_bkd\_piter
  - mln::complex\_window\_bkd\_piter, 594
- complex\_window\_fwd\_piter
  - mln::complex\_window\_fwd\_piter, 596
- compose
  - mln, 150
- composed
  - mln::fun::x2x::composed, 705
- compute
  - mln::accu, 166
  - mln::data, 206, 207
  - mln::graph, 266
  - mln::histo, 268
  - mln::labeling, 300–302
  - mln::labeling::impl::generic, 313, 314
  - mln::set, 393
- compute\_attribute\_image
  - mln::morpho::tree, 373
- compute\_attribute\_image\_from
  - mln::morpho::tree, 374
- compute\_fastest
  - mln::labeling::impl, 311
- compute\_has
  - mln::p\_queue\_fast, 909
- compute\_image
  - mln::labeling, 302, 303

- compute\_parent
  - mln::morpho::tree, 374
- compute\_with\_weights
  - mln::set, 393
- contrast
  - mln::morpho, 360
- convert
  - mln::data, 207
- convolve
  - mln::linear::local, 319
- coord
  - mln::def, 231
  - mln::doc::Dpoint, 605
  - mln::doc::Fastest\_Image, 608
  - mln::doc::Image, 618
  - mln::doc::Point\_Site, 629
  - mln::dpoint, 645
  - mln::point, 949
- coordf
  - mln::def, 231
- count
  - mln::accu::stat::mean, 519
- couple
  - mln::make, 339
- cplx
  - mln::p\_complex, 858
  - mln::p\_faces, 867, 868
  - mln::topo::algebraic\_face, 993
  - mln::topo::algebraic\_n\_face, 998
  - mln::topo::face, 1010
  - mln::topo::n\_face, 1021
- crop\_wrt
  - mln::box, 557
- cube3d
  - mln::win::cube3d, 1162
- cuboid3d
  - mln::win::cuboid3d, 1163
- cyan
  - mln::literal, 323
- D
  - mln::topo::is\_simple\_cell, 1019
- dark\_gray
  - mln::literal, 323
- dashed\_line
  - mln::draw, 235
- data
  - mln::topo::algebraic\_face, 993
  - mln::topo::algebraic\_n\_face, 998
  - mln::topo::face, 1010
  - mln::topo::n\_face, 1021
- data\_t
  - mln::fun::x2x::rotation, 708
  - mln::fun::x2x::translation, 711
- dec\_face\_id
  - mln::topo::algebraic\_face, 994
  - mln::topo::algebraic\_n\_face, 999
  - mln::topo::face, 1010
  - mln::topo::n\_face, 1021
- dec\_n
  - mln::topo::algebraic\_face, 994
  - mln::topo::face, 1010
- decorated\_image
  - mln::decorated\_image, 598
- decoration
  - mln::decorated\_image, 598
- deepness
  - mln::util::branch\_iter, 1045
  - mln::util::branch\_iter\_ind, 1047
- delete\_tree\_node
  - mln::util::tree\_node, 1100
- delta
  - mln::doc::Weighted\_Window, 641
  - mln::geom, 256
  - mln::graph\_elt\_mixed\_window, 737
  - mln::graph\_elt\_window, 745
  - mln::graph\_elt\_window\_if, 750
  - mln::graph\_window\_base, 752
  - mln::point, 949
  - mln::window, 1174
- delta\_index
  - mln::doc::Fastest\_Image, 611
  - mln::image1d, 767
  - mln::image2d, 772
  - mln::image3d, 780
- depth
  - mln::win::cuboid3d, 1164
- detach
  - mln::topo, 403
- detachment
  - mln::make, 339
- diag2d
  - mln::win::diag2d, 1165
- diameter
  - mln::win::ball, 1161
- diff
  - mln::Box, 564
  - mln::Site\_Set, 968
  - mln::win, 434
- diff\_abs
  - mln::arith, 177
- dilation
  - mln::morpho, 360
- dim
  - mln::complex\_image, 586
  - mln::doc::Dpoint, 605
  - mln::doc::Point\_Site, 630
  - mln::dpoint, 646

- mln::point, 950
- direct
  - mln::morpho::tree::filter, 380
- discrete\_plane\_1complex\_geometry
  - mln, 146
- discrete\_plane\_2complex\_geometry
  - mln, 146
- disk2d
  - modwin2d, 114
- display\_branch
  - mln::util, 417
- display\_tree
  - mln::util, 417
- distance\_and\_closest\_point\_geodesic
  - mln::transform, 410
- distance\_and\_influence\_zone\_geodesic
  - mln::transform, 411
- distance\_front
  - mln::canvas, 194
  - mln::transform, 411
- distance\_geodesic
  - mln::canvas, 194
  - mln::transform, 411
- div
  - mln::arith, 177
- div\_cst
  - mln::arith, 178
- div\_inplace
  - mln::arith, 178
- domain
  - mln::complex\_image, 586
  - mln::doc::Fastest\_Image, 611
  - mln::doc::Image, 620
  - mln::extended, 659
  - mln::flat\_image, 671
  - mln::hexa, 761
  - mln::image1d, 767
  - mln::image2d, 772
  - mln::image2d\_h, 776
  - mln::image3d, 781
  - mln::image\_if, 784
  - mln::lazy\_image, 798
  - mln::p2p\_image, 846
  - mln::slice\_image, 971
  - mln::sub\_image, 973
  - mln::sub\_image\_if, 974
  - mln::tr\_image, 1033
  - mln::unproject\_image, 1034
- Domain morphers, 97
- domain\_t
  - mln::value::stack\_image, 1143
- dp
  - mln::window, 1174
- dpoint
  - mln::doc::Dpoint, 605
  - mln::doc::Fastest\_Image, 608
  - mln::doc::Image, 618
  - mln::doc::Neighborhood, 624
  - mln::doc::Point\_Site, 629
  - mln::doc::Weighted\_Window, 640
  - mln::dpoint, 646, 647
- dpoint1d
  - mln, 146
- dpoint2d
  - mln, 146
- dpoint2d\_h
  - mln, 146
  - mln::make, 339
- dpoint3d
  - mln, 146
- dpoints\_bkd\_pixter
  - mln::dpoints\_bkd\_pixter, 650
- dpoints\_fwd\_pixter
  - mln::dpoints\_fwd\_pixter, 653
- dpsite
  - mln::point, 949
  - mln::w\_window, 1156
- draw\_graph
  - mln::debug, 226
- dual\_input\_max\_tree
  - mln::morpho::tree, 375
- dummy\_p\_edges
  - mln::make, 340
- dummy\_p\_vertices
  - mln::make, 340, 341
- duplicate
  - mln, 151
  - mln::border, 190
  - mln::extension, 241
- e\_ith\_nbh\_edge
  - mln::util::graph, 1062
  - mln::util::line\_graph, 1071
- e\_nmax
  - mln::util::graph, 1062
  - mln::util::line\_graph, 1071
- e\_nmax\_nbh\_edges
  - mln::util::graph, 1063
  - mln::util::line\_graph, 1071
- edge
  - mln::p\_edges, 861
  - mln::topo, 403
  - mln::util::edge, 1052
  - mln::util::graph, 1063
  - mln::util::line\_graph, 1071
- edge\_fwd\_iter
  - mln::util::graph, 1060
  - mln::util::line\_graph, 1070

- edge\_image
  - mln::edge\_image, 657
  - mln::make, 341, 342
- edge\_nbh\_edge\_fwd\_iter
  - mln::util::graph, 1060
  - mln::util::line\_graph, 1070
- edge\_nbh\_t
  - mln::edge\_image, 656
- edge\_win\_t
  - mln::edge\_image, 656
- edge\_with
  - mln::util::vertex, 1105
- edges
  - mln::util::graph, 1063
- edges\_set\_t
  - mln::util::graph, 1060
- edges\_t
  - mln::util::graph, 1060
  - mln::util::line\_graph, 1070
- eiter
  - mln::util::array, 1038
  - mln::util::set, 1084
- element
  - mln::box, 555
  - mln::graph\_window\_if\_piter, 754
  - mln::graph\_window\_piter, 758
  - mln::image1d, 767
  - mln::image2d, 772
  - mln::image3d, 781
  - mln::p\_array, 849
  - mln::p\_centered, 854
  - mln::p\_complex, 857
  - mln::p\_edges, 861
  - mln::p\_faces, 866
  - mln::p\_if, 871
  - mln::p\_image, 875
  - mln::p\_key, 882
  - mln::p\_line2d, 887
  - mln::p\_mutable\_array\_of, 891
  - mln::p\_priority, 898
  - mln::p\_queue, 903
  - mln::p\_queue\_fast, 908
  - mln::p\_run, 913
  - mln::p\_set, 918
  - mln::p\_set\_of, 922
  - mln::p\_transformed, 925
  - mln::p\_vaccess, 930
  - mln::p\_vertices, 934
  - mln::util::array, 1038
  - mln::util::set, 1084
  - mln::util::soft\_heap, 1090
- elt
  - mln::util::tree\_node, 1100
- empty
  - mln::p\_queue\_fast, 909
- enc
  - mln::value::float01, 1110
  - mln::value::label, 1127
  - mln::value::proxy, 1134
  - mln::value::sign, 1141
- end
  - mln::p\_line2d, 889
  - mln::p\_run, 915
- enlarge
  - mln::box, 557
- equalize
  - mln::border, 190
  - mln::histo, 268
- equiv
  - mln::value, 427
  - mln::value::float01, 1110
  - mln::value::proxy, 1134
  - mln::value::sign, 1141
- erosion
  - mln::morpho, 361
- exists\_key
  - mln::p\_key, 884
- exists\_priority
  - mln::p\_priority, 899
- extend
  - mln, 151
- extended
  - mln::extended, 658
- extension
  - mln::extension\_fun, 661
  - mln::extension\_ima, 663
  - mln::extension\_val, 665
- extension\_fun
  - mln::extension\_fun, 660
- extension\_ima
  - mln::extension\_ima, 663
- extension\_val
  - mln::extension\_val, 665
- f\_hsi\_to\_rgb\_3x8
  - mln::fun::v2v, 247
- f\_hsl\_to\_rgb\_3x8
  - mln::fun::v2v, 247
- f\_rgb\_to\_hsi\_f
  - mln::fun::v2v, 247
- f\_rgb\_to\_hsl\_f
  - mln::fun::v2v, 247
- face
  - mln::complex\_psite, 592
  - mln::faces\_psite, 668
  - mln::topo::face, 1009
- face\_bkd\_iter
  - mln::topo::face\_bkd\_iter, 1013

- face\_fwd\_iter
  - mln::topo::face\_fwd\_iter, 1014
- face\_id
  - mln::complex\_psite, 592
  - mln::faces\_psite, 668
  - mln::topo::algebraic\_face, 994
  - mln::topo::algebraic\_n\_face, 999
  - mln::topo::face, 1010
  - mln::topo::n\_face, 1021
- faces
  - mln::topo::n\_faces\_set, 1027
- faces\_psite
  - mln::faces\_psite, 667
- faces\_type
  - mln::topo::n\_faces\_set, 1026
- fast\_median
  - mln::data, 207
- fibonacci\_heap
  - mln::util::fibonacci\_heap, 1056
- filename
  - mln::debug, 227
- fill
  - mln::border, 191
  - mln::data, 208
  - mln::extension, 241
  - mln::util::array, 1040
- fill\_holes
  - mln::labeling, 304
- fill\_with\_image
  - mln::data, 208
  - mln::data::impl::generic, 221
- fill\_with\_value
  - mln::data, 208
  - mln::data::impl::generic, 221
- filter
  - mln::morpho::tree::filter, 380
- find
  - mln::border, 191
- first
  - mln::accu::pair, 501
  - mln::accu::stat::min\_max, 529
  - mln::util::couple, 1049
  - mln::util::ord\_pair, 1079
  - mln::util::site\_pair, 1088
- first\_accu
  - mln::accu::pair, 501
  - mln::accu::stat::min\_max, 529
- first\_element
  - mln::util::set, 1084
- flat\_image
  - mln::flat\_image, 670
- flat\_zones
  - mln::labeling, 304
- float01
  - mln::value::float01, 1110
- float01\_16
  - mln::value, 425
- float01\_8
  - mln::value, 425
- float01\_f
  - mln::value::float01\_f, 1112
- float\_2complex\_image3df
  - mln, 147
- flooding
  - mln::morpho::watershed, 382, 383
- foreground
  - mln::labeling, 304
- format
  - mln::debug, 227
- from\_to
  - mln::convert, 199, 200
- front
  - mln::p\_priority, 899
  - mln::p\_queue, 904
  - mln::p\_queue\_fast, 909
  - mln::util::fibonacci\_heap, 1056
- fun
  - mln::p2p\_image, 846
- fun\_image
  - mln::fun\_image, 714
- fun\_t
  - mln::p\_edges, 862
  - mln::p\_vertices, 935
- Function
  - mln::Function, 715
- function
  - mln::p\_edges, 864
  - mln::p\_transformed, 926
  - mln::p\_vertices, 937
- Functions, 100
- fwd\_citer
  - mln::topo::complex, 1005
- fwd\_eiter
  - mln::util::array, 1039
  - mln::util::set, 1084
- fwd\_niter
  - mln::doc::Neighborhood, 624
  - mln::graph\_elt\_mixed\_neighborhood, 734
  - mln::graph\_elt\_neighborhood, 740
  - mln::graph\_elt\_neighborhood\_if, 741
  - mln::mixed\_neighb, 831
  - mln::neighb, 843
- fwd\_piter
  - mln::box, 555
  - mln::doc::Box, 602
  - mln::doc::Fastest\_Image, 609
  - mln::doc::Image, 618
  - mln::doc::Site\_Set, 634



- mln::hexa, 760
- mln::image2d\_h, 775
- mln::p\_array, 849
- mln::p\_centered, 854
- mln::p\_complex, 857
- mln::p\_edges, 862
- mln::p\_faces, 866
- mln::p\_if, 871
- mln::p\_image, 875
- mln::p\_key, 882
- mln::p\_line2d, 887
- mln::p\_mutable\_array\_of, 891
- mln::p\_priority, 898
- mln::p\_queue, 903
- mln::p\_queue\_fast, 908
- mln::p\_run, 913
- mln::p\_set, 918
- mln::p\_set\_of, 922
- mln::p\_transformed, 925
- mln::p\_vaccess, 930
- mln::p\_vertices, 935
- fwd\_pixter1d
  - mln::fwd\_pixter1d, 720
- fwd\_pixter2d
  - mln::fwd\_pixter2d, 721
- fwd\_pixter3d
  - mln::fwd\_pixter3d, 723
- fwd\_qiter
  - mln::doc::Weighted\_Window, 640
  - mln::doc::Window, 642
  - mln::graph\_elt\_mixed\_window, 736
  - mln::graph\_elt\_window, 744
  - mln::graph\_elt\_window\_if, 748
  - mln::w\_window, 1156
  - mln::window, 1173
- fwd\_viter
  - mln::doc::Value\_Set, 638
  - mln::value::lut\_vec, 1131
- gaussian
  - mln::linear, 315
- gaussian\_1st\_derivative
  - mln::linear, 316
- gaussian\_2nd\_derivative
  - mln::linear, 316
- gaussian\_subsampling
  - mln::subsampling, 395
- general
  - mln::morpho, 361
- geom
  - mln::complex\_image, 585
  - mln::p\_complex, 858
- get
  - mln::border, 191
  - mln::set, 394
- get\_header
  - mln::io::dicom, 272
  - mln::io::dump, 273
  - mln::io::raw, 293
- get\_rot
  - mln::registration, 390
- gl16
  - mln::value, 425
- gl8
  - mln::value, 425
- glf
  - mln::value, 425
- gradient
  - mln::morpho, 361
- gradient\_external
  - mln::morpho, 361
- gradient\_internal
  - mln::morpho, 361
- graph
  - mln::p\_edges, 864
  - mln::p\_graph\_piter, 869
  - mln::p\_vertices, 937
  - mln::util::edge, 1052
  - mln::util::graph, 1061
  - mln::util::line\_graph, 1072
  - mln::util::vertex, 1105
- Graph based, 109
- graph\_element
  - mln::graph\_elt\_mixed\_window, 737
  - mln::graph\_elt\_window, 745
  - mln::graph\_window\_piter, 756
  - mln::p\_edges, 862
  - mln::p\_vertices, 935
- graph\_elt\_neighborhood\_if
  - mln::graph\_elt\_neighborhood\_if, 742
- graph\_elt\_window\_if
  - mln::graph\_elt\_window\_if, 750
- graph\_t
  - mln::edge\_image, 656
  - mln::p\_edges, 862
  - mln::p\_vertices, 935
  - mln::util::edge, 1052
  - mln::util::vertex, 1104
  - mln::vertex\_image, 1151
- graph\_window\_if\_piter
  - mln::graph\_window\_if\_piter, 754
- graph\_window\_piter
  - mln::graph\_window\_piter, 757
- Graphes, 94
- graylevel
  - mln::value::graylevel, 1114
- graylevel\_f
  - mln::value::graylevel\_f, 1117

- green
  - mln::literal, 323
- grid
  - mln::dpoint, 645
  - mln::point, 949
- h\_mat
  - mln::algebra::h\_mat, 545
  - mln::make, 343
- h\_vec
  - mln::algebra::h\_vec, 547
  - mln::point, 949
- has
  - mln::box, 557
  - mln::doc::Box, 603
  - mln::doc::Fastest\_Image, 611
  - mln::doc::Image, 620
  - mln::doc::Site\_Set, 634
  - mln::doc::Value\_Set, 638
  - mln::extension\_fun, 661
  - mln::extension\_ima, 663
  - mln::extension\_val, 666
  - mln::flat\_image, 671
  - mln::hexa, 761
  - mln::image1d, 768
  - mln::image2d, 773
  - mln::image2d\_h, 776
  - mln::image3d, 781
  - mln::interpolated, 786
  - mln::lazy\_image, 798
  - mln::p\_array, 850, 851
  - mln::p\_centered, 855
  - mln::p\_complex, 858
  - mln::p\_edges, 864
  - mln::p\_if, 872
  - mln::p\_image, 876
  - mln::p\_key, 884
  - mln::p\_line2d, 889
  - mln::p\_mutable\_array\_of, 892
  - mln::p\_priority, 899
  - mln::p\_queue, 904, 905
  - mln::p\_queue\_fast, 910
  - mln::p\_run, 915
  - mln::p\_set, 919
  - mln::p\_set\_of, 923
  - mln::p\_transformed, 926
  - mln::p\_vaccess, 931
  - mln::p\_vertices, 937
  - mln::set, 394
  - mln::tr\_image, 1033
  - mln::util::line\_graph, 1072
  - mln::util::set, 1085
  - mln::value::lut\_vec, 1132
  - mln::window, 1174
- has\_e
  - mln::util::graph, 1063
  - mln::util::line\_graph, 1072
- has\_index
  - mln::p\_run, 915
- has\_v
  - mln::util::graph, 1063
  - mln::util::line\_graph, 1072
- height
  - mln::morpho::attribute::sharpness, 837
  - mln::win::cuboid3d, 1164
  - mln::win::rectangle2d, 1170
- hexa
  - mln::hexa, 761
- higher\_dim\_adj\_faces
  - mln::topo::algebraic\_face, 994
  - mln::topo::algebraic\_n\_face, 999
  - mln::topo::face, 1010
  - mln::topo::n\_face, 1021
- highest\_priority
  - mln::p\_priority, 899
- histo3d\_rgb
  - mln::accu::stat::histo3d\_rgb, 514
- hit\_or\_miss
  - mln::morpho, 362
- hit\_or\_miss\_background\_closing
  - mln::morpho, 362
- hit\_or\_miss\_background\_opening
  - mln::morpho, 362
- hit\_or\_miss\_closing
  - mln::morpho, 362
- hit\_or\_miss\_opening
  - mln::morpho, 362
- hline2d
  - modwin2d, 114
- horizontal\_symmetry
  - mln::geom, 256
- hough
  - mln::transform, 411
- i\_element
  - mln::p\_array, 849
  - mln::p\_image, 875
  - mln::p\_key, 883
  - mln::p\_mutable\_array\_of, 891
  - mln::p\_priority, 898
  - mln::p\_queue, 904
  - mln::p\_queue\_fast, 908
  - mln::p\_set, 918
  - mln::p\_set\_of, 922
  - mln::p\_vaccess, 930
- icp
  - mln::registration, 390, 391
- id

- mln::graph\_window\_if\_piter, 754
- mln::graph\_window\_piter, 758
- mln::p\_graph\_piter, 869
- mln::util::edge, 1052
- mln::util::vertex, 1105
- id\_t
  - mln::util::edge, 1052
  - mln::util::vertex, 1104
- id\_value\_t
  - mln::util::edge, 1052
  - mln::util::vertex, 1104
- identity
  - mln::literal, 323
- Identity morphers, 98
- ima
  - mln::doc::Generalized\_Pixel, 616
  - mln::doc::Pixel\_Iterator, 627
  - mln::fun::x2x::linear, 706
  - mln::util::pix, 1081
- image
  - mln::bkd\_pixter1d, 549
  - mln::bkd\_pixter2d, 550
  - mln::bkd\_pixter3d, 551
  - mln::doc::Generalized\_Pixel, 615
  - mln::doc::Pixel\_Iterator, 627
  - mln::fwd\_pixter1d, 720
  - mln::fwd\_pixter2d, 721
  - mln::fwd\_pixter3d, 722
  - mln::make, 343, 344
  - mln::pw::image, 959
- Image morphers, 96
- image1d
  - mln::image1d, 766
- image2d
  - mln::image2d, 771
  - mln::make, 344
- image2d\_h
  - mln::image2d\_h, 776
- image3d
  - mln::image3d, 780
  - mln::make, 344
- image\_if
  - mln::image\_if, 783
- Images, 95
- implies
  - mln, 152
- inc\_face\_id
  - mln::topo::algebraic\_face, 994
  - mln::topo::algebraic\_n\_face, 999
  - mln::topo::face, 1010
  - mln::topo::n\_face, 1021
- inc\_n
  - mln::topo::algebraic\_face, 994
  - mln::topo::face, 1011
- index
  - mln::p\_indexed\_bkd\_piter, 878
  - mln::p\_indexed\_fwd\_piter, 880
- index\_of
  - mln::doc::Value\_Set, 638
  - mln::value::lut\_vec, 1132
- influence\_zone\_adjacency\_graph
  - mln::make, 344
- influence\_zone\_front
  - mln::transform, 412
- influence\_zone\_geodesic
  - mln::transform, 412
- influence\_zone\_geodesic\_saturated
  - mln::transform, 413
- init
  - mln::accu::center, 438
  - mln::accu::convolve, 439
  - mln::accu::count\_adjacent\_vertices, 441
  - mln::accu::count\_labels, 443
  - mln::accu::count\_value, 444
  - mln::accu::label\_used, 447
  - mln::accu::logic::land, 449
  - mln::accu::logic::land\_basic, 450
  - mln::accu::logic::lor, 452
  - mln::accu::logic::lor\_basic, 453
  - mln::accu::maj\_h, 455
  - mln::accu::math::count, 456
  - mln::accu::math::inf, 458
  - mln::accu::math::sum, 459
  - mln::accu::math::sup, 460
  - mln::accu::max\_site, 462
  - mln::accu::nil, 497
  - mln::accu::p, 498
  - mln::accu::pair, 501
  - mln::accu::rms, 503
  - mln::accu::shape::bbox, 504
  - mln::accu::shape::height, 506
  - mln::accu::shape::volume, 508
  - mln::accu::stat::deviation, 512
  - mln::accu::stat::histo3d\_rgb, 514
  - mln::accu::stat::max, 516
  - mln::accu::stat::max\_h, 517
  - mln::accu::stat::mean, 519
  - mln::accu::stat::median\_h, 523
  - mln::accu::stat::min, 525
  - mln::accu::stat::min\_h, 527
  - mln::accu::stat::min\_max, 529
  - mln::accu::stat::rank, 531
  - mln::accu::stat::rank< bool >, 532
  - mln::accu::stat::rank\_high\_quant, 534
  - mln::accu::stat::var, 536
  - mln::accu::stat::variance, 538
  - mln::accu::tuple, 540
  - mln::accu::val, 542

- mln::doc::Accumulator, 600
- mln::morpho::attribute::card, 833
- mln::morpho::attribute::count\_adjacent\_vertices, 834
- mln::morpho::attribute::height, 836
- mln::morpho::attribute::sharpness, 838
- mln::morpho::attribute::sum, 839
- mln::morpho::attribute::volume, 841
- mln::p\_run, 915
- initialize
  - mln, 152
- insert
  - mln::p\_array, 851
  - mln::p\_image, 876
  - mln::p\_key, 884
  - mln::p\_mutable\_array\_of, 892
  - mln::p\_priority, 900
  - mln::p\_queue, 905
  - mln::p\_queue\_fast, 910
  - mln::p\_set, 920
  - mln::p\_set\_of, 923
  - mln::p\_vaccess, 931, 932
  - mln::util::set, 1085
  - mln::w\_window, 1156
  - mln::window, 1174
- int\_s
  - mln::value::int\_s, 1120
- int\_s16
  - mln::value, 425
- int\_s32
  - mln::value, 426
- int\_s8
  - mln::value, 426
- int\_u
  - mln::value::int\_u, 1122
- int\_u12
  - mln::value, 426
- int\_u16
  - mln::value, 426
- int\_u32
  - mln::value, 426
- int\_u8
  - mln::value, 426
- int\_u8\_1complex\_image2d
  - mln, 147
- int\_u8\_2complex\_image2d
  - mln, 147
- int\_u8\_2complex\_image3df
  - mln, 147
- int\_u\_sat
  - mln::value::int\_u\_sat, 1124
- inter
  - mln::Box, 564
  - mln::Site\_Set, 968
- interpolated
  - mln::interpolated, 786
- inv
  - mln::fun::x2x::rotation, 709
  - mln::fun::x2x::translation, 712
- invalidate
  - mln::complex\_psite, 592
  - mln::doc::Iterator, 623
  - mln::doc::Pixel\_Iterator, 628
  - mln::doc::Site\_Iterator, 632
  - mln::doc::Value\_Iterator, 636
  - mln::dpoints\_bkd\_pixter, 651
  - mln::dpoints\_fwd\_pixter, 654
  - mln::faces\_psite, 668
  - mln::p\_edges, 864
  - mln::p\_vertices, 937
  - mln::topo::algebraic\_face, 994
  - mln::topo::algebraic\_n\_face, 999
  - mln::topo::face, 1011
  - mln::topo::n\_face, 1022
  - mln::util::branch\_iter, 1045
  - mln::util::branch\_iter\_ind, 1047
  - mln::util::edge, 1053
  - mln::util::vertex, 1105
- invert
  - mln::fun::x2x::rotation, 708
  - mln::fun::x2x::translation, 711
- iota
  - mln::debug, 228
- is\_centered
  - mln::doc::Weighted\_Window, 641
  - mln::graph\_elt\_mixed\_window, 737
  - mln::graph\_elt\_window, 745
  - mln::graph\_elt\_window\_if, 750
  - mln::graph\_window\_base, 752
  - mln::window, 1175
- is\_empty
  - mln::Box, 563
  - mln::box, 558
  - mln::doc::Weighted\_Window, 641
  - mln::graph\_elt\_mixed\_window, 738
  - mln::graph\_elt\_window, 746
  - mln::graph\_elt\_window\_if, 750
  - mln::graph\_window\_base, 752
  - mln::util::array, 1040
  - mln::util::fibonacci\_heap, 1056
  - mln::util::set, 1085
  - mln::util::soft\_heap, 1091
  - mln::window, 1175
- is\_facet
  - mln::topo, 403
- is\_subgraph\_of
  - mln::util::graph, 1064
  - mln::util::line\_graph, 1073

- is\_symmetric
  - mln::graph\_elt\_mixed\_window, 738
  - mln::graph\_elt\_window, 746
  - mln::graph\_elt\_window\_if, 750
  - mln::graph\_window\_base, 753
  - mln::w\_window, 1157
  - mln::window, 1175
- is\_valid
  - mln::accu::center, 438
  - mln::accu::convolve, 439
  - mln::accu::count\_adjacent\_vertices, 441
  - mln::accu::count\_labels, 443
  - mln::accu::count\_value, 444
  - mln::accu::histo, 446
  - mln::accu::label\_used, 447
  - mln::accu::logic::land, 449
  - mln::accu::logic::land\_basic, 450
  - mln::accu::logic::lor, 452
  - mln::accu::logic::lor\_basic, 453
  - mln::accu::maj\_h, 455
  - mln::accu::math::count, 456
  - mln::accu::math::inf, 458
  - mln::accu::math::sum, 459
  - mln::accu::math::sup, 460
  - mln::accu::max\_site, 462
  - mln::accu::nil, 497
  - mln::accu::p, 498
  - mln::accu::pair, 501
  - mln::accu::rms, 503
  - mln::accu::shape::bbox, 504
  - mln::accu::shape::height, 506
  - mln::accu::shape::volume, 508
  - mln::accu::stat::deviation, 512
  - mln::accu::stat::histo3d\_rgb, 514
  - mln::accu::stat::max, 516
  - mln::accu::stat::max\_h, 517
  - mln::accu::stat::mean, 519
  - mln::accu::stat::median\_alt, 521
  - mln::accu::stat::median\_h, 523
  - mln::accu::stat::min, 525
  - mln::accu::stat::min\_h, 527
  - mln::accu::stat::min\_max, 529
  - mln::accu::stat::rank, 531
  - mln::accu::stat::rank< bool >, 532
  - mln::accu::stat::rank\_high\_quant, 534
  - mln::accu::stat::var, 536
  - mln::accu::stat::variance, 538
  - mln::accu::tuple, 540
  - mln::accu::val, 542
  - mln::box, 558
  - mln::complex\_psite, 592
  - mln::doc::Fastest\_Image, 612
  - mln::doc::Image, 621
  - mln::doc::Iterator, 623
  - mln::doc::Pixel\_Iterator, 628
  - mln::doc::Site\_Iterator, 632
  - mln::doc::Value\_Iterator, 636
  - mln::dpoints\_bkd\_pixter, 651
  - mln::dpoints\_fwd\_pixter, 654
  - mln::faces\_psite, 668
  - mln::graph\_elt\_mixed\_window, 738
  - mln::graph\_elt\_window, 746
  - mln::graph\_elt\_window\_if, 751
  - mln::graph\_window\_base, 753
  - mln::interpolated, 786
  - mln::morpho::attribute::card, 833
  - mln::morpho::attribute::count\_adjacent\_vertices, 834
  - mln::morpho::attribute::height, 836
  - mln::morpho::attribute::sharpness, 838
  - mln::morpho::attribute::sum, 839
  - mln::morpho::attribute::volume, 841
  - mln::p\_array, 851
  - mln::p\_centered, 855
  - mln::p\_complex, 859
  - mln::p\_edges, 864
  - mln::p\_faces, 868
  - mln::p\_if, 872
  - mln::p\_image, 876
  - mln::p\_key, 885
  - mln::p\_line2d, 889
  - mln::p\_mutable\_array\_of, 892
  - mln::p\_priority, 900
  - mln::p\_queue, 905
  - mln::p\_queue\_fast, 910
  - mln::p\_run, 915
  - mln::p\_set, 920
  - mln::p\_set\_of, 923
  - mln::p\_transformed, 926
  - mln::p\_vaccess, 932
  - mln::p\_vertices, 938
  - mln::pixel, 940
  - mln::topo::algebraic\_face, 995
  - mln::topo::algebraic\_n\_face, 999
  - mln::topo::face, 1011
  - mln::topo::n\_face, 1022
  - mln::tr\_image, 1033
  - mln::util::branch\_iter, 1045
  - mln::util::branch\_iter\_ind, 1047
  - mln::util::edge, 1053
  - mln::util::fibonacci\_heap, 1057
  - mln::util::soft\_heap, 1091
  - mln::util::vertex, 1105
  - mln::value::stack\_image, 1144
- iter
  - mln::complex\_neighborhood\_bkd\_piter, 588
  - mln::complex\_neighborhood\_fwd\_piter, 590
  - mln::complex\_window\_bkd\_piter, 594

- mln::complex\_window\_fwd\_piter, 596
- iter\_type
  - mln::complex\_neighborhood\_bkd\_piter, 587
  - mln::complex\_neighborhood\_fwd\_piter, 589
  - mln::complex\_window\_bkd\_piter, 594
  - mln::complex\_window\_fwd\_piter, 595
- ith\_nbh\_edge
  - mln::util::edge, 1053
  - mln::util::vertex, 1105
- ith\_nbh\_vertex
  - mln::util::vertex, 1106
- k
  - mln::accu::stat::rank, 531
- key
  - mln::p\_key, 885
- keys
  - mln::p\_key, 885
- l1
  - mln::norm, 385
- l1\_distance
  - mln::norm, 385
- l2
  - mln::norm, 386
- l2\_distance
  - mln::norm, 386
- label
  - mln::value::label, 1128
- label\_16
  - mln::value, 426
- label\_32
  - mln::value, 426
- label\_8
  - mln::value, 426
- labeled\_image
  - mln::labeled\_image, 791
- labeled\_image\_base
  - mln::labeled\_image\_base, 794
- labeling
  - mln::graph, 266
- laplacian
  - mln::morpho, 363
- larger\_than
  - mln, 152
- last
  - mln::util::array, 1040
- last\_coord
  - mln::point, 951
- last\_element
  - mln::util::set, 1086
- lazy\_image
  - mln::lazy\_image, 798
- ldlt\_decomp
  - mln::algebra, 174
- ldlt\_solve
  - mln::algebra, 174
- lemmings
  - mln::util, 417
- len
  - mln::Box, 563
  - mln::box, 558
- length
  - mln::p\_run, 916
  - mln::win::backdiag2d, 1160
  - mln::win::cube3d, 1162
  - mln::win::diag2d, 1165
  - mln::win::line, 1167
  - mln::win::octagon2d, 1169
- light\_gray
  - mln::literal, 323
- lime
  - mln::literal, 323
- line
  - mln::accu, 166
  - mln::draw, 235
  - mln::win::line, 1166
- line\_gradient
  - mln::morpho, 363
- linear
  - mln::fun::x2x::linear, 706
- linfty
  - mln::norm, 386
- linfty\_distance
  - mln::norm, 386
- load
  - mln::io::cloud, 271
  - mln::io::dicom, 272
  - mln::io::dump, 273
  - mln::io::fits, 274, 275
  - mln::io::fld, 276
  - mln::io::magick, 277
  - mln::io::off, 278
  - mln::io::pbm, 280
  - mln::io::pbms, 281
  - mln::io::pfm, 283
  - mln::io::pgm, 284
  - mln::io::pgms, 285
  - mln::io::plot, 286
  - mln::io::pnm, 288
  - mln::io::pnms, 290
  - mln::io::ppm, 291
  - mln::io::ppms, 292
  - mln::io::raw, 293
  - mln::io::tiff, 294
- load\_ascii\_builtin
  - mln::io::pnm, 288
- load\_ascii\_value

- mln::io::pnm, 288
- load\_raw\_2d
  - mln::io::pnm, 288
- lower\_dim\_adj\_faces
  - mln::topo::algebraic\_face, 995
  - mln::topo::algebraic\_n\_face, 1000
  - mln::topo::face, 1011
  - mln::topo::n\_face, 1022
- lowest\_priority
  - mln::p\_priority, 900
- lut\_vec
  - mln::value::lut\_vec, 1132
- lvalue
  - mln::complex\_image, 585
  - mln::decorated\_image, 597
  - mln::doc::Fastest\_Image, 609
  - mln::doc::Image, 619
  - mln::doc::Pixel\_Iterator, 627
  - mln::flat\_image, 670
  - mln::fun\_image, 713
  - mln::hexa, 760
  - mln::image1d, 766
  - mln::image2d, 771
  - mln::image2d\_h, 775
  - mln::image3d, 779
  - mln::interpolated, 785
  - mln::lazy\_image, 797
  - mln::tr\_image, 1032
  - mln::value::stack\_image, 1143
  - mln::violent\_cast\_image, 1153
- magenta
  - mln::literal, 324
- main\_branch
  - mln::util::tree, 1097
- make\_algebraic\_face
  - mln::topo, 403
- make\_debug\_graph\_image
  - mln, 152
- make\_greater\_point
  - mln::util, 418
- make\_greater\_psite
  - mln::util, 418
- mask
  - mln::graph\_elt\_neighborhood\_if, 742
  - mln::graph\_elt\_window\_if, 751
- mask\_t
  - mln::graph\_elt\_window\_if, 749
- mat
  - mln::make, 345
- max
  - mln::literal, 324
  - mln::morpho::tree::filter, 380
- max\_col
  - mln::geom, 256
- max\_component
  - mln::io::pnm, 289
- max\_ind
  - mln::geom, 256
- max\_row
  - mln::geom, 257
- max\_sli
  - mln::geom, 257
- max\_tree
  - mln::morpho::tree, 376
- mean
  - mln::accu::stat::var, 536
  - mln::accu::stat::variance, 538
  - mln::estim, 238
- mean\_t
  - mln::accu::stat::var, 536
- median
  - mln::data, 209
  - mln::data::approx, 215, 216
  - mln::data::naive, 223
- medium\_gray
  - mln::literal, 324
- memory\_size
  - mln::box, 558
  - mln::p\_array, 851
  - mln::p\_centered, 855
  - mln::p\_edges, 865
  - mln::p\_if, 872
  - mln::p\_image, 876
  - mln::p\_key, 885
  - mln::p\_line2d, 889
  - mln::p\_mutable\_array\_of, 893
  - mln::p\_priority, 900
  - mln::p\_queue, 905
  - mln::p\_queue\_fast, 910
  - mln::p\_run, 916
  - mln::p\_set, 920
  - mln::p\_set\_of, 923
  - mln::p\_transformed, 926
  - mln::p\_vaccess, 932
  - mln::p\_vertices, 938
  - mln::util::array, 1041
  - mln::util::set, 1086
- merge
  - mln::box, 558
- mesh
  - mln::doc::Point\_Site, 629
- mesh\_corner\_point\_area
  - mln::geom, 257
- mesh\_curvature
  - mln::geom, 258
- mesh\_normal
  - mln::geom, 258

- meyer\_wst
  - mln::morpho, 363
- min
  - mln::arith, 179
  - mln::literal, 324
  - mln::morpho, 364
  - mln::morpho::tree::filter, 381
- min\_col
  - mln::geom, 258
- min\_ind
  - mln::geom, 259
- min\_inplace
  - mln::arith, 179
  - mln::morpho, 364
- min\_max
  - mln::estim, 238
- min\_row
  - mln::geom, 259
- min\_sli
  - mln::geom, 259
- min\_tree
  - mln::morpho::tree, 376
- minus
  - mln::arith, 179, 180
  - mln::morpho, 364
- minus\_cst
  - mln::arith, 180, 181
- minus\_cst\_inplace
  - mln::arith, 181
- minus\_infty
  - mln::point, 951
- minus\_inplace
  - mln::arith, 182
- mirror
  - mln::border, 192
- mixed\_neighb
  - mln::mixed\_neighb, 832
- mln, 121
  - a\_point\_of, 150
  - apply\_p2p, 150
  - before, 163
  - bin\_1complex\_image2d, 145
  - bin\_2complex\_image3df, 145
  - box1d, 145
  - box2d, 145
  - box2d\_h, 145
  - box3d, 146
  - compose, 150
  - discrete\_plane\_1complex\_geometry, 146
  - discrete\_plane\_2complex\_geometry, 146
  - dpoint1d, 146
  - dpoint2d, 146
  - dpoint2d\_h, 146
  - dpoint3d, 146
  - duplicate, 151
  - extend, 151
  - float\_2complex\_image3df, 147
  - implies, 152
  - initialize, 152
  - int\_u8\_1complex\_image2d, 147
  - int\_u8\_2complex\_image2d, 147
  - int\_u8\_2complex\_image3df, 147
  - larger\_than, 152
  - make\_debug\_graph\_image, 152
  - mln\_exact, 153
  - mln\_gen\_complex\_neighborhood, 153
  - mln\_gen\_complex\_window, 154
  - mln\_gen\_complex\_window\_p, 154, 155
  - mln\_regular, 155
  - mln\_trait\_op\_geq, 155
  - mln\_trait\_op\_greater, 155
  - mln\_trait\_op\_leq, 156
  - mln\_trait\_op\_neq, 156
  - operator<, 158
  - operator<<, 158, 159
  - operator<=, 159, 160
  - operator\*, 157
  - operator++, 157
  - operator-, 157
  - operator--, 158
  - operator==, 160, 161
  - p\_run2d, 147
  - p\_runs2d, 147
  - point1d, 147
  - point1df, 148
  - point2d, 148
  - point2d\_h, 148
  - point2df, 148
  - point3d, 148
  - point3df, 148
  - primary, 163
  - ptransform, 163
  - rgb8\_2complex\_image3df, 148
  - sagittal\_dec, 163
  - space\_2complex\_geometry, 148
  - unsigned\_2complex\_image3df, 149
  - up, 163
  - vec2d\_d, 149
  - vec2d\_f, 149
  - vec3d\_d, 149
  - vec3d\_f, 149
  - w\_window1d\_float, 149
  - w\_window1d\_int, 149
  - w\_window2d\_float, 149
  - w\_window2d\_int, 150
  - w\_window3d\_float, 150
  - w\_window3d\_int, 150
- mln::accu, 164



- compute, 166
- line, 166
- mln\_meta\_accu\_result, 166
- take, 167
- mln::accu::center, 437
  - init, 438
  - is\_valid, 438
  - nsites, 438
  - take\_as\_init, 438
  - take\_n\_times, 438
  - to\_result, 438
- mln::accu::convolve, 439
  - init, 439
  - is\_valid, 439
  - take\_as\_init, 440
  - take\_n\_times, 440
  - to\_result, 440
- mln::accu::count\_adjacent\_vertices, 440
  - init, 441
  - is\_valid, 441
  - set\_value, 441
  - take\_as\_init, 441
  - take\_n\_times, 441
  - to\_result, 442
- mln::accu::count\_labels, 442
  - init, 443
  - is\_valid, 443
  - set\_value, 443
  - take\_as\_init, 443
  - take\_n\_times, 443
  - to\_result, 443
- mln::accu::count\_value, 443
  - init, 444
  - is\_valid, 444
  - set\_value, 444
  - take\_as\_init, 444
  - take\_n\_times, 445
  - to\_result, 445
- mln::accu::histo, 445
  - is\_valid, 446
  - take, 446
  - take\_as\_init, 446
  - take\_n\_times, 446
  - vect, 446
- mln::accu::image, 167
- mln::accu::impl, 167
- mln::accu::label\_used, 446
  - init, 447
  - is\_valid, 447
  - take, 447
  - take\_as\_init, 447
  - take\_n\_times, 448
  - to\_result, 448
- mln::accu::logic, 168
  - mln::accu::logic::land, 448
    - init, 449
    - is\_valid, 449
    - take\_as\_init, 449
    - take\_n\_times, 449
    - to\_result, 449
  - mln::accu::logic::land\_basic, 449
    - can\_stop, 450
    - init, 450
    - is\_valid, 450
    - take\_as\_init, 450
    - take\_n\_times, 451
    - to\_result, 451
  - mln::accu::logic::lor, 451
    - init, 452
    - is\_valid, 452
    - take\_as\_init, 452
    - take\_n\_times, 452
    - to\_result, 452
  - mln::accu::logic::lor\_basic, 452
    - can\_stop, 453
    - init, 453
    - is\_valid, 453
    - take\_as\_init, 453
    - take\_n\_times, 453
    - to\_result, 454
  - mln::accu::maj\_h, 454
    - init, 455
    - is\_valid, 455
    - take\_as\_init, 455
    - take\_n\_times, 455
    - to\_result, 455
  - mln::accu::math, 168
    - mln::accu::math::count, 455
      - init, 456
      - is\_valid, 456
      - set\_value, 456
      - take\_as\_init, 456
      - take\_n\_times, 456
      - to\_result, 457
    - mln::accu::math::inf, 457
      - init, 458
      - is\_valid, 458
      - take\_as\_init, 458
      - take\_n\_times, 458
      - to\_result, 458
    - mln::accu::math::sum, 458
      - init, 459
      - is\_valid, 459
      - take\_as\_init, 459
      - take\_n\_times, 459
      - to\_result, 459
    - mln::accu::math::sup, 460
      - init, 460

- is\_valid, 460
- take\_as\_init, 461
- take\_n\_times, 461
- to\_result, 461
- mln::accu::max\_site, 461
  - init, 462
  - is\_valid, 462
  - take\_as\_init, 462
  - take\_n\_times, 462
  - to\_result, 462
- mln::accu::meta::center, 463
- mln::accu::meta::count\_adjacent\_vertices, 463
- mln::accu::meta::count\_labels, 464
- mln::accu::meta::count\_value, 465
- mln::accu::meta::histo, 466
- mln::accu::meta::label\_used, 467
- mln::accu::meta::logic, 169
- mln::accu::meta::logic::land, 468
- mln::accu::meta::logic::land\_basic, 469
- mln::accu::meta::logic::lor, 470
- mln::accu::meta::logic::lor\_basic, 471
- mln::accu::meta::maj\_h, 472
- mln::accu::meta::math, 169
- mln::accu::meta::math::count, 473
- mln::accu::meta::math::inf, 474
- mln::accu::meta::math::sum, 475
- mln::accu::meta::math::sup, 476
- mln::accu::meta::max\_site, 477
- mln::accu::meta::nil, 478
- mln::accu::meta::p, 479
- mln::accu::meta::pair, 480
- mln::accu::meta::rms, 481
- mln::accu::meta::shape, 170
- mln::accu::meta::shape::bbox, 482
- mln::accu::meta::shape::height, 483
- mln::accu::meta::shape::volume, 484
- mln::accu::meta::stat, 170
- mln::accu::meta::stat::max, 485
- mln::accu::meta::stat::max\_h, 486
- mln::accu::meta::stat::mean, 487
- mln::accu::meta::stat::median\_alt, 488
- mln::accu::meta::stat::median\_h, 489
- mln::accu::meta::stat::min, 490
- mln::accu::meta::stat::min\_h, 491
- mln::accu::meta::stat::rank, 492
- mln::accu::meta::stat::rank\_high\_quant, 493
- mln::accu::meta::tuple, 494
- mln::accu::meta::val, 495
- mln::accu::nil, 496
  - init, 497
  - is\_valid, 497
  - take\_as\_init, 497
  - take\_n\_times, 497
  - to\_result, 497
- mln::accu::p, 498
  - init, 498
  - is\_valid, 498
  - take\_as\_init, 499
  - take\_n\_times, 499
  - to\_result, 499
- mln::accu::pair, 499
  - first, 501
  - first\_accu, 501
  - init, 501
  - is\_valid, 501
  - second, 501
  - second\_accu, 501
  - take\_as\_init, 502
  - take\_n\_times, 502
  - to\_result, 502
- mln::accu::rms, 502
  - init, 503
  - is\_valid, 503
  - take\_as\_init, 503
  - take\_n\_times, 503
  - to\_result, 503
- mln::accu::shape, 171
- mln::accu::shape::bbox, 504
  - init, 504
  - is\_valid, 504
  - take\_as\_init, 504
  - take\_n\_times, 505
  - to\_result, 505
- mln::accu::shape::height, 505
  - argument, 506
  - init, 506
  - is\_valid, 506
  - set\_value, 506
  - take\_as\_init, 507
  - take\_n\_times, 507
  - to\_result, 507
  - value, 506
- mln::accu::shape::volume, 507
  - argument, 508
  - init, 508
  - is\_valid, 508
  - set\_value, 509
  - take\_as\_init, 509
  - take\_n\_times, 509
  - to\_result, 509
  - value, 508
- mln::accu::site\_set::rectangularity, 509
  - area, 510
  - bbox, 510
  - rectangularity, 510
  - take\_as\_init, 510
  - take\_n\_times, 511
  - to\_result, 511

- mln::accu::stat, 171
  - operator==, 173
- mln::accu::stat::deviation, 511
  - init, 512
  - is\_valid, 512
  - take\_as\_init, 512
  - take\_n\_times, 512
  - to\_result, 512
- mln::accu::stat::histo3d\_rgb, 513
  - histo3d\_rgb, 514
  - init, 514
  - is\_valid, 514
  - take, 514
  - take\_as\_init, 515
  - take\_n\_times, 515
  - to\_result, 515
- mln::accu::stat::max, 515
  - init, 516
  - is\_valid, 516
  - set\_value, 516
  - take\_as\_init, 516
  - take\_n\_times, 516
  - to\_result, 516
- mln::accu::stat::max\_h, 517
  - init, 517
  - is\_valid, 517
  - take\_as\_init, 518
  - take\_n\_times, 518
  - to\_result, 518
- mln::accu::stat::mean, 518
  - count, 519
  - init, 519
  - is\_valid, 519
  - sum, 519
  - take\_as\_init, 519
  - take\_n\_times, 520
  - to\_result, 520
- mln::accu::stat::median\_alt, 520
  - is\_valid, 521
  - take, 521
  - take\_as\_init, 521
  - take\_n\_times, 521
  - to\_result, 521
- mln::accu::stat::median\_h, 522
  - init, 523
  - is\_valid, 523
  - take\_as\_init, 523
  - take\_n\_times, 523
  - to\_result, 523
- mln::accu::stat::meta::deviation, 523
- mln::accu::stat::min, 524
  - init, 525
  - is\_valid, 525
  - set\_value, 525
  - take\_as\_init, 525
  - take\_n\_times, 525
  - to\_result, 526
- mln::accu::stat::min\_h, 526
  - init, 527
  - is\_valid, 527
  - take\_as\_init, 527
  - take\_n\_times, 527
  - to\_result, 527
- mln::accu::stat::min\_max, 527
  - first, 529
  - first\_accu, 529
  - init, 529
  - is\_valid, 529
  - second, 529
  - second\_accu, 529
  - take\_as\_init, 529
  - take\_n\_times, 530
  - to\_result, 530
- mln::accu::stat::rank, 530
  - init, 531
  - is\_valid, 531
  - k, 531
  - take\_as\_init, 531
  - take\_n\_times, 531
  - to\_result, 531
- mln::accu::stat::rank< bool >, 532
  - init, 532
  - is\_valid, 532
  - take\_as\_init, 533
  - take\_n\_times, 533
  - to\_result, 533
- mln::accu::stat::rank\_high\_quant, 533
  - init, 534
  - is\_valid, 534
  - take\_as\_init, 534
  - take\_n\_times, 534
  - to\_result, 534
- mln::accu::stat::var, 535
  - init, 536
  - is\_valid, 536
  - mean, 536
  - mean\_t, 536
  - n\_items, 536
  - take\_as\_init, 536
  - take\_n\_times, 536
  - to\_result, 536
  - variance, 537
- mln::accu::stat::variance, 537
  - init, 538
  - is\_valid, 538
  - mean, 538
  - n\_items, 538
  - standard\_deviation, 538

- sum, 539
- take\_as\_init, 539
- take\_n\_times, 539
- to\_result, 539
- var, 539
- mln::accu::tuple, 539
  - init, 540
  - is\_valid, 540
  - take\_as\_init, 540
  - take\_n\_times, 541
  - to\_result, 541
- mln::accu::val, 541
  - init, 542
  - is\_valid, 542
  - take\_as\_init, 542
  - take\_n\_times, 542
  - to\_result, 542
- mln::Accumulator, 542
  - take\_as\_init, 544
  - take\_n\_times, 544
- mln::algebra, 173
  - ldlt\_decomp, 174
  - ldlt\_solve, 174
  - operator\*, 174
  - vprod, 174
- mln::algebra::h\_mat, 544
  - \_1, 546
  - h\_mat, 545
  - t, 546
- mln::algebra::h\_vec, 546
  - h\_vec, 547
  - operator mat< n, 1, U >, 547
  - origin, 548
  - t, 547
  - to\_vec, 548
  - zero, 548
- mln::arith, 175
  - diff\_abs, 177
  - div, 177
  - div\_cst, 178
  - div\_inplace, 178
  - min, 179
  - min\_inplace, 179
  - minus, 179, 180
  - minus\_cst, 180, 181
  - minus\_cst\_inplace, 181
  - minus\_inplace, 182
  - plus, 182, 183
  - plus\_cst, 183, 184
  - plus\_cst\_inplace, 184
  - plus\_inplace, 185
  - revert, 185
  - revert\_inplace, 186
  - times, 186
  - times\_cst, 186
  - times\_inplace, 187
- mln::arith::impl, 187
- mln::arith::impl::generic, 188
- mln::binarization, 188
  - binarization, 188
  - threshold, 188
- mln::bkd\_pixter1d, 548
  - bkd\_pixter1d, 549
  - image, 549
  - next, 549
- mln::bkd\_pixter2d, 549
  - bkd\_pixter2d, 550
  - image, 550
  - next, 550
- mln::bkd\_pixter3d, 551
  - bkd\_pixter3d, 552
  - image, 551
  - next, 552
- mln::border, 189
  - adjust, 190
  - duplicate, 190
  - equalize, 190
  - fill, 191
  - find, 191
  - get, 191
  - mirror, 192
  - resize, 192
- mln::border::impl, 192
- mln::border::impl::generic, 193
- mln::Box, 560
  - bbox, 563
  - diff, 564
  - inter, 564
  - is\_empty, 563
  - len, 563
  - nsites, 563
  - operator<, 564
  - operator<<, 564
  - operator<=, 564, 565
  - operator==, 565
  - sym\_diff, 565
  - uni, 565
  - unique, 565
- mln::box, 552
  - bbox, 557
  - bkd\_piter, 555
  - box, 556
  - crop\_wrt, 557
  - element, 555
  - enlarge, 557
  - fwd\_piter, 555
  - has, 557
  - is\_empty, 558

- is\_valid, 558
- len, 558
- memory\_size, 558
- merge, 558
- nsites, 558
- operator<<, 560
- pcenter, 559
- piter, 556
- pmax, 559
- pmin, 559
- psite, 556
- site, 556
- to\_larger, 559
- mln::box\_runend\_piter, 566
  - box\_runend\_piter, 566
  - next, 567
  - run\_length, 567
- mln::box\_runstart\_piter, 567
  - box\_runstart\_piter, 568
  - next, 568
  - run\_length, 568
- mln::Browsing, 568
- mln::canvas, 193
  - distance\_front, 194
  - distance\_geodesic, 194
- mln::canvas::browsing, 194
- mln::canvas::browsing::backdiagonal2d\_t, 569
- mln::canvas::browsing::breadth\_first\_search\_t, 571
- mln::canvas::browsing::depth\_first\_search\_t, 571
- mln::canvas::browsing::diagonal2d\_t, 571
- mln::canvas::browsing::dir\_struct\_elt\_incr\_update\_t, 573
- mln::canvas::browsing::directional\_t, 574
- mln::canvas::browsing::fwd\_t, 576
- mln::canvas::browsing::hyper\_directional\_t, 577
- mln::canvas::browsing::snake\_fwd\_t, 579
- mln::canvas::browsing::snake\_generic\_t, 580
- mln::canvas::browsing::snake\_vert\_t, 582
- mln::canvas::chamfer, 583
- mln::canvas::impl, 195
- mln::canvas::labeling, 195
  - blobs, 196
- mln::canvas::labeling::impl, 196
- mln::canvas::morpho, 197
- mln::category< R(\*) (A) >, 583
- mln::complex\_image, 584
  - complex\_image, 586
  - dim, 586
  - domain, 586
  - geom, 585
  - lvalue, 585
  - operator(), 586
  - rvalue, 585
  - skeleton, 585
  - value, 585
  - values, 586
- mln::complex\_neighborhood\_bkd\_piter, 587
  - complex\_neighborhood\_bkd\_piter, 588
  - iter, 588
  - iter\_type, 587
  - next, 588
  - psite, 587
- mln::complex\_neighborhood\_fwd\_piter, 588
  - complex\_neighborhood\_fwd\_piter, 589
  - iter, 590
  - iter\_type, 589
  - next, 590
  - psite, 589
- mln::complex\_psite, 590
  - change\_target, 591
  - complex\_psite, 591
  - face, 592
  - face\_id, 592
  - invalidate, 592
  - is\_valid, 592
  - n, 592
  - site\_set, 592
- mln::complex\_window\_bkd\_piter, 593
  - complex\_window\_bkd\_piter, 594
  - iter, 594
  - iter\_type, 594
  - next, 594
  - psite, 594
- mln::complex\_window\_fwd\_piter, 595
  - complex\_window\_fwd\_piter, 596
  - iter, 596
  - iter\_type, 595
  - next, 596
  - psite, 595
- mln::convert, 197
  - from\_to, 199, 200
  - mln\_image\_from\_grid, 200
  - mln\_window, 200
  - to, 200
  - to\_dpoint, 201
  - to\_fun, 201, 203
  - to\_image, 201
  - to\_p\_array, 201
  - to\_p\_set, 201, 202
  - to\_qimage, 202
  - to\_upper\_window, 202, 203
  - to\_window, 203
- mln::data, 203
  - abs, 206
  - abs\_inplace, 206
  - apply, 206
  - compute, 206, 207
  - convert, 207

- fast\_median, 207
- fill, 208
- fill\_with\_image, 208
- fill\_with\_value, 208
- median, 209
- mln\_meta\_accu\_result, 209
- paste, 209
- paste\_without\_localization, 210
- replace, 210
- saturate, 210, 211
- saturate\_inplace, 211
- sort\_offsets\_increasing, 211
- sort\_psites\_decreasing, 211
- sort\_psites\_increasing, 212
- stretch, 212
- to\_enc, 212
- transform, 213
- transform\_inplace, 213, 214
- update, 214
- wrap, 214
- mln::data::approx, 215
  - median, 215, 216
- mln::data::approx::impl, 217
- mln::data::impl, 217
  - paste\_without\_localization\_fast, 218
  - paste\_without\_localization\_fastest, 218
  - paste\_without\_localization\_lines, 218
  - stretch, 219
  - transform\_inplace\_lowq, 219
  - update\_fastest, 219
- mln::data::impl::generic, 220
  - fill\_with\_image, 221
  - fill\_with\_value, 221
  - paste, 221
  - transform, 221, 222
  - transform\_inplace, 222
  - update, 223
- mln::data::naive, 223
  - median, 223
- mln::data::naive::impl, 224
- mln::debug, 224
  - draw\_graph, 226
  - filename, 227
  - format, 227
  - iota, 228
  - mosaic, 228
  - println, 228
  - println\_with\_border, 228
  - put\_word, 229
  - slices\_2d, 229
  - superpose, 229
  - z\_order, 230
- mln::debug::impl, 230
- mln::decorated\_image, 596
  - ~decorated\_image, 598
  - decorated\_image, 598
  - decoration, 598
  - lvalue, 597
  - operator decorated\_image< const I, D >, 599
  - operator(), 599
  - psite, 598
  - rvalue, 598
  - skeleton, 598
- mln::def, 230
  - coord, 231
  - coordf, 231
- mln::Delta\_Point\_Site< void >, 599
- mln::display, 231
- mln::display::impl, 232
- mln::display::impl::generic, 232
- mln::doc, 232
- mln::doc::Accumulator, 599
  - argument, 600
  - init, 600
  - take, 600
- mln::doc::Box, 601
  - bbox, 603
  - bkd\_piter, 602
  - fwd\_piter, 602
  - has, 603
  - nsites, 603
  - pmax, 603
  - pmin, 603
  - psite, 602
  - site, 602
- mln::doc::Dpoint, 604
  - coord, 605
  - dim, 605
  - dpoint, 605
  - point, 605
- mln::doc::Fastest\_Image, 606
  - bbox, 610
  - bkd\_piter, 608
  - border, 610
  - buffer, 611
  - coord, 608
  - delta\_index, 611
  - domain, 611
  - dpoint, 608
  - fwd\_piter, 609
  - has, 611
  - is\_valid, 612
  - lvalue, 609
  - nelements, 612
  - nsites, 612
  - operator(), 612
  - point, 609
  - point\_at\_index, 613

- pset, 609
- psite, 609
- rvalue, 610
- skeleton, 610
- value, 610
- values, 614
- vset, 610
- mln::doc::Generalized\_Pixel, 614
  - ima, 616
  - image, 615
  - rvalue, 615
  - val, 616
  - value, 615
- mln::doc::Image, 616
  - bbox, 620
  - bkd\_piter, 618
  - coord, 618
  - domain, 620
  - dpoint, 618
  - fwd\_piter, 618
  - has, 620
  - is\_valid, 621
  - lvalue, 619
  - nsites, 621
  - operator(), 621
  - point, 619
  - pset, 619
  - psite, 619
  - rvalue, 619
  - skeleton, 619
  - value, 620
  - values, 621
  - vset, 620
- mln::doc::Iterator, 622
  - invalidate, 623
  - is\_valid, 623
  - start, 623
- mln::doc::Neighborhood, 623
  - bkd\_niter, 624
  - dpoint, 624
  - fwd\_niter, 624
  - niter, 624
  - point, 624
- mln::doc::Object, 625
- mln::doc::Pixel\_Iterator, 625
  - ima, 627
  - image, 627
  - invalidate, 628
  - is\_valid, 628
  - lvalue, 627
  - rvalue, 627
  - start, 628
  - val, 628
  - value, 627
- mln::doc::Point\_Site
  - dim, 630
- mln::doc::Point\_Site, 628
  - coord, 629
  - dpoint, 629
  - mesh, 629
  - point, 629
  - to\_point, 630
- mln::doc::Site\_Iterator, 631
  - invalidate, 632
  - is\_valid, 632
  - operator psite, 632
  - psite, 632
  - start, 632
- mln::doc::Site\_Set, 633
  - bkd\_piter, 634
  - fwd\_piter, 634
  - has, 634
  - psite, 634
  - site, 634
- mln::doc::Value\_Iterator, 635
  - invalidate, 636
  - is\_valid, 636
  - operator value, 636
  - start, 636
  - value, 636
- mln::doc::Value\_Set, 636
  - bkd\_viter, 638
  - fwd\_viter, 638
  - has, 638
  - index\_of, 638
  - nvalues, 638
  - value, 638
- mln::doc::Weighted\_Window, 639
  - bkd\_qiter, 640
  - delta, 641
  - dpoint, 640
  - fwd\_qiter, 640
  - is\_centered, 641
  - is\_empty, 641
  - point, 640
  - sym, 641
  - weight, 641
  - win, 641
  - window, 641
- mln::doc::Window, 641
  - bkd\_qiter, 642
  - fwd\_qiter, 642
  - qiter, 643
- mln::Dpoint, 643
  - to\_dpoint, 643
- mln::dpoint, 644
  - coord, 645
  - dim, 646

- dpoint, 646, 647
- grid, 645
- operator mln::algebra::vec< dpoint< G, C  
>::dim, Q >, 647
- psite, 646
- set\_all, 648
- site, 646
- to\_vec, 648
- vec, 646
- mln::dpoints\_bkd\_pixter, 648
  - center\_val, 651
  - dpoints\_bkd\_pixter, 650
  - invalidate, 651
  - is\_valid, 651
  - next, 651
  - start, 651
  - update, 651
- mln::dpoints\_fwd\_pixter, 652
  - center\_val, 654
  - dpoints\_fwd\_pixter, 653
  - invalidate, 654
  - is\_valid, 654
  - next, 654
  - start, 654
  - update, 654
- mln::draw, 234
  - box, 234
  - box\_plain, 234
  - dashed\_line, 235
  - line, 235
  - plot, 236
  - polygon, 236
  - site\_set, 236
- mln::Edge, 655
- mln::edge\_image, 655
  - edge\_image, 657
  - edge\_nbh\_t, 656
  - edge\_win\_t, 656
  - graph\_t, 656
  - nbh\_t, 656
  - operator(), 657
  - site\_function\_t, 656
  - skeleton, 657
  - win\_t, 657
- mln::estim, 237
  - mean, 238
  - min\_max, 238
  - sum, 238, 239
- mln::extended, 657
  - domain, 659
  - extended, 658
  - skeleton, 658
  - value, 658
- mln::extension, 239
  - adjust, 240
  - adjust\_duplicate, 241
  - adjust\_fill, 241
  - duplicate, 241
  - fill, 241
  - mln::extension\_fun, 659
    - extension, 661
    - extension\_fun, 660
    - has, 661
    - operator(), 661
    - rvalue, 660
    - skeleton, 660
    - value, 660
  - mln::extension\_ima, 661
    - extension, 663
    - extension\_ima, 663
    - has, 663
    - operator(), 663
    - rvalue, 662
    - skeleton, 662
    - value, 663
  - mln::extension\_val, 664
    - change\_extension, 665
    - extension, 665
    - extension\_val, 665
    - has, 666
    - operator(), 666
    - rvalue, 665
    - skeleton, 665
    - value, 665
  - mln::faces\_psite, 666
    - change\_target, 668
    - face, 668
    - face\_id, 668
    - faces\_psite, 667
    - invalidate, 668
    - is\_valid, 668
    - n, 668
    - site\_set, 668
  - mln::flat\_image, 669
    - domain, 671
    - flat\_image, 670
    - has, 671
    - lvalue, 670
    - operator(), 671
    - rvalue, 670
    - skeleton, 670
    - value, 670
  - mln::fun, 242
  - mln::fun::access, 243
  - mln::fun::from\_accu, 671
  - mln::fun::i2v, 243
    - operator<<, 244
  - mln::fun::n2v, 244



- mln::fun::n2v::white\_gaussian, 672
- mln::fun::p2b, 244
- mln::fun::p2b::antilogy, 672
- mln::fun::p2b::tautology, 673
- mln::fun::p2p, 244
- mln::fun::p2v, 245
- mln::fun::stat, 245
- mln::fun::v2b, 245
- mln::fun::v2b::lnot, 675
- mln::fun::v2b::threshold, 676
- mln::fun::v2i, 245
- mln::fun::v2v, 246
  - f\_hsi\_to\_rgb\_3x8, 247
  - f\_hsl\_to\_rgb\_3x8, 247
  - f\_rgb\_to\_hsi\_f, 247
  - f\_rgb\_to\_hsl\_f, 247
- mln::fun::v2v::ch\_function\_value, 677
- mln::fun::v2v::component, 678
- mln::fun::v2v::l1\_norm, 678
- mln::fun::v2v::l2\_norm, 679
- mln::fun::v2v::linear, 680
- mln::fun::v2v::linfty\_norm, 681
- mln::fun::v2v::rgb8\_to\_rgbn, 682
  - operator(), 684
- mln::fun::v2w2v, 247
- mln::fun::v2w2v::cos, 684
- mln::fun::v2w\_w2v, 247
- mln::fun::v2w\_w2v::l1\_norm, 685
- mln::fun::v2w\_w2v::l2\_norm, 687
- mln::fun::v2w\_w2v::linfty\_norm, 688
- mln::fun::vv2b, 248
- mln::fun::vv2b::eq, 689
- mln::fun::vv2b::ge, 689
- mln::fun::vv2b::gt, 690
- mln::fun::vv2b::implies, 691
- mln::fun::vv2b::le, 692
- mln::fun::vv2b::lt, 693
- mln::fun::vv2v, 249
- mln::fun::vv2v::diff\_abs, 694
- mln::fun::vv2v::land, 695
- mln::fun::vv2v::land\_not, 696
- mln::fun::vv2v::lor, 697
- mln::fun::vv2v::lxor, 698
- mln::fun::vv2v::max, 699
- mln::fun::vv2v::min, 700
- mln::fun::vv2v::vec, 701
- mln::fun::x2p, 249
- mln::fun::x2p::closest\_point, 702
- mln::fun::x2v, 250
- mln::fun::x2v::bilinear, 703
  - operator(), 703
- mln::fun::x2v::trilinear, 704
- mln::fun::x2x, 250
- mln::fun::x2x::composed, 704
  - composed, 705
- mln::fun::x2x::linear, 705
  - ima, 706
  - linear, 706
  - operator(), 706
- mln::fun::x2x::rotation, 706
  - data\_t, 708
  - inv, 709
  - invert, 708
  - operator(), 709
  - rotation, 708, 709
  - set\_alpha, 709
  - set\_axis, 709
- mln::fun::x2x::translation, 710
  - data\_t, 711
  - inv, 712
  - invert, 711
  - operator(), 712
  - set\_t, 712
  - t, 712
  - translation, 711
- mln::fun\_image, 712
  - fun\_image, 714
  - lvalue, 713
  - operator(), 714
  - rvalue, 713
  - skeleton, 713
  - value, 714
- mln::Function, 715
  - Function, 715
- mln::Function< void >, 715
- mln::Function\_n2v, 716
- mln::Function\_v2b, 716
- mln::Function\_v2v, 717
- mln::Function\_vv2b, 718
- mln::Function\_vv2v, 718
- mln::fwd\_pixter1d, 719
  - fwd\_pixter1d, 720
  - image, 720
  - next, 720
- mln::fwd\_pixter2d, 720
  - fwd\_pixter2d, 721
  - image, 721
  - next, 722
- mln::fwd\_pixter3d, 722
  - fwd\_pixter3d, 723
  - image, 722
  - next, 723
- mln::Gdpoint, 723
- mln::Gdpoint< void >, 724
- mln::geom, 250
  - bbox, 255
  - chamfer, 255
  - delta, 256

- horizontal\_symmetry, 256
- max\_col, 256
- max\_ind, 256
- max\_row, 257
- max\_sli, 257
- mesh\_corner\_point\_area, 257
- mesh\_curvature, 258
- mesh\_normal, 258
- min\_col, 258
- min\_ind, 259
- min\_row, 259
- min\_sli, 259
- ncols, 259
- ninds, 260
- nrows, 260
- nsites, 260
- nslis, 260
- pmin\_pmax, 260, 261
- rotate, 261, 262
- seeds2tiling, 262
- seeds2tiling\_roundness, 263
- translate, 263, 264
- vertical\_symmetry, 264
- mln::geom::complex\_geometry, 725
  - add\_location, 726
  - complex\_geometry, 725
  - operator(), 726
- mln::geom::impl, 264
  - seeds2tiling, 265
- mln::Gpoint, 726
  - operator<<, 730
  - operator+, 728
  - operator+=", 728
  - operator-, 729
  - operator=, 729
  - operator/, 730
  - operator==, 730
- mln::Graph, 731
- mln::graph, 265
  - compute, 266
  - labeling, 266
  - to\_neighb, 266
  - to\_win, 267
- mln::graph::attribute::card\_t, 732
  - result, 732
- mln::graph::attribute::representative\_t, 732
  - result, 733
- mln::graph\_elt\_mixed\_neighborhood, 733
  - bkd\_niter, 734
  - fwd\_niter, 734
  - niter, 734
- mln::graph\_elt\_mixed\_window, 734
  - bkd\_qiter, 736
  - center\_t, 736
  - delta, 737
  - fwd\_qiter, 736
  - graph\_element, 737
  - is\_centered, 737
  - is\_empty, 738
  - is\_symmetric, 738
  - is\_valid, 738
  - psite, 737
  - qiter, 737
  - site, 737
  - sym, 738
  - target, 737
- mln::graph\_elt\_neighborhood, 738
  - bkd\_niter, 740
  - fwd\_niter, 740
  - niter, 740
- mln::graph\_elt\_neighborhood\_if, 740
  - bkd\_niter, 741
  - fwd\_niter, 741
  - graph\_elt\_neighborhood\_if, 742
  - mask, 742
  - niter, 741
- mln::graph\_elt\_window, 742
  - bkd\_qiter, 744
  - center\_t, 744
  - delta, 745
  - fwd\_qiter, 744
  - graph\_element, 745
  - is\_centered, 745
  - is\_empty, 746
  - is\_symmetric, 746
  - is\_valid, 746
  - psite, 745
  - qiter, 745
  - site, 745
  - sym, 746
  - target, 745
- mln::graph\_elt\_window\_if, 746
  - bkd\_qiter, 748
  - change\_mask, 750
  - delta, 750
  - fwd\_qiter, 748
  - graph\_elt\_window\_if, 750
  - is\_centered, 750
  - is\_empty, 750
  - is\_symmetric, 750
  - is\_valid, 751
  - mask, 751
  - mask\_t, 749
  - psite, 749
  - qiter, 749
  - site, 749
  - sym, 751
  - target, 749

- mln::graph\_window\_base, 751
  - delta, 752
  - is\_centered, 752
  - is\_empty, 752
  - is\_symmetric, 753
  - is\_valid, 753
  - site, 752
  - sym, 753
- mln::graph\_window\_if\_piter, 753
  - element, 754
  - graph\_window\_if\_piter, 754
  - id, 754
  - next, 755
  - P, 754
- mln::graph\_window\_piter, 755
  - center\_t, 756
  - change\_target\_site\_set, 757
  - element, 758
  - graph\_element, 756
  - graph\_window\_piter, 757
  - id, 758
  - next, 758
  - P, 756
  - target\_site\_set, 758
- mln::grid, 267
- mln::hexa, 758
  - bkd\_piter, 760
  - domain, 761
  - fwd\_piter, 760
  - has, 761
  - hexa, 761
  - lvalue, 760
  - operator(), 762
  - psite, 760
  - rvalue, 761
  - skeleton, 761
  - value, 761
- mln::histo, 267
  - compute, 268
  - equalize, 268
- mln::histo::array, 762
- mln::histo::impl, 269
- mln::histo::impl::generic, 269
- mln::Image, 762
- mln::image1d, 764
  - bbox, 767
  - border, 767
  - buffer, 767
  - delta\_index, 767
  - domain, 767
  - element, 767
  - has, 768
  - image1d, 766
  - lvalue, 766
  - nelements, 768
  - ninds, 768
  - operator(), 768
  - point\_at\_index, 768
  - rvalue, 766
  - skeleton, 766
  - value, 766
  - vbbox, 769
- mln::image2d, 769
  - bbox, 772
  - border, 772
  - buffer, 772
  - delta\_index, 772
  - domain, 772
  - element, 772
  - has, 773
  - image2d, 771
  - lvalue, 771
  - ncols, 773
  - nelements, 773
  - nrows, 773
  - operator(), 773
  - point\_at\_index, 773
  - rvalue, 771
  - skeleton, 771
  - value, 771
- mln::image2d\_h, 774
  - bkd\_piter, 775
  - domain, 776
  - fwd\_piter, 775
  - has, 776
  - image2d\_h, 776
  - lvalue, 775
  - operator(), 777
  - psite, 776
  - rvalue, 776
  - skeleton, 776
  - value, 776
- mln::image3d, 777
  - bbox, 780
  - border, 780
  - buffer, 780
  - delta\_index, 780
  - domain, 781
  - element, 781
  - has, 781
  - image3d, 780
  - lvalue, 779
  - ncols, 781
  - nelements, 781
  - nrows, 782
  - nslis, 782
  - operator(), 782
  - point\_at\_index, 782

- rvalue, 779
- skeleton, 779
- value, 779
- vbbox, 782
- mln::image\_if, 782
  - domain, 784
  - image\_if, 783
  - operator image\_if< const I, F >, 784
  - skeleton, 783
- mln::impl, 269
- mln::interpolated, 784
  - has, 786
  - interpolated, 786
  - is\_valid, 786
  - lvalue, 785
  - psite, 785
  - rvalue, 785
  - skeleton, 785
  - value, 785
- mln::io, 269
- mln::io::cloud, 271
  - load, 271
  - save, 271
- mln::io::dicom, 272
  - get\_header, 272
  - load, 272
- mln::io::dicom::dicom\_header, 786
- mln::io::dump, 273
  - get\_header, 273
  - load, 273
  - save, 274
- mln::io::dump::dump\_header, 786
- mln::io::fits, 274
  - load, 274, 275
- mln::io::fld, 275
  - load, 276
  - read\_header, 276
  - write\_header, 276
- mln::io::fld::fld\_header, 787
- mln::io::magick, 276
  - load, 277
  - save, 277
- mln::io::off, 278
  - load, 278
  - save, 279
  - save\_bin\_alt, 279
- mln::io::pbm, 279
  - load, 280
  - save, 280
- mln::io::pbm::impl, 281
- mln::io::pbms, 281
  - load, 281
- mln::io::pbms::impl, 282
- mln::io::pfm, 282
  - load, 283
  - save, 283
- mln::io::pfm::impl, 283
- mln::io::pgm, 284
  - load, 284
  - save, 284
- mln::io::pgms, 285
  - load, 285
- mln::io::plot, 285
  - load, 286
  - save, 286, 287
- mln::io::pnm, 287
  - load, 288
  - load\_ascii\_builtin, 288
  - load\_ascii\_value, 288
  - load\_raw\_2d, 288
  - max\_component, 289
  - save, 289
- mln::io::pnm::impl, 289
- mln::io::pnms, 289
  - load, 290
- mln::io::ppm, 290
  - load, 291
  - save, 291
- mln::io::ppms, 292
  - load, 292
- mln::io::raw, 292
  - get\_header, 293
  - load, 293
  - save, 293
- mln::io::raw::raw\_header, 787
- mln::io::tiff, 294
  - load, 294
- mln::io::txt, 294
  - save, 295
- mln::Iterator, 787
  - next, 789
- mln::labeled\_image, 789
  - bbbox, 792
  - bbbox\_t, 791
  - bboxes, 792
  - labeled\_image, 791
  - nlabels, 792
  - relabel, 792
  - skeleton, 791
  - subdomain, 792
  - update\_data, 792
- mln::labeled\_image\_base, 793
  - bbbox, 795
  - bbbox\_t, 794
  - bboxes, 795
  - labeled\_image\_base, 794
  - nlabels, 795
  - relabel, 795

- subdomain, 795
- update\_data, 796
- mln::labeling, 295
  - background, 298
  - blobs, 299
  - blobs\_and\_compute, 299
  - colorize, 299, 300
  - compute, 300–302
  - compute\_image, 302, 303
  - fill\_holes, 304
  - flat\_zones, 304
  - foreground, 304
  - pack, 305
  - pack\_inplace, 306
  - regional\_maxima, 306
  - regional\_minima, 306
  - relabel, 307
  - relabel\_inplace, 308
  - superpose, 308
  - value, 309
  - value\_and\_compute, 309
  - wrap, 310
- mln::labeling::impl, 310
  - compute\_fastest, 311
- mln::labeling::impl::generic, 312
  - compute, 313, 314
- mln::lazy\_image, 796
  - domain, 798
  - has, 798
  - lazy\_image, 798
  - lvalue, 797
  - operator(), 798
  - rvalue, 797
  - skeleton, 797
- mln::linear, 314
  - gaussian, 315
  - gaussian\_1st\_derivative, 316
  - gaussian\_2nd\_derivative, 316
  - mln\_ch\_convolve, 317
  - mln\_ch\_convolve\_grad, 318
- mln::linear::impl, 318
- mln::linear::local, 318
  - convolve, 319
- mln::linear::local::impl, 319
- mln::Literal, 799
- mln::literal, 319
  - black, 323
  - blue, 323
  - brown, 323
  - cyan, 323
  - dark\_gray, 323
  - green, 323
  - identity, 323
  - light\_gray, 323
  - lime, 323
  - magenta, 324
  - max, 324
  - medium\_gray, 324
  - min, 324
  - olive, 324
  - one, 324
  - orange, 324
  - origin, 324
  - pink, 325
  - purple, 325
  - red, 325
  - teal, 325
  - violet, 325
  - white, 325
  - yellow, 325
  - zero, 325
- mln::literal::black\_t, 801
- mln::literal::blue\_t, 802
- mln::literal::brown\_t, 802
- mln::literal::cyan\_t, 803
- mln::literal::green\_t, 804
- mln::literal::identity\_t, 805
- mln::literal::light\_gray\_t, 806
- mln::literal::lime\_t, 807
- mln::literal::magenta\_t, 808
- mln::literal::max\_t, 809
- mln::literal::min\_t, 810
- mln::literal::olive\_t, 811
- mln::literal::one\_t, 812
- mln::literal::orange\_t, 813
- mln::literal::origin\_t, 814
- mln::literal::pink\_t, 815
- mln::literal::purple\_t, 816
- mln::literal::red\_t, 817
- mln::literal::teal\_t, 818
- mln::literal::violet\_t, 819
- mln::literal::white\_t, 820
- mln::literal::yellow\_t, 821
- mln::literal::zero\_t, 822
- mln::logical, 326
  - and\_inplace, 327
  - and\_not, 327
  - and\_not\_inplace, 327
  - not\_inplace, 328
  - or\_inplace, 328
  - xor\_inplace, 329
- mln::logical::impl, 329
- mln::logical::impl::generic, 329
- mln::make, 330
  - attachment, 335
  - box1d, 335, 336
  - box2d, 336
  - box2d\_h, 337

- box3d, 338
- cell, 338
- couple, 339
- detachment, 339
- dpoint2d\_h, 339
- dummy\_p\_edges, 340
- dummy\_p\_vertices, 340, 341
- edge\_image, 341, 342
- h\_mat, 343
- image, 343, 344
- image2d, 344
- image3d, 344
- influence\_zone\_adjacency\_graph, 344
- mat, 345
- ord\_pair, 345
- p\_edges\_with\_mass\_centers, 345
- p\_vertices\_with\_mass\_centers, 346
- pix, 346
- pixel, 346
- point2d\_h, 347
- rag\_and\_labeled\_wsl, 347
- region\_adjacency\_graph, 348
- relabelfun, 348
- vec, 349, 350
- vertex\_image, 350
- voronoi, 351
- w\_window, 351
- w\_window1d, 351
- w\_window1d\_int, 352
- w\_window2d, 352
- w\_window2d\_int, 353
- w\_window3d, 353
- w\_window3d\_int, 353
- w\_window\_directional, 354
- mln::math, 354
  - abs, 355
- mln::Mesh, 823
- mln::Meta\_Accumulator, 824
- mln::Meta\_Function, 826
- mln::Meta\_Function\_v2v, 827
- mln::Meta\_Function\_vv2v, 827
- mln::metal, 355
- mln::metal::ands, 828
- mln::metal::converts\_to, 828
- mln::metal::equal, 828
- mln::metal::goes\_to, 829
- mln::metal::impl, 356
- mln::metal::is, 829
- mln::metal::is\_a, 830
- mln::metal::is\_not, 830
- mln::metal::is\_not\_a, 830
- mln::metal::math, 356
- mln::metal::math::impl, 357
- mln::mixed\_neighb, 830
- bkd\_niter, 831
- fwd\_niter, 831
- mixed\_neighb, 832
- niter, 831
- mln::morpho, 357
  - complementation, 360
  - complementation\_inplace, 360
  - contrast, 360
  - dilation, 360
  - erosion, 361
  - general, 361
  - gradient, 361
  - gradient\_external, 361
  - gradient\_internal, 361
  - hit\_or\_miss, 362
  - hit\_or\_miss\_background\_closing, 362
  - hit\_or\_miss\_background\_opening, 362
  - hit\_or\_miss\_closing, 362
  - hit\_or\_miss\_opening, 362
  - laplacian, 363
  - line\_gradient, 363
  - meyer\_wst, 363
  - min, 364
  - min\_inplace, 364
  - minus, 364
  - plus, 364
  - rank\_filter, 365
  - thick\_miss, 365
  - thickening, 365
  - thin\_fit, 365
  - thinning, 365
  - top\_hat\_black, 366
  - top\_hat\_self\_complementary, 366
  - top\_hat\_white, 366
- mln::morpho::approx, 367
- mln::morpho::attribute, 367
- mln::morpho::attribute::card, 832
  - init, 833
  - is\_valid, 833
  - take\_as\_init, 833
  - take\_n\_times, 833
  - to\_result, 833
- mln::morpho::attribute::count\_adjacent\_vertices, 833
  - init, 834
  - is\_valid, 834
  - take\_as\_init, 834
  - take\_n\_times, 834
  - to\_result, 834
- mln::morpho::attribute::height, 835
  - base\_level, 836
  - init, 836
  - is\_valid, 836
  - take\_as\_init, 836

- take\_n\_times, 836
- to\_result, 836
- mln::morpho::attribute::sharpness, 836
  - area, 837
  - height, 837
  - init, 838
  - is\_valid, 838
  - take\_as\_init, 838
  - take\_n\_times, 838
  - to\_result, 838
  - volume, 838
- mln::morpho::attribute::sum, 839
  - init, 839
  - is\_valid, 839
  - set\_value, 840
  - take\_as\_init, 840
  - take\_n\_times, 840
  - to\_result, 840
  - untake, 840
- mln::morpho::attribute::volume, 840
  - area, 841
  - init, 841
  - is\_valid, 841
  - take\_as\_init, 842
  - take\_n\_times, 842
  - to\_result, 842
- mln::morpho::closing::approx, 367
  - structural, 368
- mln::morpho::elementary, 368
  - closing, 369
    - mln\_trait\_op\_minus\_twice, 369
  - opening, 369
    - top\_hat\_black, 369
    - top\_hat\_white, 369
- mln::morpho::impl, 370
- mln::morpho::impl::generic, 370
- mln::morpho::opening::approx, 370
  - structural, 371
- mln::morpho::reconstruction, 371
- mln::morpho::reconstruction::by\_dilation, 371
- mln::morpho::reconstruction::by\_erosion, 371
- mln::morpho::tree, 372
  - compute\_attribute\_image, 373
  - compute\_attribute\_image\_from, 374
  - compute\_parent, 374
  - dual\_input\_max\_tree, 375
  - max\_tree, 376
  - min\_tree, 376
  - propagate\_if, 376, 377
    - propagate\_if\_value, 377
  - propagate\_node\_to\_ancestors, 378
  - propagate\_node\_to\_descendants, 378, 379
  - propagate\_representative, 379
- mln::morpho::tree::filter, 379
  - direct, 380
  - filter, 380
  - max, 380
  - min, 381
  - subtractive, 381
- mln::morpho::watershed, 382
  - flooding, 382, 383
  - superpose, 383
  - topological, 383
- mln::morpho::watershed::watershed, 384
- mln::morpho::watershed::watershed::generic, 384
- mln::neighb, 842
  - bkd\_niter, 843
  - fwd\_niter, 843
  - neighb, 843
  - niter, 843
- mln::Neighborhood, 844
- mln::Neighborhood< void >, 844
- mln::norm, 384
  - l1, 385
  - l1\_distance, 385
  - l2, 386
  - l2\_distance, 386
  - linfty, 386
  - linfty\_distance, 386
  - sqr\_l2, 386
- mln::norm::impl, 386
- mln::Object, 844
- mln::opt, 387
  - at, 388
- mln::opt::impl, 388
- mln::p2p\_image, 845
  - domain, 846
  - fun, 846
  - operator(), 846, 847
  - p2p\_image, 846
  - skeleton, 846
- mln::p\_array, 847
  - append, 850
  - bkd\_piter, 849
  - change, 850
  - clear, 850
  - element, 849
  - fwd\_piter, 849
  - has, 850, 851
  - i\_element, 849
  - insert, 851
  - is\_valid, 851
  - memory\_size, 851
  - nsites, 851
  - p\_array, 850
  - piter, 849
  - psite, 849
  - reserve, 852

- resize, 852
- std\_vector, 852
- mln::p\_centered, 852
  - bkd\_piter, 854
  - center, 855
  - element, 854
  - fwd\_piter, 854
  - has, 855
  - is\_valid, 855
  - memory\_size, 855
  - p\_centered, 854
  - piter, 854
  - psite, 854
  - site, 854
  - window, 855
- mln::p\_complex, 855
  - bkd\_piter, 857
  - cplx, 858
  - element, 857
  - fwd\_piter, 857
  - geom, 858
  - has, 858
  - is\_valid, 859
  - nfaces, 859
  - nfaces\_of\_dim, 859
  - nsites, 859
  - p\_complex, 858
  - piter, 857
  - psite, 857
- mln::p\_edges, 859
  - bkd\_piter, 861
  - edge, 861
  - element, 861
  - fun\_t, 862
  - function, 864
  - fwd\_piter, 862
  - graph, 864
  - graph\_element, 862
  - graph\_t, 862
  - has, 864
  - invalidate, 864
  - is\_valid, 864
  - memory\_size, 865
  - nedges, 865
  - nsites, 865
  - p\_edges, 863
  - piter, 862
  - psite, 862
- mln::p\_faces, 865
  - bkd\_piter, 866
  - cplx, 867, 868
  - element, 866
  - fwd\_piter, 866
  - is\_valid, 868
  - nfaces, 868
  - nsites, 868
  - p\_faces, 867
  - piter, 867
  - psite, 867
- mln::p\_graph\_piter, 868
  - graph, 869
  - id, 869
  - mln\_q\_subject, 869
  - next, 870
  - p\_graph\_piter, 869
- mln::p\_if, 870
  - bkd\_piter, 871
  - element, 871
  - fwd\_piter, 871
  - has, 872
  - is\_valid, 872
  - memory\_size, 872
  - overset, 872
  - p\_if, 872
  - piter, 871
  - pred, 873
  - predicate, 873
  - psite, 872
- mln::p\_image, 873
  - bkd\_piter, 875
  - clear, 876
  - element, 875
  - fwd\_piter, 875
  - has, 876
  - i\_element, 875
  - insert, 876
  - is\_valid, 876
  - memory\_size, 876
  - nsites, 877
  - operator typename internal::p\_image\_site\_-  
set< I >::ret, 877
  - p\_image, 876
  - piter, 875
  - psite, 875
  - r\_element, 875
  - remove, 877
  - S, 875
  - toggle, 877
- mln::p\_indexed\_bkd\_piter, 877
  - index, 878
  - next, 878
  - p\_indexed\_bkd\_piter, 878
- mln::p\_indexed\_fwd\_piter, 879
  - index, 880
  - next, 880
  - p\_indexed\_fwd\_piter, 879
- mln::p\_indexed\_psite, 880
- mln::p\_key, 880



- bkd\_piter, 882
- change\_key, 883
- change\_keys, 883
- clear, 884
- element, 882
- exists\_key, 884
- fwd\_piter, 882
- has, 884
- i\_element, 883
- insert, 884
- is\_valid, 885
- key, 885
- keys, 885
- memory\_size, 885
- nsites, 885
- operator(), 885
- p\_key, 883
- piter, 883
- psite, 883
- r\_element, 883
- remove, 885
- remove\_key, 885
- mln::p\_line2d, 886
  - bbox, 888
  - begin, 888
  - bkd\_piter, 887
  - element, 887
  - end, 889
  - fwd\_piter, 887
  - has, 889
  - is\_valid, 889
  - memory\_size, 889
  - nsites, 889
  - p\_line2d, 888
  - piter, 887
  - psite, 888
  - q\_box, 888
  - std\_vector, 890
- mln::p\_mutable\_array\_of, 890
  - bkd\_piter, 891
  - clear, 892
  - element, 891
  - fwd\_piter, 891
  - has, 892
  - i\_element, 891
  - insert, 892
  - is\_valid, 892
  - memory\_size, 893
  - nelements, 893
  - p\_mutable\_array\_of, 892
  - piter, 892
  - psite, 892
  - reserve, 893
- mln::p\_n\_faces\_bkd\_piter, 893
  - n, 894
  - next, 894
  - p\_n\_faces\_bkd\_piter, 894
- mln::p\_n\_faces\_fwd\_piter, 895
  - n, 895
  - next, 895
  - p\_n\_faces\_fwd\_piter, 895
- mln::p\_priority, 896
  - bkd\_piter, 898
  - clear, 899
  - element, 898
  - exists\_priority, 899
  - front, 899
  - fwd\_piter, 898
  - has, 899
  - highest\_priority, 899
  - i\_element, 898
  - insert, 900
  - is\_valid, 900
  - lowest\_priority, 900
  - memory\_size, 900
  - nsites, 900
  - operator(), 901
  - p\_priority, 899
  - piter, 898
  - pop, 901
  - pop\_front, 901
  - priorities, 901
  - psite, 898
  - push, 901
- mln::p\_queue, 902
  - bkd\_piter, 903
  - clear, 904
  - element, 903
  - front, 904
  - fwd\_piter, 903
  - has, 904, 905
  - i\_element, 904
  - insert, 905
  - is\_valid, 905
  - memory\_size, 905
  - nsites, 905
  - p\_queue, 904
  - piter, 904
  - pop, 905
  - pop\_front, 906
  - psite, 904
  - push, 906
  - std\_deque, 906
- mln::p\_queue\_fast, 906
  - bkd\_piter, 908
  - clear, 909
  - compute\_has, 909
  - element, 908

- empty, 909
- front, 909
- fwd\_piter, 908
- has, 910
- i\_element, 908
- insert, 910
- is\_valid, 910
- memory\_size, 910
- nsites, 910
- p\_queue\_fast, 909
- piter, 909
- pop, 911
- pop\_front, 911
- psite, 909
- purge, 911
- push, 911
- reserve, 911
- std\_vector, 911
- mln::p\_run, 912
  - bbox, 915
  - bkd\_piter, 913
  - element, 913
  - end, 915
  - fwd\_piter, 913
  - has, 915
  - has\_index, 915
  - init, 915
  - is\_valid, 915
  - length, 916
  - memory\_size, 916
  - nsites, 916
  - p\_run, 914
  - piter, 914
  - psite, 914
  - q\_box, 914
  - start, 916
- mln::p\_set, 916
  - bkd\_piter, 918
  - clear, 919
  - element, 918
  - fwd\_piter, 918
  - has, 919
  - i\_element, 918
  - insert, 920
  - is\_valid, 920
  - memory\_size, 920
  - nsites, 920
  - p\_set, 919
  - piter, 918
  - psite, 919
  - r\_element, 919
  - remove, 920
  - std\_vector, 920
  - util\_set, 920
- mln::p\_set\_of, 921
  - bkd\_piter, 922
  - clear, 923
  - element, 922
  - fwd\_piter, 922
  - has, 923
  - i\_element, 922
  - insert, 923
  - is\_valid, 923
  - memory\_size, 923
  - nelements, 924
  - p\_set\_of, 923
  - piter, 922
  - psite, 923
- mln::p\_transformed, 924
  - bkd\_piter, 925
  - element, 925
  - function, 926
  - fwd\_piter, 925
  - has, 926
  - is\_valid, 926
  - memory\_size, 926
  - p\_transformed, 926
  - piter, 925
  - primary\_set, 927
  - psite, 926
- mln::p\_transformed\_piter, 927
  - change\_target, 928
  - next, 928
  - p\_transformed\_piter, 928
- mln::p\_vaccess, 928
  - bkd\_piter, 930
  - element, 930
  - fwd\_piter, 930
  - has, 931
  - i\_element, 930
  - insert, 931, 932
  - is\_valid, 932
  - memory\_size, 932
  - operator(), 932
  - p\_vaccess, 931
  - piter, 930
  - pset, 931
  - psite, 931
  - value, 931
  - values, 932
  - vset, 931
- mln::p\_vertices, 932
  - bkd\_piter, 934
  - element, 934
  - fun\_t, 935
  - function, 937
  - fwd\_piter, 935
  - graph, 937

- graph\_element, 935
- graph\_t, 935
- has, 937
- invalidate, 937
- is\_valid, 938
- memory\_size, 938
- nsites, 938
- nvertices, 938
- operator(), 938
- p\_vertices, 936
- piter, 935
- psite, 935
- vertex, 935
- mln::pixel, 939
  - change\_to, 940
  - is\_valid, 940
  - pixel, 940
- mln::Pixel\_Iterator, 940
  - next, 941
- mln::plain, 942
  - operator I, 943
  - operator=, 943
  - plain, 943
  - skeleton, 942
- mln::Point, 943
  - operator+=, 945
  - operator-=, 946
  - operator/, 946
  - point, 945
  - to\_point, 945
- mln::point, 946
  - coord, 949
  - delta, 949
  - dim, 950
  - dpsite, 949
  - grid, 949
  - h\_vec, 949
  - last\_coord, 951
  - minus\_infty, 951
  - operator+=, 951
  - operator-=, 951
  - origin, 953
  - plus\_infty, 952
  - point, 950, 951
  - set\_all, 952
  - to\_h\_vec, 952
  - to\_vec, 952
  - vec, 950
- mln::Point\_Site, 953
  - operator<<, 956
  - operator+, 954
  - operator-, 955
  - operator==, 956
- mln::Point\_Site< void >, 956
- mln::Proxy, 957
- mln::Proxy< void >, 957
- mln::Pseudo\_Site, 957
- mln::Pseudo\_Site< void >, 958
- mln::pw, 389
- mln::pw::image, 958
  - image, 959
  - skeleton, 959
- mln::registration, 389
  - get\_rot, 390
  - icp, 390, 391
  - registration1, 391
  - registration2, 391
  - registration3, 391
- mln::registration::closest\_point\_basic, 959
- mln::registration::closest\_point\_with\_map, 960
- mln::Regular\_Grid, 960
- mln::safe\_image, 961
  - operator safe\_image< const I >, 961
  - skeleton, 961
- mln::select, 392
- mln::select::p\_of, 961
- mln::set, 392
  - card, 393
  - compute, 393
  - compute\_with\_weights, 393
  - get, 394
  - has, 394
  - mln\_meta\_accu\_result, 394
- mln::Site, 962
- mln::Site< void >, 963
- mln::Site\_Iterator, 963
  - next, 965
- mln::Site\_Proxy, 965
- mln::Site\_Proxy< void >, 966
- mln::Site\_Set, 966
  - diff, 968
  - inter, 968
  - operator<, 968
  - operator<<, 968
  - operator<=, 968
  - operator==, 968
  - sym\_diff, 969
  - uni, 969
  - unique, 969
- mln::Site\_Set< void >, 969
- mln::slice\_image, 969
  - domain, 971
  - operator slice\_image< const I >, 971
  - operator(), 971
  - skeleton, 970
  - sli, 971
  - slice\_image, 971
- mln::sub\_image, 972

- domain, 973
- operator sub\_image< const I, S >, 973
- skeleton, 972
- sub\_image, 973
- mln::sub\_image\_if, 973
  - domain, 974
  - skeleton, 974
  - sub\_image\_if, 974
- mln::subsampling, 394
  - antialiased, 395
  - gaussian\_subsampling, 395
  - subsampling, 396
- mln::tag, 396
- mln::test, 396
  - positive, 397
  - predicate, 397, 398
- mln::test::impl, 398
- mln::thru\_image, 975
  - operator thru\_image< const I, F >, 975
- mln::thru\_image, 975
  - operator thru\_image< const I1, const I2, F >, 976
  - psite, 976
  - rvalue, 976
  - skeleton, 976
  - value, 976
- mln::topo, 398
  - detach, 403
  - edge, 403
  - is\_facet, 403
  - make\_algebraic\_face, 403
  - operator<, 405, 406
  - operator<<, 406, 407
  - operator+, 404
  - operator-, 405
  - operator==, 407, 408
- mln::topo::adj\_higher\_dim\_connected\_n\_face\_-  
bkd\_iter, 977
  - adj\_higher\_dim\_connected\_n\_face\_bkd\_iter, 977
  - next, 978
- mln::topo::adj\_higher\_dim\_connected\_n\_face\_-  
fwd\_iter, 978
  - adj\_higher\_dim\_connected\_n\_face\_fwd\_iter, 979
  - next, 979
- mln::topo::adj\_higher\_face\_bkd\_iter, 979
  - adj\_higher\_face\_bkd\_iter, 980
  - next, 980
- mln::topo::adj\_higher\_face\_fwd\_iter, 980
  - adj\_higher\_face\_fwd\_iter, 981
  - next, 981
- mln::topo::adj\_lower\_dim\_connected\_n\_face\_-  
bkd\_iter, 981
  - adj\_lower\_dim\_connected\_n\_face\_bkd\_iter, 982
  - next, 982
- mln::topo::adj\_lower\_dim\_connected\_n\_face\_-  
fwd\_iter, 982
  - adj\_lower\_dim\_connected\_n\_face\_fwd\_iter, 983
  - next, 983
- mln::topo::adj\_lower\_face\_bkd\_iter, 984
  - adj\_lower\_face\_bkd\_iter, 984
  - next, 984
- mln::topo::adj\_lower\_face\_fwd\_iter, 985
  - adj\_lower\_face\_fwd\_iter, 985
  - next, 985
- mln::topo::adj\_lower\_higher\_face\_bkd\_iter, 986
  - adj\_lower\_higher\_face\_bkd\_iter, 986
  - next, 987
- mln::topo::adj\_lower\_higher\_face\_fwd\_iter, 987
  - adj\_lower\_higher\_face\_fwd\_iter, 988
  - next, 988
- mln::topo::adj\_m\_face\_bkd\_iter, 988
  - adj\_m\_face\_bkd\_iter, 989
  - next, 989
- mln::topo::adj\_m\_face\_fwd\_iter, 989
  - adj\_m\_face\_fwd\_iter, 990
  - next, 991
- mln::topo::algebraic\_face, 991
  - algebraic\_face, 993
  - cplx, 993
  - data, 993
  - dec\_face\_id, 994
  - dec\_n, 994
  - face\_id, 994
  - higher\_dim\_adj\_faces, 994
  - inc\_face\_id, 994
  - inc\_n, 994
  - invalidate, 994
  - is\_valid, 995
  - lower\_dim\_adj\_faces, 995
  - n, 995
  - set\_cplx, 995
  - set\_face\_id, 995
  - set\_n, 995
  - set\_sign, 996
  - sign, 996
- mln::topo::algebraic\_n\_face, 996
  - algebraic\_n\_face, 998
  - cplx, 998
  - data, 998
  - dec\_face\_id, 999
  - face\_id, 999
  - higher\_dim\_adj\_faces, 999
  - inc\_face\_id, 999
  - invalidate, 999

- is\_valid, 999
- lower\_dim\_adj\_faces, 1000
- n, 1000
- set\_cplx, 1000
- set\_face\_id, 1000
- set\_sign, 1000
- sign, 1000
- mln::topo::center\_only\_iter, 1001
  - center\_only\_iter, 1001
  - next, 1002
- mln::topo::centered\_bkd\_iter\_adapter, 1002
  - centered\_bkd\_iter\_adapter, 1003
  - next, 1003
- mln::topo::centered\_fwd\_iter\_adapter, 1003
  - centered\_fwd\_iter\_adapter, 1004
  - next, 1004
- mln::topo::complex, 1004
  - add\_face, 1006
  - addr, 1006
  - bkd\_citer, 1005
  - complex, 1006
  - fwd\_citer, 1005
  - nfaces, 1006
  - nfaces\_of\_dim, 1007
  - nfaces\_of\_static\_dim, 1007
  - print, 1007
  - print\_faces, 1007
- mln::topo::face, 1007
  - cplx, 1010
  - data, 1010
  - dec\_face\_id, 1010
  - dec\_n, 1010
  - face, 1009
  - face\_id, 1010
  - higher\_dim\_adj\_faces, 1010
  - inc\_face\_id, 1010
  - inc\_n, 1011
  - invalidate, 1011
  - is\_valid, 1011
  - lower\_dim\_adj\_faces, 1011
  - n, 1011
  - set\_cplx, 1011
  - set\_face\_id, 1011
  - set\_n, 1012
- mln::topo::face\_bkd\_iter, 1012
  - face\_bkd\_iter, 1013
  - next, 1013
  - start, 1013
- mln::topo::face\_fwd\_iter, 1013
  - face\_fwd\_iter, 1014
  - next, 1014
  - start, 1014
- mln::topo::is\_n\_face, 1014
- mln::topo::is\_simple\_2d\_t, 1015
- mln::topo::is\_simple\_cell, 1016
  - D, 1019
  - mln\_geom, 1018
  - operator(), 1018
  - psite, 1018
  - result, 1018
  - set\_image, 1019
- mln::topo::n\_face, 1019
  - cplx, 1021
  - data, 1021
  - dec\_face\_id, 1021
  - face\_id, 1021
  - higher\_dim\_adj\_faces, 1021
  - inc\_face\_id, 1021
  - invalidate, 1022
  - is\_valid, 1022
  - lower\_dim\_adj\_faces, 1022
  - n, 1022
  - n\_face, 1020
  - set\_cplx, 1022
  - set\_face\_id, 1022
- mln::topo::n\_face\_bkd\_iter, 1023
  - n, 1024
  - n\_face\_bkd\_iter, 1023
  - next, 1024
  - start, 1024
- mln::topo::n\_face\_fwd\_iter, 1024
  - n, 1025
  - n\_face\_fwd\_iter, 1025
  - next, 1025
  - start, 1025
- mln::topo::n\_faces\_set, 1026
  - add, 1027
  - faces, 1027
  - faces\_type, 1026
  - reserve, 1027
- mln::topo::skeleton::is\_simple\_point, 1027
- mln::topo::static\_n\_face\_bkd\_iter, 1028
  - next, 1029
  - start, 1029
  - static\_n\_face\_bkd\_iter, 1028
- mln::topo::static\_n\_face\_fwd\_iter, 1029
  - next, 1030
  - start, 1030
  - static\_n\_face\_fwd\_iter, 1030
- mln::tr\_image, 1030
  - domain, 1033
  - has, 1033
  - is\_valid, 1033
  - lvalue, 1032
  - operator(), 1033
  - psite, 1032
  - rvalue, 1032
  - set\_tr, 1033

- site, 1032
- skeleton, 1032
- tr, 1033
- tr\_image, 1032
- value, 1032
- mln::trace, 408
- mln::trait, 408
- mln::transform, 408
  - distance\_and\_closest\_point\_geodesic, 410
  - distance\_and\_influence\_zone\_geodesic, 411
  - distance\_front, 411
  - distance\_geodesic, 411
  - hough, 411
  - influence\_zone\_front, 412
  - influence\_zone\_geodesic, 412
  - influence\_zone\_geodesic\_saturated, 413
- mln::unproject\_image, 1033
  - domain, 1034
  - operator(), 1034, 1035
  - unproject\_image, 1034
- mln::util, 413
  - display\_branch, 417
  - display\_tree, 417
  - lemmings, 417
  - make\_greater\_point, 418
  - make\_greater\_psite, 418
  - operator<, 418
  - operator<<, 418
  - operator==, 418, 419
  - ord\_strict, 419
  - ord\_weak, 419
  - tree\_fast\_to\_image, 419
  - tree\_to\_fast, 419
  - tree\_to\_image, 420
  - vertex\_id\_t, 417
- mln::util::adjacency\_matrix, 1035
  - adjacency\_matrix, 1035, 1036
- mln::util::array, 1036
  - append, 1039, 1040
  - array, 1039
  - bkd\_eiter, 1038
  - clear, 1040
  - eiter, 1038
  - element, 1038
  - fill, 1040
  - fwd\_eiter, 1039
  - is\_empty, 1040
  - last, 1040
  - memory\_size, 1041
  - nelements, 1041
  - operator(), 1041
  - reserve, 1042
  - resize, 1042
  - result, 1039
  - size, 1042
  - std\_vector, 1043
- mln::util::branch, 1043
  - apex, 1044
  - branch, 1043
  - util\_tree, 1044
- mln::util::branch\_iter, 1044
  - deepness, 1045
  - invalidate, 1045
  - is\_valid, 1045
  - next, 1045
  - operator util::tree\_node< T > &, 1045
  - start, 1046
- mln::util::branch\_iter\_ind, 1046
  - deepness, 1047
  - invalidate, 1047
  - is\_valid, 1047
  - next, 1047
  - operator util::tree\_node< T > &, 1047
  - start, 1047
- mln::util::couple, 1047
  - change\_both, 1049
  - change\_first, 1049
  - change\_second, 1049
  - first, 1049
  - second, 1049
- mln::util::eat, 1049
- mln::util::edge, 1050
  - category, 1052
  - change\_graph, 1052
  - edge, 1052
  - graph, 1052
  - graph\_t, 1052
  - id, 1052
  - id\_t, 1052
  - id\_value\_t, 1052
  - invalidate, 1053
  - is\_valid, 1053
  - ith\_nbh\_edge, 1053
  - nmax\_nbh\_edges, 1053
  - operator edge\_id\_t, 1053
  - update\_id, 1053
  - v1, 1053
  - v2, 1054
  - v\_other, 1054
- mln::util::fibonacci\_heap, 1054
  - clear, 1056
  - fibonacci\_heap, 1056
  - front, 1056
  - is\_empty, 1056
  - is\_valid, 1057
  - nelements, 1057
  - operator=, 1057
  - pop\_front, 1057

- push, 1057, 1058
- mln::util::graph, 1058
  - add\_edge, 1062
  - add\_vertex, 1062
  - add\_vertices, 1062
  - e\_ith\_nbh\_edge, 1062
  - e\_nmax, 1062
  - e\_nmax\_nbh\_edges, 1063
  - edge, 1063
  - edge\_fwd\_iter, 1060
  - edge\_nbh\_edge\_fwd\_iter, 1060
  - edges, 1063
  - edges\_set\_t, 1060
  - edges\_t, 1060
  - graph, 1061
  - has\_e, 1063
  - has\_v, 1063
  - is\_subgraph\_of, 1064
  - v1, 1064
  - v2, 1064
  - v\_ith\_nbh\_edge, 1064
  - v\_ith\_nbh\_vertex, 1064
  - v\_nmax, 1064
  - v\_nmax\_nbh\_edges, 1065
  - v\_nmax\_nbh\_vertices, 1065
  - vertex, 1065
  - vertex\_fwd\_iter, 1061
  - vertex\_nbh\_edge\_fwd\_iter, 1061
  - vertex\_nbh\_vertex\_fwd\_iter, 1061
  - vertices\_t, 1061
- mln::util::greater\_point, 1065
- operator(), 1066
- mln::util::greater\_psite, 1066
  - operator(), 1066
- mln::util::head, 1066
- mln::util::ignore, 1067
- mln::util::ilcell, 1067
- mln::util::impl, 420
- mln::util::line\_graph, 1068
  - e\_ith\_nbh\_edge, 1071
  - e\_nmax, 1071
  - e\_nmax\_nbh\_edges, 1071
  - edge, 1071
  - edge\_fwd\_iter, 1070
  - edge\_nbh\_edge\_fwd\_iter, 1070
  - edges\_t, 1070
  - graph, 1072
  - has, 1072
  - has\_e, 1072
  - has\_v, 1072
  - is\_subgraph\_of, 1073
  - v1, 1073
  - v2, 1073
  - v\_ith\_nbh\_edge, 1073
  - v\_ith\_nbh\_vertex, 1073
  - v\_nmax, 1073
  - v\_nmax\_nbh\_edges, 1074
  - v\_nmax\_nbh\_vertices, 1074
  - vertex, 1074
  - vertex\_fwd\_iter, 1070
  - vertex\_nbh\_edge\_fwd\_iter, 1070
  - vertex\_nbh\_vertex\_fwd\_iter, 1071
  - vertices\_t, 1071
- mln::util::nil, 1074
- mln::util::node, 1075
- mln::util::object\_id, 1075
  - object\_id, 1077
  - value\_t, 1077
- mln::util::ord, 1077
- mln::util::ord\_pair, 1077
  - change\_both, 1079
  - change\_first, 1079
  - change\_second, 1079
  - first, 1079
  - second, 1079
- mln::util::pix, 1080
  - ima, 1081
  - p, 1081
  - pix, 1081
  - psite, 1080
  - v, 1081
  - value, 1080
- mln::util::set, 1081
  - bkd\_eiter, 1084
  - clear, 1084
  - eiter, 1084
  - element, 1084
  - first\_element, 1084
  - fwd\_eiter, 1084
  - has, 1085
  - insert, 1085
  - is\_empty, 1085
  - last\_element, 1086
  - memory\_size, 1086
  - nelements, 1086
  - remove, 1087
  - set, 1084
  - std\_vector, 1087
- mln::util::site\_pair, 1087
  - first, 1088
  - pair, 1088
  - second, 1089
- mln::util::soft\_heap, 1089
  - ~soft\_heap, 1091
  - clear, 1091
  - element, 1090
  - is\_empty, 1091
  - is\_valid, 1091

- nelements, 1091
- pop\_front, 1091
- push, 1091, 1092
- soft\_heap, 1090
- mln::util::timer, 1092
- mln::util::tracked\_ptr, 1093
  - ~tracked\_ptr, 1094
  - operator bool, 1094
  - operator->, 1094
  - operator=, 1095
  - tracked\_ptr, 1094
- mln::util::tree, 1095
  - add\_tree\_down, 1096
  - add\_tree\_up, 1096
  - check\_consistency, 1096
  - main\_branch, 1097
  - root, 1097
  - tree, 1096
- mln::util::tree\_node, 1097
  - add\_child, 1099
  - check\_consistency, 1099
  - children, 1100
  - delete\_tree\_node, 1100
  - elt, 1100
  - parent, 1101
  - print, 1101
  - search, 1101
  - search\_rec, 1101
  - set\_parent, 1102
  - tree\_node, 1099
- mln::util::vertex, 1102
  - Category, 1104
  - change\_graph, 1105
  - edge\_with, 1105
  - graph, 1105
  - graph\_t, 1104
  - id, 1105
  - id\_t, 1104
  - id\_value\_t, 1104
  - invalidate, 1105
  - is\_valid, 1105
  - ith\_nbh\_edge, 1105
  - ith\_nbh\_vertex, 1106
  - nmax\_nbh\_edges, 1106
  - nmax\_nbh\_vertices, 1106
  - operator vertex\_id\_t, 1106
  - other, 1106
  - update\_id, 1106
  - vertex, 1104
- mln::util::yes, 1107
- mln::Value, 1107
- mln::value, 420
  - cast, 427
  - equiv, 427
  - float01\_16, 425
  - float01\_8, 425
  - gl16, 425
  - gl8, 425
  - glf, 425
  - int\_s16, 425
  - int\_s32, 426
  - int\_s8, 426
  - int\_u12, 426
  - int\_u16, 426
  - int\_u32, 426
  - int\_u8, 426
  - label\_16, 426
  - label\_32, 426
  - label\_8, 426
  - operator<<, 428–431
  - operator\*, 427
  - operator+, 427, 428
  - operator-, 428
  - operator/, 428
  - operator==, 431
  - other, 432
  - rgb16, 427
  - rgb8, 427
  - stack, 432
- mln::value::float01, 1109
  - enc, 1110
  - equiv, 1110
  - float01, 1110
  - nbits, 1110
  - operator float, 1110
  - set\_nbits, 1111
  - to\_nbits, 1111
  - value, 1111
  - value\_ind, 1111
- mln::value::float01\_f, 1111
  - float01\_f, 1112
  - operator float, 1112
  - operator=, 1112
  - value, 1112
- mln::value::graylevel, 1112
  - graylevel, 1114
  - operator=, 1115
  - to\_float, 1115
  - value, 1115
- mln::value::graylevel\_f, 1116
  - graylevel\_f, 1117
  - operator graylevel< n >, 1117
  - operator=, 1117, 1118
  - value, 1118
- mln::value::impl, 432
- mln::value::int\_s, 1118
  - int\_s, 1120
  - one, 1120



- operator int, 1120
- operator=, 1120
- zero, 1120
- mln::value::int\_u, 1121
  - int\_u, 1122
  - next, 1122
  - operator unsigned, 1122
  - operator-, 1123
  - operator=, 1123
- mln::value::int\_u\_sat, 1123
  - int\_u\_sat, 1124
  - one, 1125
  - operator int, 1124
  - operator+/, 1125
  - operator=, 1125
  - operator=, 1125
  - zero, 1125
- mln::value::Integer, 1125
- mln::value::Integer< void >, 1126
- mln::value::label, 1126
  - enc, 1127
  - label, 1128
  - next, 1128
  - operator unsigned, 1128
  - operator++, 1128
  - operator--, 1128
  - operator=, 1128, 1129
  - prev, 1129
- mln::value::lut\_vec, 1129
  - bkd\_viter, 1131
  - fwd\_viter, 1131
  - has, 1132
  - index\_of, 1132
  - lut\_vec, 1132
  - nvalues, 1132
  - value, 1131
- mln::value::proxy, 1133
  - ~proxy, 1135
  - enc, 1134
  - equiv, 1134
  - operator=, 1135
  - proxy, 1134
  - to\_value, 1135
- mln::value::qt::rgb32, 1135
  - operator=, 1137
  - red, 1137
  - rgb32, 1136
  - zero, 1137
- mln::value::rgb, 1137
  - operator=, 1139
  - red, 1139
  - rgb, 1138
  - zero, 1139
- mln::value::set, 1139
  - the, 1140
- mln::value::sign, 1140
  - enc, 1141
  - equiv, 1141
  - one, 1142
  - operator int, 1142
  - operator=, 1142
  - sign, 1141
  - zero, 1142
- mln::value::stack\_image, 1142
  - domain\_t, 1143
  - is\_valid, 1144
  - lvalue, 1143
  - operator(), 1144, 1145
  - psite, 1143
  - rvalue, 1144
  - skeleton, 1144
  - stack\_image, 1144
  - value, 1144
- mln::value::super\_value< sign >, 1145
- mln::value::value\_array, 1145
  - operator(), 1146
  - value\_array, 1146
  - vset, 1146
- mln::Value\_Iterator, 1146
  - next, 1148
  - operator<<, 1148
- mln::Value\_Set, 1148
- mln::Vertex, 1149
- mln::vertex\_image, 1150
  - graph\_t, 1151
  - nbh\_t, 1151
  - operator(), 1152
  - site\_function\_t, 1151
  - skeleton, 1151
  - vertex\_image, 1152
  - vertex\_nbh\_t, 1151
  - vertex\_win\_t, 1151
  - win\_t, 1151
- mln::violent\_cast\_image, 1152
  - lvalue, 1153
  - operator(), 1154
  - rvalue, 1153
  - skeleton, 1153
  - value, 1153
  - violent\_cast\_image, 1154
- mln::w\_window, 1154
  - bkd\_qiter, 1156
  - clear, 1156
  - dpsite, 1156
  - fwd\_qiter, 1156
  - insert, 1156
  - is\_symmetric, 1157
  - operator<<, 1158

- operator==, 1158
- std\_vector, 1157
- sym, 1157
- w, 1157
- w\_window, 1156
- weight, 1156
- weights, 1157
- win, 1157
- mln::Weighted\_Window, 1158
  - operator-, 1159
- mln::win, 432
  - diff, 434
  - mln\_regular, 434
  - sym, 434
- mln::win::backdiag2d, 1159
  - backdiag2d, 1160
  - length, 1160
- mln::win::ball, 1160
  - ball, 1161
  - diameter, 1161
- mln::win::cube3d, 1161
  - cube3d, 1162
  - length, 1162
- mln::win::cuboid3d, 1162
  - cuboid3d, 1163
  - depth, 1164
  - height, 1164
  - volume, 1164
  - width, 1164
- mln::win::diag2d, 1164
  - diag2d, 1165
  - length, 1165
- mln::win::line, 1165
  - length, 1167
  - line, 1166
  - size, 1167
- mln::win::multiple, 1167
- mln::win::multiple\_size, 1167
- mln::win::octagon2d, 1168
  - area, 1169
  - length, 1169
  - octagon2d, 1168
- mln::win::rectangle2d, 1169
  - area, 1170
  - height, 1170
  - rectangle2d, 1170
  - std\_vector, 1170
  - width, 1170
- mln::Window, 1170
- mln::window, 1171
  - bkd\_qiter, 1173
  - clear, 1174
  - delta, 1174
  - dp, 1174
  - fwd\_qiter, 1173
  - has, 1174
  - insert, 1174
  - is\_centered, 1175
  - is\_empty, 1175
  - is\_symmetric, 1175
  - operator==, 1176
  - print, 1175
  - qiter, 1173
  - regular, 1173
  - size, 1175
  - std\_vector, 1175
  - sym, 1176
  - window, 1173
- mln::world::inter\_pixel::is\_separator, 1176
- mln\_ch\_convolve
  - mln::linear, 317
- mln\_ch\_convolve\_grad
  - mln::linear, 318
- mln\_exact
  - mln, 153
- mln\_gen\_complex\_neighborhood
  - mln, 153
- mln\_gen\_complex\_window
  - mln, 154
- mln\_gen\_complex\_window\_p
  - mln, 154, 155
- mln\_geom
  - mln::topo::is\_simple\_cell, 1018
- mln\_image\_from\_grid
  - mln::convert, 200
- mln\_meta\_accu\_result
  - mln::accu, 166
  - mln::data, 209
  - mln::set, 394
- mln\_q\_subject
  - mln::p\_graph\_piter, 869
- mln\_regular
  - mln, 155
  - mln::win, 434
- mln\_trait\_op\_geq
  - mln, 155
- mln\_trait\_op\_greater
  - mln, 155
- mln\_trait\_op\_leq
  - mln, 156
- mln\_trait\_op\_minus\_twice
  - mln::morpho::elementary, 369
- mln\_trait\_op\_neq
  - mln, 156
- mln\_window
  - mln::convert, 200
- modneighb1d
  - c2, 102

- neighb1d, 102
- modneighb2d
  - c2\_col, 103
  - c2\_row, 103
  - c4, 103
  - c8, 104
  - neighb2d, 103
- modneighb3d
  - c18, 105
  - c26, 105
  - c2\_3d\_sli, 106
  - c4\_3d, 106
  - c6, 107
  - c8\_3d, 107
  - neighb3d, 105
- modwin1d
  - segment1d, 112
  - window1d, 113
- modwin2d
  - disk2d, 114
  - hline2d, 114
  - vline2d, 114
  - win\_c4p, 115
  - win\_c8p, 115
  - window2d, 114
- modwin3d
  - sline3d, 116
  - sphere3d, 116
  - win\_c4p\_3d, 117
  - win\_c8p\_3d, 117
  - window3d, 117
- mosaic
  - mln::debug, 228
- Multiple accumulators, 94
- Multiple windows, 118
- n
  - mln::complex\_psite, 592
  - mln::faces\_psite, 668
  - mln::p\_n\_faces\_bkd\_piter, 894
  - mln::p\_n\_faces\_fwd\_piter, 895
  - mln::topo::algebraic\_face, 995
  - mln::topo::algebraic\_n\_face, 1000
  - mln::topo::face, 1011
  - mln::topo::n\_face, 1022
  - mln::topo::n\_face\_bkd\_iter, 1024
  - mln::topo::n\_face\_fwd\_iter, 1025
- N-D windows, 118
- n\_face
  - mln::topo::n\_face, 1020
- n\_face\_bkd\_iter
  - mln::topo::n\_face\_bkd\_iter, 1023
- n\_face\_fwd\_iter
  - mln::topo::n\_face\_fwd\_iter, 1025
- n\_items
  - mln::accu::stat::var, 536
  - mln::accu::stat::variance, 538
- nbh\_t
  - mln::edge\_image, 656
  - mln::vertex\_image, 1151
- nbits
  - mln::value::float01, 1110
- ncols
  - mln::geom, 259
  - mln::image2d, 773
  - mln::image3d, 781
- nedges
  - mln::p\_edges, 865
- neighb
  - mln::neighb, 843
- neighb1d
  - modneighb1d, 102
- neighb2d
  - modneighb2d, 103
- neighb3d
  - modneighb3d, 105
- Neighborhoods, 101
- nelements
  - mln::doc::Fastest\_Image, 612
  - mln::image1d, 768
  - mln::image2d, 773
  - mln::image3d, 781
  - mln::p\_mutable\_array\_of, 893
  - mln::p\_set\_of, 924
  - mln::util::array, 1041
  - mln::util::fibonacci\_heap, 1057
  - mln::util::set, 1086
  - mln::util::soft\_heap, 1091
- next
  - mln::bkd\_pixter1d, 549
  - mln::bkd\_pixter2d, 550
  - mln::bkd\_pixter3d, 552
  - mln::box\_runend\_piter, 567
  - mln::box\_runstart\_piter, 568
  - mln::complex\_neighborhood\_bkd\_piter, 588
  - mln::complex\_neighborhood\_fwd\_piter, 590
  - mln::complex\_window\_bkd\_piter, 594
  - mln::complex\_window\_fwd\_piter, 596
  - mln::dpoints\_bkd\_pixter, 651
  - mln::dpoints\_fwd\_pixter, 654
  - mln::fwd\_pixter1d, 720
  - mln::fwd\_pixter2d, 722
  - mln::fwd\_pixter3d, 723
  - mln::graph\_window\_if\_piter, 755
  - mln::graph\_window\_piter, 758
  - mln::Iterator, 789
  - mln::p\_graph\_piter, 870
  - mln::p\_indexed\_bkd\_piter, 878

- mln::p\_indexed\_fwd\_piter, 880
- mln::p\_n\_faces\_bkd\_piter, 894
- mln::p\_n\_faces\_fwd\_piter, 895
- mln::p\_transformed\_piter, 928
- mln::Pixel\_Iterator, 941
- mln::Site\_Iterator, 965
- mln::topo::adj\_higher\_dim\_connected\_n\_-  
face\_bkd\_iter, 978
- mln::topo::adj\_higher\_dim\_connected\_n\_-  
face\_fwd\_iter, 979
- mln::topo::adj\_higher\_face\_bkd\_iter, 980
- mln::topo::adj\_higher\_face\_fwd\_iter, 981
- mln::topo::adj\_lower\_dim\_connected\_n\_-  
face\_bkd\_iter, 982
- mln::topo::adj\_lower\_dim\_connected\_n\_-  
face\_fwd\_iter, 983
- mln::topo::adj\_lower\_face\_bkd\_iter, 984
- mln::topo::adj\_lower\_face\_fwd\_iter, 985
- mln::topo::adj\_lower\_higher\_face\_bkd\_iter,  
987
- mln::topo::adj\_lower\_higher\_face\_fwd\_iter,  
988
- mln::topo::adj\_m\_face\_bkd\_iter, 989
- mln::topo::adj\_m\_face\_fwd\_iter, 991
- mln::topo::center\_only\_iter, 1002
- mln::topo::centered\_bkd\_iter\_adapter, 1003
- mln::topo::centered\_fwd\_iter\_adapter, 1004
- mln::topo::face\_bkd\_iter, 1013
- mln::topo::face\_fwd\_iter, 1014
- mln::topo::n\_face\_bkd\_iter, 1024
- mln::topo::n\_face\_fwd\_iter, 1025
- mln::topo::static\_n\_face\_bkd\_iter, 1029
- mln::topo::static\_n\_face\_fwd\_iter, 1030
- mln::util::branch\_iter, 1045
- mln::util::branch\_iter\_ind, 1047
- mln::value::int\_u, 1122
- mln::value::label, 1128
- mln::Value\_Iterator, 1148
- nfaces
  - mln::p\_complex, 859
  - mln::p\_faces, 868
  - mln::topo::complex, 1006
- nfaces\_of\_dim
  - mln::p\_complex, 859
  - mln::topo::complex, 1007
- nfaces\_of\_static\_dim
  - mln::topo::complex, 1007
- ninds
  - mln::geom, 260
  - mln::image1d, 768
- niter
  - mln::doc::Neighborhood, 624
  - mln::graph\_elt\_mixed\_neighborhood, 734
  - mln::graph\_elt\_neighborhood, 740
  - mln::graph\_elt\_neighborhood\_if, 741
  - mln::mixed\_neighb, 831
  - mln::neighb, 843
- nlabels
  - mln::labeled\_image, 792
  - mln::labeled\_image\_base, 795
- nmax\_nbh\_edges
  - mln::util::edge, 1053
  - mln::util::vertex, 1106
- nmax\_nbh\_vertices
  - mln::util::vertex, 1106
- not\_inplace
  - mln::logical, 328
- nrows
  - mln::geom, 260
  - mln::image2d, 773
  - mln::image3d, 782
- nsites
  - mln::accu::center, 438
  - mln::Box, 563
  - mln::box, 558
  - mln::doc::Box, 603
  - mln::doc::Fastest\_Image, 612
  - mln::doc::Image, 621
  - mln::geom, 260
  - mln::p\_array, 851
  - mln::p\_complex, 859
  - mln::p\_edges, 865
  - mln::p\_faces, 868
  - mln::p\_image, 877
  - mln::p\_key, 885
  - mln::p\_line2d, 889
  - mln::p\_priority, 900
  - mln::p\_queue, 905
  - mln::p\_queue\_fast, 910
  - mln::p\_run, 916
  - mln::p\_set, 920
  - mln::p\_vertices, 938
- nslis
  - mln::geom, 260
  - mln::image3d, 782
- nvalues
  - mln::doc::Value\_Set, 638
  - mln::value::lut\_vec, 1132
- nvertices
  - mln::p\_vertices, 938
- object\_id
  - mln::util::object\_id, 1077
- octagon2d
  - mln::win::octagon2d, 1168
- olive
  - mln::literal, 324
- On images, 91

- On site sets, [91](#)
- On values, [92](#)
- one
  - `mln::literal`, [324](#)
  - `mln::value::int_s`, [1120](#)
  - `mln::value::int_u_sat`, [1125](#)
  - `mln::value::sign`, [1142](#)
- opening
  - `mln::morpho::elementary`, [369](#)
- operator bool
  - `mln::util::tracked_ptr`, [1094](#)
- operator decorated\_image< const I, D >
  - `mln::decorated_image`, [599](#)
- operator edge\_id\_t
  - `mln::util::edge`, [1053](#)
- operator float
  - `mln::value::float01`, [1110](#)
  - `mln::value::float01_f`, [1112](#)
- operator graylevel< n >
  - `mln::value::graylevel_f`, [1117](#)
- operator I
  - `mln::plain`, [943](#)
- operator image\_if< const I, F >
  - `mln::image_if`, [784](#)
- operator int
  - `mln::value::int_s`, [1120](#)
  - `mln::value::int_u_sat`, [1124](#)
  - `mln::value::sign`, [1142](#)
- operator mat< n, I, U >
  - `mln::algebra::h_vec`, [547](#)
- operator mln::algebra::vec< dpoint< G, C >::dim, Q >
  - `mln::dpoint`, [647](#)
- operator psite
  - `mln::doc::Site_Iterator`, [632](#)
- operator safe\_image< const I >
  - `mln::safe_image`, [961](#)
- operator slice\_image< const I >
  - `mln::slice_image`, [971](#)
- operator sub\_image< const I, S >
  - `mln::sub_image`, [973](#)
- operator thru\_image< const I, F >
  - `mln::thru_image`, [975](#)
- operator thrubin\_image< const I1, const I2, F >
  - `mln::thrubin_image`, [976](#)
- operator typename internal::p\_image\_site\_set< I >::ret
  - `mln::p_image`, [877](#)
- operator unsigned
  - `mln::value::int_u`, [1122](#)
  - `mln::value::label`, [1128](#)
- operator util::tree\_node< T > &
  - `mln::util::branch_iter`, [1045](#)
  - `mln::util::branch_iter_ind`, [1047](#)
- operator value
  - `mln::doc::Value_Iterator`, [636](#)
- operator vertex\_id\_t
  - `mln::util::vertex`, [1106](#)
- operator<
  - `mln`, [158](#)
  - `mln::Box`, [564](#)
  - `mln::Site_Set`, [968](#)
  - `mln::topo`, [405](#), [406](#)
  - `mln::util`, [418](#)
- operator<<
  - `mln`, [158](#), [159](#)
  - `mln::Box`, [564](#)
  - `mln::box`, [560](#)
  - `mln::fun::i2v`, [244](#)
  - `mln::Gpoint`, [730](#)
  - `mln::Point_Site`, [956](#)
  - `mln::Site_Set`, [968](#)
  - `mln::topo`, [406](#), [407](#)
  - `mln::util`, [418](#)
  - `mln::value`, [428–431](#)
  - `mln::Value_Iterator`, [1148](#)
  - `mln::w_window`, [1158](#)
- operator<=
  - `mln`, [159](#), [160](#)
  - `mln::Box`, [564](#), [565](#)
  - `mln::Site_Set`, [968](#)
- operator\*
  - `mln`, [157](#)
  - `mln::algebra`, [174](#)
  - `mln::value`, [427](#)
- operator()
  - `mln::complex_image`, [586](#)
  - `mln::decorated_image`, [599](#)
  - `mln::doc::Fastest_Image`, [612](#)
  - `mln::doc::Image`, [621](#)
  - `mln::edge_image`, [657](#)
  - `mln::extension_fun`, [661](#)
  - `mln::extension_ima`, [663](#)
  - `mln::extension_val`, [666](#)
  - `mln::flat_image`, [671](#)
  - `mln::fun::v2v::rgb8_to_rgbn`, [684](#)
  - `mln::fun::x2v::bilinear`, [703](#)
  - `mln::fun::x2x::linear`, [706](#)
  - `mln::fun::x2x::rotation`, [709](#)
  - `mln::fun::x2x::translation`, [712](#)
  - `mln::fun_image`, [714](#)
  - `mln::geom::complex_geometry`, [726](#)
  - `mln::hexa`, [762](#)
  - `mln::image1d`, [768](#)
  - `mln::image2d`, [773](#)
  - `mln::image2d_h`, [777](#)
  - `mln::image3d`, [782](#)
  - `mln::lazy_image`, [798](#)

- mln::p2p\_image, 846, 847
- mln::p\_key, 885
- mln::p\_priority, 901
- mln::p\_vaccess, 932
- mln::p\_vertices, 938
- mln::slice\_image, 971
- mln::topo::is\_simple\_cell, 1018
- mln::tr\_image, 1033
- mln::unproject\_image, 1034, 1035
- mln::util::array, 1041
- mln::util::greater\_point, 1066
- mln::util::greater\_psite, 1066
- mln::value::stack\_image, 1144, 1145
- mln::value::value\_array, 1146
- mln::vertex\_image, 1152
- mln::violent\_cast\_image, 1154
- operator+
  - mln::Gpoint, 728
  - mln::Point\_Site, 954
  - mln::topo, 404
  - mln::value, 427, 428
- operator++
  - mln, 157
  - mln::value::label, 1128
- operator+=
  - mln::Gpoint, 728
  - mln::Point, 945
  - mln::point, 951
  - mln::value::int\_u\_sat, 1125
- operator-
  - mln, 157
  - mln::Gpoint, 729
  - mln::Point\_Site, 955
  - mln::topo, 405
  - mln::value, 428
  - mln::value::int\_u, 1123
  - mln::Weighted\_Window, 1159
- operator->
  - mln::util::tracked\_ptr, 1094
- operator--
  - mln, 158
  - mln::value::label, 1128
- operator-=
  - mln::Gpoint, 729
  - mln::Point, 946
  - mln::point, 951
  - mln::value::int\_u\_sat, 1125
- operator/
  - mln::Gpoint, 730
  - mln::Point, 946
  - mln::value, 428
- operator=
  - mln::plain, 943
  - mln::util::fibonacci\_heap, 1057
  - mln::util::tracked\_ptr, 1095
  - mln::value::float01\_f, 1112
  - mln::value::graylevel, 1115
  - mln::value::graylevel\_f, 1117, 1118
  - mln::value::int\_s, 1120
  - mln::value::int\_u, 1123
  - mln::value::int\_u\_sat, 1125
  - mln::value::label, 1128, 1129
  - mln::value::proxy, 1135
  - mln::value::qt::rgb32, 1137
  - mln::value::rgb, 1139
  - mln::value::sign, 1142
- operator==
  - mln, 160, 161
  - mln::accu::stat, 173
  - mln::Box, 565
  - mln::Gpoint, 730
  - mln::Point\_Site, 956
  - mln::Site\_Set, 968
  - mln::topo, 407, 408
  - mln::util, 418, 419
  - mln::value, 431
  - mln::w\_window, 1158
  - mln::window, 1176
- or\_inplace
  - mln::logical, 328
- orange
  - mln::literal, 324
- ord\_pair
  - mln::make, 345
- ord\_strict
  - mln::util, 419
- ord\_weak
  - mln::util, 419
- origin
  - mln::algebra::h\_vec, 548
  - mln::literal, 324
  - mln::point, 953
- other
  - mln::util::vertex, 1106
  - mln::value, 432
- overset
  - mln::p\_if, 872
- P
  - mln::graph\_window\_if\_piter, 754
  - mln::graph\_window\_piter, 756
- p
  - mln::util::pix, 1081
- p2p\_image
  - mln::p2p\_image, 846
- p\_array
  - mln::p\_array, 850
- p\_centered

- mln::p\_centered, 854
- p\_complex
  - mln::p\_complex, 858
- p\_edges
  - mln::p\_edges, 863
- p\_edges\_with\_mass\_centers
  - mln::make, 345
- p\_faces
  - mln::p\_faces, 867
- p\_graph\_piter
  - mln::p\_graph\_piter, 869
- p\_if
  - mln::p\_if, 872
- p\_image
  - mln::p\_image, 876
- p\_indexed\_bkd\_piter
  - mln::p\_indexed\_bkd\_piter, 878
- p\_indexed\_fwd\_piter
  - mln::p\_indexed\_fwd\_piter, 879
- p\_key
  - mln::p\_key, 883
- p\_line2d
  - mln::p\_line2d, 888
- p\_mutable\_array\_of
  - mln::p\_mutable\_array\_of, 892
- p\_n\_faces\_bkd\_piter
  - mln::p\_n\_faces\_bkd\_piter, 894
- p\_n\_faces\_fwd\_piter
  - mln::p\_n\_faces\_fwd\_piter, 895
- p\_priority
  - mln::p\_priority, 899
- p\_queue
  - mln::p\_queue, 904
- p\_queue\_fast
  - mln::p\_queue\_fast, 909
- p\_run
  - mln::p\_run, 914
- p\_run2d
  - mln, 147
- p\_runs2d
  - mln, 147
- p\_set
  - mln::p\_set, 919
- p\_set\_of
  - mln::p\_set\_of, 923
- p\_transformed
  - mln::p\_transformed, 926
- p\_transformed\_piter
  - mln::p\_transformed\_piter, 928
- p\_vaccess
  - mln::p\_vaccess, 931
- p\_vertices
  - mln::p\_vertices, 936
- p\_vertices\_with\_mass\_centers
  - mln::make, 346
- pack
  - mln::labeling, 305
- pack\_inplace
  - mln::labeling, 306
- pair
  - mln::util::site\_pair, 1088
- parent
  - mln::util::tree\_node, 1101
- paste
  - mln::data, 209
  - mln::data::impl::generic, 221
- paste\_without\_localization
  - mln::data, 210
- paste\_without\_localization\_fast
  - mln::data::impl, 218
- paste\_without\_localization\_fastest
  - mln::data::impl, 218
- paste\_without\_localization\_lines
  - mln::data::impl, 218
- pcenter
  - mln::box, 559
- pink
  - mln::literal, 325
- piter
  - mln::box, 556
  - mln::p\_array, 849
  - mln::p\_centered, 854
  - mln::p\_complex, 857
  - mln::p\_edges, 862
  - mln::p\_faces, 867
  - mln::p\_if, 871
  - mln::p\_image, 875
  - mln::p\_key, 883
  - mln::p\_line2d, 887
  - mln::p\_mutable\_array\_of, 892
  - mln::p\_priority, 898
  - mln::p\_queue, 904
  - mln::p\_queue\_fast, 909
  - mln::p\_run, 914
  - mln::p\_set, 918
  - mln::p\_set\_of, 922
  - mln::p\_transformed, 925
  - mln::p\_vaccess, 930
  - mln::p\_vertices, 935
- pix
  - mln::make, 346
  - mln::util::pix, 1081
- pixel
  - mln::make, 346
  - mln::pixel, 940
- plain
  - mln::plain, 943
- plot

- mln::draw, 236
- plus
  - mln::arith, 182, 183
  - mln::morpho, 364
- plus\_cst
  - mln::arith, 183, 184
- plus\_cst\_inplace
  - mln::arith, 184
- plus\_infty
  - mln::point, 952
- plus\_inplace
  - mln::arith, 185
- pmax
  - mln::box, 559
  - mln::doc::Box, 603
- pmin
  - mln::box, 559
  - mln::doc::Box, 603
- pmin\_pmax
  - mln::geom, 260, 261
- point
  - mln::doc::Dpoint, 605
  - mln::doc::Fastest\_Image, 609
  - mln::doc::Image, 619
  - mln::doc::Neighborhood, 624
  - mln::doc::Point\_Site, 629
  - mln::doc::Weighted\_Window, 640
  - mln::Point, 945
  - mln::point, 950, 951
- point1d
  - mln, 147
- point1df
  - mln, 148
- point2d
  - mln, 148
- point2d\_h
  - mln, 148
  - mln::make, 347
- point2df
  - mln, 148
- point3d
  - mln, 148
- point3df
  - mln, 148
- point\_at\_index
  - mln::doc::Fastest\_Image, 613
  - mln::image1d, 768
  - mln::image2d, 773
  - mln::image3d, 782
- polygon
  - mln::draw, 236
- pop
  - mln::p\_priority, 901
  - mln::p\_queue, 905
  - mln::p\_queue\_fast, 911
- pop\_front
  - mln::p\_priority, 901
  - mln::p\_queue, 906
  - mln::p\_queue\_fast, 911
  - mln::util::fibonacci\_heap, 1057
  - mln::util::soft\_heap, 1091
- positive
  - mln::test, 397
- pred
  - mln::p\_if, 873
- predicate
  - mln::p\_if, 873
  - mln::test, 397, 398
- prev
  - mln::value::label, 1129
- primary
  - mln, 163
- primary\_set
  - mln::p\_transformed, 927
- print
  - mln::topo::complex, 1007
  - mln::util::tree\_node, 1101
  - mln::window, 1175
- print\_faces
  - mln::topo::complex, 1007
- println
  - mln::debug, 228
- println\_with\_border
  - mln::debug, 228
- priorities
  - mln::p\_priority, 901
- propagate\_if
  - mln::morpho::tree, 376, 377
- propagate\_if\_value
  - mln::morpho::tree, 377
- propagate\_node\_to\_ancestors
  - mln::morpho::tree, 378
- propagate\_node\_to\_descendants
  - mln::morpho::tree, 378, 379
- propagate\_representative
  - mln::morpho::tree, 379
- proxy
  - mln::value::proxy, 1134
- pset
  - mln::doc::Fastest\_Image, 609
  - mln::doc::Image, 619
  - mln::p\_vaccess, 931
- psite
  - mln::box, 556
  - mln::complex\_neighborhood\_bkd\_piter, 587
  - mln::complex\_neighborhood\_fwd\_piter, 589
  - mln::complex\_window\_bkd\_piter, 594
  - mln::complex\_window\_fwd\_piter, 595



- mln::decorated\_image, 598
- mln::doc::Box, 602
- mln::doc::Fastest\_Image, 609
- mln::doc::Image, 619
- mln::doc::Site\_Iterator, 632
- mln::doc::Site\_Set, 634
- mln::dpoint, 646
- mln::graph\_elt\_mixed\_window, 737
- mln::graph\_elt\_window, 745
- mln::graph\_elt\_window\_if, 749
- mln::hexa, 760
- mln::image2d\_h, 776
- mln::interpolated, 785
- mln::p\_array, 849
- mln::p\_centered, 854
- mln::p\_complex, 857
- mln::p\_edges, 862
- mln::p\_faces, 867
- mln::p\_if, 872
- mln::p\_image, 875
- mln::p\_key, 883
- mln::p\_line2d, 888
- mln::p\_mutable\_array\_of, 892
- mln::p\_priority, 898
- mln::p\_queue, 904
- mln::p\_queue\_fast, 909
- mln::p\_run, 914
- mln::p\_set, 919
- mln::p\_set\_of, 923
- mln::p\_transformed, 926
- mln::p\_vaccess, 931
- mln::p\_vertices, 935
- mln::thrubin\_image, 976
- mln::topo::is\_simple\_cell, 1018
- mln::tr\_image, 1032
- mln::util::pix, 1080
- mln::value::stack\_image, 1143
- pttransform
  - mln, 163
- purge
  - mln::p\_queue\_fast, 911
- purple
  - mln::literal, 325
- push
  - mln::p\_priority, 901
  - mln::p\_queue, 906
  - mln::p\_queue\_fast, 911
  - mln::util::fibonacci\_heap, 1057, 1058
  - mln::util::soft\_heap, 1091, 1092
- put\_word
  - mln::debug, 229
- q\_box
  - mln::p\_line2d, 888
- mln::p\_run, 914
- qiter
  - mln::doc::Window, 643
  - mln::graph\_elt\_mixed\_window, 737
  - mln::graph\_elt\_window, 745
  - mln::graph\_elt\_window\_if, 749
  - mln::window, 1173
- Queue based, 110
- r\_element
  - mln::p\_image, 875
  - mln::p\_key, 883
  - mln::p\_set, 919
- rag\_and\_labeled\_wsl
  - mln::make, 347
- rank\_filter
  - mln::morpho, 365
- read\_header
  - mln::io::fld, 276
- rectangle2d
  - mln::win::rectangle2d, 1170
- rectangularity
  - mln::accu::site\_set::rectangularity, 510
- red
  - mln::literal, 325
  - mln::value::qt::rgb32, 1137
  - mln::value::rgb, 1139
- region\_adjacency\_graph
  - mln::make, 348
- regional\_maxima
  - mln::labeling, 306
- regional\_minima
  - mln::labeling, 306
- registration1
  - mln::registration, 391
- registration2
  - mln::registration, 391
- registration3
  - mln::registration, 391
- regular
  - mln::window, 1173
- relabel
  - mln::labeled\_image, 792
  - mln::labeled\_image\_base, 795
  - mln::labeling, 307
- relabel\_inplace
  - mln::labeling, 308
- relabelfun
  - mln::make, 348
- remove
  - mln::p\_image, 877
  - mln::p\_key, 885
  - mln::p\_set, 920
  - mln::util::set, 1087

- remove\_key
  - mln::p\_key, 885
- replace
  - mln::data, 210
- reserve
  - mln::p\_array, 852
  - mln::p\_mutable\_array\_of, 893
  - mln::p\_queue\_fast, 911
  - mln::topo::n\_faces\_set, 1027
  - mln::util::array, 1042
- resize
  - mln::border, 192
  - mln::p\_array, 852
  - mln::util::array, 1042
- result
  - mln::graph::attribute::card\_t, 732
  - mln::graph::attribute::representative\_t, 733
  - mln::topo::is\_simple\_cell, 1018
  - mln::util::array, 1039
- revert
  - mln::arith, 185
- revert\_inplace
  - mln::arith, 186
- rgb
  - mln::value::rgb, 1138
- rgb16
  - mln::value, 427
- rgb32
  - mln::value::qt::rgb32, 1136
- rgb8
  - mln::value, 427
- rgb8\_2complex\_image3df
  - mln, 148
- root
  - mln::util::tree, 1097
- rotate
  - mln::geom, 261, 262
- rotation
  - mln::fun::x2x::rotation, 708, 709
- Routines, 99
- run\_length
  - mln::box\_runend\_piter, 567
  - mln::box\_runstart\_piter, 568
- rvalue
  - mln::complex\_image, 585
  - mln::decorated\_image, 598
  - mln::doc::Fastest\_Image, 610
  - mln::doc::Generalized\_Pixel, 615
  - mln::doc::Image, 619
  - mln::doc::Pixel\_Iterator, 627
  - mln::extension\_fun, 660
  - mln::extension\_ima, 662
  - mln::extension\_val, 665
  - mln::flat\_image, 670
  - mln::fun\_image, 713
  - mln::hexa, 761
  - mln::image1d, 766
  - mln::image2d, 771
  - mln::image2d\_h, 776
  - mln::image3d, 779
  - mln::interpolated, 785
  - mln::lazy\_image, 797
  - mln::thrubin\_image, 976
  - mln::tr\_image, 1032
  - mln::value::stack\_image, 1144
  - mln::violent\_cast\_image, 1153
- S
  - mln::p\_image, 875
- sagittal\_dec
  - mln, 163
- saturate
  - mln::data, 210, 211
- saturate\_inplace
  - mln::data, 211
- save
  - mln::io::cloud, 271
  - mln::io::dump, 274
  - mln::io::magick, 277
  - mln::io::off, 279
  - mln::io::pbm, 280
  - mln::io::pfm, 283
  - mln::io::pgm, 284
  - mln::io::plot, 286, 287
  - mln::io::pnm, 289
  - mln::io::ppm, 291
  - mln::io::raw, 293
  - mln::io::txt, 295
- save\_bin\_alt
  - mln::io::off, 279
- search
  - mln::util::tree\_node, 1101
- search\_rec
  - mln::util::tree\_node, 1101
- second
  - mln::accu::pair, 501
  - mln::accu::stat::min\_max, 529
  - mln::util::couple, 1049
  - mln::util::ord\_pair, 1079
  - mln::util::site\_pair, 1089
- second\_accu
  - mln::accu::pair, 501
  - mln::accu::stat::min\_max, 529
- seeds2tiling
  - mln::geom, 262
  - mln::geom::impl, 265
- seeds2tiling\_roundness
  - mln::geom, 263

- segmentId
  - modwinId, 112
- set
  - mln::util::set, 1084
- set\_all
  - mln::dpoint, 648
  - mln::point, 952
- set\_alpha
  - mln::fun::x2x::rotation, 709
- set\_axis
  - mln::fun::x2x::rotation, 709
- set\_cplx
  - mln::topo::algebraic\_face, 995
  - mln::topo::algebraic\_n\_face, 1000
  - mln::topo::face, 1011
  - mln::topo::n\_face, 1022
- set\_face\_id
  - mln::topo::algebraic\_face, 995
  - mln::topo::algebraic\_n\_face, 1000
  - mln::topo::face, 1011
  - mln::topo::n\_face, 1022
- set\_image
  - mln::topo::is\_simple\_cell, 1019
- set\_n
  - mln::topo::algebraic\_face, 995
  - mln::topo::face, 1012
- set\_nbits
  - mln::value::float01, 1111
- set\_parent
  - mln::util::tree\_node, 1102
- set\_sign
  - mln::topo::algebraic\_face, 996
  - mln::topo::algebraic\_n\_face, 1000
- set\_t
  - mln::fun::x2x::translation, 712
- set\_tr
  - mln::tr\_image, 1033
- set\_value
  - mln::accu::count\_adjacent\_vertices, 441
  - mln::accu::count\_labels, 443
  - mln::accu::count\_value, 444
  - mln::accu::math::count, 456
  - mln::accu::shape::height, 506
  - mln::accu::shape::volume, 509
  - mln::accu::stat::max, 516
  - mln::accu::stat::min, 525
  - mln::morpho::attribute::sum, 840
- sign
  - mln::topo::algebraic\_face, 996
  - mln::topo::algebraic\_n\_face, 1000
  - mln::value::sign, 1141
- site
  - mln::box, 556
  - mln::doc::Box, 602
  - mln::doc::Site\_Set, 634
  - mln::dpoint, 646
  - mln::graph\_elt\_mixed\_window, 737
  - mln::graph\_elt\_window, 745
  - mln::graph\_elt\_window\_if, 749
  - mln::graph\_window\_base, 752
  - mln::p\_centered, 854
  - mln::tr\_image, 1032
- Site sets, 108
- site\_function\_t
  - mln::edge\_image, 656
  - mln::vertex\_image, 1151
- site\_set
  - mln::complex\_psite, 592
  - mln::draw, 236
  - mln::faces\_psite, 668
- size
  - mln::util::array, 1042
  - mln::win::line, 1167
  - mln::window, 1175
- skeleton
  - mln::complex\_image, 585
  - mln::decorated\_image, 598
  - mln::doc::Fastest\_Image, 610
  - mln::doc::Image, 619
  - mln::edge\_image, 657
  - mln::extended, 658
  - mln::extension\_fun, 660
  - mln::extension\_ima, 662
  - mln::extension\_val, 665
  - mln::flat\_image, 670
  - mln::fun\_image, 713
  - mln::hexa, 761
  - mln::imageId, 766
  - mln::image2d, 771
  - mln::image2d\_h, 776
  - mln::image3d, 779
  - mln::image\_if, 783
  - mln::interpolated, 785
  - mln::labeled\_image, 791
  - mln::lazy\_image, 797
  - mln::p2p\_image, 846
  - mln::plain, 942
  - mln::pw::image, 959
  - mln::safe\_image, 961
  - mln::slice\_image, 970
  - mln::sub\_image, 972
  - mln::sub\_image\_if, 974
  - mln::thrubin\_image, 976
  - mln::tr\_image, 1032
  - mln::value::stack\_image, 1144
  - mln::vertex\_image, 1151
  - mln::violent\_cast\_image, 1153
- sli

- mln::slice\_image, 971
- slice\_image
  - mln::slice\_image, 971
- slices\_2d
  - mln::debug, 229
- sline3d
  - modwin3d, 116
- soft\_heap
  - mln::util::soft\_heap, 1090
- sort\_offsets\_increasing
  - mln::data, 211
- sort\_psites\_decreasing
  - mln::data, 211
- sort\_psites\_increasing
  - mln::data, 212
- space\_2complex\_geometry
  - mln, 148
- Sparse types, 109
- sphere3d
  - modwin3d, 116
- sqr\_l2
  - mln::norm, 386
- stack
  - mln::value, 432
- stack\_image
  - mln::value::stack\_image, 1144
- standard\_deviation
  - mln::accu::stat::variance, 538
- start
  - mln::doc::Iterator, 623
  - mln::doc::Pixel\_Iterator, 628
  - mln::doc::Site\_Iterator, 632
  - mln::doc::Value\_Iterator, 636
  - mln::dpoints\_bkd\_pixter, 651
  - mln::dpoints\_fwd\_pixter, 654
  - mln::p\_run, 916
  - mln::topo::face\_bkd\_iter, 1013
  - mln::topo::face\_fwd\_iter, 1014
  - mln::topo::n\_face\_bkd\_iter, 1024
  - mln::topo::n\_face\_fwd\_iter, 1025
  - mln::topo::static\_n\_face\_bkd\_iter, 1029
  - mln::topo::static\_n\_face\_fwd\_iter, 1030
  - mln::util::branch\_iter, 1046
  - mln::util::branch\_iter\_ind, 1047
- static\_n\_face\_bkd\_iter
  - mln::topo::static\_n\_face\_bkd\_iter, 1028
- static\_n\_face\_fwd\_iter
  - mln::topo::static\_n\_face\_fwd\_iter, 1030
- std\_deque
  - mln::p\_queue, 906
- std\_vector
  - mln::p\_array, 852
  - mln::p\_line2d, 890
  - mln::p\_queue\_fast, 911
- mln::p\_set, 920
- mln::util::array, 1043
- mln::util::set, 1087
- mln::w\_window, 1157
- mln::win::rectangle2d, 1170
- mln::window, 1175
- stretch
  - mln::data, 212
  - mln::data::impl, 219
- structural
  - mln::morpho::closing::approx, 368
  - mln::morpho::opening::approx, 371
- sub\_image
  - mln::sub\_image, 973
- sub\_image\_if
  - mln::sub\_image\_if, 974
- subdomain
  - mln::labeled\_image, 792
  - mln::labeled\_image\_base, 795
- subsampling
  - mln::subsampling, 396
- subtractive
  - mln::morpho::tree::filter, 381
- sum
  - mln::accu::stat::mean, 519
  - mln::accu::stat::variance, 539
  - mln::estim, 238, 239
- superpose
  - mln::debug, 229
  - mln::labeling, 308
  - mln::morpho::watershed, 383
- sym
  - mln::doc::Weighted\_Window, 641
  - mln::graph\_elt\_mixed\_window, 738
  - mln::graph\_elt\_window, 746
  - mln::graph\_elt\_window\_if, 751
  - mln::graph\_window\_base, 753
  - mln::w\_window, 1157
  - mln::win, 434
  - mln::window, 1176
- sym\_diff
  - mln::Box, 565
  - mln::Site\_Set, 969
- t
  - mln::algebra::h\_mat, 546
  - mln::algebra::h\_vec, 547
  - mln::fun::x2x::translation, 712
- take
  - mln::accu, 167
  - mln::accu::histo, 446
  - mln::accu::label\_used, 447
  - mln::accu::stat::histo3d\_rgb, 514
  - mln::accu::stat::median\_alt, 521

- mln::doc::Accumulator, 600
- take\_as\_init
  - mln::accu::center, 438
  - mln::accu::convolve, 440
  - mln::accu::count\_adjacent\_vertices, 441
  - mln::accu::count\_labels, 443
  - mln::accu::count\_value, 444
  - mln::accu::histo, 446
  - mln::accu::label\_used, 447
  - mln::accu::logic::land, 449
  - mln::accu::logic::land\_basic, 450
  - mln::accu::logic::lor, 452
  - mln::accu::logic::lor\_basic, 453
  - mln::accu::maj\_h, 455
  - mln::accu::math::count, 456
  - mln::accu::math::inf, 458
  - mln::accu::math::sum, 459
  - mln::accu::math::sup, 461
  - mln::accu::max\_site, 462
  - mln::accu::nil, 497
  - mln::accu::p, 499
  - mln::accu::pair, 502
  - mln::accu::rms, 503
  - mln::accu::shape::bbox, 504
  - mln::accu::shape::height, 507
  - mln::accu::shape::volume, 509
  - mln::accu::site\_set::rectangularity, 510
  - mln::accu::stat::deviation, 512
  - mln::accu::stat::histo3d\_rgb, 515
  - mln::accu::stat::max, 516
  - mln::accu::stat::max\_h, 518
  - mln::accu::stat::mean, 519
  - mln::accu::stat::median\_alt, 521
  - mln::accu::stat::median\_h, 523
  - mln::accu::stat::min, 525
  - mln::accu::stat::min\_h, 527
  - mln::accu::stat::min\_max, 529
  - mln::accu::stat::rank, 531
  - mln::accu::stat::rank< bool >, 533
  - mln::accu::stat::rank\_high\_quant, 534
  - mln::accu::stat::var, 536
  - mln::accu::stat::variance, 539
  - mln::accu::tuple, 540
  - mln::accu::val, 542
  - mln::Accumulator, 544
  - mln::morpho::attribute::card, 833
  - mln::morpho::attribute::count\_adjacent\_ -  
vertices, 834
  - mln::morpho::attribute::height, 836
  - mln::morpho::attribute::sharpness, 838
  - mln::morpho::attribute::sum, 840
  - mln::morpho::attribute::volume, 842
- take\_n\_times
  - mln::accu::center, 438
- mln::accu::convolve, 440
- mln::accu::count\_adjacent\_vertices, 441
- mln::accu::count\_labels, 443
- mln::accu::count\_value, 445
- mln::accu::histo, 446
- mln::accu::label\_used, 448
- mln::accu::logic::land, 449
- mln::accu::logic::land\_basic, 451
- mln::accu::logic::lor, 452
- mln::accu::logic::lor\_basic, 453
- mln::accu::maj\_h, 455
- mln::accu::math::count, 456
- mln::accu::math::inf, 458
- mln::accu::math::sum, 459
- mln::accu::math::sup, 461
- mln::accu::max\_site, 462
- mln::accu::nil, 497
- mln::accu::p, 499
- mln::accu::pair, 502
- mln::accu::rms, 503
- mln::accu::shape::bbox, 505
- mln::accu::shape::height, 507
- mln::accu::shape::volume, 509
- mln::accu::site\_set::rectangularity, 511
- mln::accu::stat::deviation, 512
- mln::accu::stat::histo3d\_rgb, 515
- mln::accu::stat::max, 516
- mln::accu::stat::max\_h, 518
- mln::accu::stat::mean, 520
- mln::accu::stat::median\_alt, 521
- mln::accu::stat::median\_h, 523
- mln::accu::stat::min, 525
- mln::accu::stat::min\_h, 527
- mln::accu::stat::min\_max, 530
- mln::accu::stat::rank, 531
- mln::accu::stat::rank< bool >, 533
- mln::accu::stat::rank\_high\_quant, 534
- mln::accu::stat::var, 536
- mln::accu::stat::variance, 539
- mln::accu::tuple, 541
- mln::accu::val, 542
- mln::Accumulator, 544
- mln::morpho::attribute::card, 833
- mln::morpho::attribute::count\_adjacent\_ -  
vertices, 834
- mln::morpho::attribute::height, 836
- mln::morpho::attribute::sharpness, 838
- mln::morpho::attribute::sum, 840
- mln::morpho::attribute::volume, 842
- target
  - mln::graph\_elt\_mixed\_window, 737
  - mln::graph\_elt\_window, 745
  - mln::graph\_elt\_window\_if, 749
- target\_site\_set

- mln::graph\_window\_piter, 758
- teal
- mln::literal, 325
- the
  - mln::value::set, 1140
- thick\_miss
  - mln::morpho, 365
- thickening
  - mln::morpho, 365
- thin\_fit
  - mln::morpho, 365
- thinning
  - mln::morpho, 365
- threshold
  - mln::binarization, 188
- times
  - mln::arith, 186
- times\_cst
  - mln::arith, 186
- times\_inplace
  - mln::arith, 187
- to
  - mln::convert, 200
- to\_dpoint
  - mln::convert, 201
  - mln::Dpoint, 643
- to\_enc
  - mln::data, 212
- to\_float
  - mln::value::graylevel, 1115
- to\_fun
  - mln::convert, 201, 203
- to\_h\_vec
  - mln::point, 952
- to\_image
  - mln::convert, 201
- to\_larger
  - mln::box, 559
- to\_nbits
  - mln::value::float01, 1111
- to\_neighb
  - mln::graph, 266
- to\_p\_array
  - mln::convert, 201
- to\_p\_set
  - mln::convert, 201, 202
- to\_point
  - mln::doc::Point\_Site, 630
  - mln::Point, 945
- to\_qimage
  - mln::convert, 202
- to\_result
  - mln::accu::center, 438
  - mln::accu::convolve, 440
  - mln::accu::count\_adjacent\_vertices, 442
  - mln::accu::count\_labels, 443
  - mln::accu::count\_value, 445
  - mln::accu::label\_used, 448
  - mln::accu::logic::land, 449
  - mln::accu::logic::land\_basic, 451
  - mln::accu::logic::lor, 452
  - mln::accu::logic::lor\_basic, 454
  - mln::accu::maj\_h, 455
  - mln::accu::math::count, 457
  - mln::accu::math::inf, 458
  - mln::accu::math::sum, 459
  - mln::accu::math::sup, 461
  - mln::accu::max\_site, 462
  - mln::accu::nil, 497
  - mln::accu::p, 499
  - mln::accu::pair, 502
  - mln::accu::rms, 503
  - mln::accu::shape::bbox, 505
  - mln::accu::shape::height, 507
  - mln::accu::shape::volume, 509
  - mln::accu::site\_set::rectangularity, 511
  - mln::accu::stat::deviation, 512
  - mln::accu::stat::histo3d\_rgb, 515
  - mln::accu::stat::max, 516
  - mln::accu::stat::max\_h, 518
  - mln::accu::stat::mean, 520
  - mln::accu::stat::median\_alt, 521
  - mln::accu::stat::median\_h, 523
  - mln::accu::stat::min, 526
  - mln::accu::stat::min\_h, 527
  - mln::accu::stat::min\_max, 530
  - mln::accu::stat::rank, 531
  - mln::accu::stat::rank < bool >, 533
  - mln::accu::stat::rank\_high\_quant, 534
  - mln::accu::stat::var, 536
  - mln::accu::stat::variance, 539
  - mln::accu::tuple, 541
  - mln::accu::val, 542
  - mln::morpho::attribute::card, 833
  - mln::morpho::attribute::count\_adjacent\_vertices, 834
  - mln::morpho::attribute::height, 836
  - mln::morpho::attribute::sharpness, 838
  - mln::morpho::attribute::sum, 840
  - mln::morpho::attribute::volume, 842
- to\_upper\_window
  - mln::convert, 202, 203
- to\_value
  - mln::value::proxy, 1135
- to\_vec
  - mln::algebra::h\_vec, 548
  - mln::dpoint, 648
  - mln::point, 952

- to\_win
  - mln::graph, 267
- to\_window
  - mln::convert, 203
- toggle
  - mln::p\_image, 877
- top\_hat\_black
  - mln::morpho, 366
  - mln::morpho::elementary, 369
- top\_hat\_self\_complementary
  - mln::morpho, 366
- top\_hat\_white
  - mln::morpho, 366
  - mln::morpho::elementary, 369
- topological
  - mln::morpho::watershed, 383
- tr
  - mln::tr\_image, 1033
- tr\_image
  - mln::tr\_image, 1032
- tracked\_ptr
  - mln::util::tracked\_ptr, 1094
- trait::graph, 1177
- trait::graph< mln::complex\_image< 1, G, V > >, 1178
- trait::graph< mln::image2d< T > >, 1178
- transform
  - mln::data, 213
  - mln::data::impl::generic, 221, 222
- transform\_inplace
  - mln::data, 213, 214
  - mln::data::impl::generic, 222
- transform\_inplace\_lowq
  - mln::data::impl, 219
- translate
  - mln::geom, 263, 264
- translation
  - mln::fun::x2x::translation, 711
- tree
  - mln::util::tree, 1096
- tree\_fast\_to\_image
  - mln::util, 419
- tree\_node
  - mln::util::tree\_node, 1099
- tree\_to\_fast
  - mln::util, 419
- tree\_to\_image
  - mln::util, 420
- Types, 98
- uni
  - mln::Box, 565
  - mln::Site\_Set, 969
- unique
  - mln::Box, 565
  - mln::Site\_Set, 969
- unproject\_image
  - mln::unproject\_image, 1034
- unsigned\_2complex\_image3df
  - mln, 149
- untake
  - mln::morpho::attribute::sum, 840
- up
  - mln, 163
- update
  - mln::data, 214
  - mln::data::impl::generic, 223
  - mln::dpoints\_bkd\_pixter, 651
  - mln::dpoints\_fwd\_pixter, 654
- update\_data
  - mln::labeled\_image, 792
  - mln::labeled\_image\_base, 796
- update\_fastest
  - mln::data::impl, 219
- update\_id
  - mln::util::edge, 1053
  - mln::util::vertex, 1106
- util\_set
  - mln::p\_set, 920
- util\_tree
  - mln::util::branch, 1044
- Utilities, 110
- v
  - mln::util::pix, 1081
- v1
  - mln::util::edge, 1053
  - mln::util::graph, 1064
  - mln::util::line\_graph, 1073
- v2
  - mln::util::edge, 1054
  - mln::util::graph, 1064
  - mln::util::line\_graph, 1073
- v2w2v functions, 118
- v2w\_w2v functions, 119
- v\_ith\_nbh\_edge
  - mln::util::graph, 1064
  - mln::util::line\_graph, 1073
- v\_ith\_nbh\_vertex
  - mln::util::graph, 1064
  - mln::util::line\_graph, 1073
- v\_nmax
  - mln::util::graph, 1064
  - mln::util::line\_graph, 1073
- v\_nmax\_nbh\_edges
  - mln::util::graph, 1065
  - mln::util::line\_graph, 1074
- v\_nmax\_nbh\_vertices

- mln::util::graph, 1065
- mln::util::line\_graph, 1074
- v\_other
  - mln::util::edge, 1054
- val
  - mln::doc::Generalized\_Pixel, 616
  - mln::doc::Pixel\_Iterator, 628
- value
  - mln::accu::shape::height, 506
  - mln::accu::shape::volume, 508
  - mln::complex\_image, 585
  - mln::doc::Fastest\_Image, 610
  - mln::doc::Generalized\_Pixel, 615
  - mln::doc::Image, 620
  - mln::doc::Pixel\_Iterator, 627
  - mln::doc::Value\_Iterator, 636
  - mln::doc::Value\_Set, 638
  - mln::extended, 658
  - mln::extension\_fun, 660
  - mln::extension\_ima, 663
  - mln::extension\_val, 665
  - mln::flat\_image, 670
  - mln::fun\_image, 714
  - mln::hexa, 761
  - mln::image1d, 766
  - mln::image2d, 771
  - mln::image2d\_h, 776
  - mln::image3d, 779
  - mln::interpolated, 785
  - mln::labeling, 309
  - mln::p\_vaccess, 931
  - mln::thruin\_image, 976
  - mln::tr\_image, 1032
  - mln::util::pix, 1080
  - mln::value::float01, 1111
  - mln::value::float01\_f, 1112
  - mln::value::graylevel, 1115
  - mln::value::graylevel\_f, 1118
  - mln::value::lut\_vec, 1131
  - mln::value::stack\_image, 1144
  - mln::violent\_cast\_image, 1153
- value\_and\_compute
  - mln::labeling, 309
- value\_array
  - mln::value::value\_array, 1146
- value\_ind
  - mln::value::float01, 1111
- value\_t
  - mln::util::object\_id, 1077
- values
  - mln::complex\_image, 586
  - mln::doc::Fastest\_Image, 614
  - mln::doc::Image, 621
  - mln::p\_vaccess, 932
- Values morphers, 96
- var
  - mln::accu::stat::variance, 539
- variance
  - mln::accu::stat::var, 537
- vbbox
  - mln::image1d, 769
  - mln::image3d, 782
- vec
  - mln::dpoint, 646
  - mln::make, 349, 350
  - mln::point, 950
- vec2d\_d
  - mln, 149
- vec2d\_f
  - mln, 149
- vec3d\_d
  - mln, 149
- vec3d\_f
  - mln, 149
- vect
  - mln::accu::histo, 446
- vertex
  - mln::p\_vertices, 935
  - mln::util::graph, 1065
  - mln::util::line\_graph, 1074
  - mln::util::vertex, 1104
- vertex\_fwd\_iter
  - mln::util::graph, 1061
  - mln::util::line\_graph, 1070
- vertex\_id\_t
  - mln::util, 417
- vertex\_image
  - mln::make, 350
  - mln::vertex\_image, 1152
- vertex\_nbh\_edge\_fwd\_iter
  - mln::util::graph, 1061
  - mln::util::line\_graph, 1070
- vertex\_nbh\_t
  - mln::vertex\_image, 1151
- vertex\_nbh\_vertex\_fwd\_iter
  - mln::util::graph, 1061
  - mln::util::line\_graph, 1071
- vertex\_win\_t
  - mln::vertex\_image, 1151
- vertical\_symmetry
  - mln::geom, 264
- vertices\_t
  - mln::util::graph, 1061
  - mln::util::line\_graph, 1071
- violent\_cast\_image
  - mln::violent\_cast\_image, 1154
- violet
  - mln::literal, 325



- vline2d
  - modwin2d, 114
- volume
  - mln::morpho::attribute::sharpness, 838
  - mln::win::cuboid3d, 1164
- voronoi
  - mln::make, 351
- vprod
  - mln::algebra, 174
- vset
  - mln::doc::Fastest\_Image, 610
  - mln::doc::Image, 620
  - mln::p\_vaccess, 931
  - mln::value::value\_array, 1146
- vv2b functions, 119
- w
  - mln::w\_window, 1157
- w\_window
  - mln::make, 351
  - mln::w\_window, 1156
- w\_window1d
  - mln::make, 351
- w\_window1d\_float
  - mln, 149
- w\_window1d\_int
  - mln, 149
  - mln::make, 352
- w\_window2d
  - mln::make, 352
- w\_window2d\_float
  - mln, 149
- w\_window2d\_int
  - mln, 150
  - mln::make, 353
- w\_window3d
  - mln::make, 353
- w\_window3d\_float
  - mln, 150
- w\_window3d\_int
  - mln, 150
  - mln::make, 353
- w\_window\_directional
  - mln::make, 354
- weight
  - mln::doc::Weighted\_Window, 641
  - mln::w\_window, 1156
- weights
  - mln::w\_window, 1157
- white
  - mln::literal, 325
- width
  - mln::win::cuboid3d, 1164
  - mln::win::rectangle2d, 1170
- win
  - mln::doc::Weighted\_Window, 641
  - mln::w\_window, 1157
- win\_c4p
  - modwin2d, 115
- win\_c4p\_3d
  - modwin3d, 117
- win\_c8p
  - modwin2d, 115
- win\_c8p\_3d
  - modwin3d, 117
- win\_t
  - mln::edge\_image, 657
  - mln::vertex\_image, 1151
- window
  - mln::doc::Weighted\_Window, 641
  - mln::p\_centered, 855
  - mln::window, 1173
- window1d
  - modwin1d, 113
- window2d
  - modwin2d, 114
- window3d
  - modwin3d, 117
- Windows, 111
- wrap
  - mln::data, 214
  - mln::labeling, 310
- write\_header
  - mln::io::fld, 276
- xor\_inplace
  - mln::logical, 329
- yellow
  - mln::literal, 325
- z\_order
  - mln::debug, 230
- zero
  - mln::algebra::h\_vec, 548
  - mln::literal, 325
  - mln::value::int\_s, 1120
  - mln::value::int\_u\_sat, 1125
  - mln::value::qt::rgb32, 1137
  - mln::value::rgb, 1139
  - mln::value::sign, 1142