Milena (Olena)

User documentation 1.0 Id

Generated by Doxygen 1.5.9

Tue Jul 14 18:36:50 2009

Contents

1 Documentation of milena **1**

1.1 Introduction 1

1.2 Overview of Milena. 1

1.3 Copyright and License. 2

2 Module Index **3**

2.1 Modules 3

3 Namespace Index **5**

3.1 Namespace List 5

4 Class Index **9**

4.1 Class Hierarchy 9

5 Class Index **51**

5.1 Class List 51

6 Module Documentation **61**

6.1 On site sets 61

6.1.1 Detailed Description 61

6.2 On images 62

6.2.1 Detailed Description 62

6.3 On values 63

6.3.1 Detailed Description 64

6.4 Multiple accumulators 65

6.4.1 Detailed Description 65

6.5 Graphes 66

6.5.1 Detailed Description 66

6.6 Images 67

6.6.1 Detailed Description 67

6.7 Basic types . . . 68
6.7.1 Detailed Description . . . 68
6.8 Image morphers . . . 69
6.9 Values morphers . . . 70
6.9.1 Detailed Description . . . 70
6.10 Domain morphers . . . 71
6.10.1 Detailed Description . . . 71
6.11 Identity morphers . . . 72
6.11.1 Detailed Description . . . 72
6.12 Types . . . 73
6.12.1 Detailed Description . . . 73
6.13 Accumulators . . . 74
6.13.1 Detailed Description . . . 74
6.14 Routines . . . 75
6.15 Canvas . . . 76
6.16 Functions . . . 77
6.16.1 Detailed Description . . . 78
6.17 Neighborhoods . . . 79
6.17.1 Detailed Description . . . 79
6.18 1D neighborhoods . . . 80
6.18.1 Detailed Description . . . 80
6.18.2 Typedef Documentation . . . 80
6.18.2.1 neighb1d . . . 80
6.18.3 Function Documentation . . . 80
6.18.3.1 c2 . . . 80
6.19 2D neighborhoods . . . 81
6.19.1 Detailed Description . . . 81
6.19.2 Typedef Documentation . . . 81
6.19.2.1 neighb2d . . . 81
6.19.3 Function Documentation . . . 81
6.19.3.1 c2_col . . . 81
6.19.3.2 c2_row . . . 82
6.19.3.3 c4 . . . 82
6.19.3.4 c8 . . . 82
6.20 3D neighborhoods . . . 83
6.20.1 Detailed Description . . . 83

6.20.2 Typedef Documentation . . . 83
6.20.2.1 neighb3d . . . 83
6.20.3 Function Documentation . . . 83
6.20.3.1 c18 . . . 83
6.20.3.2 c26 . . . 84
6.20.3.3 c4_3d . . . 84
6.20.3.4 c6 . . . 85
6.20.3.5 c8_3d . . . 85
6.21 Site sets . . . 86
6.21.1 Detailed Description . . . 86
6.22 Basic types . . . 87
6.22.1 Detailed Description . . . 87
6.23 Graph based . . . 88
6.23.1 Detailed Description . . . 88
6.24 Complex based . . . 89
6.24.1 Detailed Description . . . 89
6.25 Sparse types . . . 90
6.25.1 Detailed Description . . . 90
6.26 Queue based . . . 91
6.26.1 Detailed Description . . . 91
6.27 Utilities . . . 92
6.27.1 Detailed Description . . . 92
6.28 Windows . . . 93
6.28.1 Detailed Description . . . 93
6.29 1D windows . . . 94
6.29.1 Detailed Description . . . 94
6.29.2 Typedef Documentation . . . 94
6.29.2.1 segment1d . . . 94
6.29.2.2 window1d . . . 94
6.30 2D windows . . . 95
6.30.1 Detailed Description . . . 95
6.30.2 Typedef Documentation . . . 96
6.30.2.1 disk2d . . . 96
6.30.2.2 hline2d . . . 96
6.30.2.3 vline2d . . . 96
6.30.2.4 window2d . . . 96

6.30.3 Function Documentation . . . 96
6.30.3.1 win_c4p . . . 96
6.30.3.2 win_c8p . . . 97
6.31 3D windows . . . 98
6.31.1 Detailed Description . . . 98
6.31.2 Typedef Documentation . . . 98
6.31.2.1 sphere3d . . . 98
6.31.2.2 window3d . . . 98
6.31.3 Function Documentation . . . 99
6.31.3.1 win_c4p_3d . . . 99
6.31.3.2 win_c8p_3d . . . 99
6.32 N-D windows . . . 100
6.32.1 Detailed Description . . . 100
6.33 Multiple windows . . . 101
6.33.1 Detailed Description . . . 101
6.34 v2w2v functions . . . 102
6.35 v2w_w2v functions . . . 103
6.36 vv2b functions . . . 104

7 Namespace Documentation **105**
7.1 mln Namespace Reference . . . 105
7.1.1 Detailed Description . . . 125
7.1.2 Typedef Documentation . . . 127
7.1.2.1 bin_2complex_image3df . . . 127
7.1.2.2 box1d . . . 127
7.1.2.3 box2d . . . 127
7.1.2.4 box2d_h . . . 127
7.1.2.5 box3d . . . 127
7.1.2.6 discrete_plane_1complex_geometry . . . 127
7.1.2.7 discrete_plane_2complex_geometry . . . 127
7.1.2.8 dpoint1d . . . 128
7.1.2.9 dpoint2d . . . 128
7.1.2.10 dpoint2d_h . . . 128
7.1.2.11 dpoint3d . . . 128
7.1.2.12 float_2complex_image3df . . . 128
7.1.2.13 int_u8_1complex_image2d . . . 128
7.1.2.14 int_u8_2complex_image2d . . . 128

7.1.2.15 int_u8_2complex_image3df . . . 128
7.1.2.16 p_run2d . . . 128
7.1.2.17 p_runs2d . . . 128
7.1.2.18 point1d . . . 129
7.1.2.19 point1df . . . 129
7.1.2.20 point2d . . . 129
7.1.2.21 point2d_h . . . 129
7.1.2.22 point2df . . . 129
7.1.2.23 point3d . . . 129
7.1.2.24 point3df . . . 129
7.1.2.25 rgb8_2complex_image3df . . . 129
7.1.2.26 space_2complex_geometry . . . 129
7.1.2.27 vec2d_d . . . 129
7.1.2.28 vec2d_f . . . 129
7.1.2.29 vec3d_d . . . 130
7.1.2.30 vec3d_f . . . 130
7.1.3 Function Documentation . . . 130
7.1.3.1 a_point_of . . . 130
7.1.3.2 apply_p2p . . . 130
7.1.3.3 apply_p2p . . . 130
7.1.3.4 compose . . . 130
7.1.3.5 duplicate . . . 130
7.1.3.6 extend . . . 131
7.1.3.7 extend . . . 131
7.1.3.8 extend . . . 131
7.1.3.9 implies . . . 131
7.1.3.10 initialize . . . 131
7.1.3.11 is_simple_2d . . . 132
7.1.3.12 larger_than . . . 132
7.1.3.13 make_debug_graph_image . . . 132
7.1.3.14 mln_exact . . . 132
7.1.3.15 mln_gen_complex_neighborhood . . . 132
7.1.3.16 mln_gen_complex_neighborhood . . . 133
7.1.3.17 mln_gen_complex_neighborhood . . . 133
7.1.3.18 mln_gen_complex_neighborhood . . . 133
7.1.3.19 mln_gen_complex_neighborhood . . . 133

7.1.3.20 mln_gen_complex_neighborhood . . . 133
7.1.3.21 mln_gen_complex_window . . . 133
7.1.3.22 mln_gen_complex_window . . . 133
7.1.3.23 mln_gen_complex_window . . . 133
7.1.3.24 mln_gen_complex_window . . . 134
7.1.3.25 mln_gen_complex_window . . . 134
7.1.3.26 mln_gen_complex_window . . . 134
7.1.3.27 mln_regular . . . 134
7.1.3.28 mln_trait_op_geq . . . 134
7.1.3.29 mln_trait_op_greater . . . 134
7.1.3.30 mln_trait_op_leq . . . 135
7.1.3.31 mln_trait_op_neq . . . 135
7.1.3.32 operator!= . . . 135
7.1.3.33 operator!= . . . 135
7.1.3.34 operator∗ . . . 136
7.1.3.35 operator++ . . . 136
7.1.3.36 operator- . . . 136
7.1.3.37 operator- . . . 136
7.1.3.38 operator– . . . 136
7.1.3.39 operator< . . . 137
7.1.3.40 operator< . . . 137
7.1.3.41 operator< . . . 137
7.1.3.42 operator<< . . . 137
7.1.3.43 operator<< . . . 137
7.1.3.44 operator<< . . . 137
7.1.3.45 operator<< . . . 138
7.1.3.46 operator<= . . . 138
7.1.3.47 operator<= . . . 138
7.1.3.48 operator<= . . . 138
7.1.3.49 operator<= . . . 138
7.1.3.50 operator<= . . . 138
7.1.3.51 operator== . . . 139
7.1.3.52 operator== . . . 139
7.1.3.53 operator== . . . 139
7.1.3.54 operator== . . . 139
7.1.3.55 operator== . . . 139

7.1.3.56 operator== . . . 139
7.1.3.57 operator== . . . 140
7.1.3.58 operator| . . . 140
7.1.3.59 operator| . . . 140
7.1.3.60 operator| . . . 140
7.1.3.61 operator| . . . 140
7.1.3.62 operator| . . . 141
7.1.3.63 operator| . . . 141
7.1.3.64 primary . . . 141
7.1.3.65 ptransform . . . 141
7.1.4 Variable Documentation . . . 141
7.1.4.1 before . . . 141
7.1.4.2 sagittal_dec . . . 141
7.1.4.3 up . . . 141
7.2 mln::accu Namespace Reference . . . 142
7.2.1 Detailed Description . . . 143
7.2.2 Function Documentation . . . 144
7.2.2.1 compute . . . 144
7.2.2.2 line . . . 144
7.2.2.3 mln_meta_accu_result . . . 144
7.2.2.4 take . . . 145
7.3 mln::accu::image Namespace Reference . . . 146
7.3.1 Detailed Description . . . 146
7.4 mln::accu::impl Namespace Reference . . . 147
7.4.1 Detailed Description . . . 147
7.5 mln::accu::logic Namespace Reference . . . 148
7.5.1 Detailed Description . . . 148
7.6 mln::accu::math Namespace Reference . . . 149
7.6.1 Detailed Description . . . 149
7.7 mln::accu::meta::logic Namespace Reference . . . 150
7.7.1 Detailed Description . . . 150
7.8 mln::accu::meta::math Namespace Reference . . . 151
7.8.1 Detailed Description . . . 151
7.9 mln::accu::meta::shape Namespace Reference . . . 152
7.9.1 Detailed Description . . . 152
7.10 mln::accu::meta::stat Namespace Reference . . . 153

7.10.1 Detailed Description . . . 153
7.11 mln::accu::shape Namespace Reference . . . 154
7.11.1 Detailed Description . . . 154
7.12 mln::accu::stat Namespace Reference . . . 155
7.12.1 Detailed Description . . . 156
7.13 mln::algebra Namespace Reference . . . 157
7.13.1 Detailed Description . . . 157
7.13.2 Function Documentation . . . 157
7.13.2.1 ldlt_decomp . . . 157
7.13.2.2 ldlt_solve . . . 158
7.13.2.3 operator∗ . . . 158
7.13.2.4 vprod . . . 158
7.14 mln::arith Namespace Reference . . . 159
7.14.1 Detailed Description . . . 161
7.14.2 Function Documentation . . . 161
7.14.2.1 diff_abs . . . 161
7.14.2.2 div . . . 161
7.14.2.3 div_cst . . . 162
7.14.2.4 div_inplace . . . 162
7.14.2.5 min . . . 162
7.14.2.6 min_inplace . . . 163
7.14.2.7 minus . . . 163
7.14.2.8 minus . . . 163
7.14.2.9 minus_cst . . . 164
7.14.2.10 minus_cst . . . 164
7.14.2.11 minus_cst_inplace . . . 165
7.14.2.12 minus_inplace . . . 165
7.14.2.13 plus . . . 165
7.14.2.14 plus . . . 166
7.14.2.15 plus_cst . . . 167
7.14.2.16 plus_cst . . . 167
7.14.2.17 plus_cst_inplace . . . 167
7.14.2.18 plus_inplace . . . 168
7.14.2.19 revert . . . 168
7.14.2.20 revert_inplace . . . 168
7.14.2.21 times . . . 169

7.14.2.22 times_cst . . . 169
7.14.2.23 times_inplace . . . 169
7.15 mln::arith::impl Namespace Reference . . . 171
7.15.1 Detailed Description . . . 171
7.16 mln::arith::impl::generic Namespace Reference . . . 172
7.16.1 Detailed Description . . . 172
7.17 mln::binarization Namespace Reference . . . 173
7.17.1 Detailed Description . . . 173
7.17.2 Function Documentation . . . 173
7.17.2.1 binarization . . . 173
7.17.2.2 threshold . . . 173
7.18 mln::border Namespace Reference . . . 174
7.18.1 Detailed Description . . . 174
7.18.2 Function Documentation . . . 174
7.18.2.1 adjust . . . 174
7.18.2.2 duplicate . . . 175
7.18.2.3 equalize . . . 175
7.18.2.4 fill . . . 175
7.18.2.5 find . . . 176
7.18.2.6 get . . . 176
7.18.2.7 mirror . . . 176
7.18.2.8 resize . . . 177
7.19 mln::border::impl Namespace Reference . . . 178
7.19.1 Detailed Description . . . 178
7.20 mln::border::impl::generic Namespace Reference . . . 179
7.20.1 Detailed Description . . . 179
7.21 mln::canvas Namespace Reference . . . 180
7.21.1 Detailed Description . . . 180
7.21.2 Function Documentation . . . 180
7.21.2.1 distance_front . . . 180
7.21.2.2 distance_geodesic . . . 181
7.22 mln::canvas::browsing Namespace Reference . . . 182
7.22.1 Detailed Description . . . 182
7.23 mln::canvas::impl Namespace Reference . . . 183
7.23.1 Detailed Description . . . 183
7.24 mln::canvas::morpho Namespace Reference . . . 184

7.24.1 Detailed Description . . . 184
7.25 mln::convert Namespace Reference . . . 185
7.25.1 Detailed Description . . . 187
7.25.2 Function Documentation . . . 187
7.25.2.1 from_to . . . 187
7.25.2.2 from_to . . . 187
7.25.2.3 from_to . . . 187
7.25.2.4 from_to . . . 187
7.25.2.5 mln_image_from_grid . . . 187
7.25.2.6 mln_image_from_grid . . . 188
7.25.2.7 mln_image_from_grid . . . 188
7.25.2.8 mln_image_from_grid . . . 188
7.25.2.9 mln_window . . . 188
7.25.2.10 to . . . 188
7.25.2.11 to_dpoint . . . 188
7.25.2.12 to_fun . . . 188
7.25.2.13 to_fun . . . 188
7.25.2.14 to_image . . . 188
7.25.2.15 to_p_array . . . 189
7.25.2.16 to_p_array . . . 189
7.25.2.17 to_p_array . . . 189
7.25.2.18 to_p_set . . . 189
7.25.2.19 to_p_set . . . 189
7.25.2.20 to_p_set . . . 189
7.25.2.21 to_p_set . . . 189
7.25.2.22 to_p_set . . . 189
7.25.2.23 to_upper_window . . . 190
7.25.2.24 to_upper_window . . . 190
7.25.2.25 to_window . . . 190
7.25.2.26 to_window . . . 190
7.25.2.27 to_window . . . 190
7.26 mln::data Namespace Reference . . . 191
7.26.1 Detailed Description . . . 193
7.26.2 Function Documentation . . . 193
7.26.2.1 abs . . . 193
7.26.2.2 abs_inplace . . . 193

7.26.2.3 apply . . . 193
7.26.2.4 compute . . . 194
7.26.2.5 convert . . . 194
7.26.2.6 fast_median . . . 194
7.26.2.7 fill . . . 195
7.26.2.8 fill_with_image . . . 195
7.26.2.9 fill_with_value . . . 196
7.26.2.10 median . . . 196
7.26.2.11 mln_meta_accu_result . . . 196
7.26.2.12 paste . . . 197
7.26.2.13 paste_without_localization . . . 197
7.26.2.14 replace . . . 197
7.26.2.15 saturate . . . 198
7.26.2.16 saturate . . . 198
7.26.2.17 saturate_inplace . . . 198
7.26.2.18 sort_offsets_increasing . . . 198
7.26.2.19 sort_psites_decreasing . . . 199
7.26.2.20 sort_psites_increasing . . . 199
7.26.2.21 stretch . . . 199
7.26.2.22 to_enc . . . 200
7.26.2.23 transform . . . 200
7.26.2.24 transform . . . 200
7.26.2.25 transform_inplace . . . 201
7.26.2.26 transform_inplace . . . 201
7.26.2.27 update . . . 202
7.27 mln::data::approx Namespace Reference . . . 203
7.27.1 Detailed Description . . . 203
7.27.2 Function Documentation . . . 203
7.27.2.1 median . . . 203
7.27.2.2 median . . . 203
7.27.2.3 median . . . 204
7.28 mln::data::approx::impl Namespace Reference . . . 205
7.28.1 Detailed Description . . . 205
7.29 mln::data::impl Namespace Reference . . . 206
7.29.1 Detailed Description . . . 206
7.29.2 Function Documentation . . . 206

7.29.2.1 stretch . . . 206
7.29.2.2 transform_inplace_lowq . . . 207
7.29.2.3 update_fastest . . . 207
7.30 mln::data::impl::generic Namespace Reference . . . 208
7.30.1 Detailed Description . . . 209
7.30.2 Function Documentation . . . 209
7.30.2.1 convert . . . 209
7.30.2.2 fill_with_image . . . 209
7.30.2.3 fill_with_value . . . 209
7.30.2.4 median . . . 210
7.30.2.5 paste . . . 210
7.30.2.6 sort_offsets_increasing . . . 210
7.30.2.7 transform . . . 210
7.30.2.8 transform . . . 211
7.30.2.9 transform_inplace . . . 211
7.30.2.10 transform_inplace . . . 211
7.30.2.11 update . . . 211
7.31 mln::data::naive Namespace Reference . . . 213
7.31.1 Detailed Description . . . 213
7.31.2 Function Documentation . . . 213
7.31.2.1 median . . . 213
7.32 mln::data::naive::impl Namespace Reference . . . 214
7.32.1 Detailed Description . . . 214
7.33 mln::debug Namespace Reference . . . 215
7.33.1 Detailed Description . . . 216
7.33.2 Function Documentation . . . 216
7.33.2.1 draw_graph . . . 216
7.33.2.2 draw_graph . . . 217
7.33.2.3 draw_graph . . . 217
7.33.2.4 filename . . . 217
7.33.2.5 format . . . 217
7.33.2.6 format . . . 217
7.33.2.7 format . . . 217
7.33.2.8 format . . . 218
7.33.2.9 iota . . . 218
7.33.2.10 println . . . 218

7.33.2.11 println . . . 218
7.33.2.12 println_with_border . . . 218
7.33.2.13 put_word . . . 218
7.33.2.14 slices_2d . . . 218
7.33.2.15 slices_2d . . . 219
7.33.2.16 superpose . . . 219
7.33.2.17 superpose . . . 219
7.34 mln::debug::impl Namespace Reference . . . 220
7.34.1 Detailed Description . . . 220
7.35 mln::def Namespace Reference . . . 221
7.35.1 Detailed Description . . . 221
7.35.2 Typedef Documentation . . . 221
7.35.2.1 coord . . . 221
7.35.2.2 coordf . . . 221
7.35.3 Enumeration Type Documentation . . . 221
7.35.3.1 "@20 . . . 221
7.36 mln::display Namespace Reference . . . 222
7.36.1 Detailed Description . . . 222
7.37 mln::display::impl Namespace Reference . . . 223
7.37.1 Detailed Description . . . 223
7.38 mln::display::impl::generic Namespace Reference . . . 224
7.38.1 Detailed Description . . . 224
7.39 mln::doc Namespace Reference . . . 225
7.39.1 Detailed Description . . . 226
7.40 mln::draw Namespace Reference . . . 227
7.40.1 Detailed Description . . . 227
7.40.2 Function Documentation . . . 227
7.40.2.1 box . . . 227
7.40.2.2 line . . . 227
7.40.2.3 plot . . . 228
7.41 mln::estim Namespace Reference . . . 229
7.41.1 Detailed Description . . . 229
7.41.2 Function Documentation . . . 229
7.41.2.1 mean . . . 229
7.41.2.2 mean . . . 230
7.41.2.3 min_max . . . 230

7.41.2.4 sum . . . 230
7.41.2.5 sum . . . 230
7.42 mln::extension Namespace Reference . . . 231
7.42.1 Detailed Description . . . 231
7.42.2 Function Documentation . . . 231
7.42.2.1 adjust . . . 231
7.42.2.2 adjust . . . 232
7.42.2.3 adjust . . . 232
7.42.2.4 adjust . . . 232
7.42.2.5 adjust_duplicate . . . 232
7.42.2.6 adjust_fill . . . 232
7.42.2.7 duplicate . . . 232
7.42.2.8 fill . . . 232
7.43 mln::fun Namespace Reference . . . 234
7.43.1 Detailed Description . . . 235
7.44 mln::fun::access Namespace Reference . . . 236
7.44.1 Detailed Description . . . 236
7.45 mln::fun::i2v Namespace Reference . . . 237
7.45.1 Detailed Description . . . 237
7.46 mln::fun::p2b Namespace Reference . . . 238
7.46.1 Detailed Description . . . 238
7.47 mln::fun::p2p Namespace Reference . . . 239
7.47.1 Detailed Description . . . 239
7.48 mln::fun::p2v Namespace Reference . . . 240
7.48.1 Detailed Description . . . 240
7.49 mln::fun::stat Namespace Reference . . . 241
7.49.1 Detailed Description . . . 241
7.50 mln::fun::v2b Namespace Reference . . . 242
7.50.1 Detailed Description . . . 242
7.51 mln::fun::v2i Namespace Reference . . . 243
7.51.1 Detailed Description . . . 243
7.52 mln::fun::v2v Namespace Reference . . . 244
7.52.1 Detailed Description . . . 244
7.52.2 Variable Documentation . . . 245
7.52.2.1 f_hsi_to_rgb_3x8 . . . 245
7.52.2.2 f_hsl_to_rgb_3x8 . . . 245

7.52.2.3 f_rgb_to_hsi_f . . . 245
7.52.2.4 f_rgb_to_hsl_f . . . 245
7.53 mln::fun::v2w2v Namespace Reference . . . 246
7.53.1 Detailed Description . . . 246
7.54 mln::fun::v2w_w2v Namespace Reference . . . 247
7.54.1 Detailed Description . . . 247
7.55 mln::fun::vv2b Namespace Reference . . . 248
7.55.1 Detailed Description . . . 248
7.56 mln::fun::vv2v Namespace Reference . . . 249
7.56.1 Detailed Description . . . 249
7.57 mln::fun::x2p Namespace Reference . . . 250
7.57.1 Detailed Description . . . 250
7.58 mln::fun::x2v Namespace Reference . . . 251
7.58.1 Detailed Description . . . 251
7.59 mln::fun::x2x Namespace Reference . . . 252
7.59.1 Detailed Description . . . 252
7.60 mln::geom Namespace Reference . . . 253
7.60.1 Detailed Description . . . 256
7.60.2 Function Documentation . . . 256
7.60.2.1 bbox . . . 256
7.60.2.2 bbox . . . 256
7.60.2.3 bbox . . . 257
7.60.2.4 bbox . . . 257
7.60.2.5 chamfer . . . 257
7.60.2.6 delta . . . 257
7.60.2.7 delta . . . 257
7.60.2.8 delta . . . 257
7.60.2.9 max_col . . . 257
7.60.2.10 max_col . . . 258
7.60.2.11 max_ind . . . 258
7.60.2.12 max_row . . . 258
7.60.2.13 max_row . . . 258
7.60.2.14 max_sli . . . 258
7.60.2.15 mesh_corner_point_area . . . 258
7.60.2.16 mesh_curvature . . . 259
7.60.2.17 mesh_normal . . . 259

7.60.2.18 min_col . . . 259
7.60.2.19 min_col . . . 259
7.60.2.20 min_ind . . . 260
7.60.2.21 min_row . . . 260
7.60.2.22 min_row . . . 260
7.60.2.23 min_sli . . . 260
7.60.2.24 ncols . . . 260
7.60.2.25 ncols . . . 260
7.60.2.26 ninds . . . 260
7.60.2.27 nrows . . . 261
7.60.2.28 nrows . . . 261
7.60.2.29 nsites . . . 261
7.60.2.30 nslis . . . 261
7.60.2.31 pmin_pmax . . . 261
7.60.2.32 pmin_pmax . . . 261
7.60.2.33 pmin_pmax . . . 261
7.60.2.34 pmin_pmax . . . 262
7.60.2.35 rotate . . . 262
7.60.2.36 rotate . . . 262
7.60.2.37 rotate . . . 262
7.60.2.38 seeds2tiling . . . 262
7.60.2.39 seeds2tiling_roundness . . . 263
7.61 mln::geom::impl Namespace Reference . . . 264
7.61.1 Detailed Description . . . 264
7.61.2 Function Documentation . . . 264
7.61.2.1 seeds2tiling . . . 264
7.61.2.2 seeds2tiling_roundness . . . 264
7.62 mln::graph Namespace Reference . . . 266
7.62.1 Detailed Description . . . 266
7.62.2 Function Documentation . . . 266
7.62.2.1 compute . . . 266
7.62.2.2 labeling . . . 267
7.62.2.3 to_neighb . . . 267
7.62.2.4 to_win . . . 267
7.63 mln::grid Namespace Reference . . . 269
7.63.1 Detailed Description . . . 269

7.64 mln::histo Namespace Reference . . . 270
7.64.1 Detailed Description . . . 270
7.64.2 Function Documentation . . . 270
7.64.2.1 compute . . . 270
7.65 mln::histo::impl Namespace Reference . . . 271
7.65.1 Detailed Description . . . 271
7.66 mln::histo::impl::generic Namespace Reference . . . 272
7.66.1 Detailed Description . . . 272
7.67 mln::impl Namespace Reference . . . 273
7.67.1 Detailed Description . . . 273
7.68 mln::io Namespace Reference . . . 274
7.68.1 Detailed Description . . . 275
7.69 mln::io::cloud Namespace Reference . . . 276
7.69.1 Detailed Description . . . 276
7.69.2 Function Documentation . . . 276
7.69.2.1 load . . . 276
7.69.2.2 save . . . 276
7.70 mln::io::dicom Namespace Reference . . . 277
7.70.1 Detailed Description . . . 277
7.70.2 Function Documentation . . . 277
7.70.2.1 load . . . 277
7.70.2.2 load . . . 277
7.71 mln::io::dump Namespace Reference . . . 278
7.71.1 Detailed Description . . . 278
7.71.2 Function Documentation . . . 278
7.71.2.1 load . . . 278
7.71.2.2 save . . . 278
7.72 mln::io::fits Namespace Reference . . . 279
7.72.1 Detailed Description . . . 279
7.72.2 Function Documentation . . . 279
7.72.2.1 load . . . 279
7.72.2.2 load . . . 279
7.73 mln::io::magick Namespace Reference . . . 280
7.73.1 Detailed Description . . . 280
7.73.2 Function Documentation . . . 280
7.73.2.1 load . . . 280

7.73.2.2 save . . . 280
7.74 mln::io::off Namespace Reference . . . 281
7.74.1 Detailed Description . . . 281
7.74.2 Function Documentation . . . 281
7.74.2.1 load . . . 281
7.74.2.2 save . . . 281
7.74.2.3 save_bin_alt . . . 282
7.75 mln::io::pbm Namespace Reference . . . 283
7.75.1 Detailed Description . . . 283
7.75.2 Function Documentation . . . 283
7.75.2.1 load . . . 283
7.75.2.2 load . . . 284
7.75.2.3 save . . . 284
7.76 mln::io::pbm::impl Namespace Reference . . . 285
7.76.1 Detailed Description . . . 285
7.77 mln::io::pbms Namespace Reference . . . 286
7.77.1 Detailed Description . . . 286
7.77.2 Function Documentation . . . 286
7.77.2.1 load . . . 286
7.78 mln::io::pbms::impl Namespace Reference . . . 287
7.78.1 Detailed Description . . . 287
7.79 mln::io::pfm Namespace Reference . . . 288
7.79.1 Detailed Description . . . 288
7.79.2 Function Documentation . . . 288
7.79.2.1 load . . . 288
7.79.2.2 load . . . 289
7.79.2.3 save . . . 289
7.80 mln::io::pfm::impl Namespace Reference . . . 290
7.80.1 Detailed Description . . . 290
7.81 mln::io::pgm Namespace Reference . . . 291
7.81.1 Detailed Description . . . 291
7.81.2 Function Documentation . . . 291
7.81.2.1 load . . . 291
7.81.2.2 load . . . 292
7.81.2.3 save . . . 292
7.82 mln::io::pgms Namespace Reference . . . 293

7.82.1 Detailed Description . . . 293
7.82.2 Function Documentation . . . 293
7.82.2.1 load . . . 293
7.83 mln::io::plot Namespace Reference . . . 294
7.83.1 Detailed Description . . . 294
7.83.2 Function Documentation . . . 294
7.83.2.1 load . . . 294
7.83.2.2 save . . . 294
7.83.2.3 save . . . 295
7.84 mln::io::pnm Namespace Reference . . . 296
7.84.1 Detailed Description . . . 296
7.84.2 Function Documentation . . . 296
7.84.2.1 load . . . 296
7.84.2.2 load . . . 296
7.84.2.3 load_raw_2d . . . 297
7.84.2.4 max_component . . . 297
7.84.2.5 save . . . 297
7.85 mln::io::pnm::impl Namespace Reference . . . 298
7.85.1 Detailed Description . . . 298
7.86 mln::io::pnms Namespace Reference . . . 299
7.86.1 Detailed Description . . . 299
7.86.2 Function Documentation . . . 299
7.86.2.1 load . . . 299
7.86.2.2 load . . . 299
7.87 mln::io::ppm Namespace Reference . . . 300
7.87.1 Detailed Description . . . 300
7.87.2 Function Documentation . . . 300
7.87.2.1 load . . . 300
7.87.2.2 load . . . 301
7.87.2.3 save . . . 301
7.88 mln::io::ppms Namespace Reference . . . 302
7.88.1 Detailed Description . . . 302
7.88.2 Function Documentation . . . 302
7.88.2.1 load . . . 302
7.89 mln::io::tiff Namespace Reference . . . 303
7.89.1 Detailed Description . . . 303

7.89.2 Function Documentation . . . 303
7.89.2.1 load . . . 303
7.90 mln::io::txt Namespace Reference . . . 304
7.90.1 Detailed Description . . . 304
7.90.2 Function Documentation . . . 304
7.90.2.1 save . . . 304
7.91 mln::labeling Namespace Reference . . . 305
7.91.1 Detailed Description . . . 307
7.91.2 Function Documentation . . . 307
7.91.2.1 background . . . 307
7.91.2.2 blobs . . . 308
7.91.2.3 colorize . . . 308
7.91.2.4 colorize . . . 308
7.91.2.5 colorize . . . 309
7.91.2.6 compute . . . 309
7.91.2.7 compute . . . 309
7.91.2.8 compute . . . 310
7.91.2.9 compute . . . 310
7.91.2.10 compute . . . 311
7.91.2.11 compute_image . . . 312
7.91.2.12 compute_image . . . 312
7.91.2.13 compute_image . . . 312
7.91.2.14 fill_holes . . . 313
7.91.2.15 flat_zones . . . 313
7.91.2.16 foreground . . . 314
7.91.2.17 pack . . . 314
7.91.2.18 pack_inplace . . . 314
7.91.2.19 regional_maxima . . . 315
7.91.2.20 regional_minima . . . 315
7.91.2.21 relabel . . . 315
7.91.2.22 relabel . . . 316
7.91.2.23 relabel_inplace . . . 316
7.91.2.24 relabel_inplace . . . 316
7.91.2.25 value . . . 317
7.91.2.26 wrap . . . 317
7.91.2.27 wrap . . . 317

7.92 mln::labeling::impl Namespace Reference . . . 318
7.92.1 Detailed Description . . . 318
7.93 mln::labeling::impl::generic Namespace Reference . . . 319
7.93.1 Detailed Description . . . 319
7.93.2 Function Documentation . . . 319
7.93.2.1 compute . . . 319
7.93.2.2 compute . . . 320
7.93.2.3 compute . . . 320
7.94 mln::linear Namespace Reference . . . 321
7.94.1 Detailed Description . . . 321
7.94.2 Function Documentation . . . 322
7.94.2.1 gaussian . . . 322
7.94.2.2 gaussian . . . 322
7.94.2.3 gaussian_1st_derivative . . . 322
7.94.2.4 gaussian_1st_derivative . . . 322
7.94.2.5 gaussian_2nd_derivative . . . 323
7.94.2.6 gaussian_2nd_derivative . . . 323
7.94.2.7 mln_ch_convolve . . . 323
7.94.2.8 mln_ch_convolve . . . 323
7.94.2.9 mln_ch_convolve_grad . . . 324
7.95 mln::linear::impl Namespace Reference . . . 325
7.95.1 Detailed Description . . . 325
7.96 mln::linear::local Namespace Reference . . . 326
7.96.1 Detailed Description . . . 326
7.96.2 Function Documentation . . . 326
7.96.2.1 convolve . . . 326
7.96.2.2 convolve . . . 326
7.97 mln::linear::local::impl Namespace Reference . . . 327
7.97.1 Detailed Description . . . 327
7.98 mln::literal Namespace Reference . . . 328
7.98.1 Detailed Description . . . 331
7.98.2 Variable Documentation . . . 331
7.98.2.1 black . . . 331
7.98.2.2 blue . . . 331
7.98.2.3 brown . . . 331
7.98.2.4 cyan . . . 331

7.98.2.5 dark_gray . . . 331
7.98.2.6 green . . . 331
7.98.2.7 identity . . . 331
7.98.2.8 light_gray . . . 331
7.98.2.9 lime . . . 332
7.98.2.10 magenta . . . 332
7.98.2.11 max . . . 332
7.98.2.12 medium_gray . . . 332
7.98.2.13 min . . . 332
7.98.2.14 olive . . . 332
7.98.2.15 one . . . 332
7.98.2.16 orange . . . 332
7.98.2.17 origin . . . 332
7.98.2.18 pink . . . 332
7.98.2.19 purple . . . 332
7.98.2.20 red . . . 333
7.98.2.21 teal . . . 333
7.98.2.22 violet . . . 333
7.98.2.23 white . . . 333
7.98.2.24 yellow . . . 333
7.98.2.25 zero . . . 333
7.99 mln::logical Namespace Reference . . . 334
7.99.1 Detailed Description . . . 334
7.99.2 Function Documentation . . . 334
7.99.2.1 and_inplace . . . 334
7.99.2.2 and_not . . . 335
7.99.2.3 and_not_inplace . . . 335
7.99.2.4 not_inplace . . . 335
7.99.2.5 or_inplace . . . 336
7.99.2.6 xor_inplace . . . 336
7.100 mln::logical::impl Namespace Reference . . . 337
7.100.1 Detailed Description . . . 337
7.101 mln::logical::impl::generic Namespace Reference . . . 338
7.101.1 Detailed Description . . . 338
7.102 mln::make Namespace Reference . . . 339
7.102.1 Detailed Description . . . 343

7.102.2 Function Documentation . . . 343
7.102.2.1 attachment . . . 343
7.102.2.2 box1d . . . 344
7.102.2.3 box1d . . . 344
7.102.2.4 box2d . . . 344
7.102.2.5 box2d . . . 345
7.102.2.6 box2d_h . . . 345
7.102.2.7 box2d_h . . . 346
7.102.2.8 box3d . . . 346
7.102.2.9 box3d . . . 346
7.102.2.10 cell . . . 347
7.102.2.11 couple . . . 347
7.102.2.12 detachment . . . 347
7.102.2.13 dpoint2d_h . . . 348
7.102.2.14 dummy_p_edges . . . 348
7.102.2.15 dummy_p_edges . . . 348
7.102.2.16 dummy_p_vertices . . . 349
7.102.2.17 dummy_p_vertices . . . 349
7.102.2.18 edge_image . . . 349
7.102.2.19 edge_image . . . 349
7.102.2.20 edge_image . . . 350
7.102.2.21 edge_image . . . 350
7.102.2.22 h_mat . . . 350
7.102.2.23 image . . . 351
7.102.2.24 image . . . 351
7.102.2.25 image . . . 351
7.102.2.26 image2d . . . 351
7.102.2.27 image3d . . . 352
7.102.2.28 image3d . . . 352
7.102.2.29 influence_zone_adjacency_graph . . . 352
7.102.2.30 mat . . . 352
7.102.2.31 ord_pair . . . 353
7.102.2.32 p_edges_with_mass_centers . . . 353
7.102.2.33 p_vertices_with_mass_centers . . . 353
7.102.2.34 pix . . . 354
7.102.2.35 pixel . . . 354

7.102.2.36 pixel . . . 354
7.102.2.37 point2d_h . . . 354
7.102.2.38 rag_and_labeled_wsl . . . 354
7.102.2.39 region_adjacency_graph . . . 355
7.102.2.40 relabelfun . . . 355
7.102.2.41 relabelfun . . . 356
7.102.2.42 vec . . . 356
7.102.2.43 vec . . . 357
7.102.2.44 vec . . . 357
7.102.2.45 vec . . . 357
7.102.2.46 vertex_image . . . 357
7.102.2.47 vertex_image . . . 358
7.102.2.48 voronoi . . . 358
7.102.2.49 w_window . . . 358
7.102.2.50 w_window1d . . . 359
7.102.2.51 w_window2d . . . 359
7.102.2.52 w_window3d . . . 359
7.102.2.53 w_window_directional . . . 360
7.103 mln::math Namespace Reference . . . 361
7.103.1 Detailed Description . . . 361
7.103.2 Function Documentation . . . 361
7.103.2.1 abs . . . 361
7.103.2.2 abs . . . 361
7.103.2.3 abs . . . 361
7.104 mln::metal Namespace Reference . . . 362
7.104.1 Detailed Description . . . 362
7.105 mln::metal::impl Namespace Reference . . . 363
7.105.1 Detailed Description . . . 363
7.106 mln::metal::math Namespace Reference . . . 364
7.106.1 Detailed Description . . . 364
7.107 mln::metal::math::impl Namespace Reference . . . 365
7.107.1 Detailed Description . . . 365
7.108 mln::morpho Namespace Reference . . . 366
7.108.1 Detailed Description . . . 368
7.108.2 Function Documentation . . . 369
7.108.2.1 complementation . . . 369

7.108.2.2 complementation_inplace . . . 369
7.108.2.3 contrast . . . 369
7.108.2.4 dilation . . . 369
7.108.2.5 erosion . . . 369
7.108.2.6 general . . . 370
7.108.2.7 gradient . . . 370
7.108.2.8 gradient_external . . . 370
7.108.2.9 gradient_internal . . . 370
7.108.2.10 hit_or_miss . . . 370
7.108.2.11 hit_or_miss_background_closing . . . 370
7.108.2.12 hit_or_miss_background_opening . . . 371
7.108.2.13 hit_or_miss_closing . . . 371
7.108.2.14 hit_or_miss_opening . . . 371
7.108.2.15 laplacian . . . 371
7.108.2.16 line_gradient . . . 371
7.108.2.17 meyer_wst . . . 372
7.108.2.18 meyer_wst . . . 372
7.108.2.19 min . . . 372
7.108.2.20 min_inplace . . . 372
7.108.2.21 minus . . . 373
7.108.2.22 plus . . . 373
7.108.2.23 rank_filter . . . 373
7.108.2.24 thick_miss . . . 373
7.108.2.25 thickening . . . 373
7.108.2.26 thin_fit . . . 374
7.108.2.27 thinning . . . 374
7.108.2.28 top_hat_black . . . 374
7.108.2.29 top_hat_self_complementary . . . 374
7.108.2.30 top_hat_white . . . 374
7.109 mln::morpho::approx Namespace Reference . . . 375
7.109.1 Detailed Description . . . 375
7.110 mln::morpho::attribute Namespace Reference . . . 376
7.110.1 Detailed Description . . . 376
7.111 mln::morpho::closing::approx Namespace Reference . . . 377
7.111.1 Detailed Description . . . 377
7.111.2 Function Documentation . . . 377

7.111.2.1 structural . . . 377
7.112 mln::morpho::elementary Namespace Reference . . . 378
7.112.1 Detailed Description . . . 378
7.112.2 Function Documentation . . . 378
7.112.2.1 closing . . . 378
7.112.2.2 mln_trait_op_minus_twice . . . 379
7.112.2.3 opening . . . 379
7.112.2.4 top_hat_black . . . 379
7.112.2.5 top_hat_self_complementary . . . 379
7.112.2.6 top_hat_white . . . 379
7.113 mln::morpho::impl Namespace Reference . . . 380
7.113.1 Detailed Description . . . 380
7.114 mln::morpho::impl::generic Namespace Reference . . . 381
7.114.1 Detailed Description . . . 381
7.114.2 Function Documentation . . . 381
7.114.2.1 hit_or_miss . . . 381
7.114.2.2 rank_filter . . . 381
7.115 mln::morpho::opening::approx Namespace Reference . . . 382
7.115.1 Detailed Description . . . 382
7.115.2 Function Documentation . . . 382
7.115.2.1 structural . . . 382
7.116 mln::morpho::reconstruction Namespace Reference . . . 383
7.116.1 Detailed Description . . . 383
7.117 mln::morpho::reconstruction::by_dilation Namespace Reference . . . 384
7.117.1 Detailed Description . . . 384
7.118 mln::morpho::reconstruction::by_erosion Namespace Reference . . . 385
7.118.1 Detailed Description . . . 385
7.119 mln::morpho::tree Namespace Reference . . . 386
7.119.1 Detailed Description . . . 386
7.119.2 Function Documentation . . . 387
7.119.2.1 compute_attribute_image_from . . . 387
7.119.2.2 compute_parent . . . 387
7.119.2.3 propagate_if . . . 388
7.119.2.4 propagate_if . . . 388
7.119.2.5 propagate_if_value . . . 389
7.119.2.6 propagate_if_value . . . 389

7.119.2.7 propagate_node_to_ancestors . . . 389
7.119.2.8 propagate_node_to_ancestors . . . 389
7.119.2.9 propagate_node_to_descendants . . . 390
7.119.2.10 propagate_representative . . . 390
7.120 mln::morpho::tree::filter Namespace Reference . . . 391
7.120.1 Detailed Description . . . 391
7.120.2 Function Documentation . . . 391
7.120.2.1 direct . . . 391
7.120.2.2 filter . . . 392
7.120.2.3 max . . . 392
7.120.2.4 min . . . 392
7.120.2.5 subtractive . . . 392
7.121 mln::morpho::watershed Namespace Reference . . . 394
7.121.1 Detailed Description . . . 394
7.121.2 Function Documentation . . . 394
7.121.2.1 flooding . . . 394
7.121.2.2 flooding . . . 395
7.121.2.3 superpose . . . 395
7.121.2.4 superpose . . . 395
7.122 mln::morpho::watershed::watershed Namespace Reference . . . 396
7.122.1 Detailed Description . . . 396
7.123 mln::morpho::watershed::watershed::generic Namespace Reference . . . 397
7.123.1 Detailed Description . . . 397
7.124 mln::norm Namespace Reference . . . 398
7.124.1 Detailed Description . . . 399
7.124.2 Function Documentation . . . 399
7.124.2.1 l1 . . . 399
7.124.2.2 l1_distance . . . 399
7.124.2.3 l2 . . . 399
7.124.2.4 l2_distance . . . 399
7.124.2.5 linfty . . . 399
7.124.2.6 linfty_distance . . . 399
7.124.2.7 sqr_l2 . . . 399
7.125 mln::norm::impl Namespace Reference . . . 400
7.125.1 Detailed Description . . . 400
7.126 mln::opt Namespace Reference . . . 401

7.126.1 Detailed Description . . . 401
7.126.2 Function Documentation . . . 401
7.126.2.1 at . . . 401
7.126.2.2 at . . . 402
7.126.2.3 at . . . 402
7.126.2.4 at . . . 402
7.126.2.5 at . . . 402
7.126.2.6 at . . . 402
7.127 mln::opt::impl Namespace Reference . . . 403
7.127.1 Detailed Description . . . 403
7.128 mln::pw Namespace Reference . . . 404
7.128.1 Detailed Description . . . 404
7.129 mln::registration Namespace Reference . . . 405
7.129.1 Detailed Description . . . 405
7.129.2 Function Documentation . . . 406
7.129.2.1 get_rot . . . 406
7.129.2.2 icp . . . 406
7.129.2.3 icp . . . 406
7.129.2.4 registration1 . . . 407
7.129.2.5 registration2 . . . 407
7.129.2.6 registration3 . . . 407
7.130 mln::select Namespace Reference . . . 408
7.130.1 Detailed Description . . . 408
7.131 mln::set Namespace Reference . . . 409
7.131.1 Detailed Description . . . 409
7.131.2 Function Documentation . . . 409
7.131.2.1 card . . . 409
7.131.2.2 compute . . . 410
7.131.2.3 compute_with_weights . . . 410
7.131.2.4 compute_with_weights . . . 410
7.131.2.5 get . . . 411
7.131.2.6 has . . . 411
7.131.2.7 mln_meta_accu_result . . . 411
7.131.2.8 mln_meta_accu_result . . . 411
7.132 mln::subsampling Namespace Reference . . . 412
7.132.1 Detailed Description . . . 412

7.132.2 Function Documentation . . . 412
7.132.2.1 gaussian_subsampling . . . 412
7.132.2.2 subsampling . . . 412
7.133 mln::tag Namespace Reference . . . 413
7.133.1 Detailed Description . . . 413
7.134 mln::test Namespace Reference . . . 414
7.134.1 Detailed Description . . . 414
7.134.2 Function Documentation . . . 414
7.134.2.1 positive . . . 414
7.134.2.2 predicate . . . 415
7.134.2.3 predicate . . . 415
7.134.2.4 predicate . . . 415
7.135 mln::test::impl Namespace Reference . . . 416
7.135.1 Detailed Description . . . 416
7.136 mln::topo Namespace Reference . . . 417
7.136.1 Detailed Description . . . 421
7.136.2 Function Documentation . . . 421
7.136.2.1 detach . . . 421
7.136.2.2 edge . . . 421
7.136.2.3 is_facet . . . 422
7.136.2.4 make_algebraic_face . . . 422
7.136.2.5 make_algebraic_n_face . . . 422
7.136.2.6 operator!= . . . 422
7.136.2.7 operator!= . . . 422
7.136.2.8 operator!= . . . 422
7.136.2.9 operator!= . . . 423
7.136.2.10 operator+ . . . 423
7.136.2.11 operator- . . . 423
7.136.2.12 operator- . . . 423
7.136.2.13 operator- . . . 423
7.136.2.14 operator< . . . 423
7.136.2.15 operator< . . . 424
7.136.2.16 operator< . . . 424
7.136.2.17 operator< . . . 424
7.136.2.18 operator<< . . . 424
7.136.2.19 operator<< . . . 424

7.136.2.20 operator<< . . . 424
7.136.2.21 operator<< . . . 425
7.136.2.22 operator<< . . . 425
7.136.2.23 operator== . . . 425
7.136.2.24 operator== . . . 425
7.136.2.25 operator== . . . 425
7.136.2.26 operator== . . . 425
7.136.2.27 operator== . . . 426
7.137 mln::trace Namespace Reference . . . 427
7.137.1 Detailed Description . . . 427
7.138 mln::trait Namespace Reference . . . 428
7.138.1 Detailed Description . . . 428
7.139 mln::transform Namespace Reference . . . 429
7.139.1 Detailed Description . . . 430
7.139.2 Function Documentation . . . 430
7.139.2.1 distance_and_closest_point_geodesic . . . 430
7.139.2.2 distance_and_closest_point_geodesic . . . 430
7.139.2.3 distance_and_influence_zone_geodesic . . . 431
7.139.2.4 distance_front . . . 431
7.139.2.5 distance_geodesic . . . 431
7.139.2.6 hough . . . 432
7.139.2.7 influence_zone_front . . . 432
7.139.2.8 influence_zone_front . . . 432
7.139.2.9 influence_zone_geodesic . . . 432
7.139.2.10 influence_zone_geodesic . . . 432
7.139.2.11 influence_zone_geodesic . . . 433
7.140 mln::util Namespace Reference . . . 434
7.140.1 Detailed Description . . . 437
7.140.2 Typedef Documentation . . . 437
7.140.2.1 vertex_id_t . . . 437
7.140.3 Function Documentation . . . 437
7.140.3.1 display_branch . . . 437
7.140.3.2 display_tree . . . 438
7.140.3.3 lemmings . . . 438
7.140.3.4 make_greater_point . . . 438
7.140.3.5 make_greater_psite . . . 438

7.140.3.6 operator< . . . 438
7.140.3.7 operator<< . . . 438
7.140.3.8 operator<< . . . 439
7.140.3.9 operator== . . . 439
7.140.3.10operator== . . . 439
7.140.3.11ord_strict . . . 439
7.140.3.12ord_weak . . . 439
7.140.3.13tree_fast_to_image . . . 439
7.140.3.14tree_to_fast . . . 440
7.140.3.15tree_to_image . . . 440
7.141 mln::util::impl Namespace Reference . . . 441
7.141.1 Detailed Description . . . 441
7.141.2 Function Documentation . . . 441
7.141.2.1 tree_fast_to_image . . . 441
7.142 mln::value Namespace Reference . . . 442
7.142.1 Detailed Description . . . 446
7.142.2 Typedef Documentation . . . 446
7.142.2.1 float01_16 . . . 446
7.142.2.2 float01_8 . . . 446
7.142.2.3 gl16 . . . 446
7.142.2.4 gl8 . . . 446
7.142.2.5 glf . . . 447
7.142.2.6 int_s16 . . . 447
7.142.2.7 int_s32 . . . 447
7.142.2.8 int_s8 . . . 447
7.142.2.9 int_u12 . . . 447
7.142.2.10int_u16 . . . 447
7.142.2.11int_u32 . . . 447
7.142.2.12int_u8 . . . 447
7.142.2.13label_16 . . . 447
7.142.2.14label_8 . . . 447
7.142.2.15rgb16 . . . 447
7.142.2.16rgb8 . . . 448
7.142.3 Function Documentation . . . 448
7.142.3.1 cast . . . 448
7.142.3.2 equiv . . . 448

7.142.3.3 operator* . . . 448
7.142.3.4 operator* . . . 448
7.142.3.5 operator+ . . . 448
7.142.3.6 operator+ . . . 448
7.142.3.7 operator- . . . 448
7.142.3.8 operator- . . . 449
7.142.3.9 operator/ . . . 449
7.142.3.10 operator/ . . . 449
7.142.3.11 operator<< . . . 449
7.142.3.12 operator<< . . . 449
7.142.3.13 operator<< . . . 449
7.142.3.14 operator<< . . . 450
7.142.3.15 operator<< . . . 450
7.142.3.16 operator<< . . . 450
7.142.3.17 operator<< . . . 450
7.142.3.18 operator<< . . . 451
7.142.3.19 operator<< . . . 451
7.142.3.20 operator<< . . . 451
7.142.3.21 operator<< . . . 451
7.142.3.22 operator<< . . . 452
7.142.3.23 operator== . . . 452
7.142.3.24 operator== . . . 452
7.142.3.25 other . . . 452
7.142.3.26 stack . . . 452
7.143 mln::value::impl Namespace Reference . . . 453
7.143.1 Detailed Description . . . 453
7.144 mln::win Namespace Reference . . . 454
7.144.1 Detailed Description . . . 455
7.144.2 Function Documentation . . . 455
7.144.2.1 diff . . . 455
7.144.2.2 mln_regular . . . 455
7.144.2.3 mln_regular . . . 456
7.144.2.4 sym . . . 456
7.144.2.5 sym . . . 456

8 Class Documentation **457**

8.1 mln::accu::center< P, V > Struct Template Reference . . . 457

8.1.1 Detailed Description . . . 457
8.1.2 Member Function Documentation . . . 458
8.1.2.1 init . . . 458
8.1.2.2 is_valid . . . 458
8.1.2.3 take_as_init . . . 458
8.1.2.4 take_n_times . . . 458
8.1.2.5 to_result . . . 458
8.2 mln::accu::convolve< T1, T2, R > Struct Template Reference . . . 459
8.2.1 Detailed Description . . . 459
8.2.2 Member Function Documentation . . . 459
8.2.2.1 init . . . 459
8.2.2.2 is_valid . . . 459
8.2.2.3 take_as_init . . . 460
8.2.2.4 take_n_times . . . 460
8.2.2.5 to_result . . . 460
8.3 mln::accu::count_adjacent_vertices< F, S > Struct Template Reference . . . 461
8.3.1 Detailed Description . . . 461
8.3.2 Member Function Documentation . . . 461
8.3.2.1 init . . . 461
8.3.2.2 is_valid . . . 462
8.3.2.3 set_value . . . 462
8.3.2.4 take_as_init . . . 462
8.3.2.5 take_n_times . . . 462
8.3.2.6 to_result . . . 462
8.4 mln::accu::count_labels< L > Struct Template Reference . . . 463
8.4.1 Detailed Description . . . 463
8.4.2 Member Function Documentation . . . 463
8.4.2.1 init . . . 463
8.4.2.2 is_valid . . . 463
8.4.2.3 set_value . . . 464
8.4.2.4 take_as_init . . . 464
8.4.2.5 take_n_times . . . 464
8.4.2.6 to_result . . . 464
8.5 mln::accu::count_value< V > Struct Template Reference . . . 465
8.5.1 Detailed Description . . . 465
8.5.2 Member Function Documentation . . . 465

8.5.2.1 init . . . 465
8.5.2.2 is_valid . . . 465
8.5.2.3 set_value . . . 466
8.5.2.4 take_as_init . . . 466
8.5.2.5 take_n_times . . . 466
8.5.2.6 to_result . . . 466
8.6 mln::accu::histo< V > Struct Template Reference . . . 467
8.6.1 Detailed Description . . . 467
8.6.2 Member Function Documentation . . . 467
8.6.2.1 is_valid . . . 467
8.6.2.2 take . . . 467
8.6.2.3 take_as_init . . . 468
8.6.2.4 take_n_times . . . 468
8.6.2.5 vect . . . 468
8.7 mln::accu::label_used< L > Struct Template Reference . . . 469
8.7.1 Detailed Description . . . 469
8.7.2 Member Function Documentation . . . 469
8.7.2.1 init . . . 469
8.7.2.2 is_valid . . . 469
8.7.2.3 take . . . 470
8.7.2.4 take_as_init . . . 470
8.7.2.5 take_n_times . . . 470
8.7.2.6 to_result . . . 470
8.8 mln::accu::logic::land Struct Reference . . . 471
8.8.1 Detailed Description . . . 471
8.8.2 Member Function Documentation . . . 471
8.8.2.1 init . . . 471
8.8.2.2 is_valid . . . 471
8.8.2.3 take_as_init . . . 471
8.8.2.4 take_n_times . . . 472
8.8.2.5 to_result . . . 472
8.9 mln::accu::logic::land_basic Struct Reference . . . 473
8.9.1 Detailed Description . . . 473
8.9.2 Member Function Documentation . . . 473
8.9.2.1 can_stop . . . 473
8.9.2.2 init . . . 473

8.9.2.3 is_valid . . . 473
8.9.2.4 to_result . . . 474
8.10 mln::accu::logic::lor Struct Reference . . . 475
8.10.1 Detailed Description . . . 475
8.10.2 Member Function Documentation . . . 475
8.10.2.1 init . . . 475
8.10.2.2 is_valid . . . 475
8.10.2.3 take_as_init . . . 475
8.10.2.4 take_n_times . . . 476
8.10.2.5 to_result . . . 476
8.11 mln::accu::logic::lor_basic Struct Reference . . . 477
8.11.1 Detailed Description . . . 477
8.11.2 Member Function Documentation . . . 477
8.11.2.1 can_stop . . . 477
8.11.2.2 init . . . 477
8.11.2.3 is_valid . . . 478
8.11.2.4 take_as_init . . . 478
8.11.2.5 take_n_times . . . 478
8.11.2.6 to_result . . . 478
8.12 mln::accu::maj_h< T > Struct Template Reference . . . 479
8.12.1 Detailed Description . . . 479
8.12.2 Member Function Documentation . . . 479
8.12.2.1 init . . . 479
8.12.2.2 is_valid . . . 479
8.12.2.3 take_as_init . . . 480
8.12.2.4 take_n_times . . . 480
8.12.2.5 to_result . . . 480
8.13 mln::accu::math::count< T > Struct Template Reference . . . 481
8.13.1 Detailed Description . . . 481
8.13.2 Member Function Documentation . . . 481
8.13.2.1 init . . . 481
8.13.2.2 is_valid . . . 481
8.13.2.3 set_value . . . 482
8.13.2.4 take_as_init . . . 482
8.13.2.5 take_n_times . . . 482
8.13.2.6 to_result . . . 482

8.14 mln::accu::math::inf< T > Struct Template Reference 483
8.14.1 Detailed Description 483
8.14.2 Member Function Documentation 483
8.14.2.1 init 483
8.14.2.2 is_valid 483
8.14.2.3 take_as_init 484
8.14.2.4 take_n_times 484
8.14.2.5 to_result 484
8.15 mln::accu::math::sum< T, S > Struct Template Reference 485
8.15.1 Detailed Description 485
8.15.2 Member Function Documentation 485
8.15.2.1 init 485
8.15.2.2 is_valid 485
8.15.2.3 take_as_init 486
8.15.2.4 take_n_times 486
8.15.2.5 to_result 486
8.16 mln::accu::math::sup< T > Struct Template Reference 487
8.16.1 Detailed Description 487
8.16.2 Member Function Documentation 487
8.16.2.1 init 487
8.16.2.2 is_valid 487
8.16.2.3 take_as_init 488
8.16.2.4 take_n_times 488
8.16.2.5 to_result 488
8.17 mln::accu::max_site< I > Struct Template Reference 489
8.17.1 Detailed Description 489
8.17.2 Member Function Documentation 489
8.17.2.1 init 489
8.17.2.2 is_valid 489
8.17.2.3 take_as_init 489
8.17.2.4 take_n_times 490
8.17.2.5 to_result 490
8.18 mln::accu::meta::center Struct Reference 491
8.18.1 Detailed Description 491
8.19 mln::accu::meta::count_adjacent_vertices Struct Reference 492
8.19.1 Detailed Description 492

8.20 mln::accu::meta::count_labels Struct Reference 493
8.20.1 Detailed Description 493
8.21 mln::accu::meta::count_value Struct Reference 494
8.21.1 Detailed Description 494
8.22 mln::accu::meta::histo Struct Reference 495
8.22.1 Detailed Description 495
8.23 mln::accu::meta::label_used Struct Reference 496
8.23.1 Detailed Description 496
8.24 mln::accu::meta::logic::land Struct Reference 497
8.24.1 Detailed Description 497
8.25 mln::accu::meta::logic::land_basic Struct Reference 498
8.25.1 Detailed Description 498
8.26 mln::accu::meta::logic::lor Struct Reference 499
8.26.1 Detailed Description 499
8.27 mln::accu::meta::logic::lor_basic Struct Reference 500
8.27.1 Detailed Description 500
8.28 mln::accu::meta::maj_h Struct Reference 501
8.28.1 Detailed Description 501
8.29 mln::accu::meta::math::count Struct Reference 502
8.29.1 Detailed Description 502
8.30 mln::accu::meta::math::inf Struct Reference 503
8.30.1 Detailed Description 503
8.31 mln::accu::meta::math::sum Struct Reference 504
8.31.1 Detailed Description 504
8.32 mln::accu::meta::math::sup Struct Reference 505
8.32.1 Detailed Description 505
8.33 mln::accu::meta::max_site Struct Reference 506
8.33.1 Detailed Description 506
8.34 mln::accu::meta::nil Struct Reference 507
8.34.1 Detailed Description 507
8.35 mln::accu::meta::p< mA > Struct Template Reference 508
8.35.1 Detailed Description 508
8.36 mln::accu::meta::pair< A1, A2 > Struct Template Reference 509
8.36.1 Detailed Description 509
8.37 mln::accu::meta::rms Struct Reference 510
8.37.1 Detailed Description 510

8.38 mln::accu::meta::shape::bbox Struct Reference . . . 511
8.38.1 Detailed Description . . . 511
8.39 mln::accu::meta::shape::height Struct Reference . . . 512
8.39.1 Detailed Description . . . 512
8.40 mln::accu::meta::shape::volume Struct Reference . . . 513
8.40.1 Detailed Description . . . 513
8.41 mln::accu::meta::stat::max Struct Reference . . . 514
8.41.1 Detailed Description . . . 514
8.42 mln::accu::meta::stat::max_h Struct Reference . . . 515
8.42.1 Detailed Description . . . 515
8.43 mln::accu::meta::stat::mean Struct Reference . . . 516
8.43.1 Detailed Description . . . 516
8.44 mln::accu::meta::stat::median_alt< T > Struct Template Reference . . . 517
8.44.1 Detailed Description . . . 517
8.45 mln::accu::meta::stat::median_h Struct Reference . . . 518
8.45.1 Detailed Description . . . 518
8.46 mln::accu::meta::stat::min Struct Reference . . . 519
8.46.1 Detailed Description . . . 519
8.47 mln::accu::meta::stat::min_h Struct Reference . . . 520
8.47.1 Detailed Description . . . 520
8.48 mln::accu::meta::stat::rank Struct Reference . . . 521
8.48.1 Detailed Description . . . 521
8.49 mln::accu::meta::stat::rank_high_quant Struct Reference . . . 522
8.49.1 Detailed Description . . . 522
8.50 mln::accu::meta::tuple< n, > Struct Template Reference . . . 523
8.50.1 Detailed Description . . . 523
8.51 mln::accu::meta::val< mA > Struct Template Reference . . . 524
8.51.1 Detailed Description . . . 524
8.52 mln::accu::nil< T > Struct Template Reference . . . 525
8.52.1 Detailed Description . . . 525
8.52.2 Member Function Documentation . . . 525
8.52.2.1 init . . . 525
8.52.2.2 is_valid . . . 525
8.52.2.3 take_as_init . . . 525
8.52.2.4 take_n_times . . . 526
8.52.2.5 to_result . . . 526

8.53 mln::accu::p< A > Struct Template Reference . . . 527
8.53.1 Detailed Description . . . 527
8.53.2 Member Function Documentation . . . 527
8.53.2.1 init . . . 527
8.53.2.2 is_valid . . . 527
8.53.2.3 take_as_init . . . 528
8.53.2.4 take_n_times . . . 528
8.53.2.5 to_result . . . 528
8.54 mln::accu::pair< A1, A2, T > Struct Template Reference . . . 529
8.54.1 Detailed Description . . . 529
8.54.2 Member Function Documentation . . . 530
8.54.2.1 init . . . 530
8.54.2.2 is_valid . . . 530
8.54.2.3 take_as_init . . . 530
8.54.2.4 take_n_times . . . 530
8.54.2.5 to_result . . . 530
8.55 mln::accu::rms< T, V > Struct Template Reference . . . 531
8.55.1 Detailed Description . . . 531
8.55.2 Member Function Documentation . . . 531
8.55.2.1 init . . . 531
8.55.2.2 is_valid . . . 531
8.55.2.3 take_as_init . . . 532
8.55.2.4 take_n_times . . . 532
8.55.2.5 to_result . . . 532
8.56 mln::accu::shape::bbox< P > Struct Template Reference . . . 533
8.56.1 Detailed Description . . . 533
8.56.2 Member Function Documentation . . . 533
8.56.2.1 init . . . 533
8.56.2.2 is_valid . . . 533
8.56.2.3 take_as_init . . . 534
8.56.2.4 take_n_times . . . 534
8.56.2.5 to_result . . . 534
8.57 mln::accu::shape::height< I > Struct Template Reference . . . 535
8.57.1 Detailed Description . . . 535
8.57.2 Member Typedef Documentation . . . 536
8.57.2.1 argument . . . 536

8.57.2.2 value . . . 536
8.57.3 Member Function Documentation . . . 536
8.57.3.1 init . . . 536
8.57.3.2 is_valid . . . 536
8.57.3.3 set_value . . . 536
8.57.3.4 take_as_init . . . 536
8.57.3.5 take_n_times . . . 536
8.57.3.6 to_result . . . 536
8.58 mln::accu::shape::volume< I > Struct Template Reference . . . 537
8.58.1 Detailed Description . . . 537
8.58.2 Member Typedef Documentation . . . 538
8.58.2.1 argument . . . 538
8.58.2.2 value . . . 538
8.58.3 Member Function Documentation . . . 538
8.58.3.1 init . . . 538
8.58.3.2 is_valid . . . 538
8.58.3.3 set_value . . . 538
8.58.3.4 take_as_init . . . 538
8.58.3.5 take_n_times . . . 538
8.58.3.6 to_result . . . 539
8.59 mln::accu::site_set::rectangularity< P > Class Template Reference . . . 540
8.59.1 Detailed Description . . . 540
8.59.2 Constructor & Destructor Documentation . . . 540
8.59.2.1 rectangularity . . . 540
8.59.3 Member Function Documentation . . . 540
8.59.3.1 area . . . 540
8.59.3.2 bbox . . . 541
8.59.3.3 take_as_init . . . 541
8.59.3.4 take_n_times . . . 541
8.59.3.5 to_result . . . 541
8.60 mln::accu::stat::deviation< T, S, M > Struct Template Reference . . . 542
8.60.1 Detailed Description . . . 542
8.60.2 Member Function Documentation . . . 542
8.60.2.1 init . . . 542
8.60.2.2 is_valid . . . 542
8.60.2.3 take_as_init . . . 543

8.60.2.4 take_n_times . . . 543
8.60.2.5 to_result . . . 543
8.61 mln::accu::stat::max< T > Struct Template Reference . . . 544
8.61.1 Detailed Description . . . 544
8.61.2 Member Function Documentation . . . 544
8.61.2.1 init . . . 544
8.61.2.2 is_valid . . . 544
8.61.2.3 take_as_init . . . 545
8.61.2.4 take_n_times . . . 545
8.61.2.5 to_result . . . 545
8.62 mln::accu::stat::max_h< V > Struct Template Reference . . . 546
8.62.1 Detailed Description . . . 546
8.62.2 Member Function Documentation . . . 546
8.62.2.1 init . . . 546
8.62.2.2 is_valid . . . 546
8.62.2.3 take_as_init . . . 546
8.62.2.4 take_n_times . . . 547
8.62.2.5 to_result . . . 547
8.63 mln::accu::stat::mean< T, S, M > Struct Template Reference . . . 548
8.63.1 Detailed Description . . . 548
8.63.2 Member Function Documentation . . . 548
8.63.2.1 init . . . 548
8.63.2.2 is_valid . . . 548
8.63.2.3 take_as_init . . . 549
8.63.2.4 take_n_times . . . 549
8.63.2.5 to_result . . . 549
8.64 mln::accu::stat::median_alt< S > Struct Template Reference . . . 550
8.64.1 Detailed Description . . . 550
8.64.2 Member Function Documentation . . . 550
8.64.2.1 is_valid . . . 550
8.64.2.2 take . . . 551
8.64.2.3 take_as_init . . . 551
8.64.2.4 take_n_times . . . 551
8.64.2.5 to_result . . . 551
8.65 mln::accu::stat::median_h< V > Struct Template Reference . . . 552
8.65.1 Detailed Description . . . 552

8.65.2 Member Function Documentation . . . 552
8.65.2.1 init . . . 552
8.65.2.2 is_valid . . . 553
8.65.2.3 take_as_init . . . 553
8.65.2.4 take_n_times . . . 553
8.65.2.5 to_result . . . 553
8.66 mln::accu::stat::meta::deviation Struct Reference . . . 554
8.66.1 Detailed Description . . . 554
8.67 mln::accu::stat::min< T > Struct Template Reference . . . 555
8.67.1 Detailed Description . . . 555
8.67.2 Member Function Documentation . . . 555
8.67.2.1 init . . . 555
8.67.2.2 is_valid . . . 555
8.67.2.3 take_as_init . . . 556
8.67.2.4 take_n_times . . . 556
8.67.2.5 to_result . . . 556
8.68 mln::accu::stat::min_h< V > Struct Template Reference . . . 557
8.68.1 Detailed Description . . . 557
8.68.2 Member Function Documentation . . . 557
8.68.2.1 init . . . 557
8.68.2.2 is_valid . . . 557
8.68.2.3 take_as_init . . . 557
8.68.2.4 take_n_times . . . 558
8.68.2.5 to_result . . . 558
8.69 mln::accu::stat::min_max< V > Struct Template Reference . . . 559
8.69.1 Detailed Description . . . 559
8.69.2 Member Function Documentation . . . 560
8.69.2.1 init . . . 560
8.69.2.2 is_valid . . . 560
8.69.2.3 take_as_init . . . 560
8.69.2.4 take_n_times . . . 560
8.69.2.5 to_result . . . 560
8.70 mln::accu::stat::rank< T > Struct Template Reference . . . 561
8.70.1 Detailed Description . . . 561
8.70.2 Member Function Documentation . . . 561
8.70.2.1 init . . . 561

8.70.2.2 is_valid . . . 561
8.70.2.3 k . . . 562
8.70.2.4 take_as_init . . . 562
8.70.2.5 take_n_times . . . 562
8.70.2.6 to_result . . . 562
8.71 mln::accu::stat::rank< bool > Struct Template Reference . . . 563
8.71.1 Detailed Description . . . 563
8.71.2 Member Function Documentation . . . 563
8.71.2.1 init . . . 563
8.71.2.2 is_valid . . . 563
8.71.2.3 take_as_init . . . 563
8.71.2.4 take_n_times . . . 564
8.71.2.5 to_result . . . 564
8.72 mln::accu::stat::rank_high_quant< T > Struct Template Reference . . . 565
8.72.1 Detailed Description . . . 565
8.72.2 Member Function Documentation . . . 565
8.72.2.1 init . . . 565
8.72.2.2 is_valid . . . 565
8.72.2.3 take_as_init . . . 566
8.72.2.4 take_n_times . . . 566
8.72.2.5 to_result . . . 566
8.73 mln::accu::stat::var< T > Struct Template Reference . . . 567
8.73.1 Detailed Description . . . 567
8.73.2 Member Typedef Documentation . . . 568
8.73.2.1 mean_t . . . 568
8.73.3 Member Function Documentation . . . 568
8.73.3.1 init . . . 568
8.73.3.2 is_valid . . . 568
8.73.3.3 mean . . . 568
8.73.3.4 n_items . . . 568
8.73.3.5 take_as_init . . . 568
8.73.3.6 take_n_times . . . 568
8.73.3.7 to_result . . . 568
8.73.3.8 variance . . . 569
8.74 mln::accu::stat::variance< T, S, R > Struct Template Reference . . . 570
8.74.1 Detailed Description . . . 570

8.74.2 Member Function Documentation . . . 571
8.74.2.1 init . . . 571
8.74.2.2 is_valid . . . 571
8.74.2.3 mean . . . 571
8.74.2.4 n_items . . . 571
8.74.2.5 standard_deviation . . . 571
8.74.2.6 sum . . . 571
8.74.2.7 take_as_init . . . 571
8.74.2.8 take_n_times . . . 571
8.74.2.9 to_result . . . 572
8.74.2.10 var . . . 572
8.75 mln::accu::tuple< A, n, > Struct Template Reference . . . 573
8.75.1 Detailed Description . . . 573
8.75.2 Member Function Documentation . . . 573
8.75.2.1 init . . . 573
8.75.2.2 is_valid . . . 573
8.75.2.3 take_as_init . . . 574
8.75.2.4 take_n_times . . . 574
8.75.2.5 to_result . . . 574
8.76 mln::accu::val< A > Struct Template Reference . . . 575
8.76.1 Detailed Description . . . 575
8.76.2 Member Function Documentation . . . 575
8.76.2.1 init . . . 575
8.76.2.2 is_valid . . . 575
8.76.2.3 take_as_init . . . 575
8.76.2.4 take_n_times . . . 576
8.76.2.5 to_result . . . 576
8.77 mln::Accumulator< E > Struct Template Reference . . . 577
8.77.1 Detailed Description . . . 577
8.77.2 Member Function Documentation . . . 577
8.77.2.1 take_as_init . . . 577
8.77.2.2 take_n_times . . . 577
8.78 mln::algebra::h_mat< d, T > Struct Template Reference . . . 578
8.78.1 Detailed Description . . . 578
8.78.2 Member Enumeration Documentation . . . 578
8.78.2.1 "@7 . . . 578

8.78.3 Constructor & Destructor Documentation . . . 578
8.78.3.1 h_mat . . . 578
8.78.3.2 h_mat . . . 579
8.78.4 Member Function Documentation . . . 579
8.78.4.1 _1 . . . 579
8.78.4.2 t . . . 579
8.79 mln::algebra::h_vec< d, C > Struct Template Reference . . . 580
8.79.1 Detailed Description . . . 580
8.79.2 Member Enumeration Documentation . . . 581
8.79.2.1 "@8 . . . 581
8.79.3 Constructor & Destructor Documentation . . . 581
8.79.3.1 h_vec . . . 581
8.79.3.2 h_vec . . . 581
8.79.4 Member Function Documentation . . . 581
8.79.4.1 operator mat< n, 1, U > . . . 581
8.79.4.2 t . . . 581
8.79.4.3 to_vec . . . 581
8.79.5 Member Data Documentation . . . 581
8.79.5.1 origin . . . 581
8.79.5.2 zero . . . 581
8.80 mln::bkd_pixter1d< I > Class Template Reference . . . 582
8.80.1 Detailed Description . . . 582
8.80.2 Member Typedef Documentation . . . 582
8.80.2.1 image . . . 582
8.80.3 Constructor & Destructor Documentation . . . 582
8.80.3.1 bkd_pixter1d . . . 582
8.80.4 Member Function Documentation . . . 582
8.80.4.1 next . . . 582
8.81 mln::bkd_pixter2d< I > Class Template Reference . . . 584
8.81.1 Detailed Description . . . 584
8.81.2 Member Typedef Documentation . . . 584
8.81.2.1 image . . . 584
8.81.3 Constructor & Destructor Documentation . . . 584
8.81.3.1 bkd_pixter2d . . . 584
8.81.4 Member Function Documentation . . . 584
8.81.4.1 next . . . 584

8.82 mln::bkd_pixter3d< I > Class Template Reference . . . 586
8.82.1 Detailed Description . . . 586
8.82.2 Member Typedef Documentation . . . 586
8.82.2.1 image . . . 586
8.82.3 Constructor & Destructor Documentation . . . 586
8.82.3.1 bkd_pixter3d . . . 586
8.82.4 Member Function Documentation . . . 586
8.82.4.1 next . . . 586
8.83 mln::box< P > Struct Template Reference . . . 588
8.83.1 Detailed Description . . . 590
8.83.2 Member Typedef Documentation . . . 590
8.83.2.1 bkd_piter . . . 590
8.83.2.2 element . . . 590
8.83.2.3 fwd_piter . . . 590
8.83.2.4 piter . . . 591
8.83.2.5 psite . . . 591
8.83.2.6 site . . . 591
8.83.3 Member Enumeration Documentation . . . 591
8.83.3.1 "@30 . . . 591
8.83.4 Constructor & Destructor Documentation . . . 591
8.83.4.1 box . . . 591
8.83.4.2 box . . . 591
8.83.4.3 box . . . 591
8.83.5 Member Function Documentation . . . 591
8.83.5.1 bbox . . . 591
8.83.5.2 center . . . 592
8.83.5.3 crop_wrt . . . 592
8.83.5.4 enlarge . . . 592
8.83.5.5 enlarge . . . 592
8.83.5.6 has . . . 592
8.83.5.7 is_empty . . . 592
8.83.5.8 is_valid . . . 592
8.83.5.9 len . . . 593
8.83.5.10 memory_size . . . 593
8.83.5.11 nsites . . . 593
8.83.5.12 pmax . . . 593

8.83.5.13 pmax . . . 593
8.83.5.14 pmin . . . 593
8.83.5.15 pmin . . . 593
8.83.5.16 to_larger . . . 594
8.83.6 Friends And Related Function Documentation . . . 594
8.83.6.1 operator<< . . . 594
8.84 mln::Box< E > Struct Template Reference . . . 595
8.84.1 Detailed Description . . . 596
8.84.2 Member Function Documentation . . . 596
8.84.2.1 bbox . . . 596
8.84.2.2 is_empty . . . 596
8.84.2.3 len . . . 596
8.84.2.4 nsites . . . 597
8.84.3 Friends And Related Function Documentation . . . 597
8.84.3.1 operator< . . . 597
8.84.3.2 operator<= . . . 597
8.85 mln::box_runstart_piter< P > Class Template Reference . . . 598
8.85.1 Detailed Description . . . 598
8.85.2 Constructor & Destructor Documentation . . . 598
8.85.2.1 box_runstart_piter . . . 598
8.85.3 Member Function Documentation . . . 598
8.85.3.1 next . . . 598
8.85.3.2 run_length . . . 599
8.86 mln::Browsing< E > Struct Template Reference . . . 600
8.86.1 Detailed Description . . . 600
8.87 mln::canvas::browsing::backdiagonal2d_t Struct Reference . . . 601
8.87.1 Detailed Description . . . 601
8.88 mln::canvas::browsing::breadth_first_search_t Struct Reference . . . 603
8.88.1 Detailed Description . . . 603
8.89 mln::canvas::browsing::depth_first_search_t Struct Reference . . . 604
8.89.1 Detailed Description . . . 604
8.90 mln::canvas::browsing::diagonal2d_t Struct Reference . . . 605
8.90.1 Detailed Description . . . 605
8.91 mln::canvas::browsing::dir_struct_elt_incr_update_t Struct Reference . . . 607
8.91.1 Detailed Description . . . 607
8.92 mln::canvas::browsing::directional_t Struct Reference . . . 609

8.92.1 Detailed Description . . . 609
8.93 mln::canvas::browsing::fwd_t Struct Reference . . . 611
8.93.1 Detailed Description . . . 611
8.94 mln::canvas::browsing::hyper_directional_t Struct Reference . . . 613
8.94.1 Detailed Description . . . 613
8.95 mln::canvas::browsing::snake_fwd_t Struct Reference . . . 615
8.95.1 Detailed Description . . . 615
8.96 mln::canvas::browsing::snake_generic_t Struct Reference . . . 617
8.96.1 Detailed Description . . . 617
8.97 mln::canvas::browsing::snake_vert_t Struct Reference . . . 619
8.97.1 Detailed Description . . . 619
8.98 mln::canvas::chamfer< F > Struct Template Reference . . . 621
8.98.1 Detailed Description . . . 621
8.99 mln::category< R(*)(A) > Struct Template Reference . . . 622
8.99.1 Detailed Description . . . 622
8.100 mln::complex_image< D, G, V > Class Template Reference . . . 623
8.100.1 Detailed Description . . . 624
8.100.2 Member Typedef Documentation . . . 624
8.100.2.1 geom . . . 624
8.100.2.2 lvalue . . . 624
8.100.2.3 rvalue . . . 624
8.100.2.4 skeleton . . . 624
8.100.2.5 value . . . 624
8.100.3 Constructor & Destructor Documentation . . . 624
8.100.3.1 complex_image . . . 624
8.100.4 Member Function Documentation . . . 625
8.100.4.1 domain . . . 625
8.100.4.2 operator() . . . 625
8.100.4.3 operator() . . . 625
8.100.4.4 values . . . 625
8.100.5 Member Data Documentation . . . 625
8.100.5.1 dim . . . 625
8.101 mln::complex_neighborhood_bkd_piter< I, G, N > Class Template Reference . . . 626
8.101.1 Detailed Description . . . 626
8.101.2 Member Typedef Documentation . . . 626
8.101.2.1 iter_type . . . 626

8.101.2.2 psite . . . 626
8.101.3 Constructor & Destructor Documentation . . . 627
8.101.3.1 complex_neighborhood_bkd_piter . . . 627
8.101.4 Member Function Documentation . . . 627
8.101.4.1 iter . . . 627
8.101.4.2 next . . . 627
8.102 mln::complex_neighborhood_fwd_piter< I, G, N > Class Template Reference . . . 628
8.102.1 Detailed Description . . . 628
8.102.2 Member Typedef Documentation . . . 628
8.102.2.1 iter_type . . . 628
8.102.2.2 psite . . . 628
8.102.3 Constructor & Destructor Documentation . . . 629
8.102.3.1 complex_neighborhood_fwd_piter . . . 629
8.102.4 Member Function Documentation . . . 629
8.102.4.1 iter . . . 629
8.102.4.2 next . . . 629
8.103 mln::complex_psite< D, G > Class Template Reference . . . 630
8.103.1 Detailed Description . . . 630
8.103.2 Constructor & Destructor Documentation . . . 631
8.103.2.1 complex_psite . . . 631
8.103.2.2 complex_psite . . . 631
8.103.3 Member Function Documentation . . . 631
8.103.3.1 change_target . . . 631
8.103.3.2 face . . . 631
8.103.3.3 face_id . . . 631
8.103.3.4 invalidate . . . 631
8.103.3.5 is_valid . . . 632
8.103.3.6 n . . . 632
8.103.3.7 site_set . . . 632
8.104 mln::complex_window_bkd_piter< I, G, W > Class Template Reference . . . 633
8.104.1 Detailed Description . . . 633
8.104.2 Member Typedef Documentation . . . 633
8.104.2.1 iter_type . . . 633
8.104.2.2 psite . . . 633
8.104.3 Constructor & Destructor Documentation . . . 634
8.104.3.1 complex_window_bkd_piter . . . 634

8.104.4 Member Function Documentation . . . 634
8.104.4.1 iter . . . 634
8.104.4.2 next . . . 634
8.105 mln::complex_window_fwd_piter< I, G, W > Class Template Reference . . . 635
8.105.1 Detailed Description . . . 635
8.105.2 Member Typedef Documentation . . . 635
8.105.2.1 iter_type . . . 635
8.105.2.2 psite . . . 635
8.105.3 Constructor & Destructor Documentation . . . 636
8.105.3.1 complex_window_fwd_piter . . . 636
8.105.4 Member Function Documentation . . . 636
8.105.4.1 iter . . . 636
8.105.4.2 next . . . 636
8.106 mln::decorated_image< I, D > Struct Template Reference . . . 637
8.106.1 Detailed Description . . . 637
8.106.2 Member Typedef Documentation . . . 638
8.106.2.1 lvalue . . . 638
8.106.2.2 psite . . . 638
8.106.2.3 rvalue . . . 638
8.106.2.4 skeleton . . . 638
8.106.3 Constructor & Destructor Documentation . . . 638
8.106.3.1 decorated_image . . . 638
8.106.4 Member Function Documentation . . . 638
8.106.4.1 decoration . . . 638
8.106.4.2 decoration . . . 638
8.106.4.3 operator decorated_image< const I, D > . . . 638
8.106.4.4 operator() . . . 639
8.106.4.5 operator() . . . 639
8.107 mln::Delta_Point_Site< E > Struct Template Reference . . . 640
8.107.1 Detailed Description . . . 640
8.108 mln::Delta_Point_Site< void > Struct Template Reference . . . 641
8.108.1 Detailed Description . . . 641
8.109 mln::doc::Accumulator< E > Struct Template Reference . . . 642
8.109.1 Detailed Description . . . 642
8.109.2 Member Typedef Documentation . . . 642
8.109.2.1 argument . . . 642

8.109.3 Member Function Documentation . . . 642
8.109.3.1 init . . . 642
8.109.3.2 take . . . 642
8.109.3.3 take . . . 643
8.110 mln::doc::Box< E > Struct Template Reference . . . 644
8.110.1 Detailed Description . . . 645
8.110.2 Member Typedef Documentation . . . 645
8.110.2.1 bkd_piter . . . 645
8.110.2.2 fwd_piter . . . 645
8.110.2.3 psite . . . 645
8.110.2.4 site . . . 645
8.110.3 Member Function Documentation . . . 645
8.110.3.1 bbox . . . 645
8.110.3.2 has . . . 646
8.110.3.3 nsites . . . 646
8.110.3.4 pmax . . . 646
8.110.3.5 pmin . . . 646
8.111 mln::doc::Dpoint< E > Struct Template Reference . . . 647
8.111.1 Detailed Description . . . 647
8.111.2 Member Typedef Documentation . . . 647
8.111.2.1 coord . . . 647
8.111.2.2 dpoint . . . 648
8.111.2.3 point . . . 648
8.111.3 Member Enumeration Documentation . . . 648
8.111.3.1 "@18 . . . 648
8.111.4 Member Function Documentation . . . 648
8.111.4.1 operator[] . . . 648
8.112 mln::doc::Fastest_Image< E > Struct Template Reference . . . 649
8.112.1 Detailed Description . . . 651
8.112.2 Member Typedef Documentation . . . 651
8.112.2.1 bkd_piter . . . 651
8.112.2.2 coord . . . 651
8.112.2.3 dpoint . . . 651
8.112.2.4 fwd_piter . . . 651
8.112.2.5 lvalue . . . 652
8.112.2.6 point . . . 652

8.112.2.7 pset . . . 652
8.112.2.8 psite . . . 652
8.112.2.9 rvalue . . . 652
8.112.2.10 skeleton . . . 652
8.112.2.11 value . . . 652
8.112.2.12 vset . . . 652
8.112.3 Member Function Documentation . . . 653
8.112.3.1 bbox . . . 653
8.112.3.2 border . . . 653
8.112.3.3 buffer . . . 653
8.112.3.4 delta_index . . . 653
8.112.3.5 domain . . . 653
8.112.3.6 has . . . 654
8.112.3.7 has . . . 654
8.112.3.8 is_valid . . . 654
8.112.3.9 nelements . . . 654
8.112.3.10 nsites . . . 654
8.112.3.11 operator() . . . 654
8.112.3.12 operator() . . . 655
8.112.3.13 operator[] . . . 655
8.112.3.14 operator[] . . . 655
8.112.3.15 point_at_index . . . 656
8.112.3.16 values . . . 656
8.113 mln::doc::Generalized_Pixel< E > Struct Template Reference . . . 657
8.113.1 Detailed Description . . . 657
8.113.2 Member Typedef Documentation . . . 657
8.113.2.1 image . . . 657
8.113.2.2 rvalue . . . 658
8.113.2.3 value . . . 658
8.113.3 Member Function Documentation . . . 658
8.113.3.1 ima . . . 658
8.113.3.2 val . . . 658
8.114 mln::doc::Image< E > Struct Template Reference . . . 659
8.114.1 Detailed Description . . . 660
8.114.2 Member Typedef Documentation . . . 661
8.114.2.1 bkd_piter . . . 661

8.114.2.2 coord . . . 661
8.114.2.3 dpoint . . . 661
8.114.2.4 fwd_piter . . . 661
8.114.2.5 lvalue . . . 661
8.114.2.6 point . . . 661
8.114.2.7 pset . . . 661
8.114.2.8 psite . . . 662
8.114.2.9 rvalue . . . 662
8.114.2.10skeleton . . . 662
8.114.2.11lvalue . . . 662
8.114.2.12vset . . . 662
8.114.3 Member Function Documentation . . . 662
8.114.3.1 bbox . . . 662
8.114.3.2 domain . . . 662
8.114.3.3 has . . . 663
8.114.3.4 has . . . 663
8.114.3.5 is_valid . . . 663
8.114.3.6 nsites . . . 663
8.114.3.7 operator() . . . 663
8.114.3.8 operator() . . . 664
8.114.3.9 values . . . 664
8.115mln::doc::Iterator< E > Struct Template Reference . . . 665
8.115.1 Detailed Description . . . 665
8.115.2 Member Function Documentation . . . 665
8.115.2.1 invalidate . . . 665
8.115.2.2 is_valid . . . 665
8.115.2.3 start . . . 666
8.116mln::doc::Neighborhood< E > Struct Template Reference . . . 667
8.116.1 Detailed Description . . . 667
8.116.2 Member Typedef Documentation . . . 667
8.116.2.1 bkd_niter . . . 667
8.116.2.2 dpoint . . . 668
8.116.2.3 fwd_niter . . . 668
8.116.2.4 niter . . . 668
8.116.2.5 point . . . 668
8.117mln::doc::Object< E > Struct Template Reference . . . 669

8.117.1 Detailed Description . . . 669
8.118 mln::doc::Pixel_Iterator< E > Struct Template Reference . . . 670
8.118.1 Detailed Description . . . 671
8.118.2 Member Typedef Documentation . . . 671
8.118.2.1 image . . . 671
8.118.2.2 lvalue . . . 671
8.118.2.3 rvalue . . . 671
8.118.2.4 value . . . 671
8.118.3 Member Function Documentation . . . 671
8.118.3.1 ima . . . 671
8.118.3.2 invalidate . . . 671
8.118.3.3 is_valid . . . 671
8.118.3.4 start . . . 672
8.118.3.5 val . . . 672
8.119 mln::doc::Point_Site< E > Struct Template Reference . . . 673
8.119.1 Detailed Description . . . 673
8.119.2 Member Typedef Documentation . . . 673
8.119.2.1 coord . . . 673
8.119.2.2 dpoint . . . 673
8.119.2.3 mesh . . . 674
8.119.2.4 point . . . 674
8.119.3 Member Enumeration Documentation . . . 674
8.119.3.1 "@19 . . . 674
8.119.4 Member Function Documentation . . . 674
8.119.4.1 operator[] . . . 674
8.119.4.2 to_point . . . 675
8.120 mln::doc::Site_Iterator< E > Struct Template Reference . . . 676
8.120.1 Detailed Description . . . 676
8.120.2 Member Typedef Documentation . . . 677
8.120.2.1 psite . . . 677
8.120.3 Member Function Documentation . . . 677
8.120.3.1 invalidate . . . 677
8.120.3.2 is_valid . . . 677
8.120.3.3 operator psite . . . 677
8.120.3.4 start . . . 677
8.121 mln::doc::Site_Set< E > Struct Template Reference . . . 678

8.121.1 Detailed Description . . . 678
8.121.2 Member Typedef Documentation . . . 679
8.121.2.1 bkd_piter . . . 679
8.121.2.2 fwd_piter . . . 679
8.121.2.3 psite . . . 679
8.121.2.4 site . . . 679
8.121.3 Member Function Documentation . . . 679
8.121.3.1 has . . . 679
8.122 mln::doc::Value_Iterator< E > Struct Template Reference . . . 680
8.122.1 Detailed Description . . . 680
8.122.2 Member Typedef Documentation . . . 681
8.122.2.1 value . . . 681
8.122.3 Member Function Documentation . . . 681
8.122.3.1 invalidate . . . 681
8.122.3.2 is_valid . . . 681
8.122.3.3 operator value . . . 681
8.122.3.4 start . . . 681
8.123 mln::doc::Value_Set< E > Struct Template Reference . . . 682
8.123.1 Detailed Description . . . 682
8.123.2 Member Typedef Documentation . . . 683
8.123.2.1 bkd_viter . . . 683
8.123.2.2 fwd_viter . . . 683
8.123.2.3 value . . . 683
8.123.3 Member Function Documentation . . . 683
8.123.3.1 has . . . 683
8.123.3.2 index_of . . . 683
8.123.3.3 nvalues . . . 683
8.123.3.4 operator[] . . . 683
8.124 mln::doc::Weighted_Window< E > Struct Template Reference . . . 684
8.124.1 Detailed Description . . . 685
8.124.2 Member Typedef Documentation . . . 685
8.124.2.1 bkd_qiter . . . 685
8.124.2.2 dpoint . . . 685
8.124.2.3 fwd_qiter . . . 685
8.124.2.4 point . . . 685
8.124.2.5 weight . . . 685

8.124.2.6 window . . . 685
8.124.3 Member Function Documentation . . . 686
8.124.3.1 delta . . . 686
8.124.3.2 is_centered . . . 686
8.124.3.3 is_empty . . . 686
8.124.3.4 sym . . . 686
8.124.3.5 win . . . 686
8.125 mln::doc::Window< E > Struct Template Reference . . . 687
8.125.1 Detailed Description . . . 687
8.125.2 Member Typedef Documentation . . . 687
8.125.2.1 bkd_qiter . . . 687
8.125.2.2 fwd_qiter . . . 687
8.125.2.3 qiter . . . 688
8.126 mln::dpoint< G, C > Struct Template Reference . . . 689
8.126.1 Detailed Description . . . 690
8.126.2 Member Typedef Documentation . . . 690
8.126.2.1 coord . . . 690
8.126.2.2 grid . . . 691
8.126.2.3 psite . . . 691
8.126.2.4 site . . . 691
8.126.2.5 vec . . . 691
8.126.3 Member Enumeration Documentation . . . 691
8.126.3.1 "@21 . . . 691
8.126.4 Constructor & Destructor Documentation . . . 691
8.126.4.1 dpoint . . . 691
8.126.4.2 dpoint . . . 691
8.126.4.3 dpoint . . . 691
8.126.4.4 dpoint . . . 692
8.126.4.5 dpoint . . . 692
8.126.5 Member Function Documentation . . . 692
8.126.5.1 operator mln::algebra::vec< dpoint< G, C >::dim, Q > . . . 692
8.126.5.2 operator[] . . . 692
8.126.5.3 operator[] . . . 692
8.126.5.4 set_all . . . 693
8.126.5.5 to_vec . . . 693
8.127 mln::Dpoint< E > Struct Template Reference . . . 694

8.127.1 Detailed Description . . . 694
8.127.2 Member Function Documentation . . . 694
8.127.2.1 to_dpoint . . . 694
8.128 mln::dpoints_bkd_pixter< I > Class Template Reference . . . 695
8.128.1 Detailed Description . . . 695
8.128.2 Constructor & Destructor Documentation . . . 695
8.128.2.1 dpoints_bkd_pixter . . . 695
8.128.2.2 dpoints_bkd_pixter . . . 696
8.128.3 Member Function Documentation . . . 696
8.128.3.1 center_val . . . 696
8.128.3.2 invalidate . . . 696
8.128.3.3 is_valid . . . 696
8.128.3.4 next . . . 696
8.128.3.5 start . . . 697
8.128.3.6 update . . . 697
8.129 mln::dpoints_fwd_pixter< I > Class Template Reference . . . 698
8.129.1 Detailed Description . . . 698
8.129.2 Constructor & Destructor Documentation . . . 698
8.129.2.1 dpoints_fwd_pixter . . . 698
8.129.2.2 dpoints_fwd_pixter . . . 699
8.129.3 Member Function Documentation . . . 699
8.129.3.1 center_val . . . 699
8.129.3.2 invalidate . . . 699
8.129.3.3 is_valid . . . 699
8.129.3.4 next . . . 699
8.129.3.5 start . . . 700
8.129.3.6 update . . . 700
8.130 mln::dpsites_bkd_piter< V > Class Template Reference . . . 701
8.130.1 Detailed Description . . . 701
8.130.2 Constructor & Destructor Documentation . . . 701
8.130.2.1 dpsites_bkd_piter . . . 701
8.130.2.2 dpsites_bkd_piter . . . 701
8.130.3 Member Function Documentation . . . 701
8.130.3.1 next . . . 701
8.131 mln::dpsites_fwd_piter< V > Class Template Reference . . . 703
8.131.1 Detailed Description . . . 703

8.131.2 Constructor & Destructor Documentation . . . 703
8.131.2.1 dpsites_fwd_piter . . . 703
8.131.2.2 dpsites_fwd_piter . . . 703
8.131.3 Member Function Documentation . . . 703
8.131.3.1 next . . . 703
8.132 mln::Edge< E > Struct Template Reference . . . 705
8.132.1 Detailed Description . . . 705
8.133 mln::edge_image< P, V, G > Class Template Reference . . . 706
8.133.1 Detailed Description . . . 706
8.133.2 Member Typedef Documentation . . . 706
8.133.2.1 graph_t . . . 706
8.133.2.2 nbh_t . . . 707
8.133.2.3 site_function_t . . . 707
8.133.2.4 skeleton . . . 707
8.133.2.5 win_t . . . 707
8.133.3 Constructor & Destructor Documentation . . . 707
8.133.3.1 edge_image . . . 707
8.133.4 Member Function Documentation . . . 707
8.133.4.1 operator() . . . 707
8.134 mln::extended< I > Struct Template Reference . . . 708
8.134.1 Detailed Description . . . 708
8.134.2 Member Typedef Documentation . . . 708
8.134.2.1 skeleton . . . 708
8.134.2.2 value . . . 708
8.134.3 Constructor & Destructor Documentation . . . 709
8.134.3.1 extended . . . 709
8.134.3.2 extended . . . 709
8.134.4 Member Function Documentation . . . 709
8.134.4.1 domain . . . 709
8.135 mln::extension_fun< I, F > Class Template Reference . . . 710
8.135.1 Detailed Description . . . 710
8.135.2 Member Typedef Documentation . . . 711
8.135.2.1 rvalue . . . 711
8.135.2.2 skeleton . . . 711
8.135.2.3 value . . . 711
8.135.3 Constructor & Destructor Documentation . . . 711

8.135.3.1 extension_fun . . . 711
8.135.3.2 extension_fun . . . 711
8.135.4 Member Function Documentation . . . 711
8.135.4.1 extension . . . 711
8.135.4.2 has . . . 711
8.135.4.3 operator() . . . 711
8.135.4.4 operator() . . . 712
8.136 mln::extension_ima< I, J > Class Template Reference . . . 713
8.136.1 Detailed Description . . . 713
8.136.2 Member Typedef Documentation . . . 714
8.136.2.1 rvalue . . . 714
8.136.2.2 skeleton . . . 714
8.136.2.3 value . . . 714
8.136.3 Constructor & Destructor Documentation . . . 714
8.136.3.1 extension_ima . . . 714
8.136.3.2 extension_ima . . . 714
8.136.4 Member Function Documentation . . . 714
8.136.4.1 extension . . . 714
8.136.4.2 has . . . 714
8.136.4.3 operator() . . . 714
8.136.4.4 operator() . . . 715
8.137 mln::extension_val< I > Class Template Reference . . . 716
8.137.1 Detailed Description . . . 716
8.137.2 Member Typedef Documentation . . . 717
8.137.2.1 rvalue . . . 717
8.137.2.2 skeleton . . . 717
8.137.2.3 value . . . 717
8.137.3 Constructor & Destructor Documentation . . . 717
8.137.3.1 extension_val . . . 717
8.137.3.2 extension_val . . . 717
8.137.4 Member Function Documentation . . . 717
8.137.4.1 change_extension . . . 717
8.137.4.2 extension . . . 717
8.137.4.3 has . . . 717
8.137.4.4 operator() . . . 717
8.137.4.5 operator() . . . 718

8.138 mln::flat_image< T, S > Struct Template Reference . . . 719
8.138.1 Detailed Description . . . 719
8.138.2 Member Typedef Documentation . . . 720
8.138.2.1 lvalue . . . 720
8.138.2.2 rvalue . . . 720
8.138.2.3 skeleton . . . 720
8.138.2.4 value . . . 720
8.138.3 Constructor & Destructor Documentation . . . 720
8.138.3.1 flat_image . . . 720
8.138.3.2 flat_image . . . 720
8.138.4 Member Function Documentation . . . 720
8.138.4.1 domain . . . 720
8.138.4.2 has . . . 720
8.138.4.3 operator() . . . 720
8.138.4.4 operator() . . . 721
8.139 mln::fun::p2b::antilogy Struct Reference . . . 722
8.139.1 Detailed Description . . . 722
8.140 mln::fun::p2b::tautology Struct Reference . . . 723
8.140.1 Detailed Description . . . 723
8.141 mln::fun::v2b::lnot< V > Struct Template Reference . . . 724
8.141.1 Detailed Description . . . 724
8.142 mln::fun::v2b::threshold< V > Struct Template Reference . . . 725
8.142.1 Detailed Description . . . 725
8.143 mln::fun::v2v::ch_function_value< F, V > Class Template Reference . . . 726
8.143.1 Detailed Description . . . 726
8.144 mln::fun::v2v::component< T, i > Struct Template Reference . . . 727
8.144.1 Detailed Description . . . 727
8.145 mln::fun::v2v::l1_norm< V, R > Struct Template Reference . . . 728
8.145.1 Detailed Description . . . 728
8.146 mln::fun::v2v::l2_norm< V, R > Struct Template Reference . . . 729
8.146.1 Detailed Description . . . 729
8.147 mln::fun::v2v::linear< V, T, R > Struct Template Reference . . . 730
8.147.1 Detailed Description . . . 730
8.148 mln::fun::v2v::linfty_norm< V, R > Struct Template Reference . . . 731
8.148.1 Detailed Description . . . 731
8.149 mln::fun::v2w2v::cos< V > Struct Template Reference . . . 732

8.149.1 Detailed Description . . . 732
8.150 mln::fun::v2w_w2v::l1_norm< V, R > Struct Template Reference . . . 733
8.150.1 Detailed Description . . . 733
8.151 mln::fun::v2w_w2v::l2_norm< V, R > Struct Template Reference . . . 734
8.151.1 Detailed Description . . . 734
8.152 mln::fun::v2w_w2v::linfty_norm< V, R > Struct Template Reference . . . 735
8.152.1 Detailed Description . . . 735
8.153 mln::fun::vv2b::eq< L, R > Struct Template Reference . . . 736
8.153.1 Detailed Description . . . 736
8.154 mln::fun::vv2b::ge< L, R > Struct Template Reference . . . 737
8.154.1 Detailed Description . . . 737
8.155 mln::fun::vv2b::gt< L, R > Struct Template Reference . . . 738
8.155.1 Detailed Description . . . 738
8.156 mln::fun::vv2b::implies< L, R > Struct Template Reference . . . 739
8.156.1 Detailed Description . . . 739
8.157 mln::fun::vv2b::le< L, R > Struct Template Reference . . . 740
8.157.1 Detailed Description . . . 740
8.158 mln::fun::vv2b::lt< L, R > Struct Template Reference . . . 741
8.158.1 Detailed Description . . . 741
8.159 mln::fun::vv2v::diff_abs< V > Struct Template Reference . . . 742
8.159.1 Detailed Description . . . 742
8.160 mln::fun::vv2v::land< L, R > Struct Template Reference . . . 743
8.160.1 Detailed Description . . . 743
8.161 mln::fun::vv2v::land_not< L, R > Struct Template Reference . . . 744
8.161.1 Detailed Description . . . 744
8.162 mln::fun::vv2v::lor< L, R > Struct Template Reference . . . 745
8.162.1 Detailed Description . . . 745
8.163 mln::fun::vv2v::lxor< L, R > Struct Template Reference . . . 746
8.163.1 Detailed Description . . . 746
8.164 mln::fun::vv2v::max< V > Struct Template Reference . . . 747
8.164.1 Detailed Description . . . 747
8.165 mln::fun::vv2v::min< L, R > Struct Template Reference . . . 748
8.165.1 Detailed Description . . . 748
8.166 mln::fun::vv2v::vec< V > Struct Template Reference . . . 749
8.166.1 Detailed Description . . . 749
8.167 mln::fun::x2p::closest_point< P > Struct Template Reference . . . 750

8.167.1 Detailed Description . . . 750
8.168 mln::fun::x2v::bilinear< I > Struct Template Reference . . . 751
8.168.1 Detailed Description . . . 751
8.168.2 Member Function Documentation . . . 751
8.168.2.1 operator() . . . 751
8.168.2.2 operator() . . . 751
8.169 mln::fun::x2v::trilinear< I > Struct Template Reference . . . 752
8.169.1 Detailed Description . . . 752
8.170 mln::fun::x2x::composed< T2, T1 > Struct Template Reference . . . 753
8.170.1 Detailed Description . . . 753
8.170.2 Constructor & Destructor Documentation . . . 753
8.170.2.1 composed . . . 753
8.170.2.2 composed . . . 753
8.171 mln::fun::x2x::linear< I > Struct Template Reference . . . 754
8.171.1 Detailed Description . . . 754
8.171.2 Constructor & Destructor Documentation . . . 754
8.171.2.1 linear . . . 754
8.171.3 Member Function Documentation . . . 754
8.171.3.1 operator() . . . 754
8.171.4 Member Data Documentation . . . 755
8.171.4.1 ima . . . 755
8.172 mln::fun::x2x::rotation< n, C > Struct Template Reference . . . 756
8.172.1 Detailed Description . . . 757
8.172.2 Member Typedef Documentation . . . 757
8.172.2.1 invert . . . 757
8.172.3 Constructor & Destructor Documentation . . . 757
8.172.3.1 rotation . . . 757
8.172.3.2 rotation . . . 757
8.172.3.3 rotation . . . 757
8.172.3.4 rotation . . . 758
8.172.4 Member Function Documentation . . . 758
8.172.4.1 inv . . . 758
8.172.4.2 operator() . . . 758
8.172.4.3 set_alpha . . . 758
8.172.4.4 set_axis . . . 758
8.173 mln::fun::x2x::translation< n, C > Struct Template Reference . . . 759

8.173.1 Detailed Description . . . 760
8.173.2 Member Typedef Documentation . . . 760
8.173.2.1 invert . . . 760
8.173.3 Constructor & Destructor Documentation . . . 760
8.173.3.1 translation . . . 760
8.173.3.2 translation . . . 760
8.173.4 Member Function Documentation . . . 760
8.173.4.1 inv . . . 760
8.173.4.2 operator() . . . 761
8.173.4.3 set_t . . . 761
8.173.4.4 t . . . 761
8.174 mln::fun_image< F, I > Struct Template Reference . . . 762
8.174.1 Detailed Description . . . 762
8.174.2 Member Typedef Documentation . . . 763
8.174.2.1 lvalue . . . 763
8.174.2.2 rvalue . . . 763
8.174.2.3 skeleton . . . 763
8.174.2.4 value . . . 763
8.174.3 Constructor & Destructor Documentation . . . 763
8.174.3.1 fun_image . . . 763
8.174.3.2 fun_image . . . 763
8.174.3.3 fun_image . . . 763
8.174.4 Member Function Documentation . . . 763
8.174.4.1 operator() . . . 763
8.174.4.2 operator() . . . 763
8.175 mln::Function< E > Struct Template Reference . . . 764
8.175.1 Detailed Description . . . 764
8.175.2 Constructor & Destructor Documentation . . . 764
8.175.2.1 Function . . . 764
8.176 mln::Function< void > Struct Template Reference . . . 765
8.176.1 Detailed Description . . . 765
8.177 mln::Function_v2b< E > Struct Template Reference . . . 766
8.177.1 Detailed Description . . . 766
8.178 mln::Function_v2v< E > Struct Template Reference . . . 767
8.178.1 Detailed Description . . . 767
8.179 mln::Function_vv2b< E > Struct Template Reference . . . 768

8.179.1 Detailed Description . . . 768
8.180 mln::Function_vv2v< E > Struct Template Reference . . . 769
8.180.1 Detailed Description . . . 769
8.181 mln::fwd_pixter1d< I > Class Template Reference . . . 770
8.181.1 Detailed Description . . . 770
8.181.2 Member Typedef Documentation . . . 770
8.181.2.1 image . . . 770
8.181.3 Constructor & Destructor Documentation . . . 770
8.181.3.1 fwd_pixter1d . . . 770
8.181.4 Member Function Documentation . . . 771
8.181.4.1 next . . . 771
8.182 mln::fwd_pixter2d< I > Class Template Reference . . . 772
8.182.1 Detailed Description . . . 772
8.182.2 Member Typedef Documentation . . . 772
8.182.2.1 image . . . 772
8.182.3 Constructor & Destructor Documentation . . . 772
8.182.3.1 fwd_pixter2d . . . 772
8.182.4 Member Function Documentation . . . 773
8.182.4.1 next . . . 773
8.183 mln::fwd_pixter3d< I > Class Template Reference . . . 774
8.183.1 Detailed Description . . . 774
8.183.2 Member Typedef Documentation . . . 774
8.183.2.1 image . . . 774
8.183.3 Constructor & Destructor Documentation . . . 774
8.183.3.1 fwd_pixter3d . . . 774
8.183.4 Member Function Documentation . . . 775
8.183.4.1 next . . . 775
8.184 mln::Gdpoint< E > Struct Template Reference . . . 776
8.184.1 Detailed Description . . . 776
8.185 mln::Gdpoint< void > Struct Template Reference . . . 777
8.185.1 Detailed Description . . . 777
8.186 mln::Generalized_Pixel< E > Struct Template Reference . . . 778
8.186.1 Detailed Description . . . 778
8.187 mln::geom::complex_geometry< D, P > Class Template Reference . . . 779
8.187.1 Detailed Description . . . 779
8.187.2 Constructor & Destructor Documentation . . . 779

8.187.2.1 complex_geometry . . . 779
8.187.3 Member Function Documentation . . . 780
8.187.3.1 add_location . . . 780
8.187.3.2 operator() . . . 780
8.188 mln::Gpoint< E > Struct Template Reference . . . 781
8.188.1 Detailed Description . . . 782
8.188.2 Friends And Related Function Documentation . . . 782
8.188.2.1 operator+ . . . 782
8.188.2.2 operator+= . . . 783
8.188.2.3 operator- . . . 783
8.188.2.4 operator-= . . . 784
8.188.2.5 operator/ . . . 784
8.188.2.6 operator<< . . . 784
8.188.2.7 operator== . . . 785
8.189 mln::Graph< E > Struct Template Reference . . . 786
8.189.1 Detailed Description . . . 786
8.189.2 Member Function Documentation . . . 786
8.189.2.1 invalidate . . . 786
8.189.2.2 is_valid . . . 787
8.190 mln::graph::attribute::card_t Struct Reference . . . 788
8.190.1 Detailed Description . . . 788
8.190.2 Member Typedef Documentation . . . 788
8.190.2.1 result . . . 788
8.191 mln::graph::attribute::representative_t Struct Reference . . . 789
8.191.1 Detailed Description . . . 789
8.191.2 Member Typedef Documentation . . . 789
8.191.2.1 result . . . 789
8.192 mln::graph_elt_neighborhood< G, S > Struct Template Reference . . . 790
8.192.1 Detailed Description . . . 791
8.192.2 Member Typedef Documentation . . . 791
8.192.2.1 bkd_niter . . . 791
8.192.2.2 fwd_niter . . . 791
8.192.2.3 niter . . . 791
8.192.3 Member Function Documentation . . . 791
8.192.3.1 change_window . . . 791
8.192.3.2 win . . . 791

8.193 mln::graph_elt_neighborhood_if< G, S, I > Struct Template Reference 792
8.193.1 Detailed Description 793
8.193.2 Member Typedef Documentation 793
8.193.2.1 bkd_niter 793
8.193.2.2 fwd_niter 793
8.193.2.3 niter 793
8.193.3 Constructor & Destructor Documentation 793
8.193.3.1 graph_elt_neighborhood_if 793
8.193.3.2 graph_elt_neighborhood_if 793
8.193.4 Member Function Documentation 793
8.193.4.1 change_window 793
8.193.4.2 mask 794
8.193.4.3 win 794
8.194 mln::graph_elt_window< G, S > Class Template Reference 795
8.194.1 Detailed Description 796
8.194.2 Member Typedef Documentation 796
8.194.2.1 bkd_qiter 796
8.194.2.2 fwd_qiter 796
8.194.2.3 psite 796
8.194.2.4 qiter 797
8.194.2.5 site 797
8.194.2.6 target 797
8.194.3 Member Function Documentation 797
8.194.3.1 delta 797
8.194.3.2 is_centered 797
8.194.3.3 is_empty 797
8.194.3.4 is_symmetric 797
8.194.3.5 is_valid 797
8.194.3.6 sym 798
8.195 mln::graph_elt_window_if< G, S, I > Class Template Reference 799
8.195.1 Detailed Description 800
8.195.2 Member Typedef Documentation 800
8.195.2.1 bkd_qiter 800
8.195.2.2 fwd_qiter 801
8.195.2.3 mask_t 801
8.195.2.4 psite 801

8.195.2.5 qiter . . . 801
8.195.2.6 site . . . 801
8.195.2.7 target . . . 801
8.195.3 Constructor & Destructor Documentation . . . 801
8.195.3.1 graph_elt_window_if . . . 801
8.195.3.2 graph_elt_window_if . . . 802
8.195.4 Member Function Documentation . . . 802
8.195.4.1 change_mask . . . 802
8.195.4.2 delta . . . 802
8.195.4.3 is_centered . . . 802
8.195.4.4 is_empty . . . 802
8.195.4.5 is_symmetric . . . 802
8.195.4.6 is_valid . . . 802
8.195.4.7 mask . . . 803
8.195.4.8 sym . . . 803
8.196 mln::graph_window_base< P, E > Class Template Reference . . . 804
8.196.1 Detailed Description . . . 804
8.196.2 Member Typedef Documentation . . . 804
8.196.2.1 site . . . 804
8.196.3 Member Function Documentation . . . 805
8.196.3.1 delta . . . 805
8.196.3.2 is_centered . . . 805
8.196.3.3 is_empty . . . 805
8.196.3.4 is_symmetric . . . 805
8.196.3.5 is_valid . . . 805
8.196.3.6 sym . . . 805
8.197 mln::graph_window_if_piter< S, W, I > Class Template Reference . . . 806
8.197.1 Detailed Description . . . 806
8.197.2 Member Typedef Documentation . . . 806
8.197.2.1 P . . . 806
8.197.3 Constructor & Destructor Documentation . . . 807
8.197.3.1 graph_window_if_piter . . . 807
8.197.4 Member Function Documentation . . . 807
8.197.4.1 element . . . 807
8.197.4.2 id . . . 807
8.197.4.3 next . . . 807

8.198 mln::graph_window_piter< S, W, I > Class Template Reference . . . 808
8.198.1 Detailed Description . . . 808
8.198.2 Member Typedef Documentation . . . 808
8.198.2.1 P . . . 808
8.198.3 Constructor & Destructor Documentation . . . 808
8.198.3.1 graph_window_piter . . . 808
8.198.4 Member Function Documentation . . . 809
8.198.4.1 element . . . 809
8.198.4.2 id . . . 809
8.198.4.3 next . . . 809
8.199 mln::hexa< I > Struct Template Reference . . . 810
8.199.1 Detailed Description . . . 811
8.199.2 Member Typedef Documentation . . . 811
8.199.2.1 bkd_piter . . . 811
8.199.2.2 fwd_piter . . . 811
8.199.2.3 lvalue . . . 811
8.199.2.4 psite . . . 812
8.199.2.5 rvalue . . . 812
8.199.2.6 skeleton . . . 812
8.199.2.7 value . . . 812
8.199.3 Constructor & Destructor Documentation . . . 812
8.199.3.1 hexa . . . 812
8.199.3.2 hexa . . . 812
8.199.4 Member Function Documentation . . . 812
8.199.4.1 domain . . . 812
8.199.4.2 has . . . 812
8.199.4.3 operator() . . . 812
8.199.4.4 operator() . . . 813
8.200 mln::histo::array< T > Struct Template Reference . . . 814
8.200.1 Detailed Description . . . 814
8.201 mln::Image< E > Struct Template Reference . . . 815
8.201.1 Detailed Description . . . 817
8.202 mln::image1d< T > Struct Template Reference . . . 818
8.202.1 Detailed Description . . . 819
8.202.2 Member Typedef Documentation . . . 819
8.202.2.1 lvalue . . . 819

8.202.2.2 rvalue . . . 819
8.202.2.3 skeleton . . . 819
8.202.2.4 value . . . 820
8.202.3 Constructor & Destructor Documentation . . . 820
8.202.3.1 image1d . . . 820
8.202.3.2 image1d . . . 820
8.202.3.3 image1d . . . 820
8.202.4 Member Function Documentation . . . 820
8.202.4.1 bbox . . . 820
8.202.4.2 border . . . 820
8.202.4.3 buffer . . . 820
8.202.4.4 buffer . . . 820
8.202.4.5 delta_index . . . 820
8.202.4.6 domain . . . 821
8.202.4.7 element . . . 821
8.202.4.8 element . . . 821
8.202.4.9 has . . . 821
8.202.4.10 nelements . . . 821
8.202.4.11 ninds . . . 821
8.202.4.12 operator() . . . 821
8.202.4.13 operator() . . . 821
8.202.4.14 point_at_index . . . 822
8.203 mln::image2d< T > Class Template Reference . . . 823
8.203.1 Detailed Description . . . 824
8.203.2 Member Typedef Documentation . . . 824
8.203.2.1 lvalue . . . 824
8.203.2.2 rvalue . . . 824
8.203.2.3 skeleton . . . 825
8.203.2.4 value . . . 825
8.203.3 Constructor & Destructor Documentation . . . 825
8.203.3.1 image2d . . . 825
8.203.3.2 image2d . . . 825
8.203.3.3 image2d . . . 825
8.203.4 Member Function Documentation . . . 825
8.203.4.1 bbox . . . 825
8.203.4.2 border . . . 825

8.203.4.3 buffer . . . 825
8.203.4.4 buffer . . . 825
8.203.4.5 delta_index . . . 826
8.203.4.6 domain . . . 826
8.203.4.7 element . . . 826
8.203.4.8 element . . . 826
8.203.4.9 has . . . 826
8.203.4.10 ncols . . . 826
8.203.4.11 nelements . . . 826
8.203.4.12 nrows . . . 826
8.203.4.13 operator() . . . 826
8.203.4.14 operator() . . . 827
8.203.4.15 point_at_index . . . 827
8.204 mln::image2d_h< V > Struct Template Reference . . . 828
8.204.1 Detailed Description . . . 829
8.204.2 Member Typedef Documentation . . . 829
8.204.2.1 bkd_piter . . . 829
8.204.2.2 fwd_piter . . . 829
8.204.2.3 lvalue . . . 829
8.204.2.4 psite . . . 829
8.204.2.5 rvalue . . . 829
8.204.2.6 skeleton . . . 830
8.204.2.7 value . . . 830
8.204.3 Constructor & Destructor Documentation . . . 830
8.204.3.1 image2d_h . . . 830
8.204.4 Member Function Documentation . . . 830
8.204.4.1 domain . . . 830
8.204.4.2 has . . . 830
8.204.4.3 operator() . . . 830
8.204.4.4 operator() . . . 830
8.205 mln::image3d< T > Struct Template Reference . . . 831
8.205.1 Detailed Description . . . 832
8.205.2 Member Typedef Documentation . . . 832
8.205.2.1 lvalue . . . 832
8.205.2.2 rvalue . . . 833
8.205.2.3 skeleton . . . 833

8.205.2.4 value . . . 833
8.205.3 Constructor & Destructor Documentation . . . 833
8.205.3.1 image3d . . . 833
8.205.3.2 image3d . . . 833
8.205.3.3 image3d . . . 833
8.205.4 Member Function Documentation . . . 833
8.205.4.1 bbox . . . 833
8.205.4.2 border . . . 833
8.205.4.3 buffer . . . 833
8.205.4.4 buffer . . . 834
8.205.4.5 delta_index . . . 834
8.205.4.6 domain . . . 834
8.205.4.7 element . . . 834
8.205.4.8 element . . . 834
8.205.4.9 has . . . 834
8.205.4.10 ncols . . . 834
8.205.4.11 nelements . . . 834
8.205.4.12 nrows . . . 835
8.205.4.13 nslices . . . 835
8.205.4.14 operator() . . . 835
8.205.4.15 operator() . . . 835
8.205.4.16 point_at_index . . . 835
8.206 mln::interpolated< I, F > Struct Template Reference . . . 836
8.206.1 Detailed Description . . . 836
8.206.2 Member Typedef Documentation . . . 836
8.206.2.1 lvalue . . . 836
8.206.2.2 psite . . . 837
8.206.2.3 rvalue . . . 837
8.206.2.4 skeleton . . . 837
8.206.2.5 value . . . 837
8.206.3 Constructor & Destructor Documentation . . . 837
8.206.3.1 interpolated . . . 837
8.206.4 Member Function Documentation . . . 837
8.206.4.1 has . . . 837
8.206.4.2 is_valid . . . 837
8.207 mln::Iterator< E > Struct Template Reference . . . 838

8.207.1 Detailed Description . . . 838
8.207.2 Member Function Documentation . . . 838
8.207.2.1 next . . . 838
8.208 mln::labeled_image< I > Class Template Reference . . . 839
8.208.1 Detailed Description . . . 840
8.208.2 Member Typedef Documentation . . . 840
8.208.2.1 bbox_t . . . 840
8.208.2.2 skeleton . . . 840
8.208.3 Constructor & Destructor Documentation . . . 840
8.208.3.1 labeled_image . . . 840
8.208.3.2 labeled_image . . . 840
8.208.4 Member Function Documentation . . . 840
8.208.4.1 bbox . . . 840
8.208.4.2 bboxes . . . 841
8.208.4.3 nlabels . . . 841
8.208.4.4 relabel . . . 841
8.208.4.5 relabel . . . 841
8.208.4.6 subdomain . . . 841
8.209 mln::lazy_image< I, F, B > Struct Template Reference . . . 842
8.209.1 Detailed Description . . . 843
8.209.2 Member Typedef Documentation . . . 843
8.209.2.1 lvalue . . . 843
8.209.2.2 rvalue . . . 843
8.209.2.3 skeleton . . . 843
8.209.3 Constructor & Destructor Documentation . . . 843
8.209.3.1 lazy_image . . . 843
8.209.3.2 lazy_image . . . 843
8.209.4 Member Function Documentation . . . 843
8.209.4.1 domain . . . 843
8.209.4.2 has . . . 844
8.209.4.3 operator() . . . 844
8.209.4.4 operator() . . . 844
8.209.4.5 operator() . . . 844
8.209.4.6 operator() . . . 844
8.210 mln::Literal< E > Struct Template Reference . . . 845
8.210.1 Detailed Description . . . 846

8.211 mln::literal::black_t Struct Reference 847
8.211.1 Detailed Description 847
8.212 mln::literal::blue_t Struct Reference 848
8.212.1 Detailed Description 848
8.213 mln::literal::brown_t Struct Reference 849
8.213.1 Detailed Description 849
8.214 mln::literal::cyan_t Struct Reference 850
8.214.1 Detailed Description 850
8.215 mln::literal::green_t Struct Reference 851
8.215.1 Detailed Description 851
8.216 mln::literal::identity_t Struct Reference 852
8.216.1 Detailed Description 852
8.217 mln::literal::light_gray_t Struct Reference 853
8.217.1 Detailed Description 853
8.218 mln::literal::lime_t Struct Reference 854
8.218.1 Detailed Description 854
8.219 mln::literal::magenta_t Struct Reference 855
8.219.1 Detailed Description 855
8.220 mln::literal::max_t Struct Reference 856
8.220.1 Detailed Description 856
8.221 mln::literal::min_t Struct Reference 857
8.221.1 Detailed Description 857
8.222 mln::literal::olive_t Struct Reference 858
8.222.1 Detailed Description 858
8.223 mln::literal::one_t Struct Reference 859
8.223.1 Detailed Description 859
8.224 mln::literal::orange_t Struct Reference 860
8.224.1 Detailed Description 860
8.225 mln::literal::origin_t Struct Reference 861
8.225.1 Detailed Description 861
8.226 mln::literal::pink_t Struct Reference 862
8.226.1 Detailed Description 862
8.227 mln::literal::purple_t Struct Reference 863
8.227.1 Detailed Description 863
8.228 mln::literal::red_t Struct Reference 864
8.228.1 Detailed Description 864

8.229 mln::literal::teal_t Struct Reference . . . 865
8.229.1 Detailed Description . . . 865
8.230 mln::literal::violet_t Struct Reference . . . 866
8.230.1 Detailed Description . . . 866
8.231 mln::literal::white_t Struct Reference . . . 867
8.231.1 Detailed Description . . . 867
8.232 mln::literal::yellow_t Struct Reference . . . 868
8.232.1 Detailed Description . . . 868
8.233 mln::literal::zero_t Struct Reference . . . 869
8.233.1 Detailed Description . . . 869
8.234 mln::Mesh< E > Struct Template Reference . . . 870
8.234.1 Detailed Description . . . 870
8.235 mln::Meta_Accumulator< E > Struct Template Reference . . . 871
8.235.1 Detailed Description . . . 872
8.236 mln::Meta_Function< E > Struct Template Reference . . . 873
8.236.1 Detailed Description . . . 873
8.237 mln::Meta_Function_v2v< E > Struct Template Reference . . . 874
8.237.1 Detailed Description . . . 874
8.238 mln::Meta_Function_vv2v< E > Struct Template Reference . . . 875
8.238.1 Detailed Description . . . 875
8.239 mln::metal::ands< E1, E2, E3, E4, E5, E6, E7, E8 > Struct Template Reference . . . 876
8.239.1 Detailed Description . . . 876
8.240 mln::metal::converts_to< T, U > Struct Template Reference . . . 877
8.240.1 Detailed Description . . . 877
8.241 mln::metal::equal< T1, T2 > Struct Template Reference . . . 878
8.241.1 Detailed Description . . . 878
8.242 mln::metal::goes_to< T, U > Struct Template Reference . . . 879
8.242.1 Detailed Description . . . 879
8.243 mln::metal::is< T, U > Struct Template Reference . . . 880
8.243.1 Detailed Description . . . 880
8.244 mln::metal::is_a< T, M > Struct Template Reference . . . 881
8.244.1 Detailed Description . . . 881
8.245 mln::metal::is_not< T, U > Struct Template Reference . . . 882
8.245.1 Detailed Description . . . 882
8.246 mln::metal::is_not_a< T, M > Struct Template Reference . . . 883
8.246.1 Detailed Description . . . 883

8.247 mln::morpho::attribute::card< I > Class Template Reference . . . 884
8.247.1 Detailed Description . . . 884
8.247.2 Member Function Documentation . . . 884
8.247.2.1 init . . . 884
8.247.2.2 is_valid . . . 884
8.247.2.3 take_as_init . . . 884
8.247.2.4 take_n_times . . . 885
8.247.2.5 to_result . . . 885
8.248 mln::morpho::attribute::count_adjacent_vertices< I > Struct Template Reference . . . 886
8.248.1 Detailed Description . . . 886
8.248.2 Member Function Documentation . . . 886
8.248.2.1 init . . . 886
8.248.2.2 is_valid . . . 886
8.248.2.3 take_as_init . . . 887
8.248.2.4 take_n_times . . . 887
8.248.2.5 to_result . . . 887
8.249 mln::morpho::attribute::height< I > Struct Template Reference . . . 888
8.249.1 Detailed Description . . . 888
8.249.2 Member Function Documentation . . . 888
8.249.2.1 base_level . . . 888
8.249.2.2 init . . . 888
8.249.2.3 is_valid . . . 889
8.249.2.4 take_as_init . . . 889
8.249.2.5 take_n_times . . . 889
8.249.2.6 to_result . . . 889
8.250 mln::morpho::attribute::sharpness< I > Struct Template Reference . . . 890
8.250.1 Detailed Description . . . 890
8.250.2 Member Function Documentation . . . 890
8.250.2.1 area . . . 890
8.250.2.2 height . . . 891
8.250.2.3 init . . . 891
8.250.2.4 is_valid . . . 891
8.250.2.5 take_as_init . . . 891
8.250.2.6 take_n_times . . . 891
8.250.2.7 to_result . . . 891
8.250.2.8 volume . . . 891

8.251 mln::morpho::attribute::sum< I, S > Class Template Reference . . . 892
8.251.1 Detailed Description . . . 892
8.251.2 Member Function Documentation . . . 892
8.251.2.1 init . . . 892
8.251.2.2 is_valid . . . 893
8.251.2.3 set_value . . . 893
8.251.2.4 take_as_init . . . 893
8.251.2.5 take_n_times . . . 893
8.251.2.6 to_result . . . 893
8.251.2.7 untake . . . 893
8.252 mln::morpho::attribute::volume< I > Struct Template Reference . . . 894
8.252.1 Detailed Description . . . 894
8.252.2 Member Function Documentation . . . 894
8.252.2.1 area . . . 894
8.252.2.2 init . . . 894
8.252.2.3 is_valid . . . 895
8.252.2.4 take_as_init . . . 895
8.252.2.5 take_n_times . . . 895
8.252.2.6 to_result . . . 895
8.253 mln::neighb< W > Class Template Reference . . . 896
8.253.1 Detailed Description . . . 896
8.253.2 Member Typedef Documentation . . . 897
8.253.2.1 bkd_niter . . . 897
8.253.2.2 fwd_niter . . . 897
8.253.2.3 niter . . . 897
8.253.3 Constructor & Destructor Documentation . . . 897
8.253.3.1 neighb . . . 897
8.253.3.2 neighb . . . 897
8.253.4 Member Function Documentation . . . 897
8.253.4.1 change_window . . . 897
8.253.4.2 win . . . 897
8.254 mln::Neighborhood< E > Struct Template Reference . . . 898
8.254.1 Detailed Description . . . 898
8.255 mln::Neighborhood< void > Struct Template Reference . . . 899
8.255.1 Detailed Description . . . 899
8.256 mln::Object< E > Struct Template Reference . . . 900

8.256.1 Detailed Description . . . 900
8.257 mln::p2p_image< I, F > Struct Template Reference . . . 901
8.257.1 Detailed Description . . . 901
8.257.2 Member Typedef Documentation . . . 901
8.257.2.1 skeleton . . . 901
8.257.3 Constructor & Destructor Documentation . . . 902
8.257.3.1 p2p_image . . . 902
8.257.3.2 p2p_image . . . 902
8.257.4 Member Function Documentation . . . 902
8.257.4.1 domain . . . 902
8.257.4.2 fun . . . 902
8.257.4.3 operator() . . . 902
8.257.4.4 operator() . . . 902
8.258 mln::p_array< P > Class Template Reference . . . 903
8.258.1 Detailed Description . . . 904
8.258.2 Member Typedef Documentation . . . 904
8.258.2.1 bkd_piter . . . 904
8.258.2.2 element . . . 905
8.258.2.3 fwd_piter . . . 905
8.258.2.4 i_element . . . 905
8.258.2.5 piter . . . 905
8.258.2.6 psite . . . 905
8.258.3 Constructor & Destructor Documentation . . . 905
8.258.3.1 p_array . . . 905
8.258.3.2 p_array . . . 905
8.258.4 Member Function Documentation . . . 905
8.258.4.1 append . . . 905
8.258.4.2 append . . . 905
8.258.4.3 change . . . 906
8.258.4.4 clear . . . 906
8.258.4.5 has . . . 906
8.258.4.6 has . . . 906
8.258.4.7 insert . . . 906
8.258.4.8 is_valid . . . 906
8.258.4.9 memory_size . . . 906
8.258.4.10 nsites . . . 906

8.258.4.1 loperator[] . . . 906
8.258.4.12operator[] . . . 907
8.258.4.13operator[] . . . 907
8.258.4.14reserve . . . 907
8.258.4.15std_vector . . . 907
8.259 mln::p_centered< W > Class Template Reference . . . 908
8.259.1 Detailed Description . . . 909
8.259.2 Member Typedef Documentation . . . 909
8.259.2.1 bkd_piter . . . 909
8.259.2.2 element . . . 909
8.259.2.3 fwd_piter . . . 909
8.259.2.4 piter . . . 909
8.259.2.5 psite . . . 909
8.259.2.6 site . . . 909
8.259.3 Constructor & Destructor Documentation . . . 909
8.259.3.1 p_centered . . . 909
8.259.3.2 p_centered . . . 910
8.259.4 Member Function Documentation . . . 910
8.259.4.1 center . . . 910
8.259.4.2 has . . . 910
8.259.4.3 is_valid . . . 910
8.259.4.4 memory_size . . . 910
8.259.4.5 window . . . 910
8.260 mln::p_complex< D, G > Class Template Reference . . . 911
8.260.1 Detailed Description . . . 912
8.260.2 Member Typedef Documentation . . . 912
8.260.2.1 bkd_piter . . . 912
8.260.2.2 element . . . 912
8.260.2.3 fwd_piter . . . 912
8.260.2.4 piter . . . 912
8.260.2.5 psite . . . 912
8.260.3 Constructor & Destructor Documentation . . . 913
8.260.3.1 p_complex . . . 913
8.260.4 Member Function Documentation . . . 913
8.260.4.1 cplx . . . 913
8.260.4.2 cplx . . . 913

8.260.4.3 geom . . . 913
8.260.4.4 has . . . 913
8.260.4.5 is_valid . . . 913
8.260.4.6 nfaces . . . 914
8.260.4.7 nfaces_of_dim . . . 914
8.260.4.8 nsites . . . 914
8.261 mln::p_edges< G, F > Class Template Reference . . . 915
8.261.1 Detailed Description . . . 916
8.261.2 Member Typedef Documentation . . . 916
8.261.2.1 bkd_piter . . . 916
8.261.2.2 edge . . . 917
8.261.2.3 element . . . 917
8.261.2.4 fun_t . . . 917
8.261.2.5 fwd_piter . . . 917
8.261.2.6 graph_element . . . 917
8.261.2.7 graph_t . . . 917
8.261.2.8 piter . . . 917
8.261.2.9 psite . . . 917
8.261.3 Constructor & Destructor Documentation . . . 917
8.261.3.1 p_edges . . . 917
8.261.3.2 p_edges . . . 918
8.261.3.3 p_edges . . . 918
8.261.3.4 p_edges . . . 918
8.261.4 Member Function Documentation . . . 918
8.261.4.1 function . . . 918
8.261.4.2 graph . . . 918
8.261.4.3 has . . . 919
8.261.4.4 has . . . 919
8.261.4.5 invalidate . . . 919
8.261.4.6 is_valid . . . 919
8.261.4.7 memory_size . . . 919
8.261.4.8 nedges . . . 919
8.261.4.9 nsites . . . 919
8.262 mln::p_faces< N, D, P > Struct Template Reference . . . 920
8.262.1 Detailed Description . . . 921
8.262.2 Member Typedef Documentation . . . 921

8.262.2.1 bkd_piter . . . 921
8.262.2.2 element . . . 921
8.262.2.3 fwd_piter . . . 921
8.262.2.4 piter . . . 921
8.262.2.5 psite . . . 921
8.262.3 Constructor & Destructor Documentation . . . 921
8.262.3.1 p_faces . . . 921
8.262.3.2 p_faces . . . 922
8.262.4 Member Function Documentation . . . 922
8.262.4.1 cplx . . . 922
8.262.4.2 cplx . . . 922
8.262.4.3 is_valid . . . 922
8.262.4.4 nfaces . . . 922
8.262.4.5 nsites . . . 922
8.263 mln::p_graph_piter< S, I > Class Template Reference . . . 923
8.263.1 Detailed Description . . . 923
8.263.2 Constructor & Destructor Documentation . . . 923
8.263.2.1 p_graph_piter . . . 923
8.263.3 Member Function Documentation . . . 923
8.263.3.1 graph . . . 923
8.263.3.2 id . . . 924
8.263.3.3 mln_q_subject . . . 924
8.263.3.4 next . . . 924
8.264 mln::p_if< S, F > Class Template Reference . . . 925
8.264.1 Detailed Description . . . 926
8.264.2 Member Typedef Documentation . . . 926
8.264.2.1 bkd_piter . . . 926
8.264.2.2 element . . . 926
8.264.2.3 fwd_piter . . . 926
8.264.2.4 piter . . . 926
8.264.2.5 psite . . . 926
8.264.3 Constructor & Destructor Documentation . . . 926
8.264.3.1 p_if . . . 926
8.264.3.2 p_if . . . 926
8.264.4 Member Function Documentation . . . 927
8.264.4.1 has . . . 927

8.264.4.2 is_valid . . . 927
8.264.4.3 memory_size . . . 927
8.264.4.4 overset . . . 927
8.264.4.5 pred . . . 927
8.264.4.6 predicate . . . 927
8.265 mln::p_image< I > Class Template Reference . . . 928
8.265.1 Detailed Description . . . 929
8.265.2 Member Typedef Documentation . . . 929
8.265.2.1 bkd_piter . . . 929
8.265.2.2 element . . . 929
8.265.2.3 fwd_piter . . . 929
8.265.2.4 i_element . . . 929
8.265.2.5 piter . . . 929
8.265.2.6 psite . . . 930
8.265.2.7 r_element . . . 930
8.265.2.8 S . . . 930
8.265.3 Constructor & Destructor Documentation . . . 930
8.265.3.1 p_image . . . 930
8.265.3.2 p_image . . . 930
8.265.4 Member Function Documentation . . . 930
8.265.4.1 clear . . . 930
8.265.4.2 has . . . 930
8.265.4.3 insert . . . 930
8.265.4.4 is_valid . . . 931
8.265.4.5 memory_size . . . 931
8.265.4.6 nsites . . . 931
8.265.4.7 operator typename internal::p_image_site_set< I >::ret . . . 931
8.265.4.8 remove . . . 931
8.265.4.9 toggle . . . 931
8.266 mln::p_indexed_bkd_piter< S > Class Template Reference . . . 932
8.266.1 Detailed Description . . . 932
8.266.2 Constructor & Destructor Documentation . . . 932
8.266.2.1 p_indexed_bkd_piter . . . 932
8.266.2.2 p_indexed_bkd_piter . . . 932
8.266.3 Member Function Documentation . . . 932
8.266.3.1 index . . . 932

8.267 mln::p_indexed_fwd_piter< S > Class Template Reference . . . 933
8.267.1 Detailed Description . . . 933
8.267.2 Constructor & Destructor Documentation . . . 933
8.267.2.1 p_indexed_fwd_piter . . . 933
8.267.2.2 p_indexed_fwd_piter . . . 933
8.267.3 Member Function Documentation . . . 933
8.267.3.1 index . . . 933
8.268 mln::p_indexed_psite< S > Class Template Reference . . . 934
8.268.1 Detailed Description . . . 934
8.269 mln::p_key< K, P > Class Template Reference . . . 935
8.269.1 Detailed Description . . . 936
8.269.2 Member Typedef Documentation . . . 937
8.269.2.1 bkd_piter . . . 937
8.269.2.2 element . . . 937
8.269.2.3 fwd_piter . . . 937
8.269.2.4 i_element . . . 937
8.269.2.5 piter . . . 937
8.269.2.6 psite . . . 937
8.269.2.7 r_element . . . 937
8.269.3 Constructor & Destructor Documentation . . . 937
8.269.3.1 p_key . . . 937
8.269.4 Member Function Documentation . . . 937
8.269.4.1 change_key . . . 937
8.269.4.2 change_keys . . . 938
8.269.4.3 clear . . . 938
8.269.4.4 exists_key . . . 938
8.269.4.5 has . . . 938
8.269.4.6 has . . . 938
8.269.4.7 insert . . . 938
8.269.4.8 insert . . . 938
8.269.4.9 is_valid . . . 938
8.269.4.10 key . . . 939
8.269.4.11 keys . . . 939
8.269.4.12 memory_size . . . 939
8.269.4.13 nsites . . . 939
8.269.4.14 operator() . . . 939

8.269.4.15 remove . . . 939
8.269.4.16 remove_key . . . 939
8.270 mln::p_line2d Class Reference . . . 940
8.270.1 Detailed Description . . . 941
8.270.2 Member Typedef Documentation . . . 941
8.270.2.1 bkd_piter . . . 941
8.270.2.2 element . . . 941
8.270.2.3 fwd_piter . . . 941
8.270.2.4 piter . . . 941
8.270.2.5 psite . . . 941
8.270.2.6 q_box . . . 942
8.270.3 Constructor & Destructor Documentation . . . 942
8.270.3.1 p_line2d . . . 942
8.270.3.2 p_line2d . . . 942
8.270.4 Member Function Documentation . . . 942
8.270.4.1 bbox . . . 942
8.270.4.2 begin . . . 942
8.270.4.3 end . . . 942
8.270.4.4 has . . . 942
8.270.4.5 has . . . 942
8.270.4.6 is_valid . . . 943
8.270.4.7 memory_size . . . 943
8.270.4.8 nsites . . . 943
8.270.4.9 operator[] . . . 943
8.270.4.10 std_vector . . . 943
8.271 mln::p_mutable_array_of< S > Class Template Reference . . . 944
8.271.1 Detailed Description . . . 945
8.271.2 Member Typedef Documentation . . . 945
8.271.2.1 bkd_piter . . . 945
8.271.2.2 element . . . 945
8.271.2.3 fwd_piter . . . 945
8.271.2.4 i_element . . . 945
8.271.2.5 piter . . . 945
8.271.2.6 psite . . . 945
8.271.3 Constructor & Destructor Documentation . . . 946
8.271.3.1 p_mutable_array_of . . . 946

8.271.4 Member Function Documentation . . . 946
8.271.4.1 clear . . . 946
8.271.4.2 has . . . 946
8.271.4.3 insert . . . 946
8.271.4.4 is_valid . . . 946
8.271.4.5 memory_size . . . 946
8.271.4.6 nelements . . . 946
8.271.4.7 operator[] . . . 946
8.271.4.8 operator[] . . . 947
8.271.4.9 reserve . . . 947
8.272 mln::p_n_faces_bkd_piter< D, P > Class Template Reference . . . 948
8.272.1 Detailed Description . . . 948
8.272.2 Constructor & Destructor Documentation . . . 948
8.272.2.1 p_n_faces_bkd_piter . . . 948
8.272.3 Member Function Documentation . . . 948
8.272.3.1 n . . . 948
8.272.3.2 next . . . 948
8.273 mln::p_n_faces_fwd_piter< D, P > Class Template Reference . . . 950
8.273.1 Detailed Description . . . 950
8.273.2 Constructor & Destructor Documentation . . . 950
8.273.2.1 p_n_faces_fwd_piter . . . 950
8.273.3 Member Function Documentation . . . 950
8.273.3.1 n . . . 950
8.273.3.2 next . . . 950
8.274 mln::p_priority< P, Q > Class Template Reference . . . 952
8.274.1 Detailed Description . . . 953
8.274.2 Member Typedef Documentation . . . 953
8.274.2.1 bkd_piter . . . 953
8.274.2.2 element . . . 954
8.274.2.3 fwd_piter . . . 954
8.274.2.4 i_element . . . 954
8.274.2.5 piter . . . 954
8.274.2.6 psite . . . 954
8.274.3 Constructor & Destructor Documentation . . . 954
8.274.3.1 p_priority . . . 954
8.274.4 Member Function Documentation . . . 954

8.274.4.1 clear . . . 954
8.274.4.2 exists_priority . . . 954
8.274.4.3 front . . . 955
8.274.4.4 has . . . 955
8.274.4.5 highest_priority . . . 955
8.274.4.6 insert . . . 955
8.274.4.7 insert . . . 955
8.274.4.8 is_valid . . . 955
8.274.4.9 lowest_priority . . . 955
8.274.4.10 memory_size . . . 956
8.274.4.11 nsites . . . 956
8.274.4.12 operator() . . . 956
8.274.4.13 pop . . . 956
8.274.4.14 pop_front . . . 956
8.274.4.15 priorities . . . 957
8.274.4.16 push . . . 957
8.275 mln::p_queue< P > Class Template Reference . . . 958
8.275.1 Detailed Description . . . 959
8.275.2 Member Typedef Documentation . . . 959
8.275.2.1 bkd_piter . . . 959
8.275.2.2 element . . . 959
8.275.2.3 fwd_piter . . . 959
8.275.2.4 i_element . . . 960
8.275.2.5 piter . . . 960
8.275.2.6 psite . . . 960
8.275.3 Constructor & Destructor Documentation . . . 960
8.275.3.1 p_queue . . . 960
8.275.4 Member Function Documentation . . . 960
8.275.4.1 clear . . . 960
8.275.4.2 front . . . 960
8.275.4.3 has . . . 960
8.275.4.4 has . . . 960
8.275.4.5 insert . . . 960
8.275.4.6 is_valid . . . 961
8.275.4.7 memory_size . . . 961
8.275.4.8 nsites . . . 961

8.275.4.9 operator[] . . . 961
8.275.4.10 pop . . . 961
8.275.4.11 pop_front . . . 961
8.275.4.12 push . . . 961
8.275.4.13 std_deque . . . 961
8.276 mln::p_queue_fast< P > Class Template Reference . . . 962
8.276.1 Detailed Description . . . 963
8.276.2 Member Typedef Documentation . . . 963
8.276.2.1 bkd_piter . . . 963
8.276.2.2 element . . . 964
8.276.2.3 fwd_piter . . . 964
8.276.2.4 i_element . . . 964
8.276.2.5 piter . . . 964
8.276.2.6 psite . . . 964
8.276.3 Constructor & Destructor Documentation . . . 964
8.276.3.1 p_queue_fast . . . 964
8.276.4 Member Function Documentation . . . 964
8.276.4.1 clear . . . 964
8.276.4.2 compute_has . . . 964
8.276.4.3 front . . . 964
8.276.4.4 has . . . 965
8.276.4.5 has . . . 965
8.276.4.6 insert . . . 965
8.276.4.7 is_valid . . . 965
8.276.4.8 memory_size . . . 965
8.276.4.9 nsites . . . 965
8.276.4.10 operator[] . . . 965
8.276.4.11 pop . . . 965
8.276.4.12 pop_front . . . 966
8.276.4.13 purge . . . 966
8.276.4.14 push . . . 966
8.276.4.15 reserve . . . 966
8.276.4.16 std_vector . . . 966
8.277 mln::p_run< P > Class Template Reference . . . 967
8.277.1 Detailed Description . . . 968
8.277.2 Member Typedef Documentation . . . 968

8.277.2.1 bkd_piter . . . 968
8.277.2.2 element . . . 968
8.277.2.3 fwd_piter . . . 968
8.277.2.4 piter . . . 969
8.277.2.5 psite . . . 969
8.277.2.6 q_box . . . 969
8.277.3 Constructor & Destructor Documentation . . . 969
8.277.3.1 p_run . . . 969
8.277.3.2 p_run . . . 969
8.277.3.3 p_run . . . 969
8.277.4 Member Function Documentation . . . 969
8.277.4.1 bbox . . . 969
8.277.4.2 end . . . 969
8.277.4.3 has . . . 969
8.277.4.4 has . . . 970
8.277.4.5 has_index . . . 970
8.277.4.6 init . . . 970
8.277.4.7 is_valid . . . 970
8.277.4.8 length . . . 970
8.277.4.9 memory_size . . . 970
8.277.4.10nsites . . . 970
8.277.4.11operator[] . . . 970
8.277.4.12start . . . 971
8.278mln::p_set< P > Class Template Reference . . . 972
8.278.1 Detailed Description . . . 973
8.278.2 Member Typedef Documentation . . . 973
8.278.2.1 bkd_piter . . . 973
8.278.2.2 element . . . 973
8.278.2.3 fwd_piter . . . 973
8.278.2.4 i_element . . . 974
8.278.2.5 piter . . . 974
8.278.2.6 psite . . . 974
8.278.2.7 r_element . . . 974
8.278.3 Constructor & Destructor Documentation . . . 974
8.278.3.1 p_set . . . 974
8.278.4 Member Function Documentation . . . 974

8.278.4.1 clear . . . 974
8.278.4.2 has . . . 974
8.278.4.3 has . . . 974
8.278.4.4 has . . . 974
8.278.4.5 insert . . . 974
8.278.4.6 is_valid . . . 975
8.278.4.7 memory_size . . . 975
8.278.4.8 nsites . . . 975
8.278.4.9 operator[] . . . 975
8.278.4.10remove . . . 975
8.278.4.11std_vector . . . 975
8.278.4.12util_set . . . 975
8.279mln::p_transformed< S, F > Class Template Reference . . . 976
8.279.1 Detailed Description . . . 977
8.279.2 Member Typedef Documentation . . . 977
8.279.2.1 bkd_piter . . . 977
8.279.2.2 element . . . 977
8.279.2.3 fwd_piter . . . 977
8.279.2.4 piter . . . 977
8.279.2.5 psite . . . 977
8.279.3 Constructor & Destructor Documentation . . . 977
8.279.3.1 p_transformed . . . 977
8.279.3.2 p_transformed . . . 977
8.279.4 Member Function Documentation . . . 978
8.279.4.1 function . . . 978
8.279.4.2 has . . . 978
8.279.4.3 is_valid . . . 978
8.279.4.4 memory_size . . . 978
8.279.4.5 primary_set . . . 978
8.280mln::p_transformed_piter< Pi, S, F > Struct Template Reference . . . 979
8.280.1 Detailed Description . . . 979
8.280.2 Constructor & Destructor Documentation . . . 979
8.280.2.1 p_transformed_piter . . . 979
8.280.2.2 p_transformed_piter . . . 979
8.280.3 Member Function Documentation . . . 980
8.280.3.1 change_target . . . 980

8.281 mln::p_vaccess< V, S > Class Template Reference . . . 981
8.281.1 Detailed Description . . . 982
8.281.2 Member Typedef Documentation . . . 982
8.281.2.1 bkd_piter . . . 982
8.281.2.2 element . . . 982
8.281.2.3 fwd_piter . . . 982
8.281.2.4 i_element . . . 982
8.281.2.5 piter . . . 982
8.281.2.6 pset . . . 983
8.281.2.7 psite . . . 983
8.281.2.8 value . . . 983
8.281.2.9 vset . . . 983
8.281.3 Constructor & Destructor Documentation . . . 983
8.281.3.1 p_vaccess . . . 983
8.281.4 Member Function Documentation . . . 983
8.281.4.1 has . . . 983
8.281.4.2 has . . . 983
8.281.4.3 insert . . . 983
8.281.4.4 insert . . . 983
8.281.4.5 is_valid . . . 984
8.281.4.6 memory_size . . . 984
8.281.4.7 operator() . . . 984
8.281.4.8 values . . . 984
8.282 mln::p_vertices< G, F > Class Template Reference . . . 985
8.282.1 Detailed Description . . . 986
8.282.2 Member Typedef Documentation . . . 987
8.282.2.1 bkd_piter . . . 987
8.282.2.2 element . . . 987
8.282.2.3 fun_t . . . 987
8.282.2.4 fwd_piter . . . 987
8.282.2.5 graph_element . . . 987
8.282.2.6 graph_t . . . 987
8.282.2.7 piter . . . 987
8.282.2.8 psite . . . 987
8.282.2.9 vertex . . . 988
8.282.3 Constructor & Destructor Documentation . . . 988

8.282.3.1 p_vertices . . . 988
8.282.3.2 p_vertices . . . 988
8.282.3.3 p_vertices . . . 988
8.282.3.4 p_vertices . . . 988
8.282.3.5 p_vertices . . . 989
8.282.4 Member Function Documentation . . . 989
8.282.4.1 function . . . 989
8.282.4.2 graph . . . 989
8.282.4.3 has . . . 989
8.282.4.4 has . . . 989
8.282.4.5 invalidate . . . 989
8.282.4.6 is_valid . . . 989
8.282.4.7 memory_size . . . 990
8.282.4.8 nsites . . . 990
8.282.4.9 nvertices . . . 990
8.282.4.10operator() . . . 990
8.283mln::pixel< I > Struct Template Reference . . . 991
8.283.1 Detailed Description . . . 991
8.283.2 Constructor & Destructor Documentation . . . 991
8.283.2.1 pixel . . . 991
8.283.2.2 pixel . . . 992
8.283.3 Member Function Documentation . . . 992
8.283.3.1 change_to . . . 992
8.283.3.2 is_valid . . . 992
8.284mln::plain< I > Class Template Reference . . . 993
8.284.1 Detailed Description . . . 993
8.284.2 Member Typedef Documentation . . . 993
8.284.2.1 skeleton . . . 993
8.284.3 Constructor & Destructor Documentation . . . 994
8.284.3.1 plain . . . 994
8.284.3.2 plain . . . 994
8.284.3.3 plain . . . 994
8.284.4 Member Function Documentation . . . 994
8.284.4.1 operator I . . . 994
8.284.4.2 operator= . . . 994
8.284.4.3 operator= . . . 994

8.285 mln::Point< P > Struct Template Reference . . . 995
8.285.1 Detailed Description . . . 995
8.285.2 Member Typedef Documentation . . . 996
8.285.2.1 point . . . 996
8.285.3 Member Function Documentation . . . 996
8.285.3.1 to_point . . . 996
8.285.4 Friends And Related Function Documentation . . . 996
8.285.4.1 operator+= . . . 996
8.285.4.2 operator-= . . . 996
8.285.4.3 operator/ . . . 997
8.286 mln::point< G, C > Struct Template Reference . . . 998
8.286.1 Detailed Description . . . 1000
8.286.2 Member Typedef Documentation . . . 1000
8.286.2.1 coord . . . 1000
8.286.2.2 delta . . . 1000
8.286.2.3 dpsite . . . 1000
8.286.2.4 grid . . . 1000
8.286.2.5 h_vec . . . 1001
8.286.2.6 vec . . . 1001
8.286.3 Member Enumeration Documentation . . . 1001
8.286.3.1 "@29 . . . 1001
8.286.4 Constructor & Destructor Documentation . . . 1001
8.286.4.1 point . . . 1001
8.286.4.2 point . . . 1001
8.286.4.3 point . . . 1001
8.286.4.4 point . . . 1001
8.286.4.5 point . . . 1001
8.286.5 Member Function Documentation . . . 1002
8.286.5.1 last_coord . . . 1002
8.286.5.2 last_coord . . . 1002
8.286.5.3 minus_infty . . . 1002
8.286.5.4 operator+= . . . 1002
8.286.5.5 operator-= . . . 1002
8.286.5.6 operator[] . . . 1002
8.286.5.7 operator[] . . . 1003
8.286.5.8 plus_infty . . . 1003

8.286.5.9 set_all . . . 1003
8.286.5.10to_h_vec . . . 1003
8.286.5.11to_vec . . . 1003
8.286.6 Member Data Documentation . . . 1003
8.286.6.1 origin . . . 1003
8.287mln::Proxy< E > Struct Template Reference . . . 1004
8.287.1 Detailed Description . . . 1004
8.288mln::Proxy< void > Struct Template Reference . . . 1005
8.288.1 Detailed Description . . . 1005
8.289mln::Pseudo_Site< E > Struct Template Reference . . . 1006
8.289.1 Detailed Description . . . 1006
8.290mln::Pseudo_Site< void > Struct Template Reference . . . 1007
8.290.1 Detailed Description . . . 1007
8.291mln::pw::image< F, S > Class Template Reference . . . 1008
8.291.1 Detailed Description . . . 1008
8.291.2 Member Typedef Documentation . . . 1008
8.291.2.1 skeleton . . . 1008
8.291.3 Constructor & Destructor Documentation . . . 1008
8.291.3.1 image . . . 1008
8.291.3.2 image . . . 1008
8.292mln::registration::closest_point_basic< P > Class Template Reference . . . 1009
8.292.1 Detailed Description . . . 1009
8.293mln::registration::closest_point_with_map< P > Class Template Reference . . . 1010
8.293.1 Detailed Description . . . 1010
8.294mln::Regular_Grid< E > Struct Template Reference . . . 1011
8.294.1 Detailed Description . . . 1011
8.295mln::safe_image< I > Class Template Reference . . . 1012
8.295.1 Detailed Description . . . 1012
8.295.2 Member Typedef Documentation . . . 1012
8.295.2.1 skeleton . . . 1012
8.295.3 Member Function Documentation . . . 1012
8.295.3.1 operator safe_image< const I > . . . 1012
8.296mln::select::p_of< P > Struct Template Reference . . . 1013
8.296.1 Detailed Description . . . 1013
8.297mln::Site< E > Struct Template Reference . . . 1014
8.297.1 Detailed Description . . . 1014

8.298 mln::Site< void > Struct Template Reference . . . 1015
8.298.1 Detailed Description . . . 1015
8.299 mln::Site_Iterator< E > Struct Template Reference . . . 1016
8.299.1 Detailed Description . . . 1017
8.299.2 Member Function Documentation . . . 1017
8.299.2.1 next . . . 1017
8.300 mln::Site_Proxy< E > Struct Template Reference . . . 1018
8.300.1 Detailed Description . . . 1018
8.301 mln::Site_Proxy< void > Struct Template Reference . . . 1019
8.301.1 Detailed Description . . . 1019
8.302 mln::Site_Set< E > Struct Template Reference . . . 1020
8.302.1 Detailed Description . . . 1021
8.302.2 Friends And Related Function Documentation . . . 1021
8.302.2.1 diff . . . 1021
8.302.2.2 inter . . . 1021
8.302.2.3 operator< . . . 1022
8.302.2.4 operator<< . . . 1022
8.302.2.5 operator<= . . . 1022
8.302.2.6 operator== . . . 1022
8.302.2.7 sym_diff . . . 1022
8.302.2.8 uni . . . 1023
8.302.2.9 unique . . . 1023
8.303 mln::Site_Set< void > Struct Template Reference . . . 1024
8.303.1 Detailed Description . . . 1024
8.304 mln::sub_image< I, S > Struct Template Reference . . . 1025
8.304.1 Detailed Description . . . 1025
8.304.2 Member Typedef Documentation . . . 1025
8.304.2.1 skeleton . . . 1025
8.304.3 Constructor & Destructor Documentation . . . 1025
8.304.3.1 sub_image . . . 1025
8.304.3.2 sub_image . . . 1026
8.304.4 Member Function Documentation . . . 1026
8.304.4.1 domain . . . 1026
8.304.4.2 operator sub_image< const I, S > . . . 1026
8.305 mln::sub_image_if< I, S > Struct Template Reference . . . 1027
8.305.1 Detailed Description . . . 1027

8.305.2 Member Typedef Documentation . . . 1027
8.305.2.1 skeleton . . . 1027
8.305.3 Constructor & Destructor Documentation . . . 1027
8.305.3.1 sub_image_if . . . 1027
8.305.3.2 sub_image_if . . . 1028
8.305.4 Member Function Documentation . . . 1028
8.305.4.1 domain . . . 1028
8.306 mln::thru_image< I, F > Class Template Reference . . . 1029
8.306.1 Detailed Description . . . 1029
8.306.2 Member Function Documentation . . . 1029
8.306.2.1 operator thru_image< const I, F > . . . 1029
8.307 mln::thrubin_image< I1, I2, F > Class Template Reference . . . 1030
8.307.1 Detailed Description . . . 1030
8.307.2 Member Typedef Documentation . . . 1030
8.307.2.1 psite . . . 1030
8.307.2.2 rvalue . . . 1030
8.307.2.3 skeleton . . . 1031
8.307.2.4 value . . . 1031
8.307.3 Member Function Documentation . . . 1031
8.307.3.1 operator thrubin_image< const I1, const I2, F > . . . 1031
8.308 mln::topo::adj_higher_dim_connected_n_face_bkd_iter< D > Class Template Reference . . . 1032
8.308.1 Detailed Description . . . 1032
8.308.2 Constructor & Destructor Documentation . . . 1032
8.308.2.1 adj_higher_dim_connected_n_face_bkd_iter . . . 1032
8.308.3 Member Function Documentation . . . 1032
8.308.3.1 next . . . 1032
8.309 mln::topo::adj_higher_dim_connected_n_face_fwd_iter< D > Class Template Reference . . . 1034
8.309.1 Detailed Description . . . 1034
8.309.2 Constructor & Destructor Documentation . . . 1034
8.309.2.1 adj_higher_dim_connected_n_face_fwd_iter . . . 1034
8.309.3 Member Function Documentation . . . 1034
8.309.3.1 next . . . 1034
8.310 mln::topo::adj_higher_face_bkd_iter< D > Class Template Reference . . . 1036
8.310.1 Detailed Description . . . 1036
8.310.2 Constructor & Destructor Documentation . . . 1036
8.310.2.1 adj_higher_face_bkd_iter . . . 1036

8.310.3 Member Function Documentation . . . 1036
8.310.3.1 next . . . 1036
8.311 mln::topo::adj_higher_face_fwd_iter< D > Class Template Reference . . . 1037
8.311.1 Detailed Description . . . 1037
8.311.2 Constructor & Destructor Documentation . . . 1037
8.311.2.1 adj_higher_face_fwd_iter . . . 1037
8.311.3 Member Function Documentation . . . 1037
8.311.3.1 next . . . 1037
8.312 mln::topo::adj_lower_dim_connected_n_face_bkd_iter< D > Class Template Reference . . . 1038
8.312.1 Detailed Description . . . 1038
8.312.2 Constructor & Destructor Documentation . . . 1038
8.312.2.1 adj_lower_dim_connected_n_face_bkd_iter . . . 1038
8.312.3 Member Function Documentation . . . 1038
8.312.3.1 next . . . 1038
8.313 mln::topo::adj_lower_dim_connected_n_face_fwd_iter< D > Class Template Reference . . . 1040
8.313.1 Detailed Description . . . 1040
8.313.2 Constructor & Destructor Documentation . . . 1040
8.313.2.1 adj_lower_dim_connected_n_face_fwd_iter . . . 1040
8.313.3 Member Function Documentation . . . 1040
8.313.3.1 next . . . 1040
8.314 mln::topo::adj_lower_face_bkd_iter< D > Class Template Reference . . . 1042
8.314.1 Detailed Description . . . 1042
8.314.2 Constructor & Destructor Documentation . . . 1042
8.314.2.1 adj_lower_face_bkd_iter . . . 1042
8.315 mln::topo::adj_lower_face_fwd_iter< D > Class Template Reference . . . 1043
8.315.1 Detailed Description . . . 1043
8.315.2 Constructor & Destructor Documentation . . . 1043
8.315.2.1 adj_lower_face_fwd_iter . . . 1043
8.316 mln::topo::adj_lower_higher_face_bkd_iter< D > Class Template Reference . . . 1044
8.316.1 Detailed Description . . . 1044
8.316.2 Constructor & Destructor Documentation . . . 1044
8.316.2.1 adj_lower_higher_face_bkd_iter . . . 1044
8.316.3 Member Function Documentation . . . 1044
8.316.3.1 next . . . 1044
8.317 mln::topo::adj_lower_higher_face_fwd_iter< D > Class Template Reference . . . 1045
8.317.1 Detailed Description . . . 1045

8.317.2 Constructor & Destructor Documentation . . . 1045
8.317.2.1 adj_lower_higher_face_fwd_iter . . . 1045
8.317.3 Member Function Documentation . . . 1045
8.317.3.1 next . . . 1045
8.318 mln::topo::adj_m_face_bkd_iter< D > Class Template Reference . . . 1046
8.318.1 Detailed Description . . . 1046
8.318.2 Constructor & Destructor Documentation . . . 1046
8.318.2.1 adj_m_face_bkd_iter . . . 1046
8.318.2.2 adj_m_face_bkd_iter . . . 1046
8.318.3 Member Function Documentation . . . 1047
8.318.3.1 next . . . 1047
8.319 mln::topo::adj_m_face_fwd_iter< D > Class Template Reference . . . 1048
8.319.1 Detailed Description . . . 1048
8.319.2 Constructor & Destructor Documentation . . . 1048
8.319.2.1 adj_m_face_fwd_iter . . . 1048
8.319.2.2 adj_m_face_fwd_iter . . . 1048
8.319.3 Member Function Documentation . . . 1049
8.319.3.1 next . . . 1049
8.320 mln::topo::algebraic_face< D > Struct Template Reference . . . 1050
8.320.1 Detailed Description . . . 1051
8.320.2 Constructor & Destructor Documentation . . . 1051
8.320.2.1 algebraic_face . . . 1051
8.320.2.2 algebraic_face . . . 1051
8.320.2.3 algebraic_face . . . 1052
8.320.2.4 algebraic_face . . . 1052
8.320.3 Member Function Documentation . . . 1052
8.320.3.1 cplx . . . 1052
8.320.3.2 data . . . 1052
8.320.3.3 dec_face_id . . . 1052
8.320.3.4 dec_n . . . 1052
8.320.3.5 face_id . . . 1052
8.320.3.6 higher_dim_adj_faces . . . 1052
8.320.3.7 inc_face_id . . . 1052
8.320.3.8 inc_n . . . 1053
8.320.3.9 invalidate . . . 1053
8.320.3.10 is_valid . . . 1053

8.320.3.11 lower_dim_adj_faces . . . 1053
8.320.3.12 n . . . 1053
8.320.3.13 set_cplx . . . 1053
8.320.3.14 set_face_id . . . 1053
8.320.3.15 set_n . . . 1053
8.320.3.16 set_sign . . . 1053
8.320.3.17 sign . . . 1053
8.321 mln::topo::algebraic_n_face< N, D > Class Template Reference . . . 1054
8.321.1 Detailed Description . . . 1055
8.321.2 Constructor & Destructor Documentation . . . 1055
8.321.2.1 algebraic_n_face . . . 1055
8.321.2.2 algebraic_n_face . . . 1055
8.321.2.3 algebraic_n_face . . . 1055
8.321.3 Member Function Documentation . . . 1056
8.321.3.1 cplx . . . 1056
8.321.3.2 data . . . 1056
8.321.3.3 dec_face_id . . . 1056
8.321.3.4 face_id . . . 1056
8.321.3.5 higher_dim_adj_faces . . . 1056
8.321.3.6 inc_face_id . . . 1056
8.321.3.7 invalidate . . . 1056
8.321.3.8 is_valid . . . 1056
8.321.3.9 lower_dim_adj_faces . . . 1056
8.321.3.10 n . . . 1057
8.321.3.11 set_cplx . . . 1057
8.321.3.12 set_face_id . . . 1057
8.321.3.13 set_sign . . . 1057
8.321.3.14 sign . . . 1057
8.322 mln::topo::center_only_iter< D > Class Template Reference . . . 1058
8.322.1 Detailed Description . . . 1058
8.322.2 Constructor & Destructor Documentation . . . 1058
8.322.2.1 center_only_iter . . . 1058
8.322.3 Member Function Documentation . . . 1059
8.322.3.1 next . . . 1059
8.323 mln::topo::centered_bkd_iter_adapter< D, I > Class Template Reference . . . 1060
8.323.1 Detailed Description . . . 1060

8.323.2 Constructor & Destructor Documentation . . . 1060
8.323.2.1 centered_bkd_iter_adapter . . . 1060
8.323.3 Member Function Documentation . . . 1060
8.323.3.1 next . . . 1060
8.324 mln::topo::centered_fwd_iter_adapter< D, I > Class Template Reference . . . 1061
8.324.1 Detailed Description . . . 1061
8.324.2 Constructor & Destructor Documentation . . . 1061
8.324.2.1 centered_fwd_iter_adapter . . . 1061
8.324.3 Member Function Documentation . . . 1061
8.324.3.1 next . . . 1061
8.325 mln::topo::complex< D > Class Template Reference . . . 1062
8.325.1 Detailed Description . . . 1063
8.325.2 Member Typedef Documentation . . . 1063
8.325.2.1 bkd_citer . . . 1063
8.325.2.2 fwd_citer . . . 1063
8.325.3 Constructor & Destructor Documentation . . . 1063
8.325.3.1 complex . . . 1063
8.325.4 Member Function Documentation . . . 1063
8.325.4.1 add_face . . . 1063
8.325.4.2 add_face . . . 1063
8.325.4.3 addr . . . 1063
8.325.4.4 nfaces . . . 1064
8.325.4.5 nfaces_of_dim . . . 1064
8.325.4.6 nfaces_of_static_dim . . . 1064
8.325.4.7 print . . . 1064
8.325.4.8 print_faces . . . 1064
8.326 mln::topo::face< D > Struct Template Reference . . . 1065
8.326.1 Detailed Description . . . 1066
8.326.2 Constructor & Destructor Documentation . . . 1066
8.326.2.1 face . . . 1066
8.326.2.2 face . . . 1066
8.326.2.3 face . . . 1066
8.326.3 Member Function Documentation . . . 1066
8.326.3.1 cplx . . . 1066
8.326.3.2 data . . . 1067
8.326.3.3 dec_face_id . . . 1067

8.326.3.4 dec_n . . . 1067
8.326.3.5 face_id . . . 1067
8.326.3.6 higher_dim_adj_faces . . . 1067
8.326.3.7 inc_face_id . . . 1067
8.326.3.8 inc_n . . . 1067
8.326.3.9 invalidate . . . 1067
8.326.3.10 is_valid . . . 1067
8.326.3.11 lower_dim_adj_faces . . . 1068
8.326.3.12 n . . . 1068
8.326.3.13 set_cplx . . . 1068
8.326.3.14 set_face_id . . . 1068
8.326.3.15 set_n . . . 1068
8.327 mln::topo::face_bkd_iter< D > Class Template Reference . . . 1069
8.327.1 Detailed Description . . . 1069
8.327.2 Constructor & Destructor Documentation . . . 1069
8.327.2.1 face_bkd_iter . . . 1069
8.327.3 Member Function Documentation . . . 1069
8.327.3.1 next . . . 1069
8.327.3.2 start . . . 1070
8.328 mln::topo::face_fwd_iter< D > Class Template Reference . . . 1071
8.328.1 Detailed Description . . . 1071
8.328.2 Constructor & Destructor Documentation . . . 1071
8.328.2.1 face_fwd_iter . . . 1071
8.328.3 Member Function Documentation . . . 1071
8.328.3.1 next . . . 1071
8.328.3.2 start . . . 1072
8.329 mln::topo::is_n_face< N > Struct Template Reference . . . 1073
8.329.1 Detailed Description . . . 1073
8.330 mln::topo::is_simple_cell< I > Class Template Reference . . . 1074
8.330.1 Detailed Description . . . 1075
8.330.2 Member Typedef Documentation . . . 1075
8.330.2.1 psite . . . 1075
8.330.2.2 result . . . 1075
8.330.3 Member Function Documentation . . . 1075
8.330.3.1 mln_geom . . . 1075
8.330.3.2 operator() . . . 1075

8.330.3.3 set_image . . . 1076
8.330.4 Member Data Documentation . . . 1076
8.330.4.1 D . . . 1076
8.331 mln::topo::n_face< N, D > Class Template Reference . . . 1077
8.331.1 Detailed Description . . . 1078
8.331.2 Constructor & Destructor Documentation . . . 1078
8.331.2.1 n_face . . . 1078
8.331.2.2 n_face . . . 1078
8.331.3 Member Function Documentation . . . 1078
8.331.3.1 cplx . . . 1078
8.331.3.2 data . . . 1078
8.331.3.3 dec_face_id . . . 1078
8.331.3.4 face_id . . . 1079
8.331.3.5 higher_dim_adj_faces . . . 1079
8.331.3.6 inc_face_id . . . 1079
8.331.3.7 invalidate . . . 1079
8.331.3.8 is_valid . . . 1079
8.331.3.9 lower_dim_adj_faces . . . 1079
8.331.3.10 n . . . 1079
8.331.3.11 set_cplx . . . 1079
8.331.3.12 set_face_id . . . 1080
8.332 mln::topo::n_face_bkd_iter< D > Class Template Reference . . . 1081
8.332.1 Detailed Description . . . 1081
8.332.2 Member Function Documentation . . . 1081
8.332.2.1 n . . . 1081
8.332.2.2 next . . . 1081
8.332.2.3 start . . . 1082
8.333 mln::topo::n_face_fwd_iter< D > Class Template Reference . . . 1083
8.333.1 Detailed Description . . . 1083
8.333.2 Member Function Documentation . . . 1083
8.333.2.1 n . . . 1083
8.333.2.2 next . . . 1083
8.333.2.3 start . . . 1084
8.334 mln::topo::n_faces_set< N, D > Class Template Reference . . . 1085
8.334.1 Detailed Description . . . 1085
8.334.2 Member Typedef Documentation . . . 1085

8.334.2.1 faces_type . . . 1085
8.334.3 Member Function Documentation . . . 1085
8.334.3.1 add . . . 1085
8.334.3.2 faces . . . 1086
8.334.3.3 reserve . . . 1086
8.335 mln::topo::static_n_face_bkd_iter< N, D > Class Template Reference . . . 1087
8.335.1 Detailed Description . . . 1087
8.335.2 Constructor & Destructor Documentation . . . 1087
8.335.2.1 static_n_face_bkd_iter . . . 1087
8.335.3 Member Function Documentation . . . 1087
8.335.3.1 next . . . 1087
8.335.3.2 start . . . 1088
8.336 mln::topo::static_n_face_fwd_iter< N, D > Class Template Reference . . . 1089
8.336.1 Detailed Description . . . 1089
8.336.2 Constructor & Destructor Documentation . . . 1089
8.336.2.1 static_n_face_fwd_iter . . . 1089
8.336.3 Member Function Documentation . . . 1089
8.336.3.1 next . . . 1089
8.336.3.2 start . . . 1090
8.337 mln::tr_image< S, I, T > Struct Template Reference . . . 1091
8.337.1 Detailed Description . . . 1092
8.337.2 Member Typedef Documentation . . . 1092
8.337.2.1 lvalue . . . 1092
8.337.2.2 psite . . . 1092
8.337.2.3 rvalue . . . 1092
8.337.2.4 site . . . 1092
8.337.2.5 skeleton . . . 1092
8.337.2.6 value . . . 1092
8.337.3 Constructor & Destructor Documentation . . . 1092
8.337.3.1 tr_image . . . 1092
8.337.4 Member Function Documentation . . . 1093
8.337.4.1 domain . . . 1093
8.337.4.2 has . . . 1093
8.337.4.3 is_valid . . . 1093
8.337.4.4 operator() . . . 1093
8.337.4.5 set_tr . . . 1093

8.337.4.6 tr . . . 1093
8.338 mln::transformed_image< I, F > Struct Template Reference . . . 1094
8.338.1 Detailed Description . . . 1094
8.338.2 Member Typedef Documentation . . . 1094
8.338.2.1 skeleton . . . 1094
8.338.3 Constructor & Destructor Documentation . . . 1095
8.338.3.1 transformed_image . . . 1095
8.338.3.2 transformed_image . . . 1095
8.338.4 Member Function Documentation . . . 1095
8.338.4.1 domain . . . 1095
8.338.4.2 operator transformed_image< const I, F > . . . 1095
8.338.4.3 operator() . . . 1095
8.338.4.4 operator() . . . 1095
8.339 mln::unproject_image< I, D, F > Struct Template Reference . . . 1096
8.339.1 Detailed Description . . . 1096
8.339.2 Constructor & Destructor Documentation . . . 1096
8.339.2.1 unproject_image . . . 1096
8.339.2.2 unproject_image . . . 1096
8.339.3 Member Function Documentation . . . 1096
8.339.3.1 domain . . . 1096
8.339.3.2 operator() . . . 1097
8.339.3.3 operator() . . . 1097
8.340 mln::util::adjacency_matrix< V > Class Template Reference . . . 1098
8.340.1 Detailed Description . . . 1098
8.340.2 Constructor & Destructor Documentation . . . 1098
8.340.2.1 adjacency_matrix . . . 1098
8.340.2.2 adjacency_matrix . . . 1098
8.341 mln::util::array< T > Class Template Reference . . . 1099
8.341.1 Detailed Description . . . 1101
8.341.2 Member Typedef Documentation . . . 1101
8.341.2.1 bkd_eiter . . . 1101
8.341.2.2 eiter . . . 1101
8.341.2.3 element . . . 1101
8.341.2.4 fwd_eiter . . . 1102
8.341.2.5 result . . . 1102
8.341.3 Constructor & Destructor Documentation . . . 1102

8.341.3.1 array . . . 1102
8.341.3.2 array . . . 1102
8.341.3.3 array . . . 1102
8.341.4 Member Function Documentation . . . 1102
8.341.4.1 append . . . 1102
8.341.4.2 append . . . 1102
8.341.4.3 clear . . . 1102
8.341.4.4 fill . . . 1103
8.341.4.5 is_empty . . . 1103
8.341.4.6 memory_size . . . 1103
8.341.4.7 nelements . . . 1103
8.341.4.8 operator() . . . 1103
8.341.4.9 operator() . . . 1103
8.341.4.10 operator[] . . . 1104
8.341.4.11 operator[] . . . 1104
8.341.4.12 reserve . . . 1104
8.341.4.13 resize . . . 1104
8.341.4.14 resize . . . 1104
8.341.4.15 size . . . 1104
8.341.4.16 std_vector . . . 1105
8.342 mln::util::branch< T > Class Template Reference . . . 1106
8.342.1 Detailed Description . . . 1106
8.342.2 Constructor & Destructor Documentation . . . 1106
8.342.2.1 branch . . . 1106
8.342.3 Member Function Documentation . . . 1106
8.342.3.1 apex . . . 1106
8.342.3.2 util_tree . . . 1107
8.343 mln::util::branch_iter< T > Class Template Reference . . . 1108
8.343.1 Detailed Description . . . 1108
8.343.2 Member Function Documentation . . . 1108
8.343.2.1 deepness . . . 1108
8.343.2.2 invalidate . . . 1108
8.343.2.3 is_valid . . . 1109
8.343.2.4 next . . . 1109
8.343.2.5 operator util::tree_node< T > & . . . 1109
8.343.2.6 start . . . 1109

8.344 mln::util::branch_iter_ind< T > Class Template Reference . . . 1110
8.344.1 Detailed Description . . . 1110
8.344.2 Member Function Documentation . . . 1110
8.344.2.1 deepness . . . 1110
8.344.2.2 invalidate . . . 1110
8.344.2.3 is_valid . . . 1111
8.344.2.4 next . . . 1111
8.344.2.5 operator util::tree_node< T > & . . . 1111
8.344.2.6 start . . . 1111
8.345 mln::util::couple< T, U > Class Template Reference . . . 1112
8.345.1 Detailed Description . . . 1112
8.345.2 Member Function Documentation . . . 1113
8.345.2.1 change_both . . . 1113
8.345.2.2 change_first . . . 1113
8.345.2.3 change_second . . . 1113
8.345.2.4 first . . . 1113
8.345.2.5 second . . . 1113
8.346 mln::util::eat Struct Reference . . . 1114
8.346.1 Detailed Description . . . 1114
8.347 mln::util::edge< G > Class Template Reference . . . 1115
8.347.1 Detailed Description . . . 1116
8.347.2 Member Typedef Documentation . . . 1116
8.347.2.1 category . . . 1116
8.347.2.2 graph_t . . . 1116
8.347.2.3 id_t . . . 1116
8.347.2.4 id_value_t . . . 1116
8.347.3 Constructor & Destructor Documentation . . . 1116
8.347.3.1 edge . . . 1116
8.347.4 Member Function Documentation . . . 1117
8.347.4.1 change_graph . . . 1117
8.347.4.2 graph . . . 1117
8.347.4.3 id . . . 1117
8.347.4.4 invalidate . . . 1117
8.347.4.5 is_valid . . . 1117
8.347.4.6 ith_nbh_edge . . . 1117
8.347.4.7 nmax_nbh_edges . . . 1117

8.347.4.8 operator edge_id_t . . . 1117
8.347.4.9 update_id . . . 1118
8.347.4.10 v1 . . . 1118
8.347.4.11 v2 . . . 1118
8.347.4.12 v_other . . . 1118
8.348 mln::util::fibonacci_heap< P, T > Class Template Reference . . . 1119
8.348.1 Detailed Description . . . 1120
8.348.2 Constructor & Destructor Documentation . . . 1120
8.348.2.1 fibonacci_heap . . . 1120
8.348.2.2 fibonacci_heap . . . 1120
8.348.3 Member Function Documentation . . . 1120
8.348.3.1 clear . . . 1120
8.348.3.2 front . . . 1120
8.348.3.3 is_empty . . . 1120
8.348.3.4 is_valid . . . 1121
8.348.3.5 nelements . . . 1121
8.348.3.6 operator= . . . 1121
8.348.3.7 pop_front . . . 1121
8.348.3.8 push . . . 1121
8.348.3.9 push . . . 1121
8.349 mln::util::graph Class Reference . . . 1122
8.349.1 Detailed Description . . . 1124
8.349.2 Member Typedef Documentation . . . 1124
8.349.2.1 edge_fwd_iter . . . 1124
8.349.2.2 edge_nbh_edge_fwd_iter . . . 1124
8.349.2.3 edges_set_t . . . 1124
8.349.2.4 edges_t . . . 1124
8.349.2.5 vertex_nbh_edge_fwd_iter . . . 1124
8.349.2.6 vertex_nbh_vertex_fwd_iter . . . 1124
8.349.2.7 vertices_t . . . 1124
8.349.3 Constructor & Destructor Documentation . . . 1124
8.349.3.1 graph . . . 1124
8.349.3.2 graph . . . 1124
8.349.4 Member Function Documentation . . . 1125
8.349.4.1 add_edge . . . 1125
8.349.4.2 add_vertex . . . 1125

8.349.4.3 add_vertices . . . 1125
8.349.4.4 e_ith_nbh_edge . . . 1125
8.349.4.5 e_nmax . . . 1125
8.349.4.6 edge . . . 1126
8.349.4.7 edge . . . 1126
8.349.4.8 edges . . . 1126
8.349.4.9 has_e . . . 1126
8.349.4.10 has_v . . . 1126
8.349.4.11 invalidate . . . 1126
8.349.4.12 is_subgraph_of . . . 1126
8.349.4.13 is_valid . . . 1126
8.349.4.14 v1 . . . 1127
8.349.4.15 v2 . . . 1127
8.349.4.16 v_ith_nbh_edge . . . 1127
8.349.4.17 v_ith_nbh_vertex . . . 1127
8.349.4.18 v_nmax . . . 1127
8.349.4.19 vertex . . . 1127
8.350 mln::util::greater_point< I > Class Template Reference . . . 1128
8.350.1 Detailed Description . . . 1128
8.350.2 Member Function Documentation . . . 1128
8.350.2.1 operator() . . . 1128
8.351 mln::util::greater_psite< I > Class Template Reference . . . 1129
8.351.1 Detailed Description . . . 1129
8.351.2 Member Function Documentation . . . 1129
8.351.2.1 operator() . . . 1129
8.352 mln::util::head< T, R > Class Template Reference . . . 1130
8.352.1 Detailed Description . . . 1130
8.353 mln::util::ignore Struct Reference . . . 1131
8.353.1 Detailed Description . . . 1131
8.354 mln::util::ilcell< T > Struct Template Reference . . . 1132
8.354.1 Detailed Description . . . 1132
8.355 mln::util::line_graph< G > Class Template Reference . . . 1133
8.355.1 Detailed Description . . . 1134
8.355.2 Member Typedef Documentation . . . 1134
8.355.2.1 edge_fwd_iter . . . 1134
8.355.2.2 edges_t . . . 1135

8.355.2.3 vertex_nbh_edge_fwd_iter . . . 1135
8.355.2.4 vertex_nbh_vertex_fwd_iter . . . 1135
8.355.2.5 vertices_t . . . 1135
8.355.3 Member Function Documentation . . . 1135
8.355.3.1 e_ith_nbh_edge . . . 1135
8.355.3.2 e_nmax . . . 1135
8.355.3.3 edge . . . 1135
8.355.3.4 graph . . . 1135
8.355.3.5 has . . . 1136
8.355.3.6 has . . . 1136
8.355.3.7 has_e . . . 1136
8.355.3.8 has_v . . . 1136
8.355.3.9 invalidate . . . 1136
8.355.3.10 is_subgraph_of . . . 1136
8.355.3.11 is_valid . . . 1136
8.355.3.12 v1 . . . 1137
8.355.3.13 v2 . . . 1137
8.355.3.14 v_ith_nbh_edge . . . 1137
8.355.3.15 v_ith_nbh_vertex . . . 1137
8.355.3.16 v_nmax . . . 1137
8.355.3.17 vertex . . . 1137
8.356 mln::util::nil Struct Reference . . . 1138
8.356.1 Detailed Description . . . 1138
8.357 mln::util::node< T, R > Class Template Reference . . . 1139
8.357.1 Detailed Description . . . 1139
8.358 mln::util::object_id< Tag, V > Class Template Reference . . . 1140
8.358.1 Detailed Description . . . 1140
8.358.2 Member Typedef Documentation . . . 1141
8.358.2.1 value_t . . . 1141
8.358.3 Constructor & Destructor Documentation . . . 1141
8.358.3.1 object_id . . . 1141
8.359 mln::util::ord< T > Struct Template Reference . . . 1142
8.359.1 Detailed Description . . . 1142
8.360 mln::util::ord_pair< T > Struct Template Reference . . . 1143
8.360.1 Detailed Description . . . 1143
8.360.2 Member Function Documentation . . . 1144

8.360.2.1 change_both . . . 1144
8.360.2.2 change_first . . . 1144
8.360.2.3 change_second . . . 1144
8.360.2.4 first . . . 1144
8.360.2.5 second . . . 1144
8.361 mln::util::pix< I > Struct Template Reference . . . 1145
8.361.1 Detailed Description . . . 1145
8.361.2 Member Typedef Documentation . . . 1145
8.361.2.1 psite . . . 1145
8.361.2.2 value . . . 1145
8.361.3 Constructor & Destructor Documentation . . . 1146
8.361.3.1 pix . . . 1146
8.361.4 Member Function Documentation . . . 1146
8.361.4.1 ima . . . 1146
8.361.4.2 p . . . 1146
8.361.4.3 v . . . 1146
8.362 mln::util::set< T > Class Template Reference . . . 1147
8.362.1 Detailed Description . . . 1148
8.362.2 Member Typedef Documentation . . . 1149
8.362.2.1 bkd_eiter . . . 1149
8.362.2.2 eiter . . . 1149
8.362.2.3 element . . . 1149
8.362.2.4 fwd_eiter . . . 1149
8.362.3 Constructor & Destructor Documentation . . . 1149
8.362.3.1 set . . . 1149
8.362.4 Member Function Documentation . . . 1149
8.362.4.1 clear . . . 1149
8.362.4.2 first_element . . . 1149
8.362.4.3 has . . . 1150
8.362.4.4 insert . . . 1150
8.362.4.5 insert . . . 1150
8.362.4.6 is_empty . . . 1150
8.362.4.7 last_element . . . 1151
8.362.4.8 memory_size . . . 1151
8.362.4.9 nelements . . . 1151
8.362.4.10 operator[] . . . 1151

8.362.4.1 lremove . . . 1151
8.362.4.2 std_vector . . . 1152
8.363 mln::util::site_pair< P > Class Template Reference . . . 1153
8.363.1 Detailed Description . . . 1153
8.363.2 Member Function Documentation . . . 1153
8.363.2.1 first . . . 1153
8.363.2.2 pair . . . 1154
8.363.2.3 second . . . 1154
8.364 mln::util::soft_heap< T, R > Class Template Reference . . . 1155
8.364.1 Detailed Description . . . 1156
8.364.2 Member Typedef Documentation . . . 1156
8.364.2.1 element . . . 1156
8.364.3 Constructor & Destructor Documentation . . . 1156
8.364.3.1 soft_heap . . . 1156
8.364.3.2 ~soft_heap . . . 1156
8.364.4 Member Function Documentation . . . 1156
8.364.4.1 clear . . . 1156
8.364.4.2 is_empty . . . 1156
8.364.4.3 is_valid . . . 1157
8.364.4.4 nelements . . . 1157
8.364.4.5 pop_front . . . 1157
8.364.4.6 push . . . 1157
8.364.4.7 push . . . 1157
8.365 mln::util::timer Class Reference . . . 1158
8.365.1 Detailed Description . . . 1158
8.366 mln::util::tracked_ptr< T > Struct Template Reference . . . 1159
8.366.1 Detailed Description . . . 1159
8.366.2 Constructor & Destructor Documentation . . . 1159
8.366.2.1 tracked_ptr . . . 1159
8.366.2.2 tracked_ptr . . . 1160
8.366.2.3 ~tracked_ptr . . . 1160
8.366.3 Member Function Documentation . . . 1160
8.366.3.1 operator bool . . . 1160
8.366.3.2 operator! . . . 1160
8.366.3.3 operator-> . . . 1160
8.366.3.4 operator-> . . . 1160

8.366.3.5 operator= . . . 1160
8.366.3.6 operator= . . . 1160
8.367 mln::util::tree< T > Class Template Reference . . . 1161
8.367.1 Detailed Description . . . 1161
8.367.2 Constructor & Destructor Documentation . . . 1161
8.367.2.1 tree . . . 1161
8.367.2.2 tree . . . 1161
8.367.3 Member Function Documentation . . . 1162
8.367.3.1 add_tree_down . . . 1162
8.367.3.2 add_tree_up . . . 1162
8.367.3.3 check_consistency . . . 1162
8.367.3.4 main_branch . . . 1162
8.367.3.5 root . . . 1162
8.368 mln::util::tree_node< T > Class Template Reference . . . 1163
8.368.1 Detailed Description . . . 1164
8.368.2 Constructor & Destructor Documentation . . . 1164
8.368.2.1 tree_node . . . 1164
8.368.2.2 tree_node . . . 1164
8.368.3 Member Function Documentation . . . 1164
8.368.3.1 add_child . . . 1164
8.368.3.2 add_child . . . 1164
8.368.3.3 check_consistency . . . 1165
8.368.3.4 children . . . 1165
8.368.3.5 children . . . 1165
8.368.3.6 delete_tree_node . . . 1165
8.368.3.7 elt . . . 1165
8.368.3.8 elt . . . 1165
8.368.3.9 parent . . . 1166
8.368.3.10 print . . . 1166
8.368.3.11 search . . . 1166
8.368.3.12 search_rec . . . 1166
8.368.3.13 set_parent . . . 1166
8.369 mln::util::vertex< G > Class Template Reference . . . 1167
8.369.1 Detailed Description . . . 1168
8.369.2 Member Typedef Documentation . . . 1169
8.369.2.1 Category . . . 1169

8.369.2.2 graph_t . . . 1169
8.369.2.3 id_t . . . 1169
8.369.2.4 id_value_t . . . 1169
8.369.3 Constructor & Destructor Documentation . . . 1169
8.369.3.1 vertex . . . 1169
8.369.4 Member Function Documentation . . . 1169
8.369.4.1 change_graph . . . 1169
8.369.4.2 edge_with . . . 1169
8.369.4.3 graph . . . 1169
8.369.4.4 id . . . 1170
8.369.4.5 invalidate . . . 1170
8.369.4.6 is_valid . . . 1170
8.369.4.7 ith_nbh_edge . . . 1170
8.369.4.8 ith_nbh_vertex . . . 1170
8.369.4.9 nmax_nbh_edges . . . 1170
8.369.4.10 nmax_nbh_vertices . . . 1170
8.369.4.11 operator vertex_id_t . . . 1170
8.369.4.12 other . . . 1171
8.369.4.13 update_id . . . 1171
8.370 mln::util::yes Struct Reference . . . 1172
8.370.1 Detailed Description . . . 1172
8.371 mln::Value< E > Struct Template Reference . . . 1173
8.371.1 Detailed Description . . . 1173
8.372 mln::value::float01 Class Reference . . . 1174
8.372.1 Detailed Description . . . 1175
8.372.2 Member Typedef Documentation . . . 1175
8.372.2.1 enc . . . 1175
8.372.2.2 equiv . . . 1175
8.372.3 Constructor & Destructor Documentation . . . 1175
8.372.3.1 float01 . . . 1175
8.372.3.2 float01 . . . 1175
8.372.3.3 float01 . . . 1175
8.372.4 Member Function Documentation . . . 1175
8.372.4.1 nbits . . . 1175
8.372.4.2 operator float . . . 1175
8.372.4.3 set_nbits . . . 1175

8.372.4.4 to_nbits . . . 1176
8.372.4.5 value . . . 1176
8.372.4.6 value_ind . . . 1176
8.373 mln::value::float01_f Struct Reference . . . 1177
8.373.1 Detailed Description . . . 1177
8.373.2 Constructor & Destructor Documentation . . . 1177
8.373.2.1 float01_f . . . 1177
8.373.2.2 float01_f . . . 1177
8.373.3 Member Function Documentation . . . 1177
8.373.3.1 operator float . . . 1177
8.373.3.2 operator= . . . 1177
8.373.3.3 value . . . 1178
8.374 mln::value::graylevel< n > Struct Template Reference . . . 1179
8.374.1 Detailed Description . . . 1180
8.374.2 Constructor & Destructor Documentation . . . 1180
8.374.2.1 graylevel . . . 1180
8.374.2.2 graylevel . . . 1180
8.374.2.3 graylevel . . . 1180
8.374.2.4 graylevel . . . 1180
8.374.2.5 graylevel . . . 1180
8.374.3 Member Function Documentation . . . 1181
8.374.3.1 operator= . . . 1181
8.374.3.2 operator= . . . 1181
8.374.3.3 operator= . . . 1181
8.374.3.4 operator= . . . 1181
8.374.3.5 to_float . . . 1181
8.374.3.6 value . . . 1181
8.375 mln::value::graylevel_f Struct Reference . . . 1182
8.375.1 Detailed Description . . . 1183
8.375.2 Constructor & Destructor Documentation . . . 1183
8.375.2.1 graylevel_f . . . 1183
8.375.2.2 graylevel_f . . . 1183
8.375.2.3 graylevel_f . . . 1183
8.375.2.4 graylevel_f . . . 1183
8.375.2.5 graylevel_f . . . 1183
8.375.3 Member Function Documentation . . . 1183

8.375.3.1 operator graylevel< n > . . . 1183
8.375.3.2 operator= . . . 1183
8.375.3.3 operator= . . . 1183
8.375.3.4 operator= . . . 1184
8.375.3.5 operator= . . . 1184
8.375.3.6 value . . . 1184
8.376 mln::value::int_s< n > Struct Template Reference . . . 1185
8.376.1 Detailed Description . . . 1186
8.376.2 Constructor & Destructor Documentation . . . 1186
8.376.2.1 int_s . . . 1186
8.376.2.2 int_s . . . 1186
8.376.2.3 int_s . . . 1186
8.376.3 Member Function Documentation . . . 1186
8.376.3.1 operator int . . . 1186
8.376.3.2 operator= . . . 1186
8.376.4 Member Data Documentation . . . 1186
8.376.4.1 one . . . 1186
8.376.4.2 zero . . . 1186
8.377 mln::value::int_u< n > Struct Template Reference . . . 1187
8.377.1 Detailed Description . . . 1188
8.377.2 Constructor & Destructor Documentation . . . 1188
8.377.2.1 int_u . . . 1188
8.377.2.2 int_u . . . 1188
8.377.2.3 int_u . . . 1188
8.377.3 Member Function Documentation . . . 1188
8.377.3.1 next . . . 1188
8.377.3.2 operator unsigned . . . 1188
8.377.3.3 operator- . . . 1188
8.377.3.4 operator= . . . 1188
8.378 mln::value::int_u_sat< n > Struct Template Reference . . . 1189
8.378.1 Detailed Description . . . 1190
8.378.2 Constructor & Destructor Documentation . . . 1190
8.378.2.1 int_u_sat . . . 1190
8.378.2.2 int_u_sat . . . 1190
8.378.3 Member Function Documentation . . . 1190
8.378.3.1 operator int . . . 1190

8.378.3.2 operator+= . . . 1190
8.378.3.3 operator-= . . . 1190
8.378.3.4 operator= . . . 1190
8.378.4 Member Data Documentation . . . 1191
8.378.4.1 one . . . 1191
8.378.4.2 zero . . . 1191
8.379 mln::value::Integer< E > Struct Template Reference . . . 1192
8.379.1 Detailed Description . . . 1192
8.380 mln::value::Integer< void > Struct Template Reference . . . 1193
8.380.1 Detailed Description . . . 1193
8.381 mln::value::label< n > Struct Template Reference . . . 1194
8.381.1 Detailed Description . . . 1195
8.381.2 Member Typedef Documentation . . . 1195
8.381.2.1 enc . . . 1195
8.381.3 Constructor & Destructor Documentation . . . 1195
8.381.3.1 label . . . 1195
8.381.3.2 label . . . 1195
8.381.3.3 label . . . 1195
8.381.4 Member Function Documentation . . . 1195
8.381.4.1 next . . . 1195
8.381.4.2 operator unsigned . . . 1195
8.381.4.3 operator++ . . . 1195
8.381.4.4 operator– . . . 1195
8.381.4.5 operator= . . . 1196
8.381.4.6 operator= . . . 1196
8.381.4.7 prev . . . 1196
8.382 mln::value::lut_vec< S, T > Struct Template Reference . . . 1197
8.382.1 Detailed Description . . . 1198
8.382.2 Member Typedef Documentation . . . 1198
8.382.2.1 bkd_viter . . . 1198
8.382.2.2 fwd_viter . . . 1198
8.382.2.3 value . . . 1198
8.382.3 Constructor & Destructor Documentation . . . 1198
8.382.3.1 lut_vec . . . 1198
8.382.3.2 lut_vec . . . 1198
8.382.3.3 lut_vec . . . 1198

8.382.4 Member Function Documentation . . . 1199
8.382.4.1 has . . . 1199
8.382.4.2 index_of . . . 1199
8.382.4.3 nvalues . . . 1199
8.382.4.4 operator[] . . . 1199
8.383 mln::value::proxy< I > Class Template Reference . . . 1200
8.383.1 Detailed Description . . . 1201
8.383.2 Member Typedef Documentation . . . 1201
8.383.2.1 enc . . . 1201
8.383.2.2 equiv . . . 1201
8.383.3 Constructor & Destructor Documentation . . . 1201
8.383.3.1 proxy . . . 1201
8.383.3.2 ~proxy . . . 1201
8.383.4 Member Function Documentation . . . 1202
8.383.4.1 operator typename I::value . . . 1202
8.383.4.2 operator V . . . 1202
8.383.4.3 operator= . . . 1202
8.383.4.4 operator= . . . 1202
8.383.4.5 operator= . . . 1202
8.383.4.6 to_value . . . 1202
8.384 mln::value::proxy< const I > Class Template Reference . . . 1203
8.384.1 Detailed Description . . . 1204
8.384.2 Member Typedef Documentation . . . 1204
8.384.2.1 enc . . . 1204
8.384.2.2 equiv . . . 1204
8.384.3 Constructor & Destructor Documentation . . . 1204
8.384.3.1 proxy . . . 1204
8.384.3.2 ~proxy . . . 1204
8.384.4 Member Function Documentation . . . 1204
8.384.4.1 operator typename I::value . . . 1204
8.384.4.2 operator V . . . 1204
8.384.4.3 to_value . . . 1205
8.385 mln::value::rgb< n > Struct Template Reference . . . 1206
8.385.1 Detailed Description . . . 1206
8.385.2 Constructor & Destructor Documentation . . . 1206
8.385.2.1 rgb . . . 1206

8.385.2.2 rgb . . . 1207
8.385.2.3 rgb . . . 1207
8.385.2.4 rgb . . . 1207
8.385.3 Member Function Documentation . . . 1207
8.385.3.1 operator= . . . 1207
8.385.3.2 red . . . 1207
8.385.4 Member Data Documentation . . . 1207
8.385.4.1 zero . . . 1207
8.386 mln::value::set< T > Struct Template Reference . . . 1208
8.386.1 Detailed Description . . . 1208
8.386.2 Member Function Documentation . . . 1208
8.386.2.1 the . . . 1208
8.387 mln::value::sign Class Reference . . . 1209
8.387.1 Detailed Description . . . 1209
8.387.2 Member Typedef Documentation . . . 1210
8.387.2.1 enc . . . 1210
8.387.2.2 equiv . . . 1210
8.387.3 Constructor & Destructor Documentation . . . 1210
8.387.3.1 sign . . . 1210
8.387.3.2 sign . . . 1210
8.387.3.3 sign . . . 1210
8.387.4 Member Function Documentation . . . 1210
8.387.4.1 operator int . . . 1210
8.387.4.2 operator= . . . 1210
8.387.5 Member Data Documentation . . . 1210
8.387.5.1 one . . . 1210
8.387.5.2 zero . . . 1210
8.388 mln::value::stack_image< n, I > Struct Template Reference . . . 1211
8.388.1 Detailed Description . . . 1211
8.388.2 Member Typedef Documentation . . . 1212
8.388.2.1 domain_t . . . 1212
8.388.2.2 lvalue . . . 1212
8.388.2.3 psite . . . 1212
8.388.2.4 rvalue . . . 1212
8.388.2.5 skeleton . . . 1212
8.388.2.6 value . . . 1212

8.388.3 Constructor & Destructor Documentation . . . 1212
8.388.3.1 stack_image . . . 1212
8.388.4 Member Function Documentation . . . 1213
8.388.4.1 is_valid . . . 1213
8.388.4.2 operator() . . . 1213
8.388.4.3 operator() . . . 1213
8.389 mln::Vertex< E > Struct Template Reference . . . 1214
8.389.1 Detailed Description . . . 1214
8.390 mln::vertex_image< P, V, G > Class Template Reference . . . 1215
8.390.1 Detailed Description . . . 1215
8.390.2 Member Typedef Documentation . . . 1215
8.390.2.1 graph_t . . . 1215
8.390.2.2 nbh_t . . . 1216
8.390.2.3 site_function_t . . . 1216
8.390.2.4 skeleton . . . 1216
8.390.2.5 win_t . . . 1216
8.390.3 Constructor & Destructor Documentation . . . 1216
8.390.3.1 vertex_image . . . 1216
8.390.4 Member Function Documentation . . . 1216
8.390.4.1 operator() . . . 1216
8.391 mln::w_window< D, W > Struct Template Reference . . . 1217
8.391.1 Detailed Description . . . 1218
8.391.2 Member Typedef Documentation . . . 1218
8.391.2.1 bkd_qiter . . . 1218
8.391.2.2 dpsite . . . 1218
8.391.2.3 fwd_qiter . . . 1218
8.391.2.4 weight . . . 1218
8.391.3 Constructor & Destructor Documentation . . . 1218
8.391.3.1 w_window . . . 1218
8.391.4 Member Function Documentation . . . 1219
8.391.4.1 clear . . . 1219
8.391.4.2 insert . . . 1219
8.391.4.3 is_symmetric . . . 1219
8.391.4.4 std_vector . . . 1219
8.391.4.5 sym . . . 1219
8.391.4.6 w . . . 1219

8.391.4.7 weights . . . 1219
8.391.4.8 win . . . 1220
8.391.5 Friends And Related Function Documentation . . . 1220
8.391.5.1 operator<< . . . 1220
8.391.5.2 operator== . . . 1220
8.392 mln::Weighted_Window< E > Struct Template Reference . . . 1221
8.392.1 Detailed Description . . . 1221
8.392.2 Friends And Related Function Documentation . . . 1221
8.392.2.1 operator- . . . 1221
8.393 mln::win::backdiag2d Struct Reference . . . 1222
8.393.1 Detailed Description . . . 1222
8.393.2 Constructor & Destructor Documentation . . . 1222
8.393.2.1 backdiag2d . . . 1222
8.393.3 Member Function Documentation . . . 1222
8.393.3.1 length . . . 1222
8.394 mln::win::ball< G, C > Struct Template Reference . . . 1223
8.394.1 Detailed Description . . . 1223
8.394.2 Constructor & Destructor Documentation . . . 1223
8.394.2.1 ball . . . 1223
8.394.3 Member Function Documentation . . . 1223
8.394.3.1 diameter . . . 1223
8.395 mln::win::cube3d Struct Reference . . . 1224
8.395.1 Detailed Description . . . 1224
8.395.2 Constructor & Destructor Documentation . . . 1224
8.395.2.1 cube3d . . . 1224
8.395.3 Member Function Documentation . . . 1225
8.395.3.1 length . . . 1225
8.396 mln::win::cuboid3d Struct Reference . . . 1226
8.396.1 Detailed Description . . . 1226
8.396.2 Constructor & Destructor Documentation . . . 1227
8.396.2.1 cuboid3d . . . 1227
8.396.3 Member Function Documentation . . . 1227
8.396.3.1 depth . . . 1227
8.396.3.2 height . . . 1227
8.396.3.3 volume . . . 1227
8.396.3.4 width . . . 1227

8.397 mln::win::diag2d Struct Reference 1228
8.397.1 Detailed Description 1228
8.397.2 Constructor & Destructor Documentation 1228
8.397.2.1 diag2d 1228
8.397.3 Member Function Documentation 1228
8.397.3.1 length 1228
8.398 mln::win::line< M, i, C > Struct Template Reference 1229
8.398.1 Detailed Description 1229
8.398.2 Member Enumeration Documentation 1229
8.398.2.1 "@83 1229
8.398.3 Constructor & Destructor Documentation 1230
8.398.3.1 line 1230
8.398.4 Member Function Documentation 1230
8.398.4.1 length 1230
8.398.4.2 size 1230
8.399 mln::win::multiple< W, F > Class Template Reference 1231
8.399.1 Detailed Description 1231
8.400 mln::win::multiple_size< n, W, F > Class Template Reference 1232
8.400.1 Detailed Description 1232
8.401 mln::win::octagon2d Struct Reference 1233
8.401.1 Detailed Description 1233
8.401.2 Constructor & Destructor Documentation 1233
8.401.2.1 octagon2d 1233
8.401.3 Member Function Documentation 1234
8.401.3.1 area 1234
8.401.3.2 length 1234
8.402 mln::win::rectangle2d Struct Reference 1235
8.402.1 Detailed Description 1235
8.402.2 Constructor & Destructor Documentation 1235
8.402.2.1 rectangle2d 1235
8.402.3 Member Function Documentation 1236
8.402.3.1 area 1236
8.402.3.2 height 1236
8.402.3.3 std_vector 1236
8.402.3.4 width 1236
8.403 mln::Window< E > Struct Template Reference 1237

8.403.1 Detailed Description . . . 1237
8.404 mln::window< D > Class Template Reference . . . 1238
8.404.1 Detailed Description . . . 1239
8.404.2 Member Typedef Documentation . . . 1239
8.404.2.1 bkd_qiter . . . 1239
8.404.2.2 fwd_qiter . . . 1239
8.404.2.3 qiter . . . 1240
8.404.2.4 regular . . . 1240
8.404.3 Constructor & Destructor Documentation . . . 1240
8.404.3.1 window . . . 1240
8.404.4 Member Function Documentation . . . 1240
8.404.4.1 clear . . . 1240
8.404.4.2 delta . . . 1240
8.404.4.3 dp . . . 1240
8.404.4.4 has . . . 1240
8.404.4.5 insert . . . 1240
8.404.4.6 insert . . . 1241
8.404.4.7 insert . . . 1241
8.404.4.8 is_centered . . . 1241
8.404.4.9 is_empty . . . 1241
8.404.4.10 is_symmetric . . . 1241
8.404.4.11 print . . . 1241
8.404.4.12 size . . . 1241
8.404.4.13 std_vector . . . 1242
8.404.4.14 sym . . . 1242
8.404.5 Friends And Related Function Documentation . . . 1242
8.404.5.1 operator== . . . 1242
8.405 mln::world::inter_pixel::is_separator Struct Reference . . . 1243
8.405.1 Detailed Description . . . 1243

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 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

Module Index

2.1 Modules

Here is a list of all modules:

- Types . . . 73
  - Graphes . . . 66
  - Images . . . 67
    - Basic types . . . 68
    - Image morphers . . . 69
    - Values morphers . . . 70
    - Domain morphers . . . 71
    - Identity morphers . . . 72
  - Neighborhoods . . . 79
    - 1D neighborhoods . . . 80
    - 2D neighborhoods . . . 81
    - 3D neighborhoods . . . 83
  - Site sets . . . 86
    - Basic types . . . 87
    - Graph based . . . 88
    - Complex based . . . 89
    - Sparse types . . . 90
    - Queue based . . . 91
  - Utilities . . . 92
  - Windows . . . 93
    - 1D windows . . . 94
    - 2D windows . . . 95
    - 3D windows . . . 98
    - N-D windows . . . 100
    - Multiple windows . . . 101
- Accumulators . . . 74
  - On site sets . . . 61
  - On images . . . 62
  - On values . . . 63
  - Multiple accumulators . . . 65
- Routines . . . 75
- Canvas . . . 76

Functions 77
v2w2v functions 102
v2w_w2v functions 103
vv2b functions 104

Chapter 3

Namespace Index

3.1 Namespace List

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

mln (Mln/convert/to_image.hh) . . . 105
mln::accu (Namespace of accumulators) . . . 142
mln::accu::image (Namespace of accumulator image routines) . . . 146
mln::accu::impl (Implementation namespace of accumulator namespace) . . . 147
mln::accu::logic (Namespace of logical accumulators) . . . 148
mln::accu::math (Namespace of mathematic accumulators) . . . 149
mln::accu::meta::logic (Namespace of logical meta-accumulators) . . . 150
mln::accu::meta::math (Namespace of mathematic meta-accumulators) . . . 151
mln::accu::meta::shape (Namespace of shape meta-accumulators) . . . 152
mln::accu::meta::stat (Namespace of statistical meta-accumulators) . . . 153
mln::accu::shape (Namespace of shape accumulators) . . . 154
mln::accu::stat (Namespace of statistical accumulators) . . . 155
mln::algebra (Namespace of algebraic structure) . . . 157
mln::arith (Namespace of arithmetic) . . . 159
mln::arith::impl (Implementation namespace of arith namespace) . . . 171
mln::arith::impl::generic (Generic implementation namespace of arith namespace) . . . 172
mln::binarization (Namespace of "point-wise" expression tools) . . . 173
mln::border (Namespace of routines related to image virtual (outer) border) . . . 174
mln::border::impl (Implementation namespace of border namespace) . . . 178
mln::border::impl::generic (Generic implementation namespace of border namespace) . . . 179
mln::canvas (Namespace of canvas) . . . 180
mln::canvas::browsing (Namespace of browsing canvas) . . . 182
mln::canvas::impl (Implementation namespace of canvas namespace) . . . 183
mln::canvas::morpho (Namespace of morphological canvas) . . . 184
mln::convert (Namespace of conversion routines) . . . 185
mln::data (Namespace of image processing routines related to pixel data) . . . 191
mln::data::approx (Namespace of image processing routines related to pixel levels with approximation) . . . 203
mln::data::approx::impl (Implementation namespace of data::approx namespace) . . . 205
mln::data::impl (Implementation namespace of data namespace) . . . 206
mln::data::impl::generic (Generic implementation namespace of data namespace) . . . 208
mln::data::naive (Namespace of image processing routines related to pixel levels with naive approach) . . . 213

mln::data::naive::impl (Implementation namespace of data::naive namespace) . . . 214
mln::debug (Namespace of routines that help to debug) . . . 215
mln::debug::impl (Implementation namespace of debug namespace) . . . 220
mln::def (Namespace for core definitions) . . . 221
mln::display (Namespace of routines that help to display images) . . . 222
mln::display::impl (Implementation namespace of display namespace) . . . 223
mln::display::impl::generic (Generic implementation namespace of display namespace) . . . 224
mln::doc (The namespace mln::doc is only for documentation purpose) . . . 225
mln::draw (Namespace of drawing routines) . . . 227
mln::estim (Namespace of estimation materials) . . . 229
mln::extension (Namespace of extension tools) . . . 231
mln::fun (Namespace of functions) . . . 234
mln::fun::access (Namespace for access functions) . . . 236
mln::fun::i2v (Namespace of integer-to-value functions) . . . 237
mln::fun::p2b (Namespace of functions from point to boolean) . . . 238
mln::fun::p2p (Namespace of functions from grid point to grid point) . . . 239
mln::fun::p2v (Namespace of functions from point to value) . . . 240
mln::fun::stat (Namespace of statistical functions) . . . 241
mln::fun::v2b (Namespace of functions from value to logic value) . . . 242
mln::fun::v2i (Namespace of value-to-integer functions) . . . 243
mln::fun::v2v (Namespace of functions from value to value) . . . 244
mln::fun::v2w2v (Namespace of bijective functions) . . . 246
mln::fun::v2w_w2v (Namespace of functions from value to value) . . . 247
mln::fun::vv2b (Namespace of functions from value to value) . . . 248
mln::fun::vv2v (Namespace of functions from a couple of values to a value) . . . 249
mln::fun::x2p (Namespace of functions from point to value) . . . 250
mln::fun::x2v (Namespace of functions from vector to value) . . . 251
mln::fun::x2x (Namespace of functions from vector to vector) . . . 252
mln::geom (Namespace of all things related to geometry) . . . 253
mln::geom::impl (Implementation namespace of geom namespace) . . . 264
mln::graph (Namespace of graph related routines) . . . 266
mln::grid (Namespace of grids definitions) . . . 269
mln::histo (Namespace of histograms) . . . 270
mln::histo::impl (Implementation namespace of histo namespace) . . . 271
mln::histo::impl::generic (Generic implementation namespace of histo namespace) . . . 272
mln::impl (Implementation namespace of mln namespace) . . . 273
mln::io (Namespace of input/output handling) . . . 274
mln::io::cloud (Namespace of cloud input/output handling) . . . 276
mln::io::dicom (Namespace of DICOM input/output handling) . . . 277
mln::io::dump (Namespace of dump input/output handling) . . . 278
mln::io::fits (Namespace of fits input/output handling) . . . 279
mln::io::magick (Namespace of magick input/output handling) . . . 280
mln::io::off (Namespace of off input/output handling) . . . 281
mln::io::pbm (Namespace of pbm input/output handling) . . . 283
mln::io::pbm::impl (Namespace of pbm implementation details) . . . 285
mln::io::pbms (Namespace of pbms input/output handling) . . . 286
mln::io::pbms::impl (Namespace of pbms implementation details) . . . 287
mln::io::pfm (Namespace of pfm input/output handling) . . . 288
mln::io::pfm::impl (Implementation namespace of pfm namespace) . . . 290
mln::io::pgm (Namespace of pgm input/output handling) . . . 291
mln::io::pgms (Namespace of pgms input/output handling) . . . 293
mln::io::plot (Namespace of plot input/output handling) . . . 294
mln::io::pnm (Namespace of pnm input/output handling) . . . 296
mln::io::pnm::impl (Namespace of pnm's implementation details) . . . 298

mln::io::pnms (Namespace of pnms input/output handling) . . . 299
mln::io::ppm (Namespace of ppm input/output handling) . . . 300
mln::io::ppms (Namespace of ppms input/output handling) . . . 302
mln::io::tiff (Namespace of tiff input/output handling) . . . 303
mln::io::txt (Namespace of txt input/output handling) . . . 304
mln::labeling (Namespace of labeling routines) . . . 305
mln::labeling::impl (Implementation namespace of labeling namespace) . . . 318
mln::labeling::impl::generic (Generic implementation namespace of labeling namespace) . . . 319
mln::linear (Namespace of linear image processing routines) . . . 321
mln::linear::impl (Namespace of linear image processing routines implementation details) . . . 325
mln::linear::local (Specializations of local linear routines) . . . 326
mln::linear::local::impl (Namespace of local linear routines implementation details) . . . 327
mln::literal (Namespace of literals) . . . 328
mln::logical (Namespace of logic) . . . 334
mln::logical::impl (Implementation namespace of logical namespace) . . . 337
mln::logical::impl::generic (Generic implementation namespace of logical namespace) . . . 338
mln::make (Namespace of routines that help to make Milena's objects) . . . 339
mln::math (Namespace of mathematical routines) . . . 361
mln::metal (Namespace of meta-programming tools) . . . 362
mln::metal::impl (Implementation namespace of metal namespace) . . . 363
mln::metal::math (Namespace of static mathematical functions) . . . 364
mln::metal::math::impl (Implementation namespace of metal::math namespace) . . . 365
mln::morpho (Namespace of mathematical morphology routines) . . . 366
mln::morpho::approx (Namespace of approximate mathematical morphology routines) . . . 375
mln::morpho::attribute (Namespace of attributes used in mathematical morphology) . . . 376
mln::morpho::closing::approx (Namespace of approximate mathematical morphology closing routines) . . . 377
mln::morpho::elementary (Namespace of image processing routines of elementary mathematical morphology) . . . 378
mln::morpho::impl (Namespace of mathematical morphology routines implementations) . . . 380
mln::morpho::impl::generic (Namespace of mathematical morphology routines generic implementations) . . . 381
mln::morpho::opening::approx (Namespace of approximate mathematical morphology opening routines) . . . 382
mln::morpho::reconstruction (Namespace of morphological reconstruction routines) . . . 383
mln::morpho::reconstruction::by_dilation (Namespace of morphological reconstruction by dilation routines) . . . 384
mln::morpho::reconstruction::by_erosion (Namespace of morphological reconstruction by erosion routines) . . . 385
mln::morpho::tree (Namespace of morphological tree-related routines) . . . 386
mln::morpho::tree::filter (Namespace for attribute filtering) . . . 391
mln::morpho::watershed (Namespace of morphological watershed routines) . . . 394
mln::morpho::watershed::watershed (Namespace of morphological watershed routines implementations) . . . 396
mln::morpho::watershed::watershed::generic (Namespace of morphological watershed routines generic implementations) . . . 397
mln::norm (Namespace of norms) . . . 398
mln::norm::impl (Implementation namespace of norm namespace) . . . 400
mln::opt (Namespace of optional routines) . . . 401
mln::opt::impl (Implementation namespace of opt namespace) . . . 403
mln::pw (Namespace of "point-wise" expression tools) . . . 404
mln::registration (Namespace of "point-wise" expression tools) . . . 405
mln::select (Select namespace (FIXME doc)) . . . 408
mln::set (Namespace of image processing routines related to pixel sets) . . . 409

mln::subsampling (Namespace of "point-wise" expression tools) 412
mln::tag (Namespace of image processing routines related to tags) 413
mln::test (Namespace of image processing routines related to pixel tests) 414
mln::test::impl (Implementation namespace of test namespace) 416
mln::topo (Namespace of "point-wise" expression tools) 417
mln::trace (Namespace of routines related to the trace mechanism) 427
mln::trait (Namespace where traits are defined) 428
mln::transform (Namespace of transforms) 429
mln::util (Namespace of tools using for more complex algorithm) 434
mln::util::impl (Implementation namespace of util namespace) 441
mln::value (Namespace of materials related to pixel value types) 442
mln::value::impl (Implementation namespace of value namespace) 453
mln::win (Namespace of image processing routines related to win) 454

Chapter 4

Class Index

4.1 Class Hierarchy

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

mln::fun::internal::ch_function_value_impl< F, V >
  mln::fun::v2v::ch_function_value< F, V > 726
mln::metal::converts_to< T, U > 877
mln::util::internal::edge_impl_< G >
  mln::util::edge< G > 1115
mln::topo::face< D > 1065
  mln::topo::algebraic_face< D > 1050
mln::Generalized_Pixel< bkd_pixter1d< I > > 778
  Pixel_Iterator< bkd_pixter1d< I > >
    pixel_iterator_base_< I, bkd_pixter1d< I > >
      backward_pixel_iterator_base_< I, bkd_pixter1d< I > >
        mln::bkd_pixter1d< I > 582
mln::Generalized_Pixel< bkd_pixter2d< I > > 778
  Pixel_Iterator< bkd_pixter2d< I > >
    pixel_iterator_base_< I, bkd_pixter2d< I > >
      backward_pixel_iterator_base_< I, bkd_pixter2d< I > >
        mln::bkd_pixter2d< I > 584
mln::Generalized_Pixel< bkd_pixter3d< I > > 778
  Pixel_Iterator< bkd_pixter3d< I > >
    pixel_iterator_base_< I, bkd_pixter3d< I > >
      backward_pixel_iterator_base_< I, bkd_pixter3d< I > >
        mln::bkd_pixter3d< I > 586
mln::Generalized_Pixel< dpoints_bkd_pixter< I > > 778
  Pixel_Iterator< dpoints_bkd_pixter< I > >
    mln::dpoints_bkd_pixter< I > 695
mln::Generalized_Pixel< dpoints_fwd_pixter< I > > 778
  Pixel_Iterator< dpoints_fwd_pixter< I > >
    mln::dpoints_fwd_pixter< I > 698
mln::Generalized_Pixel< E > 778
  Pixel_Iterator< E >
    pixel_iterator_base_< I, E >

mln::doc::Generalized_Pixel< E > 657
mln::doc::Pixel_Iterator< E > 670
mln::Generalized_Pixel< fwd_pixter1d< I > > 778
Pixel_Iterator< fwd_pixter1d< I > >
pixel_iterator_base_< I, fwd_pixter1d< I > >
forward_pixel_iterator_base_< I, fwd_pixter1d< I > >
mln::fwd_pixter1d< I > 770
mln::Generalized_Pixel< fwd_pixter2d< I > > 778
Pixel_Iterator< fwd_pixter2d< I > >
pixel_iterator_base_< I, fwd_pixter2d< I > >
forward_pixel_iterator_base_< I, fwd_pixter2d< I > >
mln::fwd_pixter2d< I > 772
mln::Generalized_Pixel< fwd_pixter3d< I > > 778
Pixel_Iterator< fwd_pixter3d< I > >
pixel_iterator_base_< I, fwd_pixter3d< I > >
forward_pixel_iterator_base_< I, fwd_pixter3d< I > >
mln::fwd_pixter3d< I > 774
mln::Generalized_Pixel< pixel< I > > 778
mln::pixel< I > 991
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 > 1211
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< I > 839
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 > 1030
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 > 762
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 > 1008
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 > 706
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 > 1215
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 > 810
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 > 708
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 > 1096
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 > 637
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 > 710
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 > 713
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 > 716
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 > 836
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 > 901
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 > 993
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 > 1012
mln::internal::image_base< I::value, I::domain_t, tr_image< S, I, T > >
image_morpher< I, I::value, I::domain_t, tr_image< S, I, T > >
image_identity< I, I::domain_t, tr_image< S, I, T > >
mln::tr_image< S, I, T > 1091
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 > 1027
mln::internal::image_base< I::value, p_transformed< I::domain_t, F >, transformed_image< I, F >
>
image_morpher< I, I::value, p_transformed< I::domain_t, F >, transformed_image< I, F > >
image_domain_morpher< I, p_transformed< I::domain_t, F >, transformed_image< I, F > >
mln::transformed_image< I, F > 1094
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 > 1025
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 > > 810
mln::image2d_h< V > 828
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 > 842
mln::internal::image_base< T, box1d, image1d< T > >
image_primary< T, box1d, image1d< T > >
mln::image1d< T > 818
mln::internal::image_base< T, box3d, image3d< T > >
image_primary< T, box3d, image3d< T > >
mln::image3d< T > 831
mln::internal::image_base< T, I::domain_t, E >
image_morpher< I, T, I::domain_t, E >
mln::internal::image_base< T, mln::box2d, image2d< T > >
image_primary< T, mln::box2d, image2d< T > >
mln::image2d< T > 823
mln::internal::image_base< T, S, E >
mln::internal::image_base< T, S, flat_image< T, S > >
image_primary< T, S, flat_image< T, S > >
mln::flat_image< T, S > 719
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 > 623
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_< 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_< 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_< transformed_image< I, F >, mln::metal::equal< mln_trait_image_speed(transformed_image< I, F >), 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::value::Integer< graylevel< n > > 1192
mln::value::graylevel< n > 1179
mln::value::Integer< int_s< n > > 1192
mln::value::int_s< n > 1185
mln::value::Integer< int_u< n > > 1192

mln::value::int_u< n > . . . 1187
mln::value::Integer< int_u_sat< n > > . . . 1192
mln::value::int_u_sat< n > . . . 1189
mln::value::Integer< object_id< Tag, V > > . . . 1192
mln::util::object_id< Tag, V > . . . 1140
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 > . . . 806
mln::algebra::h_mat< d, T > . . . 578
mln::algebra::h_vec< d, C > . . . 580
mln::canvas::chamfer< F > . . . 621
mln::category< R(∗)(A) > . . . 622
mln::Delta_Point_Site< void > . . . 641
mln::doc::Accumulator< E > . . . 642
mln::doc::Generalized_Pixel< E > . . . 657
mln::doc::Object< E > . . . 669
mln::doc::Point_Site< E > . . . 673
mln::Edge< E > . . . 705
mln::fun::x2p::closest_point< P > . . . 750
mln::fun::x2x::composed< T2, T1 > . . . 753
mln::Function< void > . . . 765
mln::Gdpoint< void > . . . 777
mln::Generalized_Pixel< E > . . . 778
mln::geom::complex_geometry< D, P > . . . 779
mln::graph::attribute::card_t . . . 788
mln::graph::attribute::representative_t . . . 789
mln::histo::array< T > . . . 814
mln::metal::ands< E1, E2, E3, E4, E5, E6, E7, E8 > . . . 876
mln::metal::bool_< false >
mln::metal::equal< T1::coord, T2::coord > . . . 878
mln::metal::equal< T1::point, T2::point > . . . 878
mln::metal::equal< T1, T2 > . . . 878
mln::metal::converts_to< T, U > . . . 877
mln::metal::goes_to< T, U > . . . 879
mln::metal::is< T, U > . . . 880
mln::metal::is_a< T, M > . . . 881
mln::metal::is_not< T, U > . . . 882
mln::metal::is_not_a< T, M > . . . 883
mln::Neighborhood< void > . . . 899
mln::Object< E > . . . 900
mln::Proxy< void > . . . 1005
mln::Pseudo_Site< void > . . . 1007
mln::registration::closest_point_basic< P > . . . 1009
mln::registration::closest_point_with_map< P > . . . 1010
mln::select::p_of< P > . . . 1013
mln::Site< void > . . . 1015
mln::Site_Proxy< void > . . . 1019
mln::Site_Set< void > . . . 1024
mln::thru_image< I, F > . . . 1029
mln::topo::complex< D > . . . 1062
mln::topo::face< D > . . . 1065
mln::topo::n_face< N, D > . . . 1077
mln::topo::n_faces_set< N, D > . . . 1085

mln::util::adjacency_matrix< V > 1098
mln::util::branch< T > 1106
mln::util::branch_iter< T > 1108
mln::util::branch_iter_ind< T > 1110
mln::util::greater_point< I > 1128
mln::util::greater_psite< I > 1129
mln::util::head< T, R > 1130
mln::util::ilcell< T > 1132
mln::util::node< T, R > 1139
mln::util::ord< T > 1142
mln::util::pix< I > 1145
mln::util::tracked_ptr< T > 1159
mln::util::tree< T > 1161
mln::util::tree_node< T > 1163
mln::value::float01 1174
mln::value::Integer< E > 1192
mln::value::Integer< void > 1193
mln::value::set< T > 1208
mln::value::sign 1209
mln::Vertex< E > 1214
mln::topo::n_face< N, D > 1077
mln::topo::algebraic_n_face< N, D > 1054
mln::internal::neighborhood_base< graph_elt_window< G, S >, neighb< graph_elt_window< G, S > > >
mln::neighb< graph_elt_window< G, S > > 896
mln::graph_elt_neighborhood< G, S > 790
mln::internal::neighborhood_base< graph_elt_window_if< G, S, I >, neighb< graph_elt_window_if< G, S, I > > >
mln::neighb< graph_elt_window_if< G, S, I > > 896
mln::graph_elt_neighborhood_if< G, S, I > 792
mln::internal::neighborhood_base< W, neighb< W > >
mln::neighb< W > 896
mln::Object< abs > 900
mln::Meta_Function< abs > 873
mln::Meta_Function_v2v< abs > 874
mln::Object< abs< V > > 900
mln::Function< abs< V > > 764
mln::Function_v2v< abs< V > > 767
mln::Object< accu_result > 900
mln::Meta_Function< accu_result > 873
mln::Meta_Function_v2v< accu_result > 874
mln::Object< adj_higher_dim_connected_n_face_bkd_iter< D > > 900
mln::Iterator< adj_higher_dim_connected_n_face_bkd_iter< D > > 838
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 > 1032
mln::Object< adj_higher_dim_connected_n_face_fwd_iter< D > > 900
mln::Iterator< adj_higher_dim_connected_n_face_fwd_iter< D > > 838

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 > 1034
mln::Object< adj_higher_face_bkd_iter< D > > 900
mln::Iterator< adj_higher_face_bkd_iter< D > > 838
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 > 1036
mln::Object< adj_higher_face_fwd_iter< D > > 900
mln::Iterator< adj_higher_face_fwd_iter< D > > 838
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 > 1037
mln::Object< adj_lower_dim_connected_n_face_bkd_iter< D > > 900
mln::Iterator< adj_lower_dim_connected_n_face_bkd_iter< D > > 838
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 > 1038
mln::Object< adj_lower_dim_connected_n_face_fwd_iter< D > > 900
mln::Iterator< adj_lower_dim_connected_n_face_fwd_iter< D > > 838
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 > 1040
mln::Object< adj_lower_face_bkd_iter< D > > 900
mln::Iterator< adj_lower_face_bkd_iter< D > > 838
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 > 1042
mln::Object< adj_lower_face_fwd_iter< D > > 900
mln::Iterator< adj_lower_face_fwd_iter< D > > 838
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 > . . . 1043
mln::Object< adj_lower_higher_face_bkd_iter< D > > . . . 900
mln::Iterator< adj_lower_higher_face_bkd_iter< D > > . . . 838
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 > . . . 1044
mln::Object< adj_lower_higher_face_fwd_iter< D > > . . . 900
mln::Iterator< adj_lower_higher_face_fwd_iter< D > > . . . 838
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 > . . . 1045
mln::Object< adj_m_face_bkd_iter< D > > . . . 900
mln::Iterator< adj_m_face_bkd_iter< D > > . . . 838
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 > . . . 1046
mln::Object< adj_m_face_fwd_iter< D > > . . . 900
mln::Iterator< adj_m_face_fwd_iter< D > > . . . 838
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 > . . . 1048
mln::Object< all_to< T > > . . . 900
mln::Function< all_to< T > > . . . 764
mln::Object< antilogy > . . . 900
mln::Function< antilogy > . . . 764
mln::Function_v2v< antilogy > . . . 767
mln::Function_v2b< antilogy > . . . 766
mln::fun::p2b::antilogy . . . 722
mln::Object< array1d< T, Size > > . . . 900
mln::Object< array2d< T, r, c > > . . . 900
mln::Object< array3d< T, s, r, c > > . . . 900
mln::Object< array_bkd_iter< T > > . . . 900
mln::Proxy< array_bkd_iter< T > > . . . 1004
mln::Object< array_fwd_iter< T > > . . . 900
mln::Proxy< array_fwd_iter< T > > . . . 1004
mln::Object< asc_propagation > . . . 900
mln::Object< backdiag2d > . . . 900
mln::Window< backdiag2d > . . . 1237
window_base< dpoint2d, backdiag2d >
classical_window_base< dpoint2d, backdiag2d >

mln::win::backdiag2d . . . 1222
mln::Object< backdiagonal2d_t > . . . 900
mln::Browsing< backdiagonal2d_t > . . . 600
mln::canvas::browsing::backdiagonal2d_t . . . 601
mln::Object< ball< G, C > > . . . 900
mln::Window< ball< G, C > > . . . 1237
window_base< dpoint< G, C >, ball< G, C > >
classical_window_base< dpoint< G, C >, ball< G, C > >
mln::win::ball< G, C > . . . 1223
mln::Object< bbox > . . . 900
mln::Meta_Accumulator< bbox > . . . 871
mln::accu::meta::shape::bbox . . . 511
mln::Object< bbox< P > > . . . 900
mln::Proxy< bbox< P > > . . . 1004
mln::Accumulator< bbox< P > > . . . 577
base< const box< P > &, bbox< P > >
mln::accu::shape::bbox< P > . . . 533
mln::Object< big_chess< B > > . . . 900
mln::Function< big_chess< B > > . . . 764
mln::Function_v2v< big_chess< B > > . . . 767
mln::Object< bin_off_loader > . . . 900
mln::Object< bin_off_saver > . . . 900
mln::Object< binary< Fun, T1, T2 > > . . . 900
mln::Function< binary< Fun, T1, T2 > > . . . 764
mln::Function_v2v< binary< Fun, T1, T2 > > . . . 767
mln::Object< bkd_pixter1d< I > > . . . 900
mln::Iterator< bkd_pixter1d< I > > . . . 838
Pixel_Iterator< bkd_pixter1d< I > >
mln::Object< bkd_pixter2d< I > > . . . 900
mln::Iterator< bkd_pixter2d< I > > . . . 838
Pixel_Iterator< bkd_pixter2d< I > >
mln::Object< bkd_pixter3d< I > > . . . 900
mln::Iterator< bkd_pixter3d< I > > . . . 838
Pixel_Iterator< bkd_pixter3d< I > >
mln::Object< black_t > . . . 900
mln::Literal< black_t > . . . 845
mln::literal::black_t . . . 847
mln::Object< blue > . . . 900
mln::Meta_Function< blue > . . . 873
mln::Meta_Function_v2v< blue > . . . 874
mln::Object< blue_t > . . . 900
mln::Literal< blue_t > . . . 845
mln::literal::blue_t . . . 848
mln::Object< box< P > > . . . 900
mln::Site_Set< box< P > > . . . 1020
mln::Box< box< P > > . . . 595
mln::box< P > . . . 588
mln::Object< box_runstart_piter< P > > . . . 900

mln::Proxy< box_runstart_piter< P > > . . . 1004
mln::Site_Proxy< box_runstart_piter< P > > . . . 1018
mln::Site_Iterator< box_runstart_piter< P > > . . . 1016
site_iterator_base< box< P >, box_runstart_piter< P > >
site_set_iterator_base< box< P >, box_runstart_piter< P > >
mln::box_runstart_piter< P > . . . 598
mln::Object< breadth_first_search_t > . . . 900
mln::Browsing< breadth_first_search_t > . . . 600
graph_first_search_t< breadth_first_search_t, std::queue >
mln::canvas::browsing::breadth_first_search_t . . . 603
mln::Object< brown_t > . . . 900
mln::Literal< brown_t > . . . 845
mln::literal::brown_t . . . 849
mln::Object< C< R(∗)(A)> > . . . 900
mln::Function< C< R(∗)(A)> > . . . 764
mln::Function_v2v< C< R(∗)(A)> > . . . 767
mln::Object< card< I > > . . . 900
mln::Proxy< card< I > > . . . 1004
mln::Accumulator< card< I > > . . . 577
base< unsigned, card< I > >
mln::morpho::attribute::card< I > . . . 884
mln::Object< cast< V > > . . . 900
mln::Function< cast< V > > . . . 764
mln::Function_v2v< cast< V > > . . . 767
mln::Object< center > . . . 900
mln::Meta_Accumulator< center > . . . 871
mln::accu::meta::center . . . 491
mln::Object< center< P, V > > . . . 900
mln::Proxy< center< P, V > > . . . 1004
mln::Accumulator< center< P, V > > . . . 577
base< V, center< P, V > >
mln::accu::center< P, V > . . . 457
mln::Object< center_only_iter< D > > . . . 900
mln::Iterator< center_only_iter< D > > . . . 838
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 > . . . 1058
mln::Object< centered_bkd_iter_adapter< D, I > > . . . 900
mln::Iterator< centered_bkd_iter_adapter< D, I > > . . . 838
complex_relative_iterator_sequence< I, center_only_iter< D >, centered_bkd_iter_adapter<
D, I > >
mln::topo::centered_bkd_iter_adapter< D, I > . . . 1060
mln::Object< centered_fwd_iter_adapter< D, I > > . . . 900
mln::Iterator< centered_fwd_iter_adapter< D, I > > . . . 838
complex_relative_iterator_sequence< center_only_iter< D >, I, centered_fwd_iter_adapter<
D, I > >
mln::topo::centered_fwd_iter_adapter< D, I > . . . 1061

mln::Object< ch_function_value< F, V > > . . . 900
mln::Function< ch_function_value< F, V > > . . . 764
mln::Function_v2v< ch_function_value< F, V > > . . . 767
mln::fun::v2v::ch_function_value< F, V > . . . 726
mln::Object< ch_piter_image< I, Fwd > > . . . 900
mln::Image< ch_piter_image< I, Fwd > > . . . 815
mln::Object< chess > . . . 900
mln::Function< chess > . . . 764
mln::Function_v2v< chess > . . . 767
mln::Function_v2b< chess > . . . 766
mln::Object< col > . . . 900
mln::Meta_Function< col > . . . 873
mln::Meta_Function_v2v< col > . . . 874
mln::Object< colorize > . . . 900
mln::Function< colorize > . . . 764
mln::Function_v2v< colorize > . . . 767
mln::Object< comp > . . . 900
mln::Meta_Function< comp > . . . 873
mln::Meta_Function_v2v< comp > . . . 874
mln::Object< comp_count > . . . 900
mln::Meta_Function< comp_count > . . . 873
mln::Meta_Function_v2v< comp_count > . . . 874
mln::Object< complex_image< D, G, V > > . . . 900
mln::Image< complex_image< D, G, V > > . . . 815
mln::Object< complex_neighborhood_bkd_piter< I, G, N > > . . . 900
mln::Proxy< complex_neighborhood_bkd_piter< I, G, N > > . . . 1004
mln::Site_Proxy< complex_neighborhood_bkd_piter< I, G, N > > . . . 1018
mln::Site_Iterator< complex_neighborhood_bkd_piter< I, G, N > > . . . 1016
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 > . . . 626
mln::Object< complex_neighborhood_fwd_piter< I, G, N > > . . . 900
mln::Proxy< complex_neighborhood_fwd_piter< I, G, N > > . . . 1004
mln::Site_Proxy< complex_neighborhood_fwd_piter< I, G, N > > . . . 1018
mln::Site_Iterator< complex_neighborhood_fwd_piter< I, G, N > > . . . 1016
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 > . . . 628
mln::Object< complex_psite< D, G > > . . . 900
mln::Proxy< complex_psite< D, G > > . . . 1004
mln::Site_Proxy< complex_psite< D, G > > . . . 1018
mln::Pseudo_Site< complex_psite< D, G > > . . . 1006
pseudo_site_base_< const G::site &, complex_psite< D, G > >
mln::complex_psite< D, G > . . . 630
mln::Object< complex_window_bkd_piter< I, G, W > > . . . 900
mln::Proxy< complex_window_bkd_piter< I, G, W > > . . . 1004
mln::Site_Proxy< complex_window_bkd_piter< I, G, W > > . . . 1018
mln::Site_Iterator< complex_window_bkd_piter< I, G, W > > . . . 1016
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 > . . . 633
mln::Object< complex_window_fwd_piter< I, G, W > > . . . 900
mln::Proxy< complex_window_fwd_piter< I, G, W > > . . . 1004
mln::Site_Proxy< complex_window_fwd_piter< I, G, W > > . . . 1018
mln::Site_Iterator< complex_window_fwd_piter< I, G, W > > . . . 1016
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 > . . . 635
mln::Object< component< T, i > > . . . 900
mln::Function< component< T, i > > . . . 764
mln::Function_v2v< component< T, i > > . . . 767
mln::fun::v2v::component< T, i > . . . 727
mln::Object< compose > . . . 900
mln::Meta_Function< compose > . . . 873
mln::Meta_Function_vv2v< compose > . . . 875
mln::Object< composition< mln::Meta_Function_v2v, F, mln::Meta_Function_v2v, G > > . . . 900
mln::Meta_Function< composition< mln::Meta_Function_v2v, F, mln::Meta_Function_v2v, G > > . . . 873
mln::Meta_Function_v2v< composition< mln::Meta_Function_v2v, F, mln::Meta_Function_v2v, G > > . . . 874
mln::Object< composition< mln::Meta_Function_v2v, F, mln::Meta_Function_vv2v, G > > . . . 900
mln::Meta_Function< composition< mln::Meta_Function_v2v, F, mln::Meta_Function_vv2v, G > > . . . 873
mln::Meta_Function_vv2v< composition< mln::Meta_Function_v2v, F, mln::Meta_Function_vv2v, G > > . . . 875
mln::Object< concrete > . . . 900
mln::Object< convert< V > > . . . 900
mln::Function< convert< V > > . . . 764
mln::Function_v2v< convert< V > > . . . 767
mln::Object< convolve< T1, T2, R > > . . . 900
mln::Proxy< convolve< T1, T2, R > > . . . 1004
mln::Accumulator< convolve< T1, T2, R > > . . . 577
base< R, convolve< T1, T2, R > >
mln::accu::convolve< T1, T2, R > . . . 459
mln::Object< cos > . . . 900
mln::Meta_Function< cos > . . . 873
mln::Meta_Function_v2v< cos > . . . 874
mln::Object< cos< V > > . . . 900
mln::Function< cos< V > > . . . 764
mln::Function_v2v< cos< V > > . . . 767
mln::fun::v2w2v::cos< V > . . . 732
mln::Object< count > . . . 900
mln::Meta_Accumulator< count > . . . 871
mln::accu::meta::math::count . . . 502
mln::Object< count< T > > . . . 900
mln::Proxy< count< T > > . . . 1004
mln::Accumulator< count< T > > . . . 577
base< unsigned, count< T > >

mln::accu::math::count< T > . . . 481
mln::Object< count_adjacent_vertices > . . . 900
mln::Meta_Accumulator< count_adjacent_vertices > . . . 871
mln::accu::meta::count_adjacent_vertices . . . 492
mln::Object< count_adjacent_vertices< F, S > > . . . 900
mln::Proxy< count_adjacent_vertices< F, S > > . . . 1004
mln::Accumulator< count_adjacent_vertices< F, S > > . . . 577
base< unsigned, count_adjacent_vertices< F, S > >
mln::accu::count_adjacent_vertices< F, S > . . . 461
mln::Object< count_adjacent_vertices< I > > . . . 900
mln::Proxy< count_adjacent_vertices< I > > . . . 1004
mln::Accumulator< count_adjacent_vertices< I > > . . . 577
base< unsigned, count_adjacent_vertices< I > >
mln::morpho::attribute::count_adjacent_vertices< I > . . . 886
mln::Object< count_labels > . . . 900
mln::Meta_Accumulator< count_labels > . . . 871
mln::accu::meta::count_labels . . . 493
mln::Object< count_labels< L > > . . . 900
mln::Proxy< count_labels< L > > . . . 1004
mln::Accumulator< count_labels< L > > . . . 577
base< unsigned, count_labels< L > >
mln::accu::count_labels< L > . . . 463
mln::Object< count_value > . . . 900
mln::Meta_Accumulator< count_value > . . . 871
mln::accu::meta::count_value . . . 494
mln::Object< count_value< V > > . . . 900
mln::Proxy< count_value< V > > . . . 1004
mln::Accumulator< count_value< V > > . . . 577
base< unsigned, count_value< V > >
mln::accu::count_value< V > . . . 465
mln::Object< couple< T, U > > . . . 900
mln::util::couple< T, U > . . . 1112
mln::Object< cube > . . . 900
mln::Mesh< cube > . . . 870
mln::Regular_Grid< cube > . . . 1011
mln::Object< cube3d > . . . 900
mln::Window< cube3d > . . . 1237
window_base< dpoint3d, cube3d >
classical_window_base< dpoint3d, cube3d >
mln::win::cube3d . . . 1224
mln::Object< cuboid3d > . . . 900
mln::Window< cuboid3d > . . . 1237
window_base< dpoint3d, cuboid3d >
classical_window_base< dpoint3d, cuboid3d >
mln::win::cuboid3d . . . 1226
mln::Object< cyan_t > . . . 900
mln::Literal< cyan_t > . . . 845
mln::literal::cyan_t . . . 850
mln::Object< d_t > . . . 900

mln::Function< d_t > . . . 764
mln::Function_vv2v< d_t > . . . 769
mln::Object< dark_gray_t > . . . 900
mln::Literal< dark_gray_t > . . . 845
mln::Object< decorated_image< I, D > > . . . 900
mln::Image< decorated_image< I, D > > . . . 815
mln::Object< depth1st_piter< T > > . . . 900
mln::Proxy< depth1st_piter< T > > . . . 1004
mln::Site_Proxy< depth1st_piter< T > > . . . 1018
mln::Site_Iterator< depth1st_piter< T > > . . . 1016
mln::Object< depth_first_search_t > . . . 900
mln::Browsing< depth_first_search_t > . . . 600
graph_first_search_t< depth_first_search_t, std::stack >
mln::canvas::browsing::depth_first_search_t . . . 604
mln::Object< desc_propagation > . . . 900
mln::Object< deviation > . . . 900
mln::Meta_Accumulator< deviation > . . . 871
mln::accu::stat::meta::deviation . . . 554
mln::Object< deviation< T, S, M > > . . . 900
mln::Proxy< deviation< T, S, M > > . . . 1004
mln::Accumulator< deviation< T, S, M > > . . . 577
base< M, deviation< T, S, M > >
mln::accu::stat::deviation< T, S, M > . . . 542
mln::Object< diag2d > . . . 900
mln::Window< diag2d > . . . 1237
window_base< dpoint2d, diag2d >
classical_window_base< dpoint2d, diag2d >
mln::win::diag2d . . . 1228
mln::Object< diagonal2d_t > . . . 900
mln::Browsing< diagonal2d_t > . . . 600
mln::canvas::browsing::diagonal2d_t . . . 605
mln::Object< diff_abs< V > > . . . 900
mln::Function< diff_abs< V > > . . . 764
mln::Function_vv2v< diff_abs< V > > . . . 769
mln::fun::vv2v::diff_abs< V > . . . 742
mln::Object< dir_struct_elt_incr_update_t > . . . 900
mln::Browsing< dir_struct_elt_incr_update_t > . . . 600
mln::canvas::browsing::dir_struct_elt_incr_update_t . . . 607
mln::Object< directional_t > . . . 900
mln::Browsing< directional_t > . . . 600
mln::canvas::browsing::directional_t . . . 609
mln::Object< dist > . . . 900
mln::Function< dist > . . . 764
mln::Function_vv2v< dist > . . . 769
mln::Object< dist_t > . . . 900
mln::Function< dist_t > . . . 764
mln::Function_vv2v< dist_t > . . . 769
mln::Object< dn_leaf_piter< T > > . . . 900

mln::Proxy< dn_leaf_piter< T > > . . . 1004
mln::Site_Proxy< dn_leaf_piter< T > > . . . 1018
mln::Site_Iterator< dn_leaf_piter< T > > . . . 1016
mln::Object< dn_node_piter< T > > . . . 900
mln::Proxy< dn_node_piter< T > > . . . 1004
mln::Site_Proxy< dn_node_piter< T > > . . . 1018
mln::Site_Iterator< dn_node_piter< T > > . . . 1016
mln::Object< dn_site_piter< T > > . . . 900
mln::Proxy< dn_site_piter< T > > . . . 1004
mln::Site_Proxy< dn_site_piter< T > > . . . 1018
mln::Site_Iterator< dn_site_piter< T > > . . . 1016
mln::Object< dpoint< G, C > > . . . 900
mln::Gdpoint< dpoint< G, C > > . . . 776
mln::dpoint< G, C > . . . 689
mln::Object< dpoints_bkd_pixter< I > > . . . 900
mln::Iterator< dpoints_bkd_pixter< I > > . . . 838
Pixel_Iterator< dpoints_bkd_pixter< I > >
mln::Object< dpoints_fwd_pixter< I > > . . . 900
mln::Iterator< dpoints_fwd_pixter< I > > . . . 838
Pixel_Iterator< dpoints_fwd_pixter< I > >
mln::Object< dpsites_bkd_piter< V > > . . . 900
mln::Proxy< dpsites_bkd_piter< V > > . . . 1004
mln::Site_Proxy< dpsites_bkd_piter< V > > . . . 1018
mln::Site_Iterator< dpsites_bkd_piter< V > > . . . 1016
site_iterator_base< V, dpsites_bkd_piter< V > >
site_relative_iterator_base< V, dpsites_bkd_piter< V > >
mln::dpsites_bkd_piter< V > . . . 701
mln::Object< dpsites_fwd_piter< V > > . . . 900
mln::Proxy< dpsites_fwd_piter< V > > . . . 1004
mln::Site_Proxy< dpsites_fwd_piter< V > > . . . 1018
mln::Site_Iterator< dpsites_fwd_piter< V > > . . . 1016
site_iterator_base< V, dpsites_fwd_piter< V > >
site_relative_iterator_base< V, dpsites_fwd_piter< V > >
mln::dpsites_fwd_piter< V > . . . 703
mln::doc::Object< E > . . . 669
mln::doc::Image< E > . . . 659
mln::doc::Fastest_Image< E > . . . 649
mln::doc::Iterator< E > . . . 665
mln::doc::Pixel_Iterator< E > . . . 670
mln::doc::Site_Iterator< E > . . . 676
mln::doc::Value_Iterator< E > . . . 680
mln::doc::Dpoint< E > . . . 647
mln::doc::Image< E > . . . 659
mln::doc::Iterator< E > . . . 665
mln::doc::Neighborhood< E > . . . 667
mln::doc::Site_Set< E > . . . 678
mln::doc::Value_Set< E > . . . 682
mln::doc::Weighted_Window< E > . . . 684
mln::doc::Window< E > . . . 687
mln::doc::Site_Set< E > . . . 678

mln::doc::Box< E > . . . 644
mln::Object< E > . . . 900
mln::Delta_Point_Site< E > . . . 640
mln::Dpoint< E > . . . 694
mln::Function< E > . . . 764
mln::Function_v2v< E > . . . 767
mln::fun::x2v::trilinear< I > . . . 752
mln::fun::x2x::linear< I > . . . 754
mln::Function_v2b< E > . . . 766
mln::Function_v2v< E > . . . 767
mln::fun::x2v::bilinear< I > . . . 751
mln::Function_vv2b< E > . . . 768
mln::Function_vv2v< E > . . . 769
mln::Graph< E > . . . 786
mln::Image< E > . . . 815
mln::Iterator< E > . . . 838
complex_iterator_base< F, E >
complex_relative_iterator_base< C, F, E >
Pixel_Iterator< E >
mln::Mesh< E > . . . 870
mln::Regular_Grid< E > . . . 1011
mln::Meta_Function< E > . . . 873
mln::Meta_Function_v2v< E > . . . 874
mln::Meta_Function_vv2v< E > . . . 875
mln::Meta_Function_v2v< E > . . . 874
mln::Meta_Function_vv2v< E > . . . 875
mln::Browsing< E > . . . 600
mln::Delta_Point_Site< E > . . . 640
mln::Function< E > . . . 764
mln::Gdpoint< E > . . . 776
mln::Graph< E > . . . 786
mln::Image< E > . . . 815
mln::Iterator< E > . . . 838
mln::Literal< E > . . . 845
mln::Mesh< E > . . . 870
mln::Meta_Accumulator< E > . . . 871
mln::Meta_Function< E > . . . 873
mln::Neighborhood< E > . . . 898
mln::Proxy< E > . . . 1004
mln::Site< E > . . . 1014
mln::Site_Set< E > . . . 1020
mln::Value< E > . . . 1173
mln::Weighted_Window< E > . . . 1221
mln::Window< E > . . . 1237
mln::Neighborhood< E > . . . 898
mln::Proxy< E > . . . 1004
mln::Accumulator< E > . . . 577
base< R, E >
mln::Accumulator< E > . . . 577
mln::Site_Proxy< E > . . . 1018
mln::Site_Proxy< E > . . . 1018
mln::Pseudo_Site< E > . . . 1006

mln::Site_Iterator< E > . . . 1016
mln::Pseudo_Site< E > . . . 1006
mln::Site_Iterator< E > . . . 1016
site_iterator_base< S, E >
mln::internal::site_set_iterator_base
mln::Site< E > . . . 1014
mln::Gpoint< E > . . . 781
mln::Site_Set< E > . . . 1020
mln::Box< E > . . . 595
mln::Value< E > . . . 1173
mln::Weighted_Window< E > . . . 1221
mln::Window< E > . . . 1237
mln::graph_window_base< P, E > . . . 804
window_base< D, E >
mln::Object< eat > . . . 900
mln::util::eat . . . 1114
mln::Object< edge_bkd_iterator< G > > . . . 900
mln::Proxy< edge_bkd_iterator< G > > . . . 1004
mln::Object< edge_fwd_iterator< G > > . . . 900
mln::Proxy< edge_fwd_iterator< G > > . . . 1004
mln::Object< edge_image< P, V, G > > . . . 900
mln::Image< edge_image< P, V, G > > . . . 815
mln::Object< edge_nbh_edge_bkd_iterator< G > > . . . 900
mln::Proxy< edge_nbh_edge_bkd_iterator< G > > . . . 1004
mln::Object< edge_nbh_edge_fwd_iterator< G > > . . . 900
mln::Proxy< edge_nbh_edge_fwd_iterator< G > > . . . 1004
mln::Object< edge_to_color< I, V > > . . . 900
mln::Function< edge_to_color< I, V > > . . . 764
mln::Function_v2v< edge_to_color< I, V > > . . . 767
mln::Object< enc< V > > . . . 900
mln::Function< enc< V > > . . . 764
mln::Function_v2v< enc< V > > . . . 767
mln::Object< eq< L, R > > . . . 900
mln::Function< eq< L, R > > . . . 764
mln::Function_vv2b< eq< L, R > > . . . 768
mln::fun::vv2b::eq< L, R > . . . 736
mln::Object< extended< I > > . . . 900
mln::Image< extended< I > > . . . 815
mln::Object< extension_fun< I, F > > . . . 900
mln::Image< extension_fun< I, F > > . . . 815
mln::Object< extension_ima< I, J > > . . . 900
mln::Image< extension_ima< I, J > > . . . 815
mln::Object< extension_val< I > > . . . 900
mln::Image< extension_val< I > > . . . 815
mln::Object< f_16_to_8 > . . . 900
mln::Function< f_16_to_8 > . . . 764
mln::Function_v2v< f_16_to_8 > . . . 767
mln::Object< f_box1d_t > . . . 900

mln::Function< f_box1d_t > . . . 764
mln::Function_v2v< f_box1d_t > . . . 767
mln::Function_v2b< f_box1d_t > . . . 766
mln::Object< f_box2d_t > . . . 900
mln::Function< f_box2d_t > . . . 764
mln::Function_v2v< f_box2d_t > . . . 767
mln::Function_v2b< f_box2d_t > . . . 766
mln::Object< f_box3d_t > . . . 900
mln::Function< f_box3d_t > . . . 764
mln::Function_v2v< f_box3d_t > . . . 767
mln::Function_v2b< f_box3d_t > . . . 766
mln::Object< f_hsi_to_rgb_< T_rgb > > . . . 900
mln::Function< f_hsi_to_rgb_< T_rgb > > . . . 764
mln::Function_v2v< f_hsi_to_rgb_< T_rgb > > . . . 767
mln::Object< f_hsl_to_rgb_< T_rgb > > . . . 900
mln::Function< f_hsl_to_rgb_< T_rgb > > . . . 764
mln::Function_v2v< f_hsl_to_rgb_< T_rgb > > . . . 767
mln::Object< f_rgb_to_hsi_< T_hsi > > . . . 900
mln::Function< f_rgb_to_hsi_< T_hsi > > . . . 764
mln::Function_v2v< f_rgb_to_hsi_< T_hsi > > . . . 767
mln::Object< f_rgb_to_hsl_< T_hsl > > . . . 900
mln::Function< f_rgb_to_hsl_< T_hsl > > . . . 764
mln::Function_v2v< f_rgb_to_hsl_< T_hsl > > . . . 767
mln::Object< face_bkd_iter< D > > . . . 900
mln::Iterator< face_bkd_iter< D > > . . . 838
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 > . . . 1069
mln::Object< face_fwd_iter< D > > . . . 900
mln::Iterator< face_fwd_iter< D > > . . . 838
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 > . . . 1071
mln::Object< faces_psite< N, D, P > > . . . 900
mln::Proxy< faces_psite< N, D, P > > . . . 1004
mln::Site_Proxy< faces_psite< N, D, P > > . . . 1018
mln::Pseudo_Site< faces_psite< N, D, P > > . . . 1006
mln::Object< fibonacci_heap< P, T > > . . . 900
mln::util::fibonacci_heap< P, T > . . . 1119
mln::Object< flat_image< T, S > > . . . 900
mln::Image< flat_image< T, S > > . . . 815
mln::Object< float01 > . . . 900
mln::Value< float01 > . . . 1173
mln::Object< float01_f > . . . 900
mln::Value< float01_f > . . . 1173
mln::Object< fold< P, dir_0, dir_1, dir_2 > > . . . 900
mln::Function< fold< P, dir_0, dir_1, dir_2 > > . . . 764
mln::Function_v2v< fold< P, dir_0, dir_1, dir_2 > > . . . 767

mln::Object< from_accu< A > > 900
mln::Meta_Function< from_accu< A > > 873
mln::Meta_Function_v2v< from_accu< A > > 874
mln::Object< fun_image< F, I > > 900
mln::Image< fun_image< F, I > > 815
mln::Object< function< meta::blue< value::rgb< n > > > > 900
mln::Function< function< meta::blue< value::rgb< n > > > > 764
mln::Function_v2v< function< meta::blue< value::rgb< n > > > > 767
mln::Object< function< meta::first< util::couple< T, U > > > > 900
mln::Function< function< meta::first< util::couple< T, U > > > > 764
mln::Function_v2v< function< meta::first< util::couple< T, U > > > > 767
mln::Object< function< meta::green< value::rgb< n > > > > 900
mln::Function< function< meta::green< value::rgb< n > > > > 764
mln::Function_v2v< function< meta::green< value::rgb< n > > > > 767
mln::Object< function< meta::red< value::rgb< n > > > > 900
mln::Function< function< meta::red< value::rgb< n > > > > 764
mln::Function_v2v< function< meta::red< value::rgb< n > > > > 767
mln::Object< function< meta::second< util::couple< T, U > > > > 900
mln::Function< function< meta::second< util::couple< T, U > > > > 764
mln::Function_v2v< function< meta::second< util::couple< T, U > > > > 767
mln::Object< function< meta::to_enc< T > > > 900
mln::Function< function< meta::to_enc< T > > > 764
mln::Function_v2v< function< meta::to_enc< T > > > 767
mln::Object< fwd_pixter1d< I > > 900
mln::Iterator< fwd_pixter1d< I > > 838
Pixel_Iterator< fwd_pixter1d< I > >
mln::Object< fwd_pixter2d< I > > 900
mln::Iterator< fwd_pixter2d< I > > 838
Pixel_Iterator< fwd_pixter2d< I > >
mln::Object< fwd_pixter3d< I > > 900
mln::Iterator< fwd_pixter3d< I > > 838
Pixel_Iterator< fwd_pixter3d< I > >
mln::Object< fwd_t > 900
mln::Browsing< fwd_t > 600
mln::canvas::browsing::fwd_t 611
mln::Object< ge< L, R > > 900
mln::Function< ge< L, R > > 764
mln::Function_vv2b< ge< L, R > > 768
mln::fun::vv2b::ge< L, R > 737
mln::Object< graph > 900
mln::Graph< graph > 786
graph_base< graph >
mln::util::graph 1122
mln::Object< graph_elt_window< G, S > > 900
mln::Window< graph_elt_window< G, S > > 1237
mln::graph_window_base< S::fun_t::result, graph_elt_window< G, S > > 804
mln::graph_elt_window< G, S > 795
mln::Object< graph_elt_window_if< G, S, I > > 900

mln::Window< graph_elt_window_if< G, S, I > > . . . 1237
mln::graph_window_base< S::fun_t::result, graph_elt_window_if< G, S, I > > . . . 804
mln::graph_elt_window_if< G, S, I > . . . 799
mln::Object< graph_window_if_piter< S, W, I > > . . . 900
mln::Proxy< graph_window_if_piter< S, W, I > > . . . 1004
mln::Site_Proxy< graph_window_if_piter< S, W, I > > . . . 1018
mln::Site_Iterator< graph_window_if_piter< S, W, I > > . . . 1016
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 > . . . 806
mln::Object< graph_window_piter< S, W, I > > . . . 900
mln::Proxy< graph_window_piter< S, W, I > > . . . 1004
mln::Site_Proxy< graph_window_piter< S, W, I > > . . . 1018
mln::Site_Iterator< graph_window_piter< S, W, I > > . . . 1016
site_iterator_base< W, graph_window_piter< S, W, I > >
site_relative_iterator_base< W, graph_window_piter< S, W, I > >
mln::graph_window_piter< S, W, I > . . . 808
mln::Object< gray_f > . . . 900
mln::Value< gray_f > . . . 1173
mln::Object< graylevel< n > > . . . 900
mln::Value< graylevel< n > > . . . 1173
mln::Object< graylevel_f > . . . 900
mln::Value< graylevel_f > . . . 1173
mln::Object< green > . . . 900
mln::Meta_Function< green > . . . 873
mln::Meta_Function_v2v< green > . . . 874
mln::Object< green_t > . . . 900
mln::Literal< green_t > . . . 845
mln::literal::green_t . . . 851
mln::Object< gt< L, R > > . . . 900
mln::Function< gt< L, R > > . . . 764
mln::Function_vv2b< gt< L, R > > . . . 768
mln::fun::vv2b::gt< L, R > . . . 738
mln::Object< has< I > > . . . 900
mln::Function< has< I > > . . . 764
mln::Function_v2v< has< I > > . . . 767
mln::Function_v2b< has< I > > . . . 766
mln::Object< height > . . . 900
mln::Meta_Accumulator< height > . . . 871
mln::accu::meta::shape::height . . . 512
mln::Object< height< I > > . . . 900
mln::Proxy< height< I > > . . . 1004
mln::Accumulator< height< I > > . . . 577
base< unsigned, height< I > >
mln::accu::shape::height< I > . . . 535
mln::morpho::attribute::height< I > . . . 888
mln::Object< hexa > . . . 900
mln::Mesh< hexa > . . . 870
mln::Regular_Grid< hexa > . . . 1011

mln::Object< hexa< I > > . . . 900
mln::Image< hexa< I > > . . . 815
mln::Object< hexa< image2d< V > > > . . . 900
mln::Image< hexa< image2d< V > > > . . . 815
mln::Object< histo > . . . 900
mln::Meta_Accumulator< histo > . . . 871
mln::accu::meta::histo . . . 495
mln::Object< histo< V > > . . . 900
mln::Proxy< histo< V > > . . . 1004
mln::Accumulator< histo< V > > . . . 577
base< const std::vector< unsigned > &, histo< V > >
mln::accu::histo< V > . . . 467
mln::Object< hyper_directional_t > . . . 900
mln::Browsing< hyper_directional_t > . . . 600
mln::canvas::browsing::hyper_directional_t . . . 613
mln::Object< i2v::array< T > > . . . 900
mln::Function< i2v::array< T > > . . . 764
mln::Function_v2v< i2v::array< T > > . . . 767
mln::Function_v2b< i2v::array< T > > . . . 766
mln::Object< id2element< G, Elt > > . . . 900
mln::Function< id2element< G, Elt > > . . . 764
mln::Function_v2v< id2element< G, Elt > > . . . 767
mln::Object< identity_t > . . . 900
mln::Literal< identity_t > . . . 845
mln::literal::identity_t . . . 852
mln::Object< ignore > . . . 900
mln::util::ignore . . . 1131
mln::Object< image1d< T > > . . . 900
mln::Image< image1d< T > > . . . 815
mln::Object< image2d< T > > . . . 900
mln::Image< image2d< T > > . . . 815
mln::Object< image3d< T > > . . . 900
mln::Image< image3d< T > > . . . 815
mln::Object< image< F, S > > . . . 900
mln::Image< image< F, S > > . . . 815
mln::Object< image_if< I, F > > . . . 900
mln::Image< image_if< I, F > > . . . 815
mln::Object< implies< L, R > > . . . 900
mln::Function< implies< L, R > > . . . 764
mln::Function_vv2b< implies< L, R > > . . . 768
mln::fun::vv2b::implies< L, R > . . . 739
mln::Object< index_of_value< bool > > . . . 900
mln::Function< index_of_value< bool > > . . . 764
mln::Function_v2v< index_of_value< bool > > . . . 767
mln::Object< index_of_value< T > > . . . 900
mln::Function< index_of_value< T > > . . . 764
mln::Function_v2v< index_of_value< T > > . . . 767

mln::Object< inf > . . . 900
mln::Meta_Accumulator< inf > . . . 871
mln::accu::meta::math::inf . . . 503
mln::Meta_Function< inf > . . . 873
mln::Meta_Function_vv2v< inf > . . . 875
mln::Object< inf< T > > . . . 900
mln::Proxy< inf< T > > . . . 1004
mln::Accumulator< inf< T > > . . . 577
base< const T &, inf< T > >
mln::accu::math::inf< T > . . . 483
mln::Object< int_s< n > > . . . 900
mln::Object< int_u8_off_saver > . . . 900
mln::Object< int_u< n > > . . . 900
mln::Object< int_u_sat< n > > . . . 900
mln::Value< int_u_sat< n > > . . . 1173
mln::Object< interpolated< I, F > > . . . 900
mln::Image< interpolated< I, F > > . . . 815
mln::Object< iota > . . . 900
mln::Function< iota > . . . 764
mln::Function_v2v< iota > . . . 767
mln::Object< is_dot > . . . 900
mln::Function< is_dot > . . . 764
mln::Function_v2v< is_dot > . . . 767
mln::Function_v2b< is_dot > . . . 766
mln::Object< is_edge > . . . 900
mln::Function< is_edge > . . . 764
mln::Function_v2v< is_edge > . . . 767
mln::Function_v2b< is_edge > . . . 766
mln::Object< is_n_face< N > > . . . 900
mln::Function< is_n_face< N > > . . . 764
mln::Function_v2v< is_n_face< N > > . . . 767
mln::Function_v2b< is_n_face< N > > . . . 766
mln::topo::is_n_face< N > . . . 1073
mln::Object< is_pixel > . . . 900
mln::Function< is_pixel > . . . 764
mln::Function_v2v< is_pixel > . . . 767
mln::Function_v2b< is_pixel > . . . 766
mln::Object< is_row_odd > . . . 900
mln::Function< is_row_odd > . . . 764
mln::Function_v2v< is_row_odd > . . . 767
mln::Function_v2b< is_row_odd > . . . 766
mln::Object< is_separator > . . . 900
mln::Function< is_separator > . . . 764
mln::Function_v2v< is_separator > . . . 767
mln::Function_v2b< is_separator > . . . 766
mln::world::inter_pixel::is_separator . . . 1243
mln::Object< is_simple_cell< I > > . . . 900
mln::Function< is_simple_cell< I > > . . . 764
mln::Function_v2v< is_simple_cell< I > > . . . 767

mln::Function_v2b< is_simple_cell< I > > . . . 766
mln::topo::is_simple_cell< I > . . . 1074
mln::Object< ithcomp > . . . 900
mln::Meta_Function< ithcomp > . . . 873
mln::Meta_Function_vv2v< ithcomp > . . . 875
mln::Object< keep_specific_colors > . . . 900
mln::Function< keep_specific_colors > . . . 764
mln::Function_v2v< keep_specific_colors > . . . 767
mln::Function_v2b< keep_specific_colors > . . . 766
mln::Object< l1 > . . . 900
mln::Meta_Function< l1 > . . . 873
mln::Meta_Function_v2v< l1 > . . . 874
mln::Object< l1_norm< V > > . . . 900
mln::Function< l1_norm< V > > . . . 764
mln::Function_v2v< l1_norm< V > > . . . 767
mln::Object< l1_norm< V, R > > . . . 900
mln::Function< l1_norm< V, R > > . . . 764
mln::Function_v2v< l1_norm< V, R > > . . . 767
mln::fun::v2v::l1_norm< V, R > . . . 728
mln::fun::v2w_w2v::l1_norm< V, R > . . . 733
mln::Object< l2 > . . . 900
mln::Meta_Function< l2 > . . . 873
mln::Meta_Function_v2v< l2 > . . . 874
mln::Object< l2_norm< V, R > > . . . 900
mln::Function< l2_norm< V, R > > . . . 764
mln::Function_v2v< l2_norm< V, R > > . . . 767
mln::fun::v2v::l2_norm< V, R > . . . 729
mln::fun::v2w_w2v::l2_norm< V, R > . . . 734
mln::Object< label< n > > . . . 900
mln::Value< label< n > > . . . 1173
mln::Object< label_used > . . . 900
mln::Meta_Accumulator< label_used > . . . 871
mln::accu::meta::label_used . . . 496
mln::Object< label_used< L > > . . . 900
mln::Proxy< label_used< L > > . . . 1004
mln::Accumulator< label_used< L > > . . . 577
base< const fun::i2v::array< bool > &, label_used< L > >
mln::accu::label_used< L > . . . 469
mln::Object< labeled_image< I > > . . . 900
mln::Image< labeled_image< I > > . . . 815
mln::Object< land > . . . 900
mln::Meta_Accumulator< land > . . . 871
mln::accu::meta::logic::land . . . 497
mln::Proxy< land > . . . 1004
mln::Accumulator< land > . . . 577
base< bool, land >
mln::accu::logic::land . . . 471
mln::Object< land< L, R > > . . . 900

mln::Function< land< L, R > > . . . 764
mln::Function_vv2v< land< L, R > > . . . 769
mln::fun::vv2v::land< L, R > . . . 743
mln::Object< land_basic > . . . 900
mln::Meta_Accumulator< land_basic > . . . 871
mln::accu::meta::logic::land_basic . . . 498
mln::Proxy< land_basic > . . . 1004
mln::Accumulator< land_basic > . . . 577
base< bool, land_basic >
mln::accu::logic::land_basic . . . 473
mln::Object< land_not< L, R > > . . . 900
mln::Function< land_not< L, R > > . . . 764
mln::Function_vv2v< land_not< L, R > > . . . 769
mln::fun::vv2v::land_not< L, R > . . . 744
mln::Object< lazy_image< I, F, B > > . . . 900
mln::Image< lazy_image< I, F, B > > . . . 815
mln::Object< le< L, R > > . . . 900
mln::Function< le< L, R > > . . . 764
mln::Function_vv2b< le< L, R > > . . . 768
mln::fun::vv2b::le< L, R > . . . 740
mln::Object< light_gray_t > . . . 900
mln::Literal< light_gray_t > . . . 845
mln::literal::light_gray_t . . . 853
mln::Object< lime_t > . . . 900
mln::Literal< lime_t > . . . 845
mln::literal::lime_t . . . 854
mln::Object< line< M, i, C > > . . . 900
mln::Window< line< M, i, C > > . . . 1237
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 > . . . 1229
mln::Object< line_graph< G > > . . . 900
mln::Graph< line_graph< G > > . . . 786
graph_base< line_graph< G > >
mln::util::line_graph< G > . . . 1133
mln::Object< linear< V, T, R > > . . . 900
mln::Function< linear< V, T, R > > . . . 764
mln::Function_v2v< linear< V, T, R > > . . . 767
mln::fun::v2v::linear< V, T, R > . . . 730
mln::Object< linear_sat< V, T, R > > . . . 900
mln::Function< linear_sat< V, T, R > > . . . 764
mln::Function_v2v< linear_sat< V, T, R > > . . . 767
mln::Object< linfty > . . . 900
mln::Meta_Function< linfty > . . . 873
mln::Meta_Function_v2v< linfty > . . . 874
mln::Object< linfty_norm< V, R > > . . . 900
mln::Function< linfty_norm< V, R > > . . . 764
mln::Function_v2v< linfty_norm< V, R > > . . . 767

mln::fun::v2v::linfty_norm< V, R > . . . 731
mln::fun::v2w_w2v::linfty_norm< V, R > . . . 735
mln::Object< lnot< V > > . . . 900
mln::Function< lnot< V > > . . . 764
mln::Function_v2v< lnot< V > > . . . 767
mln::Function_v2b< lnot< V > > . . . 766
mln::fun::v2b::lnot< V > . . . 724
mln::Object< lor > . . . 900
mln::Meta_Accumulator< lor > . . . 871
mln::accu::meta::logic::lor . . . 499
mln::Proxy< lor > . . . 1004
mln::Accumulator< lor > . . . 577
base< bool, lor >
mln::accu::logic::lor . . . 475
mln::Object< lor< L, R > > . . . 900
mln::Function< lor< L, R > > . . . 764
mln::Function_vv2v< lor< L, R > > . . . 769
mln::fun::vv2v::lor< L, R > . . . 745
mln::Object< lor_basic > . . . 900
mln::Meta_Accumulator< lor_basic > . . . 871
mln::accu::meta::logic::lor_basic . . . 500
mln::Proxy< lor_basic > . . . 1004
mln::Accumulator< lor_basic > . . . 577
base< bool, lor_basic >
mln::accu::logic::lor_basic . . . 477
mln::Object< lt< L, R > > . . . 900
mln::Function< lt< L, R > > . . . 764
mln::Function_vv2b< lt< L, R > > . . . 768
mln::fun::vv2b::lt< L, R > . . . 741
mln::Object< lut_vec< S, T > > . . . 900
Value_Set< lut_vec< S, T > >
mln::value::lut_vec< S, T > . . . 1197
mln::Object< lxor< L, R > > . . . 900
mln::Function< lxor< L, R > > . . . 764
mln::Function_vv2v< lxor< L, R > > . . . 769
mln::fun::vv2v::lxor< L, R > . . . 746
mln::Object< magenta_t > . . . 900
mln::Literal< magenta_t > . . . 845
mln::literal::magenta_t . . . 855
mln::Object< mahalanobis< V > > . . . 900
mln::Function< mahalanobis< V > > . . . 764
mln::Function_v2v< mahalanobis< V > > . . . 767
mln::Object< maj_h > . . . 900
mln::Meta_Accumulator< maj_h > . . . 871
mln::accu::meta::maj_h . . . 501
mln::Object< maj_h< T > > . . . 900
mln::Proxy< maj_h< T > > . . . 1004
mln::Accumulator< maj_h< T > > . . . 577
base< const T &, maj_h< T > >

mln::accu::maj_h< T > . . . 479
mln::Object< mat< n, m, T > > . . . 900
mln::Object< max > . . . 900
mln::Meta_Accumulator< max > . . . 871
mln::accu::meta::stat::max . . . 514
mln::Object< max< T > > . . . 900
mln::Proxy< max< T > > . . . 1004
mln::Accumulator< max< T > > . . . 577
base< const T &, max< T > >
mln::accu::stat::max< T > . . . 544
mln::Object< max< V > > . . . 900
mln::Function< max< V > > . . . 764
mln::Function_vv2v< max< V > > . . . 769
mln::fun::vv2v::max< V > . . . 747
mln::Object< max_h > . . . 900
mln::Meta_Accumulator< max_h > . . . 871
mln::accu::meta::stat::max_h . . . 515
mln::Object< max_h< V > > . . . 900
mln::Proxy< max_h< V > > . . . 1004
mln::Accumulator< max_h< V > > . . . 577
base< const V &, max_h< V > >
mln::accu::stat::max_h< V > . . . 546
mln::Object< max_site > . . . 900
mln::Meta_Accumulator< max_site > . . . 871
mln::accu::meta::max_site . . . 506
mln::Object< max_site< I > > . . . 900
mln::Proxy< max_site< I > > . . . 1004
mln::Accumulator< max_site< I > > . . . 577
base< I::psite, max_site< I > >
mln::accu::max_site< I > . . . 489
mln::Object< max_t > . . . 900
mln::Literal< max_t > . . . 845
mln::literal::max_t . . . 856
mln::Object< mean > . . . 900
mln::Meta_Accumulator< mean > . . . 871
mln::accu::meta::stat::mean . . . 516
mln::Meta_Function< mean > . . . 873
mln::Meta_Function_v2v< mean > . . . 874
mln::Object< mean< T, S, M > > . . . 900
mln::Proxy< mean< T, S, M > > . . . 1004
mln::Accumulator< mean< T, S, M > > . . . 577
base< M, mean< T, S, M > >
mln::accu::stat::mean< T, S, M > . . . 548
mln::Object< median_alt< S > > . . . 900
mln::Proxy< median_alt< S > > . . . 1004
mln::Accumulator< median_alt< S > > . . . 577
base< const S::value &, median_alt< S > >
mln::accu::stat::median_alt< S > . . . 550
mln::Object< median_alt< T > > . . . 900

mln::Meta_Accumulator< median_alt< T > > . . . 871
mln::accu::meta::stat::median_alt< T > . . . 517
mln::Object< median_alt< value::set< T > > > . . . 900
mln::Proxy< median_alt< value::set< T > > > . . . 1004
mln::Accumulator< median_alt< value::set< T > > > . . . 577
base< const value::set< T >::value &, median_alt< value::set< T > > >
mln::accu::stat::median_alt< value::set< T > > . . . 550
mln::Object< median_h > . . . 900
mln::Meta_Accumulator< median_h > . . . 871
mln::accu::meta::stat::median_h . . . 518
mln::Object< median_h< V > > . . . 900
mln::Proxy< median_h< V > > . . . 1004
mln::Accumulator< median_h< V > > . . . 577
base< const V &, median_h< V > >
mln::accu::stat::median_h< V > . . . 552
mln::Object< medium_gray_t > . . . 900
mln::Literal< medium_gray_t > . . . 845
mln::Object< min > . . . 900
mln::Meta_Accumulator< min > . . . 871
mln::accu::meta::stat::min . . . 519
mln::Object< min< L, R > > . . . 900
mln::Function< min< L, R > > . . . 764
mln::Function_vv2v< min< L, R > > . . . 769
mln::fun::vv2v::min< L, R > . . . 748
mln::Object< min< T > > . . . 900
mln::Proxy< min< T > > . . . 1004
mln::Accumulator< min< T > > . . . 577
base< const T &, min< T > >
mln::accu::stat::min< T > . . . 555
mln::Object< min_h > . . . 900
mln::Meta_Accumulator< min_h > . . . 871
mln::accu::meta::stat::min_h . . . 520
mln::Object< min_h< V > > . . . 900
mln::Proxy< min_h< V > > . . . 1004
mln::Accumulator< min_h< V > > . . . 577
base< const V &, min_h< V > >
mln::accu::stat::min_h< V > . . . 557
mln::Object< min_t > . . . 900
mln::Literal< min_t > . . . 845
mln::literal::min_t . . . 857
mln::Object< mirror< B > > . . . 900
mln::Function< mirror< B > > . . . 764
mln::Function_v2v< mirror< B > > . . . 767
mln::Object< mln::util::array< T > > . . . 900
mln::Function< mln::util::array< T > > . . . 764
mln::Function_v2v< mln::util::array< T > > . . . 767
mln::util::array< T > . . . 1099
mln::Object< mln::util::set< T > > . . . 900

mln::util::set< T > . . . 1147
mln::Object< multi_site< P > > . . . 900
mln::Object< multiple< W, F > > . . . 900
mln::Window< multiple< W, F > > . . . 1237
window_base< W::dpsite, multiple< W, F > >
mln::win::multiple< W, F > . . . 1231
mln::Object< multiple_qiter< W, F > > . . . 900
mln::Proxy< multiple_qiter< W, F > > . . . 1004
mln::Site_Proxy< multiple_qiter< W, F > > . . . 1018
mln::Site_Iterator< multiple_qiter< W, F > > . . . 1016
mln::Object< multiple_size< n, W, F > > . . . 900
mln::Window< multiple_size< n, W, F > > . . . 1237
window_base< W::dpsite, multiple_size< n, W, F > >
mln::win::multiple_size< n, W, F > . . . 1232
mln::Object< multiple_size_qiter< n, W, F > > . . . 900
mln::Proxy< multiple_size_qiter< n, W, F > > . . . 1004
mln::Site_Proxy< multiple_size_qiter< n, W, F > > . . . 1018
mln::Site_Iterator< multiple_size_qiter< n, W, F > > . . . 1016
mln::Object< my_box2d > . . . 900
mln::Function< my_box2d > . . . 764
mln::Function_v2v< my_box2d > . . . 767
mln::Function_v2b< my_box2d > . . . 766
mln::Object< my_ext > . . . 900
mln::Function< my_ext > . . . 764
mln::Function_v2v< my_ext > . . . 767
mln::Object< my_fun< G > > . . . 900
mln::Function< my_fun< G > > . . . 764
mln::Object< my_image2d< T > > . . . 900
mln::Image< my_image2d< T > > . . . 815
mln::Object< my_values_t > . . . 900
mln::Function< my_values_t > . . . 764
mln::Function_v2v< my_values_t > . . . 767
mln::Object< myfun > . . . 900
mln::Function< myfun > . . . 764
mln::Function_vv2v< myfun > . . . 769
mln::Object< mysqrt > . . . 900
mln::Function< mysqrt > . . . 764
mln::Function_v2v< mysqrt > . . . 767
mln::Object< n_face_bkd_iter< D > > . . . 900
mln::Iterator< n_face_bkd_iter< D > > . . . 838
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 > . . . 1081
mln::Object< n_face_fwd_iter< D > > . . . 900
mln::Iterator< n_face_fwd_iter< D > > . . . 838
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 > . . . 1083

mln::Object< neighb< graph_elt_window< G, S > > > . . . 900
mln::Neighborhood< neighb< graph_elt_window< G, S > > > . . . 898
mln::Object< neighb< graph_elt_window_if< G, S, I > > > . . . 900
mln::Neighborhood< neighb< graph_elt_window_if< G, S, I > > > . . . 898
mln::Object< neighb< W > > . . . 900
mln::Neighborhood< neighb< W > > . . . 898
mln::Object< neighb_bkd_niter< W > > . . . 900
mln::Proxy< neighb_bkd_niter< W > > . . . 1004
mln::Site_Proxy< neighb_bkd_niter< W > > . . . 1018
mln::Site_Iterator< neighb_bkd_niter< W > > . . . 1016
mln::Object< neighb_fwd_niter< W > > . . . 900
mln::Proxy< neighb_fwd_niter< W > > . . . 1004
mln::Site_Proxy< neighb_fwd_niter< W > > . . . 1018
mln::Site_Iterator< neighb_fwd_niter< W > > . . . 1016
mln::Object< nil > . . . 900
mln::Meta_Accumulator< nil > . . . 871
mln::accu::meta::nil . . . 507
mln::util::nil . . . 1138
mln::Object< nil< T > > . . . 900
mln::Proxy< nil< T > > . . . 1004
mln::Accumulator< nil< T > > . . . 577
base< util::ignore, nil< T > >
mln::accu::nil< T > . . . 525
mln::Object< not_to_remove > . . . 900
mln::Function< not_to_remove > . . . 764
mln::Function_v2v< not_to_remove > . . . 767
mln::Function_v2b< not_to_remove > . . . 766
mln::Object< object_id< Tag, V > > . . . 900
mln::Value< object_id< Tag, V > > . . . 1173
mln::Object< octagon2d > . . . 900
mln::Window< octagon2d > . . . 1237
window_base< dpoint2d, octagon2d >
classical_window_base< dpoint2d, octagon2d >
mln::win::octagon2d . . . 1233
mln::Object< olive_t > . . . 900
mln::Literal< olive_t > . . . 845
mln::literal::olive_t . . . 858
mln::Object< one_t > . . . 900
mln::Literal< one_t > . . . 845
mln::literal::one_t . . . 859
mln::Object< orange_t > . . . 900
mln::Literal< orange_t > . . . 845
mln::literal::orange_t . . . 860
mln::Object< ord_pair< T > > . . . 900
mln::util::ord_pair< T > . . . 1143
mln::Object< origin_t > . . . 900
mln::Literal< origin_t > . . . 845
mln::literal::origin_t . . . 861

mln::Object< P > 900
Point_Site< P >
mln::Point< P > 995
mln::Object< p2p_image< I, F > > 900
mln::Image< p2p_image< I, F > > 815
mln::Object< p< A > > 900
mln::Proxy< p< A > > 1004
mln::Accumulator< p< A > > 577
base< const A::result &, p< A > >
mln::accu::p< A > 527
mln::Object< p< mA > > 900
mln::Meta_Accumulator< p< mA > > 871
mln::accu::meta::p< mA > 508
mln::Object< p_array< P > > 900
mln::Site_Set< p_array< P > > 1020
site_set_base_< P, p_array< P > >
mln::p_array< P > 903
mln::Object< p_centered< W > > 900
mln::Site_Set< p_centered< W > > 1020
site_set_base_< W::psite, p_centered< W > >
mln::p_centered< W > 908
mln::Object< p_centered_piter< W > > 900
mln::Proxy< p_centered_piter< W > > 1004
mln::Site_Proxy< p_centered_piter< W > > 1018
mln::Site_Iterator< p_centered_piter< W > > 1016
mln::Object< p_complex< D, G > > 900
mln::Site_Set< p_complex< D, G > > 1020
site_set_base_< complex_psite< D, G >, p_complex< D, G > >
mln::p_complex< D, G > 911
mln::Object< p_double_piter< S, I1, I2 > > 900
mln::Proxy< p_double_piter< S, I1, I2 > > 1004
mln::Site_Proxy< p_double_piter< S, I1, I2 > > 1018
mln::Site_Iterator< p_double_piter< S, I1, I2 > > 1016
mln::Object< p_double_psite< S, Sp > > 900
mln::Proxy< p_double_psite< S, Sp > > 1004
mln::Site_Proxy< p_double_psite< S, Sp > > 1018
mln::Pseudo_Site< p_double_psite< S, Sp > > 1006
mln::Object< p_edges< G, F > > 900
mln::Site_Set< p_edges< G, F > > 1020
site_set_base_< F::result, p_edges< G, F > >
mln::p_edges< G, F > 915
mln::Object< p_edges_psite< G, F > > 900
mln::Proxy< p_edges_psite< G, F > > 1004
mln::Site_Proxy< p_edges_psite< G, F > > 1018
mln::Pseudo_Site< p_edges_psite< G, F > > 1006
mln::Object< p_faces< N, D, P > > 900
mln::Site_Set< p_faces< N, D, P > > 1020
site_set_base_< faces_psite< N, D, P >, p_faces< N, D, P > >
mln::p_faces< N, D, P > 920

mln::Object< p_graph_piter< S, I > > . . . 900
mln::Proxy< p_graph_piter< S, I > > . . . 1004
mln::Site_Proxy< p_graph_piter< S, I > > . . . 1018
mln::Site_Iterator< p_graph_piter< S, I > > . . . 1016
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 > . . . 923
mln::Object< p_if< S, F > > . . . 900
mln::Site_Set< p_if< S, F > > . . . 1020
site_set_base_< S::psite, p_if< S, F > >
mln::p_if< S, F > . . . 925
mln::Object< p_image< I > > . . . 900
mln::Site_Set< p_image< I > > . . . 1020
site_set_base_< I::psite, p_image< I > >
mln::p_image< I > . . . 928
mln::Object< p_indexed_bkd_piter< S > > . . . 900
mln::Proxy< p_indexed_bkd_piter< S > > . . . 1004
mln::Site_Proxy< p_indexed_bkd_piter< S > > . . . 1018
mln::Site_Iterator< p_indexed_bkd_piter< S > > . . . 1016
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 > . . . 932
mln::Object< p_indexed_fwd_piter< S > > . . . 900
mln::Proxy< p_indexed_fwd_piter< S > > . . . 1004
mln::Site_Proxy< p_indexed_fwd_piter< S > > . . . 1018
mln::Site_Iterator< p_indexed_fwd_piter< S > > . . . 1016
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 > . . . 933
mln::Object< p_indexed_psite< S > > . . . 900
mln::Proxy< p_indexed_psite< S > > . . . 1004
mln::Site_Proxy< p_indexed_psite< S > > . . . 1018
mln::Pseudo_Site< p_indexed_psite< S > > . . . 1006
pseudo_site_base_< const S::element &, p_indexed_psite< S > >
mln::p_indexed_psite< S > . . . 934
mln::Object< p_key< K, P > > . . . 900
mln::Site_Set< p_key< K, P > > . . . 1020
site_set_base_< P, p_key< K, P > >
mln::p_key< K, P > . . . 935
mln::Object< p_line2d > . . . 900
mln::Site_Set< p_line2d > . . . 1020
site_set_base_< point2d, p_line2d >
mln::p_line2d . . . 940
mln::Object< p_mutable_array_of< S > > . . . 900
mln::Site_Set< p_mutable_array_of< S > > . . . 1020
site_set_base_< S::site, p_mutable_array_of< S > >
mln::p_mutable_array_of< S > . . . 944
mln::Object< p_n_faces_bkd_piter< D, P > > . . . 900
mln::Proxy< p_n_faces_bkd_piter< D, P > > . . . 1004
mln::Site_Proxy< p_n_faces_bkd_piter< D, P > > . . . 1018

mln::Site_Iterator< p_n_faces_bkd_piter< D, P > > 1016
site_iterator_base< p_complex< D, P >, p_n_faces_bkd_piter< D, P > >
site_set_iterator_base< p_complex< D, P >, p_n_faces_bkd_piter< D, P > >
p_complex_piter_base_< topo::n_face_bkd_iter< D >, p_complex< D, P >, P, p_-
n_faces_bkd_piter< D, P > >
mln::p_n_faces_bkd_piter< D, P > 948
mln::Object< p_n_faces_fwd_piter< D, P > > 900
mln::Proxy< p_n_faces_fwd_piter< D, P > > 1004
mln::Site_Proxy< p_n_faces_fwd_piter< D, P > > 1018
mln::Site_Iterator< p_n_faces_fwd_piter< D, P > > 1016
site_iterator_base< p_complex< D, P >, p_n_faces_fwd_piter< D, P > >
site_set_iterator_base< p_complex< D, P >, p_n_faces_fwd_piter< D, P > >
p_complex_piter_base_< topo::n_face_fwd_iter< D >, p_complex< D, P >, P, p_-
n_faces_fwd_piter< D, P > >
mln::p_n_faces_fwd_piter< D, P > 950
mln::Object< p_priority< P, Q > > 900
mln::Site_Set< p_priority< P, Q > > 1020
site_set_base_< Q::site, p_priority< P, Q > >
mln::p_priority< P, Q > 952
mln::Object< p_queue< P > > 900
mln::Site_Set< p_queue< P > > 1020
site_set_base_< P, p_queue< P > >
mln::p_queue< P > 958
mln::Object< p_queue_fast< P > > 900
mln::Site_Set< p_queue_fast< P > > 1020
site_set_base_< P, p_queue_fast< P > >
mln::p_queue_fast< P > 962
mln::Object< p_run< P > > 900
mln::Site_Set< p_run< P > > 1020
site_set_base_< P, p_run< P > >
mln::p_run< P > 967
mln::Object< p_run_psite< P > > 900
mln::Proxy< p_run_psite< P > > 1004
mln::Site_Proxy< p_run_psite< P > > 1018
mln::Pseudo_Site< p_run_psite< P > > 1006
mln::Object< p_set< P > > 900
mln::Site_Set< p_set< P > > 1020
site_set_base_< P, p_set< P > >
mln::p_set< P > 972
mln::Object< p_set_of< S > > 900
mln::Site_Set< p_set_of< S > > 1020
mln::Object< p_transformed< S, F > > 900
mln::Site_Set< p_transformed< S, F > > 1020
site_set_base_< S::psite, p_transformed< S, F > >
mln::p_transformed< S, F > 976
mln::Object< p_transformed_piter< Pi, S, F > > 900
mln::Proxy< p_transformed_piter< Pi, S, F > > 1004
mln::Site_Proxy< p_transformed_piter< Pi, S, F > > 1018
mln::Site_Iterator< p_transformed_piter< Pi, S, F > > 1016
site_iterator_base< p_transformed< S, F >, p_transformed_piter< Pi, S, F > >

site_set_iterator_base< p_transformed< S, F >,p_transformed_piter< Pi, S, F > >
mln::p_transformed_piter< Pi, S, F > . . . 979
mln::Object< p_vaccess< V, S > > . . . 900
mln::Site_Set< p_vaccess< V, S > > . . . 1020
site_set_base_< S::site, p_vaccess< V, S > >
mln::p_vaccess< V, S > . . . 981
mln::Object< p_vertices< G, F > > . . . 900
mln::Site_Set< p_vertices< G, F > > . . . 1020
site_set_base_< F::result, p_vertices< G, F > >
mln::p_vertices< G, F > . . . 985
mln::Object< p_vertices_psite< G, F > > . . . 900
mln::Proxy< p_vertices_psite< G, F > > . . . 1004
mln::Site_Proxy< p_vertices_psite< G, F > > . . . 1018
mln::Pseudo_Site< p_vertices_psite< G, F > > . . . 1006
mln::Object< pair< A1, A2 > > . . . 900
mln::Meta_Accumulator< pair< A1, A2 > > . . . 871
mln::accu::meta::pair< A1, A2 > . . . 509
mln::Object< pair< A1, A2, T > > . . . 900
mln::Proxy< pair< A1, A2, T > > . . . 1004
mln::Accumulator< pair< A1, A2, T > > . . . 577
base< std::pair< A1::result, A2::result >, pair< A1, A2, T > >
mln::accu::pair< A1, A2, T > . . . 529
mln::Object< pair< min< V >, max< V >, mln_argument(min< V >) > > . . . 900
mln::Proxy< pair< min< V >, max< V >, mln_argument(min< V >) > > . . . 1004
mln::Accumulator< pair< min< V >, max< V >, mln_argument(min< V >) > > . . . 577
base< std::pair< min< V >::result, max< V >::result >, pair< min< V >,max< V >,mln_argument(min< V >) > >
mln::accu::pair< min< V >, max< V > > . . . 529
mln::accu::stat::min_max< V > . . . 559
mln::Object< pink_t > . . . 900
mln::Literal< pink_t > . . . 845
mln::literal::pink_t . . . 862
mln::Object< pixel< I > > . . . 900
mln::pixel< I > . . . 991
mln::Object< plain< I > > . . . 900
mln::Image< plain< I > > . . . 815
mln::Object< point< G, C > > . . . 900
mln::Site< point< G, C > > . . . 1014
mln::Gpoint< point< G, C > > . . . 781
mln::point< G, C > . . . 998
mln::Object< point_from_value< T > > . . . 900
mln::Function< point_from_value< T > > . . . 764
mln::Function_v2v< point_from_value< T > > . . . 767
mln::Object< projection< P, dir > > . . . 900
mln::Function< projection< P, dir > > . . . 764
mln::Function_v2v< projection< P, dir > > . . . 767
mln::Object< proxy< const I > > . . . 900
mln::Value< proxy< const I > > . . . 1173

mln::value::proxy< const I > . . . 1203
mln::Object< proxy< I > > . . . 900
mln::Value< proxy< I > > . . . 1173
mln::value::proxy< I > . . . 1200
mln::Object< purple_t > . . . 900
mln::Literal< purple_t > . . . 845
mln::literal::purple_t . . . 863
mln::Object< qrde > . . . 900
mln::Function< qrde > . . . 764
mln::Function_v2v< qrde > . . . 767
mln::Object< quat > . . . 900
mln::Value< quat > . . . 1173
mln::Object< rank > . . . 900
mln::Meta_Accumulator< rank > . . . 871
mln::accu::meta::stat::rank . . . 521
mln::Object< rank< bool > > . . . 900
mln::Proxy< rank< bool > > . . . 1004
mln::Accumulator< rank< bool > > . . . 577
base< bool, rank< bool > >
mln::accu::stat::rank< bool > . . . 563
mln::Object< rank< T > > . . . 900
mln::Proxy< rank< T > > . . . 1004
mln::Accumulator< rank< T > > . . . 577
base< const T &, rank< T > >
mln::accu::stat::rank< T > . . . 561
mln::Object< rank_high_quant > . . . 900
mln::Meta_Accumulator< rank_high_quant > . . . 871
mln::accu::meta::stat::rank_high_quant . . . 522
mln::Object< rank_high_quant< T > > . . . 900
mln::Proxy< rank_high_quant< T > > . . . 1004
mln::Accumulator< rank_high_quant< T > > . . . 577
base< const T &, rank_high_quant< T > >
mln::accu::stat::rank_high_quant< T > . . . 565
mln::Object< rectangle2d > . . . 900
mln::Window< rectangle2d > . . . 1237
window_base< dpoint2d, rectangle2d >
classical_window_base< dpoint2d, rectangle2d >
mln::win::rectangle2d . . . 1235
mln::Object< rectangularity< P > > . . . 900
mln::Proxy< rectangularity< P > > . . . 1004
mln::Accumulator< rectangularity< P > > . . . 577
base< float,rectangularity< P > >
couple< accu::shape::bbox< P >, accu::math::count< P >, float, rectangularity< P > >
mln::accu::site_set::rectangularity< P > . . . 540
mln::Object< red > . . . 900
mln::Meta_Function< red > . . . 873
mln::Meta_Function_v2v< red > . . . 874
mln::Object< red_t > . . . 900

mln::Literal< red_t > . . . 845
mln::literal::red_t . . . 864
mln::Object< ref_data > . . . 900
mln::Function< ref_data > . . . 764
mln::Function_v2v< ref_data > . . . 767
mln::Object< rgb8_off_loader > . . . 900
mln::Object< rgb8_off_saver > . . . 900
mln::Object< rgb< n > > . . . 900
mln::Value< rgb< n > > . . . 1173
mln::Object< rms > . . . 900
mln::Meta_Accumulator< rms > . . . 871
mln::accu::meta::rms . . . 510
mln::Object< rms< T, V > > . . . 900
mln::Proxy< rms< T, V > > . . . 1004
mln::Accumulator< rms< T, V > > . . . 577
base< V, rms< T, V > >
mln::accu::rms< T, V > . . . 531
mln::Object< rotation< n, C > > . . . 900
mln::Function< rotation< n, C > > . . . 764
mln::Function_v2v< rotation< n, C > > . . . 767
mln::fun::x2x::rotation< n, C > . . . 756
mln::Object< round< R > > . . . 900
mln::Function< round< R > > . . . 764
mln::Function_v2v< round< R > > . . . 767
mln::Object< row > . . . 900
mln::Meta_Function< row > . . . 873
mln::Meta_Function_v2v< row > . . . 874
mln::Object< safe_image< I > > . . . 900
mln::Image< safe_image< I > > . . . 815
mln::Object< saturate< V > > . . . 900
mln::Function< saturate< V > > . . . 764
mln::Function_v2v< saturate< V > > . . . 767
mln::Object< saturate_rgb8 > . . . 900
mln::Function< saturate_rgb8 > . . . 764
mln::Function_v2v< saturate_rgb8 > . . . 767
mln::Object< scomp< ith > > . . . 900
mln::Meta_Function< scomp< ith > > . . . 873
mln::Meta_Function_v2v< scomp< ith > > . . . 874
mln::Object< set_bkd_iter< T > > . . . 900
mln::Proxy< set_bkd_iter< T > > . . . 1004
mln::Object< set_fwd_iter< T > > . . . 900
mln::Proxy< set_fwd_iter< T > > . . . 1004
mln::Object< sharpness< I > > . . . 900
mln::Proxy< sharpness< I > > . . . 1004
mln::Accumulator< sharpness< I > > . . . 577
base< double, sharpness< I > >
mln::morpho::attribute::sharpness< I > . . . 890
mln::Object< shell< F, I > > . . . 900

mln::Proxy< shell< F, I > > . . . 1004
mln::Object< sign > . . . 900
mln::Value< sign > . . . 1173
mln::Object< site_pair< P > > . . . 900
mln::util::site_pair< P > . . . 1153
mln::Object< sli > . . . 900
mln::Meta_Function< sli > . . . 873
mln::Meta_Function_v2v< sli > . . . 874
mln::Object< slice_image< I > > . . . 900
mln::Image< slice_image< I > > . . . 815
mln::Object< snake_fwd_t > . . . 900
mln::Browsing< snake_fwd_t > . . . 600
mln::canvas::browsing::snake_fwd_t . . . 615
mln::Object< snake_generic_t > . . . 900
mln::Browsing< snake_generic_t > . . . 600
mln::canvas::browsing::snake_generic_t . . . 617
mln::Object< snake_vert_t > . . . 900
mln::Browsing< snake_vert_t > . . . 600
mln::canvas::browsing::snake_vert_t . . . 619
mln::Object< soft_heap< T, R > > . . . 900
mln::util::soft_heap< T, R > . . . 1155
mln::Object< sqrt > . . . 900
mln::Function< sqrt > . . . 764
mln::Function_v2v< sqrt > . . . 767
mln::Object< square > . . . 900
mln::Mesh< square > . . . 870
mln::Regular_Grid< square > . . . 1011
mln::Object< stack_image< n, I > > . . . 900
mln::Image< stack_image< n, I > > . . . 815
mln::Object< static_n_face_bkd_iter< N, D > > . . . 900
mln::Iterator< static_n_face_bkd_iter< N, D > > . . . 838
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 > . . . 1087
mln::Object< static_n_face_fwd_iter< N, D > > . . . 900
mln::Iterator< static_n_face_fwd_iter< N, D > > . . . 838
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 > . . . 1089
mln::Object< sub_image< I, S > > . . . 900
mln::Image< sub_image< I, S > > . . . 815
mln::Object< sub_image_if< I, S > > . . . 900
mln::Image< sub_image_if< I, S > > . . . 815
mln::Object< sum > . . . 900
mln::Meta_Accumulator< sum > . . . 871
mln::accu::meta::math::sum . . . 504
mln::Object< sum< I, S > > . . . 900
mln::Proxy< sum< I, S > > . . . 1004

mln::Accumulator< sum< I, S > > . . . 577
base< S, sum< I, S > >
mln::morpho::attribute::sum< I, S > . . . 892
mln::Object< sum< T, S > > . . . 900
mln::Proxy< sum< T, S > > . . . 1004
mln::Accumulator< sum< T, S > > . . . 577
base< const S &, sum< T, S > >
mln::accu::math::sum< T, S > . . . 485
mln::Object< sup > . . . 900
mln::Meta_Accumulator< sup > . . . 871
mln::accu::meta::math::sup . . . 505
mln::Meta_Function< sup > . . . 873
mln::Meta_Function_vv2v< sup > . . . 875
mln::Object< sup< T > > . . . 900
mln::Proxy< sup< T > > . . . 1004
mln::Accumulator< sup< T > > . . . 577
base< const T &, sup< T > >
mln::accu::math::sup< T > . . . 487
mln::Object< tautology > . . . 900
mln::Function< tautology > . . . 764
mln::Function_v2v< tautology > . . . 767
mln::Function_v2b< tautology > . . . 766
mln::fun::p2b::tautology . . . 723
mln::Object< teal_t > . . . 900
mln::Literal< teal_t > . . . 845
mln::literal::teal_t . . . 865
mln::Object< test > . . . 900
mln::Function< test > . . . 764
mln::Function_v2v< test > . . . 767
mln::Object< threshold< V > > . . . 900
mln::Function< threshold< V > > . . . 764
mln::Function_v2v< threshold< V > > . . . 767
mln::Function_v2b< threshold< V > > . . . 766
mln::fun::v2b::threshold< V > . . . 725
mln::Object< thru_image< I, F > > . . . 900
mln::Image< thru_image< I, F > > . . . 815
mln::Object< thrubin_image< I1, I2, F > > . . . 900
mln::Image< thrubin_image< I1, I2, F > > . . . 815
mln::Object< tick > . . . 900
mln::Mesh< tick > . . . 870
mln::Regular_Grid< tick > . . . 1011
mln::Object< timer > . . . 900
mln::Proxy< timer > . . . 1004
mln::util::timer . . . 1158
mln::Object< to16bits > . . . 900
mln::Function< to16bits > . . . 764
mln::Function_v2v< to16bits > . . . 767
mln::Object< to19bits > . . . 900

mln::Function< to19bits > . . . 764
mln::Function_v2v< to19bits > . . . 767
mln::Object< to23bits > . . . 900
mln::Function< to23bits > . . . 764
mln::Function_v2v< to23bits > . . . 767
mln::Object< to27bits > . . . 900
mln::Function< to27bits > . . . 764
mln::Function_v2v< to27bits > . . . 767
mln::Object< to8bits > . . . 900
mln::Function< to8bits > . . . 764
mln::Function_v2v< to8bits > . . . 767
mln::Object< tofloat01 > . . . 900
mln::Function< tofloat01 > . . . 764
mln::Function_v2v< tofloat01 > . . . 767
mln::Object< tr_image< S, I, T > > . . . 900
mln::Image< tr_image< S, I, T > > . . . 815
mln::Object< transformed_image< I, F > > . . . 900
mln::Image< transformed_image< I, F > > . . . 815
mln::Object< translation< n, C > > . . . 900
mln::Function< translation< n, C > > . . . 764
mln::Function_v2v< translation< n, C > > . . . 767
mln::fun::x2x::translation< n, C > . . . 759
mln::Object< translation_t< P > > . . . 900
mln::Function< translation_t< P > > . . . 764
mln::Function_v2v< translation_t< P > > . . . 767
mln::Object< tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T)> > . . . 900
mln::Proxy< tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T)> > . . . 1004
mln::Accumulator< tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T)> > . . . 577
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)> >
mln::accu::tuple< A, n, > . . . 573
mln::Object< tuple< n, BOOST_PP_ENUM_PARAMS(10, T)> > . . . 900
mln::Meta_Accumulator< tuple< n, BOOST_PP_ENUM_PARAMS(10, T)> > . . . 871
mln::accu::meta::tuple< n, > . . . 523
mln::Object< unary< Fun, T > > . . . 900
mln::Function< unary< Fun, T > > . . . 764
mln::Function_v2v< unary< Fun, T > > . . . 767
mln::Object< unproject_image< I, D, F > > . . . 900
mln::Image< unproject_image< I, D, F > > . . . 815
mln::Object< up_leaf_piter< T > > . . . 900
mln::Proxy< up_leaf_piter< T > > . . . 1004
mln::Site_Proxy< up_leaf_piter< T > > . . . 1018
mln::Site_Iterator< up_leaf_piter< T > > . . . 1016
mln::Object< up_node_piter< T > > . . . 900
mln::Proxy< up_node_piter< T > > . . . 1004
mln::Site_Proxy< up_node_piter< T > > . . . 1018
mln::Site_Iterator< up_node_piter< T > > . . . 1016
mln::Object< up_site_piter< T > > . . . 900

mln::Proxy< up_site_piter< T > > . . . 1004
mln::Site_Proxy< up_site_piter< T > > . . . 1018
mln::Site_Iterator< up_site_piter< T > > . . . 1016
mln::Object< val< A > > . . . 900
mln::Proxy< val< A > > . . . 1004
mln::Accumulator< val< A > > . . . 577
base< const A::result &, val< A > >
mln::accu::val< A > . . . 575
mln::Object< val< mA > > . . . 900
mln::Meta_Accumulator< val< mA > > . . . 871
mln::accu::meta::val< mA > . . . 524
mln::Object< value_at_index< bool > > . . . 900
mln::Function< value_at_index< bool > > . . . 764
mln::Function_v2v< value_at_index< bool > > . . . 767
mln::Object< value_at_index< T > > . . . 900
mln::Function< value_at_index< T > > . . . 764
mln::Function_v2v< value_at_index< T > > . . . 767
mln::Object< var< T > > . . . 900
mln::Proxy< var< T > > . . . 1004
mln::Accumulator< var< T > > . . . 577
base< algebra::mat< T::dim, T::dim, float >, var< T > >
mln::accu::stat::var< T > . . . 567
mln::Object< variance< T, S, R > > . . . 900
mln::Proxy< variance< T, S, R > > . . . 1004
mln::Accumulator< variance< T, S, R > > . . . 577
base< R, variance< T, S, R > >
mln::accu::stat::variance< T, S, R > . . . 570
mln::Object< vec< 1, T > > . . . 900
mln::Object< vec< 2, T > > . . . 900
mln::Object< vec< 3, T > > . . . 900
mln::Object< vec< 4, T > > . . . 900
mln::Object< vec< n, C > > . . . 900
mln::Object< vec< n, T > > . . . 900
mln::Object< vec< V > > . . . 900
mln::Function< vec< V > > . . . 764
mln::Function_vv2v< vec< V > > . . . 769
mln::fun::vv2v::vec< V > . . . 749
mln::Object< vertex< G > > . . . 900
mln::Site< vertex< G > > . . . 1014
mln::util::vertex< G > . . . 1167
mln::Object< vertex_bkd_iterator< G > > . . . 900
mln::Object< vertex_fwd_iterator< G > > . . . 900
mln::Object< vertex_image< P, V, G > > . . . 900
mln::Image< vertex_image< P, V, G > > . . . 815
mln::Object< vertex_nbh_edge_bkd_iterator< G > > . . . 900
mln::Proxy< vertex_nbh_edge_bkd_iterator< G > > . . . 1004
mln::Object< vertex_nbh_edge_fwd_iterator< G > > . . . 900
mln::Proxy< vertex_nbh_edge_fwd_iterator< G > > . . . 1004
mln::Object< vertex_nbh_vertex_bkd_iterator< G > > . . . 900

mln::Object< vertex_nbh_vertex_fwd_iterator< G > > . . . 900
mln::Proxy< vertex_nbh_vertex_fwd_iterator< G > > . . . 1004
mln::Object< violent_cast_image< T, I > > . . . 900
mln::Image< violent_cast_image< T, I > > . . . 815
mln::Object< violet_t > . . . 900
mln::Literal< violet_t > . . . 845
mln::literal::violet_t . . . 866
mln::Object< viota_t > . . . 900
mln::Function< viota_t > . . . 764
mln::Function_v2v< viota_t > . . . 767
mln::Object< viota_t< S > > . . . 900
mln::Function< viota_t< S > > . . . 764
mln::Function_v2v< viota_t< S > > . . . 767
mln::Object< volume > . . . 900
mln::Meta_Accumulator< volume > . . . 871
mln::accu::meta::shape::volume . . . 513
mln::Object< volume< I > > . . . 900
mln::Proxy< volume< I > > . . . 1004
mln::Accumulator< volume< I > > . . . 577
base< unsigned, volume< I > >
mln::accu::shape::volume< I > . . . 537
mln::morpho::attribute::volume< I > . . . 894
mln::Object< W > . . . 900
mln::Object< w_window< D, W > > . . . 900
mln::Weighted_Window< w_window< D, W > > . . . 1221
weighted_window_base< mln::window< D >, w_window< D, W > >
mln::w_window< D, W > . . . 1217
mln::Object< white_t > . . . 900
mln::Literal< white_t > . . . 845
mln::literal::white_t . . . 867
mln::Object< window< D > > . . . 900
mln::Window< window< D > > . . . 1237
window_base< D, window< D > >
mln::window< D > . . . 1238
mln::Object< wrap > . . . 900
mln::Function< wrap > . . . 764
mln::Function_v2v< wrap > . . . 767
mln::Object< wrap< L > > . . . 900
mln::Function< wrap< L > > . . . 764
mln::Function_v2v< wrap< L > > . . . 767
mln::Object< yellow_t > . . . 900
mln::Literal< yellow_t > . . . 845
mln::literal::yellow_t . . . 868
mln::Object< yes > . . . 900
mln::util::yes . . . 1172
mln::Object< zero_t > . . . 900
mln::Literal< zero_t > . . . 845
mln::literal::zero_t . . . 869

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 > 695
mln::internal::pixel_impl_< I, dpoints_fwd_pixter< I > >
mln::dpoints_fwd_pixter< I > 698
mln::internal::pixel_impl_< I, E >
pixel_iterator_base_< I, E >
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 > 991
mln::value::internal::value_like_< algebra::vec< 3, int_u< n > >,algebra::vec< 3, int_u< n > >,algebra::vec< 3, int >,rgb< n > >
mln::value::rgb< n > 1206
mln::value::internal::value_like_< float,float,float,float01_f >
mln::value::float01_f 1177
mln::value::internal::value_like_< float01_f,float01_f::enc, internal::gray_f,graylevel_f >
mln::value::graylevel_f 1182
mln::value::internal::value_like_< int,internal::encoding_signed_< n >::ret,int,int_s< n > >
mln::value::int_s< n > 1185
mln::value::internal::value_like_< int_u< n >,int_u< n >::enc,internal::gray_< n >, graylevel< n > >
mln::value::graylevel< n > 1179
mln::value::internal::value_like_< int_u< n >,int_u< n >::enc,unsigned,int_u_sat< n > >
mln::value::int_u_sat< n > 1189
mln::value::internal::value_like_< unsigned,internal::encoding_unsigned_< n >::ret,int,int_u< n > >
mln::value::int_u< n > 1187
mln::value::internal::value_like_< unsigned,internal::encoding_unsigned_< n >::ret,int,label< n > >
mln::value::label< n > 1194
mln::util::internal::vertex_impl_< G >
mln::util::vertex< G > 1167

Chapter 5

Class Index

5.1 Class List

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

mln::accu::center< P, V > (Mass center accumulator) . . . 457
mln::accu::convolve< T1, T2, R > (Generic convolution accumulator class) . . . 459
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)) . . . 461
mln::accu::count_labels< L > (Count the number of different labels in an image) . . . 463
mln::accu::count_value< V > (Count a given value) . . . 465
mln::accu::histo< V > (Generic histogram class over a value set with type V) . . . 467
mln::accu::label_used< L > (References all the labels used) . . . 469
mln::accu::logic::land ("Logical-and" accumulator) . . . 471
mln::accu::logic::land_basic ("Logical-and" accumulator) . . . 473
mln::accu::logic::lor ("Logical-or" accumulator) . . . 475
mln::accu::logic::lor_basic ("Logical-or" accumulator class) . . . 477
mln::accu::maj_h< T > (Compute the majority value) . . . 479
mln::accu::math::count< T > (Generic counter accumulator) . . . 481
mln::accu::math::inf< T > (Generic inf accumulator class) . . . 483
mln::accu::math::sum< T, S > (Generic sum accumulator class) . . . 485
mln::accu::math::sup< T > (Generic sup accumulator class) . . . 487
mln::accu::max_site< I > (Define an accumulator that computes the first site with the maximum value in an image) . . . 489
mln::accu::meta::center (Meta accumulator for center) . . . 491
mln::accu::meta::count_adjacent_vertices (Meta accumulator for count_adjacent_vertices) . . . 492
mln::accu::meta::count_labels (Meta accumulator for count_labels) . . . 493
mln::accu::meta::count_value (FIXME: How to write a meta accumulator with a constructor taking a generic argument? Meta accumulator for count_value) . . . 494
mln::accu::meta::histo (Meta accumulator for histo) . . . 495
mln::accu::meta::label_used (Meta accumulator for label_used) . . . 496
mln::accu::meta::logic::land (Meta accumulator for land) . . . 497
mln::accu::meta::logic::land_basic (Meta accumulator for land_basic) . . . 498
mln::accu::meta::logic::lor (Meta accumulator for lor) . . . 499
mln::accu::meta::logic::lor_basic (Meta accumulator for lor_basic) . . . 500
mln::accu::meta::maj_h (Meta accumulator for maj_h) . . . 501
mln::accu::meta::math::count (Meta accumulator for count) . . . 502
mln::accu::meta::math::inf (Meta accumulator for inf) . . . 503

mln::accu::meta::math::sum (Meta accumulator for sum) . . . 504
mln::accu::meta::math::sup (Meta accumulator for sup) . . . 505
mln::accu::meta::max_site (Meta accumulator for max_site) . . . 506
mln::accu::meta::nil (Meta accumulator for nil) . . . 507
mln::accu::meta::p< mA > (Meta accumulator for p) . . . 508
mln::accu::meta::pair< A1, A2 > (Meta accumulator for pair) . . . 509
mln::accu::meta::rms (Meta accumulator for rms) . . . 510
mln::accu::meta::shape::bbox (Meta accumulator for bbox) . . . 511
mln::accu::meta::shape::height (Meta accumulator for height) . . . 512
mln::accu::meta::shape::volume (Meta accumulator for volume) . . . 513
mln::accu::meta::stat::max (Meta accumulator for max) . . . 514
mln::accu::meta::stat::max_h (Meta accumulator for max) . . . 515
mln::accu::meta::stat::mean (Meta accumulator for mean) . . . 516
mln::accu::meta::stat::median_alt< T > (Meta accumulator for median_alt) . . . 517
mln::accu::meta::stat::median_h (Meta accumulator for median_h) . . . 518
mln::accu::meta::stat::min (Meta accumulator for min) . . . 519
mln::accu::meta::stat::min_h (Meta accumulator for min) . . . 520
mln::accu::meta::stat::rank (Meta accumulator for rank) . . . 521
mln::accu::meta::stat::rank_high_quant (Meta accumulator for rank_high_quant) . . . 522
mln::accu::meta::tuple< n, > (Meta accumulator for tuple) . . . 523
mln::accu::meta::val< mA > (Meta accumulator for val) . . . 524
mln::accu::nil< T > (Define an accumulator that does nothing) . . . 525
mln::accu::p< A > (Generic p of accumulators) . . . 527
mln::accu::pair< A1, A2, T > (Generic pair of accumulators) . . . 529
mln::accu::rms< T, V > (Generic root mean square accumulator class) . . . 531
mln::accu::shape::bbox< P > (Generic bounding box accumulator class) . . . 533
mln::accu::shape::height< I > (Height accumulator) . . . 535
mln::accu::shape::volume< I > (Volume accumulator class) . . . 537
mln::accu::site_set::rectangularity< P > (Compute the rectangularity of a site set) . . . 540
mln::accu::stat::deviation< T, S, M > (Generic standard deviation accumulator class) . . . 542
mln::accu::stat::max< T > (Generic max accumulator class) . . . 544
mln::accu::stat::max_h< V > (Generic max function based on histogram over a value set with type V) . . . 546
mln::accu::stat::mean< T, S, M > (Generic mean accumulator class) . . . 548
mln::accu::stat::median_alt< S > (Generic median_alt function based on histogram over a value set with type S) . . . 550
mln::accu::stat::median_h< V > (Generic median function based on histogram over a value set with type V) . . . 552
mln::accu::stat::meta::deviation (Meta accumulator for deviation) . . . 554
mln::accu::stat::min< T > (Generic min accumulator class) . . . 555
mln::accu::stat::min_h< V > (Generic min function based on histogram over a value set with type V) . . . 557
mln::accu::stat::min_max< V > (Generic min and max accumulator class) . . . 559
mln::accu::stat::rank< T > (Generic rank accumulator class) . . . 561
mln::accu::stat::rank< bool > (Rank accumulator class for Boolean) . . . 563
mln::accu::stat::rank_high_quant< T > (Generic rank accumulator class) . . . 565
mln::accu::stat::var< T > (Var accumulator class) . . . 567
mln::accu::stat::variance< T, S, R > (Variance accumulator class) . . . 570
mln::accu::tuple< A, n, > (Generic tuple of accumulators) . . . 573
mln::accu::val< A > (Generic val of accumulators) . . . 575
mln::Accumulator< E > (Base class for implementation of accumulators) . . . 577
mln::algebra::h_mat< d, T > (N-Dimensional matrix with homogeneous coordinates) . . . 578
mln::algebra::h_vec< d, C > (N-Dimensional vector with homogeneous coordinates) . . . 580
mln::bkd_pixter1d< I > (Backward pixel iterator on a 1-D image with border) . . . 582

mln::bkd_pixter2d< I > (Backward pixel iterator on a 2-D image with border) . . . 584
mln::bkd_pixter3d< I > (Backward pixel iterator on a 3-D image with border) . . . 586
mln::box< P > (Generic box class: site set containing points of a regular grid) . . . 588
mln::Box< E > (Base class for implementation classes of boxes) . . . 595
mln::box_runstart_piter< P > (A generic forward iterator on points by lines) . . . 598
mln::Browsing< E > (Base class for implementation classes that are browsings) . . . 600
mln::canvas::browsing::backdiagonal2d_t (Browsing in a certain direction) . . . 601
mln::canvas::browsing::breadth_first_search_t (Breadth-first search algorithm for graph, on vertices) . . . 603
mln::canvas::browsing::depth_first_search_t (Breadth-first search algorithm for graph, on vertices) . . . 604
mln::canvas::browsing::diagonal2d_t (Browsing in a certain direction) . . . 605
mln::canvas::browsing::dir_struct_elt_incr_update_t (Browsing in a certain direction with a segment) . . . 607
mln::canvas::browsing::directional_t (Browsing in a certain direction) . . . 609
mln::canvas::browsing::fwd_t (Canvas for forward browsing) . . . 611
mln::canvas::browsing::hyper_directional_t (Browsing in a certain direction) . . . 613
mln::canvas::browsing::snake_fwd_t (Browsing in a snake-way, forward) . . . 615
mln::canvas::browsing::snake_generic_t (Multidimentional Browsing in a given-way) . . . 617
mln::canvas::browsing::snake_vert_t (Browsing in a snake-way, forward) . . . 619
mln::canvas::chamfer< F > (Compute chamfer distance) . . . 621
mln::category< R(*)(A) > (Category declaration for a unary C function) . . . 622
mln::complex_image< D, G, V > (Image based on a complex) . . . 623
mln::complex_neighborhood_bkd_piter< I, G, N > (Backward iterator on complex neighborhood) . . . 626
mln::complex_neighborhood_fwd_piter< I, G, N > (Forward iterator on complex neighborhood) 628
mln::complex_psite< D, G > (Point site associated to a mln::p_complex) . . . 630
mln::complex_window_bkd_piter< I, G, W > (Backward iterator on complex window) . . . 633
mln::complex_window_fwd_piter< I, G, W > (Forward iterator on complex window) . . . 635
mln::decorated_image< I, D > (Image that can have additional features) . . . 637
mln::Delta_Point_Site< E > (FIXME: Doc!) . . . 640
mln::Delta_Point_Site< void > (Delta point site category flag type) . . . 641
mln::doc::Accumulator< E > (Documentation class for mln::Accumulator) . . . 642
mln::doc::Box< E > (Documentation class for mln::Box) . . . 644
mln::doc::Dpoint< E > (Documentation class for mln::Dpoint) . . . 647
mln::doc::Fastest_Image< E > (Documentation class for the concept of images that have the speed property set to "fastest") . . . 649
mln::doc::Generalized_Pixel< E > (Documentation class for mln::Generalized_Pixel) . . . 657
mln::doc::Image< E > (Documentation class for mln::Image) . . . 659
mln::doc::Iterator< E > (Documentation class for mln::Iterator) . . . 665
mln::doc::Neighborhood< E > (Documentation class for mln::Neighborhood) . . . 667
mln::doc::Object< E > (Documentation class for mln::Object) . . . 669
mln::doc::Pixel_Iterator< E > (Documentation class for mln::Iterator) . . . 670
mln::doc::Point_Site< E > (Documentation class for mln::Point_Site) . . . 673
mln::doc::Site_Iterator< E > (Documentation class for mln::Site_Iterator) . . . 676
mln::doc::Site_Set< E > (Documentation class for mln::Site_Set) . . . 678
mln::doc::Value_Iterator< E > (Documentation class for mln::Value_Iterator) . . . 680
mln::doc::Value_Set< E > (Documentation class for mln::Value_Set) . . . 682
mln::doc::Weighted_Window< E > (Documentation class for mln::Weighted_Window) . . . 684
mln::doc::Window< E > (Documentation class for mln::Window) . . . 687
mln::dpoint< G, C > (Generic delta-point class) . . . 689
mln::Dpoint< E > (Base class for implementation of delta-point classes) . . . 694
mln::dpoints_bkd_pixter< I > (A generic backward iterator on the pixels of a dpoint-based window or neighborhood) . . . 695

mln::dpoints_fwd_pixter< I > (A generic forward iterator on the pixels of a dpoint-based window or neighborhood) . . . 698
mln::dpsites_bkd_piter< V > (A generic backward iterator on points of windows and of neighborhoods) . . . 701
mln::dpsites_fwd_piter< V > (A generic forward iterator on points of windows and of neighborhoods) . . . 703
mln::Edge< E > (Edge category flag type) . . . 705
mln::edge_image< P, V, G > (Image based on graph edges) . . . 706
mln::extended< I > (Makes an image become restricted by a point set) . . . 708
mln::extension_fun< I, F > (Extends the domain of an image with a function) . . . 710
mln::extension_ima< I, J > (Extends the domain of an image with an image) . . . 713
mln::extension_val< I > (Extends the domain of an image with a value) . . . 716
mln::flat_image< T, S > (Image with a single value) . . . 719
mln::fun::p2b::antilogy (A p2b function always returning `false`) . . . 722
mln::fun::p2b::tautology (A p2b function always returning `true`) . . . 723
mln::fun::v2b::lnot< V > (Functor computing logical-not on a value) . . . 724
mln::fun::v2b::threshold< V > (Threshold function) . . . 725
mln::fun::v2v::ch_function_value< F, V > (Wrap a function v2v and convert its result to another type) . . . 726
mln::fun::v2v::component< T, i > (Functor that accesses the i-th component of a value) . . . 727
mln::fun::v2v::l1_norm< V, R > (L1-norm) . . . 728
mln::fun::v2v::l2_norm< V, R > (L2-norm) . . . 729
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) . . . 730
mln::fun::v2v::linfty_norm< V, R > (L-infty norm) . . . 731
mln::fun::v2w2v::cos< V > (Cosinus bijective functor) . . . 732
mln::fun::v2w_w2v::l1_norm< V, R > (L1-norm) . . . 733
mln::fun::v2w_w2v::l2_norm< V, R > (L2-norm) . . . 734
mln::fun::v2w_w2v::linfty_norm< V, R > (L-infty norm) . . . 735
mln::fun::vv2b::eq< L, R > (Functor computing equal between two values) . . . 736
mln::fun::vv2b::ge< L, R > (Functor computing "greater or equal than" between two values) . . . 737
mln::fun::vv2b::gt< L, R > (Functor computing "greater than" between two values) . . . 738
mln::fun::vv2b::implies< L, R > (Functor computing logical-implies between two values) . . . 739
mln::fun::vv2b::le< L, R > (Functor computing "lower or equal than" between two values) . . . 740
mln::fun::vv2b::lt< L, R > (Functor computing "lower than" between two values) . . . 741
mln::fun::vv2v::diff_abs< V > (A functor computing the diff_absimum of two values) . . . 742
mln::fun::vv2v::land< L, R > (Functor computing logical-and between two values) . . . 743
mln::fun::vv2v::land_not< L, R > (Functor computing logical and-not between two values) . . . 744
mln::fun::vv2v::lor< L, R > (Functor computing logical-or between two values) . . . 745
mln::fun::vv2v::lxor< L, R > (Functor computing logical-xor between two values) . . . 746
mln::fun::vv2v::max< V > (A functor computing the maximum of two values) . . . 747
mln::fun::vv2v::min< L, R > (A functor computing the minimum of two values) . . . 748
mln::fun::vv2v::vec< V > (A functor computing the vecimum of two values) . . . 749
mln::fun::x2p::closest_point< P > (FIXME: doxygen + concept checking) . . . 750
mln::fun::x2v::bilinear< I > (Represent a bilinear interolation of values from an underlying image) . . . 751
mln::fun::x2v::trilinear< I > (Represent a trilinear interolation of values from an underlying image) . . . 752
mln::fun::x2x::composed< T2, T1 > (Represent a composition of two transformations) . . . 753
mln::fun::x2x::linear< I > (Represent a linear interolation of values from an underlying image) 754
mln::fun::x2x::rotation< n, C > (Represent a rotation function) . . . 756
mln::fun::x2x::translation< n, C > (Translation function-object) . . . 759
mln::fun_image< F, I > (Image read through a function) . . . 762
mln::Function< E > (Base class for implementation of function-objects) . . . 764

mln::Function< void > (Function category flag type) . . . 765
mln::Function_v2b< E > (Base class for implementation of function-objects from a value to a Boolean) . . . 766
mln::Function_v2v< E > (Base class for implementation of function-objects from value to value) 767
mln::Function_vv2b< E > (Base class for implementation of function-objects from a couple of values to a Boolean) . . . 768
mln::Function_vv2v< E > (Base class for implementation of function-objects from a couple of values to a value) . . . 769
mln::fwd_pixter1d< I > (Forward pixel iterator on a 1-D image with border) . . . 770
mln::fwd_pixter2d< I > (Forward pixel iterator on a 2-D image with border) . . . 772
mln::fwd_pixter3d< I > (Forward pixel iterator on a 3-D image with border) . . . 774
mln::Gdpoint< E > (FIXME: Doc!) . . . 776
mln::Gdpoint< void > (Delta point site category flag type) . . . 777
mln::Generalized_Pixel< E > (Base class for implementation classes that are pixels or that have the behavior of pixels) . . . 778
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) . . . 779
mln::Gpoint< E > (Base class for implementation of point classes) . . . 781
mln::Graph< E > (Base class for implementation of graph classes) . . . 786
mln::graph::attribute::card_t (Compute the cardinality of every component in a graph) . . . 788
mln::graph::attribute::representative_t (Compute the representative vertex of every component in a graph) . . . 789
mln::graph_elt_neighborhood< G, S > (Elementary neighborhood on graph class) . . . 790
mln::graph_elt_neighborhood_if< G, S, I > (Elementary neighborhood_if on graph class) . . . 792
mln::graph_elt_window< G, S > (Elementary window on graph class) . . . 795
mln::graph_elt_window_if< G, S, I > (Custom window on graph class) . . . 799
mln::graph_window_base< P, E > . . . 804
mln::graph_window_if_piter< S, W, I > (Forward iterator on line graph window) . . . 806
mln::graph_window_piter< S, W, I > (Forward iterator on line graph window) . . . 808
mln::hexa< I > (Hexagonal image class) . . . 810
mln::histo::array< T > (Generic histogram class over a value set with type T) . . . 814
mln::Image< E > (Base class for implementation of image classes) . . . 815
mln::image1d< T > (Basic 1D image class) . . . 818
mln::image2d< T > (Basic 2D image class) . . . 823
mln::image2d_h< V > (2d image based on an hexagonal mesh) . . . 828
mln::image3d< T > (Basic 3D image class) . . . 831
mln::interpolated< I, F > (Makes the underlying image being accessed with floating coordinates) 836
mln::Iterator< E > (Base class for implementation classes that are iterators) . . . 838
mln::labeled_image< I > (Morpher providing an improved interface for labeled image) . . . 839
mln::lazy_image< I, F, B > (Image values are computed on the fly) . . . 842
mln::Literal< E > (Base class for implementation classes of literals) . . . 845
mln::literal::black_t (Type of literal black) . . . 847
mln::literal::blue_t (Type of literal blue) . . . 848
mln::literal::brown_t (Type of literal brown) . . . 849
mln::literal::cyan_t (Type of literal cyan) . . . 850
mln::literal::green_t (Type of literal green) . . . 851
mln::literal::identity_t (Type of literal identity) . . . 852
mln::literal::light_gray_t (Type of literal grays) . . . 853
mln::literal::lime_t (Type of literal lime) . . . 854
mln::literal::magenta_t (Type of literal magenta) . . . 855
mln::literal::max_t (Type of literal max) . . . 856
mln::literal::min_t (Type of literal min) . . . 857
mln::literal::olive_t (Type of literal olive) . . . 858
mln::literal::one_t (Type of literal one) . . . 859

mln::literal::orange_t (Type of literal orange) . . . 860
mln::literal::origin_t (Type of literal origin) . . . 861
mln::literal::pink_t (Type of literal pink) . . . 862
mln::literal::purple_t (Type of literal purple) . . . 863
mln::literal::red_t (Type of literal red) . . . 864
mln::literal::teal_t (Type of literal teal) . . . 865
mln::literal::violet_t (Type of literal violet) . . . 866
mln::literal::white_t (Type of literal white) . . . 867
mln::literal::yellow_t (Type of literal yellow) . . . 868
mln::literal::zero_t (Type of literal zero) . . . 869
mln::Mesh< E > (Base class for implementation classes of meshes) . . . 870
mln::Meta_Accumulator< E > (Base class for implementation of meta accumulators) . . . 871
mln::Meta_Function< E > (Base class for implementation of meta functions) . . . 873
mln::Meta_Function_v2v< E > (Base class for implementation of function-objects from value to value) . . . 874
mln::Meta_Function_vv2v< E > (Base class for implementation of function-objects from value to value) . . . 875
mln::metal::ands< E1, E2, E3, E4, E5, E6, E7, E8 > (Ands type) . . . 876
mln::metal::converts_to< T, U > ("converts-to" check) . . . 877
mln::metal::equal< T1, T2 > (Definition of a static 'equal' test) . . . 878
mln::metal::goes_to< T, U > ("goes-to" check) . . . 879
mln::metal::is< T, U > ("is" check) . . . 880
mln::metal::is_a< T, M > ("is_a" check) . . . 881
mln::metal::is_not< T, U > ("is_not" check) . . . 882
mln::metal::is_not_a< T, M > ("is_not_a" static Boolean expression) . . . 883
mln::morpho::attribute::card< I > (Cardinality accumulator class) . . . 884
mln::morpho::attribute::count_adjacent_vertices< I > (Count_Adjacent_Vertices accumulator class) . . . 886
mln::morpho::attribute::height< I > (Height accumulator class) . . . 888
mln::morpho::attribute::sharpness< I > (Sharpness accumulator class) . . . 890
mln::morpho::attribute::sum< I, S > (Suminality accumulator class) . . . 892
mln::morpho::attribute::volume< I > (Volume accumulator class) . . . 894
mln::neighb< W > (Adapter class from window to neighborhood) . . . 896
mln::Neighborhood< E > (Base class for implementation classes that are neighborhoods) . . . 898
mln::Neighborhood< void > (Neighborhood category flag type) . . . 899
mln::Object< E > (Base class for almost every class defined in Milena) . . . 900
mln::p2p_image< I, F > (FIXME: Doc!) . . . 901
mln::p_array< P > (Multi-set of sites) . . . 903
mln::p_centered< W > (Site set corresponding to a window centered on a site) . . . 908
mln::p_complex< D, G > (A complex psite set based on the N-faces of a complex of dimension D (a D-complex)) . . . 911
mln::p_edges< G, F > (Site set mapping graph edges and image sites) . . . 915
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)) . . . 920
mln::p_graph_piter< S, I > (Generic iterator on point sites of a mln::S) . . . 923
mln::p_if< S, F > (Site set restricted w.r.t) . . . 925
mln::p_image< I > (Site set based on an image of Booleans) . . . 928
mln::p_indexed_bkd_piter< S > (Backward iterator on sites of an indexed site set) . . . 932
mln::p_indexed_fwd_piter< S > (Forward iterator on sites of an indexed site set) . . . 933
mln::p_indexed_psite< S > (Psite class for indexed site sets such as p_array) . . . 934
mln::p_key< K, P > (Priority queue class) . . . 935
mln::p_line2d (2D discrete line of points) . . . 940
mln::p_mutable_array_of< S > (P_mutable_array_of is a mutable array of site sets) . . . 944

mln::p_n_faces_bkd_piter< D, P > (Backward iterator on the n-faces sites of an mln::p_complex<D, P>) . . . 948
mln::p_n_faces_fwd_piter< D, P > (Forward iterator on the n-faces sites of an mln::p_complex<D, P>) . . . 950
mln::p_priority< P, Q > (Priority queue) . . . 952
mln::p_queue< P > (Queue of sites (based on std::deque)) . . . 958
mln::p_queue_fast< P > (Queue of sites class (based on p_array) . . . 962
mln::p_run< P > (Point set class in run) . . . 967
mln::p_set< P > (Mathematical set of sites (based on util::set)) . . . 972
mln::p_transformed< S, F > (Site set transformed through a function) . . . 976
mln::p_transformed_piter< Pi, S, F > (Iterator on p_transformed<S,F>) . . . 979
mln::p_vaccess< V, S > (Site set in which sites are grouped by their associated value) . . . 981
mln::p_vertices< G, F > (Site set based mapping graph vertices to sites) . . . 985
mln::pixel< I > (Generic pixel class) . . . 991
mln::plain< I > (Prevents an image from sharing its data) . . . 993
mln::Point< P > (Base class for implementation of point classes) . . . 995
mln::point< G, C > (Generic point class) . . . 998
mln::Proxy< E > (Base class for implementation classes of the notion of "proxy") . . . 1004
mln::Proxy< void > (Proxy category flag type) . . . 1005
mln::Pseudo_Site< E > (Base class for implementation classes of the notion of "pseudo site") . . . 1006
mln::Pseudo_Site< void > (Pseudo_Site category flag type) . . . 1007
mln::pw::image< F, S > (A generic point-wise image implementation) . . . 1008
mln::registration::closest_point_basic< P > (Closest point functor based on map distance) . . . 1009
mln::registration::closest_point_with_map< P > (Closest point functor based on map distance) 1010
mln::Regular_Grid< E > (Base class for implementation classes of regular grids) . . . 1011
mln::safe_image< I > (Makes an image accessible at undefined location) . . . 1012
mln::select::p_of< P > (Structure p_of) . . . 1013
mln::Site< E > (Base class for classes that are explicitly sites) . . . 1014
mln::Site< void > (Site category flag type) . . . 1015
mln::Site_Iterator< E > (Base class for implementation of classes of iterator on points) . . . 1016
mln::Site_Proxy< E > (Base class for implementation classes of the notion of "site proxy") . . 1018
mln::Site_Proxy< void > (Site_Proxy category flag type) . . . 1019
mln::Site_Set< E > (Base class for implementation classes of site sets) . . . 1020
mln::Site_Set< void > (Site_Set category flag type) . . . 1024
mln::sub_image< I, S > (Image having its domain restricted by a site set) . . . 1025
mln::sub_image_if< I, S > (Image having its domain restricted by a site set and a function) . . 1027
mln::thru_image< I, F > (Morph image values through a function) . . . 1029
mln::thrubin_image< I1, I2, F > (Morphes values from two images through a binary function) . 1030
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>) . . . 1032
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>) . . . 1034
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>) . . . 1036
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>) . . . 1037
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>) . . . 1038
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>) . . . 1040

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>) . . . 1042
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>) . . . 1043
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>) . . . 1044
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>) . . . 1045
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) . . . 1046
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) . . . 1048
mln::topo::algebraic_face< D > (Algebraic face handle in a complex; the face dimension is dynamic) . . . 1050
mln::topo::algebraic_n_face< N, D > (Algebraic N-face handle in a complex) . . . 1054
mln::topo::center_only_iter< D > (Iterator on all the adjacent (n-1)-faces of the n-face of an mln::complex<D>) . . . 1058
mln::topo::centered_bkd_iter_adapter< D, I > (Forward complex relative iterator adapters adding the central (reference) point to the set of iterated faces) . . . 1060
mln::topo::centered_fwd_iter_adapter< D, I > (Backward complex relative iterator adapters adding the central (reference) point to the set of iterated faces) . . . 1061
mln::topo::complex< D > (General complex of dimension D) . . . 1062
mln::topo::face< D > (Face handle in a complex; the face dimension is dynamic) . . . 1065
mln::topo::face_bkd_iter< D > (Backward iterator on all the faces of an mln::complex<D>) . . . 1069
mln::topo::face_fwd_iter< D > (Forward iterator on all the faces of an mln::complex<D>) . . . 1071
mln::topo::is_n_face< N > (A functor testing wheter a mln::complex_psite is an N -face) . . . 1073
mln::topo::is_simple_cell< I > (A predicate for the simplicity of a point based on the collapse property of the attachment) . . . 1074
mln::topo::n_face< N, D > (N-face handle in a complex) . . . 1077
mln::topo::n_face_bkd_iter< D > (Backward iterator on all the faces of an mln::complex<D>) 1081
mln::topo::n_face_fwd_iter< D > (Forward iterator on all the faces of an mln::complex<D>) . 1083
mln::topo::n_faces_set< N, D > (Set of face handles of dimension N) . . . 1085
mln::topo::static_n_face_bkd_iter< N, D > (Backward iterator on all the N-faces of a mln::complex<D>) . . . 1087
mln::topo::static_n_face_fwd_iter< N, D > (Forward iterator on all the N-faces of a mln::complex<D>) . . . 1089
mln::tr_image< S, I, T > (Transform an image by a given transformation) . . . 1091
mln::transformed_image< I, F > (Image having its domain restricted by a site set) . . . 1094
mln::unproject_image< I, D, F > (Un-projects an image) . . . 1096
mln::util::adjacency_matrix< V > (A class of adjacency matrix) . . . 1098
mln::util::array< T > (A dynamic array class) . . . 1099
mln::util::branch< T > (Class of generic branch) . . . 1106
mln::util::branch_iter< T > (Basic 2D image class) . . . 1108
mln::util::branch_iter_ind< T > (Basic 2D image class) . . . 1110
mln::util::couple< T, U > (Definition of a couple) . . . 1112
mln::util::eat (Eat structure) . . . 1114
mln::util::edge< G > (Edge of a graph G) . . . 1115
mln::util::fibonacci_heap< P, T > (Fibonacci heap) . . . 1119
mln::util::graph (Undirected graph) . . . 1122
mln::util::greater_point< I > (A “greater than” functor comparing points w.r.t) . . . 1128
mln::util::greater_psite< I > (A “greater than” functor comparing psites w.r.t) . . . 1129
mln::util::head< T, R > (Top structure of the soft heap) . . . 1130
mln::util::ignore (Ignore structure) . . . 1131
mln::util::ilcell< T > (Element of an item list. Store the data (key) used in soft_heap) . . . 1132

mln::util::line_graph< G > (Undirected line graph of a graph of type G) . . . 1133
mln::util::nil (Nil structure) . . . 1138
mln::util::node< T, R > (Meta-data of an element in the heap) . . . 1139
mln::util::object_id< Tag, V > (Base class of an object id) . . . 1140
mln::util::ord< T > (Function-object that defines an ordering between objects with type T: *lhs* R *rhs*) . . . 1142
mln::util::ord_pair< T > (Ordered pair structure s.a) . . . 1143
mln::util::pix< I > (Structure pix) . . . 1145
mln::util::set< T > (An "efficient" mathematical set class) . . . 1147
mln::util::site_pair< P > (A pair of sites) . . . 1153
mln::util::soft_heap< T, R > (Soft heap) . . . 1155
mln::util::timer (Timer structure) . . . 1158
mln::util::tracked_ptr< T > (Smart pointer for shared data with tracking) . . . 1159
mln::util::tree< T > (Class of generic tree) . . . 1161
mln::util::tree_node< T > (Class of generic tree_node for tree) . . . 1163
mln::util::vertex< G > (Vertex of a graph G) . . . 1167
mln::util::yes (Object that always says "yes") . . . 1172
mln::Value< E > (Base class for implementation classes of values) . . . 1173
mln::value::float01 (Class for floating values restricted to the interval [0) . . . 1174
mln::value::float01_f (Class for floating values restricted to the interval [0..1]) . . . 1177
mln::value::graylevel< n > (General gray-level class on n bits) . . . 1179
mln::value::graylevel_f (General gray-level class on n bits) . . . 1182
mln::value::int_s< n > (Signed integer value class) . . . 1185
mln::value::int_u< n > (Unsigned integer value class) . . . 1187
mln::value::int_u_sat< n > (Unsigned integer value class with saturation behavior) . . . 1189
mln::value::Integer< E > (Concept of integer) . . . 1192
mln::value::Integer< void > (Category flag type) . . . 1193
mln::value::label< n > (Label value class) . . . 1194
mln::value::lut_vec< S, T > (Class that defines FIXME) . . . 1197
mln::value::proxy< I > (Generic proxy class for an image pixel value) . . . 1200
mln::value::proxy< const I > (Generic proxy class for an image pixel value) . . . 1203
mln::value::rgb< n > (Color class for red-green-blue where every component is n-bit encoded) 1206
mln::value::set< T > (Class that defines the set of values of type T) . . . 1208
mln::value::sign (Value type composed by the set (-1, 0, 1) sign value type is a subset of the int value type) . . . 1209
mln::value::stack_image< n, I > (Stack image class) . . . 1211
mln::Vertex< E > (Vertex category flag type) . . . 1214
mln::vertex_image< P, V, G > (Image based on graph vertices) . . . 1215
mln::w_window< D, W > (Generic w_window class) . . . 1217
mln::Weighted_Window< E > (Base class for implementation classes that are weighted_-windows) . . . 1221
mln::win::backdiag2d (Diagonal line window defined on the 2D square grid) . . . 1222
mln::win::ball< G, C > (Generic ball window defined on a given grid) . . . 1223
mln::win::cube3d (Cube window defined on the 3D grid) . . . 1224
mln::win::cuboid3d (Cuboid defined on the 3-D square grid) . . . 1226
mln::win::diag2d (Diagonal line window defined on the 2D square grid) . . . 1228
mln::win::line< M, i, C > (Generic line window defined on a given grid in the given dimension) 1229
mln::win::multiple< W, F > (Multiple window) . . . 1231
mln::win::multiple_size< n, W, F > (Definition of a multiple-size window) . . . 1232
mln::win::octagon2d (Octagon window defined on the 2D square grid) . . . 1233
mln::win::rectangle2d (Rectangular window defined on the 2D square grid) . . . 1235
mln::Window< E > (Base class for implementation classes that are windows) . . . 1237
mln::window< D > (Generic window class) . . . 1238

mln::world::inter_pixel::is_separator (Functor returning whether a site is a separator in an inter-pixel image) 1243

Chapter 6

Module Documentation

6.1 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.

6.1.1 Detailed Description

Accumulators working on site sets.

6.2 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.

6.2.1 Detailed Description

Accumulators working on images.

6.3 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 >

  Count 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::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.

6.3.1 Detailed Description

Accumulators working on image values.

6.4 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.

6.4.1 Detailed Description

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

6.5 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.

6.5.1 Detailed Description

All graphes implementations.

6.6 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 functionnalities.

6.6.1 Detailed Description

All the generic image types provided in Olena.

6.7 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::image1d< 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.

6.7.1 Detailed Description

Concrete images.

6.8 Image morphers

Morpher on both image values and domain.

Morpher on both image values and domain.

6.9 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.

6.9.1 Detailed Description

Morpher on image values.

6.10 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::p2p_image< I, F >

  FIXME: Doc!

- struct 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::transformed_image< I, F >

  Image having its domain restricted by a site set.

- struct mln::unproject_image< I, D, F >

  Un-projects an image.

6.10.1 Detailed Description

Morpher on image domain.

6.11 Identity morphers

Morpher adding new functionnalities.

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.

6.11.1 Detailed Description

Morpher adding new functionnalities.

6.12 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

  Miscalleneous useful containers/structures.

- Windows

  All the predefined generic windows.

6.12.1 Detailed Description

Milena Object types.

6.13 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.

6.13.1 Detailed Description

All accumulator types.

6.14 Routines

All algorithms/routines provided in Milena.

All algorithms/routines provided in Milena.

6.15 Canvas

All canvas.

All canvas.

6.16 Functions

All predefined functions.

Classes

- struct mln::Function< E >

  Base class for implementation of function-objects.

- 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::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.

6.16.1 Detailed Description

All predefined functions.

6.17 Neighborhoods

All the predefined generic neighborhoods.

Modules

- 1D neighborhoods

  Predefined 1D neighborhoods.

- 2D neighborhoods

  Predefined 2D neighborhoods.

- 3D neighborhoods

  Predefined 3D neighborhoods.

6.17.1 Detailed Description

All the predefined generic neighborhoods.

6.18 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.

6.18.1 Detailed Description

Predefined 1D neighborhoods.

6.18.2 Typedef Documentation

6.18.2.1 typedef neighb<window1d> mln::neighb1d

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

6.18.3 Function Documentation

6.18.3.1 const neighb1d & mln::c2 () [inline]

2-connectivity neighborhood on the 1D grid.

```
o x o
```

Returns:

A neighb1d.

Referenced by mln::geom::mesh_curvature().

6.19 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.

6.19.1 Detailed Description

Predefined 2D neighborhoods.

6.19.2 Typedef Documentation

6.19.2.1 typedef neighb<window2d> mln::neighb2d

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

6.19.3 Function Documentation

6.19.3.1 const neighb2d & mln::c2_col () [inline]

Vertical 2-connectivity neighborhood on the 2D grid.

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

Returns:

A neighb2d.

6.19.3.2 const neighb2d & mln::c2_row () [inline]

Horizontal 2-connectivity neighborhood on the 2D grid.

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

Returns:

A neighb2d.

6.19.3.3 const neighb2d & mln::c4 () [inline]

4-connectivity neighborhood on the 2D grid.

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

Returns:

A neighb2d.

6.19.3.4 const neighb2d & mln::c8 () [inline]

8-connectivity neighborhood on the 2D grid.

```
o o o
o x o
o o o
```

Returns:

A neighb2d.

6.20 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::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.

6.20.1 Detailed Description

Predefined 3D neighborhoods.

6.20.2 Typedef Documentation

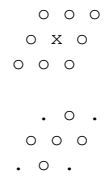
6.20.2.1 typedef neighb<window3d> mln::neighb3d

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

6.20.3 Function Documentation

6.20.3.1 const neighb3d & mln::c18 () [inline]

18-connectivity neighborhood on the 3D grid.



Returns:

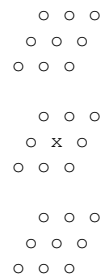
A neighb3d.

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

Referenced by mln::c26().

6.20.3.2 const neighb3d & mln::c26 () [inline]

26-connectivity neighborhood on the 3D grid.



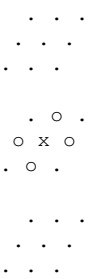
Returns:

A neighb3d.

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

6.20.3.3 const neighb3d & mln::c4_3d () [inline]

4-connectivity neighborhood on the 3D grid.



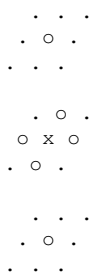
Returns:

A neighb3d.

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

6.20.3.4 const neighb3d & mln::c6 () [inline]

6-connectivity neighborhood on the 3D grid.



Returns:

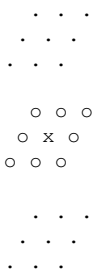
A neighb3d.

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

Referenced by mln::c18().

6.20.3.5 const neighb3d & mln::c8_3d () [inline]

8-connectivity neighborhood on the 3D grid.



Returns:

A neighb3d.

6.21 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.

6.21.1 Detailed Description

All Site set types.

6.22 Basic types

Basic site sets.

Classes

- struct 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.

6.22.1 Detailed Description

Basic site sets.

6.23 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 a 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.

6.23.1 Detailed Description

Site sets based on a graph.

6.24 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`*).*

6.24.1 Detailed Description

Site sets based on a complexes.

6.25 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.

6.25.1 Detailed Description

Sparse site sets.

6.26 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.

6.26.1 Detailed Description

Site sets based on a queue.

6.27 Utilities

Miscalleneous 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".

6.27.1 Detailed Description

Miscalleneous useful containers/structures.

6.28 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.

6.28.1 Detailed Description

All the predefined generic windows.

6.29 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.

6.29.1 Detailed Description

Predefined 1D windows.

6.29.2 Typedef Documentation

6.29.2.1 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:

```
o x o
```

is defined with length = 3.

6.29.2.2 typedef window<mln::dpoint1d> mln::window1d

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

6.30 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.

6.30.1 Detailed Description

Predefined 2D windows.

6.30.2 Typedef Documentation

6.30.2.1 typedef ball<grid::square, def::coord> mln::win::disk2d

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

6.30.2.2 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.

6.30.2.3 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.

6.30.2.4 typedef window<mln::dpoint2d> mln::window2d

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

6.30.3 Function Documentation

6.30.3.1 const window2d & mln::win_c4p () [inline]

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

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

Returns:

A window2d.

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

6.30.3.2 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.

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

6.31 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 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.

6.31.1 Detailed Description

Predefined 3D windows.

6.31.2 Typedef Documentation

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

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

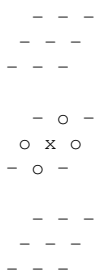
6.31.2.2 typedef window<mln::dpoint3d> mln::window3d

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

6.31.3 Function Documentation

6.31.3.1 const window3d & mln::win_c4p_3d () [inline]

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



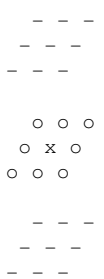
Returns:

A window3d.

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

6.31.3.2 const window3d & mln::win_c8p_3d () [inline]

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



Returns:

A window3d.

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

6.32 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.

6.32.1 Detailed Description

Predefined N-D windows.

6.33 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.

6.33.1 Detailed Description

Generic multiple windows.

6.34 v2w2v functions

All bijective functions.

All bijective functions.

6.35 v2w_w2v functions

All bijective function.

All bijective function.

6.36 vv2b functions

All functions mapping two values to a logical value.

All functions mapping two values to a logical value.

Chapter 7

Namespace Documentation

7.1 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.

- struct box

  Generic box class: site set containing points of a regular grid.

- struct Box

  Base class for implementation classes of boxes.

- 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

  FIXME: Doc!

- 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.

- class dpsites_bkd_piter

  A generic backward iterator on points of windows and of neighborhoods.

- class dpsites_fwd_piter

  A generic forward iterator on points of windows and of neighborhoods.

- 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.

- 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_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 Generalized_Pixel

  Base class for implementation classes that are pixels or that have the behavior of pixels.

- struct Gpoint

  Base class for implementation of point classes.

- struct Graph

Base class for implementation of graph classes.

- 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 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.

- 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 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, P>.

- class p_n_faces_fwd_piter

  Forward iterator on the n-faces sites of an mln::p_complex<D, P>.

- 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_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.

- class plain

  Prevents an image from sharing its data.

- struct point

  Generic point class.

- struct Point

  Base class for implementation of point classes.

- 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 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 transformed_image
  Image having its domain restricted by a site set.
- struct unproject_image
  Un-projects an image.
- struct Value
  Base class for implementation classes of values.
- struct Vertex
  Vertex category flag type.
- class vertex_image
  Image based on graph vertices.
- 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< 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 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 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::psite a_point_of (const Image< I > &ima)

  Give a point of an image.

- template<typename I , typename F >
  p2p_image< const I, F > apply_p2p (const Image< I > &ima, const Function_v2v< F > &f)

  FIXME: Doc!

- template<typename I , typename F >
  p2p_image< I, F > apply_p2p (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 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 >
  extension_val< const I > extend (const Image< I > &ima, const typename I::value &val)

  Routines for domain extension with a value.

- 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 , typename F >
  extension_fun< const I, F > extend (const Image< I > &ima, const Function_v2v< F > &fun)

  Routines for domain extension with a function.

- bool implies (bool lexpr, bool rexpr)

  Implication.

- template<typename I , typename J >
  void initialize (Image< I > &target, const Image< J > &model)
- template<typename I , typename N >
  bool is_simple_2d (const Image< I > &ima, const Neighborhood< N > &nbh, const typename I::psite &p)

  Test if a point is simple or not.

- 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_m_face_neighborhood, topo::adj_m_face_fwd_iter, topo::adj_m_face_bkd_iter)

  Neighborhood centered on an n-face of complex returning the m-faces transitively adjacent to this center n-face.

- mln_gen_complex_neighborhood (complex_higher_dim_connected_n_face_neighborhood, topo::adj_higher_dim_connected_n_face_fwd_iter, topo::adj_higher_dim_connected_n_face_bkd_iter)

  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_lower_dim_connected_n_face_neighborhood, topo::adj_lower_dim_connected_n_face_fwd_iter, topo::adj_lower_dim_connected_n_face_bkd_iter)

  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_lower_higher_neighborhood, topo::adj_lower_higher_face_fwd_iter, topo::adj_lower_higher_face_bkd_iter)

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

- mln_gen_complex_neighborhood (complex_higher_neighborhood, topo::adj_higher_face_fwd_iter, topo::adj_higher_face_bkd_iter)

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

- mln_gen_complex_neighborhood (complex_lower_neighborhood, topo::adj_lower_face_fwd_iter, topo::adj_lower_face_bkd_iter)

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

- 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 (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 (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 (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 (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 (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.

- template<typename W1 , typename W2 >
  mln_regular (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 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 P , typename D >
  P operator- (const Gpoint< P > &p, const Gdpoint< D > &dp)

  Substract a delta-point `dp` *to a grid point* `p`.

- 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_bkd_piter< I, G, W > &p)

  Print an mln::complex_window_bkd_piter.

- 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 N >
  std::ostream & operator<< (std::ostream &ostr, const complex_neighborhood_bkd_piter< I, G, N > &p)

  Print an mln::complex_neighborhood_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 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<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<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<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 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<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<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_edges< G, F > &lhs, const p_edges< G, F > &rhs)

  Comparison between two mln::p_edges's.

- 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<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 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 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 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 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 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 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<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)
  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*?*

- 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 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<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.

Variables

- const dpoint1d before = dpoint1d(-1)
  Definition of a shortcut for delta point in 1d.

- const dpoint3d sagittal_dec = dpoint3d(0, 0, -1)
  Definition of a shortcut for delta point in 3d.

- const dpoint2d up = dpoint2d(-1, 0)
  Definition of a shortcut for delta point in 2d.

7.1.1 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".

mln/linear/convolve_directional.hh

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 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 olena 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.

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

7.1.2 Typedef Documentation

7.1.2.1 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.

7.1.2.2 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.

7.1.2.3 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.

7.1.2.4 typedef box<point2d_h> mln::box2d_h

FIXME.

7.1.2.5 typedef box<point3d> mln::box3d

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

See also:

mln::win::rectangle3d.

7.1.2.6 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).

7.1.2.7 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).

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

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

7.1.2.9 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.

7.1.2.10 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.

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

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

7.1.2.12 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.

7.1.2.13 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.

7.1.2.14 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.

7.1.2.15 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.

7.1.2.16 typedef p_run<point2d> mln::p_run2d

Type alias for a run of 2d points.

7.1.2.17 typedef p_set_of<p_run2d> mln::p_runs2d

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

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

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

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

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

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

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

7.1.2.21 typedef point< grid::hexa, def::coord > mln::point2d_h

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

7.1.2.22 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.

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

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

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

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

7.1.2.25 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.

7.1.2.26 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).

7.1.2.27 typedef algebra::vec<2u,double> mln::vec2d_d

2D vector with double coordinates.

7.1.2.28 typedef algebra::vec<2u,float> mln::vec2d_f

2D vector with float coordinates.

7.1.2.29 typedef algebra::vec<3u,double> mln::vec3d_d

3D vector with double coordinates.

7.1.2.30 typedef algebra::vec<3u,float> mln::vec3d_f

3D vector with float coordinates.

7.1.3 Function Documentation

7.1.3.1 template<typename I > I::psite mln::a_point_of (const Image< I > & *ima*) [inline]

Give a point of an image.

7.1.3.2 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!

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

FIXME: Doc!

Referenced by mln::debug::slices_2d().

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

Do a composition of two transformations.

Parameters:

← *f* The second transformation.

← *g* The first transformation.

Returns:

The composed transformation fog.

References compose().

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

7.1.3.5 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:

← *model* The image to be duplicated.

Returns:

The duplicate.

Precondition:

model.is_valid

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

Referenced by mln::registration::icp(), mln::plain< I >::operator I(), mln::geom::impl::seeds2tiling(), and mln::geom::impl::seeds2tiling_roundness().

7.1.3.6 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.

7.1.3.7 template<typename I , typename J > extension_ima< const I, const J > mln::extend (const Image< I > & *ima*, const Image< J > & *ext*) [inline]

Routines for domain extension with an image.

7.1.3.8 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.

Referenced by mln::geom::rotate().

7.1.3.9 bool mln::implies (bool *lexpr*, bool *rexpr*) [inline]

Implication.

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

7.1.3.10 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:

↔ ***target*** The image to be initialized.

← ***model*** The image to provide data for the initialization.

Precondition:

(not target.is_valid) and model.is_valid

Referenced by duplicate(), mln::labeling::fill_holes(), mln::morpho::tree::filter::filter(), mln::linear::gaussian(), mln::linear::gaussian_1st_derivative(), mln::linear::gaussian_2nd_derivative(), mln::morpho::impl::generic::hit_or_miss(), mln::graph::labeling(), mln::io::magick::load(),

mln::io::dicom::load(), make_debug_graph_image(), mln::morpho::tree::filter::max(), mln::data::impl::generic::median(), mln::morpho::meyer_wst(), mln::morpho::tree::filter::min(), mln::arith::min(), mln::arith::minus(), mln::arith::plus(), mln::morpho::impl::generic::rank_filter(), mln::arith::revert(), mln::geom::rotate(), mln::data::impl::stretch(), and mln::data::impl::generic::transform().

7.1.3.11 template<typename I , typename N > bool mln::is_simple_2d (const Image< I > & *ima*, const Neighborhood< N > & *nbh*, const typename I::psite & *p*) [inline]

Test if a point is simple or not.

A point of an object is simple if in its c8 neiborhood, 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. | | |

7.1.3.12 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`.

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

7.1.3.13 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.

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().

7.1.3.14 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).

Referenced by mln::geom::rotate(), and mln::convert::to().

7.1.3.15 mln::mln_gen_complex_neighborhood (complex_m_face_neighborhood, topo::adj_m_face_fwd_iter, topo::adj_m_face_bkd_iter)

Neighborhood centered on an n-face of complex returning the m-faces transitively adjacent to this center n-face.

7.1.3.16 mln::mln_gen_complex_neighborhood (complex_higher_dim_connected_n_face_neighborhood, topo::adj_higher_dim_connected_n_face_fwd_iter, topo::adj_higher_dim_connected_n_face_bkd_iter)

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

7.1.3.17 mln::mln_gen_complex_neighborhood (complex_lower_dim_connected_n_face_neighborhood, topo::adj_lower_dim_connected_n_face_fwd_iter, topo::adj_lower_dim_connected_n_face_bkd_iter)

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

7.1.3.18 mln::mln_gen_complex_neighborhood (complex_lower_higher_neighborhood, topo::adj_lower_higher_face_fwd_iter, topo::adj_lower_higher_face_bkd_iter)

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

7.1.3.19 mln::mln_gen_complex_neighborhood (complex_higher_neighborhood, topo::adj_higher_face_fwd_iter, topo::adj_higher_face_bkd_iter)

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

7.1.3.20 mln::mln_gen_complex_neighborhood (complex_lower_neighborhood, topo::adj_lower_face_fwd_iter, topo::adj_lower_face_bkd_iter)

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

7.1.3.21 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.

7.1.3.22 mln::mln_gen_complex_window (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.

7.1.3.23 mln::mln_gen_complex_window (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.

7.1.3.24 mln::mln_gen_complex_window (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.

7.1.3.25 mln::mln_gen_complex_window (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.

7.1.3.26 mln::mln_gen_complex_window (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.

7.1.3.27 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.

7.1.3.28 template<typename O1 , typename O2 > mln::mln_trait_op_geq (O1, O2) [inline]

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.

7.1.3.29 template<typename O1 , typename O2 > mln::mln_trait_op_greater (O1, O2) const [inline]

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.

7.1.3.30 template<typename O1 , typename O2 > mln::mln_trait_op_leq (O1, O2) [inline]

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.

7.1.3.31 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.

7.1.3.32 template<unsigned D, typename G > bool mln::operator!= (const complex_psite< D, G > & *lhs*, const complex_psite< D, G > & *rhs*) [inline]

Is *lhs* not equal to *rhs*?

Precondition:

Arguments *lhs* and *rhs* must belong to the same mln::p_complex.

References mln::complex_psite< D, G >::face(), and mln::complex_psite< D, G >::site_set().

7.1.3.33 template<unsigned N, unsigned D, typename P > bool mln::operator!= (const faces_psite< N, D, P > & *lhs*, const faces_psite< N, D, P > & *rhs*) [inline]

Is *lhs* equal to *rhs*?

Precondition:

Arguments *lhs* and *rhs* must belong to the same mln::complex.

Referenced by mln::topo::operator!=().

7.1.3.34 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`.

7.1.3.35 template<typename S > S & mln::operator++ (value::Scalar< S > & *rhs*) [inline]

Pre-incrementation for any scalar type.

References mln::literal::one.

7.1.3.36 template<typename N1 , typename N2 > N2 neighb< typename N1::window::regular > mln::operator- (const Neighborhood< N1 > & *nbh1*, const Neighborhood< N2 > & *nbh2*) [inline]

Set difference between a couple of neighborhoods `nbh1` and `nbh2`.

It just calls mln::win::diff.

References mln::win::diff().

7.1.3.37 template<typename P , typename D > P mln::operator- (const Gpoint< P > & *p*, const Gdpoint< D > & *dp*) [inline]

Substract a delta-point `dp` to a grid point `p`.

Parameters:

← ***p*** A grid point.

← ***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

7.1.3.38 template<typename S > S & mln::operator– (value::Scalar< S > & *rhs*) [inline]

Pre-decrementation for any scalar type.

References mln::literal::one.

7.1.3.39 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:

← ***lhs*** A first image.

← ***rhs*** A second image.

Precondition:

lhs.domain == rhs.domain

References mln::test::predicate().

7.1.3.40 template<unsigned D, typename G > bool mln::operator< (const complex_psite< D, G > & *lhs*, const complex_psite< D, G > & *rhs*) [inline]

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.

7.1.3.41 template<unsigned N, unsigned D, typename P > bool mln::operator< (const faces_psite< N, D, P > & *lhs*, const faces_psite< N, D, P > & *rhs*) [inline]

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.

7.1.3.42 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.

7.1.3.43 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.

7.1.3.44 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.

7.1.3.45 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.

7.1.3.46 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 or equal to the pixel values of `rhs`.

Parameters:

← ***lhs*** A first image.

← ***rhs*** A second image.

Precondition:

lhs.domain == rhs.domain

References mln::test::predicate().

7.1.3.47 template<typename G , typename F > bool mln::operator<= (const p_vertices< G, F > & *lhs*, const p_vertices< G, F > & *rhs*) [inline]

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.

7.1.3.48 template<unsigned N, unsigned D, typename P > bool mln::operator<= (const p_faces< N, D, P > & *lhs*, const p_faces< N, D, P > & *rhs*) [inline]

Inclusion of a mln::p_faces in another one.

This inclusion relation is very strict for the moment, since our infrastrure for complexs is simple: a mln::p_faces is included in another one if their are equal.

7.1.3.49 template<typename G , typename F > bool mln::operator<= (const p_edges< G, F > & *lhs*, const p_edges< G, F > & *rhs*) [inline]

Inclusion of a mln::p_edges in another one.

7.1.3.50 template<unsigned D, typename G > bool mln::operator<= (const p_complex< D, G > & *lhs*, const p_complex< D, G > & *rhs*) [inline]

Inclusion of a mln::p_complex in another one.

This inclusion relation is very strict for the moment, since our infrastrure for complexs is simple: a mln::p_complex is included in another one if their are equal.

7.1.3.51 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:

← ***lhs*** A first image.

← ***rhs*** A second image.

Precondition:

lhs.domain == rhs.domain

References mln::test::predicate().

7.1.3.52 template<typename G , typename F > bool mln::operator== (const p_vertices< G, F > & *lhs*, const p_vertices< G, F > & *rhs*) [inline]

Comparison between two mln::p_vertices's.

Two mln::p_vertices's are considered equal if they share the same graph.

References mln::p_vertices< G, F >::graph().

7.1.3.53 template<unsigned N, unsigned D, typename P > bool mln::operator== (const p_faces< N, D, P > & *lhs*, const p_faces< N, D, P > & *rhs*) [inline]

Comparison between two mln::p_faces's.

Two mln::p_faces's are considered equal if they share the same complex.

References mln::p_faces< N, D, P >::cplx().

7.1.3.54 template<typename G , typename F > bool mln::operator== (const p_edges< G, F > & *lhs*, const p_edges< G, F > & *rhs*) [inline]

Comparison between two mln::p_edges's.

Two mln::p_edges's are considered equal if they share the same graph.

References mln::p_edges< G, F >::graph().

7.1.3.55 template<unsigned D, typename G > bool mln::operator== (const p_complex< D, G > & *lhs*, const p_complex< D, G > & *rhs*) [inline]

Comparison between two mln::p_complex's.

Two mln::p_complex's are considered equal if they share the same complex.

References mln::p_complex< D, G >::cplx().

7.1.3.56 template<unsigned D, typename G > bool mln::operator== (const complex_psite< D, G > & *lhs*, const complex_psite< D, G > & *rhs*) [inline]

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.

References mln::complex_psite< D, G >::face(), and mln::complex_psite< D, G >::site_set().

7.1.3.57 template<unsigned N, unsigned D, typename P > bool mln::operator== (const faces_psite< N, D, P > & *lhs*, const faces_psite< N, D, P > & *rhs*) [inline]

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.

7.1.3.58 template<typename F , typename S > pw::image< F, S > mln::operator| (const Function_v2v< F > & *f*, const Site_Set< S > & *ps*) [inline]

Construct an image from a function and a site set.

image = function | site_set.

7.1.3.59 template<typename S , typename F > p_if< S, F > mln::operator| (const Site_Set< S > & *s*, const Function_v2b< F > & *f*) [inline]

Restrict a site set `s` to points that verify `f`.

Parameters:

← ***s*** A site set.

← ***f*** A function from point to Boolean.

Returns:

A subset of points.

7.1.3.60 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*) [inline]

Construct a vertex image from a fun::i2v::array and a p_vertices.

image = fun::i2v::array | p_vertices.

7.1.3.61 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*) [inline]

Construct a edge image from a fun::i2v::array and a p_edges.

image = fun::i2v::array | p_edges.

7.1.3.62 template<typename I , typename F > image_if< const I, F > mln::operator| (const Image< I > & *ima*, const Function_v2b< F > & *f*) [inline]

ima | f creates an image_if with the image ima and the function f.

7.1.3.63 template<typename I , typename F > image_if< I, F > mln::operator| (Image< I > & *ima*, const Function_v2b< F > & *f*) [inline]

ima | f creates an image_if with the image ima and the function f.

7.1.3.64 template<typename I > const internal::primary_type< I >::ret & mln::primary (const Image< I > & *input*) [inline]

FIXME: Doc!

Referenced by mln::border::resize().

7.1.3.65 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:

← ***s*** A site set.

← ***f*** A function from site to site.

Returns:

The transformed site set.

7.1.4 Variable Documentation

7.1.4.1 const dpoint1d mln::before = dpoint1d(-1)

Definition of a shortcut for delta point in 1d.

7.1.4.2 const dpoint3d mln::sagittal_dec = dpoint3d(0, 0, -1)

Definition of a shortcut for delta point in 3d.

7.1.4.3 const dpoint2d mln::up = dpoint2d(-1, 0)

Definition of a shortcut for delta point in 2d.

7.2 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

  Count 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`.

7.2.1 Detailed Description

Namespace of accumulators.

7.2.2 Function Documentation

7.2.2.1 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:

← ***input*** The input image.

← ***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().

7.2.2.2 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*) [inline]

Line an accumulator onto the pixel values of the image `input`.

Parameters:

← ***input*** The input image.

← ***p_start*** The starting site of the line.

← ***len*** The line length.

← ***half_length*** The half length of the line.

↔ ***output*** The resulting image.

This routine runs:

tmp = `a`

tmp.init()

accu::take(`input`, tmp)

return tmp.to_result()

7.2.2.3 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:

← ***input*** The input image.

← ***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().

7.2.2.4 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:

← ***input*** The input image.

↔ ***a*** The accumulator.

This routine runs:

for all p of `input`, `a.take`(pix(`input`, p))

Warning:

This routine does not perform a.init().

7.3 mln::accu::image Namespace Reference

Namespace of accumulator image routines.

7.3.1 Detailed Description

Namespace of accumulator image routines.

7.4 mln::accu::impl Namespace Reference

Implementation namespace of accumulator namespace.

7.4.1 Detailed Description

Implementation namespace of accumulator namespace.

7.5 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.

7.5.1 Detailed Description

Namespace of logical accumulators.

7.6 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.

7.6.1 Detailed Description

Namespace of mathematic accumulators.

7.7 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.

7.7.1 Detailed Description

Namespace of logical meta-accumulators.

7.8 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.

7.8.1 Detailed Description

Namespace of mathematic meta-accumulators.

7.9 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.

7.9.1 Detailed Description

Namespace of shape meta-accumulators.

7.10 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.

7.10.1 Detailed Description

Namespace of statistical meta-accumulators.

7.11 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.

7.11.1 Detailed Description

Namespace of shape accumulators.

7.12 mln::accu::stat Namespace Reference

Namespace of statistical accumulators.

Classes

- struct deviation

  Generic standard deviation accumulator class.

- 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.

7.12.1 Detailed Description

Namespace of statistical accumulators.

7.13 mln::algebra Namespace Reference

Namespace of algebraic structure.

Classes

- struct h_mat

  N-Dimensional matrix with homogeneous coordinates.

- struct 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 A x = 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).

7.13.1 Detailed Description

Namespace of algebraic structure.

7.13.2 Function Documentation

7.13.2.1 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.

Referenced by mln::geom::mesh_curvature().

7.13.2.2 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 A x = B after mln::algebra::ldlt_decomp.

Referenced by mln::geom::mesh_curvature().

7.13.2.3 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).

References mln::literal::zero.

7.13.2.4 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).

References vprod().

Referenced by mln::geom::mesh_corner_point_area(), mln::geom::mesh_curvature(), mln::geom::mesh_normal(), and vprod().

7.14 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 , 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 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 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 >
  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 >
  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 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 , 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 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 >
  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`.

7.14.1 Detailed Description

Namespace of arithmetic.

7.14.2 Function Documentation

7.14.2.1 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:

← ***lhs*** First operand image.

← ***rhs*** Second operand image.

Returns:

The result image.

Precondition:

`lhs.domain == rhs.domain`

References mln::data::transform().

7.14.2.2 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:

← ***lhs*** First operand image.

← ***rhs*** Second operand image.

→ ***output*** The result image.

Precondition:

`output.domain == lhs.domain == rhs.domain`

7.14.2.3 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:

← ***input*** The image.

← ***val*** The value.

→ ***output*** The result image.

Precondition:

`output.domain == input.domain`

References div_cst().

Referenced by div_cst().

7.14.2.4 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:

← ***lhs*** First operand image (subject to division).

↔ ***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`

References div_inplace().

Referenced by div_inplace().

7.14.2.5 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:

← ***lhs*** First operand image.

← ***rhs*** Second operand image.

Returns:

The result image.

Precondition:

`lhs.domain == rhs.domain`

References mln::initialize().

7.14.2.6 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:

↔ ***lhs*** First operand image.

← ***rhs*** Second operand image.

Precondition:

`rhs.domain == lhs.domain`

7.14.2.7 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:

← ***lhs*** First operand image.

← ***rhs*** Second operand image.

← ***f*** Function.

Returns:

The result image.

Precondition:

`lhs.domain == rhs.domain`

References mln::initialize().

7.14.2.8 template<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:

← ***lhs*** First operand image.

← ***rhs*** Second operand image.

Returns:

The result image.

Precondition:

`lhs.domain == rhs.domain`

Parameters:

← ***lhs*** First operand image.

← ***rhs*** Second operand image.

Returns:

The result image.

The free parameter `V` sets the destination value type.

Precondition:

`lhs.domain == rhs.domain`

References mln::initialize().

7.14.2.9 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:

← ***input*** The image.

← ***val*** The value.

← ***f*** Function.

Returns:

The result image.

Precondition:

`input.is_valid`

7.14.2.10 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:

← ***input*** The image.

← ***val*** The value.

Returns:

The result image.

Precondition:

`input.is_valid`

7.14.2.11 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:

↔ ***input*** The image.

← ***val*** The value.

Precondition:

`input.is_valid`

References minus_cst_inplace(), and minus_inplace().

Referenced by minus_cst_inplace().

7.14.2.12 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:

↔ ***lhs*** First operand image (subject to addition).

← ***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`

References minus_inplace().

Referenced by minus_cst_inplace(), and minus_inplace().

7.14.2.13 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:

← ***lhs*** First operand image.

← ***rhs*** Second operand image.

← ***f*** Function.

Returns:

The result image.

Precondition:

`lhs.domain == rhs.domain`

References mln::initialize().

7.14.2.14 template<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:

← ***lhs*** First operand image.

← ***rhs*** Second operand image.

Returns:

The result image.

Precondition:

`lhs.domain == rhs.domain`

Parameters:

← ***lhs*** First operand image.

← ***rhs*** Second operand image.

Returns:

The result image.

The free parameter `V` sets the destination value type.

Precondition:

`lhs.domain == rhs.domain`

References mln::initialize().

7.14.2.15 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:

← ***input*** The image.

← ***val*** The value.

← ***f*** Function.

Returns:

The result image.

Precondition:

`input.is_valid`

7.14.2.16 template<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:

← ***input*** The image.

← ***val*** The value.

Returns:

The result image.

Precondition:

`input.is_valid`

7.14.2.17 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:

↔ ***input*** The image.

← ***val*** The value.

Precondition:

`input.is_valid`

References plus_cst_inplace(), and plus_inplace().

Referenced by plus_cst_inplace().

7.14.2.18 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:

↔ ***lhs*** First operand image (subject to addition).

← ***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`

Referenced by plus_cst_inplace().

7.14.2.19 template<typename I > mln::trait::concrete< I >::ret mln::arith::revert (const Image< I > & *input*) [inline]

Point-wise reversion of image `input`.

Parameters:

← ***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))

References mln::initialize().

7.14.2.20 template<typename I > void mln::arith::revert_inplace (Image< I > & *input*) [inline]

Point-wise in-place reversion of image `input`.

Parameters:

↔ ***input*** The target image.

Precondition:

`input.is_valid`

It performs:

for all p of input.domain

input(p) = min + (max - input(p))

7.14.2.21 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:

← ***lhs*** First operand image.

← ***rhs*** Second operand image.

→ ***output*** The result image.

Precondition:

`output.domain == lhs.domain == rhs.domain`

7.14.2.22 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:

← ***input*** The image.

← ***val*** The value.

→ ***output*** The result image.

Precondition:

`output.domain == input.domain`

References times_cst().

Referenced by times_cst().

7.14.2.23 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:

← ***lhs*** First operand image (subject to addition).

↔ ***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`

References times_inplace().

Referenced by times_inplace().

7.15 mln::arith::impl Namespace Reference

Implementation namespace of arith namespace.

Namespaces

- namespace generic

  Generic implementation namespace of arith namespace.

7.15.1 Detailed Description

Implementation namespace of arith namespace.

7.16 mln::arith::impl::generic Namespace Reference

Generic implementation namespace of arith namespace.

7.16.1 Detailed Description

Generic implementation namespace of arith namespace.

7.17 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.*

7.17.1 Detailed Description

Namespace of "point-wise" expression tools.

7.17.2 Function Documentation

7.17.2.1 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:

← ***input*** The input image.

← ***fun*** The thresholding function, from value(I) to bool.

for_all(p), output(p) = fun(p)

Referenced by threshold().

7.17.2.2 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:

← ***input*** The input image.

← ***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.

References binarization().

7.18 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.

7.18.1 Detailed Description

Namespace of routines related to image virtual (outer) border.

7.18.2 Function Documentation

7.18.2.1 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:

↔ ***ima*** The image whose border is to be adjusted.

← ***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.

References get(), and resize().

7.18.2.2 template<typename I > void mln::border::duplicate (const Image< I > & *ima*) [inline]

Assign the virtual (outer) border of image `ima` with the duplicate of the inner border of this image.

Parameters:

↔ ***ima*** The image whose border is to be duplicated.

Precondition:

`ima` has to be initialized.

References get().

7.18.2.3 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:

↔ ***ima1*** The first image whose border is to be equalizeed.

↔ ***ima2*** The second image whose border is to be equalizeed.

← ***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.

References get().

7.18.2.4 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:

↔ ***ima*** The image whose border is to be filled.

← ***v*** The value to assign to all border pixels.

Precondition:

`ima` has to be initialized.

7.18.2.5 template<typename I > unsigned mln::border::find (const Image< I > & *ima*) [inline]

Find the virtual (outer) border thickness of image `ima`.

Parameters:

← ***ima*** The image.

Returns:

The border thickness (0 if there is no border).

Precondition:

`ima` has to be initialized.

7.18.2.6 template<typename I > unsigned mln::border::get (const Image< I > & *ima*) [inline]

Get the virtual (outer) border thickness of image *ima*.

Parameters:

← ***ima*** The image.

Returns:

The border thickness (0 if there is no border).

Precondition:

ima has to be initialized.

Referenced by adjust(), duplicate(), and equalize().

7.18.2.7 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:

↔ ***ima*** The image whose border is to be mirrored.

Precondition:

`ima` has to be initialized.

7.18.2.8 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:

↔ ***ima*** The image whose border is to be resized.

← ***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.

References mln::primary(), and resize().

Referenced by adjust(), and resize().

7.19 mln::border::impl Namespace Reference

Implementation namespace of border namespace.

Namespaces

- namespace generic

  Generic implementation namespace of border namespace.

7.19.1 Detailed Description

Implementation namespace of border namespace.

7.20 mln::border::impl::generic Namespace Reference

Generic implementation namespace of border namespace.

7.20.1 Detailed Description

Generic implementation namespace of border namespace.

7.21 mln::canvas Namespace Reference

Namespace of canvas.

Namespaces

- namespace browsing

  Namespace of browsing canvas.

- namespace impl

  Implementation namespace of canvas namespace.

- 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.

7.21.1 Detailed Description

Namespace of canvas.

7.21.2 Function Documentation

7.21.2.1 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.

Referenced by mln::transform::influence_zone_front().

7.21.2.2 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.

Referenced by mln::transform::influence_zone_geodesic().

7.22 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

  Multidimentional Browsing in a given-way.

- struct snake_vert_t

  Browsing in a snake-way, forward.

7.22.1 Detailed Description

Namespace of browsing canvas.

7.23 mln::canvas::impl Namespace Reference

Implementation namespace of canvas namespace.

7.23.1 Detailed Description

Implementation namespace of canvas namespace.

7.24 mln::canvas::morpho Namespace Reference

Namespace of morphological canvas.

7.24.1 Detailed Description

Namespace of morphological canvas.

7.25 mln::convert Namespace Reference

Namespace of conversion routines.

Functions

- template<typename V >
  void from_to (const unsigned &from, Value< V > &to)

  Conversion of an unsigned `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 float &from, Value< V > &to)

  Conversion of a float `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 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 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 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 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 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 R , typename A >
  fun::C< R(∗)(A)> to_fun (R(∗f)(A))

  Convert a C unary function into an mln::fun::C.

- template<typename T >
  image1d< unsigned > to_image (const histo::array< T > &h)

  Convert an histo `h` *into an image1d<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 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 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 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 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 W >
  p_set< typename W::psite > to_p_set (const Window< W > &win)

  Convert a Window `win` *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 N >
  p_set< typename N::psite > to_p_set (const Neighborhood< N > &nbh)

  Convert a neighborhood `nbh` *into a site set.*

- 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 S >
  window< typename S::site::dpsite > to_window (const Site_Set< S > &pset)

  Convert a site set `pset` *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.*

7.25.1 Detailed Description

Namespace of conversion routines.

7.25.2 Function Documentation

7.25.2.1 template<typename V > void mln::convert::from_to (const unsigned & *from*, Value< V > & *to*) [inline]

Conversion of an unsigned `from` towards a value `to`.

7.25.2.2 template<typename V > void mln::convert::from_to (const int & *from*, Value< V > & *to*) [inline]

Conversion of a int `from` towards a value `to`.

7.25.2.3 template<typename V > void mln::convert::from_to (const float & *from*, Value< V > & *to*) [inline]

Conversion of a float `from` towards a value `to`.

7.25.2.4 template<typename V > void mln::convert::from_to (const double & *from*, Value< V > & *to*) [inline]

Conversion of a double `from` towards a value `to`.

7.25.2.5 template<typename N > mln::convert::mln_image_from_grid (typename N::site::grid, bool) const [inline]

Convert a neighborhood `nbh` into a binary image.

7.25.2.6 template<typename W > mln::convert::mln_image_from_grid (typename W::site::grid, mln_weight(W)) const [inline]

Convert a weighted window `w_win` into an image.

7.25.2.7 template<typename W > mln::convert::mln_image_from_grid (typename W::site::grid, bool) const [inline]

Convert a window `win` into a binary image.

7.25.2.8 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`.

7.25.2.9 template<typename N > mln::convert::mln_window (N) const [inline]

Convert a neighborhood `nbh` into a window.

7.25.2.10 template<typename T , typename O > T mln::convert::to (const O & *from*) [inline]

Conversion of the object `from` towards an object with type `T`.

References mln::mln_exact().

Referenced by mln::make_debug_graph_image().

7.25.2.11 template<typename P > P::dpoint mln::convert::to_dpoint (const Point_Site< P > & *p*) [inline]

Convert a point site `p` into a delta-point.

7.25.2.12 template<typename I > pw::value_< I > mln::convert::to_fun (const Image< I > & *ima*) [inline]

Convert an image into a function.

7.25.2.13 template<typename R , typename A > fun::C< R(*)(A)> mln::convert::to_fun (R(*)(A) *f*) [inline]

Convert a C unary function into an mln::fun::C.

7.25.2.14 template<typename T > image1d<unsigned> mln::convert::to_image (const histo::array< T > & *h*) [inline]

Convert an histo `h` into an image1d<unsigned>.

7.25.2.15 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.

References mln::p_array< P >::append().

7.25.2.16 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).

7.25.2.17 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).

7.25.2.18 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.

References mln::p_set< P >::insert().

7.25.2.19 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.

References mln::p_set< P >::insert().

7.25.2.20 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.

References mln::p_set< P >::insert().

7.25.2.21 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.

References mln::p_set< P >::insert().

7.25.2.22 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.

References mln::p_set< P >::insert().

7.25.2.23 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.

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

7.25.2.24 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.

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

7.25.2.25 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.

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

7.25.2.26 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.

References to_window().

7.25.2.27 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.

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

Referenced by to_window().

7.26 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 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)

  Generic implementation of data::stretch.

- 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)

  Generic implementation of data::transform.

- 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 >
  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.

- 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`.

7.26.1 Detailed Description

Namespace of image processing routines related to pixel data.

7.26.2 Function Documentation

7.26.2.1 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:

← ***input*** The input image.

→ ***output*** The output image.

References transform().

7.26.2.2 template<typename I > void mln::data::abs_inplace (Image< I > & *input*) [inline]

Apply the absolute value (abs) function to image pixel values.

Parameters:

↔ ***input*** The input image.

References apply().

7.26.2.3 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:

↔ ***input*** The input image.

← ***f*** The function-object.

This routine runs:

for all p of `input`, `input(p) = f( input(p) )`

This routine is equivalent to data::tranform(input, f, input) but it is faster since a single iterator is required.

Referenced by abs_inplace(), and saturate_inplace().

7.26.2.4 template<typename A , typename I > A::result mln::data::compute (const Accumulator< A > &, const Image< I > & *input_*) [inline]

Compute an accumulator onto the pixel values of the image `input`.

Parameters:

← ***a*** An accumulator.

← ***input*** The input image.

Returns:

The accumulator result.

It fully relies on data::update.

Compute an accumulator onto the pixel values of the image `input`.

Parameters:

← ***input*** The input image.

← ***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().

Referenced by mln::estim::mean(), mln::estim::min_max(), and mln::estim::sum().

7.26.2.5 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:

← ***v*** A value of the destination type.

← ***input*** The input image.

References mln::data::impl::generic::transform().

Referenced by mln::morpho::watershed::superpose(), and mln::debug::superpose().

7.26.2.6 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:

← ***input*** The image to be filtered.

← ***win*** The window.

↔ ***output*** The output image.

Precondition:

`input` and `output` have to be initialized.

7.26.2.7 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:

↔ ***ima*** The image to be filled.

← ***data*** The auxiliary data to fill the image `ima`.

Precondition:

`ima` has to be initialized.

Referenced by mln::topo::detach(), mln::util::display_branch(), mln::transform::distance_and_closest_-point_geodesic(), mln::duplicate(), mln::labeling::fill_holes(), mln::morpho::tree::filter::filter(), mln::morpho::impl::generic::hit_or_miss(), mln::transform::hough(), mln::registration::icp(), 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::slices_2d(), mln::morpho::watershed::superpose(), and mln::debug::superpose().

7.26.2.8 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:

↔ ***ima*** The image to be filled.

← ***data*** The image.

Warning:

The definition domain of `ima` has to be included in the one of `data`.

Precondition:

`ima.domain <= data.domain`.

Fill the image `ima` with the values of the image `data`.

Parameters:

↔ ***ima_*** The image to be filled.

← ***data_*** The image.

7.26.2.9 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:

↔ ***ima*** The image to be filled.

← ***val*** The value to assign to all sites.

Precondition:

ima has to be initialized.

Parameters:

↔ ***ima_*** The image to be filled.

← ***val*** The value to assign to all sites.

Precondition:

ima has to be initialized.

Referenced by mln::p_image< I >::clear().

7.26.2.10 template<typename I , typename W > mln::trait::concrete< I >::ret mln::data::median (const Image< I > & *input*, const Window< W > & *win*) [inline]

Compute in output the median filter of image input by the window win.

Parameters:

← ***input*** The image to be filtered.

← ***win*** The window.

Precondition:

input have to be initialized.

References mln::extension::adjust(), and mln::initialize().

7.26.2.11 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:

← ***a*** A meta-accumulator.

← ***input*** The input image.

Returns:

The accumulator result.

7.26.2.12 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:

← ***input*** The input image providing pixels values.

↔ ***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`

Paste the contents of image `input` into the image `output`.

Parameters:

← ***input_*** The input image providing pixels values.

↔ ***output_*** The image in which values are assigned.

Referenced by mln::make::image3d(), mln::draw::line(), mln::geom::rotate(), and mln::debug::slices_2d().

7.26.2.13 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:

← ***input*** The input image providing pixels values.

↔ ***output*** The image in which values are assigned.

7.26.2.14 template<typename I > void mln::data::replace (Image< I > & *input*, const typename I::value & *old_value*, const typename I::value & *new_value*) [inline]

Replace `old_value` by `new_value` in the image `input`

Parameters:

← ***input*** The input image.

← ***old_value*** The value to be replaced...

← ***new_value*** ...by this one.

7.26.2.15 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:

← ***input*** The input image.

← ***min*** The minimum output value.

← ***max*** The maximum output value.

References transform().

7.26.2.16 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:

← ***v*** A value of the output type.

← ***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.

References transform().

7.26.2.17 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:

↔ ***input*** The input image.

← ***min*** The minimum output value.

← ***max*** The maximum output value

References apply().

7.26.2.18 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.

References mln::util::array< T >::append(), and mln::util::array< T >::reserve().

7.26.2.19 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:

← ***input*** The input image.

Precondition:

`input.is_valid`

7.26.2.20 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:

← ***input*** The input image.

Precondition:

`input.is_valid`

7.26.2.21 template<typename V , typename I > mln::trait::ch_value< I, V >::ret mln::data::stretch (const V & *v*, const Image< I > & *input*) [inline]

Generic implementation of data::stretch.

Stretch the values of `input` so that they can be stored in `output`.

Parameters:

← ***v*** A value to set the output value type.

← ***input*** The input image.

Returns:

A stretch image with values of the same type as `v`.

Precondition:

input.is_valid

Parameters:

← ***v*** A value to set the output value type.

← ***input*** The input image.

Returns:

A stretch image with values of the same type as `v`.

References mln::initialize(), mln::estim::min_max(), mln::data::impl::stretch(), and transform().

Referenced by stretch().

7.26.2.22 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:

← ***input*** The input image.

→ ***output*** The result image.

Precondition:

`output.domain >= input.domain`

References transform().

7.26.2.23 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]

Generic implementation of data::transform.

Transform two images `input1` `input2` through a function `f`.

Parameters:

← ***input1*** The 1st input image.

← ***input2*** The 2nd input image.

← ***f*** The function.

This routine runs:

for all p of `input`, `output(p) = f( input1(p), input2(p) )`.

Parameters:

← ***input1_*** The 1st input image.

← ***input2_*** The 2nd input image.

← ***f_*** The function.

References mln::initialize().

7.26.2.24 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]

Generic implementation of data::transform.

Transform the image `input` through a function `f`.

Parameters:

← ***input*** The input image.

← ***f*** The function.

This routine runs:

for all p of `input`, `output(p) = f( input(p) )`.

Parameters:

← ***input_*** The input image.

← ***f_*** The function.

References mln::initialize().

Referenced by abs(), mln::logical::and_not(), mln::labeling::colorize(), mln::data::impl::generic::convert(), mln::arith::diff_abs(), mln::linear::mln_ch_convolve_grad(), mln::labeling::pack(), mln::labeling::pack_inplace(), mln::labeling::relabel(), saturate(), mln::data::impl::stretch(), to_enc(), and mln::labeling::wrap().

7.26.2.25 template<typename I1 , typename I2 , typename F > void mln::data::transform_inplace (Image< I1 > & *ima_*, const Image< I2 > & *aux_*, const Function_vv2v< F > & *f_*) [inline]

Generic implementation of transform_inplace.

Transform inplace the image `ima` with the image `aux` through a function `f`.

Parameters:

← ***ima*** The image to be transformed.

← ***aux*** The auxiliary image.

← ***f*** The function.

This routine runs:

for all p of `ima`, `ima(p) = f( ima(p), aux(p) )`.

Parameters:

← ***ima_*** The image to be transformed.

← ***aux_*** The auxiliary image.

← ***f_*** The function.

7.26.2.26 template<typename I , typename F > void mln::data::transform_inplace (Image< I > & *ima_*, const Function_v2v< F > & *f_*) [inline]

Generic implementation of transform_inplace.

Transform inplace the image `ima` through a function `f`.

Parameters:

↔ ***ima*** The image to be transformed.

← ***f*** The function.

This routine runs:

for all p of `ima`, `ima(p) = f( ima(p) )`.

Parameters:

↔ ***ima_*** The image to be transformed.

← ***f_*** The function.

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().

7.26.2.27 template<typename A , typename I > A::result mln::data::update (Accumulator< A > & *a_*, const Image< I > & *input_*) [inline]

Generic implementation of data::update.

Update an accumulator with the pixel values of the image `input`.

Parameters:

← ***a*** The accumulator.

← ***input*** The input image.

Returns:

The accumulator result.

Parameters:

← ***a_*** The accumulator.

← ***input_*** The input image.

Returns:

The accumulator result.

7.27 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::octagon2d &win)
- template<typename I >
  mln::trait::concrete< I >::ret median (const Image< I > &input, const win::disk2d &win)
- template<typename I >
  mln::trait::concrete< I >::ret median (const Image< I > &input, const win::rectangle2d &win)

7.27.1 Detailed Description

Namespace of image processing routines related to pixel levels with approximation.

7.27.2 Function Documentation

7.27.2.1 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:

← ***input*** The image to be filtered.

← ***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.

References median().

7.27.2.2 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:

← ***input*** The image to be filtered.

← ***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.

References median().

7.27.2.3 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:

← ***input*** The image to be filtered.

← ***win*** The rectangle.

The approximation is based on a vertical median ran after an horizontal median.

Precondition:

`input` and `output` have to be initialized.

Referenced by median().

7.28 mln::data::approx::impl Namespace Reference

Implementation namespace of data::approx namespace.

7.28.1 Detailed Description

Implementation namespace of data::approx namespace.

7.29 mln::data::impl Namespace Reference

Implementation namespace of data namespace.

Namespaces

- namespace generic

  Generic implementation namespace of data namespace.

Functions

- 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.

7.29.1 Detailed Description

Implementation namespace of data namespace.

7.29.2 Function Documentation

7.29.2.1 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:

← ***v*** A value to set the output value type.

← ***input*** The input image.

Returns:

A stretch image with values of the same type as v.

References mln::initialize(), mln::estim::min_max(), stretch(), and mln::data::transform().

Referenced by mln::data::stretch().

7.29.2.2 template<typename I , typename F > void mln::data::impl::transform_inplace_lowq (Image< I > & *input_*, const Function_v2v< F > & *f_*) [inline]

Specialized implementation.

7.29.2.3 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:

← ***a_*** The accumulator.

← ***input_*** The input image.

Returns:

The accumulator result.

7.30 mln::data::impl::generic Namespace Reference

Generic implementation namespace of data namespace.

Functions

- 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 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 W >
  mln::trait::concrete< I >::ret median (const Image< I > &input, const Window< W > &win)
- template<typename I , typename J >
  void paste (const Image< I > &input_, Image< J > &output_)

  Generic implementation of data::paste.

- 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 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 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 >
  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.

7.30.1 Detailed Description

Generic implementation namespace of data namespace.

7.30.2 Function Documentation

7.30.2.1 template<typename V , typename I > mln::trait::ch_value< I , V >::ret mln::data::impl::generic::convert (const V & *v*, const Image< I > & *input*) [inline]

Convert the image `input` by changing the value type.

Parameters:

← ***v*** A value of the destination type.

← ***input*** The input image.

References transform().

Referenced by mln::morpho::watershed::superpose(), and mln::debug::superpose().

7.30.2.2 template<typename I , typename J > void mln::data::impl::generic::fill_with_image (Image< I > & *ima_*, const Image< J > & *data_*) [inline]

Generic implementation.

Fill the image `ima` with the values of the image `data`.

Parameters:

↔ ***ima_*** The image to be filled.

← ***data_*** The image.

7.30.2.3 template<typename I , typename V > void mln::data::impl::generic::fill_with_value (Image< I > & *ima_*, const V & *val*) [inline]

Fill the whole image `ima` with the single value `v`.

Parameters:

↔ ***ima_*** The image to be filled.

← ***val*** The value to assign to all sites.

Precondition:

`ima` has to be initialized.

Referenced by mln::p_image< I >::clear().

7.30.2.4 template<typename I , typename W > mln::trait::concrete< I >::ret mln::data::impl::generic::median (const Image< I > & *input*, const Window< W > & *win*) [inline]

Compute in `output` the median filter of image `input` by the window `win`.

Parameters:

← ***input*** The image to be filtered.

← ***win*** The window.

Precondition:

`input` have to be initialized.

References mln::extension::adjust(), and mln::initialize().

7.30.2.5 template<typename I , typename J > void mln::data::impl::generic::paste (const Image< I > & *input_*, Image< J > & *output_*) [inline]

Generic implementation of data::paste.

Paste the contents of image `input` into the image `output`.

Parameters:

← ***input_*** The input image providing pixels values.

↔ ***output_*** The image in which values are assigned.

Referenced by mln::make::image3d(), mln::draw::line(), mln::geom::rotate(), and mln::debug::slices_2d().

7.30.2.6 template<typename I > util::array<unsigned> mln::data::impl::generic::sort_offsets_increasing (const Image< I > & *input_*) [inline]

Sort pixel offsets of the image `input` wrt increasing pixel values.

References mln::util::array< T >::append(), and mln::util::array< T >::reserve().

7.30.2.7 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_*) [inline]

Generic implementation of data::transform.

Parameters:

← ***input1_*** The 1st input image.

← ***input2_*** The 2nd input image.

← ***f_*** The function.

References mln::initialize().

7.30.2.8 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_*) [inline]

Generic implementation of data::transform.

Parameters:

← ***input_*** The input image.

← ***f_*** The function.

References mln::initialize().

Referenced by mln::data::abs(), mln::logical::and_not(), mln::labeling::colorize(), convert(), mln::arith::diff_abs(), mln::linear::mln_ch_convolve_grad(), mln::labeling::pack(), mln::labeling::pack_inplace(), mln::labeling::relabel(), mln::data::saturate(), mln::data::impl::stretch(), mln::data::to_enc(), and mln::labeling::wrap().

7.30.2.9 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_*) [inline]

Generic implementation of transform_inplace.

Parameters:

← ***ima_*** The image to be transformed.

← ***aux_*** The auxiliary image.

← ***f_*** The function.

7.30.2.10 template<typename I , typename F > void mln::data::impl::generic::transform_inplace (Image< I > & *ima_*, const Function_v2v< F > & *f_*) [inline]

Generic implementation of transform_inplace.

Parameters:

↔ ***ima_*** The image to be transformed.

← ***f_*** The function.

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().

7.30.2.11 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:

← ***a_*** The accumulator.

← ***input_*** The input image.

Returns:

The accumulator result.

7.31 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`.

7.31.1 Detailed Description

Namespace of image processing routines related to pixel levels with naive approach.

7.31.2 Function Documentation

7.31.2.1 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:

← ***input*** The image to be filtered.

← ***win*** The window.

↔ ***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

7.32 mln::data::naive::impl Namespace Reference

Implementation namespace of data::naive namespace.

7.32.1 Detailed Description

Implementation namespace of data::naive namespace.

7.33 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 , 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`.

- 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 >
  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.*

- std::string filename (const std::string &filename, int postfix_id)

  Constructs and returns a formatted output file name.

- unsigned short format (unsigned char v)

  Format an unsigned char to print it properly, i.e., like an integer value.

- signed short format (signed char v)

  Format a signed char to print it properly, i.e., like an integer value.

- char format (bool v)

  Format a Boolean to print it nicely: "|" for true and "-" for false.

- template<typename T >
  const T & format (const T &v)

  Default version for formatting a value is a no-op.

- template<typename I >
  void iota (Image< I > &input)
- 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, float ratio_hv, 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, unsigned n_horizontal, unsigned n_vertical, 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.

7.33.1 Detailed Description

Namespace of routines that help to debug.

7.33.2 Function Documentation

7.33.2.1 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 though `ecolor_f_`.

References mln::p_line2d::begin(), mln::p_line2d::end(), mln::p_vertices< G, F >::graph(), and mln::draw::line().

7.33.2.2 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_.

References mln::p_vertices< G, F >::graph(), and mln::draw::line().

7.33.2.3 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.

References mln::p_vertices< G, F >::graph(), and mln::draw::line().

Referenced by mln::make_debug_graph_image().

7.33.2.4 std::string mln::debug::filename (const std::string & *filename*, int *postfix_id* = -1) [inline]

Constructs and returns a formatted output file name.

The file name is formatted as follow:

'filename_prefix'_'id'_'filename'_'postfix_id'

Where:

- 'filename_prefix' can be set through the global variable debug::internal::filename_prefix.
- 'id' is auto-incremented and cannot be controlled.
- 'filename' is the given filename
- 'postfix_id' is an optional counter which can be controlled contrary to 'id'.

7.33.2.5 unsigned short mln::debug::format (unsigned char *v*) [inline]

Format an unsigned char to print it properly, i.e., like an integer value.

7.33.2.6 signed short mln::debug::format (signed char *v*) [inline]

Format a signed char to print it properly, i.e., like an integer value.

7.33.2.7 char mln::debug::format (bool *v*) [inline]

Format a Boolean to print it nicely: "|" for true and "-" for false.

7.33.2.8 template<typename T > const T & mln::debug::format (const T & *v*) [inline]

Default version for formatting a value is a no-op.

Referenced by mln::value::operator<<(), and mln::Gpoint< E >::operator<<().

7.33.2.9 template<typename I > void mln::debug::iota (Image< I > & *input*) [inline]

Fill the image `input` with successive values.

Parameters:

↔ ***input*** The image in which values are assigned.

7.33.2.10 template<typename I > void mln::debug::println (const std::string & *msg*, const Image< I > & *input*) [inline]

Print the message `msg` and the image `input` on the standard output.

References println().

7.33.2.11 template<typename I > void mln::debug::println (const Image< I > & *input*) [inline]

Print the image `input` on the standard output.

References mln::geom::bbox().

Referenced by println().

7.33.2.12 template<typename I > void mln::debug::println_with_border (const Image< I > & *input*) [inline]

Print the image `input` on the standard output.

References mln::geom::bbox().

7.33.2.13 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`.

References mln::image2d< T >::has(), and mln::point< G, C >::last_coord().

7.33.2.14 template<typename I > image2d< typename I::value > mln::debug::slices_2d (const Image< I > & *input*, float *ratio_hv*, const typename I::value & *bg*) [inline]

Create a 2D image of the slices of the 3D image `input`.

References slices_2d().

7.33.2.15 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`.

References mln::apply_p2p(), mln::data::fill(), and mln::data::paste().

Referenced by slices_2d().

7.33.2.16 template<typename I , typename J > mln::trait::ch_value< I, value::rgb8 >::ret mln::debug::superpose (const Image< I > & *input*, const Image< J > & *object*) [inline]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

References mln::literal::red, and superpose().

7.33.2.17 template<typename I , typename J > mln::trait::ch_value< I, value::rgb8 >::ret mln::debug::superpose (const Image< I > & *input_*, const Image< J > & *object_*, const value::rgb8 & *object_color*) [inline]

Superpose two images.

Parameters:

← ***input_*** An image. Its value type must be convertible toward value::rgb8 thanks to a conversion operator or convert::from_to.

← ***object_*** A binary image. Objects used for superposition are set to 'true'.

← ***object_color*** The color used to draw the objects in `object_`.

Precondition:

`input_` and `object_` must have the same domain.

Returns:

A color image.

References mln::data::convert(), and mln::data::fill().

Referenced by superpose().

7.34 mln::debug::impl Namespace Reference

Implementation namespace of debug namespace.

7.34.1 Detailed Description

Implementation namespace of debug namespace.

7.35 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.

7.35.1 Detailed Description

Namespace for core definitions.

7.35.2 Typedef Documentation

7.35.2.1 typedef short mln::def::coord

Definition of the default coordinate type: 'short'.

7.35.2.2 typedef float mln::def::coordf

Definition of the floating coordinate type.

7.35.3 Enumeration Type Documentation

7.35.3.1 anonymous enum

Definition of the number of bits of the low quantization threshold.

7.36 mln::display Namespace Reference

Namespace of routines that help to display images.

Namespaces

- namespace impl

  Implementation namespace of display namespace.

7.36.1 Detailed Description

Namespace of routines that help to display images.

7.37 mln::display::impl Namespace Reference

Implementation namespace of display namespace.

Namespaces

- namespace generic

  Generic implementation namespace of display namespace.

7.37.1 Detailed Description

Implementation namespace of display namespace.

7.38 mln::display::impl::generic Namespace Reference

Generic implementation namespace of display namespace.

7.38.1 Detailed Description

Generic implementation namespace of display namespace.

7.39 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.

7.39.1 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.

7.40 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 >
  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)

7.40.1 Detailed Description

Namespace of drawing routines.

7.40.2 Function Documentation

7.40.2.1 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:

↔ ***ima*** The image to be drawn.

← ***b*** the boxto draw.

← ***v*** The value to assign to all drawn pixels.

Precondition:

`ima` has to be initialized.
`ima` has `beg`.
`ima` has `end`.

References line().

7.40.2.2 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:

↔ ***ima*** The image to be drawn.

← ***beg*** The start point to drawn line.

← ***end*** The end point to drawn line.

← ***v*** The value to assign to all drawn pixels.

Precondition:

`ima` has to be initialized.
`ima` has `beg`.
`ima` has `end`.

References mln::data::paste().

Referenced by box(), and mln::debug::draw_graph().

7.40.2.3 template<typename I > void mln::draw::plot (Image< I > & *ima*, const typename I::point & *p*, const typename I::value & *v*) [inline]

Plot a point at level `v` in image `ima`

Parameters:

↔ ***ima*** The image to be drawn.

← ***p*** The point to be plotted.

← ***v*** The value to assign to all drawn pixels.

Precondition:

`ima` has to be initialized.
`ima` has `p`.

7.41 mln::estim Namespace Reference

Namespace of estimation materials.

Functions

- 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 >
  mln::value::props< typename I::value >::sum mean (const Image< I > &input)

  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 , typename S >
  void sum (const Image< I > &input, S &result)

  Compute the sum value 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`.

7.41.1 Detailed Description

Namespace of estimation materials.

7.41.2 Function Documentation

7.41.2.1 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:

← ***input*** The image.

→ ***result*** The mean value.

The free parameter `S` is the type used to compute the summation.

References mln::data::compute().

7.41.2.2 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:

← ***input*** The image.

Returns:

The mean value.

References mln::data::compute().

7.41.2.3 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:

← ***input*** The image.
→ ***min*** The minimum pixel value of `input`.
→ ***max*** The maximum pixel value of `input`.

References mln::data::compute().

Referenced by mln::data::impl::stretch(), and mln::make::voronoi().

7.41.2.4 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:

← ***input*** The image.
→ ***result*** The sum value.

References mln::data::compute().

7.41.2.5 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:

← ***input*** The image.

Returns:

The sum value.

References mln::data::compute().

7.42 mln::extension Namespace Reference

Namespace of extension tools.

Functions

- 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 (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 , 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_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)

7.42.1 Detailed Description

Namespace of extension tools.

7.42.2 Function Documentation

7.42.2.1 template<typename I > void mln::extension::adjust (const Image< I > & *ima*, unsigned *delta*) [inline]

Adjust the domain extension of image `ima` with the size `delta`.

References adjust().

7.42.2.2 template<typename I , typename N > void mln::extension::adjust (const Image< I > & *ima*, const Neighborhood< N > & *nbh*) [inline]

Adjust the domain extension of image ima with the size of the neighborhood nbh.

References adjust(), and mln::geom::delta().

7.42.2.3 template<typename I , typename W > void mln::extension::adjust (const Image< I > & *ima*, const Weighted_Window< W > & *wwin*) [inline]

Adjust the domain extension of image ima with the size of the weighted window wwin.

References adjust(), and mln::geom::delta().

7.42.2.4 template<typename I , typename W > void mln::extension::adjust (const Image< I > & *ima*, const Window< W > & *win*) [inline]

Adjust the domain extension of image ima with the size of the window win.

References mln::geom::delta().

Referenced by adjust(), adjust_duplicate(), adjust_fill(), and mln::data::impl::generic::median().

7.42.2.5 template<typename I , typename W > void mln::extension::adjust_duplicate (const Image< I > & *ima*, const Window< W > & *win*) [inline]

Adjust then duplicate.

References adjust(), and duplicate().

7.42.2.6 template<typename I , typename W > void mln::extension::adjust_fill (const Image< I > & *ima*, const Window< W > & *win*, const typename I::value & *val*) [inline]

Adjust then fill.

References adjust(), and fill().

Referenced by mln::morpho::impl::generic::rank_filter().

7.42.2.7 template<typename I > void mln::extension::duplicate (const Image< I > & *ima*) [inline]

Assign the contents of the domain extension by duplicating the values of the inner boundary of image ima.

Referenced by adjust_duplicate().

7.42.2.8 template<typename I > void mln::extension::fill (const Image< I > & *ima*, const typename I::value & *val*) [inline]

Fill the domain extension of image ima with the single value v.

Parameters:

↔ ***ima*** The image whose domain extension is to be filled.

← ***val*** The value to assign.

Precondition:

`ima` has to be initialized.

Referenced by adjust_fill().

7.43 mln::fun Namespace Reference

Namespace of functions.

Namespaces

- namespace access

  Namespace for access functions.

- namespace i2v

  Namespace of integer-to-value functions.

- 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.

7.43.1 Detailed Description

Namespace of functions.

Forward declarations.

fun::i2v::array

Forward declaration.

7.44 mln::fun::access Namespace Reference

Namespace for access functions.

7.44.1 Detailed Description

Namespace for access functions.

7.45 mln::fun::i2v Namespace Reference

Namespace of integer-to-value functions.

7.45.1 Detailed Description

Namespace of integer-to-value functions.

7.46 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`.

7.46.1 Detailed Description

Namespace of functions from point to boolean.

7.47 mln::fun::p2p Namespace Reference

Namespace of functions from grid point to grid point.

7.47.1 Detailed Description

Namespace of functions from grid point to grid point.

7.48 mln::fun::p2v Namespace Reference

Namespace of functions from point to value.

7.48.1 Detailed Description

Namespace of functions from point to value.

7.49 mln::fun::stat Namespace Reference

Namespace of statistical functions.

7.49.1 Detailed Description

Namespace of statistical functions.

7.50 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.

7.50.1 Detailed Description

Namespace of functions from value to logic value.

7.51 mln::fun::v2i Namespace Reference

Namespace of value-to-integer functions.

7.51.1 Detailed Description

Namespace of value-to-integer functions.

7.52 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.

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.

7.52.1 Detailed Description

Namespace of functions from value to value.

7.52.2 Variable Documentation

7.52.2.1 f_hsi_to_rgb_3x8_t mln::fun::v2v::f_hsi_to_rgb_3x8

Global variable.

7.52.2.2 f_hsl_to_rgb_3x8_t mln::fun::v2v::f_hsl_to_rgb_3x8

Global variables.

7.52.2.3 f_rgb_to_hsi_f_t mln::fun::v2v::f_rgb_to_hsi_f

Global variables.

7.52.2.4 f_rgb_to_hsl_f_t mln::fun::v2v::f_rgb_to_hsl_f

Global variables.

7.53 mln::fun::v2w2v Namespace Reference

Namespace of bijective functions.

Classes

- struct cos

  Cosinus bijective functor.

7.53.1 Detailed Description

Namespace of bijective functions.

7.54 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.

7.54.1 Detailed Description

Namespace of functions from value to value.

7.55 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.

7.55.1 Detailed Description

Namespace of functions from value to value.

7.56 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.

7.56.1 Detailed Description

Namespace of functions from a couple of values to a value.

7.57 mln::fun::x2p Namespace Reference

Namespace of functions from point to value.

Classes

- struct closest_point

  FIXME: doxygen + concept checking.

7.57.1 Detailed Description

Namespace of functions from point to value.

7.58 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.

7.58.1 Detailed Description

Namespace of functions from vector to value.

7.59 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.

7.59.1 Detailed Description

Namespace of functions from vector to vector.

7.60 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 W >
  box< typename W::psite > bbox (const Weighted_Window< W > &win)

  Compute the precise bounding box of a weighted window `win`.

- template<typename W >
  box< typename W::psite > bbox (const Window< W > &win)

  Compute the precise bounding box of a window `win`.

- template<typename I >
  box< typename I::site > bbox (const Image< I > &ima)

  Compute the precise bounding box of a point set `pset`.

- 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 , 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 N >
  unsigned delta (const Neighborhood< N > &nbh)

  Compute the delta of a neighborhood `nbh`.

- template<typename W >
  unsigned delta (const Weighted_Window< W > &wwin)

  Compute the delta of a weighted window `wwin`.

- template<typename W >
  unsigned delta (const Window< W > &win)

Compute the delta of a window win.

- template<typename B >
  B::point::coord max_col (const Box< B > &b)

  Give the maximum col of an box 2d or 3d.

- template<typename I >
  I::site::coord max_col (const Image< I > &ima)

  Give the maximum column of an image.

- template<typename I >
  I::site::coord max_ind (const Image< I > &ima)

  Give the maximum ind of an image.

- template<typename B >
  B::point::coord max_row (const Box< B > &b)

  Give the maximum row of an box 2d or 3d.

- template<typename I >
  I::site::coord max_row (const Image< I > &ima)

  Give the maximum row of an image.

- 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 B >
  B::point::coord min_col (const Box< B > &b)

  Give the minimum column of an box 2d or 3d.

- template<typename I >
  I::site::coord min_col (const Image< I > &ima)

  Give the minimum column of an image.

- template<typename I >
  I::site::coord min_ind (const Image< I > &ima)

  Give the minimum ind of an image.

- template<typename B >
  B::point::coord min_row (const Box< B > &b)

  Give the minimum row of an box 2d or 3d.

- template<typename I >
  I::site::coord min_row (const Image< I > &ima)

  Give the minimum row of an image.

- template<typename I >
  I::site::coord min_sli (const Image< I > &ima)

  Give the minimum sli 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 ncols (const Image< I > &ima)

  Give the number of columns of an image.

- template<typename I >
  unsigned ninds (const Image< I > &ima)

  Give the number of inds 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 nrows (const Image< I > &ima)

  Give the number of rows of an image.

- 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 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 >
  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 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 >
  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 I , typename Ext >
  mln::trait::concrete< I >::ret rotate (const Image< I > &input, double angle, const Ext &extension)
- 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 , 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 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.*

7.60.1 Detailed Description

Namespace of all things related to geometry.

Namespace of essential things related to geometry.

7.60.2 Function Documentation

7.60.2.1 template<typename W > box< typename W::psite > mln::geom::bbox (const Weighted_Window< W > & *win*) [inline]

Compute the precise bounding box of a weighted window `win`.

References bbox().

7.60.2.2 template<typename W > box< typename W::psite > mln::geom::bbox (const Window< W > & *win*) [inline]

Compute the precise bounding box of a window `win`.

References mln::literal::origin.

7.60.2.3 template<typename I > box< typename I::site > mln::geom::bbox (const Image< I > & *ima*) [inline]

Compute the precise bounding box of a point set `pset`.

References bbox().

7.60.2.4 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`.

Referenced by bbox(), mln::transform::distance_and_closest_point_geodesic(), mln::registration::icp(), max_col(), max_row(), min_col(), min_row(), mln::debug::println(), mln::debug::println_with_border(), and rotate().

7.60.2.5 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)`) [inline]

Apply chamfer algorithm to a binary image.

Referenced by mln::geom::impl::seeds2tiling_roundness().

7.60.2.6 template<typename N > unsigned mln::geom::delta (const Neighborhood< N > & *nbh*) [inline]

Compute the delta of a neighborhood `nbh`.

References delta().

7.60.2.7 template<typename W > unsigned mln::geom::delta (const Weighted_Window< W > & *wwin*) [inline]

Compute the delta of a weighted window `wwin`.

References delta().

7.60.2.8 template<typename W > unsigned mln::geom::delta (const Window< W > & *win*) [inline]

Compute the delta of a window `win`.

Referenced by mln::extension::adjust(), delta(), and mln::morpho::impl::generic::rank_filter().

7.60.2.9 template<typename B > B::point::coord mln::geom::max_col (const Box< B > & *b*) [inline]

Give the maximum col of an box 2d or 3d.

7.60.2.10 template<typename I > I::site::coord mln::geom::max_col (const Image< I > & *ima*) [inline]

Give the maximum column of an image.

References bbox().

Referenced by ncols().

7.60.2.11 template<typename I > I::site::coord mln::geom::max_ind (const Image< I > & *ima*) [inline]

Give the maximum ind of an image.

Referenced by ninds().

7.60.2.12 template<typename B > B::point::coord mln::geom::max_row (const Box< B > & *b*) [inline]

Give the maximum row of an box 2d or 3d.

7.60.2.13 template<typename I > I::site::coord mln::geom::max_row (const Image< I > & *ima*) [inline]

Give the maximum row of an image.

References bbox().

Referenced by nrows().

7.60.2.14 template<typename I > I::site::coord mln::geom::max_sli (const Image< I > & *ima*) [inline]

Give the maximum sli of an image.

Referenced by nslis().

7.60.2.15 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).”

References mln::data::fill(), mln::norm::sqr_l2(), mln::algebra::vprod(), and mln::literal::zero.

Referenced by mesh_curvature().

7.60.2.16 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 princpal 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:

← ***mesh*** The surface (triangle mesh) on which the curvature is to be computed.

References mln::c2(), mln::algebra::ldlt_decomp(), mln::algebra::ldlt_solve(), mesh_corner_point_area(), mesh_normal(), mln::algebra::vprod(), and mln::literal::zero.

7.60.2.17 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.

References mln::data::fill(), mln::norm::sqr_l2(), mln::algebra::vprod(), and mln::literal::zero.

Referenced by mesh_curvature().

7.60.2.18 template<typename B > B::point::coord mln::geom::min_col (const Box< B > & *b*) [inline]

Give the minimum column of an box 2d or 3d.

7.60.2.19 template<typename I > I::site::coord mln::geom::min_col (const Image< I > & *ima*) [inline]

Give the minimum column of an image.

References bbox().

Referenced by mln::transform::hough(), and ncols().

7.60.2.20 template<typename I > I::site::coord mln::geom::min_ind (const Image< I > & *ima*) [inline]

Give the minimum ind of an image.

Referenced by ninds().

7.60.2.21 template<typename B > B::point::coord mln::geom::min_row (const Box< B > & *b*) [inline]

Give the minimum row of an box 2d or 3d.

7.60.2.22 template<typename I > I::site::coord mln::geom::min_row (const Image< I > & *ima*) [inline]

Give the minimum row of an image.

References bbox().

Referenced by mln::transform::hough(), and nrows().

7.60.2.23 template<typename I > I::site::coord mln::geom::min_sli (const Image< I > & *ima*) [inline]

Give the minimum sli of an image.

Referenced by nslis().

7.60.2.24 template<typename B > unsigned mln::geom::ncols (const Box< B > & *b*) [inline]

Give the number of cols of a box 2d or 3d.

References max_col(), min_col(), and ncols().

7.60.2.25 template<typename I > unsigned mln::geom::ncols (const Image< I > & *ima*) [inline]

Give the number of columns of an image.

References max_col(), and min_col().

Referenced by mln::subsampling::gaussian_subsampling(), mln::transform::hough(), ncols(), and mln::subsampling::subsampling().

7.60.2.26 template<typename I > unsigned mln::geom::ninds (const Image< I > & *ima*) [inline]

Give the number of inds of an image.

References max_ind(), and min_ind().

7.60.2.27 template<typename B > unsigned mln::geom::nrows (const Box< B > & *b*) [inline]

Give the number of rows of a box 2d or 3d.

References max_row(), min_row(), and nrows().

7.60.2.28 template<typename I > unsigned mln::geom::nrows (const Image< I > & *ima*) [inline]

Give the number of rows of an image.

References max_row(), and min_row().

Referenced by mln::subsampling::gaussian_subsampling(), mln::transform::hough(), nrows(), and mln::subsampling::subsampling().

7.60.2.29 template<typename I > unsigned mln::geom::nsites (const Image< I > & *input*) [inline]

Compute the number of sites of the image `input`.

Referenced by pmin_pmax().

7.60.2.30 template<typename I > unsigned mln::geom::nslis (const Image< I > & *ima*) [inline]

Give the number of slices of an image.

References max_sli(), and min_sli().

7.60.2.31 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`.

7.60.2.32 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`.

References pmin_pmax().

7.60.2.33 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`.

References nsites().

7.60.2.34 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`.

References nsites().

Referenced by pmin_pmax().

7.60.2.35 template<typename I > mln::trait::concrete< I >::ret mln::geom::rotate (const Image< I > & *input*, double *angle*) [inline]

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.

References rotate(), and mln::literal::zero.

7.60.2.36 template<typename I , typename Ext > mln::trait::concrete< I >::ret mln::geom::rotate (const Image< I > & *input*, double *angle*, const Ext & *extension*) [inline]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

References rotate().

7.60.2.37 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*) [inline]

Perform a rotation from the center of an image.

Parameters:

← ***input*** An image.

← ***angle*** An angle in degrees.

← ***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.

← ***output_domain*** The domain of the output image. An invalid domain, causes the routine to use the rotated input_ domain.

Returns:

An image with the same domain as `input`.

References bbox(), mln::compose(), mln::extend(), mln::initialize(), mln::mln_exact(), mln::literal::origin, mln::data::paste(), and mln::accu::shape::bbox< P >::to_result().

Referenced by rotate().

7.60.2.38 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:

↔ ***ima_*** The labeled image with seed.

← ***nbh*** The neighborhood to use on this algorithm.

Returns:

A tiled image.

Precondition:

`ima_` has to be initialized.

Take a labeled image `ima_` with seeds and extend them until creating tiles.

Parameters:

↔ ***ima_*** The labeled image with seed.

← ***nbh_*** The neighborhood to use on this algorithm.

References mln::duplicate(), and mln::geom::impl::seeds2tiling().

Referenced by seeds2tiling().

7.60.2.39 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:

↔ ***ima_*** The labeled image with seed.

← ***w_win*** The weight window using by geom::chamfer to compute distance.

← ***max*** Unsigned using by geom::chamfer to compute the distance.

← ***nbh_*** The neighborhood to use on this algorithm.

Precondition:

`ima_` has to be initialized.

References chamfer(), mln::duplicate(), mln::p_priority< P, Q >::pop_front(), mln::p_priority< P, Q >::push(), mln::geom::impl::seeds2tiling_roundness(), and mln::literal::zero.

Referenced by seeds2tiling_roundness().

7.61 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.

- 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.*

7.61.1 Detailed Description

Implementation namespace of geom namespace.

7.61.2 Function Documentation

7.61.2.1 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.

Take a labeled image `ima_` with seeds and extend them until creating tiles.

Parameters:

↔ ***ima_*** The labeled image with seed.

← ***nbh_*** The neighborhood to use on this algorithm.

References mln::duplicate(), and seeds2tiling().

Referenced by mln::geom::seeds2tiling().

7.61.2.2 template<typename I , typename N > I mln::geom::impl::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:

↔ ***ima_*** The labeled image with seed.

← ***w_win*** The weight window using by geom::chamfer to compute distance.

← ***max*** Unsigned using by geom::chamfer to compute the distance.

← ***nbh_*** The neighborhood to use on this algorithm.

Precondition:

`ima_` has to be initialized.

References mln::geom::chamfer(), mln::duplicate(), mln::p_priority< P, Q >::pop_front(), mln::p_priority< P, Q >::push(), seeds2tiling_roundness(), and mln::literal::zero.

Referenced by mln::geom::seeds2tiling_roundness().

7.62 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.

7.62.1 Detailed Description

Namespace of graph related routines.

7.62.2 Function Documentation

7.62.2.1 template<typename G , typename F > F::result mln::graph::compute (const Graph< G > & *g_*, F & *functor*) [inline]

Base routine to compute attributes on a graph.

Parameters:

← ***g_*** A graph.

← ***functor*** A functor implementing the right interface.

Returns:

The computed data.

See also:

canvas::browsing::depth_first_search

7.62.2.2 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*) [inline]

Label graph components.

Vertex with id 0, usuallly used to represent the background component, will be labeled with an id different from 0. Therefore, the labeling starts from 1.

Parameters:

← ***graph_image_*** A graph image (

See also:

vertex_image, edge_image).

Parameters:

← ***nbh_*** A graph neighborhood.

↔ ***nlabels*** The number of labels found.

Returns:

a Graph image of labels.

References mln::labeling::blobs(), mln::data::fill(), and mln::initialize().

7.62.2.3 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_*) [inline]

Make a custom graph neighborhood from a mask image.

Parameters:

← ***graph_image_*** A graph image (

See also:

vertex_image and edge_image).

Parameters:

← ***graph_mask_image_*** A graph image of bool used as a mask.

Returns:

A masked neighborhood on graph.

7.62.2.4 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_*) [inline]

Make a custom graph window from a mask image.

Parameters:

← ***graph_image_*** A graph image (

See also:

vertex_image and edge_image).

Parameters:

← ***graph_mask_image_*** A graph image of bool used as a mask.

Returns:

A masked window on graph.

7.63 mln::grid Namespace Reference

Namespace of grids definitions.

7.63.1 Detailed Description

Namespace of grids definitions.

Compute the image::space trait from a point type.

7.64 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 >
  array< typename I::value > compute (const Image< I > &input)

  Compute the histogram of image `input`.

7.64.1 Detailed Description

Namespace of histograms.

7.64.2 Function Documentation

7.64.2.1 template<typename I > array< typename I::value > mln::histo::compute (const Image< I > & *input*) [inline]

Compute the histogram of image `input`.

7.65 mln::histo::impl Namespace Reference

Implementation namespace of histo namespace.

Namespaces

- namespace generic

  Generic implementation namespace of histo namespace.

7.65.1 Detailed Description

Implementation namespace of histo namespace.

7.66 mln::histo::impl::generic Namespace Reference

Generic implementation namespace of histo namespace.

7.66.1 Detailed Description

Generic implementation namespace of histo namespace.

7.67 mln::impl Namespace Reference

Implementation namespace of mln namespace.

7.67.1 Detailed Description

Implementation namespace of mln namespace.

7.68 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 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 tiff

  Namespace of tiff input/output handling.

- namespace txt

  Namespace of txt input/output handling.

7.68.1 Detailed Description

Namespace of input/output handling.

7.69 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.

7.69.1 Detailed Description

Namespace of cloud input/output handling.

7.69.2 Function Documentation

7.69.2.1 template<typename P > void mln::io::cloud::load (p_array< P > & *arr*, const std::string & *filename*) [inline]

Load a cloud of points.

Parameters:

↔ ***arr*** the site set where to load the data.

← ***filename*** file to load.

7.69.2.2 template<typename P > void mln::io::cloud::save (const p_array< P > & *arr*, const std::string & *filename*) [inline]

Load a cloud of points.

Parameters:

← ***arr*** the cloud of points to save.

← ***filename*** the destination.

7.70 mln::io::dicom Namespace Reference

Namespace of DICOM input/output handling.

Functions

- template<typename V >
  image2d< V > load (const std::string &filename)
  Load a fits image in a image2d<float>.

- template<typename I >
  void load (Image< I > &ima, const std::string &filename)
  Load a DICOM file in a Milena image.

7.70.1 Detailed Description

Namespace of DICOM input/output handling.

7.70.2 Function Documentation

7.70.2.1 template<typename V > image3d< V > mln::io::dicom::load (const std::string & *filename*) [inline]

Load a fits image in a image2d<float>.

Load a ppm image in a Milena image.

Load a pgm image in a Milena image.

Load a pfm image in a image2d<float>.

Load a pbm image in a image2d<float>.

Parameters:

←*filename* The image source.

Returns:

An image2d<float> which contains loaded data.

7.70.2.2 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:

→*ima* A reference to the image which will receive data.

←*filename* The source.

References mln::initialize(), and mln::point< G, C >::to_vec().

7.71 mln::io::dump Namespace Reference

Namespace of dump input/output handling.

Functions

- 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.

7.71.1 Detailed Description

Namespace of dump input/output handling.

7.71.2 Function Documentation

7.71.2.1 template<typename I > void mln::io::dump::load (Image< I > & *ima_*, const std::string & *filename*) [inline]

Load a Milena image by dumped into a file.

Parameters:

↔ ***ima_*** The image to load.

← ***filename*** the destination.

7.71.2.2 template<typename I > void mln::io::dump::save (const Image< I > & *ima_*, const std::string & *filename*) [inline]

Save a Milena image by dumping its data to a file.

Parameters:

← ***ima_*** The image to save.

← ***filename*** the destination.

7.72 mln::io::fits Namespace Reference

Namespace of fits input/output handling.

Functions

- image2d< float > load (const std::string &filename)
  Load a fits image in a image2d<float>.
- void load (image2d< float > &ima, const std::string &filename)
  Load a fits image in a Milena image.

7.72.1 Detailed Description

Namespace of fits input/output handling.

7.72.2 Function Documentation

7.72.2.1 image2d< float > mln::io::fits::load (const std::string & *filename*) [inline]

Load a fits image in a image2d<float>.

Parameters:

←*filename* The image source.

Returns:

An image2d<float> which contains loaded data.

7.72.2.2 void mln::io::fits::load (image2d< float > & *ima*, const std::string & *filename*) [inline]

Load a fits image in a Milena image.

Parameters:

→*ima* A reference to the image2d<float> which will receive data.

←*filename* The source.

7.73 mln::io::magick Namespace Reference

Namespace of magick input/output handling.

Functions

- template<typename I >
  void load (Image< I > &ima, const std::string &filename)
- template<typename I >
  void save (const Image< I > &ima, const std::string &filename)

7.73.1 Detailed Description

Namespace of magick input/output handling.

7.73.2 Function Documentation

7.73.2.1 template<typename I > void mln::io::magick::load (Image< I > & *ima*, const std::string & *filename*) [inline]

Load a magick image in a Milena image.

Parameters:

→ ***ima*** A reference to the image which will receive data.

← ***filename*** The source.

References mln::initialize(), and mln::point< G, C >::to_vec().

7.73.2.2 template<typename I > void mln::io::magick::save (const Image< I > & *ima*, const std::string & *filename*) [inline]

Save a Milena image in a magick image.

Parameters:

→ ***ima*** A reference to the image to save.

← ***filename*** The output.

References mln::point< G, C >::to_vec().

7.74 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'.

7.74.1 Detailed Description

Namespace of off input/output handling.

7.74.2 Function Documentation

7.74.2.1 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:

→ ***ima*** A reference to the image to construct.

← ***filename*** The name of the file to load.

The image is said binary since data only represent the existence of faces.

Parameters:

→ ***ima*** A reference to the image to construct.

← ***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.

7.74.2.2 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:

← ***ima*** The image to save.

← ***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:

← ***ima*** The image to save.

← ***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.

7.74.2.3 template<typename I > void mln::io::off::save_bin_alt (const I & *ima*, const std::string & *filename*) [inline]

FIXME: Similar to mln::io::off::save(const bin_2complex_image3df&, const std::string&), but does not save faces whose value is 'false'.

7.75 mln::io::pbm Namespace Reference

Namespace of pbm input/output handling.

Namespaces

- namespace impl

  Namespace of pbm implementation details.

Functions

- image2d< bool > load (const std::string &filename)

  Load a pbm image in a image2d<float>.

- void load (image2d< bool > &ima, const std::string &filename)

  Load a pbm image in a Milena image.

- template<typename I >
  void save (const Image< I > &ima, const std::string &filename)

7.75.1 Detailed Description

Namespace of pbm input/output handling.

7.75.2 Function Documentation

7.75.2.1 image2d< bool > mln::io::pbm::load (const std::string & *filename*) [inline]

Load a pbm image in a image2d<float>.

Parameters:

←*filename* The image source.

Returns:

An image2d<float> which contains loaded data.

Load a pbm image in a image2d<float>.

Parameters:

←*filename* The image source.

Returns:

An image2d<float> which contains loaded data.

7.75.2.2 void mln::io::pbm::load (image2d< bool > & *ima*, const std::string & *filename*) [inline]

Load a pbm image in a Milena image.

Parameters:

→ ***ima*** A reference to the image2d<bool> which will receive data.

← ***filename*** The source.

7.75.2.3 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:

← ***ima*** The image to save.

↔ ***filename*** the destination.

7.76 mln::io::pbm::impl Namespace Reference

Namespace of pbm implementation details.

7.76.1 Detailed Description

Namespace of pbm implementation details.

7.77 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.

7.77.1 Detailed Description

Namespace of pbms input/output handling.

7.77.2 Function Documentation

7.77.2.1 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:

→ ***ima*** A reference to the 3D image which will receive data.

← ***filenames*** The list of 2D images to load..

7.78 mln::io::pbms::impl Namespace Reference

Namespace of pbms implementation details.

7.78.1 Detailed Description

Namespace of pbms implementation details.

7.79 mln::io::pfm Namespace Reference

Namespace of pfm input/output handling.

Namespaces

- namespace impl
  Implementation namespace of pfm namespace.

Functions

- image2d< float > load (const std::string &filename)
  Load a pfm image in a image2d<float>.
- void load (image2d< float > &ima, const std::string &filename)
  Load a pfm image in a Milena image.
- template<typename I >
  void save (const Image< I > &ima, const std::string &filename)
  Save a Milena image as a pfm image.

7.79.1 Detailed Description

Namespace of pfm input/output handling.

7.79.2 Function Documentation

7.79.2.1 image2d< float > mln::io::pfm::load (const std::string & *filename*) [inline]

Load a pfm image in a image2d<float>.

Parameters:

←*filename* The image source.

Returns:

An image2d<float> which contains loaded data.

Load a pfm image in a image2d<float>.

Load a pbm image in a image2d<float>.

Parameters:

←*filename* The image source.

Returns:

An image2d<float> which contains loaded data.

7.79.2.2 void mln::io::pfm::load (image2d< float > & *ima*, const std::string & *filename*) [inline]

Load a pfm image in a Milena image.

Parameters:

→ ***ima*** A reference to the image2d<float> which will receive data.

← ***filename*** The source.

7.79.2.3 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:

← ***ima*** The image to save.

↔ ***filename*** the destination.

7.80 mln::io::pfm::impl Namespace Reference

Implementation namespace of pfm namespace.

7.80.1 Detailed Description

Implementation namespace of pfm namespace.

7.81 mln::io::pgm Namespace Reference

Namespace of pgm input/output handling.

Functions

- template<typename V >
  image2d< V > load (const std::string &filename)

  Load a pgm image in a Milena image.

- template<typename I >
  void load (Image< I > &ima, 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)

7.81.1 Detailed Description

Namespace of pgm input/output handling.

7.81.2 Function Documentation

7.81.2.1 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 whith the value type of the image loaded. (ex : load<value::int_u8>("..."))

Parameters:

←*filename* The image source.

Returns:

An image2d which contains loaded data.

Load a pgm image in a Milena image.

Load a pfm image in a image2d<float>.

Load a pbm image in a image2d<float>.

Parameters:

←*filename* The image source.

Returns:

An image2d<float> which contains loaded data.

7.81.2.2 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:

→ ***ima*** A reference to the image which will receive data.

← ***filename*** The source.

7.81.2.3 template<typename I > void mln::io::pgm::save (const Image< I > & *ima*, const std::string & *filename*) [inline]

Save a Milena image as a pgm image.

Parameters:

← ***ima*** The image to save.

↔ ***filename*** the destination.

7.82 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.

7.82.1 Detailed Description

Namespace of pgms input/output handling.

7.82.2 Function Documentation

7.82.2.1 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:

→ ***ima*** A reference to the 3D image which will receive data.

← ***filenames*** The list of 2D images to load..

7.83 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 I >
  void save (util::array< I > &arr, const std::string &filename, int start_value=0)
- template<typename I >
  void save (image1d< I > &ima, const std::string &filename, int start_value=0)

7.83.1 Detailed Description

Namespace of plot input/output handling.

7.83.2 Function Documentation

7.83.2.1 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:

← ***ima*** A reference to the image to load.
→ ***filename*** The output file.
← ***start_value*** The start index value of the plot (optional).

Load a Milena array from a plot file.

Parameters:

← ***arr*** A reference to the array to load.
→ ***filename*** The output file.

References mln::util::array< T >::append(), and mln::util::array< T >::clear().

7.83.2.2 template<typename I > void mln::io::plot::save (util::array< I > & *arr*, const std::string & *filename*, int *start_value* = 0) [inline]

Save a Milena array in a plot file.

Parameters:

← ***arr*** A reference to the array to save.
→ ***filename*** The output file.
← ***start_value*** The start index value of the plot (optional).

References mln::util::array< T >::nelements().

7.83.2.3 template<typename I > void mln::io::plot::save (image1d< I > & *ima*, const std::string & *filename*, int *start_value* = 0) [inline]

Save a Milena 1D image in a plot file.

Parameters:

← ***ima*** A reference to the image to save.

→ ***filename*** The output file.

← ***start_value*** The start index value of the plot (optional).

References mln::image1d< T >::ninds().

7.84 mln::io::pnm Namespace Reference

Namespace of pnm input/output handling.

Namespaces

- namespace impl

  Namespace of pnm's implementation details.

Functions

- 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 V >
  image2d< V > load (char type_, const std::string &filename)

  main function : load pnm format

- 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)

7.84.1 Detailed Description

Namespace of pnm input/output handling.

7.84.2 Function Documentation

7.84.2.1 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.

References mln::make::box2d(), load_raw_2d(), and max_component().

7.84.2.2 template<typename V > image2d<V> mln::io::pnm::load (char *type_*, const std::string & *filename*) [inline]

main function : load pnm format

References load_raw_2d(), and max_component().

7.84.2.3 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

Referenced by load().

7.84.2.4 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.

Referenced by load().

7.84.2.5 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:

← ***type*** The type of the image to save (can be PPM, PGM, PBM).

← ***ima_*** The image to save.

↔ ***filename*** the destination.

7.85 mln::io::pnm::impl Namespace Reference

Namespace of pnm's implementation details.

7.85.1 Detailed Description

Namespace of pnm's implementation details.

7.86 mln::io::pnms Namespace Reference

Namespace of pnms input/output handling.

Functions

- void load (char type, image3d< bool > &ima, const util::array< std::string > &filenames)
- 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.

7.86.1 Detailed Description

Namespace of pnms input/output handling.

7.86.2 Function Documentation

7.86.2.1 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.

References mln::make::image3d(), mln::util::array< T >::is_empty(), load(), and mln::util::array< T >::nelements().

7.86.2.2 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:

← ***type*** The type of the pnm files.

→ ***ima*** A reference to the 3D image which will receive data.

← ***filenames*** The list of 2D images to load..

References mln::make::image3d(), mln::util::array< T >::is_empty(), and mln::util::array< T >::nelements().

Referenced by load().

7.87 mln::io::ppm Namespace Reference

Namespace of ppm input/output handling.

Functions

- template<typename V >
  image2d< V > load (const std::string &filename)

  Load a ppm image in a Milena image.

- template<typename I >
  void load (Image< I > &ima, 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)

7.87.1 Detailed Description

Namespace of ppm input/output handling.

7.87.2 Function Documentation

7.87.2.1 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 whith the value type of the image loaded. (ex : load<value::int_u8>("..."))

Parameters:

←*filename* The image source.

Returns:

An image2d which contains loaded data.

Load a ppm image in a Milena image.

Load a pgm image in a Milena image.

Load a pfm image in a image2d<float>.

Load a pbm image in a image2d<float>.

Parameters:

←*filename* The image source.

Returns:

An image2d<float> which contains loaded data.

7.87.2.2 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:

→ ***ima*** A reference to the image which will receive data.

← ***filename*** The source.

7.87.2.3 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:

← ***ima*** The image to save.

↔ ***filename*** the destination.

Referenced by mln::registration::icp().

7.88 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.

7.88.1 Detailed Description

Namespace of ppms input/output handling.

7.88.2 Function Documentation

7.88.2.1 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:

→ ***ima*** A reference to the 3D image which will receive data.

← ***filenames*** The list of 2D images to load..

7.89 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.

7.89.1 Detailed Description

Namespace of tiff input/output handling.

7.89.2 Function Documentation

7.89.2.1 template<typename I > void mln::io::tiff::load (Image< I > & *ima_*, const std::string & *filename*) [inline]

Load a TIFF image to a Milena image.

7.90 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.

7.90.1 Detailed Description

Namespace of txt input/output handling.

7.90.2 Function Documentation

7.90.2.1 void mln::io::txt::save (const image2d< char > & *ima*, const std::string & *filename*) [inline]

Save an image as txt file.

Parameters:

← ***ima*** The image to save. Must be an image of char.

← ***filename*** the destination.

References mln::image2d< T >::domain().

7.91 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 L >
  mln::trait::ch_value< L, mln::value::rgb8 >::ret colorize (const Image< L > &input, const typename L::value &nlabels)
- template<typename V , typename L >
  mln::trait::ch_value< L, V >::ret colorize (const V &value, const Image< L > &labeled_image)
- 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 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 L >
  util::array< typename A::result > compute (const 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< 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 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)

  Compute an accumulator onto the pixel values of the image `input`.

- 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 >
  mln::trait::ch_value< L, mln_meta_accu_result(A, typename I::value) >::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 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 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 I , typename N , typename L >
  I 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)

  Relabel a labeled image in order to have a contiguous labeling.

- template<typename I >
  void pack_inplace (Image< I > &label, typename I::value &new_nlabels)

  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, const Function_v2v< F > &fv2v)

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, typename I::value &new_nlabels, const Function_v2b< F > &fv2b)

  Remove components and relabel a labeled image.

- template<typename I , typename F >
  void relabel_inplace (Image< I > &label, 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, typename I::value &nlabels, const Function_v2b< F > &fv2b)

  Remove components and relabel a labeled image inplace.

- 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 >
  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).

- 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).

7.91.1 Detailed Description

Namespace of labeling routines.

7.91.2 Function Documentation

7.91.2.1 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:

← ***input*** The input image.

← ***nbh*** The connexity of the background.

→ ***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

References value().

Referenced by fill_holes().

7.91.2.2 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:

← ***input*** The input image.

← ***nbh*** The connexity of the objects.

→ ***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).

Referenced by mln::graph::labeling().

7.91.2.3 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.

References colorize().

7.91.2.4 template<typename V , typename L > mln::trait::ch_value< L, V >::ret mln::labeling::colorize (const V & *value*, const Image< L > & *labeled_image*) [inline]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

References colorize(), and compute().

7.91.2.5 template<typename V , typename L > mln::trait::ch_value< L, V >::ret mln::labeling::colorize (const V & *value*, const Image< L > & *labeled_image*, 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:

← ***value*** value type used in the returned image.

← ***labeled_image*** A labeled image (

See also:

labeling::blobs).

Parameters:

← ***nlabels*** Number of labels.

References mln::literal::black, and mln::data::transform().

Referenced by colorize().

7.91.2.6 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:

← ***a*** A meta-accumulator.

← ***label*** The labeled image.

← ***nlabels*** The number of labels in `label`.

Returns:

A mln::p_array of accumulator result (one result per label).

References compute().

7.91.2.7 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:

← ***a*** An accumulator.

← ***label*** The labeled image.

← ***nlabels*** The number of labels in `label`.

Returns:

A mln::p_array of accumulator result (one result per label).

Compute an accumulator onto the pixel sites of each component domain of `label`.

Parameters:

← ***a_*** An accumulator.

← ***label_*** The labeled image.

← ***nlabels*** The number of labels in `label`.

Returns:

A mln::p_array of accumulator result (one result per label).

7.91.2.8 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:

← ***a*** A meta-accumulator.

← ***input*** The input image.

← ***label*** The labeled image.

← ***nlabels*** The number of labels in `label`.

Returns:

A mln::p_array of accumulator result (one result per label).

References compute().

7.91.2.9 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:

← ***a*** An accumulator.

← ***input*** The input image.

← ***label*** The labeled image.

← ***nlabels*** The number of labels in `label`.

Returns:

A mln::p_array of accumulator result (one result per label).

Compute an accumulator onto the pixel values of the image `input`.

Parameters:

← ***a_*** An accumulator.

← ***input_*** The input image.

← ***label_*** The labeled image.

← ***nlabels*** The number of labels in `label`.

Returns:

A mln::p_array of accumulator result (one result per label).

7.91.2.10 template<typename A , typename I , typename L > util::array< typename A::result > mln::labeling::compute (util::array< A > & *accus*, 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:

← ***a*** An array of accumulator.

← ***input*** The input image.

← ***label*** The labeled image.

← ***nlabels*** The number of labels in `label`.

Returns:

A mln::p_array of accumulator result (one result per label).

Compute an accumulator onto the pixel values of the image `input`.

Parameters:

← ***accus*** An array of accumulators.

← ***input_*** The input image.

← ***label_*** The labeled image.

← ***nlabels*** The number of labels in `label`.

Returns:

A mln::p_array of accumulator result (one result per label).

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().

7.91.2.11 template<typename A , typename I , typename L > mln::trait::ch_value< L, mln_meta_accu_result(A, typename I::value) >::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:

← ***accu*** The meta-accumulator.

← ***input*** The input image (values).

← ***labels*** The label image.

← ***nlabels*** The count of labels.

Returns:

The image where labels are replaced by the result of the accumulator.

References compute(), and compute_image().

7.91.2.12 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:

← ***accu*** The accumulator.

← ***input*** The input image (values).

← ***labels*** The label image.

← ***nlabels*** The count of labels.

Returns:

The image where labels are replaced by the result of the accumulator.

References compute(), and compute_image().

7.91.2.13 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*) [inline]

Compute an accumulator onto the pixel values of the image `input`.

for each component of the image `label`.

Parameters:

← ***a*** The mln::p_array of accumulator result.

← ***input*** The input image (values).

← ***labels*** The label image.

← ***nlabels*** The count of labels.

Returns:

The image where labels are replaced by the result of the accumulator.

Referenced by compute_image().

7.91.2.14 template<typename I , typename N , typename L > I 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:

← ***input*** The input image.

← ***nbh*** The connexity of the background.

→ ***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

References background(), compute(), mln::data::fill(), mln::initialize(), and mln::util::array< T >::nelements().

7.91.2.15 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*) [inline]

Connected component labeling of the flat zones of an image.

Parameters:

← ***input*** The input image.

← ***nbh*** The connexity of the flat zones.

→ ***nlabels*** The number of labels.

Returns:

The label image.

7.91.2.16 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:

← ***input*** The input image.

← ***nbh*** The connexity of the foreground.

→ ***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

References value().

7.91.2.17 template<typename I > mln::trait::concrete< I >::ret mln::labeling::pack (const Image< I > & *label*, typename I::value & *new_nlabels*) [inline]

Relabel a labeled image in order to have a contiguous labeling.

Parameters:

← ***label*** The labeled image.

→ ***new_nlabels*** The number of labels after relabeling.

Returns:

The relabeled image.

References compute(), mln::make::relabelfun(), and mln::data::transform().

7.91.2.18 template<typename I > void mln::labeling::pack_inplace (Image< I > & *label*, typename I::value & *new_nlabels*) [inline]

Relabel inplace a labeled image in order to have a contiguous labeling.

Parameters:

← ***label*** The labeled image.

→ ***new_nlabels*** The number of labels after relabeling.

References compute(), mln::make::relabelfun(), and mln::data::transform().

7.91.2.19 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*) [inline]

Connected component labeling of the regional maxima of an image.

Parameters:

← ***input*** The input image.

← ***nbh*** The connexity of the regional maxima.

→ ***nlabels*** The number of labeled regions.

Returns:

The label image.

7.91.2.20 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*) [inline]

Connected component labeling of the regional minima of an image.

Parameters:

← ***input*** The input image.

← ***nbh*** The connexity of the regional minima.

→ ***nlabels*** The number of labeled regions.

Returns:

The label image.

Referenced by mln::morpho::meyer_wst().

7.91.2.21 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:

← ***label*** the labeled image.

← ***nlabels*** the number of labels in `label`.

← ***fv2v*** function returning the new component id for each pixel value.

Returns:

the relabeled image.

References mln::data::transform().

7.91.2.22 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:

← ***label*** the labeled image.

← ***nlabels*** the number of labels in `label`.

→ ***new_nlabels*** the number of labels after relabeling.

← ***fv2b*** function returning whether a label must be replaced by the background.

Returns:

the relabeled image.

References mln::make::relabelfun().

7.91.2.23 template<typename I , typename F > void mln::labeling::relabel_inplace (Image< I > & *label*, typename I::value & *nlabels*, const Function_v2v< F > & *fv2v*) [inline]

Remove components and relabel a labeled image inplace.

Parameters:

↔ ***label*** the labeled image.

↔ ***nlabels*** the number of labels in `label`.

← ***fv2v*** function returning the new component id for each pixel value.

References mln::data::transform_inplace().

7.91.2.24 template<typename I , typename F > void mln::labeling::relabel_inplace (Image< I > & *label*, typename I::value & *nlabels*, const Function_v2b< F > & *fv2b*) [inline]

Remove components and relabel a labeled image inplace.

Parameters:

↔ ***label*** the labeled image.

↔ ***nlabels*** the number of labels in `label`.

← ***fv2b*** function returning whether a label must be replaced by the background.

References mln::make::relabelfun().

Referenced by mln::labeled_image< I >::relabel().

7.91.2.25 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*) [inline]

Connected component labeling of the image sites at a given value.

Parameters:

← ***input*** The input image.

← ***val*** The value to consider.

← ***nbh*** The connectivity of components.

→ ***nlabels*** The number of labels.

Returns:

The label image.

Referenced by background(), and foreground().

7.91.2.26 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:

← ***input*** The label image.

Returns:

A new image with values wrapped with type label_8.

References wrap().

7.91.2.27 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:

← ***value_type*** The type used to wrap the label type.

← ***input*** The label image.

Returns:

A new image with values wrapped with type V.

References mln::data::transform().

Referenced by wrap().

7.92 mln::labeling::impl Namespace Reference

Implementation namespace of labeling namespace.

Namespaces

- namespace generic

  Generic implementation namespace of labeling namespace.

7.92.1 Detailed Description

Implementation namespace of labeling namespace.

7.93 mln::labeling::impl::generic Namespace Reference

Generic implementation namespace of labeling namespace.

Functions

- 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.

- 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.

7.93.1 Detailed Description

Generic implementation namespace of labeling namespace.

7.93.2 Function Documentation

7.93.2.1 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.

Compute an accumulator onto the pixel values of the image `input`.

Parameters:

← ***accus*** An array of accumulators.

← ***input_*** The input image.

← ***label_*** The labeled image.

← ***nlabels*** The number of labels in `label`.

Returns:

A mln::p_array of accumulator result (one result per label).

Referenced by mln::labeling::colorize(), mln::labeling::compute(), mln::labeling::compute_image(), mln::labeling::fill_holes(), mln::make::p_edges_with_mass_centers(), mln::make::p_vertices_with_mass_centers(), mln::labeling::pack(), and mln::labeling::pack_inplace().

7.93.2.2 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.

Compute an accumulator onto the pixel values of the image `input`.

Parameters:

← ***a_*** An accumulator.

← ***input_*** The input image.

← ***label_*** The labeled image.

← ***nlabels*** The number of labels in `label`.

Returns:

A mln::p_array of accumulator result (one result per label).

7.93.2.3 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.

Compute an accumulator onto the pixel sites of each component domain of `label`.

Parameters:

← ***a_*** An accumulator.

← ***label_*** The labeled image.

← ***nlabels*** The number of labels in `label`.

Returns:

A mln::p_array of accumulator result (one result per label).

7.94 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, int dir)
- 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_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)
- template<typename I >
  mln::trait::concrete< I >::ret gaussian_2nd_derivative (const Image< I > &input, float sigma, int dir)
- template<typename I , typename W >
  mln_ch_convolve (I, W) convolve(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.

7.94.1 Detailed Description

Namespace of linear image processing routines.

7.94.2 Function Documentation

7.94.2.1 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)

References mln::initialize().

7.94.2.2 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

References mln::initialize().

Referenced by mln::subsampling::gaussian_subsampling().

7.94.2.3 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

References mln::initialize().

7.94.2.4 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)

References mln::initialize().

7.94.2.5 template<typename I > mln::trait::concrete< I >::ret mln::linear::gaussian_2nd_derivative (const Image< I > & *input*, float *sigma*) [inline]

Apply an approximated second derivative gaussian filter of `sigma` on `input` This filter is applied in all the input image direction.

Precondition:

input.is_valid

References mln::initialize().

7.94.2.6 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)

References mln::initialize().

7.94.2.7 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.

References mln_ch_convolve(), and mln::make::w_window2d().

7.94.2.8 template<typename I , typename W > mln::linear::mln_ch_convolve (I, W) const [inline]

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

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

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

Referenced by mln_ch_convolve(), and mln_ch_convolve_grad().

7.94.2.9 template<typename I > mln::linear::mln_ch_convolve_grad (I, int) const [inline]

Compute the vertical component of the 2D Sobel gradient.

References mln_ch_convolve(), and mln::data::transform().

7.95 mln::linear::impl Namespace Reference

Namespace of linear image processing routines implementation details.

7.95.1 Detailed Description

Namespace of linear image processing routines implementation details.

7.96 mln::linear::local Namespace Reference

Specializations of local linear routines.

Namespaces

- namespace impl

  Namespace of local linear routines implementation details.

Functions

- template<typename P , typename W , typename R >
  void convolve (const Generalized_Pixel< P > &p, const Weighted_Window< W > &w_win, R &result)
- 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)

7.96.1 Detailed Description

Specializations of local linear routines.

7.96.2 Function Documentation

7.96.2.1 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.

References convolve().

7.96.2.2 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.

Referenced by convolve().

7.97 mln::linear::local::impl Namespace Reference

Namespace of local linear routines implementation details.

7.97.1 Detailed Description

Namespace of local linear routines implementation details.

7.98 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.

7.98.1 Detailed Description

Namespace of literals.

7.98.2 Variable Documentation

7.98.2.1 const black_t & mln::literal::black = black_t()

Literal black.

Referenced by mln::labeling::colorize(), and mln::registration::icp().

7.98.2.2 const blue_t & mln::literal::blue = blue_t()

Literal blue.

7.98.2.3 const brown_t & mln::literal::brown = brown_t()

Literal brown.

7.98.2.4 const cyan_t & mln::literal::cyan = cyan_t()

Literal cyan.

7.98.2.5 const dark_gray_t & mln::literal::dark_gray = dark_gray_t()

Literal dark gray.

7.98.2.6 const green_t & mln::literal::green = green_t()

Literal green.

Referenced by mln::registration::icp(), and mln::make_debug_graph_image().

7.98.2.7 const identity_t & mln::literal::identity = identity_t()

Literal identity.

7.98.2.8 const light_gray_t & mln::literal::light_gray = light_gray_t()

Literal light gray.

7.98.2.9 const lime_t & mln::literal::lime = lime_t()

Literal lime.

7.98.2.10 const magenta_t & mln::literal::magenta = magenta_t()

Literal magenta.

7.98.2.11 const max_t & mln::literal::max = max_t()

Literal max.

7.98.2.12 const medium_gray_t & mln::literal::medium_gray = medium_gray_t()

Literal medium_gray.

7.98.2.13 const min_t & mln::literal::min = min_t()

Literal min.

7.98.2.14 const olive_t & mln::literal::olive = olive_t()

Literal olive.

7.98.2.15 const one_t & mln::literal::one = one_t()

Literal one.

Referenced by mln::algebra::h_vec< d, C >::h_vec(), mln::operator++(), and mln::operator–().

7.98.2.16 const orange_t & mln::literal::orange = orange_t()

Literal orange.

7.98.2.17 const origin_t & mln::literal::origin = origin_t()

Literal origin.

Referenced by mln::win::ball< G, C >::ball(), mln::geom::bbox(), mln::box< P >::box(), mln::geom::rotate(), and mln::make::w_window().

7.98.2.18 const pink_t & mln::literal::pink = pink_t()

Literal pink.

7.98.2.19 const purple_t & mln::literal::purple = purple_t()

Literal purple.

7.98.2.20 const red_t & mln::literal::red = red_t()

Literal red.

Referenced by mln::morpho::watershed::superpose(), and mln::debug::superpose().

7.98.2.21 const teal_t & mln::literal::teal = teal_t()

Literal teal.

7.98.2.22 const violet_t & mln::literal::violet = violet_t()

Literal violet.

7.98.2.23 const white_t & mln::literal::white = white_t()

Literal white.

Referenced by mln::registration::icp().

7.98.2.24 const yellow_t & mln::literal::yellow = yellow_t()

Literal yellow.

7.98.2.25 const zero_t & mln::literal::zero = zero_t()

Literal zero.

Referenced by mln::morpho::impl::generic::hit_or_miss(), mln::transform::influence_zone_geodesic(), mln::accu::shape::volume< I >::init(), mln::morpho::attribute::sum< I, S >::init(), mln::accu::math::sum< T, S >::init(), mln::accu::rms< T, V >::init(), mln::accu::convolve< T1, T2, R >::init(), mln::accu::center< P, V >::init(), mln::window< D >::is_centered(), 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::geom::impl::seeds2tiling_roundness(), mln::accu::shape::volume< I >::set_value(), mln::morpho::watershed::superpose(), mln::accu::stat::var< T >::to_result(), and mln::make::w_window_directional().

7.99 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)

7.99.1 Detailed Description

Namespace of logic.

7.99.2 Function Documentation

7.99.2.1 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:

↔ ***lhs*** First operand image.

← ***rhs*** Second operand image.

It performs:

for all p of rhs.domain

lhs(p) = lhs(p) and rhs(p)

Precondition:

`rhs.domain >= lhs.domain`

References mln::data::transform_inplace().

7.99.2.2 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:

← ***lhs*** First operand image.

← ***rhs*** Second operand image.

Returns:

The result image.

Precondition:

`lhs.domain == rhs.domain`

References mln::data::transform().

7.99.2.3 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:

↔ ***lhs*** First operand image.

← ***rhs*** Second operand image.

It performs:

for all p of rhs.domain

lhs(p) = lhs(p) and not rhs(p)

Precondition:

`rhs.domain >= lhs.domain`

References mln::data::transform_inplace().

7.99.2.4 template<typename I > void mln::logical::not_inplace (Image< I > & *input*) [inline]

Point-wise in-place "logical not" of image `input`.

Parameters:

↔ ***input*** The target image.

It performs:

for all p of input.domain

input(p) = not input(p)

Precondition:

`input.is_valid`

References mln::data::transform_inplace().

7.99.2.5 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:

↔ ***lhs*** First operand image.

← ***rhs*** Second operand image.

It performs:

for all p of rhs.domain

lhs(p) = lhs(p) or rhs(p)

Precondition:

`rhs.domain >= lhs.domain`

References mln::data::transform_inplace().

7.99.2.6 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:

↔ ***lhs*** First operand image.

← ***rhs*** Second operand image.

It performs:

for all p of rhs.domain

lhs(p) = lhs(p) xor rhs(p)

Precondition:

`rhs.domain >= lhs.domain`

References mln::data::transform_inplace().

7.100 mln::logical::impl Namespace Reference

Implementation namespace of logical namespace.

Namespaces

- namespace generic

  Generic implementation namespace of logical namespace.

7.100.1 Detailed Description

Implementation namespace of logical namespace.

7.101 mln::logical::impl::generic Namespace Reference

Generic implementation namespace of logical namespace.

7.101.1 Detailed Description

Generic implementation namespace of logical namespace.

7.102 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 (def::coord min_row, def::coord min_col, def::coord max_row, def::coord max_col)

  Create an mln::box2d.

- mln::box2d box2d (unsigned nrows, unsigned ncols)

  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 (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.

- mln::box3d box3d (unsigned nslis, unsigned nrows, unsigned ncols)

  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_psite< D, G > > detachment (const complex_psite< 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 >
  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_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_vertices< G > dummy_p_vertices (const Graph< G > &g)

  Create a p_vertices 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 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 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 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 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 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 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 R, unsigned C>
  mln::image2d< V > image (V(&values)[R][C])

Create an image2d from an 2D array of values.

- 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 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_)

  Construct 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_v2v< F > &fv2v, const V &nlabels, V &new_nlabels)

  Create a i2v function from a v2v function.

- 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 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 T >
  algebra::vec< 1, T > vec (const T &v_0)

  Create an mln::algebra::vec<n,T>.

- 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 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 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<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<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<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.

7.102.1 Detailed Description

Namespace of routines that help to make Milena's objects.

7.102.2 Function Documentation

7.102.2.1 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 fomal definition of the attachment here (see couprie.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.

References cell(), and mln::topo::is_facet().

Referenced by mln::topo::is_simple_cell< I >::operator()().

7.102.2.2 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:

← ***min_ind*** Minimum index.

← ***max_ind*** Maximum index.

Precondition:

`max_ind` >= `min_ind`.

Returns:

A 1D box.

References box1d().

7.102.2.3 mln::box1d mln::make::box1d (unsigned *ninds*) [inline]

Create an mln::box1d.

Parameters:

← ***ninds*** Number of indices.

Precondition:

`ninds` != 0 and `ncols` != 0.

Returns:

A 1D box.

Referenced by box1d(), and mln::image1d< T >::image1d().

7.102.2.4 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:

← ***min_row*** Index of the top most row.

← ***min_col*** Index of the left most column.

← ***max_row*** Index of the botton most row.

← ***max_col*** Index of the right most column.

Precondition:

`max_row` >= `min_row` and `max_col` >= `min_col`.

Returns:

A 2D box.

7.102.2.5 mln::box2d mln::make::box2d (unsigned *nrows*, unsigned *ncols*) [inline]

Create an mln::box2d.

Parameters:

← ***nrows*** Number of rows.

← ***ncols*** Number of columns.

Precondition:

`nrows` != 0 and `ncols` != 0.

Returns:

A 2D box.

Referenced by mln::image2d< T >::image2d(), and mln::io::pnm::load().

7.102.2.6 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:

← ***min_row*** Index of the top most row.

← ***min_col*** Index of the left most column.

← ***max_row*** Index of the botton most row.

← ***max_col*** Index of the right most column.

Precondition:

`max_row` >= `min_row` and `max_col` >= `min_col`.

Returns:

A 2D_H box.

References point2d_h().

7.102.2.7 mln::box2d_h mln::make::box2d_h (unsigned *nrows*, unsigned *ncols*) [inline]

Create an mln::box2d_h.

Parameters:

← ***nrows*** Number of rows.

← ***ncols*** Number of columns.

Precondition:

`nrows` != 0 and `ncols` != 0.

Returns:

A 2D_H box.

References point2d_h().

7.102.2.8 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:

← ***min_sli*** Index of the lowest slice.

← ***min_row*** Index of the top most row.

← ***min_col*** Index of the left most column.

← ***max_sli*** Index of the highest slice.

← ***max_row*** Index of the botton most row.

← ***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.

7.102.2.9 mln::box3d mln::make::box3d (unsigned *nslis*, unsigned *nrows*, unsigned *ncols*) [inline]

Create an mln::box3d.

Parameters:

← ***nslis*** Number of slices.

← ***nrows*** Number of rows.

← ***ncols*** Number of columns.

Precondition:

`ninds` != 0 and `ncols` != 0 and `nslis` != 0.

Returns:

A 3D box.

Referenced by image3d(), and mln::image3d< T >::image3d().

7.102.2.10 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.

References mln::topo::is_facet(), and mln::complex_psite< D, G >::n().

Referenced by attachment(), and detachment().

7.102.2.11 template<typename T , typename U > util::couple<T,U> mln::make::couple (const T & *val1*, const T & *val2*) [inline]

Construct an mln::util::couple on-the-fly.

Referenced by mln::transform::distance_and_closest_point_geodesic(), and mln::transform::distance_and_influence_zone_geodesic().

7.102.2.12 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 fomal definition of the detachment here (see couprie.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.

References cell(), and mln::topo::is_facet().

Referenced by mln::topo::detach().

7.102.2.13 mln::dpoint2d_h mln::make::dpoint2d_h (def::coord *row*, def::coord *col*) [inline]

Create an mln::dpoint2d_h.

Parameters:

← ***row*** Row coordinate.

← ***col*** Column coordinate.

Returns:

A 2D dpoint.

7.102.2.14 template<typename G > p_edges< G > mln::make::dummy_p_edges (const Graph< G > & *g*) [inline]

Create a p_edges which associate a graph element to a constant site.

`0` (int) is used as dummy site.

Parameters:

← ***g*** A graph.

Returns:

A p_edges.

7.102.2.15 template<typename G , typename P > p_edges< G, pw::cst_< P > > mln::make::dummy_p_edges (const Graph< G > & *g_*, const P & *dummy_site*) [inline]

Create a p_edges which associate a graph element to a constant site.

Parameters:

← ***g_*** A graph.

← ***dummy_site*** The dummy site mapped to graph edges.

Returns:

A p_edges.

7.102.2.16 template<typename G > p_vertices< G > mln::make::dummy_p_vertices (const Graph< G > & *g*) [inline]

Create a p_vertices which associate a graph element to a constant site.

0 (int) is used as dummy site.

Parameters:

← ***g*** A graph.

Returns:

A p_vertices.

7.102.2.17 template<typename G , typename P > p_vertices< G, pw::cst_< P > > mln::make::dummy_p_vertices (const Graph< G > & *g_*, const P & *dummy_site*) [inline]

Create a p_vertices which associate a graph element to a constant site.

Parameters:

← ***g_*** A graph.

← ***dummy_site*** The dummy site mapped to graph vertices.

Returns:

A p_vertices.

7.102.2.18 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:

← ***v_ima_*** A vertex image.

← ***fv_*** A function mapping two vertex ids to a value. The result is associated to the corresponding edge.

Returns:

an edge image without localization information mapped to graph elements.

7.102.2.19 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:

← ***v_ima_*** A vertex image.

← ***pe*** A p_edges mapping graph element to sites .

← ***fv_*** A function mapping two vertex ids to a value. The result is associated to the corresponding edge.

Returns:

an edge image.

7.102.2.20 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:

← ***g_*** A graph

← ***fp*** A function mapping edge ids to sites.

← ***fv*** A function mapping edge ids to values.

Returns:

an edge image.

7.102.2.21 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:

← ***g*** A graph

← ***fv*** A function mapping edge ids to values.

Returns:

an edge image.

7.102.2.22 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>.

Referenced by mln::fun::x2x::rotation< n, C >::rotation().

7.102.2.23 template<typename V , unsigned S, unsigned R, unsigned C> mln::image3d< V > mln::make::image (V(&) *values*[S][R][C]) [inline]

Create an image3d from an 3D array of values.

Parameters:

← ***values*** 3D array.

Returns:

A 3D image.

References mln::opt::at().

7.102.2.24 template<typename V , unsigned R, unsigned C> mln::image2d< V > mln::make::image (V(&) *values*[R][C]) [inline]

Create an image2d from an 2D array of values.

Parameters:

← ***values*** 2D array.

Returns:

A 2D image.

References mln::opt::at().

7.102.2.25 template<typename V , unsigned L> mln::image1d< V > mln::make::image (V(&) *values*[L]) [inline]

Create an image1d from an 1D array of values.

Parameters:

← ***values*** 1D array.

Returns:

A 1D image.

7.102.2.26 template<typename V , unsigned S> mln::image2d< V > mln::make::image2d (V(&) *values*[S]) [inline]

Create an image2d from an 2D array of values.

Parameters:

← ***values*** 2D array.

Returns:

A 2D image.

7.102.2.27 template<typename I > mln::image3d< typename I::value > mln::make::image3d (const Image< I > & *ima*) [inline]

Create an image3d from a 2D image.

References box3d(), and mln::data::paste().

7.102.2.28 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.

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().

7.102.2.29 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:

← ***iz*** influence zone image.

← ***nbh*** A neighborhood.

← ***nlabels*** number of influence zone in `iz`.

Returns:

util::graph Graph based on the adjacency of the influence zones.

Create a graph from an influence zone image.

Parameters:

← ***iz_*** influence zone image.

← ***nbh_*** A neighborhood.

← ***nlabels*** number of influence zone in `iz`.

Returns:

util::graph Graph based on the adjacency of the influence zones.

7.102.2.30 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:

← ***tab*** Array of values.

Precondition:

The array dimension has to be n ∗ m.

7.102.2.31 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.

7.102.2.32 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

References mln::labeling::compute().

7.102.2.33 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

References mln::labeling::compute().

7.102.2.34 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:

← ***ima*** The input image.

← ***p*** The point site.

Returns:

An mln::util::pix.

7.102.2.35 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`.

7.102.2.36 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`.

7.102.2.37 mln::point2d_h mln::make::point2d_h (def::coord *row*, def::coord *col*) [inline]

Create an mln::point2d_h.

Parameters:

← ***row*** Row coordinate.

← ***col*** Column coordinate.

Returns:

A 2D point.

Referenced by box2d_h().

7.102.2.38 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:

← ***wshd_*** Watershed image.

← ***nbh_*** Neighborhood

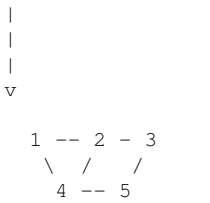
← ***nbasins*** Number of influence zone in `wshd`.

Returns:

A couple. First element is the graph, second element is an image with a labeled watershed line.

Watershed image (watershed line labeled with 0)

Labeled watershed line



Region Adjacency graph (RAG)

7.102.2.39 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:

← ***wshd_*** watershed image.

← ***nbh*** A neighborhood.

← ***nbasins*** number of influence zone in wshd.

Returns:

util::graph Graph based on the adjacency of the influence zones.

7.102.2.40 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:

← ***fv2v*** A v2v function. This function maps an id to an already existing one.

← ***nlabels*** The number of labels.

← ***new_nlabels*** The number of labels after relabeling.

Returns:

a i2v function.

See also:

mln::labeling::relabel

References mln::literal::zero.

7.102.2.41 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:

← ***fv2b*** A v2b function.

← ***nlabels*** The number of labels.

← ***new_nlabels*** The number of labels after relabeling.

Returns:

a i2v function.

See also:

mln::labeling::relabel

References mln::literal::zero.

Referenced by mln::labeling::pack(), mln::labeling::pack_inplace(), mln::labeling::relabel(), and mln::labeling::relabel_inplace().

7.102.2.42 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:

← ***v_0*** First coordinate.

← ***v_1*** Second coordinate.

← ***v_2*** Third coordinate.

← ***v_3*** Fourth coordinate.

Returns:

A 4D vector.

7.102.2.43 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:

← ***v_0*** First coordinate.

← ***v_1*** Second coordinate.

← ***v_2*** Third coordinate.

Returns:

A 3D vector.

7.102.2.44 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:

← ***v_0*** First coordinate.

← ***v_1*** Second coordinate.

Returns:

A 2D vector.

7.102.2.45 template<typename T > algebra::vec< 1, T > mln::make::vec (const T & *v_0*) [inline]

Create an mln::algebra::vec<n,T>.

Parameters:

← ***v_0*** First coordinate.

Returns:

A 1D vector.

7.102.2.46 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*) [inline]

Construct a vertex image.

Parameters:

← ***g_*** A graph.

← ***fp*** A function mapping vertex ids to sites.

← ***fv*** A function mapping vertex ids to values.

Returns:

A vertex image.

7.102.2.47 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*) [inline]

Construct a vertex image.

Parameters:

← ***g*** A graph.

← ***fv*** A function mapping vertex ids to values.

Returns:

A vertex image.

7.102.2.48 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:

← ***ima_*** The labeling image.

← ***orig_*** The original image.

← ***nbh*** The neighborhood for computing algorithm.

Returns:

The computed graph.

References mln::util::graph::add_edge(), mln::util::graph::add_vertex(), and mln::estim::min_max().

7.102.2.49 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:

← ***win*** A simple window.

← ***wei*** A weight function.

Returns:

A weighted window.

References mln::w_window< D, W >::insert(), and mln::literal::origin.

7.102.2.50 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:

← ***weights*** Array.

Precondition:

The array size, `M`, has to be a square of an odd integer.

Returns:

A 1D weighted window.

References mln::w_window< D, W >::insert().

7.102.2.51 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:

← ***weights*** Array.

Precondition:

The array size, `S`, has to be a square of an odd integer.

Returns:

A 2D weighted window.

Referenced by mln::linear::mln_ch_convolve().

7.102.2.52 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:

← ***weights*** Array.

Precondition:

The array size, `M`, has to be a cube of an odd integer.

Returns:

A 3D weighted window.

References mln::w_window< D, W >::insert().

7.102.2.53 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:

← ***dp*** A delta-point to set the orientation.

← ***weights*** An array of weights.

Returns:

A weighted window.

The window length `L` has to be odd.

References mln::w_window< D, W >::insert(), and mln::literal::zero.

7.103 mln::math Namespace Reference

Namespace of mathematical routines.

Functions

- template<unsigned n>
  value::int_u< n > abs (const value::int_u< n > &v)

  Specialization for mln::value::int_u.

- template<typename T >
  T abs (const T &v)

  Generic version.

- int abs (int v)

  Specializations for existing overloads of std::abs.

7.103.1 Detailed Description

Namespace of mathematical routines.

7.103.2 Function Documentation

7.103.2.1 template<unsigned n> value::int_u< n > mln::math::abs (const value::int_u< n > & *v*) [inline]

Specialization for mln::value::int_u.

7.103.2.2 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]).

References abs().

7.103.2.3 template<typename T > T mln::math::abs (const T & *v*) [inline]

Generic version.

Referenced by abs(), and mln::morpho::line_gradient().

7.104 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.

7.104.1 Detailed Description

Namespace of meta-programming tools.

7.105 mln::metal::impl Namespace Reference

Implementation namespace of metal namespace.

7.105.1 Detailed Description

Implementation namespace of metal namespace.

7.106 mln::metal::math Namespace Reference

Namespace of static mathematical functions.

Namespaces

- namespace impl

  Implementation namespace of metal::math namespace.

7.106.1 Detailed Description

Namespace of static mathematical functions.

7.107 mln::metal::math::impl Namespace Reference

Implementation namespace of metal::math namespace.

7.107.1 Detailed Description

Implementation namespace of metal::math namespace.

7.108 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)

  Meyer's Watershed Transform (WST) algorithm, with no count of basins.

- 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 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).

7.108.1 Detailed Description

Namespace of mathematical morphology routines.

7.108.2 Function Documentation

7.108.2.1 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).

Referenced by hit_or_miss_background_closing(), hit_or_miss_background_opening(), hit_or_miss_closing(), and thinning().

7.108.2.2 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).

7.108.2.3 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.

References plus(), top_hat_black(), and top_hat_white().

7.108.2.4 template<typename I , typename W > mln::trait::concrete< I >::ret mln::morpho::dilation (const Image< I > & *input*, const Window< W > & *win*) [inline]

Morphological dilation.

References general().

Referenced by gradient(), gradient_external(), mln::morpho::impl::generic::hit_or_miss(), hit_or_miss_background_opening(), hit_or_miss_opening(), laplacian(), mln::morpho::opening::approx::structural(), and mln::morpho::closing::approx::structural().

7.108.2.5 template<typename I , typename W > mln::trait::concrete< I >::ret mln::morpho::erosion (const Image< I > & *input*, const Window< W > & *win*) [inline]

Morphological erosion.

References general().

Referenced by gradient(), gradient_internal(), mln::morpho::impl::generic::hit_or_miss(), laplacian(), mln::morpho::opening::approx::structural(), and mln::morpho::closing::approx::structural().

7.108.2.6 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.

Referenced by dilation(), and erosion().

7.108.2.7 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.

References dilation(), erosion(), minus(), and mln::test::positive().

7.108.2.8 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.

References dilation(), minus(), and mln::test::positive().

7.108.2.9 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 Id - e_B.

References erosion(), minus(), and mln::test::positive().

7.108.2.10 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 HMT_(Bh,Bm) = e_Bh /\ (e_Bm o C).

References dilation(), erosion(), mln::data::fill(), mln::initialize(), and mln::literal::zero.

Referenced by thickening(), and thinning().

7.108.2.11 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 o HMTopeBG o C.

References complementation(), hit_or_miss_background_opening(), and hit_or_miss_closing().

7.108.2.12 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 HMTopeBG = HMTope_(Bm,Bh) o C = d_(-Bm) o HMT_(Bh,Bm).

References complementation(), dilation(), hit_or_miss_opening(), and mln::win::sym().

Referenced by hit_or_miss_background_closing(), and thick_miss().

7.108.2.13 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 o HMTope o C.

References complementation(), and hit_or_miss_opening().

Referenced by hit_or_miss_background_closing().

7.108.2.14 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 HMTope_(Bh,Bm) = d_(-Bh) o HMT_(Bh,Bm).

References dilation(), and mln::win::sym().

Referenced by hit_or_miss_background_opening(), hit_or_miss_closing(), and thin_fit().

7.108.2.15 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).

References dilation(), erosion(), mln::data::fill(), and minus().

7.108.2.16 template<typename V > edge_image< util::site_pair< point2d >, V, util::graph > mln::morpho::line_gradient (const mln::image2d< V > & *ima*) [inline]

Create a line graph image representing the gradient norm of a mln::image2d.

References mln::math::abs(), mln::image2d< T >::domain(), mln::box< P >::has(), mln::window< D >::insert(), and mln::Box< E >::nsites().

7.108.2.17 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*) [inline]

Meyer's Watershed Transform (WST) algorithm, with no count of basins.

Parameters:

← ***input*** The input image.

← ***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.

7.108.2.18 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*, L & *nbasins*) [inline]

Meyer's Watershed Transform (WST) algorithm.

Parameters:

← ***input*** The input image.

← ***nbh*** The connexity of markers.

→ ***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.

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.

7.108.2.19 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).

7.108.2.20 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).

7.108.2.21 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).

Referenced by gradient(), gradient_external(), gradient_internal(), laplacian(), thin_fit(), thinning(), top_hat_black(), mln::morpho::elementary::top_hat_black(), top_hat_self_complementary(), mln::morpho::elementary::top_hat_self_complementary(), top_hat_white(), and mln::morpho::elementary::top_hat_white().

7.108.2.22 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).

Referenced by contrast(), thick_miss(), and thickening().

7.108.2.23 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.

References mln::extension::adjust_fill(), mln::geom::delta(), mln::accu::stat::rank< T >::init(), and mln::initialize().

7.108.2.24 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).

References hit_or_miss_background_opening(), and plus().

7.108.2.25 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).

References hit_or_miss(), and plus().

Referenced by thinning().

7.108.2.26 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).

References hit_or_miss_opening(), and minus().

7.108.2.27 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 THIN_B = Id - HMT_B, where B = (Bfg, Bbg).

References complementation(), hit_or_miss(), minus(), and thickening().

7.108.2.28 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 clo_B - Id.

References minus(), and mln::test::positive().

Referenced by contrast().

7.108.2.29 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

= top_hat_white + top_hat_black

= (input - opening) + (closing - input)

= closing - opening.

References minus(), and mln::test::positive().

7.108.2.30 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 Id - ope_B.

References minus(), and mln::test::positive().

Referenced by contrast().

7.109 mln::morpho::approx Namespace Reference

Namespace of approximate mathematical morphology routines.

7.109.1 Detailed Description

Namespace of approximate mathematical morphology routines.

7.110 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.

7.110.1 Detailed Description

Namespace of attributes used in mathematical morphology.

7.111 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.

7.111.1 Detailed Description

Namespace of approximate mathematical morphology closing routines.

7.111.2 Function Documentation

7.111.2.1 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} o d_B.

References mln::morpho::dilation(), mln::morpho::erosion(), and mln::win::sym().

7.112 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_self_complementary (const Image< I > &input, const Neighborhood< N > &nbh)

  Morphological elementary self-complementary top-hat.

- 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).

7.112.1 Detailed Description

Namespace of image processing routines of elementary mathematical morphology.

7.112.2 Function Documentation

7.112.2.1 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.

Referenced by top_hat_black(), and top_hat_self_complementary().

7.112.2.2 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).

7.112.2.3 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.

Referenced by top_hat_self_complementary(), and top_hat_white().

7.112.2.4 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.

References closing(), mln::morpho::minus(), and mln::test::positive().

7.112.2.5 template<typename I , typename N > mln::trait::concrete< I >::ret mln::morpho::elementary::top_hat_self_complementary (const Image< I > & *input*, const Neighborhood< N > & *nbh*) [inline]

Morphological elementary self-complementary top-hat.

This operator is

= top_hat_white + top_hat_black

= (Id - opening) + (closing - Id)

= closing - opening.

References closing(), mln::morpho::minus(), opening(), and mln::test::positive().

7.112.2.6 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.

References mln::morpho::minus(), opening(), and mln::test::positive().

7.113 mln::morpho::impl Namespace Reference

Namespace of mathematical morphology routines implementations.

Namespaces

- namespace generic

  Namespace of mathematical morphology routines generic implementations.

7.113.1 Detailed Description

Namespace of mathematical morphology routines implementations.

7.114 mln::morpho::impl::generic Namespace Reference

Namespace of mathematical morphology routines generic implementations.

Functions

- 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 W >
  mln::trait::concrete< I >::ret rank_filter (const Image< I > &input_, const Window< W > &win_, unsigned k)

  Morphological rank_filter.

7.114.1 Detailed Description

Namespace of mathematical morphology routines generic implementations.

7.114.2 Function Documentation

7.114.2.1 template<typename I , typename Wh , typename Wm > mln::trait::concrete< I >::ret mln::morpho::impl::generic::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 HMT_(Bh,Bm) = e_Bh /\ (e_Bm o C).

References mln::morpho::dilation(), mln::morpho::erosion(), mln::data::fill(), mln::initialize(), and mln::literal::zero.

Referenced by mln::morpho::thickening(), and mln::morpho::thinning().

7.114.2.2 template<typename I , typename W > mln::trait::concrete< I >::ret mln::morpho::impl::generic::rank_filter (const Image< I > & *input_*, const Window< W > & *win_*, unsigned *k*) [inline]

Morphological rank_filter.

References mln::extension::adjust_fill(), mln::geom::delta(), mln::accu::stat::rank< T >::init(), and mln::initialize().

7.115 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.

7.115.1 Detailed Description

Namespace of approximate mathematical morphology opening routines.

7.115.2 Function Documentation

7.115.2.1 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} o e_B.

References mln::morpho::dilation(), mln::morpho::erosion(), and mln::win::sym().

7.116 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.

7.116.1 Detailed Description

Namespace of morphological reconstruction routines.

7.117 mln::morpho::reconstruction::by_dilation Namespace Reference

Namespace of morphological reconstruction by dilation routines.

7.117.1 Detailed Description

Namespace of morphological reconstruction by dilation routines.

7.118 mln::morpho::reconstruction::by_erosion Namespace Reference

Namespace of morphological reconstruction by erosion routines.

7.118.1 Detailed Description

Namespace of morphological reconstruction by erosion routines.

7.119 mln::morpho::tree Namespace Reference

Namespace of morphological tree-related routines.

Namespaces

- namespace filter

  Namespace for attribute filtering.

Functions

- 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 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 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 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_)
- template<typename T , typename A >
  void propagate_node_to_ancestors (typename A::psite n, const T &t, Image< A > &a_, const typename A::value &v)
- template<typename T , typename A >
  void propagate_node_to_descendants (typename A::psite &n, const T &t, Image< A > &a_, unsigned *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.

7.119.1 Detailed Description

Namespace of morphological tree-related routines.

7.119.2 Function Documentation

7.119.2.1 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:

← ***a*** Attribute.

← ***t*** Component tree.

← ***values*** Value image.

→ ***accu_image*** Optional argument used to store image.

Returns:

7.119.2.2 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 parent(p) == p
- p is a node iff either p is root or f(parent(p)) != 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 < .

7.119.2.3 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.

References propagate_if().

7.119.2.4 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 acsendant 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).

Referenced by propagate_if(), propagate_if_value(), and mln::morpho::tree::filter::subtractive().

7.119.2.5 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.

References propagate_if().

7.119.2.6 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 acsendant 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).

References propagate_if().

7.119.2.7 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:

← ***n*** Node to propagate.

← ***t*** Component tree used for propagation.

↔ ***a_*** Attribute image where values are propagated.

References propagate_node_to_ancestors().

7.119.2.8 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*) [inline]

Propagate a value `v` from a node `n` to its ancestors.

Parameters:

← ***n*** Node to propagate.

← ***t*** Component tree used for propagation.

← ***a_*** Attribute image where values are propagated.

← ***v*** Value to propagate.

Referenced by propagate_node_to_ancestors().

7.119.2.9 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:

← ***n*** Node to propagate.

← ***t*** Component tree used for propagation.

← ***a_*** Attribute image where values are propagated.

→ ***nb_leaves*** Optional. Store the number of leaves in the component.

7.119.2.10 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.

7.120 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.

7.120.1 Detailed Description

Namespace for attribute filtering.

7.120.2 Function Documentation

7.120.2.1 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:

← ***tree*** Component tree.

→ ***f_*** Image to filter.

← ***pred_*** Filtering criterion.

7.120.2.2 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.

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

7.120.2.3 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:

← ***tree*** Component tree.

→ ***f_*** Image to filter.

← ***pred_*** Filtering criterion.

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

7.120.2.4 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:

← ***tree*** Component tree.

→ ***f_*** Image to filter.

← ***pred_*** Filtering criterion.

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

7.120.2.5 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:

← ***tree*** Component tree.

→ ***f_*** Image to filter.

← ***pred_*** Filtering criterion.

References mln::morpho::tree::propagate_if().

7.121 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)

  Meyer's Watershed Transform (WST) algorithm, with no count of basins.

- 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 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.

7.121.1 Detailed Description

Namespace of morphological watershed routines.

7.121.2 Function Documentation

7.121.2.1 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*) [inline]

Meyer's Watershed Transform (WST) algorithm, with no count of basins.

Parameters:

← ***input*** The input image.

← ***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.

7.121.2.2 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*, L & *n_basins*) [inline]

Meyer's Watershed Transform (WST) algorithm.

Parameters:

← ***input*** The input image.

← ***nbh*** The connexity of markers.

→ ***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.

7.121.2.3 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.

References mln::literal::red, and superpose().

7.121.2.4 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.

References mln::data::convert(), mln::data::fill(), and mln::literal::zero.

Referenced by superpose().

7.122 mln::morpho::watershed::watershed Namespace Reference

Namespace of morphological watershed routines implementations.

Namespaces

- namespace generic

  Namespace of morphological watershed routines generic implementations.

7.122.1 Detailed Description

Namespace of morphological watershed routines implementations.

7.123 mln::morpho::watershed::watershed::generic Namespace Reference

Namespace of morphological watershed routines generic implementations.

7.123.1 Detailed Description

Namespace of morphological watershed routines generic implementations.

7.124 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 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 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 l2_distance (const C(&vec1)[n], const C(&vec2)[n])

  L2-norm distance between vectors vec1 *and* `vec2`.

- template<unsigned n, typename C >
  C linfty (const C(&vec)[n])

  L-infinity-norm of a vector vec.

- template<unsigned n, typename C >
  C linfty_distance (const C(&vec1)[n], const C(&vec2)[n])

  L-infinity-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.

7.124.1 Detailed Description

Namespace of norms.

7.124.2 Function Documentation

7.124.2.1 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*.

7.124.2.2 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*.

7.124.2.3 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*.

7.124.2.4 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`.

7.124.2.5 template<unsigned n, typename C > C mln::norm::linfty (const C(&) *vec*[n]) [inline]

L-infinity-norm of a vector *vec*.

7.124.2.6 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*.

7.124.2.7 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*.

Referenced by mln::geom::mesh_corner_point_area(), and mln::geom::mesh_normal().

7.125 mln::norm::impl Namespace Reference

Implementation namespace of norm namespace.

7.125.1 Detailed Description

Implementation namespace of norm namespace.

7.126 mln::opt Namespace Reference

Namespace of optional routines.

Namespaces

- namespace impl

  Implementation namespace of opt namespace.

Functions

- 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`*).*

- 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::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 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 ind)

  Read-write access to the `ima` *value located at (*`ind`*).*

- 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`*).*

7.126.1 Detailed Description

Namespace of optional routines.

7.126.2 Function Documentation

7.126.2.1 template<typename I > I::lvalue mln::opt::at (Image< I > & *ima*, def::coord *sli*, def::coord *row*, def::coord *col*) [inline]

Read-write access to the `ima` value located at (`sli`, `row`, `col`).

7.126.2.2 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).

7.126.2.3 template<typename I > I::lvalue mln::opt::at (Image< I > & *ima*, def::coord *row*, def::coord *col*) [inline]

Read-write access to the ima value located at (row, col).

7.126.2.4 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).

7.126.2.5 template<typename I > I::lvalue mln::opt::at (Image< I > & *ima*, def::coord *ind*) [inline]

Read-write access to the ima value located at (ind).

7.126.2.6 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).

Referenced by mln::transform::hough(), and mln::make::image().

7.127 mln::opt::impl Namespace Reference

Implementation namespace of opt namespace.

7.127.1 Detailed Description

Implementation namespace of opt namespace.

Three dimensions.

Two dimensions.

One dimension.

7.128 mln::pw Namespace Reference

Namespace of "point-wise" expression tools.

Classes

- class image

  A generic point-wise image implementation.

7.128.1 Detailed Description

Namespace of "point-wise" expression tools.

7.129 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 >
  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 , 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 >
  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.

7.129.1 Detailed Description

Namespace of "point-wise" expression tools.

7.129.2 Function Documentation

7.129.2.1 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*) [inline]

FIXME: work only for 3d images.

References mln::p_array< P >::nsites().

7.129.2.2 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*) [inline]

Register point in `c` using a function of closest points `closest_point`.

Parameters:

← ***P_*** The cloud of points.

← ***X*** the reference surface.

← ***closest_point*** The function of closest points.

Returns:

the rigid transformation which may be use later to create a registered image.

7.129.2.3 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:

← ***P_*** The cloud of points.

← ***X*** the reference surface.

← ***closest_point*** The function of closest points.

← ***initial_rot*** An initial rotation.

← ***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.

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.

7.129.2.4 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.

7.129.2.5 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.

7.129.2.6 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.

7.130 mln::select Namespace Reference

Select namespace (FIXME doc).

Classes

- struct p_of

  Structure p_of.

7.130.1 Detailed Description

Select namespace (FIXME doc).

7.131 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 , typename L >
  util::array< typename A::result > compute_with_weights (const Accumulator< A > &a, const Image< I > &w, const Image< L > &label, const typename L::value &nlabels)
  Compute an accumulator on every labeled sub-site-sets.

- 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
  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
  Compute an accumulator onto a site set.

7.131.1 Detailed Description

Namespace of image processing routines related to pixel sets.

7.131.2 Function Documentation

7.131.2.1 template<typename S > unsigned mln::set::card (const Site_Set< S > & *s*) [inline]

Compute the cardinality of the site set s.

7.131.2.2 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:

← ***a*** An accumulator.

← ***s*** A site set.

Returns:

The accumulator result.

Referenced by mln::registration::icp().

7.131.2.3 template<typename A , typename I , typename L > util::array< typename A::result > mln::set::compute_with_weights (const Accumulator< A > & *a_*, const Image< I > & *w_*, const Image< L > & *label_*, const typename L::value & *nlabels*) [inline]

Compute an accumulator on every labeled sub-site-sets.

Parameters:

← ***a*** An accumulator.

← ***w*** An image of weights (a site -> a weight).

← ***label*** A label image.

← ***nlabels*** The number of labels in `label`.

Returns:

An array of accumulator result. One per label.

Compute an accumulator on every labeled sub-site-sets.

Parameters:

← ***a_*** An accumulator.

← ***w_*** An image of weights (a site -> a weight).

← ***label_*** A label image.

← ***nlabels*** The number of labels in `label`.

Returns:

An array of accumulator result. One per label.

References compute_with_weights().

7.131.2.4 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:

← ***a*** An accumulator.
← ***w*** An image of weights (a site -> a weight).

Returns:

The accumulator result.

Compute an accumulator on a site set described by an image.

Parameters:

← ***a_*** An accumulator.
← ***w_*** An image of weights (a site -> a weight).

Returns:

The accumulator result.

Referenced by compute_with_weights().

7.131.2.5 template<typename S > S::site mln::set::get (const Site_Set< S > & *s*, size_t *index*) [inline]

FIXME.

7.131.2.6 template<typename S > bool mln::set::has (const Site_Set< S > & *s*, const typename S::site & *e*) [inline]

FIXME.

7.131.2.7 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:

← ***a*** A meta-accumulator.
← ***w*** An image of weights (a site -> a weight).

Returns:

The accumulator result.

7.131.2.8 template<typename A , typename S > mln::set::mln_meta_accu_result (A, typename S::site) const [inline]

Compute an accumulator onto a site set.

Parameters:

← ***a*** A meta-accumulator.
← ***s*** A site set.

7.132 mln::subsampling Namespace Reference

Namespace of "point-wise" expression tools.

Functions

- 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.

7.132.1 Detailed Description

Namespace of "point-wise" expression tools.

7.132.2 Function Documentation

7.132.2.1 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.

References mln::linear::gaussian(), mln::geom::ncols(), and mln::geom::nrows().

7.132.2.2 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.

References mln::geom::ncols(), and mln::geom::nrows().

7.133 mln::tag Namespace Reference

Namespace of image processing routines related to tags.

7.133.1 Detailed Description

Namespace of image processing routines related to tags.

7.134 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`.

7.134.1 Detailed Description

Namespace of image processing routines related to pixel tests.

7.134.2 Function Documentation

7.134.2.1 template<typename I > bool mln::test::positive (const Image< I > & *input*) [inline]

Test if an image only contains positive values.

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::elementary::top_hat_self_complementary(), mln::morpho::top_hat_white(), and mln::morpho::elementary::top_hat_white().

7.134.2.2 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:

← ***pset*** The point set.

← ***f*** The predicate.

7.134.2.3 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:

← ***lhs*** The image.

← ***rhs*** The image.

← ***f*** The predicate.

7.134.2.4 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:

← ***ima*** The image.

← ***f*** The predicate.

Referenced by mln::operator<(), mln::operator<=(), mln::operator==(), and positive().

7.135 mln::test::impl Namespace Reference

Implementation namespace of test namespace.

7.135.1 Detailed Description

Implementation namespace of test namespace.

7.136 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.

- struct 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`.

- struct 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_psite is an `N` *-face.*

- 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_psite< 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_psite< 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 N, unsigned D>
  algebraic_n_face< N, D > make_algebraic_n_face (const n_face< N, D > &f, bool sign)

  Create an algebraic `N-face` *handle of a* `D-complex`.

- 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 face< D > &f)

  Print an mln::topo::face.

- template<unsigned D>
  std::ostream & operator<< (std::ostream &ostr, const complex< D > &c)

  Pretty print a complex.

- 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>
  std::ostream & operator<< (std::ostream &ostr, const algebraic_face< D > &f)

  Print an mln::topo::algebraic_face.

- template<unsigned D>
  bool operator== (const complex< D > &lhs, const complex< D > &rhs)

  Compare two complexes for equality.

- template<unsigned D>
  algebraic_n_face< 1, D > edge (const n_face< 0, D > &f1, const n_face< 0, D > &f2)

  Helpers.

- 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>
  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 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 D>
  bool operator== (const face< D > &lhs, const face< D > &rhs)
  Comparison of two instances of mln::topo::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 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 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 D>
  bool operator== (const algebraic_face< D > &lhs, const algebraic_face< D > &rhs)
  Comparison of two instances of mln::topo::algebraic_face.

- 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.

- template<unsigned N, unsigned D>
  algebraic_n_face< N, D > operator- (const n_face< N, D > &f)
  Inversion operators.

- template<unsigned D>
  algebraic_face< D > operator- (const face< D > &f)
  Inversion operators.

7.136.1 Detailed Description

Namespace of "point-wise" expression tools.

7.136.2 Function Documentation

7.136.2.1 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.

References mln::make::detachment(), mln::data::fill(), and is_facet().

7.136.2.2 template<unsigned D> algebraic_n_face< 1, D > mln::topo::edge (const n_face< 0, D > & *f1*, const n_face< 0, D > & *f2*) [inline]

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 ajacent 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).

References mln::topo::n_face< N, D >::higher_dim_adj_faces().

7.136.2.3 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?

Referenced by mln::make::attachment(), mln::make::cell(), detach(), and mln::make::detachment().

7.136.2.4 template<unsigned D> algebraic_face< D > mln::topo::make_algebraic_face (const face< D > & *f*, bool *sign*) [inline]

Create an algebraic face handle of a `D-complex`.

7.136.2.5 template<unsigned N, unsigned D> algebraic_n_face< N, D > mln::topo::make_algebraic_n_face (const n_face< N, D > & *f*, bool *sign*) [inline]

Create an algebraic `N-face` handle of a `D-complex`.

7.136.2.6 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.

References mln::topo::n_face< N, D >::cplx().

7.136.2.7 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.

References mln::topo::face< D >::cplx(), and mln::operator!=().

7.136.2.8 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.

References mln::topo::n_face< N, D >::cplx().

7.136.2.9 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.

References mln::topo::face< D >::cplx().

7.136.2.10 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.

References mln::topo::n_faces_set< N, D >::add().

7.136.2.11 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.

References mln::topo::n_faces_set< N, D >::add().

7.136.2.12 template<unsigned N, unsigned D> algebraic_n_face< N, D > mln::topo::operator- (const n_face< N, D > & *f*) [inline]

Inversion operators.

7.136.2.13 template<unsigned D> algebraic_face< D > mln::topo::operator- (const face< D > & *f*) [inline]

Inversion operators.

7.136.2.14 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.

7.136.2.15 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.

7.136.2.16 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.

7.136.2.17 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.

7.136.2.18 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.

7.136.2.19 template<unsigned D> std::ostream & mln::topo::operator<< (std::ostream & *ostr*, const face< D > & *f*) [inline]

Print an mln::topo::face.

7.136.2.20 template<unsigned D> std::ostream & mln::topo::operator<< (std::ostream & *ostr*, const complex< D > & *c*) [inline]

Pretty print a complex.

References mln::topo::complex< D >::print().

7.136.2.21 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.

7.136.2.22 template<unsigned D> std::ostream & mln::topo::operator<< (std::ostream & *ostr*, const algebraic_face< D > & *f*) [inline]

Print an mln::topo::algebraic_face.

7.136.2.23 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.

References mln::topo::n_face< N, D >::cplx(), and mln::topo::n_face< N, D >::face_id().

7.136.2.24 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.

References mln::topo::face< D >::cplx(), mln::topo::face< D >::face_id(), and mln::topo::face< D >::n().

7.136.2.25 template<unsigned D> bool mln::topo::operator== (const complex< D > & *lhs*, const complex< D > & *rhs*) [inline]

Compare two complexes for equality.

7.136.2.26 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.

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().

7.136.2.27 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.

References mln::topo::face< D >::cplx(), mln::topo::face< D >::face_id(), mln::topo::face< D >::n(), and mln::topo::algebraic_face< D >::sign().

7.137 mln::trace Namespace Reference

Namespace of routines related to the trace mechanism.

7.137.1 Detailed Description

Namespace of routines related to the trace mechanism.

7.138 mln::trait Namespace Reference

Namespace where traits are defined.

7.138.1 Detailed Description

Namespace where traits are defined.

Namespace for image traits.

7.139 mln::transform Namespace Reference

Namespace of transforms.

Functions

- 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, 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 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 Neighborhood< 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 Neighborhood< 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 Neighborhood< 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 Neighborhood< N > &nbh)

- template<typename I , typename N , typename D >
  mln::trait::concrete< I >::ret influence_zone_geodesic (const Image< I > &input, const Neighborhood< N > &nbh, const D &max)
- template<typename I , typename N , typename D >
  mln::trait::concrete< I >::ret influence_zone_geodesic (const Image< I > &input, const Neighborhood< N > &nbh, const D &max, const typename I::value &background_value)

  Geodesic influence zone transform.

7.139.1 Detailed Description

Namespace of transforms.

7.139.2 Function Documentation

7.139.2.1 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:

← ***pset*** an array of sites.

← ***closest_point_domain*** domain of the returned image.

← ***nbh*** neighborhood

← ***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`.

References mln::geom::bbox(), mln::make::couple(), distance_geodesic(), mln::data::fill(), and mln::box< P >::is_valid().

7.139.2.2 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:

← ***input*** Image from which the geodesic distance is computed.

← ***nbh*** Neighborhood

← ***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`.

References mln::make::couple(), and distance_geodesic().

7.139.2.3 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:

← ***input*** Image from which the geodesic distance is computed.

← ***nbh*** Neighborhood

← ***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`.

References mln::make::couple(), and distance_geodesic().

7.139.2.4 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.

7.139.2.5 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.

Referenced by distance_and_closest_point_geodesic(), and distance_and_influence_zone_geodesic().

7.139.2.6 template<typename I > image2d< float > mln::transform::hough (const Image< I > & *input_*) [inline]

Compute the hough transform from a binary image.

Objects used for computation must be set to 'true'.

Parameters:

← ***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.

References mln::opt::at(), mln::data::fill(), mln::geom::min_col(), mln::geom::min_row(), mln::geom::ncols(), and mln::geom::nrows().

7.139.2.7 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*) [inline]

Influence zone transform.

References influence_zone_front().

7.139.2.8 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*) [inline]

Influence zone transform.

References mln::canvas::distance_front().

Referenced by influence_zone_front().

7.139.2.9 template<typename I , typename N > mln::trait::concrete< I >::ret mln::transform::influence_zone_geodesic (const Image< I > & *input*, const Neighborhood< N > & *nbh*) [inline]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

References influence_zone_geodesic().

7.139.2.10 template<typename I , typename N , typename D > mln::trait::concrete< I >::ret mln::transform::influence_zone_geodesic (const Image< I > & *input*, const Neighborhood< N > & *nbh*, const D & *max*) [inline]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

References influence_zone_geodesic(), and mln::literal::zero.

7.139.2.11 template<typename I , typename N , typename D > mln::trait::concrete< I >::ret mln::transform::influence_zone_geodesic (const Image< I > & *input*, const Neighborhood< N > & *nbh*, const D & *max*, const typename I::value & *background_value*) [inline]

Geodesic influence zone transform.

Parameters:

← ***input*** An image.

← ***nbh*** A neighborhood.

← ***max*** The maximum influence zone distance.

← ***background_value*** The value used as background (i.e. not propagated).

Returns:

An image of influence zone.

References mln::canvas::distance_geodesic().

Referenced by influence_zone_geodesic().

7.140 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.

7.140.1 Detailed Description

Namespace of tools using for more complex algorithm.

Forward declaration.

7.140.2 Typedef Documentation

7.140.2.1 typedef object_id<vertex_tag, unsigned> mln::util::vertex_id_t

Vertex id type.

7.140.3 Function Documentation

7.140.3.1 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:

← ***ima_*** The domain of output image.

← ***tree_node*** The root tree_node to display.

References mln::data::fill().

7.140.3.2 template<typename I , typename J > void mln::util::display_tree (const Image< J > & *ima_*, tree< I > & *tree*) [inline]

Display a tree.

Parameters:

← ***ima_*** The domain of output image.

← ***tree*** The tree to display.

References mln::util::tree< T >::root().

7.140.3.3 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*) [inline]

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`.

7.140.3.4 template<typename I > greater_point< I > mln::util::make_greater_point (const Image< I > & *ima*) [inline]

Helper to build a mln::util::greater_point.

7.140.3.5 template<typename I > greater_psite< I > mln::util::make_greater_psite (const Image< I > & *ima*) [inline]

Helper to build a mln::util::greater_psite.

7.140.3.6 template<typename G > bool mln::util::operator< (const vertex< G > & *lhs*, const vertex< G > & *rhs*) [inline]

Less operator. Test whether lhs.id() < rhs.id().

7.140.3.7 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`.

7.140.3.8 template<typename T > std::ostream & mln::util::operator<< (std::ostream & *ostr*, const array< T > & *a*) [inline]

Operator<<.

References mln::util::array< T >::nelements().

7.140.3.9 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.

References mln::util::vertex< G >::graph(), and mln::util::vertex< G >::id().

7.140.3.10 template<typename T > bool mln::util::operator== (const array< T > & *lhs*, const array< T > & *rhs*) [inline]

Operator==.

References mln::util::array< T >::std_vector().

7.140.3.11 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*.

Referenced by mln::util::ord_pair< T >::change_both(), mln::util::ord_pair< T >::change_first(), and mln::util::ord_pair< T >::change_second().

7.140.3.12 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*.

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().

7.140.3.13 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:

← ***tree*** The tree to convert.

→ ***output_*** The image containing tree informations.

References mln::util::impl::tree_fast_to_image().

Referenced by tree_fast_to_image().

7.140.3.14 template<typename T > tree_fast< T > mln::util::tree_to_fast (tree< T > & *input*) [inline]

Facade.

Convert a tree into an tree_fast.

Parameters:

← ***input*** The tree to convert.

Returns:

The tree_fast containing tree informations.

References mln::util::tree< T >::root().

7.140.3.15 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:

← ***tree*** The tree to convert.

→ ***output_*** The image containing tree information.

7.141 mln::util::impl Namespace Reference

Implementation namespace of util namespace.

Functions

- template<typename T , typename I >
  void tree_fast_to_image (tree_fast< T > &tree, Image< I > &output_)

7.141.1 Detailed Description

Implementation namespace of util namespace.

7.141.2 Function Documentation

7.141.2.1 template<typename T , typename I > void mln::util::impl::tree_fast_to_image (tree_fast< T > & *tree*, Image< I > & *output_*) [inline]

Convert a tree_fast into an image.

Parameters:

← ***tree*** The tree to convert.

→ ***output_*** The image containing tree informations.

References tree_fast_to_image().

Referenced by mln::util::tree_fast_to_image().

7.142 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.

- 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.

- class proxy< const I >

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.

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 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.

- std::ostream & operator<< (std::ostream &ostr, const sign &i)

  Print an signed integer `i` *into the output stream* `ostr`.

- template<typename T >
  std::ostream & operator<< (std::ostream &ostr, const scalar_< T > &s)

  Print a scalar `s` *in an 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`.

- template<typename I >
  std::ostream & operator<< (std::ostream &ostr, const proxy< I > &x)

  Print a value proxy `x` *into the output stream* `ostr`.

- 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 int_u_sat< n > &i)

  Print a saturated unsigned 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<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`.

- std::ostream & operator<< (std::ostream &ostr, const graylevel_f &g)

  Op<<.

- 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<<.

- bool operator== (const sign &lhs, const sign &rhs)

  Comparaison operator.

- template<typename V >
  V other (const V &val)

  Give an other value than `val`.

  - template<unsigned n, typename S >
    rgb< n >::interop operator* (const rgb< n > &lhs, const mln::value::scalar_< S > &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)

    Product.

  - template<unsigned n>
    rgb< n >::interop operator- (const rgb< n > &lhs, const rgb< n > &rhs)

Subtraction.

- 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<unsigned n, typename S >
  rgb< n >::interop operator/ (const rgb< n > &lhs, const mln::value::scalar_< S > &s)
  Division.

- 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<typename I >
  stack_image< 2, const I > stack (const Image< I > &ima1, const Image< I > &ima2)
  Shortcut to build a stack with two images.

7.142.1 Detailed Description

Namespace of materials related to pixel value types.

7.142.2 Typedef Documentation

7.142.2.1 typedef float01_<16> mln::value::float01_16

Alias for 16 bit float01.

7.142.2.2 typedef float01_<8> mln::value::float01_8

Alias for 8 bit float01.

7.142.2.3 typedef graylevel<16> mln::value::gl16

Alias for 16 bit graylevel.

7.142.2.4 typedef graylevel<8> mln::value::gl8

Alias for 8 bit graylevel.

7.142.2.5 typedef graylevel_f mln::value::glf

Alias for graylevels encoded by float.

7.142.2.6 typedef int_s<16> mln::value::int_s16

Alias for signed 16-bit integers.

7.142.2.7 typedef int_s<32> mln::value::int_s32

Alias for signed 32-bit integers.

7.142.2.8 typedef int_s<8> mln::value::int_s8

Alias for signed 8-bit integers.

7.142.2.9 typedef int_u<12> mln::value::int_u12

Alias for unsigned 12-bit integers.

7.142.2.10 typedef int_u<16> mln::value::int_u16

Alias for unsigned 16-bit integers.

7.142.2.11 typedef mln::value::int_u<32> mln::value::int_u32

Alias for unsigned 32-bit integers.

7.142.2.12 typedef mln::value::int_u<8> mln::value::int_u8

Alias for unsigned 8-bit integers.

7.142.2.13 typedef label<16> mln::value::label_16

Alias for 16-bit integers.

7.142.2.14 typedef mln::value::label<8> mln::value::label_8

Alias for 8-bit labels.

7.142.2.15 typedef rgb<16> mln::value::rgb16

Color class for red-green-blue where every component is 16-bit encoded.

7.142.2.16 typedef rgb<8> mln::value::rgb8

Color class for red-green-blue where every component is 8-bit encoded.

7.142.3 Function Documentation

7.142.3.1 template<typename Dest , typename Src > Dest mln::value::cast (const Src & *src*) [inline]

Cast a value `src` from type `Src` to type `Dest`.

7.142.3.2 template<typename V > internal::equiv_< V >::ret mln::value::equiv (const mln::Value< V > & *v*) [inline]

Access to the equivalent value.

7.142.3.3 template<unsigned n, typename S > rgb< n >::interop mln::value::operator∗ (const rgb< n > & *lhs*, const mln::value::scalar_< S > & *s*) [inline]

Product.

7.142.3.4 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*) [inline]

Product.

7.142.3.5 template<unsigned n> rgb< n >::interop mln::value::operator+ (const rgb< n > & *lhs*, const rgb< n > & *rhs*) [inline]

Addition.

{

7.142.3.6 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*) [inline]

Addition.

{

7.142.3.7 template<unsigned n> rgb< n >::interop mln::value::operator- (const rgb< n > & *lhs*, const rgb< n > & *rhs*) [inline]

Subtraction.

7.142.3.8 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*) [inline]

Subtraction.

7.142.3.9 template<unsigned n, typename S > rgb< n >::interop mln::value::operator/ (const rgb< n > & *lhs*, const mln::value::scalar_< S > & *s*) [inline]

Division.

7.142.3.10 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*) [inline]

Division.

7.142.3.11 std::ostream & mln::value::operator<< (std::ostream & *ostr*, const sign & *i*) [inline]

Print an signed integer `i` into the output stream `ostr`.

Parameters:

↔ ***ostr*** An output stream.
← ***i*** An sign value

Returns:

The modified output stream `ostr`.

References mln::debug::format().

7.142.3.12 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`.

7.142.3.13 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:

↔ ***ostr*** An output stream.
← ***c*** An rgb.

Returns:

The modified output stream `ostr`.

References mln::debug::format().

7.142.3.14 template<typename I > std::ostream & mln::value::operator<< (std::ostream & *ostr*, const proxy< I > & *x*) [inline]

Print a value proxy `x` into the output stream `ostr`.

Parameters:

↔ ***ostr*** An output stream.

← ***x*** A value proxy.

Returns:

The modified output stream `ostr`.

7.142.3.15 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:

↔ ***ostr*** An output stream.

← ***l*** A label.

Returns:

The modified output stream `ostr`.

References mln::debug::format().

7.142.3.16 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:

↔ ***ostr*** An output stream.

← ***i*** A saturated unsigned integer.

Returns:

The modified output stream `ostr`.

References mln::debug::format().

7.142.3.17 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:

↔ ***ostr*** An output stream.

← ***i*** An unsigned integer.

Returns:

The modified output stream `ostr`.

References mln::debug::format().

7.142.3.18 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:

↔ ***ostr*** An output stream.

← ***i*** An signed integer.

Returns:

The modified output stream `ostr`.

References mln::debug::format().

7.142.3.19 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:

↔ ***ostr*** An output stream.

← ***c*** An rgb.

Returns:

The modified output stream `ostr`.

References mln::debug::format().

7.142.3.20 std::ostream & mln::value::operator<< (std::ostream & *ostr*, const graylevel_f & *g*) [inline]

Op<<.

References mln::value::graylevel_f::value().

7.142.3.21 template<unsigned n> std::ostream & mln::value::operator<< (std::ostream & *ostr*, const graylevel< n > & *g*) [inline]

Op<<.

7.142.3.22 template<unsigned n> std::ostream & mln::value::operator<< (std::ostream & *ostr*, const float01_< n > & *f*) [inline]

Op<<.

7.142.3.23 bool mln::value::operator== (const sign & *lhs*, const sign & *rhs*) [inline]

Comparaison operator.

7.142.3.24 template<typename H , typename S , typename L > bool mln::value::operator== (const hsl_< H, S, L > & *lhs*, const hsl_< H, S, L > & *rhs*) [inline]

Comparison.

7.142.3.25 template<typename V > V mln::value::other (const V & *val*) [inline]

Give an other value than `val`.

7.142.3.26 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.

7.143 mln::value::impl Namespace Reference

Implementation namespace of value namespace.

7.143.1 Detailed Description

Implementation namespace of value namespace.

7.144 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 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 W >
  mln_regular (W) shift(const Window< W > &win

  Shift a window `win` *with a delta-point* `dp`.

- 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 >
  W sym (const Weighted_Window< W > &w_win)

  Give the symmetrical weighted window of `w_win`.

- template<typename W >
  W sym (const Window< W > &win)

  Give the symmetrical window of `win`.

7.144.1 Detailed Description

Namespace of image processing routines related to win.

7.144.2 Function Documentation

7.144.2.1 template<typename N1 , typename N2 > N2 neighb< typename N1::window::regular > mln::win::diff (const Neighborhood< N1 > & *nbh1*, const Neighborhood< N2 > & *nbh2*) [inline]

Set difference between a couple of neighborhoods `nbh1` and `nbh2`.

Referenced by mln::operator-().

7.144.2.2 template<typename W > mln::win::mln_regular (W) const [inline]

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

7.144.2.3 template<typename W1 , typename W2 > mln::win::mln_regular (W1) const [inline]

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

7.144.2.4 template<typename W > W mln::win::sym (const Weighted_Window< W > & *w_win*) [inline]

Give the symmetrical weighted window of `w_win`.

7.144.2.5 template<typename W > W mln::win::sym (const Window< W > & *win*) [inline]

Give the symmetrical window of `win`.

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().

Chapter 8

Class Documentation

8.1 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.
- 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.

8.1.1 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.

8.1.2 Member Function Documentation

8.1.2.1 template<typename P , typename V > void mln::accu::center< P, V >::init () [inline]

Manipulators.

References mln::literal::zero.

8.1.2.2 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.

Referenced by mln::accu::center< P, V >::to_result().

8.1.2.3 void mln::Accumulator< center< P, V > >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.1.2.4 void mln::Accumulator< center< P, V > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.1.2.5 template<typename P , typename V > V mln::accu::center< P, V >::to_result () const [inline]

Get the value of the accumulator.

References mln::accu::center< P, V >::is_valid().

8.2 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.

8.2.1 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.

8.2.2 Member Function Documentation

8.2.2.1 template<typename T1 , typename T2 , typename R > void mln::accu::convolve< T1, T2, R >::init () [inline]

Manipulators.

References mln::literal::zero.

8.2.2.2 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.

8.2.2.3 void mln::Accumulator< convolve< T1, T2, R > >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.2.2.4 void mln::Accumulator< convolve< T1, T2, R > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.2.2.5 template<typename T1 , typename T2 , typename R > R mln::accu::convolve< T1, T2, R >::to_result () const [inline]

Get the value of the accumulator.

8.3 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.

8.3.1 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.

8.3.2 Member Function Documentation

8.3.2.1 template<typename F , typename S > void mln::accu::count_adjacent_vertices< F, S >::init () [inline]

Manipulators.

8.3.2.2 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.

8.3.2.3 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*.

8.3.2.4 void mln::Accumulator< count_adjacent_vertices< F, S > >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.3.2.5 void mln::Accumulator< count_adjacent_vertices< F, S > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.3.2.6 template<typename F , typename S > unsigned mln::accu::count_adjacent_vertices< F, S >::to_result () const [inline]

Get the value of the accumulator.

8.4 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.

8.4.1 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.

8.4.2 Member Function Documentation

8.4.2.1 template<typename L > void mln::accu::count_labels< L >::init () [inline]

Manipulators.

8.4.2.2 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.

8.4.2.3 template<typename L > void mln::accu::count_labels< L >::set_value (unsigned *c*) [inline]

Force the value of the counter to *c*.

8.4.2.4 void mln::Accumulator< count_labels< L > >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.4.2.5 void mln::Accumulator< count_labels< L > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.4.2.6 template<typename L > unsigned mln::accu::count_labels< L >::to_result () const [inline]

Get the value of the accumulator.

8.5 mln::accu::count_value< V > Struct Template Reference

Count 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.

8.5.1 Detailed Description

template<typename V> struct mln::accu::count_value< V >

Count a given value.

8.5.2 Member Function Documentation

8.5.2.1 template<typename V > void mln::accu::count_value< V >::init () [inline]

Manipulators.

8.5.2.2 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.

8.5.2.3 template<typename V > void mln::accu::count_value< V >::set_value (unsigned *c*) [inline]

Force the value of the counter to *c*.

8.5.2.4 void mln::Accumulator< count_value< V > >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.5.2.5 void mln::Accumulator< count_value< V > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.5.2.6 template<typename V > unsigned mln::accu::count_value< V >::to_result () const [inline]

Get the value of the accumulator.

8.6 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.

8.6.1 Detailed Description

template<typename V> struct mln::accu::histo< V >

Generic histogram class over a value set with type V.

8.6.2 Member Function Documentation

8.6.2.1 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.

8.6.2.2 template<typename V > void mln::accu::histo< V >::take (const argument & *t*) [inline]

Manipulators.

8.6.2.3 void mln::Accumulator< histo< V > >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.6.2.4 void mln::Accumulator< histo< V > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.6.2.5 template<typename V > const std::vector< unsigned > & mln::accu::histo< V >::vect () const [inline]

Get the value of the accumulator.

8.7 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.

8.7.1 Detailed Description

template<typename L> struct mln::accu::label_used< L >

References all the labels used.

The parameter *L* is the label type.

8.7.2 Member Function Documentation

8.7.2.1 template<typename L > void mln::accu::label_used< L >::init () [inline]

Initialize accumulator attributes.

8.7.2.2 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.

8.7.2.3 template<typename L > void mln::accu::label_used< L >::take (const argument & *l*) [inline]

Manipulators.

8.7.2.4 void mln::Accumulator< label_used< L > >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.7.2.5 void mln::Accumulator< label_used< L > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.7.2.6 template<typename L > const fun::i2v::array< bool > & mln::accu::label_used< L >::to_result () const [inline]

Get the value of the accumulator.

8.8 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.

8.8.1 Detailed Description

"Logical-and" accumulator.

8.8.2 Member Function Documentation

8.8.2.1 void mln::accu::logic::land::init () [inline]

Manipulators.

8.8.2.2 bool mln::accu::logic::land::is_valid () const [inline]

Check whether this accu is able to return a result.

Always true here.

8.8.2.3 void mln::Accumulator< land >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.8.2.4 void mln::Accumulator< land >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take n times the value t.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.8.2.5 bool mln::accu::logic::land::to_result () const [inline]

Get the value of the accumulator.

8.9 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.

- bool to_result () const

  Get the value of the accumulator.

- void init ()

  Manipulators.

8.9.1 Detailed Description

"Logical-and" accumulator.

Conversely to accu::logic::land, this version does not have the 'untake' method but features the 'can_stop' method.

8.9.2 Member Function Documentation

8.9.2.1 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.

8.9.2.2 void mln::accu::logic::land_basic::init () [inline]

Manipulators.

8.9.2.3 bool mln::accu::logic::land_basic::is_valid () const [inline]

Check whether this accu is able to return a result.

Always true here.

8.9.2.4 bool mln::accu::logic::land_basic::to_result () const [inline]

Get the value of the accumulator.

8.10 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.

8.10.1 Detailed Description

"Logical-or" accumulator.

8.10.2 Member Function Documentation

8.10.2.1 void mln::accu::logic::lor::init () [inline]

Manipulators.

8.10.2.2 bool mln::accu::logic::lor::is_valid () const [inline]

Check whether this accu is able to return a result.

Always true here.

8.10.2.3 void mln::Accumulator< lor >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.10.2.4 void mln::Accumulator< lor >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take n times the value t.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.10.2.5 bool mln::accu::logic::lor::to_result () const [inline]

Get the value of the accumulator.

8.11 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.

8.11.1 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.

8.11.2 Member Function Documentation

8.11.2.1 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.

8.11.2.2 void mln::accu::logic::lor_basic::init () [inline]

Manipulators.

8.11.2.3 bool mln::accu::logic::lor_basic::is_valid () const [inline]

Check whether this accu is able to return a result.

Always true here.

8.11.2.4 void mln::Accumulator< lor_basic >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.11.2.5 void mln::Accumulator< lor_basic >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.11.2.6 bool mln::accu::logic::lor_basic::to_result () const [inline]

Get the value of the accumulator.

8.12 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.

8.12.1 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.

8.12.2 Member Function Documentation

8.12.2.1 template<typename T > void mln::accu::maj_h< T >::init () [inline]

Manipulators.

8.12.2.2 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.

8.12.2.3 void mln::Accumulator< maj_h< T > >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.12.2.4 void mln::Accumulator< maj_h< T > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.12.2.5 template<typename T > const T & mln::accu::maj_h< T >::to_result () const [inline]

Get the value of the accumulator.

8.13 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.

8.13.1 Detailed Description

template<typename T> struct mln::accu::math::count< T >

Generic counter accumulator.

The parameter *T* is the type to be count.

8.13.2 Member Function Documentation

8.13.2.1 template<typename T > void mln::accu::math::count< T >::init () [inline]

Manipulators.

8.13.2.2 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.

8.13.2.3 template<typename T > void mln::accu::math::count< T >::set_value (unsigned *c*) [inline]

Force the value of the counter to *c*.

8.13.2.4 void mln::Accumulator< count< T > >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.13.2.5 void mln::Accumulator< count< T > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.13.2.6 template<typename T > unsigned mln::accu::math::count< T >::to_result () const [inline]

Get the value of the accumulator.

8.14 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.

8.14.1 Detailed Description

template<typename T> struct mln::accu::math::inf< T >

Generic inf accumulator class.

The parameter `T` is the type of values.

8.14.2 Member Function Documentation

8.14.2.1 template<typename T > void mln::accu::math::inf< T >::init () [inline]

Manipulators.

8.14.2.2 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.

8.14.2.3 void mln::Accumulator< inf< T > >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.14.2.4 void mln::Accumulator< inf< T > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.14.2.5 template<typename T > const T & mln::accu::math::inf< T >::to_result () const [inline]

Get the value of the accumulator.

8.15 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.

8.15.1 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`.

8.15.2 Member Function Documentation

8.15.2.1 template<typename T , typename S > void mln::accu::math::sum< T, S >::init () [inline]

Manipulators.

References mln::literal::zero.

8.15.2.2 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.

8.15.2.3 void mln::Accumulator< sum< T, S > >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.15.2.4 void mln::Accumulator< sum< T, S > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.15.2.5 template<typename T , typename S > const S & mln::accu::math::sum< T, S >::to_result () const [inline]

Get the value of the accumulator.

8.16 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.

8.16.1 Detailed Description

template<typename T> struct mln::accu::math::sup< T >

Generic sup accumulator class.

The parameter `T` is the type of values.

8.16.2 Member Function Documentation

8.16.2.1 template<typename T > void mln::accu::math::sup< T >::init () [inline]

Manipulators.

8.16.2.2 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.

8.16.2.3 void mln::Accumulator< sup< T > >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.16.2.4 void mln::Accumulator< sup< T > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.16.2.5 template<typename T > const T & mln::accu::math::sup< T >::to_result () const [inline]

Get the value of the accumulator.

8.17 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.

8.17.1 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.

8.17.2 Member Function Documentation

8.17.2.1 template<typename I > void mln::accu::max_site< I >::init () [inline]

Manipulators.

8.17.2.2 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.

8.17.2.3 void mln::Accumulator< max_site< I > >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.17.2.4 void mln::Accumulator< max_site< I > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.17.2.5 template<typename I > I::psite mln::accu::max_site< I >::to_result () const [inline]

Get the value of the accumulator.

8.18 mln::accu::meta::center Struct Reference

Meta accumulator for center.

`#include <center.hh>`

Inheritance diagram for mln::accu::meta::center:

8.18.1 Detailed Description

Meta accumulator for center.

8.19 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:

8.19.1 Detailed Description

Meta accumulator for count_adjacent_vertices.

8.20 mln::accu::meta::count_labels Struct Reference

Meta accumulator for count_labels.

`#include <count_labels.hh>`

Inheritance diagram for mln::accu::meta::count_labels:

8.20.1 Detailed Description

Meta accumulator for count_labels.

8.21 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:

8.21.1 Detailed Description

FIXME: How to write a meta accumulator with a constructor taking a generic argument? Meta accumulator for count_value.

8.22 mln::accu::meta::histo Struct Reference

Meta accumulator for histo.

`#include <histo.hh>`

Inheritance diagram for mln::accu::meta::histo:

8.22.1 Detailed Description

Meta accumulator for histo.

8.23 mln::accu::meta::label_used Struct Reference

Meta accumulator for label_used.

`#include <label_used.hh>`

Inheritance diagram for mln::accu::meta::label_used:

8.23.1 Detailed Description

Meta accumulator for label_used.

8.24 mln::accu::meta::logic::land Struct Reference

Meta accumulator for land.

`#include <land.hh>`

Inheritance diagram for mln::accu::meta::logic::land:

8.24.1 Detailed Description

Meta accumulator for land.

8.25 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:

8.25.1 Detailed Description

Meta accumulator for land_basic.

8.26 mln::accu::meta::logic::lor Struct Reference

Meta accumulator for lor.

`#include <lor.hh>`

Inheritance diagram for mln::accu::meta::logic::lor:

8.26.1 Detailed Description

Meta accumulator for lor.

8.27 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:

8.27.1 Detailed Description

Meta accumulator for lor_basic.

8.28 mln::accu::meta::maj_h Struct Reference

Meta accumulator for maj_h.

`#include <maj_h.hh>`

Inheritance diagram for mln::accu::meta::maj_h:

8.28.1 Detailed Description

Meta accumulator for maj_h.

8.29 mln::accu::meta::math::count Struct Reference

Meta accumulator for count.

`#include <count.hh>`

Inheritance diagram for mln::accu::meta::math::count:

8.29.1 Detailed Description

Meta accumulator for count.

8.30 mln::accu::meta::math::inf Struct Reference

Meta accumulator for inf.

`#include <inf.hh>`

Inheritance diagram for mln::accu::meta::math::inf:

8.30.1 Detailed Description

Meta accumulator for inf.

8.31 mln::accu::meta::math::sum Struct Reference

Meta accumulator for sum.

`#include <sum.hh>`

Inheritance diagram for mln::accu::meta::math::sum:

8.31.1 Detailed Description

Meta accumulator for sum.

8.32 mln::accu::meta::math::sup Struct Reference

Meta accumulator for sup.

`#include <sup.hh>`

Inheritance diagram for mln::accu::meta::math::sup:

8.32.1 Detailed Description

Meta accumulator for sup.

8.33 mln::accu::meta::max_site Struct Reference

Meta accumulator for max_site.

`#include <max_site.hh>`

Inheritance diagram for mln::accu::meta::max_site:

8.33.1 Detailed Description

Meta accumulator for max_site.

8.34 mln::accu::meta::nil Struct Reference

Meta accumulator for nil.

`#include <nil.hh>`

Inheritance diagram for mln::accu::meta::nil:

8.34.1 Detailed Description

Meta accumulator for nil.

8.35 mln::accu::meta::p< mA > Struct Template Reference

Meta accumulator for p.

`#include <p.hh>`

Inheritance diagram for mln::accu::meta::p< mA >:

8.35.1 Detailed Description

template<typename mA> struct mln::accu::meta::p< mA >

Meta accumulator for p.

8.36 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 >:

8.36.1 Detailed Description

template<typename A1, typename A2> struct mln::accu::meta::pair< A1, A2 >

Meta accumulator for pair.

8.37 mln::accu::meta::rms Struct Reference

Meta accumulator for rms.

`#include <rms.hh>`

Inheritance diagram for mln::accu::meta::rms:

8.37.1 Detailed Description

Meta accumulator for rms.

8.38 mln::accu::meta::shape::bbox Struct Reference

Meta accumulator for bbox.

`#include <bbox.hh>`

Inheritance diagram for mln::accu::meta::shape::bbox:

8.38.1 Detailed Description

Meta accumulator for bbox.

8.39 mln::accu::meta::shape::height Struct Reference

Meta accumulator for height.

`#include <height.hh>`

Inheritance diagram for mln::accu::meta::shape::height:

8.39.1 Detailed Description

Meta accumulator for height.

8.40 mln::accu::meta::shape::volume Struct Reference

Meta accumulator for volume.

`#include <volume.hh>`

Inheritance diagram for mln::accu::meta::shape::volume:

8.40.1 Detailed Description

Meta accumulator for volume.

8.41 mln::accu::meta::stat::max Struct Reference

Meta accumulator for max.

`#include <max.hh>`

Inheritance diagram for mln::accu::meta::stat::max:

8.41.1 Detailed Description

Meta accumulator for max.

8.42 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:

8.42.1 Detailed Description

Meta accumulator for max.

8.43 mln::accu::meta::stat::mean Struct Reference

Meta accumulator for mean.

`#include <mean.hh>`

Inheritance diagram for mln::accu::meta::stat::mean:

8.43.1 Detailed Description

Meta accumulator for mean.

8.44 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 >:

8.44.1 Detailed Description

template<typename T> struct mln::accu::meta::stat::median_alt< T >

Meta accumulator for median_alt.

8.45 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:

8.45.1 Detailed Description

Meta accumulator for median_h.

8.46 mln::accu::meta::stat::min Struct Reference

Meta accumulator for min.

`#include <min.hh>`

Inheritance diagram for mln::accu::meta::stat::min:

8.46.1 Detailed Description

Meta accumulator for min.

8.47 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:

8.47.1 Detailed Description

Meta accumulator for min.

8.48 mln::accu::meta::stat::rank Struct Reference

Meta accumulator for rank.

`#include <rank.hh>`

Inheritance diagram for mln::accu::meta::stat::rank:

8.48.1 Detailed Description

Meta accumulator for rank.

8.49 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:

8.49.1 Detailed Description

Meta accumulator for rank_high_quant.

8.50 mln::accu::meta::tuple< n, > Struct Template Reference

Meta accumulator for tuple.

`#include <tuple.hh>`

Inheritance diagram for mln::accu::meta::tuple< n, >:

8.50.1 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.

8.51 mln::accu::meta::val< mA > Struct Template Reference

Meta accumulator for val.

`#include <v.hh>`

Inheritance diagram for mln::accu::meta::val< mA >:

8.51.1 Detailed Description

template<typename mA> struct mln::accu::meta::val< mA >

Meta accumulator for val.

8.52 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.

8.52.1 Detailed Description

template<typename T> struct mln::accu::nil< T >

Define an accumulator that does nothing.

8.52.2 Member Function Documentation

8.52.2.1 template<typename T > void mln::accu::nil< T >::init () [inline]

Manipulators.

8.52.2.2 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.

8.52.2.3 void mln::Accumulator< nil< T > >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.52.2.4 void mln::Accumulator< nil< T > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take n times the value t.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.52.2.5 template<typename T > util::ignore mln::accu::nil< T >::to_result () const [inline]

Get the value of the accumulator.

8.53 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.

8.53.1 Detailed Description

template<typename A> struct mln::accu::p< A >

Generic p of accumulators.

The parameter `V` is the type of values.

8.53.2 Member Function Documentation

8.53.2.1 template<typename A > void mln::accu::p< A >::init () [inline]

Manipulators.

8.53.2.2 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.

8.53.2.3 void mln::Accumulator< p< A > >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.53.2.4 void mln::Accumulator< p< A > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.53.2.5 template<typename A > const A::result & mln::accu::p< A >::to_result () const [inline]

Get the value of the accumulator.

8.54 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

- 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`.
- std::pair< typename A1::result, typename A2::result > to_result () const
  Get the value of the accumulator.
- void init ()
  Manipulators.

8.54.1 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.

8.54.2 Member Function Documentation

8.54.2.1 template<typename A1 , typename A2 , typename T > void mln::accu::pair< A1, A2, T >::init () [inline]

Manipulators.

8.54.2.2 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.

8.54.2.3 void mln::Accumulator< pair< A1, A2, T > >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.54.2.4 void mln::Accumulator< pair< A1, A2, T > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.54.2.5 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.

8.55 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.

8.55.1 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.

8.55.2 Member Function Documentation

8.55.2.1 template<typename T , typename V > void mln::accu::rms< T, V >::init () [inline]

Manipulators.

References mln::literal::zero.

8.55.2.2 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.

8.55.2.3 void mln::Accumulator< rms< T, V > >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.55.2.4 void mln::Accumulator< rms< T, V > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.55.2.5 template<typename T , typename V > V mln::accu::rms< T, V >::to_result () const [inline]

Get the value of the accumulator.

8.56 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.

8.56.1 Detailed Description

template<typename P> struct mln::accu::shape::bbox< P >

Generic bounding box accumulator class.

The parameter `P` is the type of points.

8.56.2 Member Function Documentation

8.56.2.1 template<typename P > void mln::accu::shape::bbox< P >::init () [inline]

Manipulators.

8.56.2.2 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.

8.56.2.3 void mln::Accumulator< bbox< P > >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.56.2.4 void mln::Accumulator< bbox< P > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.56.2.5 template<typename P > const box< P > & mln::accu::shape::bbox< P >::to_result () const [inline]

Get the value of the accumulator.

Referenced by mln::geom::rotate().

8.57 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.

8.57.1 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.

8.57.2 Member Typedef Documentation

8.57.2.1 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

8.57.2.2 template<typename I> typedef argument::value mln::accu::shape::height< I >::value

The value type associated to the pixel type.

8.57.3 Member Function Documentation

8.57.3.1 template<typename I > void mln::accu::shape::height< I >::init () [inline]

Manipulators.

8.57.3.2 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.

8.57.3.3 template<typename I > void mln::accu::shape::height< I >::set_value (unsigned *h*) [inline]

Force the value of the counter to *h*.

8.57.3.4 void mln::Accumulator< height< I > >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.57.3.5 void mln::Accumulator< height< I > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.57.3.6 template<typename I > unsigned mln::accu::shape::height< I >::to_result () const [inline]

Get the value of the accumulator.

8.58 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.

8.58.1 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.

8.58.2 Member Typedef Documentation

8.58.2.1 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

8.58.2.2 template<typename I> typedef argument::value mln::accu::shape::volume< I >::value

The value type associated to the pixel type.

8.58.3 Member Function Documentation

8.58.3.1 template<typename I > void mln::accu::shape::volume< I >::init () [inline]

Manipulators.

References mln::literal::zero.

8.58.3.2 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.

8.58.3.3 template<typename I > void mln::accu::shape::volume< I >::set_value (unsigned *v*) [inline]

Force the value of the counter to *v*.

References mln::literal::zero.

8.58.3.4 void mln::Accumulator< volume< I > >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.58.3.5 void mln::Accumulator< volume< I > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.58.3.6 template<typename I > unsigned mln::accu::shape::volume< I >::to_result () const [inline]

Get the value of the accumulator.

8.59 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.
- 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`.
- result to_result () const
  Return the rectangularity value.

8.59.1 Detailed Description

template<typename P> class mln::accu::site_set::rectangularity< P >

Compute the rectangularity of a site set.

8.59.2 Constructor & Destructor Documentation

8.59.2.1 template<typename P > mln::accu::site_set::rectangularity< P >::rectangularity () [inline]

Constructor.

8.59.3 Member Function Documentation

8.59.3.1 template<typename P > rectangularity< P >::A2::result mln::accu::site_set::rectangularity< P >::area () const [inline]

Return the site set area.

8.59.3.2 template<typename P > rectangularity< P >::A1::result mln::accu::site_set::rectangularity< P >::bbox () const [inline]

Return the site set bounding box.

8.59.3.3 void mln::Accumulator< rectangularity< P > >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.59.3.4 void mln::Accumulator< rectangularity< P > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.59.3.5 template<typename P > rectangularity< P >::result mln::accu::site_set::rectangularity< P >::to_result () const [inline]

Return the rectangularity value.

8.60 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.

8.60.1 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`.

8.60.2 Member Function Documentation

8.60.2.1 template<typename T , typename S , typename M > void mln::accu::stat::deviation< T, S, M >::init () [inline]

Manipulators.

8.60.2.2 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.

8.60.2.3 void mln::Accumulator< deviation< T, S, M > >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.60.2.4 void mln::Accumulator< deviation< T, S, M > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.60.2.5 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.

8.61 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 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.

8.61.1 Detailed Description

template<typename T> struct mln::accu::stat::max< T >

Generic max accumulator class.

The parameter `T` is the type of values.

8.61.2 Member Function Documentation

8.61.2.1 template<typename T > void mln::accu::stat::max< T >::init () [inline]

Manipulators.

8.61.2.2 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.

8.61.2.3 void mln::Accumulator< max< T > >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.61.2.4 void mln::Accumulator< max< T > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.61.2.5 template<typename T > const T & mln::accu::stat::max< T >::to_result () const [inline]

Get the value of the accumulator.

8.62 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.

8.62.1 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.

8.62.2 Member Function Documentation

8.62.2.1 template<typename V > void mln::accu::stat::max_h< V >::init () [inline]

Manipulators.

8.62.2.2 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.

8.62.2.3 void mln::Accumulator< max_h< V > >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.62.2.4 void mln::Accumulator< max_h< V > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take n times the value t.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.62.2.5 template<typename V > const max_h< V >::argument & mln::accu::stat::max_h< V >::to_result () const [inline]

Get the value of the accumulator.

8.63 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

- 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.

8.63.1 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`.

8.63.2 Member Function Documentation

8.63.2.1 template<typename T , typename S , typename M > void mln::accu::stat::mean< T, S, M >::init () `[inline]`

Manipulators.

8.63.2.2 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.

8.63.2.3 void mln::Accumulator< mean< T, S, M > >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.63.2.4 void mln::Accumulator< mean< T, S, M > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.63.2.5 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.

8.64 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.

8.64.1 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`.

8.64.2 Member Function Documentation

8.64.2.1 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.

8.64.2.2 template<typename S > void mln::accu::stat::median_alt< S >::take (const argument & *t*) [inline]

Manipulators.

8.64.2.3 void mln::Accumulator< median_alt< S > >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.64.2.4 void mln::Accumulator< median_alt< S > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.64.2.5 template<typename S > const median_alt< S >::argument & mln::accu::stat::median_alt< S >::to_result () const [inline]

Get the value of the accumulator.

8.65 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 `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.

8.65.1 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.

8.65.2 Member Function Documentation

8.65.2.1 template<typename V > void mln::accu::stat::median_h< V >::init () [inline]

Manipulators.

8.65.2.2 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.

8.65.2.3 void mln::Accumulator< median_h< V > >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.65.2.4 void mln::Accumulator< median_h< V > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.65.2.5 template<typename V > const median_h< V >::argument & mln::accu::stat::median_h< V >::to_result () const [inline]

Get the value of the accumulator.

8.66 mln::accu::stat::meta::deviation Struct Reference

Meta accumulator for deviation.

`#include <deviation.hh>`

Inheritance diagram for mln::accu::stat::meta::deviation:

8.66.1 Detailed Description

Meta accumulator for deviation.

8.67 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 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.

8.67.1 Detailed Description

template<typename T> struct mln::accu::stat::min< T >

Generic min accumulator class.

The parameter `T` is the type of values.

8.67.2 Member Function Documentation

8.67.2.1 template<typename T > void mln::accu::stat::min< T >::init () [inline]

Manipulators.

8.67.2.2 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.

8.67.2.3 void mln::Accumulator< min< T > >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.67.2.4 void mln::Accumulator< min< T > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.67.2.5 template<typename T > const T & mln::accu::stat::min< T >::to_result () const [inline]

Get the value of the accumulator.

8.68 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.

8.68.1 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.

8.68.2 Member Function Documentation

8.68.2.1 template<typename V > void mln::accu::stat::min_h< V >::init () [inline]

Manipulators.

8.68.2.2 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.

8.68.2.3 void mln::Accumulator< min_h< V > >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.68.2.4 void mln::Accumulator< min_h< V > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take n times the value t.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.68.2.5 template<typename V > const min_h< V >::argument & mln::accu::stat::min_h< V >::to_result () const [inline]

Get the value of the accumulator.

8.69 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

- 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`.
- std::pair< typename min< V >::result, typename max< V >::result > to_result () const
  Get the value of the accumulator.
- void init ()
  Manipulators.

8.69.1 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.

8.69.2 Member Function Documentation

8.69.2.1 void mln::accu::pair< min< V > , max< V > , mln_argument(min< V >) >::init () [inherited]

Manipulators.

8.69.2.2 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.

8.69.2.3 void mln::Accumulator< pair< min< V >, max< V >, mln_argument(min< V >) > >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.69.2.4 void mln::Accumulator< pair< min< V >, max< V >, mln_argument(min< V >) > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.69.2.5 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.

8.70 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.

8.70.1 Detailed Description

template<typename T> struct mln::accu::stat::rank< T >

Generic rank accumulator class.

The parameter `T` is the type of values.

8.70.2 Member Function Documentation

8.70.2.1 template<typename T > void mln::accu::stat::rank< T >::init () [inline]

Manipulators.

Referenced by mln::morpho::impl::generic::rank_filter().

8.70.2.2 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.

8.70.2.3 template<typename T > unsigned mln::accu::stat::rank< T >::k () const [inline]

Give the rank.

8.70.2.4 void mln::Accumulator< rank< T > >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.70.2.5 void mln::Accumulator< rank< T > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.70.2.6 template<typename T > const T & mln::accu::stat::rank< T >::to_result () const [inline]

Get the value of the accumulator.

8.71 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.

8.71.1 Detailed Description

template<> struct mln::accu::stat::rank< bool >

rank accumulator class for Boolean.

8.71.2 Member Function Documentation

8.71.2.1 void mln::accu::stat::rank< bool >::init () `[inline]`

Manipulators.

8.71.2.2 bool mln::accu::stat::rank< bool >::is_valid () const `[inline]`

Check whether this accu is able to return a result.

Always true here.

8.71.2.3 void mln::Accumulator< rank< bool > >::take_as_init (const T & *t*) `[inline, inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.71.2.4 void mln::Accumulator< rank< bool > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.71.2.5 bool mln::accu::stat::rank< bool >::to_result () const [inline]

Get the value of the accumulator.

8.72 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.

8.72.1 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.

8.72.2 Member Function Documentation

8.72.2.1 template<typename T > void mln::accu::stat::rank_high_quant< T >::init () [inline]

Manipulators.

8.72.2.2 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.

8.72.2.3 void mln::Accumulator< rank_high_quant< T > >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.72.2.4 void mln::Accumulator< rank_high_quant< T > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.72.2.5 template<typename T > const T & mln::accu::stat::rank_high_quant< T >::to_result () const [inline]

Get the value of the accumulator.

8.73 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 `t`.

- void take_n_times (unsigned n, const T &t)

  Take `n` *times the value* `t`.

- result to_result () const

  Get the accumulator result (the var value).

- result variance () const

  Get the variance matrix.

- void init ()

  Manipulators.

8.73.1 Detailed Description

template<typename T> struct mln::accu::stat::var< T >

Var accumulator class.

Parameter `T` is the type of vectors

8.73.2 Member Typedef Documentation

8.73.2.1 template<typename T> typedef algebra::vec<dim,float> mln::accu::stat::var< T >::mean_t

Type equipment.

8.73.3 Member Function Documentation

8.73.3.1 template<typename T > void mln::accu::stat::var< T >::init () [inline]

Manipulators.

8.73.3.2 template<typename T > bool mln::accu::stat::var< T >::is_valid () const [inline]

Check whether this accu returns a valid result.

8.73.3.3 template<typename T > var< T >::mean_t mln::accu::stat::var< T >::mean () const [inline]

Get the mean vector.

References mln::literal::zero.

8.73.3.4 template<typename T > unsigned mln::accu::stat::var< T >::n_items () const [inline]

Get the number of items.

8.73.3.5 void mln::Accumulator< var< T > >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.73.3.6 void mln::Accumulator< var< T > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.73.3.7 template<typename T > var< T >::result mln::accu::stat::var< T >::to_result () const [inline]

Get the accumulator result (the var value).

References mln::literal::zero.

8.73.3.8 template<typename T > var< T >::result mln::accu::stat::var< T >::variance () const [inline]

Get the variance matrix.

8.74 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_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.
- void take_as_init (const argument &t)
  Take as initialization the value `t`.

8.74.1 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`.

8.74.2 Member Function Documentation

8.74.2.1 template<typename T , typename S , typename R > void mln::accu::stat::variance< T, S, R >::init () [inline]

Manipulators.

8.74.2.2 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.

8.74.2.3 template<typename T , typename S , typename R > R mln::accu::stat::variance< T, S, R >::mean () const [inline]

Get the mean value.

8.74.2.4 template<typename T , typename S , typename R > unsigned mln::accu::stat::variance< T, S, R >::n_items () const [inline]

Get the number of items.

8.74.2.5 template<typename T , typename S , typename R > R mln::accu::stat::variance< T, S, R >::standard_deviation () const [inline]

Get the standard deviation value.

References mln::accu::stat::variance< T, S, R >::to_result().

8.74.2.6 template<typename T , typename S , typename R > S mln::accu::stat::variance< T, S, R >::sum () const [inline]

Get the sum value.

8.74.2.7 template<typename T , typename S , typename R > void mln::accu::stat::variance< T, S, R >::take_as_init (const argument & *t*) [inline]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

Reimplemented from mln::Accumulator< variance< T, S, R > >.

8.74.2.8 void mln::Accumulator< variance< T, S, R > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.74.2.9 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).

Referenced by mln::accu::stat::variance< T, S, R >::standard_deviation(), and mln::accu::stat::variance< T, S, R >::var().

8.74.2.10 template<typename T , typename S , typename R > R mln::accu::stat::variance< T, S, R >::var () const [inline]

Get the variance value.

References mln::accu::stat::variance< T, S, R >::to_result().

8.75 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.

8.75.1 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.

8.75.2 Member Function Documentation

8.75.2.1 template<typename A , unsigned n, BOOST_PP_ENUM_PARAMS(10, typename T) > void mln::accu::tuple< A, n, >::init () [inline]

Manipulators.

8.75.2.2 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.

8.75.2.3 void mln::Accumulator< tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T)> >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.75.2.4 void mln::Accumulator< tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T)> >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.75.2.5 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.

8.76 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.

8.76.1 Detailed Description

template<typename A> struct mln::accu::val< A >

Generic val of accumulators.

8.76.2 Member Function Documentation

8.76.2.1 template<typename A > void mln::accu::val< A >::init () [inline]

Manipulators.

8.76.2.2 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.

8.76.2.3 void mln::Accumulator< val< A > >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.76.2.4 void mln::Accumulator< val< A > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take n times the value t.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.76.2.5 template<typename A > const A::result & mln::accu::val< A >::to_result () const [inline]

Get the value of the accumulator.

8.77 mln::Accumulator< E > Struct Template Reference

Base class for implementation of accumulators.

`#include <accumulator.hh>`

Inherits Proxy< 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`.

8.77.1 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.

8.77.2 Member Function Documentation

8.77.2.1 template<typename E > template<typename T > void mln::Accumulator< E >::take_as_init (const T & *t*) [inline]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

Reimplemented in mln::accu::stat::variance< T, S, R >.

8.77.2.2 template<typename E > template<typename T > void mln::Accumulator< E >::take_n_times (unsigned *n*, const T & *t*) [inline]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.78 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 (const mat< d+1, d+1, T > &x)

  Constructor with the underlying matrix.

- h_mat ()

  Constructor without argument.

- mat< m, n, T > t () const

  Return the transpose of the matrix.

8.78.1 Detailed Description

template<unsigned d, typename T> struct mln::algebra::h_mat< d, T >

N-Dimensional matrix with homogeneous coordinates.

8.78.2 Member Enumeration Documentation

8.78.2.1 template<unsigned d, typename T> anonymous enum

Dimension is the 'natural' one (3 for 3D), not the one of the vector (dim + 1).

8.78.3 Constructor & Destructor Documentation

8.78.3.1 template<unsigned d, typename T > mln::algebra::h_mat< d, T >::h_mat () [inline]

Constructor without argument.

8.78.3.2 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.

8.78.4 Member Function Documentation

8.78.4.1 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.

8.78.4.2 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.

8.79 mln::algebra::h_vec< d, C > Struct 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 (const vec< d+1, C > &other)

  Constructor with the underlying vector.

- h_vec ()

  Constructor without argument.

- 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.

8.79.1 Detailed Description

template<unsigned d, typename C> struct mln::algebra::h_vec< d, C >

N-Dimensional vector with homogeneous coordinates.

8.79.2 Member Enumeration Documentation

8.79.2.1 template<unsigned d, typename C> anonymous enum

Dimension is the 'natural' one (3 for 3D), not the one of the vector (dim + 1).

8.79.3 Constructor & Destructor Documentation

8.79.3.1 template<unsigned d, typename C > mln::algebra::h_vec< d, C >::h_vec () [inline]

Constructor without argument.

References mln::literal::one.

8.79.3.2 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.

8.79.4 Member Function Documentation

8.79.4.1 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.

8.79.4.2 template<unsigned n, typename T > mat< 1, n, T > mln::algebra::vec< n, T >::t () const [inline, inherited]

Transposition.

8.79.4.3 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.

8.79.5 Member Data Documentation

8.79.5.1 template<unsigned n, typename T> const vec< n, T > mln::algebra::vec< n, T >::origin = all_to(0) [inline, static, inherited]

Origin value.

8.79.5.2 template<unsigned n, typename T> const vec< n, T > mln::algebra::vec< n, T >::zero = all_to(0) [inline, static, inherited]

Zero value.

8.80 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.

8.80.1 Detailed Description

template<typename I> class mln::bkd_pixter1d< I >

Backward pixel iterator on a 1-D image with border.

8.80.2 Member Typedef Documentation

8.80.2.1 template<typename I > typedef I mln::bkd_pixter1d< I >::image

Image type.

8.80.3 Constructor & Destructor Documentation

8.80.3.1 template<typename I > mln::bkd_pixter1d< I >::bkd_pixter1d (I & *image*) [inline]

Constructor.

Parameters:

← ***image*** The image this pixel iterator is bound to.

8.80.4 Member Function Documentation

8.80.4.1 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.

8.81 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.

8.81.1 Detailed Description

template<typename I> class mln::bkd_pixter2d< I >

Backward pixel iterator on a 2-D image with border.

8.81.2 Member Typedef Documentation

8.81.2.1 template<typename I > typedef I mln::bkd_pixter2d< I >::image

Image type.

8.81.3 Constructor & Destructor Documentation

8.81.3.1 template<typename I > mln::bkd_pixter2d< I >::bkd_pixter2d (I & *image*) [inline]

Constructor.

Parameters:

← ***image*** The image this pixel iterator is bound to.

8.81.4 Member Function Documentation

8.81.4.1 void mln::Iterator< bkd_pixter2d< I > >::next () [inherited]

Go to the next element.

Warning:

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition:

The iterator is valid.

8.82 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.

8.82.1 Detailed Description

template<typename I> class mln::bkd_pixter3d< I >

Backward pixel iterator on a 3-D image with border.

8.82.2 Member Typedef Documentation

8.82.2.1 template<typename I > typedef I mln::bkd_pixter3d< I >::image

Image type.

8.82.3 Constructor & Destructor Documentation

8.82.3.1 template<typename I > mln::bkd_pixter3d< I >::bkd_pixter3d (I & *image*) [inline]

Constructor.

Parameters:

← ***image*** The image this pixel iterator is bound to.

8.82.4 Member Function Documentation

8.82.4.1 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.

8.83 mln::box< P > Struct 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.

- 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 (const site &pmin, const site &pmax)

  Constructor of a box going from `pmin` *to* `pmax`.

- box ()

  Constructor without argument.

- P center () const

  Return the approximated central site of this box.

- void crop_wrt (const box< P > &b)

  Crop this bbox in order to fit in the reference box `b`.

- void enlarge (unsigned dim, unsigned b)

  Enlarge the box with a border `b` *for dimension* `dim`.

- void enlarge (unsigned b)

  Enlarge the box with a border `b`.

- 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.

- unsigned nsites () const

Give the number of sites of this box.

- P & pmax ()
  Reference to the maximum point.
- P pmax () const
  Maximum point.
- P & pmin ()
  Reference to the minimum point.
- P pmin () const
  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`.

8.83.1 Detailed Description

template<typename P> struct mln::box< P >

Generic box class: site set containing points of a regular grid.

Parameter `P` is the corresponding type of point.

8.83.2 Member Typedef Documentation

8.83.2.1 template<typename P> typedef box_bkd_piter_<P> mln::box< P >::bkd_piter

Backward Site_Iterator associated type.

8.83.2.2 template<typename P> typedef P mln::box< P >::element

Element associated type.

8.83.2.3 template<typename P> typedef box_fwd_piter_<P> mln::box< P >::fwd_piter

Forward Site_Iterator associated type.

8.83.2.4 template<typename P> typedef fwd_piter mln::box< P >::piter

Site_Iterator associated type.

8.83.2.5 template<typename P> typedef P mln::box< P >::psite

Psite associated type.

8.83.2.6 template<typename P> typedef P mln::box< P >::site

Site associated type.

8.83.3 Member Enumeration Documentation

8.83.3.1 template<typename P> anonymous enum

Dimension.

8.83.4 Constructor & Destructor Documentation

8.83.4.1 template<typename P > mln::box< P >::box () [inline]

Constructor without argument.

8.83.4.2 template<typename P > mln::box< P >::box (const site & *pmin*, const site & *pmax*) [inline]

Constructor of a box going from `pmin` to `pmax`.

References mln::box< P >::is_valid().

8.83.4.3 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.

References mln::literal::origin.

8.83.5 Member Function Documentation

8.83.5.1 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.

8.83.5.2 template<typename P > P mln::box< P >::center () const [inline]

Return the approximated central site of this box.

References mln::box< P >::is_valid().

8.83.5.3 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.

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

Referenced by mln::make_debug_graph_image().

8.83.5.4 template<typename P > void mln::box< P >::enlarge (unsigned *dim*, unsigned *b*) [inline]

Enlarge the box with a border b for dimension dim.

References mln::box< P >::is_valid().

8.83.5.5 template<typename P > void mln::box< P >::enlarge (unsigned *b*) [inline]

Enlarge the box with a border b.

References mln::box< P >::is_valid().

Referenced by mln::registration::icp().

8.83.5.6 template<typename P > bool mln::box< P >::has (const P & *p*) const [inline]

Test if p belongs to the box.

Parameters:

← ***p*** A point site.

References mln::box< P >::is_valid().

Referenced by mln::morpho::line_gradient().

8.83.5.7 bool mln::Box< box< P > >::is_empty () const [inherited]

Test if this box is empty.

8.83.5.8 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.

References mln::util::ord_weak().

Referenced by mln::box< P >::box(), mln::box< P >::center(), mln::transform::distance_and_closest_point_geodesic(), mln::box< P >::enlarge(), mln::box< P >::has(), mln::box< P >::pmax(), mln::box< P >::pmin(), and mln::box< P >::to_larger().

8.83.5.9 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.

8.83.5.10 template<typename P > std::size_t mln::box< P >::memory_size () const [inline]

Return the size of this site set in memory.

8.83.5.11 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.

8.83.5.12 template<typename P > P & mln::box< P >::pmax () [inline]

Reference to the maximum point.

8.83.5.13 template<typename P > P mln::box< P >::pmax () const [inline]

Maximum point.

References mln::box< P >::is_valid().

Referenced by mln::box< P >::crop_wrt(), mln::make::image3d(), and mln::larger_than().

8.83.5.14 template<typename P > P & mln::box< P >::pmin () [inline]

Reference to the minimum point.

8.83.5.15 template<typename P > P mln::box< P >::pmin () const [inline]

Minimum point.

References mln::box< P >::is_valid().

Referenced by mln::box< P >::crop_wrt(), mln::make::image3d(), and mln::larger_than().

8.83.5.16 template<typename P > box< P > mln::box< P >::to_larger (unsigned *b*) const [inline]

Give a larger box.

References mln::box< P >::is_valid().

8.83.6 Friends And Related Function Documentation

8.83.6.1 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:

↔ ***ostr*** An output stream.

← ***b*** A generic box.

Returns:

The modified output stream `ostr`.

8.84 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 Bl , typename Br >
  bool operator< (const Box< Bl > &lhs, const Box< Br > &rhs)

  Strict inclusion test between boxes `lhs` *and* `rhs`.

- template<typename Bl , typename Br >
  bool operator<= (const Box< Bl > &lhs, const Box< Br > &rhs)

  Inclusion test between boxes `lhs` *and* `rhs`.

8.84.1 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.

8.84.2 Member Function Documentation

8.84.2.1 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.

8.84.2.2 template<typename E > bool mln::Box< E >::is_empty () const [inline]

Test if this box is empty.

8.84.2.3 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 < site::dim

Warning:

This method is final for all box classes.

8.84.2.4 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.

Referenced by mln::morpho::line_gradient().

8.84.3 Friends And Related Function Documentation

8.84.3.1 template<typename Bl , typename Br > bool operator< (const Box< Bl > & *lhs*, const Box< Br > & *rhs*) [related]

Strict inclusion test between boxes `lhs` and `rhs`.

Parameters:

← ***lhs*** A box (strictly included?).

← ***rhs*** Another box (includor?).

8.84.3.2 template<typename Bl , typename Br > bool operator<= (const Box< Bl > & *lhs*, const Box< Br > & *rhs*) [related]

Inclusion test between boxes `lhs` and `rhs`.

Parameters:

← ***lhs*** A box (included?).

← ***rhs*** Another box (includor?).

8.85 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 lenght of the run.

8.85.1 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.

8.85.2 Constructor & Destructor Documentation

8.85.2.1 template<typename P > mln::box_runstart_piter< P >::box_runstart_piter (const box< P > & *b*) [inline]

Constructor.

Parameters:

← ***b*** A box.

8.85.3 Member Function Documentation

8.85.3.1 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.

8.85.3.2 template<typename P > unsigned mln::box_runstart_piter< P >::run_length () const [inline]

Give the lenght of the run.

8.86 mln::Browsing< E > Struct Template Reference

Base class for implementation classes that are browsings.

`#include <browsing.hh>`

Inheritance diagram for mln::Browsing< E >:

8.86.1 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.

8.87 mln::canvas::browsing::backdiagonal2d_t Struct Reference

Browsing in a certain direction.

`#include <backdiagonal2d.hh>`

Inheritance diagram for mln::canvas::browsing::backdiagonal2d_t:

8.87.1 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 fonctor).
- 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

8.88 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 >.

8.88.1 Detailed Description

Breadth-first search algorithm for graph, on vertices.

8.89 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 >.

8.89.1 Detailed Description

Breadth-first search algorithm for graph, on vertices.

8.90 mln::canvas::browsing::diagonal2d_t Struct Reference

Browsing in a certain direction.

`#include <diagonal2d.hh>`

Inheritance diagram for mln::canvas::browsing::diagonal2d_t:

8.90.1 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 fonctor).
- 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——>

8.91 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:

8.91.1 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();

}

8.92 mln::canvas::browsing::directional_t Struct Reference

Browsing in a certain direction.

`#include <directional.hh>`

Inheritance diagram for mln::canvas::browsing::directional_t:

8.92.1 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 fonctor).
- 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

8.93 mln::canvas::browsing::fwd_t Struct Reference

Canvas for forward browsing.

`#include <fwd.hh>`

Inheritance diagram for mln::canvas::browsing::fwd_t:

8.93.1 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 fonctor 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();

}

8.94 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:

8.94.1 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 fonctor).
- 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();

}

8.95 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:

8.95.1 Detailed Description

Browsing in a snake-way, forward.

This canvas browse all the point of an image 'input' like this :

——-> <——' '——>

The fonctor 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 ba called after each moving left.

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 fwd();

void bkd();

}

8.96 mln::canvas::browsing::snake_generic_t Struct Reference

Multidimentional Browsing in a given-way.

`#include <snake_generic.hh>`

Inheritance diagram for mln::canvas::browsing::snake_generic_t:

8.96.1 Detailed Description

Multidimentional 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 3nd dimension to follow (forward). dps[3] = delta psite following the 3nd dimension to follow (backward).

moves contains pointer to f's members. These merbers 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]

8.97 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:

8.97.1 Detailed Description

Browsing in a snake-way, forward.

This canvas browse all the point of an image 'input' like this :

| /\ | | | | \/ | \/

The fonctor 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 ba 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();

}

8.98 mln::canvas::chamfer< F > Struct Template Reference

Compute chamfer distance.

`#include <chamfer.hh>`

8.98.1 Detailed Description

template<typename F> struct mln::canvas::chamfer< F >

Compute chamfer distance.

8.99 mln::category< R(*)(A) > Struct Template Reference

Category declaration for a unary C function.

`#include <c.hh>`

8.99.1 Detailed Description

template<typename R, typename A> struct mln::category< R(*)(A) >

Category declaration for a unary C function.

8.100 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 std::vector< V >::reference lvalue
  Return type of read-write access.
- typedef std::vector< V >::const_reference 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

- lvalue operator() (const complex_psite< D, G > &p)
  Read-write access of face value at point site p.
- rvalue operator() (const complex_psite< D, G > &p) const
  Read-only 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< 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.

8.100.1 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.

8.100.2 Member Typedef Documentation

8.100.2.1 template<unsigned D, typename G, typename V> typedef G mln::complex_image< D, G, V >::geom

The geometry type of the complex.

8.100.2.2 template<unsigned D, typename G, typename V> typedef std::vector<V>::reference mln::complex_image< D, G, V >::lvalue

Return type of read-write access.

We use the associated type `reference` instead of a plain reference on the value type (`V`), because it's the only way to safely form a reference on the element in the case of a std::vector<bool>.

8.100.2.3 template<unsigned D, typename G, typename V> typedef std::vector<V>::const_reference mln::complex_image< D, G, V >::rvalue

Return type of read-only access.

8.100.2.4 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.

8.100.2.5 template<unsigned D, typename G, typename V> typedef V mln::complex_image< D, G, V >::value

Value associated type.

8.100.3 Constructor & Destructor Documentation

8.100.3.1 template<unsigned D, typename G , typename V > mln::complex_image< D, G, V >::complex_image () `[inline]`

Constructors.

8.100.4 Member Function Documentation

8.100.4.1 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 od the image.

8.100.4.2 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.

References mln::complex_psite< D, G >::face_id(), and mln::complex_psite< D, G >::n().

8.100.4.3 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.

References mln::complex_psite< D, G >::face_id(), and mln::complex_psite< D, G >::n().

8.100.4.4 template<unsigned D, typename G , typename V > const metal::vec< D+1, std::vector< V > > & mln::complex_image< D, G, V >::values () const [inline]

Return the array of values associated to the faces.

8.100.5 Member Data Documentation

8.100.5.1 template<unsigned D, typename G, typename V> const unsigned mln::complex_image< D, G, V >::dim = D [static]

The dimension of the complex.

8.101 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.

8.101.1 Detailed Description

template<typename I, typename G, typename N> class mln::complex_neighborhood_bkd_piter< I, G, N >

Backward iterator on complex neighborhood.

8.101.2 Member Typedef Documentation

8.101.2.1 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.

8.101.2.2 template<typename I, typename G, typename N> typedef N ::psite mln::complex_neighborhood_bkd_piter< I, G, N >::psite

The Pseudo_Site type.

8.101.3 Constructor & Destructor Documentation

8.101.3.1 template<typename I , typename G , typename N > mln::complex_neighborhood_bkd_piter< I, G, N >::complex_neighborhood_bkd_piter () [inline]

Construction.

8.101.4 Member Function Documentation

8.101.4.1 template<typename I , typename G , typename N > const N::complex_bkd_iter & mln::complex_neighborhood_bkd_piter< I, G, N >::iter () const [inline]

Accessors.

8.101.4.2 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.

8.102 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.

8.102.1 Detailed Description

template<typename I, typename G, typename N> class mln::complex_neighborhood_fwd_piter< I, G, N >

Forward iterator on complex neighborhood.

8.102.2 Member Typedef Documentation

8.102.2.1 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.

8.102.2.2 template<typename I, typename G, typename N> typedef N ::psite mln::complex_neighborhood_fwd_piter< I, G, N >::psite

The Pseudo_Site type.

8.102.3 Constructor & Destructor Documentation

8.102.3.1 template<typename I , typename G , typename N > mln::complex_neighborhood_fwd_piter< I, G, N >::complex_neighborhood_fwd_piter () [inline]

Construction.

8.102.4 Member Function Documentation

8.102.4.1 template<typename I , typename G , typename N > const N::complex_fwd_iter & mln::complex_neighborhood_fwd_piter< I, G, N >::iter () const [inline]

Accessors.

8.102.4.2 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.

8.103 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

- void change_target (const target &new_target)
  Set the target site_set.
- const target & site_set () const
  Site set manipulators.
- complex_psite (const p_complex< D, G > &pc, const topo::face< D > &face)
- complex_psite ()
  Construction and assignment.
- const topo::face< D > & face () const
  Face handle manipulators.
- unsigned face_id () const
  Return the id of the face of this psite.
- unsigned n () const
  Return the dimension of the face of this psite.
- void invalidate ()
  Invalidate this psite.
- bool is_valid () const
  Psite manipulators.

8.103.1 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.

8.103.2 Constructor & Destructor Documentation

8.103.2.1 template<unsigned D, typename G > mln::complex_psite< D, G >::complex_psite () [inline]

Construction and assignment.

References mln::complex_psite< D, G >::invalidate().

8.103.2.2 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().

References mln::topo::face< D >::cplx(), mln::p_complex< D, G >::cplx(), and mln::complex_psite< D, G >::is_valid().

8.103.3 Member Function Documentation

8.103.3.1 template<unsigned D, typename G > void mln::complex_psite< D, G >::change_target (const target & *new_target*) [inline]

Set the target site_set.

References mln::p_complex< D, G >::cplx(), and mln::complex_psite< D, G >::invalidate().

8.103.3.2 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.

Referenced by mln::operator!=(), and mln::operator==().

8.103.3.3 template<unsigned D, typename G > unsigned mln::complex_psite< D, G >::face_id () const [inline]

Return the id of the face of this psite.

Referenced by mln::complex_image< D, G, V >::operator()().

8.103.3.4 template<unsigned D, typename G > void mln::complex_psite< D, G >::invalidate () [inline]

Invalidate this psite.

Referenced by mln::complex_psite< D, G >::change_target(), and mln::complex_psite< D, G >::complex_psite().

8.103.3.5 template<unsigned D, typename G > bool mln::complex_psite< D, G >::is_valid () const [inline]

Psite manipulators.

Is this psite valid?

Referenced by mln::complex_psite< D, G >::complex_psite(), and mln::p_complex< D, G >::has().

8.103.3.6 template<unsigned D, typename G > unsigned mln::complex_psite< D, G >::n () const [inline]

Return the dimension of the face of this psite.

Referenced by mln::make::cell(), and mln::complex_image< D, G, V >::operator()().

8.103.3.7 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.

Referenced by mln::p_complex< D, G >::has(), mln::operator!=(), and mln::operator==().

8.104 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.

8.104.1 Detailed Description

template<typename I, typename G, typename W> class mln::complex_window_bkd_piter< I, G, W >

Backward iterator on complex window.

8.104.2 Member Typedef Documentation

8.104.2.1 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.

8.104.2.2 template<typename I, typename G, typename W> typedef W ::psite mln::complex_window_bkd_piter< I, G, W >::psite

The Pseudo_Site type.

8.104.3 Constructor & Destructor Documentation

8.104.3.1 template<typename I , typename G , typename W > mln::complex_window_bkd_piter< I, G, W >::complex_window_bkd_piter () [inline]

Construction.

8.104.4 Member Function Documentation

8.104.4.1 template<typename I , typename G , typename W > const W::complex_bkd_iter & mln::complex_window_bkd_piter< I, G, W >::iter () const [inline]

Accessors.

8.104.4.2 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.

8.105 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.

8.105.1 Detailed Description

template<typename I, typename G, typename W> class mln::complex_window_fwd_piter< I, G, W >

Forward iterator on complex window.

8.105.2 Member Typedef Documentation

8.105.2.1 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.

8.105.2.2 template<typename I, typename G, typename W> typedef W ::psite mln::complex_window_fwd_piter< I, G, W >::psite

The Pseudo_Site type.

8.105.3 Constructor & Destructor Documentation

8.105.3.1 template<typename I , typename G , typename W > mln::complex_window_fwd_piter< I, G, W >::complex_window_fwd_piter () [inline]

Construction.

8.105.4 Member Function Documentation

8.105.4.1 template<typename I , typename G , typename W > const W::complex_fwd_iter & mln::complex_window_fwd_piter< I, G, W >::iter () const [inline]

Accessors.

8.105.4.2 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.

8.106 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 > >.

Package 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.

Package 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.
- lvalue operator() (const psite &p)
  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.

8.106.1 Detailed Description

template<typename I, typename D> struct mln::decorated_image< I, D >

Image that can have additional features.

8.106.2 Member Typedef Documentation

8.106.2.1 template<typename I, typename D> typedef impl_::lvalue mln::decorated_image< I, D >::lvalue [package]

Return type of read-write access.

8.106.2.2 template<typename I, typename D> typedef I ::psite mln::decorated_image< I, D >::psite [package]

Type of the psite.

8.106.2.3 template<typename I, typename D> typedef I ::rvalue mln::decorated_image< I, D >::rvalue [package]

Return type of read-only access.

8.106.2.4 template<typename I, typename D> typedef decorated_image< tag::image_<I>, tag::data_<D> > mln::decorated_image< I, D >::skeleton [package]

Skeleton.

8.106.3 Constructor & Destructor Documentation

8.106.3.1 template<typename I , typename D > mln::decorated_image< I, D >::decorated_image () [inline, package]

Ctors.

8.106.4 Member Function Documentation

8.106.4.1 template<typename I , typename D > D & mln::decorated_image< I, D >::decoration () [inline, package]

Give the decoration.

8.106.4.2 template<typename I , typename D > const D & mln::decorated_image< I, D >::decoration () const [inline, package]

Give the decoration.

8.106.4.3 template<typename I , typename D > mln::decorated_image< I, D >::operator decorated_image< const I, D > () const [inline, package]

Const promotion via conversion.

8.106.4.4 template<typename I , typename D > decorated_image< I, D >::lvalue mln::decorated_image< I, D >::operator() (const psite & *p*) [inline, package]

Read-write access of pixel value at point site p.

8.106.4.5 template<typename I , typename D > decorated_image< I, D >::rvalue mln::decorated_image< I, D >::operator() (const psite & *p*) const [inline, package]

Read-only access of pixel value at point site p.

8.107 mln::Delta_Point_Site< E > Struct Template Reference

FIXME: Doc!

`#include <delta_point_site.hh>`

Inheritance diagram for mln::Delta_Point_Site< E >:

8.107.1 Detailed Description

template<typename E> struct mln::Delta_Point_Site< E >

FIXME: Doc!

8.108 mln::Delta_Point_Site< void > Struct Template Reference

Delta point site category flag type.

`#include <delta_point_site.hh>`

8.108.1 Detailed Description

template<> struct mln::Delta_Point_Site< void >

Delta point site category flag type.

8.109 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).*

8.109.1 Detailed Description

template<typename E> struct mln::doc::Accumulator< E >

Documentation class for mln::Accumulator.

See also:

mln::Accumulator

8.109.2 Member Typedef Documentation

8.109.2.1 template<typename E > typedef void mln::doc::Accumulator< E >::argument

The argument type of elements to accumulate.

8.109.3 Member Function Documentation

8.109.3.1 template<typename E > void mln::doc::Accumulator< E >::init ()

Initialize the accumulator.

8.109.3.2 template<typename E > void mln::doc::Accumulator< E >::take (const E & *other*)

Take into account another accumulator `other`.

8.109.3.3 template<typename E > void mln::doc::Accumulator< E >::take (const argument & *t*)

Take into account a argument `t` (an element).

8.110 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.

8.110.1 Detailed Description

template<typename E> struct mln::doc::Box< E >

Documentation class for mln::Box.

See also:

mln::Box

8.110.2 Member Typedef Documentation

8.110.2.1 typedef void mln::doc::Site_Set< E >::bkd_piter [inherited]

Backward Site_Iterator associated type.

8.110.2.2 typedef void mln::doc::Site_Set< E >::fwd_piter [inherited]

Forward Site_Iterator associated type.

8.110.2.3 typedef void mln::doc::Site_Set< E >::psite [inherited]

PSite associated type.

8.110.2.4 typedef void mln::doc::Site_Set< E >::site [inherited]

Site associated type.

8.110.3 Member Function Documentation

8.110.3.1 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.

8.110.3.2 bool mln::doc::Site_Set< E >::has (const psite & *p*) const [inherited]

Test if p belongs to this site set.

Parameters:

← ***p*** A psite.

Returns:

True if p is an element of the site set.

8.110.3.3 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.

8.110.3.4 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.

8.110.3.5 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.

8.111 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.*

8.111.1 Detailed Description

template<typename E> struct mln::doc::Dpoint< E >

Documentation class for mln::Dpoint.

See also:

mln::Dpoint

8.111.2 Member Typedef Documentation

8.111.2.1 template<typename E > typedef void mln::doc::Dpoint< E >::coord

Coordinate associated type.

8.111.2.2 template<typename E > typedef void mln::doc::Dpoint< E >::dpoint

Dpsite associated type.

Invariant:

This type has to derive from mln::Dpoint.

8.111.2.3 template<typename E > typedef void mln::doc::Dpoint< E >::point

Site associated type.

Invariant:

This type has to derive from mln::Point.

8.111.3 Member Enumeration Documentation

8.111.3.1 template<typename E > anonymous enum

Enumerator:

dim Dimension of the space.

Invariant:

dim > 0

8.111.4 Member Function Documentation

8.111.4.1 template<typename E > coord mln::doc::Dpoint< E >::operator[ ] (unsigned *i*) const

Read-only access to the `i-th` coordinate value.

Parameters:

← ***i*** The coordinate index.

Precondition:

`i < dim`

Returns:

The value of the `i-th` coordinate.

8.112 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 `p` *belongs to the image domain.*

- bool has (const psite &p) const

  Test if the image owns the point site `p`.

- 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`.

- lvalue operator[ ] (unsigned o)
  Read-write access to the image value at offset o.
- rvalue operator[ ] (unsigned o) const
  Read-only 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.

8.112.1 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".

8.112.2 Member Typedef Documentation

8.112.2.1 typedef void mln::doc::Image< E >::bkd_piter [inherited]

Backward point iterator associated type.

Invariant:

This type has to derive from mln::Site_Iterator.

8.112.2.2 typedef void mln::doc::Image< E >::coord [inherited]

Coordinate associated type.

8.112.2.3 typedef void mln::doc::Image< E >::dpoint [inherited]

Dpsite associated type.

Invariant:

This type has to derive from mln::Dpoint.

8.112.2.4 typedef void mln::doc::Image< E >::fwd_piter [inherited]

Forward point iterator associated type.

Invariant:

This type has to derive from mln::Site_Iterator.

8.112.2.5 typedef void mln::doc::Image< E >::lvalue [inherited]

Type returned by the read-write pixel value operator.

8.112.2.6 typedef void mln::doc::Image< E >::point [inherited]

Site associated type.

Invariant:

This type has to derive from mln::Point.

8.112.2.7 typedef void mln::doc::Image< E >::pset [inherited]

Point set associated type.

Invariant:

This type has to derive from mln::Site_Set.

8.112.2.8 typedef void mln::doc::Image< E >::psite [inherited]

Point_Site associated type.

Invariant:

This type has to derive from mln::Point_Site.

8.112.2.9 typedef void mln::doc::Image< E >::rvalue [inherited]

Type returned by the read pixel value operator.

8.112.2.10 typedef void mln::doc::Image< E >::skeleton [inherited]

Associate type that describes how this type of image is constructed.

8.112.2.11 typedef void mln::doc::Image< E >::value [inherited]

Value associated type.

Invariant:

This type is neither qualified by const, nor by reference.

8.112.2.12 typedef void mln::doc::Image< E >::vset [inherited]

Value set associated type.

Invariant:

This type has to derive from mln::Value_Set.

8.112.3 Member Function Documentation

8.112.3.1 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.

8.112.3.2 template<typename E > unsigned mln::doc::Fastest_Image< E >::border ()

Give the border thickness.

Precondition:

The image has to be initialized.

8.112.3.3 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.

8.112.3.4 template<typename E > int mln::doc::Fastest_Image< E >::delta_index (const dpoint & *dp*)

Give the offset corresponding to the delta-point `dp`.

Parameters:

← ***dp*** A delta-point.

Precondition:

The image has to be initialized.

8.112.3.5 const pset& mln::doc::Image< E >::domain () const [inherited]

Give the definition domain of the image.

Returns:

A reference to the domain point set.

8.112.3.6 bool mln::doc::Image< E >::has (const psite & *p*) const [inherited]

Test if p belongs to the image domain.

Parameters:

← ***p*** A point site.

Returns:

True if p belongs to the image domain.

Invariant:

has(p) is true => has(p) is also true.

8.112.3.7 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.

8.112.3.8 bool mln::doc::Image< E >::is_valid () const [inherited]

Test if the image have been initialized.

8.112.3.9 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.

8.112.3.10 unsigned mln::doc::Image< E >::nsites () const [inherited]

Give the number of points of the image domain.

8.112.3.11 lvalue mln::doc::Image< E >::operator() (const psite & *p*) [inherited]

Read-write access to the image value located at p.

Parameters:

← ***p*** A point site.

Precondition:

The image has to own the site p.

Returns:

The value at p (assignable).

8.112.3.12 rvalue mln::doc::Image< E >::operator() (const psite & *p*) const [inherited]

Read-only access to the image value located at p.

Parameters:

← ***p*** A point site.

Precondition:

The image has to own the site p.

Returns:

The value at p (not assignable).

8.112.3.13 template<typename E > lvalue mln::doc::Fastest_Image< E >::operator[ ] (unsigned *o*)

Read-write access to the image value at offset o.

Parameters:

← ***o*** An offset.

Precondition:

o < nelements()

Returns:

The value at o (assignable).

8.112.3.14 template<typename E > rvalue mln::doc::Fastest_Image< E >::operator[ ] (unsigned *o*) const

Read-only access to the image value at offset o.

Parameters:

← ***o*** An offset.

Precondition:

o < nelements()

Returns:

The value at o (not assignable).

8.112.3.15 template<typename E > point mln::doc::Fastest_Image< E >::point_at_index (unsigned *o*) const

Give the point at offset o.

Parameters:

← ***o*** An offset.

Precondition:

The image has to be initialized.
o < nelements()

8.112.3.16 const vset& mln::doc::Image< E >::values () const [inherited]

Give the set of values of the image.

Returns:

A reference to the value set.

8.113 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.

8.113.1 Detailed Description

template<typename E> struct mln::doc::Generalized_Pixel< E >

Documentation class for mln::Generalized_Pixel.

See also:

mln::Generalized_Pixel

8.113.2 Member Typedef Documentation

8.113.2.1 template<typename E> typedef void mln::doc::Generalized_Pixel< E >::image

Image associated type (with possible const qualification).

8.113.2.2 template<typename E> typedef void mln::doc::Generalized_Pixel< E >::rvalue

Read-only value associated type.

8.113.2.3 template<typename E> typedef void mln::doc::Generalized_Pixel< E >::value

Value associated type.

8.113.3 Member Function Documentation

8.113.3.1 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.

8.113.3.2 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 >.

8.114 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 p *belongs to the image domain.*

- bool has (const psite &p) const

  Test if the image owns the point site p.

- bool is_valid () const

  Test if the image have been initialized.

- 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.

- const vset & values () const

  Give the set of values of the image.

8.114.1 Detailed Description

template<typename E> struct mln::doc::Image< E >

Documentation class for mln::Image.

See also:

mln::Image

8.114.2 Member Typedef Documentation

8.114.2.1 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.

8.114.2.2 template<typename E> typedef void mln::doc::Image< E >::coord

Coordinate associated type.

8.114.2.3 template<typename E> typedef void mln::doc::Image< E >::dpoint

Dpsite associated type.

Invariant:

This type has to derive from mln::Dpoint.

8.114.2.4 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.

8.114.2.5 template<typename E> typedef void mln::doc::Image< E >::lvalue

Type returned by the read-write pixel value operator.

8.114.2.6 template<typename E> typedef void mln::doc::Image< E >::point

Site associated type.

Invariant:

This type has to derive from mln::Point.

8.114.2.7 template<typename E> typedef void mln::doc::Image< E >::pset

Point set associated type.

Invariant:

This type has to derive from mln::Site_Set.

8.114.2.8 template<typename E> typedef void mln::doc::Image< E >::psite

Point_Site associated type.

Invariant:

This type has to derive from mln::Point_Site.

8.114.2.9 template<typename E> typedef void mln::doc::Image< E >::rvalue

Type returned by the read pixel value operator.

8.114.2.10 template<typename E> typedef void mln::doc::Image< E >::skeleton

Associate type that describes how this type of image is constructed.

8.114.2.11 template<typename E> typedef void mln::doc::Image< E >::value

Value associated type.

Invariant:

This type is neither qualified by const, nor by reference.

8.114.2.12 template<typename E> typedef void mln::doc::Image< E >::vset

Value set associated type.

Invariant:

This type has to derive from mln::Value_Set.

8.114.3 Member Function Documentation

8.114.3.1 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.

8.114.3.2 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.

8.114.3.3 template<typename E> bool mln::doc::Image< E >::has (const psite & *p*) const

Test if p belongs to the image domain.

Parameters:

← ***p*** A point site.

Returns:

True if p belongs to the image domain.

Invariant:

has(p) is true => has(p) is also true.

8.114.3.4 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.

8.114.3.5 template<typename E> bool mln::doc::Image< E >::is_valid () const

Test if the image have been initialized.

8.114.3.6 template<typename E> unsigned mln::doc::Image< E >::nsites () const

Give the number of points of the image domain.

8.114.3.7 template<typename E> lvalue mln::doc::Image< E >::operator() (const psite & *p*)

Read-write access to the image value located at p.

Parameters:

← ***p*** A point site.

Precondition:

The image has to own the site p.

Returns:

The value at p (assignable).

8.114.3.8 template<typename E> rvalue mln::doc::Image< E >::operator() (const psite & *p*) const

Read-only access to the image value located at p.

Parameters:

← ***p*** A point site.

Precondition:

The image has to own the site p.

Returns:

The value at p (not assignable).

8.114.3.9 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.

8.115 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.

8.115.1 Detailed Description

template<typename E> struct mln::doc::Iterator< E >

Documentation class for mln::Iterator.

See also:

mln::Iterator

8.115.2 Member Function Documentation

8.115.2.1 template<typename E> void mln::doc::Iterator< E >::invalidate ()

Invalidate the iterator.

8.115.2.2 template<typename E> bool mln::doc::Iterator< E >::is_valid () const

Returns true if the iterator is valid, that is, designates an element.

8.115.2.3 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.

8.116 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.

8.116.1 Detailed Description

template<typename E> struct mln::doc::Neighborhood< E >

Documentation class for mln::Neighborhood.

See also:

mln::Neighborhood

8.116.2 Member Typedef Documentation

8.116.2.1 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.

8.116.2.2 template<typename E > typedef void mln::doc::Neighborhood< E >::dpoint

Dpsite associated type.

8.116.2.3 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.

8.116.2.4 template<typename E > typedef void mln::doc::Neighborhood< E >::niter

Site_Iterator type associated to this neighborhood to browse neighbors.

8.116.2.5 template<typename E > typedef void mln::doc::Neighborhood< E >::point

Site associated type.

8.117 mln::doc::Object< E > Struct Template Reference

Documentation class for mln::Object.

`#include <object.hh>`

Inheritance diagram for mln::doc::Object< E >:

8.117.1 Detailed Description

template<typename E> struct mln::doc::Object< E >

Documentation class for mln::Object.

See also:

mln::Object

8.118 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.

8.118.1 Detailed Description

template<typename E> struct mln::doc::Pixel_Iterator< E >

Documentation class for mln::Iterator.

See also:

mln::Pixel_Iterator

8.118.2 Member Typedef Documentation

8.118.2.1 typedef void mln::doc::Generalized_Pixel< E >::image [inherited]

Image associated type (with possible const qualification).

8.118.2.2 template<typename E > typedef void mln::doc::Pixel_Iterator< E >::lvalue

Type returned by the read-write dereference operator.

8.118.2.3 typedef void mln::doc::Generalized_Pixel< E >::rvalue [inherited]

Read-only value associated type.

8.118.2.4 typedef void mln::doc::Generalized_Pixel< E >::value [inherited]

Value associated type.

8.118.3 Member Function Documentation

8.118.3.1 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.

8.118.3.2 void mln::doc::Iterator< E >::invalidate () [inherited]

Invalidate the iterator.

8.118.3.3 bool mln::doc::Iterator< E >::is_valid () const [inherited]

Returns true if the iterator is valid, that is, designates an element.

8.118.3.4 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.

8.118.3.5 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 >.

8.119 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.

8.119.1 Detailed Description

template<typename E> struct mln::doc::Point_Site< E >

Documentation class for mln::Point_Site.

See also:

mln::Point_Site

8.119.2 Member Typedef Documentation

8.119.2.1 template<typename E > typedef void mln::doc::Point_Site< E >::coord

Coordinate associated type.

8.119.2.2 template<typename E > typedef void mln::doc::Point_Site< E >::dpoint

Dpsite associated type.

Invariant:

This type has to derive from mln::Dpoint.

8.119.2.3 template<typename E > typedef void mln::doc::Point_Site< E >::mesh

Mesh associated type.

Invariant:

This type has to derive from mln::Mesh.

8.119.2.4 template<typename E > typedef void mln::doc::Point_Site< E >::point

Site associated type.

Invariant:

This type has to derive from mln::Point.

8.119.3 Member Enumeration Documentation

8.119.3.1 template<typename E > anonymous enum

Enumerator:

dim Dimension of the space.

Invariant:

dim > 0

8.119.4 Member Function Documentation

8.119.4.1 template<typename E > coord mln::doc::Point_Site< E >::operator[ ] (unsigned *i*) const

Read-only access to the `i-th` coordinate value.

Parameters:

← ***i*** The coordinate index.

Precondition:

`i` < `dim`

Returns:

The value of the `i-th` coordinate.

8.119.4.2 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.

8.120 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.

8.120.1 Detailed Description

template<typename E> struct mln::doc::Site_Iterator< E >

Documentation class for mln::Site_Iterator.

See also:

mln::Site_Iterator

8.120.2 Member Typedef Documentation

8.120.2.1 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.

8.120.3 Member Function Documentation

8.120.3.1 void mln::doc::Iterator< E >::invalidate () [inherited]

Invalidate the iterator.

8.120.3.2 bool mln::doc::Iterator< E >::is_valid () const [inherited]

Returns true if the iterator is valid, that is, designates an element.

8.120.3.3 template<typename E > mln::doc::Site_Iterator< E >::operator psite () const

Conversion into a point-site.

Returns:

A point site.

8.120.3.4 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.

8.121 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.*

8.121.1 Detailed Description

template<typename E> struct mln::doc::Site_Set< E >

Documentation class for mln::Site_Set.

See also:

mln::Site_Set

8.121.2 Member Typedef Documentation

8.121.2.1 template<typename E> typedef void mln::doc::Site_Set< E >::bkd_piter

Backward Site_Iterator associated type.

8.121.2.2 template<typename E> typedef void mln::doc::Site_Set< E >::fwd_piter

Forward Site_Iterator associated type.

8.121.2.3 template<typename E> typedef void mln::doc::Site_Set< E >::psite

PSite associated type.

8.121.2.4 template<typename E> typedef void mln::doc::Site_Set< E >::site

Site associated type.

8.121.3 Member Function Documentation

8.121.3.1 template<typename E> bool mln::doc::Site_Set< E >::has (const psite & *p*) const

Test if p belongs to this site set.

Parameters:

← ***p*** A psite.

Returns:

True if p is an element of the site set.

8.122 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.

8.122.1 Detailed Description

template<typename E> struct mln::doc::Value_Iterator< E >

Documentation class for mln::Value_Iterator.

See also:

mln::Value_Iterator

8.122.2 Member Typedef Documentation

8.122.2.1 template<typename E > typedef void mln::doc::Value_Iterator< E >::value

Value associated type.

8.122.3 Member Function Documentation

8.122.3.1 void mln::doc::Iterator< E >::invalidate () [inherited]

Invalidate the iterator.

8.122.3.2 bool mln::doc::Iterator< E >::is_valid () const [inherited]

Returns true if the iterator is valid, that is, designates an element.

8.122.3.3 template<typename E > mln::doc::Value_Iterator< E >::operator value () const

Conversion into a value.

Returns:

A value.

8.122.3.4 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.

8.123 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.*

8.123.1 Detailed Description

template<typename E> struct mln::doc::Value_Set< E >

Documentation class for mln::Value_Set.

See also:

mln::Value_Set

8.123.2 Member Typedef Documentation

8.123.2.1 template<typename E > typedef void mln::doc::Value_Set< E >::bkd_viter

Backward Value_Iterator associated type.

8.123.2.2 template<typename E > typedef void mln::doc::Value_Set< E >::fwd_viter

Forward Value_Iterator associated type.

8.123.2.3 template<typename E > typedef void mln::doc::Value_Set< E >::value

Value associated type.

8.123.3 Member Function Documentation

8.123.3.1 template<typename E > bool mln::doc::Value_Set< E >::has (const value & *v*) const

Test if v belongs to this set of values.

Parameters:

← ***v*** A value.

Returns:

True if v is an element of the set of values.

8.123.3.2 template<typename E > unsigned mln::doc::Value_Set< E >::index_of (const value & *v*) const

Give the index of value v in this set.

8.123.3.3 template<typename E > unsigned mln::doc::Value_Set< E >::nvalues () const

Give the number of values in this set.

8.123.3.4 template<typename E > value mln::doc::Value_Set< E >::operator[ ] (unsigned *i*) const

Give the i-th value of this set.

8.124 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.

8.124.1 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

8.124.2 Member Typedef Documentation

8.124.2.1 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.

8.124.2.2 template<typename E > typedef void mln::doc::Weighted_Window< E >::dpoint

Dpsite associated type.

8.124.2.3 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.

8.124.2.4 template<typename E > typedef void mln::doc::Weighted_Window< E >::point

Site associated type.

8.124.2.5 template<typename E > typedef void mln::doc::Weighted_Window< E >::weight

Weight associated type.

8.124.2.6 template<typename E > typedef void mln::doc::Weighted_Window< E >::window

Window associated type.

8.124.3 Member Function Documentation

8.124.3.1 template<typename E > unsigned mln::doc::Weighted_Window< E >::delta () const

Give the maximum coordinate gap between the window center and a window point.

8.124.3.2 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.

8.124.3.3 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.

8.124.3.4 template<typename E > E& mln::doc::Weighted_Window< E >::sym ()

Apply a central symmetry to the target weighted window.

8.124.3.5 template<typename E > const window& mln::doc::Weighted_Window< E >::win () const

Give the corresponding window.

8.125 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.

8.125.1 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

8.125.2 Member Typedef Documentation

8.125.2.1 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.

8.125.2.2 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.

8.125.2.3 template<typename E > typedef void mln::doc::Window< E >::qiter

Site_Iterator type associated to this window to browse its points.

8.126 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

- template<typename F >
  dpoint (const Function_v2v< F > &f)

  Constructor; coordinates are set by function `f`.

- template<typename C2 >
  dpoint (const algebra::vec< dim, C2 > &v)

  Constructor from an algebra vector.

- dpoint ()

  Constructor without argument.

- template<typename Q >
  operator mln::algebra::vec< dpoint< G, C >::dim, Q > () const

  Conversion towards a algebra::vec.

- C & operator[ ] (unsigned i)

  Read-write access to the `i-th` *coordinate value.*

- C operator[ ] (unsigned i) const

  Read-only 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 (const literal::zero_t &)

  Constructors/assignments with literals.

- dpoint (C ind)

8.126.1 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.

8.126.2 Member Typedef Documentation

8.126.2.1 template<typename G, typename C> typedef C mln::dpoint< G, C >::coord

Coordinate associated type.

8.126.2.2 template<typename G, typename C> typedef G mln::dpoint< G, C >::grid

Grid associated type.

8.126.2.3 template<typename G, typename C> typedef point<G,C> mln::dpoint< G, C >::psite

Psite associated type.

8.126.2.4 template<typename G, typename C> typedef point<G,C> mln::dpoint< G, C >::site

Site associated type.

8.126.2.5 template<typename G, typename C> typedef algebra::vec<G::dim, C> mln::dpoint< G, C >::vec

Algebra vector (vec) associated type.

8.126.3 Member Enumeration Documentation

8.126.3.1 template<typename G, typename C> anonymous enum

Enumerator:

dim Dimension of the space.

Invariant:

dim > 0

8.126.4 Constructor & Destructor Documentation

8.126.4.1 template<typename G , typename C > mln::dpoint< G, C >::dpoint () [inline]

Constructor without argument.

8.126.4.2 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.

References mln::dpoint< G, C >::dim.

8.126.4.3 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.

8.126.4.4 template<typename G , typename C> mln::dpoint< G, C >::dpoint (const literal::zero_t &) [inline]

Constructors/assignments with literals.

8.126.4.5 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`.

References mln::dpoint< G, C >::dim.

8.126.5 Member Function Documentation

8.126.5.1 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.

References mln::dpoint< G, C >::to_vec().

8.126.5.2 template<typename G , typename C > C & mln::dpoint< G, C >::operator[ ] (unsigned *i*) [inline]

Read-write access to the `i-th` coordinate value.

Parameters:

← ***i*** The coordinate index.

Precondition:

`i < dim`

References mln::dpoint< G, C >::dim.

8.126.5.3 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:

← ***i*** The coordinate index.

Precondition:

`i < dim`

References mln::dpoint< G, C >::dim.

8.126.5.4 template<typename G , typename C> void mln::dpoint< G, C >::set_all (C *c*) [inline]

Set all coordinates to the value `c`.

References mln::dpoint< G, C >::dim.

Referenced by mln::win::line< M, i, C >::line().

8.126.5.5 template<typename G , typename C > dpoint< G, C >::vec mln::dpoint< G, C >::to_vec () const [inline]

Explicit conversion.

References mln::dpoint< G, C >::dim.

Referenced by mln::dpoint< G, C >::operator mln::algebra::vec< dpoint< G, C >::dim, Q >().

8.127 mln::Dpoint< E > Struct Template Reference

Base class for implementation of delta-point classes.

`#include <dpoint.hh>`

Inheritance diagram for mln::Dpoint< E >:

Public Member Functions

- const E & to_dpoint () const

  It is a Dpoint so it returns itself.

8.127.1 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.

8.127.2 Member Function Documentation

8.127.2.1 template<typename E > const E & mln::Dpoint< E >::to_dpoint () const [inline]

It is a Dpoint so it returns itself.

8.128 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>`

Inherits Pixel_Iterator< dpoints_bkd_pixter< I > >, and pixel_impl_< I, 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 invalidate ()
  Invalidate the iterator.
- bool is_valid () const
  Test the iterator validity.
- void start ()
  Manipulation.
- void update ()
  Force this iterator to update its location to take into account that its center point may have moved.

8.128.1 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.

8.128.2 Constructor & Destructor Documentation

8.128.2.1 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:

← ***image*** The image to iterate over.

← ***dps*** An object (neighborhood or window) that can provide a set of delta-points.

← ***p_ref*** Center (resp. reference) point of the neighborhood (resp. window).

8.128.2.2 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:

← ***pxl_ref*** Center (generalized) pixel to iterate around.

← ***dps*** An object (neighborhood or window) that can provide a set of delta-points.

8.128.3 Member Function Documentation

8.128.3.1 template<typename I > const I::value & mln::dpoints_bkd_pixter< I >::center_val () const [inline]

The value around which this iterator moves.

8.128.3.2 template<typename I > void mln::dpoints_bkd_pixter< I >::invalidate () [inline]

Invalidate the iterator.

8.128.3.3 template<typename I > bool mln::dpoints_bkd_pixter< I >::is_valid () const [inline]

Test the iterator validity.

Referenced by mln::dpoints_bkd_pixter< I >::update().

8.128.3.4 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.

8.128.3.5 template<typename I > void mln::dpoints_bkd_pixter< I >::start () [inline]

Manipulation.

Start an iteration.

References mln::dpoints_bkd_pixter< I >::update().

8.128.3.6 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.

References mln::dpoints_bkd_pixter< I >::is_valid().

Referenced by mln::dpoints_bkd_pixter< I >::start().

8.129 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>`

Inherits Pixel_Iterator< dpoints_fwd_pixter< I > >, and pixel_impl_< I, 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 invalidate ()
  Invalidate the iterator.
- bool is_valid () const
  Test the iterator validity.
- void start ()
  Manipulation.
- void update ()
  Force this iterator to update its location to take into account that its center point may have moved.

8.129.1 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.

8.129.2 Constructor & Destructor Documentation

8.129.2.1 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:

← ***image*** The image to iterate over.

← ***dps*** An object (neighborhood or window) that can provide a set of delta-points.

← ***p_ref*** Center (resp. reference) point of the neighborhood (resp. window).

8.129.2.2 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:

← ***pxl_ref*** Center (generalized) pixel to iterate around.

← ***dps*** An object (neighborhood or window) that can provide a set of delta-points.

8.129.3 Member Function Documentation

8.129.3.1 template<typename I > const I::value & mln::dpoints_fwd_pixter< I >::center_val () const [inline]

The value around which this iterator moves.

8.129.3.2 template<typename I > void mln::dpoints_fwd_pixter< I >::invalidate () [inline]

Invalidate the iterator.

8.129.3.3 template<typename I > bool mln::dpoints_fwd_pixter< I >::is_valid () const [inline]

Test the iterator validity.

Referenced by mln::dpoints_fwd_pixter< I >::update().

8.129.3.4 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.

8.129.3.5 template<typename I > void mln::dpoints_fwd_pixter< I >::start () [inline]

Manipulation.

Start an iteration.

References mln::dpoints_fwd_pixter< I >::update().

8.129.3.6 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.

References mln::dpoints_fwd_pixter< I >::is_valid().

Referenced by mln::dpoints_fwd_pixter< I >::start().

8.130 mln::dpsites_bkd_piter< V > Class Template Reference

A generic backward iterator on points of windows and of neighborhoods.

`#include <dpsites_piter.hh>`

Inherits site_relative_iterator_base< V, dpsites_bkd_piter< V > >.

Public Member Functions

- dpsites_bkd_piter ()
  Constructor without argument.
- template<typename P >
  dpsites_bkd_piter (const V &v, const P &c)
  Constructor.
- void next ()
  Go to the next element.

8.130.1 Detailed Description

template<typename V> class mln::dpsites_bkd_piter< V >

A generic backward iterator on points of windows and of neighborhoods.

The parameter `V` is the type of std::vector enclosing structure.

8.130.2 Constructor & Destructor Documentation

8.130.2.1 template<typename V > template<typename P > mln::dpsites_bkd_piter< V >::dpsites_bkd_piter (const V & *v*, const P & *c*) [inline]

Constructor.

Parameters:

← ***v*** Object that can provide an array of delta-points.
← ***c*** Center point to iterate around.

8.130.2.2 template<typename V > mln::dpsites_bkd_piter< V >::dpsites_bkd_piter () [inline]

Constructor without argument.

8.130.3 Member Function Documentation

8.130.3.1 void mln::Site_Iterator< dpsites_bkd_piter< V > >::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.

8.131 mln::dpsites_fwd_piter< V > Class Template Reference

A generic forward iterator on points of windows and of neighborhoods.

`#include <dpsites_piter.hh>`

Inherits site_relative_iterator_base< V, dpsites_fwd_piter< V > >.

Public Member Functions

- dpsites_fwd_piter ()
  Constructor without argument.
- template<typename P >
  dpsites_fwd_piter (const V &v, const P &c)
  Constructor.
- void next ()
  Go to the next element.

8.131.1 Detailed Description

template<typename V> class mln::dpsites_fwd_piter< V >

A generic forward iterator on points of windows and of neighborhoods.

The parameter `V` is the type of std::vector enclosing structure.

8.131.2 Constructor & Destructor Documentation

8.131.2.1 template<typename V > template<typename P > mln::dpsites_fwd_piter< V >::dpsites_fwd_piter (const V & *v*, const P & *c*) [inline]

Constructor.

Parameters:

← ***v*** Object that can provide an array of delta-points.
← ***c*** Center point to iterate around.

8.131.2.2 template<typename V > mln::dpsites_fwd_piter< V >::dpsites_fwd_piter () [inline]

Constructor without argument.

8.131.3 Member Function Documentation

8.131.3.1 void mln::Site_Iterator< dpsites_fwd_piter< V > >::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.

8.132 mln::Edge< E > Struct Template Reference

edge category flag type.

`#include <edge.hh>`

8.132.1 Detailed Description

template<typename E> struct mln::Edge< E >

edge category flag type.

8.133 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 G graph_t
  The type of the underlying graph.
- typedef graph_elt_neighborhood< G, p_edges< G, site_function_t > > nbh_t
  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 graph_elt_window< G, p_edges< G, site_function_t > > win_t
  Window type.

Public Member Functions

- edge_image ()
  Constructors.
- rvalue operator() (unsigned e_id) const
  Value accessors/operators overloads.

8.133.1 Detailed Description

template<typename P, typename V, typename G = util::graph> class mln::edge_image< P, V, G >

Image based on graph edges.

8.133.2 Member Typedef Documentation

8.133.2.1 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.

8.133.2.2 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 >::nbh_t

Neighborhood type.

8.133.2.3 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.

8.133.2.4 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.

8.133.2.5 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 >::win_t

Window type.

8.133.3 Constructor & Destructor Documentation

8.133.3.1 template<typename P , typename V , typename G > mln::edge_image< P, V, G >::edge_image () [inline]

Constructors.

8.133.4 Member Function Documentation

8.133.4.1 template<typename P , typename V , typename G > edge_image< P, V, G >::rvalue mln::edge_image< P, V, G >::operator() (unsigned *e_id*) const [inline]

Value accessors/operators overloads.

8.134 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 (I &ima, const box< typename I::site > &b)

  Constructor.

- extended ()

  Constructor without argument.

8.134.1 Detailed Description

template<typename I> struct mln::extended< I >

Makes an image become restricted by a point set.

8.134.2 Member Typedef Documentation

8.134.2.1 template<typename I> typedef tag::image_<I> mln::extended< I >::skeleton

Skeleton.

8.134.2.2 template<typename I> typedef I ::value mln::extended< I >::value

Value type.

8.134.3 Constructor & Destructor Documentation

8.134.3.1 template<typename I > mln::extended< I >::extended () [inline]

Constructor without argument.

8.134.3.2 template<typename I > mln::extended< I >::extended (I & *ima*, const box< typename I::site > & *b*) [inline]

Constructor.

8.134.4 Member Function Documentation

8.134.4.1 template<typename I > const box< typename I::site > & mln::extended< I >::domain () const [inline]

Give the definition domain.

8.135 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`;.

8.135.1 Detailed Description

template<typename I, typename F> class mln::extension_fun< I, F >

Extends the domain of an image with a function.

8.135.2 Member Typedef Documentation

8.135.2.1 template<typename I, typename F> typedef I ::value mln::extension_fun< I, F >::rvalue

Return type of read-only access.

8.135.2.2 template<typename I, typename F> typedef extension_fun< tag::image_<I>, tag::function_<F> > mln::extension_fun< I, F >::skeleton

Skeleton.

8.135.2.3 template<typename I, typename F> typedef I ::value mln::extension_fun< I, F >::value

Image value type.

8.135.3 Constructor & Destructor Documentation

8.135.3.1 template<typename I , typename F > mln::extension_fun< I, F >::extension_fun () [inline]

Constructor without argument.

8.135.3.2 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`.

8.135.4 Member Function Documentation

8.135.4.1 template<typename I , typename F > const F & mln::extension_fun< I, F >::extension () const [inline]

Give the extension function.

8.135.4.2 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`.

8.135.4.3 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`.

8.135.4.4 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;.

8.136 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_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`;.

8.136.1 Detailed Description

template<typename I, typename J> class mln::extension_ima< I, J >

Extends the domain of an image with an image.

8.136.2 Member Typedef Documentation

8.136.2.1 template<typename I, typename J> typedef I ::value mln::extension_ima< I, J >::rvalue

Return type of read-only access.

8.136.2.2 template<typename I, typename J> typedef extension_ima< tag::image_<I>, tag::ext_<J> > mln::extension_ima< I, J >::skeleton

Skeleton.

8.136.2.3 template<typename I, typename J> typedef I ::value mln::extension_ima< I, J >::value

Image value type.

8.136.3 Constructor & Destructor Documentation

8.136.3.1 template<typename I , typename J > mln::extension_ima< I, J >::extension_ima () [inline]

Constructor without argument.

8.136.3.2 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`.

8.136.4 Member Function Documentation

8.136.4.1 template<typename I , typename J > const J & mln::extension_ima< I, J >::extension () const [inline]

Read-only access to the extension domain (image).

8.136.4.2 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.

8.136.4.3 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`.

8.136.4.4 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;.

8.137 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`;.

8.137.1 Detailed Description

template<typename I> class mln::extension_val< I >

Extends the domain of an image with a value.

8.137.2 Member Typedef Documentation

8.137.2.1 template<typename I> typedef I ::value mln::extension_val< I >::rvalue

Return type of read-only access.

8.137.2.2 template<typename I> typedef extension_val< tag::image_<I> > mln::extension_val< I >::skeleton

Skeleton.

8.137.2.3 template<typename I> typedef I ::value mln::extension_val< I >::value

Image value type.

8.137.3 Constructor & Destructor Documentation

8.137.3.1 template<typename I > mln::extension_val< I >::extension_val () [inline]

Constructor without argument.

8.137.3.2 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`.

8.137.4 Member Function Documentation

8.137.4.1 template<typename I > void mln::extension_val< I >::change_extension (const typename I::value & *val*) [inline]

Change the value of the extension domain.

8.137.4.2 template<typename I > const I::value & mln::extension_val< I >::extension () const [inline]

Read-only access to the value of the extension domain.

8.137.4.3 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.

8.137.4.4 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`.

8.137.4.5 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;.

8.138 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.*
- T & operator() (const typename S::psite &p)
  Read-write access to the image value located at point `p`.
- const T & operator() (const typename S::psite &p) const
  Read-only access to the image value located at point `p`.

8.138.1 Detailed Description

template<typename T, typename S> struct mln::flat_image< T, S >

Image with a single value.

8.138.2 Member Typedef Documentation

8.138.2.1 template<typename T, typename S> typedef T& mln::flat_image< T, S >::lvalue

Return type of read-write access.

8.138.2.2 template<typename T, typename S> typedef const T& mln::flat_image< T, S >::rvalue

Return type of read-only access.

8.138.2.3 template<typename T, typename S> typedef flat_image< tag::value_<T>, tag::domain_<S> > mln::flat_image< T, S >::skeleton

Skeleton.

8.138.2.4 template<typename T, typename S> typedef T mln::flat_image< T, S >::value

Value associated type.

8.138.3 Constructor & Destructor Documentation

8.138.3.1 template<typename T , typename S > mln::flat_image< T, S >::flat_image () [inline]

Constructor without argument.

8.138.3.2 template<typename T , typename S > mln::flat_image< T, S >::flat_image (const T & *val*, const S & *pset*) [inline]

Constructor.

8.138.4 Member Function Documentation

8.138.4.1 template<typename T , typename S > const S & mln::flat_image< T, S >::domain () const [inline]

Give the definition domain.

8.138.4.2 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.

8.138.4.3 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.

8.138.4.4 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.

8.139 mln::fun::p2b::antilogy Struct Reference

A p2b function always returning `false`.

`#include <antilogy.hh>`

Inheritance diagram for mln::fun::p2b::antilogy:

8.139.1 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.

8.140 mln::fun::p2b::tautology Struct Reference

A p2b function always returning `true`.

`#include <tautology.hh>`

Inheritance diagram for mln::fun::p2b::tautology:

8.140.1 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.

8.141 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 >:

8.141.1 Detailed Description

template<typename V> struct mln::fun::v2b::lnot< V >

Functor computing logical-not on a value.

8.142 mln::fun::v2b::threshold< V > Struct Template Reference

Threshold function.

`#include <threshold.hh>`

Inheritance diagram for mln::fun::v2b::threshold< V >:

8.142.1 Detailed Description

template<typename V> struct mln::fun::v2b::threshold< V >

Threshold function.

f(v) = (v >= threshold).

8.143 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 >:

8.143.1 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.

8.144 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 >:

8.144.1 Detailed Description

template<typename T, unsigned i> struct mln::fun::v2v::component< T, i >

Functor that accesses the i-th component of a value.

8.145 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 >:

8.145.1 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.

8.146 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 >:

8.146.1 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.

8.147 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 >:

8.147.1 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 defaut, `T` is `V` and `R` is `T`.

8.148 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 >:

8.148.1 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.

8.149 mln::fun::v2w2v::cos< V > Struct Template Reference

Cosinus bijective functor.

`#include <cos.hh>`

Inheritance diagram for mln::fun::v2w2v::cos< V >:

8.149.1 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.

8.150 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 >:

8.150.1 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.

8.151 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 >:

8.151.1 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.

8.152 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 >:

8.152.1 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.

8.153 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 >:

8.153.1 Detailed Description

template<typename L, typename R = L> struct mln::fun::vv2b::eq< L, R >

Functor computing equal between two values.

8.154 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 >:

8.154.1 Detailed Description

template<typename L, typename R = L> struct mln::fun::vv2b::ge< L, R >

Functor computing "greater or equal than" between two values.

8.155 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 >:

8.155.1 Detailed Description

template<typename L, typename R = L> struct mln::fun::vv2b::gt< L, R >

Functor computing "greater than" between two values.

8.156 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 >:

8.156.1 Detailed Description

template<typename L, typename R = L> struct mln::fun::vv2b::implies< L, R >

Functor computing logical-implies between two values.

8.157 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 >:

8.157.1 Detailed Description

template<typename L, typename R = L> struct mln::fun::vv2b::le< L, R >

Functor computing "lower or equal than" between two values.

8.158 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 >:

8.158.1 Detailed Description

template<typename L, typename R = L> struct mln::fun::vv2b::lt< L, R >

Functor computing "lower than" between two values.

8.159 mln::fun::vv2v::diff_abs< V > Struct Template Reference

A functor computing the diff_absimum of two values.

`#include <diff_abs.hh>`

Inheritance diagram for mln::fun::vv2v::diff_abs< V >:

8.159.1 Detailed Description

template<typename V> struct mln::fun::vv2v::diff_abs< V >

A functor computing the diff_absimum of two values.

8.160 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 >:

8.160.1 Detailed Description

template<typename L, typename R = L> struct mln::fun::vv2v::land< L, R >

Functor computing logical-and between two values.

8.161 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 >:

8.161.1 Detailed Description

template<typename L, typename R = L> struct mln::fun::vv2v::land_not< L, R >

Functor computing logical and-not between two values.

8.162 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 >:

8.162.1 Detailed Description

template<typename L, typename R = L> struct mln::fun::vv2v::lor< L, R >

Functor computing logical-or between two values.

8.163 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 >:

8.163.1 Detailed Description

template<typename L, typename R = L> struct mln::fun::vv2v::lxor< L, R >

Functor computing logical-xor between two values.

8.164 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 >:

8.164.1 Detailed Description

template<typename V> struct mln::fun::vv2v::max< V >

A functor computing the maximum of two values.

8.165 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 >:

8.165.1 Detailed Description

template<typename L, typename R = L> struct mln::fun::vv2v::min< L, R >

A functor computing the minimum of two values.

8.166 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 >:

8.166.1 Detailed Description

template<typename V> struct mln::fun::vv2v::vec< V >

A functor computing the vecimum of two values.

8.167 mln::fun::x2p::closest_point< P > Struct Template Reference

FIXME: doxygen + concept checking.

`#include <closest_point.hh>`

8.167.1 Detailed Description

template<typename P> struct mln::fun::x2p::closest_point< P >

FIXME: doxygen + concept checking.

8.168 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.

8.168.1 Detailed Description

template<typename I> struct mln::fun::x2v::bilinear< I >

Represent a bilinear interolation of values from an underlying image.

8.168.2 Member Function Documentation

8.168.2.1 template<typename I > template<typename T > I::value mln::fun::x2v::bilinear< I >::operator() (const algebra::vec< 3, T > & *v*) const [inline]

Bilinear filtering on 3d images. Work on slices.

8.168.2.2 template<typename I > template<typename T > I::value mln::fun::x2v::bilinear< I >::operator() (const algebra::vec< 2, T > & *v*) const [inline]

Bilinear filtering on 2d images.

8.169 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 >:

8.169.1 Detailed Description

template<typename I> struct mln::fun::x2v::trilinear< I >

Represent a trilinear interolation of values from an underlying image.

8.170 mln::fun::x2x::composed< T2, T1 > Struct Template Reference

Represent a composition of two transformations.

`#include <composed.hh>`

Public Member Functions

- composed (const T2 &f, const T1 &g)

  Constructor with the two transformation to be composed.

- composed ()

  Constructor without argument.

8.170.1 Detailed Description

template<typename T2, typename T1> struct mln::fun::x2x::composed< T2, T1 >

Represent a composition of two transformations.

8.170.2 Constructor & Destructor Documentation

8.170.2.1 template<typename T2, typename T1> mln::fun::x2x::composed< T2, T1 >::composed () [inline]

Constructor without argument.

8.170.2.2 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.

8.171 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.

8.171.1 Detailed Description

template<typename I> struct mln::fun::x2x::linear< I >

Represent a linear interolation of values from an underlying image.

8.171.2 Constructor & Destructor Documentation

8.171.2.1 template<typename I > mln::fun::x2x::linear< I >::linear (const I & *ima*) [inline]

Constructor with the underlying image.

8.171.3 Member Function Documentation

8.171.3.1 template<typename I > template<typename C > I::value mln::fun::x2x::linear< I >::operator() (const algebra::vec< 1, C > & *v*) const [inline]

Return the interpolated value in the underlying image at the given 'point' v.

8.171.4 Member Data Documentation

8.171.4.1 template<typename I > const I& mln::fun::x2x::linear< I >::ima

Underlying image.

8.172 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 rotation< n, C > invert
  Type of the inverse function.

Public Member Functions

- invert inv () const
  Return the invere function.
- algebra::vec< n, C > operator() (const algebra::vec< n, C > &v) const
  Perform the rotation of the given vector.
- rotation (const algebra::h_mat< n, C > &m)

Constructor with h_mat.

- rotation (const algebra::quat &q)

  Constructor with quaternion.

- rotation (C alpha, const algebra::vec< n, C > &axis)

  Constructor with radian alpha and a facultative direction (rotation axis).

- rotation ()

  Constructor without argument.

- void set_alpha (C alpha)

  Set a new grade alpha.

- void set_axis (const algebra::vec< n, C > &axis)

  Set a new rotation axis.

8.172.1 Detailed Description

template<unsigned n, typename C> struct mln::fun::x2x::rotation< n, C >

Represent a rotation function.

8.172.2 Member Typedef Documentation

8.172.2.1 template<unsigned n, typename C > typedef rotation<n,C> mln::fun::x2x::rotation< n, C >::invert

Type of the inverse function.

8.172.3 Constructor & Destructor Documentation

8.172.3.1 template<unsigned n, typename C > mln::fun::x2x::rotation< n, C >::rotation () [inline]

Constructor without argument.

8.172.3.2 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).

8.172.3.3 template<unsigned n, typename C > mln::fun::x2x::rotation< n, C >::rotation (const algebra::quat & *q*) [inline]

Constructor with quaternion.

References mln::make::h_mat().

8.172.3.4 template<unsigned n, typename C > mln::fun::x2x::rotation< n, C >::rotation (const algebra::h_mat< n, C > & *m*) [inline]

Constructor with h_mat.

8.172.4 Member Function Documentation

8.172.4.1 template<unsigned n, typename C > rotation< n, C > mln::fun::x2x::rotation< n, C >::inv () const [inline]

Return the invere function.

8.172.4.2 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.

8.172.4.3 template<unsigned n, typename C > void mln::fun::x2x::rotation< n, C >::set_alpha (C *alpha*) [inline]

Set a new grade alpha.

8.172.4.4 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.

8.173 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 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.

8.173.1 Detailed Description

template<unsigned n, typename C> struct mln::fun::x2x::translation< n, C >

Translation function-object.

8.173.2 Member Typedef Documentation

8.173.2.1 template<unsigned n, typename C > typedef translation<n,C> mln::fun::x2x::translation< n, C >::invert

Type of the inverse function.

8.173.3 Constructor & Destructor Documentation

8.173.3.1 template<unsigned n, typename C > mln::fun::x2x::translation< n, C >::translation () [inline]

Constructor without argument.

8.173.3.2 template<unsigned n, typename C > mln::fun::x2x::translation< n, C >::translation (const algebra::vec< n, C > & *t*) [inline]

Constructor with the translation vector.

8.173.4 Member Function Documentation

8.173.4.1 template<unsigned n, typename C > translation< n, C > mln::fun::x2x::translation< n, C >::inv () const [inline]

Return the inverse function.

8.173.4.2 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.

8.173.4.3 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.

8.173.4.4 template<unsigned n, typename C > const algebra::vec< n, C > & mln::fun::x2x::translation< n, C >::t () const [inline]

Return the translation vector.

8.174 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 (const Image< I > &ima)

  Constructor.

- fun_image (const Function_v2v< F > &f, const Image< I > &ima)

  Constructor.

- fun_image ()

  Constructor.

- F::result operator() (const typename I::psite &p)

  Mutable access is for reading only.

- F::result operator() (const typename I::psite &p) const

  Read-only access of pixel value at point site `p`.

8.174.1 Detailed Description

template<typename F, typename I> struct mln::fun_image< F, I >

Image read through a function.

8.174.2 Member Typedef Documentation

8.174.2.1 template<typename F, typename I> typedef F ::result mln::fun_image< F, I >::lvalue

Return type of read-write access.

8.174.2.2 template<typename F, typename I> typedef F ::result mln::fun_image< F, I >::rvalue

Return type of read-only access.

8.174.2.3 template<typename F, typename I> typedef fun_image< tag::value_<typename F ::result>, tag::image_<I> > mln::fun_image< F, I >::skeleton

Skeleton.

8.174.2.4 template<typename F, typename I> typedef F ::result mln::fun_image< F, I >::value

Value associated type.

8.174.3 Constructor & Destructor Documentation

8.174.3.1 template<typename F , typename I > mln::fun_image< F, I >::fun_image () [inline]

Constructor.

8.174.3.2 template<typename F , typename I > mln::fun_image< F, I >::fun_image (const Function_v2v< F > & *f*, const Image< I > & *ima*) [inline]

Constructor.

8.174.3.3 template<typename F , typename I > mln::fun_image< F, I >::fun_image (const Image< I > & *ima*) [inline]

Constructor.

8.174.4 Member Function Documentation

8.174.4.1 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.

8.174.4.2 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`.

8.175 mln::Function< E > Struct Template Reference

Base class for implementation of function-objects.

`#include <function.hh>`

Inherits Object< E >.

Protected Member Functions

- Function ()

  An operator() has to be provided.

8.175.1 Detailed Description

template<typename E> struct mln::Function< E >

Base class for implementation of function-objects.

The parameter *E* is the exact type.

8.175.2 Constructor & Destructor Documentation

8.175.2.1 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.

8.176 mln::Function< void > Struct Template Reference

Function category flag type.

`#include <function.hh>`

8.176.1 Detailed Description

template<> struct mln::Function< void >

Function category flag type.

8.177 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 >:

8.177.1 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.

8.178 mln::Function_v2v< E > Struct Template Reference

Base class for implementation of function-objects from value to value.

`#include <function.hh>`

Inherits Function< E >.

Inherited by mln::fun::v2v::inc< T >, mln::fun::x2v::bilinear< I >, and mln::fun::x2x::nneighbor< I >.

8.178.1 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.

8.179 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 >:

8.179.1 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.

8.180 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 >:

8.180.1 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.

8.181 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.

8.181.1 Detailed Description

template<typename I> class mln::fwd_pixter1d< I >

Forward pixel iterator on a 1-D image with border.

8.181.2 Member Typedef Documentation

8.181.2.1 template<typename I > typedef I mln::fwd_pixter1d< I >::image

Image type.

8.181.3 Constructor & Destructor Documentation

8.181.3.1 template<typename I > mln::fwd_pixter1d< I >::fwd_pixter1d (I & *image*) [inline]

Constructor.

Parameters:

← ***image*** The image this pixel iterator is bound to.

8.181.4 Member Function Documentation

8.181.4.1 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.

8.182 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.

8.182.1 Detailed Description

template<typename I> class mln::fwd_pixter2d< I >

Forward pixel iterator on a 2-D image with border.

8.182.2 Member Typedef Documentation

8.182.2.1 template<typename I > typedef I mln::fwd_pixter2d< I >::image

Image type.

8.182.3 Constructor & Destructor Documentation

8.182.3.1 template<typename I > mln::fwd_pixter2d< I >::fwd_pixter2d (I & *image*) [inline]

Constructor.

Parameters:

← ***image*** The image this pixel iterator is bound to.

8.182.4 Member Function Documentation

8.182.4.1 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.

8.183 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.

8.183.1 Detailed Description

template<typename I> class mln::fwd_pixter3d< I >

Forward pixel iterator on a 3-D image with border.

8.183.2 Member Typedef Documentation

8.183.2.1 template<typename I > typedef I mln::fwd_pixter3d< I >::image

Image type.

8.183.3 Constructor & Destructor Documentation

8.183.3.1 template<typename I > mln::fwd_pixter3d< I >::fwd_pixter3d (I & *image*) [inline]

Constructor.

Parameters:

← ***image*** The image this pixel iterator is bound to.

8.183.4 Member Function Documentation

8.183.4.1 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.

8.184 mln::Gdpoint< E > Struct Template Reference

FIXME: Doc!

`#include <gdpoint.hh>`

Inheritance diagram for mln::Gdpoint< E >:

8.184.1 Detailed Description

template<typename E> struct mln::Gdpoint< E >

FIXME: Doc!

8.185 mln::Gdpoint< void > Struct Template Reference

Delta point site category flag type.

`#include <gdpoint.hh>`

8.185.1 Detailed Description

template<> struct mln::Gdpoint< void >

Delta point site category flag type.

8.186 mln::Generalized_Pixel< E > Struct Template Reference

Base class for implementation classes that are pixels or that have the behavior of pixels.

`#include <generalized_pixel.hh>`

Inheritance diagram for mln::Generalized_Pixel< E >:

8.186.1 Detailed Description

template<typename E> struct mln::Generalized_Pixel< E >

Base class for implementation classes that are pixels or that have the behavior of pixels.

Warning:

This class does *not* derive from mln::Object; it is for use as a parallel hierarchy.

See also:

mln::doc::Generalized_Pixel for a complete documentation of this class contents.

8.187 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.

8.187.1 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.

8.187.2 Constructor & Destructor Documentation

8.187.2.1 template<unsigned D, typename P > mln::geom::complex_geometry< D, P >::complex_geometry () [inline]

Build a complex geometry object.

8.187.3 Member Function Documentation

8.187.3.1 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 corresonding 0-face in the complex).

8.187.3.2 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*.

References mln::topo::face< D >::face_id(), and mln::topo::face< D >::n().

8.188 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`.

8.188.1 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.

8.188.2 Friends And Related Function Documentation

8.188.2.1 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:

← ***p*** A grid point.

← ***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

8.188.2.2 template<typename P , typename D > P & operator+= (Gpoint< P > & *p*, const Gdpoint< D > & *dp*) [related]

Shift a point by a delta-point dp.

Parameters:

↔ ***p*** The targeted point.

← ***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.

8.188.2.3 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:

← ***lhs*** A first grid point.

← ***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

8.188.2.4 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:

↔ ***p*** The targeted point.

← ***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.

8.188.2.5 template<typename P , typename D > P operator/ (const Gpoint< P > & *p*, const value::scalar_< D > & *dp*) [related]

Divise a point by a scalar s.

Parameters:

↔ ***p*** The targeted point.

← ***dp*** A scalar.

Returns:

A reference to the point p once divised by s.

8.188.2.6 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:

↔ ***ostr*** An output stream.

← ***p*** A grid point.

Returns:

The modified output stream ostr.

References mln::debug::format().

8.188.2.7 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:

← ***lhs*** A first grid point.

← ***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.

8.189 mln::Graph< E > Struct Template Reference

Base class for implementation of graph classes.

`#include <graph.hh>`

Inheritance diagram for mln::Graph< E >:

Public Member Functions

- void invalidate ()

  Invalidate the graph.

- bool is_valid () const

  Return true if this graph is valid.

8.189.1 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.

8.189.2 Member Function Documentation

8.189.2.1 template<typename E > void mln::Graph< E >::invalidate () [inline]

Invalidate the graph.

FIXME: does nothing!

8.189.2.2 template<typename E > bool mln::Graph< E >::is_valid () const [inline]

Return true if this graph is valid.

FIXME: currently it always returns true.

8.190 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.

8.190.1 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.

8.190.2 Member Typedef Documentation

8.190.2.1 typedef util::array<unsigned> mln::graph::attribute::card_t::result

Type of the computed value.

8.191 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.

8.191.1 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.

8.191.2 Member Typedef Documentation

8.191.2.1 typedef util::array<unsigned> mln::graph::attribute::representative_t::result

Type of the computed value.

8.192 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.

Public Member Functions

- void change_window (const graph_elt_window< G, S > &new_win)

  Change the corresponding window.

- const graph_elt_window< G, S > & win () const

  Get the corresponding window.

8.192.1 Detailed Description

template<typename G, typename S> struct mln::graph_elt_neighborhood< G, S >

Elementary neighborhood on graph class.

8.192.2 Member Typedef Documentation

8.192.2.1 typedef neighb_bkd_niter<graph_elt_window< G, S > > mln::neighb< graph_elt_window< G, S > >::bkd_niter [inherited]

Backward site iterator associated type.

8.192.2.2 typedef neighb_fwd_niter<graph_elt_window< G, S > > mln::neighb< graph_elt_window< G, S > >::fwd_niter [inherited]

Forward site iterator associated type.

8.192.2.3 typedef fwd_niter mln::neighb< graph_elt_window< G, S > >::niter [inherited]

Site iterator associated type.

8.192.3 Member Function Documentation

8.192.3.1 void mln::neighb< graph_elt_window< G, S > >::change_window (const graph_elt_window< G, S > & *new_win*) [inherited]

Change the corresponding window.

8.192.3.2 const graph_elt_window< G, S > & mln::neighb< graph_elt_window< G, S > >::win () const [inherited]

Get the corresponding window.

8.193 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

- void change_window (const graph_elt_window_if< G, S, I > &new_win)
  Change the corresponding window.
- graph_elt_neighborhood_if (const Image< I > &mask)
- graph_elt_neighborhood_if ()
  Constructors @{ Construct an invalid neighborhood.
- const I & mask () const
  @}
- const graph_elt_window_if< G, S, I > & win () const

Get the corresponding window.

8.193.1 Detailed Description

template<typename G, typename S, typename I> struct mln::graph_elt_neighborhood_if< G, S, I >

Elementary neighborhood_if on graph class.

8.193.2 Member Typedef Documentation

8.193.2.1 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.

8.193.2.2 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.

8.193.2.3 typedef fwd_niter mln::neighb< graph_elt_window_if< G, S, I > >::niter [inherited]

Site iterator associated type.

8.193.3 Constructor & Destructor Documentation

8.193.3.1 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.

8.193.3.2 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:

← ***mask*** A graph image of Boolean.

8.193.4 Member Function Documentation

8.193.4.1 void mln::neighb< graph_elt_window_if< G, S, I > >::change_window (const graph_elt_window_if< G, S, I > & *new_win*) [inherited]

Change the corresponding window.

8.193.4.2 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.

8.193.4.3 const graph_elt_window_if< G, S, I > & mln::neighb< graph_elt_window_if< G, S, I > >::win () const [inherited]

Get the corresponding window.

8.194 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 graph_window_piter< target, self_, nbh_bkd_iter_ > bkd_qiter
  Site_Iterator type to browse the psites of the window w.r.t.
- 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 target::psite psite
  The type of psite corresponding to the window.
- typedef fwd_qiter qiter
  The default qiter type.
- typedef S target
  Associated types.
- typedef S::fun_t::result site
  Associated types.

Public Member Functions

- bool is_valid () const

Return true by default.

- unsigned delta () const
  Return the maximum coordinate gap between the window center and a window point.
- bool is_centered () const
  Is the window centered?
- bool is_empty () const
  Interface of the concept Window.
- bool is_symmetric () const
  Is the window symmetric?
- self_ & sym ()
  Apply a central symmetry to the target window.

8.194.1 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 the image site set.

8.194.2 Member Typedef Documentation

8.194.2.1 template<typename G , typename S > typedef graph_window_piter<target,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.

8.194.2.2 template<typename G , typename S > typedef graph_window_piter<target,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.

8.194.2.3 template<typename G , typename S > typedef target ::psite mln::graph_elt_window< G, S >::psite

The type of psite corresponding to the window.

8.194.2.4 template<typename G , typename S > typedef fwd_qiter mln::graph_elt_window< G, S >::qiter

The default qiter type.

8.194.2.5 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.

8.194.2.6 template<typename G , typename S > typedef S mln::graph_elt_window< G, S >::target

Associated types.

8.194.3 Member Function Documentation

8.194.3.1 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.

8.194.3.2 bool mln::graph_window_base< S::fun_t::result , graph_elt_window< G, S > >::is_centered () const [inherited]

Is the window centered?

8.194.3.3 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?

8.194.3.4 bool mln::graph_window_base< S::fun_t::result , graph_elt_window< G, S > >::is_symmetric () const [inherited]

Is the window symmetric?

8.194.3.5 bool mln::graph_window_base< S::fun_t::result , graph_elt_window< G, S > >::is_valid () const [inherited]

Return true by default.

8.194.3.6 self_& mln::graph_window_base< S::fun_t::result , graph_elt_window< G, S > >::sym () [inherited]

Apply a central symmetry to the target window.

8.195 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 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 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 target::psite psite
  The type of psite corresponding to the window.

- typedef fwd_qiter qiter
  The default qiter type.

- typedef S target
  @}

- 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 (const Image< I > &mask)
- graph_elt_window_if ()

  Constructor.

- bool is_valid () const

  Return true by default.

- const I & mask () const

  Return the graph image used as mask.

- unsigned delta () const

  Return the maximum coordinate gap between the window center and a window point.

- bool is_centered () const

  Is the window centered?

- bool is_empty () const

  Interface of the concept Window.

- bool is_symmetric () const

  Is the window symmetric?

- self_ & sym ()

  Apply a central symmetry to the target window.

8.195.1 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.

8.195.2 Member Typedef Documentation

8.195.2.1 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.

8.195.2.2 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.

8.195.2.3 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.

8.195.2.4 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.

8.195.2.5 template<typename G , typename S , typename I > typedef fwd_qiter mln::graph_elt_window_if< G, S, I >::qiter

The default qiter type.

8.195.2.6 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.

8.195.2.7 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.

8.195.3 Constructor & Destructor Documentation

8.195.3.1 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.

8.195.3.2 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:

← ***mask*** A graph image of bool.

See also:

vertex_image, edge_image.

8.195.4 Member Function Documentation

8.195.4.1 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.

References mln::graph_elt_window_if< G, S, I >::is_valid().

8.195.4.2 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.

8.195.4.3 bool mln::graph_window_base< S::fun_t::result , graph_elt_window_if< G, S, I > >::is_centered () const [inherited]

Is the window centered?

8.195.4.4 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?

8.195.4.5 bool mln::graph_window_base< S::fun_t::result , graph_elt_window_if< G, S, I > >::is_symmetric () const [inherited]

Is the window symmetric?

8.195.4.6 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 > >.

Referenced by mln::graph_elt_window_if< G, S, I >::change_mask().

8.195.4.7 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.

8.195.4.8 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.

8.196 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.
- unsigned delta () const
  Return the maximum coordinate gap between the window center and a window point.
- bool is_centered () const
  Is the window centered?
- bool is_empty () const
  Interface of the concept Window.
- bool is_symmetric () const
  Is the window symmetric?
- self_ & sym ()
  Apply a central symmetry to the target window.

8.196.1 Detailed Description

template<typename P, typename E> class mln::graph_window_base< P, E >

Template Parameters:

P Site type.

8.196.2 Member Typedef Documentation

8.196.2.1 template<typename P, typename E> typedef P mln::graph_window_base< P, E >::site

Associated types.

The type of site corresponding to the window.

8.196.3 Member Function Documentation

8.196.3.1 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.

8.196.3.2 template<typename P , typename E > bool mln::graph_window_base< P, E >::is_centered () const [inline]

Is the window centered?

8.196.3.3 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?

8.196.3.4 template<typename P , typename E > bool mln::graph_window_base< P, E >::is_symmetric () const [inline]

Is the window symmetric?

8.196.3.5 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 >.

8.196.3.6 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.

8.197 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.

- const S::graph_element & element () const

  Return the graph element pointed by this iterator.

- unsigned id () const

  Return the graph element id.

- graph_window_if_piter ()

  Construction.

8.197.1 Detailed Description

template<typename S, typename W, typename I> class mln::graph_window_if_piter< S, W, I >

Forward iterator on line graph window.

8.197.2 Member Typedef Documentation

8.197.2.1 template<typename S , typename W , typename I > typedef S::fun_t ::result mln::graph_window_if_piter< S, W, I >::P

Associated types.

8.197.3 Constructor & Destructor Documentation

8.197.3.1 template<typename S , typename W , typename I > mln::graph_window_if_piter< S, W, I >::graph_window_if_piter () [inline]

Construction.

8.197.4 Member Function Documentation

8.197.4.1 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.

8.197.4.2 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.

8.197.4.3 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.

8.198 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 > >.

Public Types

- typedef S::fun_t::result P
  Associated types.

Public Member Functions

- void next ()
  Go to the next element.

- const S::graph_element & element () const
  Return the graph element pointed by this iterator.

- unsigned id () const
  Return the graph element id.

- graph_window_piter ()
  Construction.

8.198.1 Detailed Description

template<typename S, typename W, typename I> class mln::graph_window_piter< S, W, I >

Forward iterator on line graph window.

8.198.2 Member Typedef Documentation

8.198.2.1 template<typename S , typename W , typename I > typedef S::fun_t ::result mln::graph_window_piter< S, W, I >::P

Associated types.

8.198.3 Constructor & Destructor Documentation

8.198.3.1 template<typename S , typename W , typename I > mln::graph_window_piter< S, W, I >::graph_window_piter () [inline]

Construction.

8.198.4 Member Function Documentation

8.198.4.1 template<typename S , typename W , typename I > const S::graph_element & mln::graph_window_piter< S, W, I >::element () const [inline]

Return the graph element pointed by this iterator.

8.198.4.2 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.

8.198.4.3 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.

8.199 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.

8.199.1 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 ———————-

8.199.2 Member Typedef Documentation

8.199.2.1 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.

8.199.2.2 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.

8.199.2.3 template<typename I> typedef I ::lvalue mln::hexa< I >::lvalue

Lvalue associated type.

8.199.2.4 template<typename I> typedef point2d_h mln::hexa< I >::psite

Point site type.

Reimplemented in mln::image2d_h< V >.

8.199.2.5 template<typename I> typedef I ::rvalue mln::hexa< I >::rvalue

Return type of read-only access.

8.199.2.6 template<typename I> typedef hexa< tag::image_<I> > mln::hexa< I >::skeleton

Skeleton.

8.199.2.7 template<typename I> typedef I ::value mln::hexa< I >::value

Value associated type.

8.199.3 Constructor & Destructor Documentation

8.199.3.1 template<typename I > mln::hexa< I >::hexa () [inline]

Constructor without argument.

8.199.3.2 template<typename I> mln::hexa< I >::hexa (I & *ima*) [inline]

Constructor with an base image.

8.199.4 Member Function Documentation

8.199.4.1 template<typename I > const box2d_h & mln::hexa< I >::domain () const [inline]

Give the definition domain.

8.199.4.2 template<typename I > bool mln::hexa< I >::has (const psite & *p*) const [inline]

Test if p belongs to the image domain.

Referenced by mln::hexa< I >::operator()().

8.199.4.3 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.

References mln::hexa< I >::has().

8.199.4.4 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.

References mln::hexa< I >::has().

8.200 mln::histo::array< T > Struct Template Reference

Generic histogram class over a value set with type `T`.

`#include <array.hh>`

8.200.1 Detailed Description

template<typename T> struct mln::histo::array< T >

Generic histogram class over a value set with type `T`.

8.201 mln::Image< E > Struct Template Reference

Base class for implementation of image classes.

`#include <image.hh>`

Inheritance diagram for mln::Image< E >:

8.201.1 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.

8.202 mln::image1d< T > Struct Template Reference

Basic 1D image class.

`#include <image1d.hh>`

Inherits image_primary< T, box1d, image1d< T > >.

Package 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.

Package Functions

- const box1d & bbox () const

  Give the bounding box domain.

- unsigned border () const

  Give the border thickness.

- T * buffer ()

  Give a hook to the value buffer.

- const T * buffer () const

  Give a hook to the value buffer.

- int delta_index (const dpoint1d &dp) const

  Give the offset corresponding to the delta-point `dp`.

- const box1d & domain () const

  Give the definition domain.

- T & element (unsigned i)

  Read-write access to the `i-th` *image value (including the border).*

- const T & element (unsigned i) const

  Read-only access to the `i-th` *image value (including the border).*

- bool has (const point1d &p) const

Test if p *is valid.*

- 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.
- image1d ()
  Constructor without argument.
- 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.

8.202.1 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.

8.202.2 Member Typedef Documentation

8.202.2.1 template<typename T> typedef T& mln::image1d< T >::lvalue [package]

Return type of read-write access.

8.202.2.2 template<typename T> typedef const T& mln::image1d< T >::rvalue [package]

Return type of read-only access.

8.202.2.3 template<typename T> typedef image1d< tag::value_<T> > mln::image1d< T >::skeleton [package]

Skeleton.

8.202.2.4 template<typename T> typedef T mln::image1d< T >::value [package]

Value associated type.

8.202.3 Constructor & Destructor Documentation

8.202.3.1 template<typename T > mln::image1d< T >::image1d () [inline, package]

Constructor without argument.

8.202.3.2 template<typename T > mln::image1d< T >::image1d (unsigned *ninds*, unsigned *bdr* = border::thickness) [inline, package]

Constructor with the number of indices and the border thickness.

References mln::make::box1d().

8.202.3.3 template<typename T > mln::image1d< T >::image1d (const box1d & *b*, unsigned *bdr* = border::thickness) [inline, package]

Constructor with a box and the border thickness.

8.202.4 Member Function Documentation

8.202.4.1 template<typename T > const box1d & mln::image1d< T >::bbox () const [inline, package]

Give the bounding box domain.

8.202.4.2 template<typename T > unsigned mln::image1d< T >::border () const [inline, package]

Give the border thickness.

8.202.4.3 template<typename T > T * mln::image1d< T >::buffer () [inline, package]

Give a hook to the value buffer.

8.202.4.4 template<typename T > const T * mln::image1d< T >::buffer () const [inline, package]

Give a hook to the value buffer.

8.202.4.5 template<typename T > int mln::image1d< T >::delta_index (const dpoint1d & *dp*) const [inline, package]

Give the offset corresponding to the delta-point dp.

8.202.4.6 template<typename T > const box1d & mln::image1d< T >::domain () const [inline, package]

Give the definition domain.

8.202.4.7 template<typename T > T & mln::image1d< T >::element (unsigned *i*) [inline, package]

Read-write access to the i-th image value (including the border).

References mln::image1d< T >::nelements().

8.202.4.8 template<typename T > const T & mln::image1d< T >::element (unsigned *i*) const [inline, package]

Read-only access to the i-th image value (including the border).

References mln::image1d< T >::nelements().

8.202.4.9 template<typename T > bool mln::image1d< T >::has (const point1d & *p*) const [inline, package]

Test if p is valid.

Referenced by mln::image1d< T >::operator()().

8.202.4.10 template<typename T > unsigned mln::image1d< T >::nelements () const [inline, package]

Give the number of cells (points including border ones).

Referenced by mln::image1d< T >::element(), and mln::image1d< T >::point_at_index().

8.202.4.11 template<typename T > unsigned mln::image1d< T >::ninds () const [inline, package]

Give the number of indexes.

Referenced by mln::io::plot::save().

8.202.4.12 template<typename T > T & mln::image1d< T >::operator() (const point1d & *p*) [inline, package]

Read-write access to the image value located at point p.

References mln::image1d< T >::has().

8.202.4.13 template<typename T > const T & mln::image1d< T >::operator() (const point1d & *p*) const [inline, package]

Read-only access to the image value located at point p.

References mln::image1d< T >::has().

8.202.4.14 template<typename T > point1d mln::image1d< T >::point_at_index (unsigned *i*) const [inline, package]

Give the point corresponding to the offset o.

References mln::image1d< T >::nelements().

8.203 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.

- T * buffer ()

  Give a hook to the value buffer.

- const T * buffer () const

  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 (const box2d &b, unsigned bdr=border::thickness)

  Constructor with a box and the border thickness (default is 3).

- 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.

- 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.

- T & operator() (const point2d &p)

  Read-write access to the image value located at point p.

- const T & operator() (const point2d &p) const

  Read-only access to the image value located at point p.

- point2d point_at_index (unsigned i) const

  Give the point corresponding to the index i.

8.203.1 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.

8.203.2 Member Typedef Documentation

8.203.2.1 template<typename T> typedef T& mln::image2d< T >::lvalue

Return type of read-write access.

8.203.2.2 template<typename T> typedef const T& mln::image2d< T >::rvalue

Return type of read-only access.

8.203.2.3 template<typename T> typedef image2d< tag::value_<T> > mln::image2d< T >::skeleton

Skeleton.

8.203.2.4 template<typename T> typedef T mln::image2d< T >::value

Value associated type.

8.203.3 Constructor & Destructor Documentation

8.203.3.1 template<typename T > mln::image2d< T >::image2d () [inline]

Constructor without argument.

8.203.3.2 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.

References mln::make::box2d().

8.203.3.3 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).

8.203.4 Member Function Documentation

8.203.4.1 template<typename T > const box2d & mln::image2d< T >::bbox () const [inline]

Give the bounding box domain.

8.203.4.2 template<typename T > unsigned mln::image2d< T >::border () const [inline]

Give the border thickness.

8.203.4.3 template<typename T > T * mln::image2d< T >::buffer () [inline]

Give a hook to the value buffer.

8.203.4.4 template<typename T > const T * mln::image2d< T >::buffer () const [inline]

Give a hook to the value buffer.

8.203.4.5 template<typename T > int mln::image2d< T >::delta_index (const dpoint2d & *dp*) const [inline]

Give the delta-index corresponding to the delta-point `dp`.

8.203.4.6 template<typename T > const box2d & mln::image2d< T >::domain () const [inline]

Give the definition domain.

Referenced by mln::morpho::line_gradient(), mln::make_debug_graph_image(), and mln::io::txt::save().

8.203.4.7 template<typename T > T & mln::image2d< T >::element (unsigned *i*) [inline]

Read-write access to the image value located at index `i`.

References mln::image2d< T >::nelements().

8.203.4.8 template<typename T > const T & mln::image2d< T >::element (unsigned *i*) const [inline]

Read-only access to the image value located at index `i`.

References mln::image2d< T >::nelements().

8.203.4.9 template<typename T > bool mln::image2d< T >::has (const point2d & *p*) const [inline]

Test if `p` is valid.

Referenced by mln::image2d< T >::operator()(), and mln::debug::put_word().

8.203.4.10 template<typename T > unsigned mln::image2d< T >::ncols () const [inline]

Give the number of columns.

8.203.4.11 template<typename T > unsigned mln::image2d< T >::nelements () const [inline]

Give the number of elements (points including border ones).

Referenced by mln::image2d< T >::element(), and mln::image2d< T >::point_at_index().

8.203.4.12 template<typename T > unsigned mln::image2d< T >::nrows () const [inline]

Give the number of rows.

8.203.4.13 template<typename T > T & mln::image2d< T >::operator() (const point2d & *p*) [inline]

Read-write access to the image value located at point `p`.

References mln::image2d< T >::has().

8.203.4.14 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.

References mln::image2d< T >::has().

8.203.4.15 template<typename T > point2d mln::image2d< T >::point_at_index (unsigned *i*) const [inline]

Give the point corresponding to the index i.

References mln::image2d< T >::nelements().

8.204 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.

8.204.1 Detailed Description

template<typename V> struct mln::image2d_h< V >

2d image based on an hexagonal mesh.

8.204.2 Member Typedef Documentation

8.204.2.1 typedef hexa_bkd_piter_<box2d> mln::hexa< image2d< V > >::bkd_piter [inherited]

FIXME : should it be in box2d_h? Backward Site_Iterator associated type.

8.204.2.2 typedef hexa_fwd_piter_<box2d> mln::hexa< image2d< V > >::fwd_piter [inherited]

FIXME : should it be in box2d_h? Forward Site_Iterator associated type.

8.204.2.3 typedef image2d< V > ::lvalue mln::hexa< image2d< V > >::lvalue [inherited]

Lvalue associated type.

8.204.2.4 template<typename V > typedef point2d_h mln::image2d_h< V >::psite

Point site type.

Reimplemented from mln::hexa< image2d< V > >.

8.204.2.5 typedef image2d< V > ::rvalue mln::hexa< image2d< V > >::rvalue [inherited]

Return type of read-only access.

8.204.2.6 typedef hexa< tag::image_<image2d< V > > > mln::hexa< image2d< V > >::skeleton [inherited]

Skeleton.

8.204.2.7 typedef image2d< V >::value mln::hexa< image2d< V > >::value [inherited]

Value associated type.

8.204.3 Constructor & Destructor Documentation

8.204.3.1 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

8.204.4 Member Function Documentation

8.204.4.1 const box2d_h& mln::hexa< image2d< V > >::domain () const [inherited]

Give the definition domain.

8.204.4.2 bool mln::hexa< image2d< V > >::has (const psite & *p*) const [inherited]

Test if p belongs to the image domain.

8.204.4.3 lvalue mln::hexa< image2d< V > >::operator() (const point2d_h & *p*) [inherited]

Read-write access of pixel value at hexa point site p.

8.204.4.4 rvalue mln::hexa< image2d< V > >::operator() (const point2d_h & *p*) const [inherited]

Read-only access of pixel value at hexa point site p.

8.205 mln::image3d< T > Struct Template Reference

Basic 3D image class.

`#include <image3d.hh>`

Inherits image_primary< T, box3d, image3d< T > >.

Package 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.

Package Functions

- const box3d & bbox () const

  Give the bounding box domain.

- unsigned border () const

  Give the border thickness.

- T * buffer ()

  Give a hook to the value buffer.

- const T * buffer () const

  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.

- 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 point3d &p) const

Test if p *is valid.*

- image3d (int nslis, int nrows, int ncols, unsigned bdr=border::thickness)

  Constructor with the numbers of indexes and the border thickness.

- image3d (const box3d &b, unsigned bdr=border::thickness)

  Constructor with a box and the border thickness (default is 3).

- image3d ()

  Constructor without argument.

- 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 nslices () const

  Give the number of slices.

- T & operator() (const point3d &p)

  Read-write access to the image value located at point p.

- const T & operator() (const point3d &p) const

  Read-only access to the image value located at point p.

- point3d point_at_index (unsigned o) const

  Give the point corresponding to the offset o.

8.205.1 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.

8.205.2 Member Typedef Documentation

8.205.2.1 template<typename T> typedef T& mln::image3d< T >::lvalue [package]

Return type of read-write access.

8.205.2.2 template<typename T> typedef const T& mln::image3d< T >::rvalue [package]

Return type of read-only access.

8.205.2.3 template<typename T> typedef image3d< tag::value_<T> > mln::image3d< T >::skeleton [package]

Skeleton.

8.205.2.4 template<typename T> typedef T mln::image3d< T >::value [package]

Value associated type.

8.205.3 Constructor & Destructor Documentation

8.205.3.1 template<typename T > mln::image3d< T >::image3d () [inline, package]

Constructor without argument.

8.205.3.2 template<typename T > mln::image3d< T >::image3d (const box3d & *b*, unsigned *bdr* = border::thickness) [inline, package]

Constructor with a box and the border thickness (default is 3).

8.205.3.3 template<typename T > mln::image3d< T >::image3d (int *nslis*, int *nrows*, int *ncols*, unsigned *bdr* = border::thickness) [inline, package]

Constructor with the numbers of indexes and the border thickness.

References mln::make::box3d().

8.205.4 Member Function Documentation

8.205.4.1 template<typename T > const box3d & mln::image3d< T >::bbox () const [inline, package]

Give the bounding box domain.

8.205.4.2 template<typename T > unsigned mln::image3d< T >::border () const [inline, package]

Give the border thickness.

8.205.4.3 template<typename T > T * mln::image3d< T >::buffer () [inline, package]

Give a hook to the value buffer.

8.205.4.4 template<typename T > const T * mln::image3d< T >::buffer () const [inline, package]

Give a hook to the value buffer.

8.205.4.5 template<typename T > int mln::image3d< T >::delta_index (const dpoint3d & *dp*) const [inline, package]

Fast Image method.

Give the offset corresponding to the delta-point `dp`.

8.205.4.6 template<typename T > const box3d & mln::image3d< T >::domain () const [inline, package]

Give the definition domain.

8.205.4.7 template<typename T > T & mln::image3d< T >::element (unsigned *i*) [inline, package]

Read-write access to the image value located at index `i`.

References mln::image3d< T >::nelements().

8.205.4.8 template<typename T > const T & mln::image3d< T >::element (unsigned *i*) const [inline, package]

Read-only access to the image value located at index `i`.

References mln::image3d< T >::nelements().

8.205.4.9 template<typename T > bool mln::image3d< T >::has (const point3d & *p*) const [inline, package]

Test if `p` is valid.

Referenced by mln::image3d< T >::operator()().

8.205.4.10 template<typename T > unsigned mln::image3d< T >::ncols () const [inline, package]

Give the number of columns.

8.205.4.11 template<typename T > unsigned mln::image3d< T >::nelements () const [inline, package]

Give the number of cells (points including border ones).

Referenced by mln::image3d< T >::element(), and mln::image3d< T >::point_at_index().

8.205.4.12 template<typename T > unsigned mln::image3d< T >::nrows () const [inline, package]

Give the number of rows.

8.205.4.13 template<typename T > unsigned mln::image3d< T >::nslices () const [inline, package]

Give the number of slices.

8.205.4.14 template<typename T > T & mln::image3d< T >::operator() (const point3d & *p*) [inline, package]

Read-write access to the image value located at point p.

References mln::image3d< T >::has().

8.205.4.15 template<typename T > const T & mln::image3d< T >::operator() (const point3d & *p*) const [inline, package]

Read-only access to the image value located at point p.

References mln::image3d< T >::has().

8.205.4.16 template<typename T > point3d mln::image3d< T >::point_at_index (unsigned *o*) const [inline, package]

Give the point corresponding to the offset o.

References mln::image3d< T >::nelements().

8.206 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

- bool has (const mln::algebra::vec< I::psite::dim, float > &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.

8.206.1 Detailed Description

template<typename I, template< class > class F> struct mln::interpolated< I, F >

Makes the underlying image being accessed with floating coordinates.

8.206.2 Member Typedef Documentation

8.206.2.1 template<typename I , template< class > class F> typedef I ::lvalue mln::interpolated< I, F >::lvalue

Return type of read-write access.

8.206.2.2 template<typename I , template< class > class F> typedef I ::psite mln::interpolated< I, F >::psite

Point_Site associated type.

8.206.2.3 template<typename I , template< class > class F> typedef I ::rvalue mln::interpolated< I, F >::rvalue

Return type of read-only access.

8.206.2.4 template<typename I , template< class > class F> typedef interpolated< tag::image_<I>, F > mln::interpolated< I, F >::skeleton

Skeleton.

8.206.2.5 template<typename I , template< class > class F> typedef I ::value mln::interpolated< I, F >::value

Value associated type.

8.206.3 Constructor & Destructor Documentation

8.206.3.1 template<typename I , template< class > class F> mln::interpolated< I, F >::interpolated (I & *ima*) [inline]

Constructors.

FIXME: don't we want a 'const' here?

8.206.4 Member Function Documentation

8.206.4.1 template<typename I , template< class > class F> bool mln::interpolated< I, F >::has (const mln::algebra::vec< I::psite::dim, float > & *v*) const [inline]

Test if a pixel value is accessible at `v`.

8.206.4.2 template<typename I , template< class > class F> bool mln::interpolated< I, F >::is_valid () const [inline]

Test if this image has been initialized.

8.207 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.

8.207.1 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.

8.207.2 Member Function Documentation

8.207.2.1 template<typename E > void mln::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.

8.208 mln::labeled_image< I > Class Template Reference

Morpher providing an improved interface for labeled image.

`#include <labeled_image.hh>`

Inherits image_identity< const I, I::domain_t, 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 (const I &ima, const typename I::value &nlabels)

  Constructor from an image `ima` *and the number of labels* `nlabels`.

- labeled_image ()

  Constructors
  Constructor without argument.

- template<typename F >
  void relabel (const Function_v2b< F > &f)

  Labels may be removed.

- template<typename F >
  void relabel (const Function_v2v< F > &f)

  Relabel according to a function.

8.208.1 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).

8.208.2 Member Typedef Documentation

8.208.2.1 template<typename I> typedef accu::shape::bbox<typename I ::psite>::result mln::labeled_image< I >::bbox_t

Type of the bounding component bounding boxes.

8.208.2.2 template<typename I> typedef labeled_image< tag::image_<I> > mln::labeled_image< I >::skeleton

Skeleton.

8.208.3 Constructor & Destructor Documentation

8.208.3.1 template<typename I > mln::labeled_image< I >::labeled_image () [inline]

Constructors

Constructor without argument.

8.208.3.2 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`.

8.208.4 Member Function Documentation

8.208.4.1 template<typename I > const labeled_image< I >::bbox_t & mln::labeled_image< I >::bbox (const typename I::value & *label*) const [inline]

Return the bounding box of the component `label`.

Referenced by mln::labeled_image< I >::subdomain().

8.208.4.2 template<typename I > const util::array< typename labeled_image< I >::bbox_t > & mln::labeled_image< I >::bboxes () const [inline]

Return the component bounding boxes.

8.208.4.3 template<typename I > I::value mln::labeled_image< I >::nlabels () const [inline]

Return the number of labels;.

8.208.4.4 template<typename I > template<typename F > void mln::labeled_image< I >::relabel (const Function_v2b< F > & *f*) [inline]

Labels may be removed.

This overload make sure the labeling is still contiguous.

References mln::labeling::relabel_inplace().

8.208.4.5 template<typename I > template<typename F > void mln::labeled_image< I >::relabel (const Function_v2v< F > & *f*) [inline]

Relabel according to a function.

Labels may be removed and the labeling may not be contiguous afterwards. FIXME: currently the labels are packed after relabeling for performance reasons. Do we want to be less restrictive?

See also:

pack_().

References mln::labeling::relabel_inplace().

8.208.4.6 template<typename I > p_if< mln_box(I), fun::eq_v2b_expr_< pw::value_< I >, pw::cst_< typename I::value > > > mln::labeled_image< I >::subdomain (const typename I::value & *label*) const [inline]

Return the domain of the component with label `label`.

References mln::labeled_image< I >::bbox().

8.209 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.

- lvalue operator() (const typename I::psite &p)

  Read and "write if possible" access of pixel value at point site `p`.

- rvalue operator() (const typename I::psite &p) const

  Read-only 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`.

- F::result operator() (const typename F::input &x) const

  Read-only access of pixel value at F::input `x`.

8.209.1 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 tage a functor `fun` and a box `box`. Access to ima(p) where `p` include `box` return fun(b) lazily.

8.209.2 Member Typedef Documentation

8.209.2.1 template<typename I, typename F, typename B> typedef F ::result mln::lazy_image< I, F, B >::lvalue

Return type of read-write access.

8.209.2.2 template<typename I, typename F, typename B> typedef F ::result mln::lazy_image< I, F, B >::rvalue

Return type of read access.

8.209.2.3 template<typename I, typename F, typename B> typedef lazy_image< tag::image_<I>, F, B > mln::lazy_image< I, F, B >::skeleton

Skeleton.

8.209.3 Constructor & Destructor Documentation

8.209.3.1 template<typename I, typename F, typename B> mln::lazy_image< I, F, B >::lazy_image ()

Constructors.

8.209.3.2 template<typename I , typename F, typename B> mln::lazy_image< I, F, B >::lazy_image (const F & *fun*, const B & *box*) [inline]

Constructors.

8.209.4 Member Function Documentation

8.209.4.1 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.

8.209.4.2 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.

8.209.4.3 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.

8.209.4.4 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.

8.209.4.5 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.

8.209.4.6 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.

8.210 mln::Literal< E > Struct Template Reference

Base class for implementation classes of literals.

`#include <literal.hh>`

Inheritance diagram for mln::Literal< E >:

8.210.1 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.

8.211 mln::literal::black_t Struct Reference

Type of literal black.

`#include <black.hh>`

Inheritance diagram for mln::literal::black_t:

8.211.1 Detailed Description

Type of literal black.

8.212 mln::literal::blue_t Struct Reference

Type of literal blue.

`#include <colors.hh>`

Inheritance diagram for mln::literal::blue_t:

8.212.1 Detailed Description

Type of literal blue.

8.213 mln::literal::brown_t Struct Reference

Type of literal brown.

`#include <colors.hh>`

Inheritance diagram for mln::literal::brown_t:

8.213.1 Detailed Description

Type of literal brown.

8.214 mln::literal::cyan_t Struct Reference

Type of literal cyan.

`#include <colors.hh>`

Inheritance diagram for mln::literal::cyan_t:

8.214.1 Detailed Description

Type of literal cyan.

8.215 mln::literal::green_t Struct Reference

Type of literal green.

`#include <colors.hh>`

Inheritance diagram for mln::literal::green_t:

8.215.1 Detailed Description

Type of literal green.

8.216 mln::literal::identity_t Struct Reference

Type of literal identity.

`#include <identity.hh>`

Inheritance diagram for mln::literal::identity_t:

8.216.1 Detailed Description

Type of literal identity.

8.217 mln::literal::light_gray_t Struct Reference

Type of literal grays.

`#include <grays.hh>`

Inheritance diagram for mln::literal::light_gray_t:

8.217.1 Detailed Description

Type of literal grays.

8.218 mln::literal::lime_t Struct Reference

Type of literal lime.

`#include <colors.hh>`

Inheritance diagram for mln::literal::lime_t:

8.218.1 Detailed Description

Type of literal lime.

8.219 mln::literal::magenta_t Struct Reference

Type of literal magenta.

`#include <colors.hh>`

Inheritance diagram for mln::literal::magenta_t:

8.219.1 Detailed Description

Type of literal magenta.

8.220 mln::literal::max_t Struct Reference

Type of literal max.

`#include <max.hh>`

Inheritance diagram for mln::literal::max_t:

8.220.1 Detailed Description

Type of literal max.

8.221 mln::literal::min_t Struct Reference

Type of literal min.

`#include <min.hh>`

Inheritance diagram for mln::literal::min_t:

8.221.1 Detailed Description

Type of literal min.

8.222 mln::literal::olive_t Struct Reference

Type of literal olive.

`#include <colors.hh>`

Inheritance diagram for mln::literal::olive_t:

8.222.1 Detailed Description

Type of literal olive.

8.223 mln::literal::one_t Struct Reference

Type of literal one.

`#include <one.hh>`

Inheritance diagram for mln::literal::one_t:

8.223.1 Detailed Description

Type of literal one.

8.224 mln::literal::orange_t Struct Reference

Type of literal orange.

`#include <colors.hh>`

Inheritance diagram for mln::literal::orange_t:

8.224.1 Detailed Description

Type of literal orange.

8.225 mln::literal::origin_t Struct Reference

Type of literal origin.

`#include <origin.hh>`

Inheritance diagram for mln::literal::origin_t:

8.225.1 Detailed Description

Type of literal origin.

8.226 mln::literal::pink_t Struct Reference

Type of literal pink.

`#include <colors.hh>`

Inheritance diagram for mln::literal::pink_t:

8.226.1 Detailed Description

Type of literal pink.

8.227 mln::literal::purple_t Struct Reference

Type of literal purple.

`#include <colors.hh>`

Inheritance diagram for mln::literal::purple_t:

8.227.1 Detailed Description

Type of literal purple.

8.228 mln::literal::red_t Struct Reference

Type of literal red.

`#include <colors.hh>`

Inheritance diagram for mln::literal::red_t:

8.228.1 Detailed Description

Type of literal red.

8.229 mln::literal::teal_t Struct Reference

Type of literal teal.

`#include <colors.hh>`

Inheritance diagram for mln::literal::teal_t:

8.229.1 Detailed Description

Type of literal teal.

8.230 mln::literal::violet_t Struct Reference

Type of literal violet.

`#include <colors.hh>`

Inheritance diagram for mln::literal::violet_t:

8.230.1 Detailed Description

Type of literal violet.

8.231 mln::literal::white_t Struct Reference

Type of literal white.

`#include <white.hh>`

Inheritance diagram for mln::literal::white_t:

8.231.1 Detailed Description

Type of literal white.

8.232 mln::literal::yellow_t Struct Reference

Type of literal yellow.

`#include <colors.hh>`

Inheritance diagram for mln::literal::yellow_t:

8.232.1 Detailed Description

Type of literal yellow.

8.233 mln::literal::zero_t Struct Reference

Type of literal zero.

`#include <zero.hh>`

Inheritance diagram for mln::literal::zero_t:

8.233.1 Detailed Description

Type of literal zero.

8.234 mln::Mesh< E > Struct Template Reference

Base class for implementation classes of meshes.

`#include <mesh.hh>`

Inheritance diagram for mln::Mesh< E >:

8.234.1 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.

8.235 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 >:

8.235.1 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.

8.236 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 >:

8.236.1 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.

8.237 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 >:

8.237.1 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.

8.238 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 >:

8.238.1 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.

8.239 mln::metal::ands< E1, E2, E3, E4, E5, E6, E7, E8 > Struct Template Reference

Ands type.

`#include <ands.hh>`

Inherits bool_<(E1::value && E2::value && E3::value && E4::value && E5::value && E6::value && E7::value && E8::value)>.

8.239.1 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.

8.240 mln::metal::converts_to< T, U > Struct Template Reference

"converts-to" check.

`#include <converts_to.hh>`

Inheritance diagram for mln::metal::converts_to< T, U >:

8.240.1 Detailed Description

template<typename T, typename U> struct mln::metal::converts_to< T, U >

"converts-to" check.

8.241 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 >:

8.241.1 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.

8.242 mln::metal::goes_to< T, U > Struct Template Reference

"goes-to" check.

`#include <goes_to.hh>`

8.242.1 Detailed Description

template<typename T, typename U> struct mln::metal::goes_to< T, U >

"goes-to" check.

FIXME: Doc!

8.243 mln::metal::is< T, U > Struct Template Reference

"is" check.

`#include <is.hh>`

Inherits bool_<(sizeof(internal::helper_is_<T, U>::selector(internal::make_<T>::ptr())) == sizeof(internal::yes_))>.

8.243.1 Detailed Description

template<typename T, typename U> struct mln::metal::is< T, U >

"is" check.

Check whether T inherits from U.

8.244 mln::metal::is_a< T, M > Struct Template Reference

"is_a" check.

`#include <is_a.hh>`

Inherits bool_<(sizeof(internal::helper_is_a_< T, M >::selector(internal::make_< T >::ptr())) == sizeof(internal::yes_))>.

8.244.1 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.

8.245 mln::metal::is_not< T, U > Struct Template Reference

"is_not" check.

`#include <is_not.hh>`

8.245.1 Detailed Description

template<typename T, typename U> struct mln::metal::is_not< T, U >

"is_not" check.

FIXME: Doc!

8.246 mln::metal::is_not_a< T, M > Struct Template Reference

"is_not_a" static Boolean expression.

`#include <is_not_a.hh>`

8.246.1 Detailed Description

template<typename T, template< class > class M> struct mln::metal::is_not_a< T, M >

"is_not_a" static Boolean expression.

8.247 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.

8.247.1 Detailed Description

template<typename I> class mln::morpho::attribute::card< I >

Cardinality accumulator class.

8.247.2 Member Function Documentation

8.247.2.1 template<typename I > void mln::morpho::attribute::card< I >::init () `[inline]`

Manipulators.

8.247.2.2 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.

8.247.2.3 void mln::Accumulator< card< I > >::take_as_init (const T & *t*) `[inline, inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.247.2.4 void mln::Accumulator< card< I > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take n times the value t.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.247.2.5 template<typename I > unsigned mln::morpho::attribute::card< I >::to_result () const [inline]

Get the value of the accumulator.

8.248 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.

8.248.1 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.

8.248.2 Member Function Documentation

8.248.2.1 template<typename I > void mln::morpho::attribute::count_adjacent_vertices< I >::init () [inline]

Manipulators.

8.248.2.2 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.

8.248.2.3 void mln::Accumulator< count_adjacent_vertices< I > >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.248.2.4 void mln::Accumulator< count_adjacent_vertices< I > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.248.2.5 template<typename I > unsigned mln::morpho::attribute::count_adjacent_vertices< I >::to_result () const [inline]

Get the value of the accumulator.

8.249 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 `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.

8.249.1 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.

8.249.2 Member Function Documentation

8.249.2.1 template<typename I > unsigned mln::morpho::attribute::height< I >::base_level () const [inline]

Get base & current level of the accumulator.

8.249.2.2 template<typename I > void mln::morpho::attribute::height< I >::init () [inline]

Manipulators.

8.249.2.3 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.

Referenced by mln::morpho::attribute::height< I >::to_result().

8.249.2.4 void mln::Accumulator< height< I > >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.249.2.5 void mln::Accumulator< height< I > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.249.2.6 template<typename I > unsigned mln::morpho::attribute::height< I >::to_result () const [inline]

Get the value of the accumulator.

References mln::morpho::attribute::height< I >::is_valid().

8.250 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.

8.250.1 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.

8.250.2 Member Function Documentation

8.250.2.1 template<typename I > unsigned mln::morpho::attribute::sharpness< I >::area () const [inline]

Give the area of the component.

8.250.2.2 template<typename I > unsigned mln::morpho::attribute::sharpness< I >::height () const [inline]

Give the height.

8.250.2.3 template<typename I > void mln::morpho::attribute::sharpness< I >::init () [inline]

Manipulators.

8.250.2.4 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.

8.250.2.5 void mln::Accumulator< sharpness< I > >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.250.2.6 void mln::Accumulator< sharpness< I > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.250.2.7 template<typename I > double mln::morpho::attribute::sharpness< I >::to_result () const [inline]

Get the value of the accumulator.

8.250.2.8 template<typename I > unsigned mln::morpho::attribute::sharpness< I >::volume () const [inline]

Give the volume of the component.

8.251 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 accumalator.

- 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.

8.251.1 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.

8.251.2 Member Function Documentation

8.251.2.1 template<typename I , typename S > void mln::morpho::attribute::sum< I, S >::init () [inline]

Manipulators.

References mln::literal::zero.

8.251.2.2 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.

8.251.2.3 template<typename I , typename S > void mln::morpho::attribute::sum< I, S >::set_value (const argument & *v*) [inline]

Set the return value of the accumalator.

8.251.2.4 void mln::Accumulator< sum< I, S > >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.251.2.5 void mln::Accumulator< sum< I, S > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.251.2.6 template<typename I , typename S > S mln::morpho::attribute::sum< I, S >::to_result () const [inline]

Get the value of the accumulator.

8.251.2.7 template<typename I , typename S > void mln::morpho::attribute::sum< I, S >::untake (const argument & *v*) [inline]

Untake a value from the accumulator.

8.252 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 `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.

8.252.1 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.

8.252.2 Member Function Documentation

8.252.2.1 template<typename I > unsigned mln::morpho::attribute::volume< I >::area () const [inline]

Give the area.

8.252.2.2 template<typename I > void mln::morpho::attribute::volume< I >::init () [inline]

Manipulators.

8.252.2.3 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.

8.252.2.4 void mln::Accumulator< volume< I > >::take_as_init (const T & *t*) [inline, inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.252.2.5 void mln::Accumulator< volume< I > >::take_n_times (unsigned *n*, const T & *t*) [inline, inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

8.252.2.6 template<typename I > unsigned mln::morpho::attribute::volume< I >::to_result () const [inline]

Get the value of the accumulator.

8.253 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

- void change_window (const W &new_win)

  Change the corresponding window.

- neighb (const W &win)

  Constructor from a window `win`.

- neighb ()

  Constructor without argument.

- const W & win () const

  Get the corresponding window.

8.253.1 Detailed Description

template<typename W> class mln::neighb< W >

Adapter class from window to neighborhood.

8.253.2 Member Typedef Documentation

8.253.2.1 template<typename W> typedef neighb_bkd_niter<W> mln::neighb< W >::bkd_niter

Backward site iterator associated type.

8.253.2.2 template<typename W> typedef neighb_fwd_niter<W> mln::neighb< W >::fwd_niter

Forward site iterator associated type.

8.253.2.3 template<typename W> typedef fwd_niter mln::neighb< W >::niter

Site iterator associated type.

8.253.3 Constructor & Destructor Documentation

8.253.3.1 template<typename W > mln::neighb< W >::neighb () [inline]

Constructor without argument.

8.253.3.2 template<typename W> mln::neighb< W >::neighb (const W & *win*) [inline]

Constructor from a window `win`.

References mln::neighb< W >::change_window().

8.253.4 Member Function Documentation

8.253.4.1 template<typename W> void mln::neighb< W >::change_window (const W & *new_win*) [inline]

Change the corresponding window.

Referenced by mln::neighb< W >::neighb().

8.253.4.2 template<typename W > const W & mln::neighb< W >::win () const [inline]

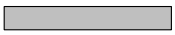
Get the corresponding window.

8.254 mln::Neighborhood< E > Struct Template Reference

Base class for implementation classes that are neighborhoods.

`#include <neighborhood.hh>`

Inheritance diagram for mln::Neighborhood< E >:



8.254.1 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.

8.255 mln::Neighborhood< void > Struct Template Reference

Neighborhood category flag type.

`#include <neighborhood.hh>`

8.255.1 Detailed Description

template<> struct mln::Neighborhood< void >

Neighborhood category flag type.

8.256 mln::Object< E > Struct Template Reference

Base class for almost every class defined in Milena.

`#include <object.hh>`

8.256.1 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.

8.257 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.

- internal::morpher_lvalue_< I >::ret operator() (const typename I::psite &p)

  Read-write access to the image value located at point `p`.

- I::rvalue operator() (const typename I::psite &p) const

  Read-only 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.

8.257.1 Detailed Description

template<typename I, typename F> struct mln::p2p_image< I, F >

FIXME: Doc!

8.257.2 Member Typedef Documentation

8.257.2.1 template<typename I, typename F> typedef p2p_image< tag::image_<I>, tag::function_<F> > mln::p2p_image< I, F >::skeleton

Skeleton.

8.257.3 Constructor & Destructor Documentation

8.257.3.1 template<typename I , typename F > mln::p2p_image< I, F >::p2p_image () [inline]

Constructor without argument.

8.257.3.2 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`.

8.257.4 Member Function Documentation

8.257.4.1 template<typename I , typename F > const I::domain_t & mln::p2p_image< I, F >::domain () const [inline]

Give the definition domain.

8.257.4.2 template<typename I , typename F > const F & mln::p2p_image< I, F >::fun () const [inline]

Give the p2p function.

8.257.4.3 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`.

8.257.4.4 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`.

8.258 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_array< P > &other)
  Append an array `other` *of points.*
- p_array< P > & append (const P &p)
  Append a point `p`.
- 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 (const std::vector< P > &vect)

  Constructor from a vector `vect`.

- p_array ()

  Constructor.

- void reserve (size_type n)

  Reserve `n` *cells.*

- const std::vector< P > & std_vector () const

  Return the corresponding std::vector of points.

8.258.1 Detailed Description

template<typename P> class mln::p_array< P >

Multi-set of sites.

Site set class based on std::vector.

8.258.2 Member Typedef Documentation

8.258.2.1 template<typename P> typedef p_indexed_bkd_piter<self_> mln::p_array< P >::bkd_piter

Backward Site_Iterator associated type.

8.258.2.2 template<typename P> typedef P mln::p_array< P >::element

Element associated type.

8.258.2.3 template<typename P> typedef p_indexed_fwd_piter<self_> mln::p_array< P >::fwd_piter

Forward Site_Iterator associated type.

8.258.2.4 template<typename P> typedef P mln::p_array< P >::i_element

Insertion element associated type.

8.258.2.5 template<typename P> typedef fwd_piter mln::p_array< P >::piter

Site_Iterator associated type.

8.258.2.6 template<typename P> typedef p_indexed_psite<self_> mln::p_array< P >::psite

Psite associated type.

8.258.3 Constructor & Destructor Documentation

8.258.3.1 template<typename P > mln::p_array< P >::p_array () [inline]

Constructor.

8.258.3.2 template<typename P > mln::p_array< P >::p_array (const std::vector< P > & *vect*) [inline]

Constructor from a vector `vect`.

8.258.4 Member Function Documentation

8.258.4.1 template<typename P > p_array< P > & mln::p_array< P >::append (const p_array< P > & *other*) [inline]

Append an array `other` of points.

References mln::p_array< P >::std_vector().

8.258.4.2 template<typename P > p_array< P > & mln::p_array< P >::append (const P & *p*) [inline]

Append a point `p`.

Referenced by mln::convert::to_p_array().

8.258.4.3 template<typename P > void mln::p_array< P >::change (const psite & *p*, const P & *new_p*) [inline]

Change site p into new_p.

References mln::p_array< P >::has().

8.258.4.4 template<typename P > void mln::p_array< P >::clear () [inline]

Clear this set.

8.258.4.5 template<typename P > bool mln::p_array< P >::has (const util::index & *i*) const [inline]

Test is index i belongs to this site set.

References mln::p_array< P >::nsites().

8.258.4.6 template<typename P > bool mln::p_array< P >::has (const psite & *p*) const [inline]

Test is p belongs to this site set.

Referenced by mln::p_array< P >::change(), and mln::p_array< P >::operator[ ]().

8.258.4.7 template<typename P > void mln::p_array< P >::insert (const P & *p*) [inline]

Insert a point p (equivalent as 'append').

8.258.4.8 template<typename P > bool mln::p_array< P >::is_valid () const [inline]

Test this set validity so returns always true.

8.258.4.9 template<typename P > std::size_t mln::p_array< P >::memory_size () const [inline]

Return the size of this site set in memory.

References mln::p_array< P >::nsites().

8.258.4.10 template<typename P > unsigned mln::p_array< P >::nsites () const [inline]

Give the number of sites.

Referenced by mln::registration::get_rot(), mln::p_array< P >::has(), mln::p_array< P >::memory_size(), and mln::p_array< P >::operator[ ]().

8.258.4.11 template<typename P > const P & mln::p_array< P >::operator[ ] (const util::index & *i*) const [inline]

Return the i-th element.

References mln::p_array< P >::has().

8.258.4.12 template<typename P > P & mln::p_array< P >::operator[ ] (unsigned *i*) [inline]

Return the `i-th` site (mutable).

References mln::p_array< P >::nsites().

8.258.4.13 template<typename P > const P & mln::p_array< P >::operator[ ] (unsigned *i*) const [inline]

Return the `i-th` site (constant).

References mln::p_array< P >::nsites().

8.258.4.14 template<typename P > void mln::p_array< P >::reserve (size_type *n*) [inline]

Reserve `n` cells.

8.258.4.15 template<typename P > const std::vector< P > & mln::p_array< P >::std_vector () const [inline]

Return the corresponding std::vector of points.

Referenced by mln::p_array< P >::append().

8.259 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.

8.259.1 Detailed Description

template<typename W> class mln::p_centered< W >

Site set corresponding to a window centered on a site.

8.259.2 Member Typedef Documentation

8.259.2.1 template<typename W> typedef p_centered_piter<W> mln::p_centered< W >::bkd_piter

Backward Site_Iterator associated type.

8.259.2.2 template<typename W> typedef psite mln::p_centered< W >::element

Element associated type.

8.259.2.3 template<typename W> typedef p_centered_piter<W> mln::p_centered< W >::fwd_piter

Forward Site_Iterator associated type.

8.259.2.4 template<typename W> typedef fwd_piter mln::p_centered< W >::piter

Site_Iterator associated type.

8.259.2.5 template<typename W> typedef W ::psite mln::p_centered< W >::psite

Psite associated type.

8.259.2.6 template<typename W> typedef W ::site mln::p_centered< W >::site

Site associated type.

8.259.3 Constructor & Destructor Documentation

8.259.3.1 template<typename W > mln::p_centered< W >::p_centered () [inline]

Constructor without argument.

8.259.3.2 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`.

References mln::p_centered< W >::is_valid().

8.259.4 Member Function Documentation

8.259.4.1 template<typename W > const W::psite & mln::p_centered< W >::center () const [inline]

Give the center of this site set.

8.259.4.2 template<typename W > template<typename P > bool mln::p_centered< W >::has (const P & *p*) const [inline]

Test if `p` belongs to the box.

References mln::p_centered< W >::is_valid().

8.259.4.3 template<typename W > bool mln::p_centered< W >::is_valid () const [inline]

Test if this site set is initialized.

Referenced by mln::p_centered< W >::has(), and mln::p_centered< W >::p_centered().

8.259.4.4 template<typename W > std::size_t mln::p_centered< W >::memory_size () const [inline]

Return the size of this site set in memory.

8.259.4.5 template<typename W > const W & mln::p_centered< W >::window () const [inline]

Give the window this site set is defined upon.

8.260 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 p_complex_bkd_piter_< D, G > bkd_piter
  Backward Site_Iterator associated type.
- typedef super_::site element
  Associated types.
- typedef p_complex_fwd_piter_< D, G > fwd_piter
  Forward Site_Iterator associated type.
- typedef fwd_piter piter
  Site_Iterator associated type.
- typedef complex_psite< D, G > psite
  Point_Site 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 ()
  Return the complex associated to the p_complex domain (mutable version).
- topo::complex< D > & cplx () const
  Accessors.

- const G & geom () const

  Return the geometry of the complex.

8.260.1 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.

8.260.2 Member Typedef Documentation

8.260.2.1 template<unsigned D, typename G> typedef p_complex_bkd_piter_<D, G> mln::p_complex< D, G >::bkd_piter

Backward Site_Iterator associated type.

8.260.2.2 template<unsigned D, typename G> typedef super_ ::site mln::p_complex< D, G >::element

Associated types.

Element associated type.

8.260.2.3 template<unsigned D, typename G> typedef p_complex_fwd_piter_<D, G> mln::p_complex< D, G >::fwd_piter

Forward Site_Iterator associated type.

8.260.2.4 template<unsigned D, typename G> typedef fwd_piter mln::p_complex< D, G >::piter

Site_Iterator associated type.

8.260.2.5 template<unsigned D, typename G> typedef complex_psite<D, G> mln::p_complex< D, G >::psite

Point_Site associated type.

8.260.3 Constructor & Destructor Documentation

8.260.3.1 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

8.260.4 Member Function Documentation

8.260.4.1 template<unsigned D, typename G > topo::complex< D > & mln::p_complex< D, G >::cplx () [inline]

Return the complex associated to the p_complex domain (mutable version).

References mln::p_complex< D, G >::is_valid().

8.260.4.2 template<unsigned D, typename G > topo::complex< D > & mln::p_complex< D, G >::cplx () const [inline]

Accessors.

Return the complex associated to the p_complex domain (const version)

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==().

8.260.4.3 template<unsigned D, typename G > const G & mln::p_complex< D, G >::geom () const [inline]

Return the geometry of the complex.

8.260.4.4 template<unsigned D, typename G > bool mln::p_complex< D, G >::has (const psite & *p*) const [inline]

Does this site set has p?

References mln::complex_psite< D, G >::is_valid(), mln::p_complex< D, G >::is_valid(), and mln::complex_psite< D, G >::site_set().

8.260.4.5 template<unsigned D, typename G > bool mln::p_complex< D, G >::is_valid () const [inline]

Is this site set valid?

Referenced by mln::p_complex< D, G >::cplx(), and mln::p_complex< D, G >::has().

8.260.4.6 template<unsigned D, typename G > unsigned mln::p_complex< D, G >::nfaces () const [inline]

Return the number of faces in the complex.

Referenced by mln::p_complex< D, G >::nsites().

8.260.4.7 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.

8.260.4.8 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'.)

References mln::p_complex< D, G >::nfaces().

8.261 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 p_graph_piter< self_, mln_edge_bkd_iter(G) > bkd_piter
  Backward Site_Iterator associated type.
- typedef super_::site element
  Associated types.
- typedef p_graph_piter< self_, mln_edge_fwd_iter(G) > fwd_piter
  Forward Site_Iterator associated type.
- typedef fwd_piter piter
  Site_Iterator associated type.
- typedef p_edges_psite< G, F > psite
  Point_Site associated type.

Public Member Functions

- template<typename G2 >
  bool has (const util::edge< G2 > &e) const
  Does this site set has edge e*?*
- bool has (const psite &p) const
  Does this site set has site p*?*
- 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*.*

- const F & function () const

  Return the mapping function.

- const G & graph () const

  Accessors.

- 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.

- p_edges (const Graph< G > &gr, const Function< F > &f)

  Construct a graph edge psite set from a graph and a function.

- p_edges (const Graph< G > &gr)

  Construct a graph edge psite set from a graph.

- p_edges ()

  Constructors
  Default constructor.

8.261.1 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.

8.261.2 Member Typedef Documentation

8.261.2.1 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.

8.261.2.2 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.

8.261.2.3 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.

8.261.2.4 template<typename G, typename F = util::internal::id2element<G,util::edge<G> >> typedef F mln::p_edges< G, F >::fun_t

Function associated type.

8.261.2.5 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.

8.261.2.6 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.

8.261.2.7 template<typename G, typename F = util::internal::id2element<G,util::edge<G> >> typedef G mln::p_edges< G, F >::graph_t

Graph associated type.

8.261.2.8 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.

8.261.2.9 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.

8.261.3 Constructor & Destructor Documentation

8.261.3.1 template<typename G , typename F > mln::p_edges< G, F >::p_edges () [inline]

Constructors

Default constructor.

8.261.3.2 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.

References mln::p_edges< G, F >::is_valid().

8.261.3.3 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.

References mln::p_edges< G, F >::is_valid().

8.261.3.4 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`.

References mln::p_edges< G, F >::is_valid().

8.261.4 Member Function Documentation

8.261.4.1 template<typename G , typename F > const F & mln::p_edges< G, F >::function () const [inline]

Return the mapping function.

8.261.4.2 template<typename G , typename F > const G & mln::p_edges< G, F >::graph () const [inline]

Accessors.

Return the graph associated to this site set

References mln::p_edges< G, F >::is_valid().

Referenced by mln::operator==().

8.261.4.3 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*?

References mln::util::edge< G >::graph(), mln::util::edge< G >::is_valid(), and mln::p_edges< G, F >::is_valid().

8.261.4.4 template<typename G , typename F > bool mln::p_edges< G, F >::has (const psite & *p*) const [inline]

Does this site set has site *p*?

References mln::p_edges< G, F >::is_valid().

8.261.4.5 template<typename G , typename F > void mln::p_edges< G, F >::invalidate () [inline]

Invalidate this site set.

8.261.4.6 template<typename G , typename F > bool mln::p_edges< G, F >::is_valid () const [inline]

Is this site set valid?

Referenced by mln::p_edges< G, F >::graph(), mln::p_edges< G, F >::has(), and mln::p_edges< G, F >::p_edges().

8.261.4.7 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;.

8.261.4.8 template<typename G , typename F > unsigned mln::p_edges< G, F >::nedges () const [inline]

Return The number of edges in the graph.

Referenced by mln::p_edges< G, F >::nsites().

8.261.4.9 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*.

References mln::p_edges< G, F >::nedges().

8.262 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 > >.

Package Types

- typedef p_faces_bkd_piter_< N, D, P > bkd_piter
  Backward Site_Iterator associated type.
- typedef super_::site element
  Associated types.
- typedef p_faces_fwd_piter_< N, D, P > fwd_piter
  Forward Site_Iterator associated type.
- typedef fwd_piter piter
  Site_Iterator associated type.
- typedef faces_psite< N, D, P > psite
  Point_Site associated type.

Package 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 p_complex< D, P > &pc)
  Construct a faces psite set from an mln::p_complex.
- p_faces (const topo::complex< D > &cplx)
  Construct a faces psite set from an mln::complex.
- topo::complex< D > & cplx ()
  Return the complex associated to the p_faces domain (mutable version).
- topo::complex< D > & cplx () const
  Accessors.

8.262.1 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).

8.262.2 Member Typedef Documentation

8.262.2.1 template<unsigned N, unsigned D, typename P> typedef p_faces_bkd_piter_<N, D, P> mln::p_faces< N, D, P >::bkd_piter [package]

Backward Site_Iterator associated type.

8.262.2.2 template<unsigned N, unsigned D, typename P> typedef super_ ::site mln::p_faces< N, D, P >::element [package]

Associated types.

Element associated type.

8.262.2.3 template<unsigned N, unsigned D, typename P> typedef p_faces_fwd_piter_<N, D, P> mln::p_faces< N, D, P >::fwd_piter [package]

Forward Site_Iterator associated type.

8.262.2.4 template<unsigned N, unsigned D, typename P> typedef fwd_piter mln::p_faces< N, D, P >::piter [package]

Site_Iterator associated type.

8.262.2.5 template<unsigned N, unsigned D, typename P> typedef faces_psite<N, D, P> mln::p_faces< N, D, P >::psite [package]

Point_Site associated type.

8.262.3 Constructor & Destructor Documentation

8.262.3.1 template<unsigned N, unsigned D, typename P > mln::p_faces< N, D, P >::p_faces (const topo::complex< D > & *cplx*) [inline, package]

Construct a faces psite set from an mln::complex.

Parameters:

cplx The complex upon which the complex psite set is built.

8.262.3.2 template<unsigned N, unsigned D, typename P > mln::p_faces< N, D, P >::p_faces (const p_complex< D, P > & *pc*) [inline, package]

Construct a faces psite set from an mln::p_complex.

Parameters:

pc The complex upon which the complex psite set is built.

8.262.4 Member Function Documentation

8.262.4.1 template<unsigned N, unsigned D, typename P > topo::complex< D > & mln::p_faces< N, D, P >::cplx () [inline, package]

Return the complex associated to the p_faces domain (mutable version).

References mln::p_faces< N, D, P >::is_valid().

8.262.4.2 template<unsigned N, unsigned D, typename P > topo::complex< D > & mln::p_faces< N, D, P >::cplx () const [inline, package]

Accessors.

Return the complex associated to the p_faces domain (const version).

References mln::p_faces< N, D, P >::is_valid().

Referenced by mln::operator==().

8.262.4.3 template<unsigned N, unsigned D, typename P > bool mln::p_faces< N, D, P >::is_valid () const [inline, package]

Is this site set valid?

Referenced by mln::p_faces< N, D, P >::cplx().

8.262.4.4 template<unsigned N, unsigned D, typename P > unsigned mln::p_faces< N, D, P >::nfaces () const [inline, package]

Return The number of faces in the complex.

Referenced by mln::p_faces< N, D, P >::nsites().

8.262.4.5 template<unsigned N, unsigned D, typename P > unsigned mln::p_faces< N, D, P >::nsites () const [inline, package]

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'.)

References mln::p_faces< N, D, P >::nfaces().

8.263 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.

8.263.1 Detailed Description

template<typename S, typename I> class mln::p_graph_piter< S, I >

Generic iterator on point sites of a mln::S.

8.263.2 Constructor & Destructor Documentation

8.263.2.1 template<typename S , typename I > mln::p_graph_piter< S, I >::p_graph_piter () [inline]

Constructors.

8.263.3 Member Function Documentation

8.263.3.1 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.

8.263.3.2 template<typename S , typename I > unsigned mln::p_graph_piter< S, I >::id () const [inline]

Return the graph element id.

8.263.3.3 template<typename S , typename I > mln::p_graph_piter< S, I >::mln_q_subject (iter)

Return the underlying graph element.

8.263.3.4 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.

8.264 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.

8.264.1 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.

8.264.2 Member Typedef Documentation

8.264.2.1 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.

8.264.2.2 template<typename S, typename F> typedef S ::element mln::p_if< S, F >::element

Element associated type.

8.264.2.3 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.

8.264.2.4 template<typename S, typename F> typedef fwd_piter mln::p_if< S, F >::piter

Site_Iterator associated type.

8.264.2.5 template<typename S, typename F> typedef S ::psite mln::p_if< S, F >::psite

Psite associated type.

8.264.3 Constructor & Destructor Documentation

8.264.3.1 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`.

8.264.3.2 template<typename S , typename F > mln::p_if< S, F >::p_if () `[inline]`

Constructor without argument.

8.264.4 Member Function Documentation

8.264.4.1 template<typename S , typename F > bool mln::p_if< S, F >::has (const psite & *p*) const [inline]

Test if p belongs to the subset.

References mln::p_if< S, F >::has().

Referenced by mln::p_if< S, F >::has().

8.264.4.2 template<typename S , typename F > bool mln::p_if< S, F >::is_valid () const [inline]

Test if this site set is valid.

8.264.4.3 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.

8.264.4.4 template<typename S , typename F > const S & mln::p_if< S, F >::overset () const [inline]

Give the primary overset.

8.264.4.5 template<typename S , typename F > bool mln::p_if< S, F >::pred (const psite & *p*) const [inline]

Test predicate on point site p.

8.264.4.6 template<typename S , typename F > const F & mln::p_if< S, F >::predicate () const [inline]

Give the predicate function.

8.265 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 is 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 (const I &ima)
  Constructor.
- p_image ()
  Constructor without argument.
- void remove (const psite &p)
  Remove a site p.
- void toggle (const psite &p)
  Change the status in/out of a site p.

8.265.1 Detailed Description

template<typename I> class mln::p_image< I >

Site set based on an image of Booleans.

8.265.2 Member Typedef Documentation

8.265.2.1 template<typename I > typedef S ::bkd_piter mln::p_image< I >::bkd_piter

Backward Site_Iterator associated type.

8.265.2.2 template<typename I > typedef I ::psite mln::p_image< I >::element

Element associated type.

8.265.2.3 template<typename I > typedef S ::fwd_piter mln::p_image< I >::fwd_piter

Forward Site_Iterator associated type.

8.265.2.4 template<typename I > typedef psite mln::p_image< I >::i_element

Insertion element associated type.

8.265.2.5 template<typename I > typedef S ::piter mln::p_image< I >::piter

Site_Iterator associated type.

8.265.2.6 template<typename I > typedef I ::psite mln::p_image< I >::psite

Psite associated type.

8.265.2.7 template<typename I > typedef psite mln::p_image< I >::r_element

Removal element associated type.

8.265.2.8 template<typename I > typedef internal::p_image_site_set<I>::ret mln::p_image< I >::S

Equivalent site_set type.

8.265.3 Constructor & Destructor Documentation

8.265.3.1 template<typename I > mln::p_image< I >::p_image () [inline]

Constructor without argument.

8.265.3.2 template<typename I > mln::p_image< I >::p_image (const I & *ima*) [inline]

Constructor.

References mln::p_image< I >::clear().

8.265.4 Member Function Documentation

8.265.4.1 template<typename I > void mln::p_image< I >::clear () [inline]

Clear this set.

References mln::data::fill_with_value(), and mln::p_image< I >::is_valid().

Referenced by mln::p_image< I >::p_image().

8.265.4.2 template<typename I > bool mln::p_image< I >::has (const psite & *p*) const [inline]

Test is the psite p belongs to this site set.

References mln::p_image< I >::is_valid().

8.265.4.3 template<typename I > void mln::p_image< I >::insert (const psite & *p*) [inline]

Insert a site p.

References mln::p_image< I >::is_valid().

8.265.4.4 template<typename I > bool mln::p_image< I >::is_valid () const [inline]

Test if this site set is valid, i.e., initialized.

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().

8.265.4.5 template<typename I > std::size_t mln::p_image< I >::memory_size () const [inline]

Return the size of this site set in memory.

References mln::p_image< I >::is_valid().

8.265.4.6 template<typename I > unsigned mln::p_image< I >::nsites () const [inline]

Give the number of sites.

8.265.4.7 template<typename I > mln::p_image< I >::operator typename internal::p_image_site_set< I >::ret () const [inline]

Conversion towards the equivalent site set.

8.265.4.8 template<typename I > void mln::p_image< I >::remove (const psite & *p*) [inline]

Remove a site p.

References mln::p_image< I >::is_valid().

8.265.4.9 template<typename I > void mln::p_image< I >::toggle (const psite & *p*) [inline]

Change the status in/out of a site p.

References mln::p_image< I >::is_valid().

8.266 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.
- p_indexed_bkd_piter (const S &s)
  Constructor.
- p_indexed_bkd_piter ()
  Constructor with no argument.

8.266.1 Detailed Description

template<typename S> class mln::p_indexed_bkd_piter< S >

Backward iterator on sites of an indexed site set.

8.266.2 Constructor & Destructor Documentation

8.266.2.1 template<typename S > mln::p_indexed_bkd_piter< S >::p_indexed_bkd_piter ()

`[inline]`

Constructor with no argument.

8.266.2.2 template<typename S > mln::p_indexed_bkd_piter< S >::p_indexed_bkd_piter (const S & *s*)

`[inline]`

Constructor.

8.266.3 Member Function Documentation

8.266.3.1 template<typename S > int mln::p_indexed_bkd_piter< S >::index () const

`[inline]`

Return the current index.

8.267 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.
- p_indexed_fwd_piter (const S &s)
  Constructor.
- p_indexed_fwd_piter ()
  Constructor with no argument.

8.267.1 Detailed Description

template<typename S> class mln::p_indexed_fwd_piter< S >

Forward iterator on sites of an indexed site set.

8.267.2 Constructor & Destructor Documentation

8.267.2.1 template<typename S > mln::p_indexed_fwd_piter< S >::p_indexed_fwd_piter () [inline]

Constructor with no argument.

8.267.2.2 template<typename S > mln::p_indexed_fwd_piter< S >::p_indexed_fwd_piter (const S & *s*) [inline]

Constructor.

8.267.3 Member Function Documentation

8.267.3.1 template<typename S > int mln::p_indexed_fwd_piter< S >::index () const [inline]

Return the current index.

8.268 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 > >.

8.268.1 Detailed Description

template<typename S> class mln::p_indexed_psite< S >

Psite class for indexed site sets such as p_array.

.

8.269 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 P &p) const

Test is the psite `p` *belongs to this site set.*

- bool has (const psite &) const

  Test is the psite `p` *belongs to this site set.*

- void insert (const K &k, const P &p)

  Insert a pair (key `k`*, site* `p`*).*

- void insert (const i_element &k_p)

  Insert a pair `k_p` *(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`*.*

8.269.1 Detailed Description

template<typename K, typename P> class mln::p_key< K, P >

Priority queue class.

8.269.2 Member Typedef Documentation

8.269.2.1 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.

8.269.2.2 template<typename K , typename P > typedef P mln::p_key< K, P >::element

Element associated type.

8.269.2.3 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.

8.269.2.4 template<typename K , typename P > typedef std::pair<K,P> mln::p_key< K, P >::i_element

Insertion element associated type.

8.269.2.5 template<typename K , typename P > typedef fwd_piter mln::p_key< K, P >::piter

Site_Iterator associated type.

8.269.2.6 template<typename K , typename P > typedef p_double_psite< self_, p_set<P> > mln::p_key< K, P >::psite

Psite associated type.

8.269.2.7 template<typename K , typename P > typedef P mln::p_key< K, P >::r_element

Removal element associated type.

8.269.3 Constructor & Destructor Documentation

8.269.3.1 template<typename K , typename P > mln::p_key< K, P >::p_key () [inline]

Constructor.

8.269.4 Member Function Documentation

8.269.4.1 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`.

References mln::p_set< P >::nsites().

8.269.4.2 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`.

References mln::util::set< T >::insert().

8.269.4.3 template<typename K , typename P > void mln::p_key< K, P >::clear () [inline]

Clear this site set.

8.269.4.4 template<typename K , typename P > bool mln::p_key< K, P >::exists_key (const K & *key*) const [inline]

Test if the `priority` exists.

Referenced by mln::p_key< K, P >::operator()().

8.269.4.5 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.

8.269.4.6 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.

Referenced by mln::p_key< K, P >::insert().

8.269.4.7 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`).

References mln::p_key< K, P >::has().

8.269.4.8 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).

8.269.4.9 template<typename K , typename P > bool mln::p_key< K, P >::is_valid () const [inline]

Test this set validity so returns always true.

8.269.4.10 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.

8.269.4.11 template<typename K , typename P > const util::set< K > & mln::p_key< K, P >::keys () const [inline]

Give the set of keys.

8.269.4.12 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.

8.269.4.13 template<typename K , typename P > unsigned mln::p_key< K, P >::nsites () const [inline]

Give the number of sites.

8.269.4.14 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.

References mln::p_key< K, P >::exists_key().

8.269.4.15 template<typename K , typename P > void mln::p_key< K, P >::remove (const P & *p*) [inline]

Remove a site p.

8.269.4.16 template<typename K , typename P > void mln::p_key< K, P >::remove_key (const K & *k*) [inline]

Remove all sites with key k.

References mln::p_set< P >::nsites().

8.270 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.

8.270.1 Detailed Description

2D discrete line of points.

It is based on p_array.

8.270.2 Member Typedef Documentation

8.270.2.1 typedef p_indexed_bkd_piter<self_> mln::p_line2d::bkd_piter

Backward Site_Iterator associated type.

8.270.2.2 typedef point2d mln::p_line2d::element

Element associated type.

8.270.2.3 typedef p_indexed_fwd_piter<self_> mln::p_line2d::fwd_piter

Forward Site_Iterator associated type.

8.270.2.4 typedef p_indexed_fwd_piter<self_> mln::p_line2d::piter

Site_Iterator associated type.

8.270.2.5 typedef p_indexed_psite<self_> mln::p_line2d::psite

Psite associated type.

8.270.2.6 typedef const box2d& mln::p_line2d::q_box

Box (qualified) associated type.

8.270.3 Constructor & Destructor Documentation

8.270.3.1 mln::p_line2d::p_line2d () [inline]

Constructor without argument.

References is_valid().

8.270.3.2 mln::p_line2d::p_line2d (const point2d & *beg*, const point2d & *end*, bool *is_end_excluded* = false) [inline]

Constructor from point `beg` to point `end`.

References is_valid().

8.270.4 Member Function Documentation

8.270.4.1 const box2d & mln::p_line2d::bbox () const [inline]

Give the exact bounding box.

References is_valid().

8.270.4.2 const point2d & mln::p_line2d::begin () const [inline]

Give the point that begins the line.

References is_valid().

Referenced by mln::debug::draw_graph().

8.270.4.3 const point2d & mln::p_line2d::end () const [inline]

Give the point that ends the line.

References is_valid(), and nsites().

Referenced by mln::debug::draw_graph().

8.270.4.4 bool mln::p_line2d::has (const util::index & *i*) const [inline]

Test if index `i` belongs to this point set.

References nsites().

8.270.4.5 bool mln::p_line2d::has (const psite & *p*) const [inline]

Test if `p` belongs to this point set.

8.270.4.6 bool mln::p_line2d::is_valid () const [inline]

Test if this line is valid, i.e., initialized.

References mln::implies().

Referenced by bbox(), begin(), end(), and p_line2d().

8.270.4.7 std::size_t mln::p_line2d::memory_size () const [inline]

Return the size of this site set in memory.

8.270.4.8 unsigned mln::p_line2d::nsites () const [inline]

Give the number of points.

Referenced by end(), has(), and operator[ ]().

8.270.4.9 const point2d & mln::p_line2d::operator[ ] (unsigned *i*) const [inline]

Return the i-th point of the line.

References nsites().

8.270.4.10 const std::vector< point2d > & mln::p_line2d::std_vector () const [inline]

Return the corresponding std::vector of points.

8.271 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.*

8.271.1 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.

8.271.2 Member Typedef Documentation

8.271.2.1 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.

8.271.2.2 template<typename S > typedef S mln::p_mutable_array_of< S >::element

Element associated type.

8.271.2.3 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.

8.271.2.4 template<typename S > typedef S mln::p_mutable_array_of< S >::i_element

Insertion element associated type.

8.271.2.5 template<typename S > typedef fwd_piter mln::p_mutable_array_of< S >::piter

Site_Iterator associated type.

8.271.2.6 template<typename S > typedef p_double_psite<self_, element> mln::p_mutable_array_of< S >::psite

Psite associated type.

8.271.3 Constructor & Destructor Documentation

8.271.3.1 template<typename S > mln::p_mutable_array_of< S >::p_mutable_array_of () [inline]

Constructor without arguments.

8.271.4 Member Function Documentation

8.271.4.1 template<typename S > void mln::p_mutable_array_of< S >::clear () [inline]

Clear this set.

8.271.4.2 template<typename S > bool mln::p_mutable_array_of< S >::has (const psite & *p*) const [inline]

Test if p belongs to this point set.

8.271.4.3 template<typename S > void mln::p_mutable_array_of< S >::insert (const S & *s*) [inline]

Insert a site set s.

Precondition:

s is valid.

8.271.4.4 template<typename S > bool mln::p_mutable_array_of< S >::is_valid () const [inline]

Test this set validity so returns always true.

8.271.4.5 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.

8.271.4.6 template<typename S > unsigned mln::p_mutable_array_of< S >::nelements () const [inline]

Give the number of elements (site sets) of this composite.

8.271.4.7 template<typename S > S & mln::p_mutable_array_of< S >::operator[ ] (unsigned *i*) [inline]

Return the i-th site set (mutable version).

8.271.4.8 template<typename S > const S & mln::p_mutable_array_of< S >::operator[ ] (unsigned *i*) const [inline]

Return the i-th site set (const version).

8.271.4.9 template<typename S > void mln::p_mutable_array_of< S >::reserve (unsigned *n*) [inline]

Reserve memory for n elements.

8.272 mln::p_n_faces_bkd_piter< D, P > Class Template Reference

Backward iterator on the n-faces sites of an mln::p_complex<D, P>.

`#include <p_n_faces_piter.hh>`

Inherits p_complex_piter_base_< topo::n_face_bkd_iter< D >, p_complex< D, P >, P, p_n_faces_bkd_-piter< D, P > >.

Public Member Functions

- void next ()
  Go to the next element.
- unsigned n () const
  Accessors.
- p_n_faces_bkd_piter ()
  Construction and assignment.

8.272.1 Detailed Description

template<unsigned D, typename P> class mln::p_n_faces_bkd_piter< D, P >

Backward iterator on the n-faces sites of an mln::p_complex<D, P>.

8.272.2 Constructor & Destructor Documentation

8.272.2.1 template<unsigned D, typename P > mln::p_n_faces_bkd_piter< D, P >::p_n_faces_bkd_piter () [inline]

Construction and assignment.

8.272.3 Member Function Documentation

8.272.3.1 template<unsigned D, typename P > unsigned mln::p_n_faces_bkd_piter< D, P >::n () const [inline]

Accessors.

Shortcuts to face_'s accessors.

8.272.3.2 void mln::Site_Iterator< p_n_faces_bkd_piter< D, 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.

8.273 mln::p_n_faces_fwd_piter< D, P > Class Template Reference

Forward iterator on the n-faces sites of an mln::p_complex<D, P>.

`#include <p_n_faces_piter.hh>`

Inherits p_complex_piter_base_< topo::n_face_fwd_iter< D >, p_complex< D, P >, P, p_n_faces_fwd_-piter< D, P > >.

Public Member Functions

- void next ()
  Go to the next element.
- unsigned n () const
  Accessors.
- p_n_faces_fwd_piter ()
  Construction and assignment.

8.273.1 Detailed Description

template<unsigned D, typename P> class mln::p_n_faces_fwd_piter< D, P >

Forward iterator on the n-faces sites of an mln::p_complex<D, P>.

8.273.2 Constructor & Destructor Documentation

8.273.2.1 template<unsigned D, typename P > mln::p_n_faces_fwd_piter< D, P >::p_n_faces_fwd_piter () [inline]

Construction and assignment.

8.273.3 Member Function Documentation

8.273.3.1 template<unsigned D, typename P > unsigned mln::p_n_faces_fwd_piter< D, P >::n () const [inline]

Accessors.

Shortcuts to face_'s accessors.

8.273.3.2 void mln::Site_Iterator< p_n_faces_fwd_piter< D, 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.

8.274 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 p_priority< P, Q > &other)

  Insert elements from another priority queue.

- void insert (const i_element &p_e)
  Insert a pair `p_e` *(priority p, element e).*
- 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`.

8.274.1 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>).

8.274.2 Member Typedef Documentation

8.274.2.1 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.

8.274.2.2 template<typename P, typename Q> typedef Q ::element mln::p_priority< P, Q >::element

Element associated type.

8.274.2.3 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.

8.274.2.4 template<typename P, typename Q> typedef std::pair<P, element> mln::p_priority< P, Q >::i_element

Insertion element associated type.

8.274.2.5 template<typename P, typename Q> typedef fwd_piter mln::p_priority< P, Q >::piter

Site_Iterator associated type.

8.274.2.6 template<typename P, typename Q> typedef p_double_psite<self_, Q> mln::p_priority< P, Q >::psite

Psite associated type.

8.274.3 Constructor & Destructor Documentation

8.274.3.1 template<typename P , typename Q > mln::p_priority< P, Q >::p_priority () [inline]

Constructor.

8.274.4 Member Function Documentation

8.274.4.1 template<typename P , typename Q > void mln::p_priority< P, Q >::clear () [inline]

Clear the queue.

8.274.4.2 template<typename P , typename Q > bool mln::p_priority< P, Q >::exists_priority (const P & *priority*) const [inline]

Test if the `priority` exists.

Referenced by mln::p_priority< P, Q >::operator()().

8.274.4.3 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()

References mln::p_priority< P, Q >::highest_priority().

Referenced by mln::morpho::meyer_wst().

8.274.4.4 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.

8.274.4.5 template<typename P , typename Q > const P mln::p_priority< P, Q >::highest_priority () const [inline]

Give the highest priority.

Precondition:

! is_empty()

Referenced by mln::p_priority< P, Q >::front(), and mln::p_priority< P, Q >::pop().

8.274.4.6 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.

8.274.4.7 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).

References mln::p_priority< P, Q >::push().

8.274.4.8 template<typename P , typename Q > bool mln::p_priority< P, Q >::is_valid () const [inline]

Test this set validity so returns always true.

8.274.4.9 template<typename P , typename Q > const P mln::p_priority< P, Q >::lowest_priority () const [inline]

Give the lowest priority.

Precondition:

! is_empty()

8.274.4.10 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.

8.274.4.11 template<typename P , typename Q > unsigned mln::p_priority< P, Q >::nsites () const [inline]

Give the number of sites.

Referenced by mln::p_priority< P, Q >::operator()().

8.274.4.12 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.

References mln::p_priority< P, Q >::exists_priority(), and mln::p_priority< P, Q >::nsites().

8.274.4.13 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()

References mln::p_priority< P, Q >::highest_priority().

Referenced by mln::morpho::meyer_wst().

8.274.4.14 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()

Referenced by mln::geom::impl::seeds2tiling_roundness().

8.274.4.15 template<typename P , typename Q > const util::set< P > & mln::p_priority< P, Q >::priorities () const [inline]

Give the set of priorities.

8.274.4.16 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`.

Referenced by mln::p_priority< P, Q >::insert(), mln::morpho::meyer_wst(), and mln::geom::impl::seeds2tiling_roundness().

8.275 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 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 ()

  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.

8.275.1 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.

8.275.2 Member Typedef Documentation

8.275.2.1 template<typename P> typedef p_indexed_bkd_piter<self_> mln::p_queue< P >::bkd_piter

Backward Site_Iterator associated type.

8.275.2.2 template<typename P> typedef P mln::p_queue< P >::element

Element associated type.

8.275.2.3 template<typename P> typedef p_indexed_fwd_piter<self_> mln::p_queue< P >::fwd_piter

Forward Site_Iterator associated type.

8.275.2.4 template<typename P> typedef P mln::p_queue< P >::i_element

Insertion element associated type.

8.275.2.5 template<typename P> typedef fwd_piter mln::p_queue< P >::piter

Site_Iterator associated type.

8.275.2.6 template<typename P> typedef p_indexed_psite<self_> mln::p_queue< P >::psite

Psite associated type.

8.275.3 Constructor & Destructor Documentation

8.275.3.1 template<typename P > mln::p_queue< P >::p_queue () [inline]

Constructor without argument.

8.275.4 Member Function Documentation

8.275.4.1 template<typename P > void mln::p_queue< P >::clear () [inline]

Clear the queue.

8.275.4.2 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.

Referenced by mln::p_queue< P >::pop_front().

8.275.4.3 template<typename P > bool mln::p_queue< P >::has (const util::index & *i*) const [inline]

Test if index `i` belongs to this site set.

References mln::p_queue< P >::nsites().

8.275.4.4 template<typename P > bool mln::p_queue< P >::has (const psite & *p*) const [inline]

Test if `p` belongs to this site set.

References mln::p_queue< P >::nsites().

8.275.4.5 template<typename P > void mln::p_queue< P >::insert (const P & *p*) [inline]

Insert a site `p` (equivalent as 'push').

References mln::p_queue< P >::push().

8.275.4.6 template<typename P > bool mln::p_queue< P >::is_valid () const [inline]

This set is always valid so it returns true.

8.275.4.7 template<typename P > std::size_t mln::p_queue< P >::memory_size () const [inline]

Return the size of this site set in memory.

References mln::p_queue< P >::nsites().

8.275.4.8 template<typename P > unsigned mln::p_queue< P >::nsites () const [inline]

Give the number of sites.

Referenced by mln::p_queue< P >::has(), mln::p_queue< P >::memory_size(), and mln::p_queue< P >::operator[ ]().

8.275.4.9 template<typename P > const P & mln::p_queue< P >::operator[ ] (unsigned *i*) const [inline]

Return the i-th site.

References mln::p_queue< P >::nsites().

8.275.4.10 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.

Referenced by mln::p_queue< P >::pop_front().

8.275.4.11 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.

References mln::p_queue< P >::front(), and mln::p_queue< P >::pop().

8.275.4.12 template<typename P > void mln::p_queue< P >::push (const P & *p*) [inline]

Push a site p in the queue.

Referenced by mln::p_queue< P >::insert().

8.275.4.13 template<typename P > const std::deque< P > & mln::p_queue< P >::std_deque () const [inline]

Return the corresponding std::deque of sites.

8.276 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.*
- 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.

8.276.1 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.

8.276.2 Member Typedef Documentation

8.276.2.1 template<typename P > typedef p_indexed_bkd_piter<self_> mln::p_queue_fast< P >::bkd_piter

Backward Site_Iterator associated type.

8.276.2.2 template<typename P > typedef P mln::p_queue_fast< P >::element

Element associated type.

8.276.2.3 template<typename P > typedef p_indexed_fwd_piter<self_> mln::p_queue_fast< P >::fwd_piter

Forward Site_Iterator associated type.

8.276.2.4 template<typename P > typedef P mln::p_queue_fast< P >::i_element

Insertion element associated type.

8.276.2.5 template<typename P > typedef fwd_piter mln::p_queue_fast< P >::piter

Site_Iterator associated type.

8.276.2.6 template<typename P > typedef p_indexed_psite<self_> mln::p_queue_fast< P >::psite

Psite associated type.

8.276.3 Constructor & Destructor Documentation

8.276.3.1 template<typename P > mln::p_queue_fast< P >::p_queue_fast () [inline]

Constructor without argument.

8.276.4 Member Function Documentation

8.276.4.1 template<typename P > void mln::p_queue_fast< P >::clear () [inline]

Clear the queue.

8.276.4.2 template<typename P > bool mln::p_queue_fast< P >::compute_has (const P & *p*) const [inline]

Test if p belongs to this site set.

8.276.4.3 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.

Referenced by mln::p_queue_fast< P >::pop_front().

8.276.4.4 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.

References mln::p_queue_fast< P >::nsites().

8.276.4.5 template<typename P > bool mln::p_queue_fast< P >::has (const psite & *p*) const [inline]

Test if `p` belongs to this site set.

References mln::p_queue_fast< P >::nsites().

8.276.4.6 template<typename P > void mln::p_queue_fast< P >::insert (const P & *p*) [inline]

Insert a site `p` (equivalent as 'push').

References mln::p_queue_fast< P >::push().

8.276.4.7 template<typename P > bool mln::p_queue_fast< P >::is_valid () const [inline]

This set is always valid so it returns true.

8.276.4.8 template<typename P > std::size_t mln::p_queue_fast< P >::memory_size () const [inline]

Return the size of this site set in memory.

8.276.4.9 template<typename P > unsigned mln::p_queue_fast< P >::nsites () const [inline]

Give the number of sites.

Referenced by mln::p_queue_fast< P >::has(), and mln::p_queue_fast< P >::operator[ ]().

8.276.4.10 template<typename P > const P & mln::p_queue_fast< P >::operator[ ] (unsigned *i*) const [inline]

Return the `i-th` site.

References mln::p_queue_fast< P >::nsites().

8.276.4.11 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.

Referenced by mln::p_queue_fast< P >::pop_front().

8.276.4.12 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.

References mln::p_queue_fast< P >::front(), and mln::p_queue_fast< P >::pop().

8.276.4.13 template<typename P > void mln::p_queue_fast< P >::purge () [inline]

Purge the queue to save (free) some memory.

8.276.4.14 template<typename P > void mln::p_queue_fast< P >::push (const P & *p*) [inline]

Push a site p in the queue.

Referenced by mln::p_queue_fast< P >::insert().

8.276.4.15 template<typename P > void mln::p_queue_fast< P >::reserve (typename p_array< P >::size_type *n*) [inline]

Reserve n cells.

8.276.4.16 template<typename P > const std::vector< P > & mln::p_queue_fast< P >::std_vector () const [inline]

Return the corresponding std::vector of sites.

8.277 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 (const P &start, const P &end)

  Constructor.

- p_run (const P &start, unsigned short len)

  Constructor.

- p_run ()

  Constructor without argument.

- const P & start () const

  Return the starting point.

8.277.1 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.

8.277.2 Member Typedef Documentation

8.277.2.1 template<typename P> typedef p_run_bkd_piter_<P> mln::p_run< P >::bkd_piter

Backward Site_Iterator associated type.

8.277.2.2 template<typename P> typedef P mln::p_run< P >::element

Element associated type.

8.277.2.3 template<typename P> typedef p_run_fwd_piter_<P> mln::p_run< P >::fwd_piter

Forward Site_Iterator associated type.

8.277.2.4 template<typename P> typedef fwd_piter mln::p_run< P >::piter

Site_Iterator associated type.

8.277.2.5 template<typename P> typedef p_run_psite<P> mln::p_run< P >::psite

Psite associated type.

8.277.2.6 template<typename P> typedef mln::box<P> mln::p_run< P >::q_box

Box associated type.

8.277.3 Constructor & Destructor Documentation

8.277.3.1 template<typename P > mln::p_run< P >::p_run () [inline]

Constructor without argument.

8.277.3.2 template<typename P > mln::p_run< P >::p_run (const P & *start*, unsigned short *len*) [inline]

Constructor.

References mln::p_run< P >::init().

8.277.3.3 template<typename P > mln::p_run< P >::p_run (const P & *start*, const P & *end*) [inline]

Constructor.

8.277.4 Member Function Documentation

8.277.4.1 template<typename P > mln::box< P > mln::p_run< P >::bbox () const [inline]

Give the exact bounding box.

References mln::p_run< P >::end().

8.277.4.2 template<typename P > P mln::p_run< P >::end () const [inline]

Return (compute) the ending point.

References mln::point< G, C >::last_coord().

Referenced by mln::p_run< P >::bbox().

8.277.4.3 template<typename P > bool mln::p_run< P >::has (const P & *p*) const [inline]

Test if p belongs to this point set.

References mln::p_run< P >::is_valid().

8.277.4.4 template<typename P > bool mln::p_run< P >::has (const psite & *p*) const [inline]

Test if p belongs to this point set.

8.277.4.5 template<typename P > bool mln::p_run< P >::has_index (unsigned short *i*) const [inline]

Test if index i belongs to this point set.

8.277.4.6 template<typename P > void mln::p_run< P >::init (const P & *start*, unsigned short *len*) [inline]

Set the starting point.

Referenced by mln::p_run< P >::p_run().

8.277.4.7 template<typename P > bool mln::p_run< P >::is_valid () const [inline]

Test if this run is valid, i.e., with length > 0.

Referenced by mln::p_run< P >::has(), mln::p_run< P >::length(), mln::p_run< P >::nsites(), and mln::p_run< P >::operator[ ]().

8.277.4.8 template<typename P > unsigned short mln::p_run< P >::length () const [inline]

Give the length of the run.

References mln::p_run< P >::is_valid().

8.277.4.9 template<typename P > std::size_t mln::p_run< P >::memory_size () const [inline]

Return the size of this site set in memory.

8.277.4.10 template<typename P > unsigned mln::p_run< P >::nsites () const [inline]

Give the number of sites.

References mln::p_run< P >::is_valid().

8.277.4.11 template<typename P > P mln::p_run< P >::operator[ ] (unsigned short *i*) const [inline]

Return the i-th point.

References mln::p_run< P >::is_valid(), and mln::point< G, C >::last_coord().

8.277.4.12 template<typename P > const P & mln::p_run< P >::start () const [inline]

Return the starting point.

8.278 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 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.
- bool has (const psite &p) const
  Test if psite `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.

8.278.1 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.

8.278.2 Member Typedef Documentation

8.278.2.1 template<typename P> typedef p_indexed_bkd_piter<self_> mln::p_set< P >::bkd_piter

Backward Site_Iterator associated type.

8.278.2.2 template<typename P> typedef P mln::p_set< P >::element

Element associated type.

8.278.2.3 template<typename P> typedef p_indexed_fwd_piter<self_> mln::p_set< P >::fwd_piter

Forward Site_Iterator associated type.

8.278.2.4 template<typename P> typedef P mln::p_set< P >::i_element

Insertion element associated type.

8.278.2.5 template<typename P> typedef fwd_piter mln::p_set< P >::piter

Site_Iterator associated type.

8.278.2.6 template<typename P> typedef p_indexed_psite<self_> mln::p_set< P >::psite

Psite associated type.

8.278.2.7 template<typename P> typedef P mln::p_set< P >::r_element

Removal element associated type.

8.278.3 Constructor & Destructor Documentation

8.278.3.1 template<typename P > mln::p_set< P >::p_set () [inline]

Constructor.

8.278.4 Member Function Documentation

8.278.4.1 template<typename P > void mln::p_set< P >::clear () [inline]

Clear this set.

8.278.4.2 template<typename P > bool mln::p_set< P >::has (const util::index & *i*) const [inline]

Test if index `i` belongs to this point set.

References mln::p_set< P >::nsites().

8.278.4.3 template<typename P > bool mln::p_set< P >::has (const P & *p*) const [inline]

Test if `p` belongs to this point set.

8.278.4.4 template<typename P > bool mln::p_set< P >::has (const psite & *p*) const [inline]

Test if psite `p` belongs to this point set.

8.278.4.5 template<typename P > void mln::p_set< P >::insert (const P & *p*) [inline]

Insert a site `p`.

Referenced by mln::convert::to_p_set().

8.278.4.6 template<typename P > bool mln::p_set< P >::is_valid () const [inline]

Test this set validity so returns always true.

8.278.4.7 template<typename P > std::size_t mln::p_set< P >::memory_size () const [inline]

Return the size of this site set in memory.

8.278.4.8 template<typename P > unsigned mln::p_set< P >::nsites () const [inline]

Give the number of sites.

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().

8.278.4.9 template<typename P > const P & mln::p_set< P >::operator[ ] (unsigned *i*) const [inline]

Return the `i-th` site.

References mln::p_set< P >::nsites().

8.278.4.10 template<typename P > void mln::p_set< P >::remove (const P & *p*) [inline]

Remove a site `p`.

8.278.4.11 template<typename P > const std::vector< P > & mln::p_set< P >::std_vector () const [inline]

Return the corresponding std::vector of sites.

8.278.4.12 template<typename P > const util::set< P > & mln::p_set< P >::util_set () const [inline]

Return the corresponding util::set of sites.

8.279 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.

8.279.1 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.

8.279.2 Member Typedef Documentation

8.279.2.1 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.

8.279.2.2 template<typename S, typename F> typedef S ::element mln::p_transformed< S, F >::element

Element associated type.

8.279.2.3 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.

8.279.2.4 template<typename S, typename F> typedef fwd_piter mln::p_transformed< S, F >::piter

Site_Iterator associated type.

8.279.2.5 template<typename S, typename F> typedef S ::psite mln::p_transformed< S, F >::psite

Psite associated type.

8.279.3 Constructor & Destructor Documentation

8.279.3.1 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`.

8.279.3.2 template<typename S , typename F > mln::p_transformed< S, F >::p_transformed () [inline]

Constructor without argument.

8.279.4 Member Function Documentation

8.279.4.1 template<typename S , typename F > const F & mln::p_transformed< S, F >::function () const [inline]

Return the transformation function.

8.279.4.2 template<typename S , typename F > bool mln::p_transformed< S, F >::has (const psite & *p*) const [inline]

Test if p belongs to the subset.

8.279.4.3 template<typename S , typename F > bool mln::p_transformed< S, F >::is_valid () const [inline]

Test if this site set is valid.

8.279.4.4 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.

8.279.4.5 template<typename S , typename F > const S & mln::p_transformed< S, F >::primary_set () const [inline]

Return the primary set.

Referenced by mln::p_transformed_piter< Pi, S, F >::change_target().

8.280 mln::p_transformed_piter< Pi, S, F > Struct Template Reference

Iterator on p_transformed<S,F>.

`#include <p_transformed_piter.hh>`

Inherits 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.

- p_transformed_piter (const p_transformed< S, F > &s)

  Constructor from a site set.

- p_transformed_piter ()

  Constructor without argument.

8.280.1 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

8.280.2 Constructor & Destructor Documentation

8.280.2.1 template<typename Pi , typename S , typename F > mln::p_transformed_piter< Pi, S, F >::p_transformed_piter () `[inline]`

Constructor without argument.

8.280.2.2 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.

References mln::p_transformed_piter< Pi, S, F >::change_target().

8.280.3 Member Function Documentation

8.280.3.1 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.

References mln::p_transformed< S, F >::primary_set().

Referenced by mln::p_transformed_piter< Pi, S, F >::p_transformed_piter().

8.281 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 V &v, const typename S::psite &p) const
  Test if the couple (value v, psite p) belongs to this site set.
- bool has (const psite &p) const
  Test if p belongs to this site set.
- void insert (const V &v, const element &e)
  Insert e at value v.
- void insert (const i_element &v_e)

Insert a pair v_e *(value v, element e).*

- 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.

8.281.1 Detailed Description

template<typename V, typename S> class mln::p_vaccess< V, S >

Site set in which sites are grouped by their associated value.

8.281.2 Member Typedef Documentation

8.281.2.1 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.

8.281.2.2 template<typename V , typename S > typedef S ::element mln::p_vaccess< V, S >::element

Element associated type.

8.281.2.3 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.

8.281.2.4 template<typename V , typename S > typedef std::pair<V, element> mln::p_vaccess< V, S >::i_element

Insertion element associated type.

8.281.2.5 template<typename V , typename S > typedef fwd_piter mln::p_vaccess< V, S >::piter

Site_Iterator associated type.

8.281.2.6 template<typename V , typename S > typedef S mln::p_vaccess< V, S >::pset

Inner site set associated type.

8.281.2.7 template<typename V , typename S > typedef p_double_psite<self_, S> mln::p_vaccess< V, S >::psite

Psite associated type.

8.281.2.8 template<typename V , typename S > typedef V mln::p_vaccess< V, S >::value

Value associated type.

8.281.2.9 template<typename V , typename S > typedef mln::value::set<V> mln::p_vaccess< V, S >::vset

Value_Set associated type.

8.281.3 Constructor & Destructor Documentation

8.281.3.1 template<typename V , typename S > mln::p_vaccess< V, S >::p_vaccess () [inline]

Constructor.

8.281.4 Member Function Documentation

8.281.4.1 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.

8.281.4.2 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.

8.281.4.3 template<typename V , typename S > void mln::p_vaccess< V, S >::insert (const V & *v*, const element & *e*) [inline]

Insert e at value v.

8.281.4.4 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).

8.281.4.5 template<typename V , typename S > bool mln::p_vaccess< V, S >::is_valid () const [inline]

Test if this site set is valid.

8.281.4.6 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.

8.281.4.7 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`.

8.281.4.8 template<typename V , typename S > const mln::value::set< V > & mln::p_vaccess< V, S >::values () const [inline]

Give the set of values.

8.282 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 p_graph_piter< self_, mln_vertex_bkd_iter(G) > bkd_piter
  Backward Site_Iterator associated type.
- typedef super_::site element
  Associated types.
- typedef p_graph_piter< self_, mln_vertex_fwd_iter(G) > fwd_piter
  Forward Site_Iterator associated type.
- typedef fwd_piter piter
  Site_Iterator associated type.
- typedef p_vertices_psite< G, F > psite
  Point_Site associated type.

Public Member Functions

- template<typename G2 >
  bool has (const util::vertex< G2 > &v) const
  Does this site set has v*?*
- bool has (const psite &p) const
  Does this site set has p*?*
- 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.

- template<typename F2 >
  p_vertices (const p_vertices< G, F2 > &other)

  Copy constructor.

- template<typename F2 >
  p_vertices (const Graph< G > &gr, const Function< F2 > &f)

  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.

- p_vertices (const Graph< G > &gr)

  Construct a graph psite set from a graph of points.

- p_vertices ()

  Constructor without argument.

  - const F & function () const

    Return the association function.

  - const G & graph () const

    Accessors.

  - F::result operator() (const psite &p) const

    Return the value associated to an element of this site set.

8.282.1 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.

8.282.2 Member Typedef Documentation

8.282.2.1 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.

8.282.2.2 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.

8.282.2.3 template<typename G, typename F = util::internal::id2element<G,util::vertex<G> >> typedef F mln::p_vertices< G, F >::fun_t

Function associated type.

8.282.2.4 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.

8.282.2.5 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.

8.282.2.6 template<typename G, typename F = util::internal::id2element<G,util::vertex<G> >> typedef G mln::p_vertices< G, F >::graph_t

Graph associated type.

8.282.2.7 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.

8.282.2.8 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.

8.282.2.9 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.

8.282.3 Constructor & Destructor Documentation

8.282.3.1 template<typename G , typename F > mln::p_vertices< G, F >::p_vertices () [inline]

Constructor without argument.

8.282.3.2 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.

References mln::p_vertices< G, F >::is_valid().

8.282.3.3 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.

References mln::p_vertices< G, F >::is_valid().

8.282.3.4 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`.

References mln::p_vertices< G, F >::is_valid().

8.282.3.5 template<typename G , typename F > template<typename F2 > mln::p_vertices< G, F >::p_vertices (const p_vertices< G, F2 > & *other*) [inline]

Copy constructor.

References mln::p_vertices< G, F >::function(), mln::p_vertices< G, F >::graph(), and mln::p_vertices< G, F >::is_valid().

8.282.4 Member Function Documentation

8.282.4.1 template<typename G , typename F > const F & mln::p_vertices< G, F >::function () const [inline]

Return the association function.

Referenced by mln::p_vertices< G, F >::p_vertices().

8.282.4.2 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)

References mln::p_vertices< G, F >::is_valid().

Referenced by mln::debug::draw_graph(), mln::operator==(), and mln::p_vertices< G, F >::p_vertices().

8.282.4.3 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*?

References mln::util::vertex< G >::graph(), mln::util::vertex< G >::is_valid(), and mln::p_vertices< G, F >::is_valid().

8.282.4.4 template<typename G , typename F > bool mln::p_vertices< G, F >::has (const psite & *p*) const [inline]

Does this site set has *p*?

References mln::p_vertices< G, F >::is_valid().

8.282.4.5 template<typename G , typename F > void mln::p_vertices< G, F >::invalidate () [inline]

Invalidate this site set.

8.282.4.6 template<typename G , typename F > bool mln::p_vertices< G, F >::is_valid () const [inline]

Test this site set validity.

Referenced by mln::p_vertices< G, F >::graph(), mln::p_vertices< G, F >::has(), and mln::p_vertices< G, F >::p_vertices().

8.282.4.7 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;.

8.282.4.8 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.

References mln::p_vertices< G, F >::nvertices().

8.282.4.9 template<typename G , typename F > unsigned mln::p_vertices< G, F >::nvertices () const [inline]

Return The number of vertices in the graph.

Referenced by mln::p_vertices< G, F >::nsites().

8.282.4.10 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.

8.283 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, const typename I::psite &p)
  Constructor.
- pixel (I &image)
  Constructor.

8.283.1 Detailed Description

template<typename I> struct mln::pixel< I >

Generic pixel class.

The parameter is `I` the type of the image it belongs to.

8.283.2 Constructor & Destructor Documentation

8.283.2.1 template<typename I > mln::pixel< I >::pixel (I & *image*) [inline]

Constructor.

8.283.2.2 template<typename I > mln::pixel< I >::pixel (I & *image*, const typename I::psite & *p*) [inline]

Constructor.

References mln::pixel< I >::change_to().

8.283.3 Member Function Documentation

8.283.3.1 template<typename I > void mln::pixel< I >::change_to (const typename I::psite & *p*) [inline]

Change the pixel to the one at point p.

Referenced by mln::pixel< I >::pixel().

8.283.3.2 template<typename I > bool mln::pixel< I >::is_valid () const [inline]

Test if this pixel is valid.

8.284 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 I &ima)

  Copy constructor from an image `ima`.

- plain (const plain< I > &rhs)

  Copy constructor.

- plain ()

  Constructor without argument.

8.284.1 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.

8.284.2 Member Typedef Documentation

8.284.2.1 template<typename I> typedef plain< tag::image_<I> > mln::plain< I >::skeleton

Skeleton.

8.284.3 Constructor & Destructor Documentation

8.284.3.1 template<typename I > mln::plain< I >::plain () [inline]

Constructor without argument.

8.284.3.2 template<typename I > mln::plain< I >::plain (const plain< I > & *rhs*) [inline]

Copy constructor.

8.284.3.3 template<typename I > mln::plain< I >::plain (const I & *ima*) [inline]

Copy constructor from an image `ima`.

8.284.4 Member Function Documentation

8.284.4.1 template<typename I > mln::plain< I >::operator I () const [inline]

Conversion into an image with type `I`.

References mln::duplicate().

8.284.4.2 template<typename I > plain< I > & mln::plain< I >::operator= (const I & *ima*) [inline]

Assignment operator from an image `ima`.

8.284.4.3 template<typename I > plain< I > & mln::plain< I >::operator= (const plain< I > & *rhs*) [inline]

Assignment operator.

8.285 mln::Point< P > Struct Template Reference

Base class for implementation of point classes.

`#include <point.hh>`

Inherits Point_Site< 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)

  Divise a point `by` *a scalar* `s`.

8.285.1 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.

8.285.2 Member Typedef Documentation

8.285.2.1 template<typename P > typedef P mln::Point< P >::point

The associated point type is itself.

8.285.3 Member Function Documentation

8.285.3.1 template<typename P > const P & mln::Point< P >::to_point () const [inline]

It is a Point so it returns itself.

8.285.4 Friends And Related Function Documentation

8.285.4.1 template<typename P , typename D > P & operator+= (Point< P > & *p*, const Dpoint< D > & *dp*) [related]

Shift a point by a delta-point dp.

Parameters:

↔ ***p*** The targeted point.

← ***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.

8.285.4.2 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:

↔ ***p*** The targeted point.

← ***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.

8.285.4.3 template<typename P , typename D > P & operator/ (Point< P > & *p*, const value::Scalar< D > & *dp*) [related]

Divise a point by a scalar s.

Parameters:

↔ ***p*** The targeted point.

← ***dp*** A scalar.

Returns:

A reference to the point p once divised by s.

8.286 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 algebra::h_vec< G::dim, float > h_vec
  Algebra hexagonal vector (hvec) associated type.
- typedef algebra::vec< G::dim, float > vec
  Algebra vector (vec) associated type.

Public Member Functions

- C & last_coord ()
  Read-write access to the last coordinate.
- const C & last_coord () const
  Read-only 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`.
- template<typename C2 >
  point (const algebra::vec< dim, C2 > &v)
  Constructor from an algebra vector.
- point ()
  Constructor without argument.
- void set_all (C c)
  Set all coordinates to the value `c`.
- h_vec to_h_vec () const
  Transform to point in homogene coordinate system.
- vec to_vec () const
  Explicit conversion towards mln::algebra::vec.

- point (const literal::origin_t &)

Constructors/assignments with literals.

- point (C ind)

Static Public Member Functions

- static const point< G, C > & minus_infty ()

  Point with all coordinates set to the mininum 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).

8.286.1 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.

8.286.2 Member Typedef Documentation

8.286.2.1 template<typename G, typename C> typedef C mln::point< G, C >::coord

Coordinate associated type.

8.286.2.2 template<typename G, typename C> typedef dpoint<G,C> mln::point< G, C >::delta

Delta associated type.

8.286.2.3 template<typename G, typename C> typedef dpoint<G,C> mln::point< G, C >::dpsite

DPsite associated type.

8.286.2.4 template<typename G, typename C> typedef G mln::point< G, C >::grid

Grid associated type.

8.286.2.5 template<typename G, typename C> typedef algebra::h_vec<G::dim, float> mln::point< G, C >::h_vec

Algebra hexagonal vector (hvec) associated type.

8.286.2.6 template<typename G, typename C> typedef algebra::vec<G::dim, float> mln::point< G, C >::vec

Algebra vector (vec) associated type.

8.286.3 Member Enumeration Documentation

8.286.3.1 template<typename G, typename C> anonymous enum

Enumerator:

dim Dimension of the space.

Invariant:

dim > 0

8.286.4 Constructor & Destructor Documentation

8.286.4.1 template<typename G , typename C > mln::point< G, C >::point () [inline]

Constructor without argument.

8.286.4.2 template<typename G , typename C > template<typename C2 > mln::point< G, C >::point (const algebra::vec< dim, C2 > & *v*) [inline]

Constructor from an algebra vector.

References mln::point< G, C >::dim.

8.286.4.3 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.

8.286.4.4 template<typename G , typename C> mln::point< G, C >::point (const literal::origin_t &) [inline]

Constructors/assignments with literals.

8.286.4.5 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.

References mln::point< G, C >::dim.

8.286.5 Member Function Documentation

8.286.5.1 template<typename G , typename C > C & mln::point< G, C >::last_coord () [inline]

Read-write access to the last coordinate.

References mln::point< G, C >::dim.

8.286.5.2 template<typename G , typename C > const C & mln::point< G, C >::last_coord () const [inline]

Read-only access to the last coordinate.

References mln::point< G, C >::dim.

Referenced by mln::p_run< P >::end(), mln::p_run< P >::operator[ ](), and mln::debug::put_word().

8.286.5.3 template<typename G , typename C > const point< G, C > & mln::point< G, C >::minus_infty () [inline, static]

Point with all coordinates set to the mininum value.

8.286.5.4 template<typename G , typename C > point< G, C > & mln::point< G, C >::operator+= (const delta & *dp*) [inline]

Shifting by dp.

References mln::point< G, C >::dim.

8.286.5.5 template<typename G , typename C > point< G, C > & mln::point< G, C >::operator-= (const delta & *dp*) [inline]

Shifting by the inverse of dp.

References mln::point< G, C >::dim.

8.286.5.6 template<typename G , typename C > C & mln::point< G, C >::operator[ ] (unsigned *i*) [inline]

Read-write access to the i-th coordinate value.

Parameters:

← ***i*** The coordinate index.

Precondition:

i < dim

References mln::point< G, C >::dim.

8.286.5.7 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:

← ***i*** The coordinate index.

Precondition:

`i < dim`

References mln::point< G, C >::dim.

8.286.5.8 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.

8.286.5.9 template<typename G , typename C> void mln::point< G, C >::set_all (C *c*) [inline]

Set all coordinates to the value `c`.

8.286.5.10 template<typename G , typename C > point< G, C >::h_vec mln::point< G, C >::to_h_vec () const [inline]

Transform to point in homogene coordinate system.

References mln::point< G, C >::dim.

8.286.5.11 template<typename G , typename C > point< G, C >::vec mln::point< G, C >::to_vec () const [inline]

Explicit conversion towards mln::algebra::vec.

References mln::point< G, C >::dim.

Referenced by mln::io::magick::load(), mln::io::dicom::load(), and mln::io::magick::save().

8.286.6 Member Data Documentation

8.286.6.1 template<typename G, typename C> const point< G, C > mln::point< G, C >::origin = all_to(0) [inline, static]

Origin point (all coordinates are 0).

8.287 mln::Proxy< E > Struct Template Reference

Base class for implementation classes of the notion of "proxy".

`#include <proxy.hh>`

Inherits Object< E >.

8.287.1 Detailed Description

template<typename E> struct mln::Proxy< E >

Base class for implementation classes of the notion of "proxy".

8.288 mln::Proxy< void > Struct Template Reference

Proxy category flag type.

`#include <proxy.hh>`

8.288.1 Detailed Description

template<> struct mln::Proxy< void >

Proxy category flag type.

8.289 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 >:

8.289.1 Detailed Description

template<typename E> struct mln::Pseudo_Site< E >

Base class for implementation classes of the notion of "pseudo site".

FIXME: Explain...

8.290 mln::Pseudo_Site< void > Struct Template Reference

Pseudo_Site category flag type.

`#include <pseudo_site.hh>`

8.290.1 Detailed Description

template<> struct mln::Pseudo_Site< void >

Pseudo_Site category flag type.

8.291 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 (const Function_v2v< F > &f, const Site_Set< S > &ps)
  Constructor.
- image ()
  Constructor without argument.

8.291.1 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.

8.291.2 Member Typedef Documentation

8.291.2.1 template<typename F, typename S> typedef image< tag::function_<F>, tag::domain_<S> > mln::pw::image< F, S >::skeleton

Skeleton.

8.291.3 Constructor & Destructor Documentation

8.291.3.1 template<typename F , typename S > mln::pw::image< F, S >::image () [inline]

Constructor without argument.

8.291.3.2 template<typename F , typename S > mln::pw::image< F, S >::image (const Function_v2v< F > & *f*, const Site_Set< S > & *ps*) [inline]

Constructor.

8.292 mln::registration::closest_point_basic< P > Class Template Reference

Closest point functor based on map distance.

`#include <icp.hh>`

8.292.1 Detailed Description

template<typename P> class mln::registration::closest_point_basic< P >

Closest point functor based on map distance.

8.293 mln::registration::closest_point_with_map< P > Class Template Reference

Closest point functor based on map distance.

`#include <icp.hh>`

8.293.1 Detailed Description

template<typename P> class mln::registration::closest_point_with_map< P >

Closest point functor based on map distance.

8.294 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 >:

8.294.1 Detailed Description

template<typename E> struct mln::Regular_Grid< E >

Base class for implementation classes of regular grids.

8.295 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.

8.295.1 Detailed Description

template<typename I> class mln::safe_image< I >

Makes an image accessible at undefined location.

8.295.2 Member Typedef Documentation

8.295.2.1 template<typename I> typedef safe_image< tag::image_<I> > mln::safe_image< I >::skeleton

Skeleton.

8.295.3 Member Function Documentation

8.295.3.1 template<typename I > mln::safe_image< I >::operator safe_image< const I > () const [inline]

Const promotion via conversion.

8.296 mln::select::p_of< P > Struct Template Reference

Structure p_of.

`#include <pix.hh>`

8.296.1 Detailed Description

template<typename P> struct mln::select::p_of< P >

Structure p_of.

8.297 mln::Site< E > Struct Template Reference

Base class for classes that are explicitly sites.

`#include <site.hh>`

Inheritance diagram for mln::Site< E >:

8.297.1 Detailed Description

template<typename E> struct mln::Site< E >

Base class for classes that are explicitly sites.

8.298 mln::Site< void > Struct Template Reference

Site category flag type.

`#include <site.hh>`

8.298.1 Detailed Description

template<> struct mln::Site< void >

Site category flag type.

8.299 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.

8.299.1 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.

8.299.2 Member Function Documentation

8.299.2.1 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.

8.300 mln::Site_Proxy< E > Struct Template Reference

Base class for implementation classes of the notion of "site proxy".

`#include <site_proxy.hh>`

Inherits Proxy< E >.

8.300.1 Detailed Description

template<typename E> struct mln::Site_Proxy< E >

Base class for implementation classes of the notion of "site proxy".

FIXME: Explain...

8.301 mln::Site_Proxy< void > Struct Template Reference

Site_Proxy category flag type.

`#include <site_proxy.hh>`

8.301.1 Detailed Description

template<> struct mln::Site_Proxy< void >

Site_Proxy category flag type.

8.302 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 Sl , typename Sr >
  p_set< typename Sl::site > diff (const Site_Set< Sl > &lhs, const Site_Set< Sr > &rhs)

  Set theoretic difference of `lhs` *and* `rhs`.

- template<typename Sl , typename Sr >
  p_set< typename Sl::site > inter (const Site_Set< Sl > &lhs, const Site_Set< Sr > &rhs)

  Intersection between a couple of point sets.

- template<typename Sl , typename Sr >
  bool operator< (const Site_Set< Sl > &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 Sl , typename Sr >
  bool operator<= (const Site_Set< Sl > &lhs, const Site_Set< Sr > &rhs)

  Inclusion test between site sets `lhs` *and* `rhs`.

- template<typename Sl , typename Sr >
  bool operator== (const Site_Set< Sl > &lhs, const Site_Set< Sr > &rhs)

  Equality test between site sets `lhs` *and* `rhs`.

- template<typename Sl , typename Sr >
  p_set< typename Sl::site > sym_diff (const Site_Set< Sl > &lhs, const Site_Set< Sr > &rhs)

  Set theoretic symmetrical difference of `lhs` *and* `rhs`.

- template<typename Sl , typename Sr >
  p_set< typename Sl::site > uni (const Site_Set< Sl > &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`.

8.302.1 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.

8.302.2 Friends And Related Function Documentation

8.302.2.1 template<typename Sl , typename Sr > p_set< typename Sl::site > diff (const Site_Set< Sl > & *lhs*, const Site_Set< Sr > & *rhs*) [related]

Set theoretic difference of `lhs` and `rhs`.

8.302.2.2 template<typename Sl , typename Sr > p_set< typename Sl::site > inter (const Site_Set< Sl > & *lhs*, const Site_Set< Sr > & *rhs*) [related]

Intersection between a couple of point sets.

8.302.2.3 template<typename Sl , typename Sr > bool operator< (const Site_Set< Sl > & *lhs*, const Site_Set< Sr > & *rhs*) [related]

Strict inclusion test between site sets `lhs` and `rhs`.

Parameters:

← ***lhs*** A site set (strictly included?).

← ***rhs*** Another site set (includer?).

8.302.2.4 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:

↔ ***ostr*** An output stream.

← ***set*** A site set.

Returns:

The modified output stream `ostr`.

8.302.2.5 template<typename Sl , typename Sr > bool operator<= (const Site_Set< Sl > & *lhs*, const Site_Set< Sr > & *rhs*) [related]

Inclusion test between site sets `lhs` and `rhs`.

Parameters:

← ***lhs*** A site set (included?).

← ***rhs*** Another site set (includer?).

8.302.2.6 template<typename Sl , typename Sr > bool operator== (const Site_Set< Sl > & *lhs*, const Site_Set< Sr > & *rhs*) [related]

Equality test between site sets `lhs` and `rhs`.

Parameters:

← ***lhs*** A site set.

← ***rhs*** Another site set.

8.302.2.7 template<typename Sl , typename Sr > p_set< typename Sl::site > sym_diff (const Site_Set< Sl > & *lhs*, const Site_Set< Sr > & *rhs*) [related]

Set theoretic symmetrical difference of `lhs` and `rhs`.

8.302.2.8 template<typename Sl , typename Sr > p_set< typename Sl::site > uni (const Site_Set< Sl > & *lhs*, const Site_Set< Sr > & *rhs*) [related]

Union of a couple of point sets.

8.302.2.9 template<typename S > p_set< typename S::site > unique (const Site_Set< S > & *s*) [related]

Give the unique set of `s`.

8.303 mln::Site_Set< void > Struct Template Reference

Site_Set category flag type.

`#include <site_set.hh>`

8.303.1 Detailed Description

template<> struct mln::Site_Set< void >

Site_Set category flag type.

8.304 mln::sub_image< I, S > Struct 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 (const I &ima, const S &pset)
  Constructor.
- sub_image ()
  Constructor without argument.

8.304.1 Detailed Description

template<typename I, typename S> struct mln::sub_image< I, S >

Image having its domain restricted by a site set.

8.304.2 Member Typedef Documentation

8.304.2.1 template<typename I, typename S> typedef sub_image< tag::image_<I>, tag::domain_<S> > mln::sub_image< I, S >::skeleton

Skeleton.

8.304.3 Constructor & Destructor Documentation

8.304.3.1 template<typename I , typename S > mln::sub_image< I, S >::sub_image () [inline]

Constructor without argument.

8.304.3.2 template<typename I , typename S > mln::sub_image< I, S >::sub_image (const I & *ima*, const S & *pset*) [inline]

Constructor.

8.304.4 Member Function Documentation

8.304.4.1 template<typename I , typename S > const S & mln::sub_image< I, S >::domain () const [inline]

Give the definition domain.

8.304.4.2 template<typename I , typename S > mln::sub_image< I, S >::operator sub_image< const I, S > () const [inline]

Const promotion via conversion.

8.305 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 (I &ima, const S &s)

  Constructor.

- sub_image_if ()

  Constructor without argument.

8.305.1 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.

8.305.2 Member Typedef Documentation

8.305.2.1 template<typename I, typename S> typedef sub_image_if< tag::image_<I>, tag::domain_<S> > mln::sub_image_if< I, S >::skeleton

Skeleton.

8.305.3 Constructor & Destructor Documentation

8.305.3.1 template<typename I , typename S > mln::sub_image_if< I, S >::sub_image_if () [inline]

Constructor without argument.

8.305.3.2 template<typename I , typename S > mln::sub_image_if< I, S >::sub_image_if (I & *ima*, const S & *s*) [inline]

Constructor.

8.305.4 Member Function Documentation

8.305.4.1 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.

8.306 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.

8.306.1 Detailed Description

template<typename I, typename F> class mln::thru_image< I, F >

Morph image values through a function.

8.306.2 Member Function Documentation

8.306.2.1 template<typename I , typename F > mln::thru_image< I, F >::operator thru_image< const I, F > () const [inline]

Const promotion via conversion.

8.307 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.

8.307.1 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.

8.307.2 Member Typedef Documentation

8.307.2.1 template<typename I1, typename I2, typename F> typedef I1 ::psite mln::thrubin_image< I1, I2, F >::psite

Point_Site associated type.

8.307.2.2 template<typename I1, typename I2, typename F> typedef value mln::thrubin_image< I1, I2, F >::rvalue

Return type of read-only access.

8.307.2.3 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.

8.307.2.4 template<typename I1, typename I2, typename F> typedef F ::result mln::thrubin_image< I1, I2, F >::value

Value associated type.

8.307.3 Member Function Documentation

8.307.3.1 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.

8.308 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 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.

8.308.1 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.

8.308.2 Constructor & Destructor Documentation

8.308.2.1 template<unsigned D> mln::topo::adj_higher_dim_connected_n_face_bkd_iter< D >::adj_higher_dim_connected_n_face_bkd_iter () [inline]

Construction.

8.308.3 Member Function Documentation

8.308.3.1 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.

8.309 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 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.

8.309.1 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.

8.309.2 Constructor & Destructor Documentation

8.309.2.1 template<unsigned D> mln::topo::adj_higher_dim_connected_n_face_fwd_iter< D >::adj_higher_dim_connected_n_face_fwd_iter () [inline]

Construction.

8.309.3 Member Function Documentation

8.309.3.1 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-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition:

The iterator is valid.

8.310 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.

8.310.1 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.

8.310.2 Constructor & Destructor Documentation

8.310.2.1 template<unsigned D> mln::topo::adj_higher_face_bkd_iter< D >::adj_higher_face_bkd_iter () [inline]

Construction.

8.310.3 Member Function Documentation

8.310.3.1 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.

8.311 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.

8.311.1 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.

8.311.2 Constructor & Destructor Documentation

8.311.2.1 template<unsigned D> mln::topo::adj_higher_face_fwd_iter< D >::adj_higher_face_fwd_iter () [inline]

Construction.

8.311.3 Member Function Documentation

8.311.3.1 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.

8.312 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 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.

8.312.1 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.

8.312.2 Constructor & Destructor Documentation

8.312.2.1 template<unsigned D> mln::topo::adj_lower_dim_connected_n_face_bkd_iter< D >::adj_lower_dim_connected_n_face_bkd_iter () [inline]

Construction.

8.312.3 Member Function Documentation

8.312.3.1 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.

8.313 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>.

`#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 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.

8.313.1 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.

8.313.2 Constructor & Destructor Documentation

8.313.2.1 template<unsigned D> mln::topo::adj_lower_dim_connected_n_face_fwd_iter< D >::adj_lower_dim_connected_n_face_fwd_iter () [inline]

Construction.

8.313.3 Member Function Documentation

8.313.3.1 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.

8.314 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

- adj_lower_face_bkd_iter ()
  Construction.

8.314.1 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.

8.314.2 Constructor & Destructor Documentation

8.314.2.1 template<unsigned D> mln::topo::adj_lower_face_bkd_iter< D >::adj_lower_face_bkd_iter () [inline]

Construction.

8.315 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

- adj_lower_face_fwd_iter ()
  Construction.

8.315.1 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.

8.315.2 Constructor & Destructor Documentation

8.315.2.1 template<unsigned D> mln::topo::adj_lower_face_fwd_iter< D >::adj_lower_face_fwd_iter () [inline]

Construction.

8.316 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.

8.316.1 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.

8.316.2 Constructor & Destructor Documentation

8.316.2.1 template<unsigned D> mln::topo::adj_lower_higher_face_bkd_iter< D >::adj_lower_higher_face_bkd_iter () [inline]

Construction.

8.316.3 Member Function Documentation

8.316.3.1 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.

8.317 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.

8.317.1 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.

8.317.2 Constructor & Destructor Documentation

8.317.2.1 template<unsigned D> mln::topo::adj_lower_higher_face_fwd_iter< D >::adj_lower_higher_face_fwd_iter () [inline]

Construction.

8.317.3 Member Function Documentation

8.317.3.1 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.

8.318 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 adj_m_face_iterator< D >.

Public Member Functions

- void next ()
  Go to the next element.

- 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.
- adj_m_face_bkd_iter ()
  Construction.

8.318.1 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.

8.318.2 Constructor & Destructor Documentation

8.318.2.1 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.

8.318.2.2 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*.

8.318.3 Member Function Documentation

8.318.3.1 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.

8.319 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 adj_m_face_iterator< D >.

Public Member Functions

- void next ()
  Go to the next element.

- 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.
- adj_m_face_fwd_iter ()
  Construction.

8.319.1 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.

8.319.2 Constructor & Destructor Documentation

8.319.2.1 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.

8.319.2.2 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*.

8.319.3 Member Function Documentation

8.319.3.1 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.

8.320 mln::topo::algebraic_face< D > Struct 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

- 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.
- algebraic_face (complex< D > &complex, unsigned n, unsigned face_id, bool sign)
  Build an algebraic face handle from complex and face_id.
- algebraic_face ()
  Build a non-initialized algebraic face handle.
- void invalidate ()
  Invalidate this handle.
- bool is_valid () const
  Is this handle valid?

- complex< D > cplx () const
  Accessors.
- face_data< N, D > & data () const
  Return the mln::topo::face_data pointed by this handle.
- void dec_face_id ()
  Decrement the id of the face.
- void dec_n ()
  Decrement the dimension of the face.

- unsigned face_id () const
  Return the id of the face.
- std::vector< algebraic_face< D > > higher_dim_adj_faces () const
  Return an array of face handles pointing to adjacent (n+1)-faces.
- void inc_face_id ()
  Increment the id of the face.
- void inc_n ()
  Increment the dimension of the face.
- std::vector< algebraic_face< D > > lower_dim_adj_faces () const
  Return an array of face handles pointing to adjacent (n-1)-faces.
- unsigned n () const
  Return the dimension of the face.
- void set_cplx (const complex< D > &cplx)
  Set the complex the face belongs to.
- void set_face_id (unsigned face_id)
  Set the id of the face.
- void set_n (unsigned n)
  Set the dimension of the face.
- void set_sign (bool sign)
  Set the sign of this face.
- bool sign () const
  Accessors.

8.320.1 Detailed Description

template<unsigned D> struct 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.

8.320.2 Constructor & Destructor Documentation

8.320.2.1 template<unsigned D> mln::topo::algebraic_face< D >::algebraic_face () [inline]

Build a non-initialized algebraic face handle.

8.320.2.2 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*.

8.320.2.3 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.

References mln::topo::face< D >::n().

8.320.2.4 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.

8.320.3 Member Function Documentation

8.320.3.1 complex<D> mln::topo::face< D >::cplx () const [inherited]

Accessors.

Return the complex the face belongs to.

Referenced by mln::topo::operator!=(), and mln::topo::operator==().

8.320.3.2 face_data<N, D>& mln::topo::face< D >::data () const [inline, inherited]

Return the mln::topo::face_data pointed by this handle.

8.320.3.3 void mln::topo::face< D >::dec_face_id () [inherited]

Decrement the id of the face.

8.320.3.4 void mln::topo::face< D >::dec_n () [inherited]

Decrement the dimension of the face.

8.320.3.5 unsigned mln::topo::face< D >::face_id () const [inherited]

Return the id of the face.

Referenced by mln::topo::operator==().

8.320.3.6 std::vector< algebraic_face<D> > mln::topo::face< D >::higher_dim_adj_faces () const [inherited]

Return an array of face handles pointing to adjacent (n+1)-faces.

8.320.3.7 void mln::topo::face< D >::inc_face_id () [inherited]

Increment the id of the face.

8.320.3.8 void mln::topo::face< D >::inc_n () [inherited]

Increment the dimension of the face.

8.320.3.9 void mln::topo::face< D >::invalidate () [inherited]

Invalidate this handle.

8.320.3.10 bool mln::topo::face< D >::is_valid () const [inherited]

Is this handle valid?

8.320.3.11 std::vector< algebraic_face<D> > mln::topo::face< D >::lower_dim_adj_faces () const [inherited]

Return an array of face handles pointing to adjacent (n-1)-faces.

8.320.3.12 unsigned mln::topo::face< D >::n () const [inherited]

Return the dimension of the face.

Referenced by mln::topo::operator==().

8.320.3.13 void mln::topo::face< D >::set_cplx (const complex< D > & *cplx*) [inherited]

Set the complex the face belongs to.

8.320.3.14 void mln::topo::face< D >::set_face_id (unsigned *face_id*) [inherited]

Set the id of the face.

8.320.3.15 void mln::topo::face< D >::set_n (unsigned *n*) [inherited]

Set the dimension of the face.

8.320.3.16 template<unsigned D> void mln::topo::algebraic_face< D >::set_sign (bool *sign*) [inline]

Set the sign of this face.

8.320.3.17 template<unsigned D> bool mln::topo::algebraic_face< D >::sign () const [inline]

Accessors.

Return the sign of this face.

Referenced by mln::topo::operator==().

8.321 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 (const n_face< N, D > &f, bool sign)
  Build an algebraic face handle from an mln::n_face.
- algebraic_n_face (complex< D > &complex, unsigned face_id, bool sign)
  Build an algebraic face handle from complex *and* face_id.
- algebraic_n_face ()
  Build a non-initialized algebraic face handle.
- void invalidate ()
  Invalidate this handle.
- bool is_valid () const
  Is this handle valid?
- complex< D > cplx () const
  Accessors.
- face_data< N, D > & data () const
  Return the mln::topo::face_data pointed by this handle.
- void dec_face_id ()
  Decrement the id of the face.
- unsigned face_id () const
  Return the id of the face.
- 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.

- void inc_face_id ()
  Increment the id of the face.
- 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.
- unsigned n () const
  Return the dimension of the face.
- void set_cplx (const complex< D > &cplx)
  Set the complex the face belongs to.
- void set_face_id (unsigned face_id)
  Set the id of the face.
- void set_sign (bool sign)
  Set the sign of this face.
- bool sign () const
  Accessors.

8.321.1 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.

8.321.2 Constructor & Destructor Documentation

8.321.2.1 template<unsigned N, unsigned D> mln::topo::algebraic_n_face< N, D >::algebraic_n_face () [inline]

Build a non-initialized algebraic face handle.

References mln::topo::n_face< N, D >::is_valid().

8.321.2.2 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*.

8.321.2.3 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.

8.321.3 Member Function Documentation

8.321.3.1 complex<D> mln::topo::n_face< N, D >::cplx () const [inherited]

Accessors.

Return the complex the face belongs to.

Referenced by mln::topo::n_faces_set< N, D >::add(), mln::topo::operator!=(), and mln::topo::operator==().

8.321.3.2 face_data<N, D>& mln::topo::n_face< N, D >::data () const [inherited]

Return the mln::topo::face_data pointed by this handle.

8.321.3.3 void mln::topo::n_face< N, D >::dec_face_id () [inherited]

Decrement the id of the face.

8.321.3.4 unsigned mln::topo::n_face< N, D >::face_id () const [inherited]

Return the id of the face.

Referenced by mln::topo::operator==().

8.321.3.5 std::vector< algebraic_n_face<N + 1, D> > mln::topo::n_face< N, D >::higher_dim_adj_faces () const [inherited]

Return an array of face handles pointing to adjacent (n+1)-faces.

8.321.3.6 void mln::topo::n_face< N, D >::inc_face_id () [inherited]

Increment the id of the face.

8.321.3.7 void mln::topo::n_face< N, D >::invalidate () [inherited]

Invalidate this handle.

8.321.3.8 bool mln::topo::n_face< N, D >::is_valid () const [inherited]

Is this handle valid?

Referenced by mln::topo::algebraic_n_face< N, D >::algebraic_n_face().

8.321.3.9 std::vector< algebraic_n_face<N - 1, D> > mln::topo::n_face< N, D >::lower_dim_adj_faces () const [inherited]

Return an array of face handles pointing to adjacent (n-1)-faces.

8.321.3.10 unsigned mln::topo::n_face< N, D >::n () const [inherited]

Return the dimension of the face.

8.321.3.11 void mln::topo::n_face< N, D >::set_cplx (const complex< D > & *cplx*) [inherited]

Set the complex the face belongs to.

8.321.3.12 void mln::topo::n_face< N, D >::set_face_id (unsigned *face_id*) [inherited]

Set the id of the face.

8.321.3.13 template<unsigned N, unsigned D> void mln::topo::algebraic_n_face< N, D >::set_sign (bool *sign*) [inline]

Set the sign of this face.

8.321.3.14 template<unsigned N, unsigned D> bool mln::topo::algebraic_n_face< N, D >::sign () const [inline]

Accessors.

Return the sign of this face.

Referenced by mln::topo::operator==().

8.322 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.

8.322.1 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

8.322.2 Constructor & Destructor Documentation

8.322.2.1 template<unsigned D> mln::topo::center_only_iter< D >::center_only_iter () [inline]

Construction.

8.322.3 Member Function Documentation

8.322.3.1 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.

8.323 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.

8.323.1 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 adapated complex relative iterator.

8.323.2 Constructor & Destructor Documentation

8.323.2.1 template<unsigned D, typename I > mln::topo::centered_bkd_iter_adapter< D, I >::centered_bkd_iter_adapter () [inline]

Construction.

8.323.3 Member Function Documentation

8.323.3.1 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.

8.324 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.

8.324.1 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 adapated complex relative iterator.

8.324.2 Constructor & Destructor Documentation

8.324.2.1 template<unsigned D, typename I > mln::topo::centered_fwd_iter_adapter< D, I >::centered_fwd_iter_adapter () `[inline]`

Construction.

8.324.3 Member Function Documentation

8.324.3.1 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.

8.325 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.

- 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).
- n_face< 0u, D > add_face ()
  Add a 0-face to the complex.
- complex ()
  Complex construction.

- 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.

8.325.1 Detailed Description

template<unsigned D> class mln::topo::complex< D >

General complex of dimension `D`.

8.325.2 Member Typedef Documentation

8.325.2.1 template<unsigned D> typedef face_bkd_iter<D> mln::topo::complex< D >::bkd_citer

Backward mln::Iterator type iterating on all faces.

8.325.2.2 template<unsigned D> typedef face_fwd_iter<D> mln::topo::complex< D >::fwd_citer

Forward mln::Iterator type iterating on all faces.

8.325.3 Constructor & Destructor Documentation

8.325.3.1 template<unsigned D> mln::topo::complex< D >::complex () [inline]

Complex construction.

Create a new `D-complex`.

8.325.4 Member Function Documentation

8.325.4.1 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` >= 0).

Parameters:

adjacent_faces The (`N-1`)-faces adjacent to the new `N-face`.

References mln::topo::n_faces_set< N, D >::faces().

8.325.4.2 template<unsigned D> n_face< 0u, D > mln::topo::complex< D >::add_face () [inline]

Add a 0-face to the complex.

8.325.4.3 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.

8.325.4.4 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.

8.325.4.5 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).

8.325.4.6 template<unsigned D> template<unsigned N> unsigned mln::topo::complex< D >::nfaces_of_static_dim () const [inline]

Return the number of `N-faces`.

8.325.4.7 template<unsigned D> void mln::topo::complex< D >::print (std::ostream & *ostr*) const [inline]

Pretty-printing.

Print the complex.

Referenced by mln::topo::operator<<().

8.325.4.8 template<unsigned D> template<unsigned N> void mln::topo::complex< D >::print_faces (std::ostream & *ostr*) const [inline]

Print the faces of dimension `N`.

8.326 mln::topo::face< D > Struct 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

- template<unsigned N>
  face (const n_face< N, D > &f)
  Build a face handle from an mln::topo::n_face.
- face (complex< D > &complex, unsigned n, unsigned face_id)
  Build a face handle from complex and face_id.
- face ()
  Build a non-initialized face handle.
- void invalidate ()
  Invalidate this handle.
- bool is_valid () const
  Is this handle valid?
- complex< D > cplx () const
  Accessors.
- template<unsigned N>
  face_data< N, D > & data () const
  Return the mln::topo::face_data pointed by this handle.
- void dec_face_id ()
  Decrement the id of the face.
- void dec_n ()
  Decrement the dimension of the face.
- unsigned face_id () const
  Return the id of the face.
- std::vector< algebraic_face< D > > higher_dim_adj_faces () const
  Return an array of face handles pointing to adjacent (n+1)-faces.
- void inc_face_id ()
  Increment the id of the face.

- void inc_n ()
  Increment the dimension of the face.
- std::vector< algebraic_face< D > > lower_dim_adj_faces () const
  Return an array of face handles pointing to adjacent (n-1)-faces.
- unsigned n () const
  Return the dimension of the face.
- void set_cplx (const complex< D > &cplx)
  Set the complex the face belongs to.
- void set_face_id (unsigned face_id)
  Set the id of the face.
- void set_n (unsigned n)
  Set the dimension of the face.

8.326.1 Detailed Description

template<unsigned D> struct 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.

8.326.2 Constructor & Destructor Documentation

8.326.2.1 template<unsigned D> mln::topo::face< D >::face () [inline]

Build a non-initialized face handle.

8.326.2.2 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*.

8.326.2.3 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.

8.326.3 Member Function Documentation

8.326.3.1 template<unsigned D> complex< D > mln::topo::face< D >::cplx () const [inline]

Accessors.

Return the complex the face belongs to.

Referenced by mln::complex_psite< D, G >::complex_psite(), mln::topo::operator!=(), and mln::topo::operator==().

8.326.3.2 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.

8.326.3.3 template<unsigned D> void mln::topo::face< D >::dec_face_id () [inline]

Decrement the id of the face.

8.326.3.4 template<unsigned D> void mln::topo::face< D >::dec_n () [inline]

Decrement the dimension of the face.

8.326.3.5 template<unsigned D> unsigned mln::topo::face< D >::face_id () const [inline]

Return the id of the face.

Referenced by mln::geom::complex_geometry< D, P >::operator()(), and mln::topo::operator==().

8.326.3.6 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.

8.326.3.7 template<unsigned D> void mln::topo::face< D >::inc_face_id () [inline]

Increment the id of the face.

8.326.3.8 template<unsigned D> void mln::topo::face< D >::inc_n () [inline]

Increment the dimension of the face.

8.326.3.9 template<unsigned D> void mln::topo::face< D >::invalidate () [inline]

Invalidate this handle.

References mln::topo::face< D >::set_face_id(), and mln::topo::face< D >::set_n().

8.326.3.10 template<unsigned D> bool mln::topo::face< D >::is_valid () const [inline]

Is this handle valid?

8.326.3.11 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.

8.326.3.12 template<unsigned D> unsigned mln::topo::face< D >::n () const [inline]

Return the dimension of the face.

Referenced by mln::topo::algebraic_face< D >::algebraic_face(), mln::geom::complex_geometry< D, P >::operator()(), and mln::topo::operator==().

8.326.3.13 template<unsigned D> void mln::topo::face< D >::set_cplx (const complex< D > & *cplx*) [inline]

Set the complex the face belongs to.

8.326.3.14 template<unsigned D> void mln::topo::face< D >::set_face_id (unsigned *face_id*) [inline]

Set the id of the face.

Referenced by mln::topo::face< D >::invalidate().

8.326.3.15 template<unsigned D> void mln::topo::face< D >::set_n (unsigned *n*) [inline]

Set the dimension of the face.

Referenced by mln::topo::face< D >::invalidate().

8.327 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.

8.327.1 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.

8.327.2 Constructor & Destructor Documentation

8.327.2.1 template<unsigned D> mln::topo::face_bkd_iter< D >::face_bkd_iter () [inline]

Construction and assignment.

8.327.3 Member Function Documentation

8.327.3.1 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.

8.327.3.2 template<unsigned D> void mln::topo::face_bkd_iter< D >::start () [inline]

Manipulation.

Start an iteration.

8.328 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.

8.328.1 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.

8.328.2 Constructor & Destructor Documentation

8.328.2.1 template<unsigned D> mln::topo::face_fwd_iter< D >::face_fwd_iter () [inline]

Construction and assignment.

8.328.3 Member Function Documentation

8.328.3.1 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.

8.328.3.2 template<unsigned D> void mln::topo::face_fwd_iter< D >::start () [inline]

Manipulation.

Test if the iterator is valid.

8.329 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 >:

8.329.1 Detailed Description

template<unsigned N> struct mln::topo::is_n_face< N >

A functor testing wheter a mln::complex_psite is an N -face.

8.330 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).

8.330.1 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.

8.330.2 Member Typedef Documentation

8.330.2.1 template<typename I > typedef mln::complex_psite<D, G> mln::topo::is_simple_cell< I >::psite

Psite type.

8.330.2.2 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 > >.

8.330.3 Member Function Documentation

8.330.3.1 template<typename I > typedef mln::topo::is_simple_cell< I >::mln_geom (I)

Geometry of the image.

8.330.3.2 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.

References mln::make::attachment().

8.330.3.3 template<typename I > void mln::topo::is_simple_cell< I >::set_image (const mln::Image< I > & *ima*) [inline]

Set the underlying image.

8.330.4 Member Data Documentation

8.330.4.1 template<typename I > const unsigned mln::topo::is_simple_cell< I >::D = I::dim [static]

Dimension of the image (and therefore of the complex).

8.331 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 (complex< D > &complex, unsigned face_id)
  Build a face handle from complex *and* face_id.
- n_face ()
  Build a non-initialized face handle.

- complex< D > cplx () const
  Accessors.
- face_data< N, D > & data () const
  Return the mln::topo::face_data pointed by this handle.
- void dec_face_id ()
  Decrement the id of the face.
- unsigned face_id () const
  Return the id of the face.
- 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.
- void inc_face_id ()
  Increment the id of the face.
- 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.
- unsigned n () const
  Return the dimension of the face.
- void set_cplx (const complex< D > &cplx)

Set the complex the face belongs to.

- void set_face_id (unsigned face_id)
  Set the id of the face.

8.331.1 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.

8.331.2 Constructor & Destructor Documentation

8.331.2.1 template<unsigned N, unsigned D> mln::topo::n_face< N, D >::n_face () [inline]

Build a non-initialized face handle.

References mln::topo::n_face< N, D >::is_valid().

8.331.2.2 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*.

8.331.3 Member Function Documentation

8.331.3.1 template<unsigned N, unsigned D> complex< D > mln::topo::n_face< N, D >::cplx () const [inline]

Accessors.

Return the complex the face belongs to.

Referenced by mln::topo::operator!=(), and mln::topo::operator==().

8.331.3.2 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.

References mln::topo::n_face< N, D >::is_valid().

8.331.3.3 template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::dec_face_id () [inline]

Decrement the id of the face.

8.331.3.4 template<unsigned N, unsigned D> unsigned mln::topo::n_face< N, D >::face_id () const [inline]

Return the id of the face.

Referenced by mln::topo::operator==().

8.331.3.5 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.

References mln::topo::n_face< N, D >::is_valid().

Referenced by mln::topo::edge().

8.331.3.6 template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::inc_face_id () [inline]

Increment the id of the face.

8.331.3.7 template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::invalidate () [inline]

Invalidate this handle.

References mln::topo::n_face< N, D >::set_face_id().

8.331.3.8 template<unsigned N, unsigned D> bool mln::topo::n_face< N, D >::is_valid () const [inline]

Is this handle valid?

Referenced by mln::topo::n_face< N, D >::data(), mln::topo::n_face< N, D >::higher_dim_adj_faces(), and mln::topo::n_face< N, D >::n_face().

8.331.3.9 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.

8.331.3.10 template<unsigned N, unsigned D> unsigned mln::topo::n_face< N, D >::n () const [inline]

Return the dimension of the face.

8.331.3.11 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.

8.331.3.12 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.

Referenced by mln::topo::n_face< N, D >::invalidate().

8.332 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.
- unsigned n () const
  Accessors.
- void start ()
  Manipulation.

8.332.1 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.

8.332.2 Member Function Documentation

8.332.2.1 template<unsigned D> unsigned mln::topo::n_face_bkd_iter< D >::n () const [inline]

Accessors.

Shortcuts to face_'s accessors.

Referenced by mln::topo::n_face_bkd_iter< D >::start().

8.332.2.2 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.

8.332.2.3 template<unsigned D> void mln::topo::n_face_bkd_iter< D >::start () [inline]

Manipulation.

Start an iteration.

References mln::topo::n_face_bkd_iter< D >::n().

8.333 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.
- unsigned n () const
  Accessors.
- void start ()
  Manipulation.

8.333.1 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.

8.333.2 Member Function Documentation

8.333.2.1 template<unsigned D> unsigned mln::topo::n_face_fwd_iter< D >::n () const [inline]

Accessors.

Shortcuts to face_'s accessors.

8.333.2.2 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.

8.333.2.3 template<unsigned D> void mln::topo::n_face_fwd_iter< D >::start () [inline]

Manipulation.

Test if the iterator is valid.

8.334 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.

8.334.1 Detailed Description

template<unsigned N, unsigned D> class mln::topo::n_faces_set< N, D >

Set of face handles of dimension N.

8.334.2 Member Typedef Documentation

8.334.2.1 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.

8.334.3 Member Function Documentation

8.334.3.1 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.

References mln::topo::n_face< N, D >::cplx().

Referenced by mln::topo::operator+(), and mln::topo::operator-().

8.334.3.2 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.

Referenced by mln::topo::complex< D >::add_face().

8.334.3.3 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.

8.335 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.
- void start ()
  Manipulation.
- static_n_face_bkd_iter ()
  Construction and assignment.

8.335.1 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.

8.335.2 Constructor & Destructor Documentation

8.335.2.1 template<unsigned N, unsigned D> mln::topo::static_n_face_bkd_iter< N, D >::static_n_face_bkd_iter () `[inline]`

Construction and assignment.

8.335.3 Member Function Documentation

8.335.3.1 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-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition:

The iterator is valid.

8.335.3.2 template<unsigned N, unsigned D> void mln::topo::static_n_face_bkd_iter< N, D >::start () [inline]

Manipulation.

Start an iteration.

8.336 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.
- void start ()
  Manipulation.
- static_n_face_fwd_iter ()
  Construction and assignment.

8.336.1 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.

8.336.2 Constructor & Destructor Documentation

8.336.2.1 template<unsigned N, unsigned D> mln::topo::static_n_face_fwd_iter< N, D >::static_n_face_fwd_iter () `[inline]`

Construction and assignment.

8.336.3 Member Function Documentation

8.336.3.1 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.

8.336.3.2 template<unsigned N, unsigned D> void mln::topo::static_n_face_fwd_iter< N, D >::start () [inline]

Manipulation.

Test if the iterator is valid.

8.337 mln::tr_image< S, I, T > Struct Template Reference

Transform an image by a given transformation.

`#include <tr_image.hh>`

Inherits image_identity< I, I::domain_t, 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.

8.337.1 Detailed Description

template<typename S, typename I, typename T> struct mln::tr_image< S, I, T >

Transform an image by a given transformation.

8.337.2 Member Typedef Documentation

8.337.2.1 template<typename S, typename I, typename T> typedef I ::value mln::tr_image< S, I, T >::lvalue

Return type of read-write access.

8.337.2.2 template<typename S, typename I, typename T> typedef I ::psite mln::tr_image< S, I, T >::psite

Point_Site associated type.

8.337.2.3 template<typename S, typename I, typename T> typedef I ::value mln::tr_image< S, I, T >::rvalue

Return type of read-only access.

8.337.2.4 template<typename S, typename I, typename T> typedef I ::site mln::tr_image< S, I, T >::site

Site associated type.

8.337.2.5 template<typename S, typename I, typename T> typedef tr_image< S, tag::image_<I>, T> mln::tr_image< S, I, T >::skeleton

Skeleton.

8.337.2.6 template<typename S, typename I, typename T> typedef I ::value mln::tr_image< S, I, T >::value

Value associated type.

8.337.3 Constructor & Destructor Documentation

8.337.3.1 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.

8.337.4 Member Function Documentation

8.337.4.1 template<typename S , typename I , typename T > const S & mln::tr_image< S, I, T >::domain () const [inline]

Return the domain morpher.

8.337.4.2 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`.

8.337.4.3 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.

8.337.4.4 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).

8.337.4.5 template<typename S , typename I , typename T > void mln::tr_image< S, I, T >::set_tr (T & *tr*) [inline]

Set the transformation.

8.337.4.6 template<typename S , typename I , typename T > const T & mln::tr_image< S, I, T >::tr () const [inline]

Return the underlying transformation.

8.338 mln::transformed_image< I, F > Struct Template Reference

Image having its domain restricted by a site set.

`#include <transformed_image.hh>`

Inherits image_domain_morpher< I, p_transformed< I::domain_t, F >, transformed_image< I, F > >.

Public Types

- typedef transformed_image< tag::image_< I >, tag::function_< F > > skeleton

  Skeleton.

Public Member Functions

- const p_transformed< typename I::domain_t, F > & domain () const

  Give the definition domain.

- operator transformed_image< const I, F > () const

  Const promotion via conversion.

- internal::morpher_lvalue_< I >::ret operator() (const typename I::psite &p)

  Read and "write if possible" access of pixel value at point site `p`.

- I::rvalue operator() (const typename I::psite &p) const

  Read-only access of pixel value at point site `p`.

- transformed_image (I &ima, const F &f)

  Constructor.

- transformed_image ()

  Constructor without argument.

8.338.1 Detailed Description

template<typename I, typename F> struct mln::transformed_image< I, F >

Image having its domain restricted by a site set.

8.338.2 Member Typedef Documentation

8.338.2.1 template<typename I, typename F> typedef transformed_image< tag::image_<I>, tag::function_<F> > mln::transformed_image< I, F >::skeleton

Skeleton.

8.338.3 Constructor & Destructor Documentation

8.338.3.1 template<typename I , typename F > mln::transformed_image< I, F >::transformed_image () [inline]

Constructor without argument.

8.338.3.2 template<typename I , typename F > mln::transformed_image< I, F >::transformed_image (I & *ima*, const F & *f*) [inline]

Constructor.

8.338.4 Member Function Documentation

8.338.4.1 template<typename I , typename F > const p_transformed< typename I::domain_t, F > & mln::transformed_image< I, F >::domain () const [inline]

Give the definition domain.

8.338.4.2 template<typename I , typename F > mln::transformed_image< I, F >::operator transformed_image< const I, F > () const [inline]

Const promotion via conversion.

8.338.4.3 template<typename I , typename F > internal::morpher_lvalue_< I >::ret mln::transformed_image< I, F >::operator() (const typename I::psite & *p*) [inline]

Read and "write if possible" access of pixel value at point site p.

8.338.4.4 template<typename I , typename F > I::rvalue mln::transformed_image< I, F >::operator() (const typename I::psite & *p*) const [inline]

Read-only access of pixel value at point site p.

8.339 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 (I &ima, const D &dom, const F &f)
  Constructor from an image `ima`*, a domain* `dom`*, and a function* `f`.
- unproject_image ()
  Constructor without argument.

8.339.1 Detailed Description

template<typename I, typename D, typename F> struct mln::unproject_image< I, D, F >

Un-projects an image.

8.339.2 Constructor & Destructor Documentation

8.339.2.1 template<typename I , typename D , typename F > mln::unproject_image< I, D, F >::unproject_image () [inline]

Constructor without argument.

8.339.2.2 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`.

8.339.3 Member Function Documentation

8.339.3.1 template<typename I , typename D , typename F > const D & mln::unproject_image< I, D, F >::domain () const [inline]

Give the definition domain.

8.339.3.2 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.

8.339.3.3 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.

8.340 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_quant(V), trait::value::quant::low >::eval >.

Public Member Functions

- adjacency_matrix (const V &nelements)

  Construct an adjacency matrix with `nelements` *elements maximum.*

- adjacency_matrix ()

  Constructors.

8.340.1 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.

8.340.2 Constructor & Destructor Documentation

8.340.2.1 template<typename V > mln::util::adjacency_matrix< V >::adjacency_matrix () [inline]

Constructors.

@{

Default

8.340.2.2 template<typename V > mln::util::adjacency_matrix< V >::adjacency_matrix (const V & *nelements*) [inline]

Construct an adjacency matrix with `nelements` elements maximum.

8.341 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 array_bkd_iter< T > bkd_eiter
  Backward iterator associated type.

- typedef fwd_eiter eiter
  Iterator associated type.

- typedef array_fwd_iter< T > fwd_eiter
  Iterator types
  Forward iterator associated type.

- typedef T result
  Returned value types.

Public Member Functions

- template<typename U >
  array< T > & append (const array< U > &other)
  Add the elements of `other` *at the end of this array.*

- array< T > & append (const T &elt)
  Add the element `elt` *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.

- 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.

- mutable_result operator() (unsigned i)
  Return the `i-th` *element of the array.*

- const T & 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.*

- const T & operator[ ] (unsigned i) const
  Return the `i-th` *element of the array.*

- void reserve (unsigned n)
  Reserve memory for `n` *elements.*

- void resize (unsigned n, const T &value)

Resize this array to n *elements with* value *as value.*

- void resize (unsigned n)

  Resize this array to n *elements.*

- 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 (unsigned n, const T &value)

  Construct a new array, resize it to elements and fill it with default_value.

- array (unsigned n)

  Construct a new array and resize it to elements.

- array ()

  Constructors
  Constructor without arguments.

8.341.1 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.

8.341.2 Member Typedef Documentation

8.341.2.1 template<typename T> typedef array_bkd_iter<T> mln::util::array< T >::bkd_eiter

Backward iterator associated type.

8.341.2.2 template<typename T> typedef fwd_eiter mln::util::array< T >::eiter

Iterator associated type.

8.341.2.3 template<typename T> typedef T mln::util::array< T >::element

Element associated type.

8.341.2.4 template<typename T> typedef array_fwd_iter<T> mln::util::array< T >::fwd_eiter

Iterator types

Forward iterator associated type.

8.341.2.5 template<typename T> typedef T mln::util::array< T >::result

Returned value types.

Related to the Function_v2v concept.

8.341.3 Constructor & Destructor Documentation

8.341.3.1 template<typename T > mln::util::array< T >::array () [inline]

Constructors

Constructor without arguments.

8.341.3.2 template<typename T > mln::util::array< T >::array (unsigned *n*) [inline]

Construct a new array and resize it to

elements.

8.341.3.3 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`.

8.341.4 Member Function Documentation

8.341.4.1 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.

References mln::util::array< T >::is_empty(), and mln::util::array< T >::std_vector().

8.341.4.2 template<typename T> array< T > & mln::util::array< T >::append (const T & *elt*) [inline]

Add the element `elt` at the end of this array.

Referenced by mln::io::plot::load(), and mln::data::impl::generic::sort_offsets_increasing().

8.341.4.3 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

References mln::util::array< T >::is_empty().

Referenced by mln::io::plot::load().

8.341.4.4 template<typename T> void mln::util::array< T >::fill (const T & *value*) [inline]

Fill the whole array with value `value`.

8.341.4.5 template<typename T > bool mln::util::array< T >::is_empty () const [inline]

Test if the array is empty.

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().

8.341.4.6 template<typename T > std::size_t mln::util::array< T >::memory_size () const [inline]

Return the size of this array in memory.

References mln::util::array< T >::nelements().

8.341.4.7 template<typename T > unsigned mln::util::array< T >::nelements () const [inline]

Return the number of elements of the array.

Referenced by mln::labeling::fill_holes(), mln::make::image3d(), mln::util::array< T >::is_empty(), 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().

8.341.4.8 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()

8.341.4.9 template<typename T > const T & mln::util::array< T >::operator() (unsigned *i*) const [inline]

Return the `i-th` element of the array.

Precondition:

i < nelements()

8.341.4.10 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()

References mln::util::array< T >::nelements().

8.341.4.11 template<typename T > const T & mln::util::array< T >::operator[ ] (unsigned *i*) const [inline]

Return the i-th element of the array.

Precondition:

i < nelements()

References mln::util::array< T >::nelements().

8.341.4.12 template<typename T > void mln::util::array< T >::reserve (unsigned *n*) [inline]

Reserve memory for n elements.

Referenced by mln::data::impl::generic::sort_offsets_increasing().

8.341.4.13 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.

8.341.4.14 template<typename T > void mln::util::array< T >::resize (unsigned *n*) [inline]

Resize this array to n elements.

8.341.4.15 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

References mln::util::array< T >::nelements().

Referenced by mln::value::lut_vec< S, T >::lut_vec().

8.341.4.16 template<typename T > const std::vector< T > & mln::util::array< T >::std_vector () const [inline]

Return the corresponding std::vector of elements.

Referenced by mln::util::array< T >::append(), mln::value::lut_vec< S, T >::lut_vec(), and mln::util::operator==().

8.342 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 appex.

- branch (tree< T > &tree, tree_node< T > &apex)

  Constructor.

- tree< T > & util_tree ()

  The getter of the tree.

8.342.1 Detailed Description

template<typename T> class mln::util::branch< T >

Class of generic branch.

8.342.2 Constructor & Destructor Documentation

8.342.2.1 template<typename T > mln::util::branch< T >::branch (util::tree< T > & *tree*, util::tree_node< T > & *apex*) [inline]

Constructor.

Parameters:

← ***tree*** The tree of the branch.

← ***apex*** The apex of the branch.

8.342.3 Member Function Documentation

8.342.3.1 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.

8.342.3.2 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.

8.343 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.

8.343.1 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.

8.343.2 Member Function Documentation

8.343.2.1 template<typename T > unsigned mln::util::branch_iter< T >::deepness () const [inline]

Give how deep is the iterator in the branch.

References mln::util::branch_iter< T >::is_valid(), and mln::util::tree_node< T >::parent().

8.343.2.2 template<typename T > void mln::util::branch_iter< T >::invalidate () [inline]

Invalidate the iterator.

Referenced by mln::util::branch_iter< T >::next().

8.343.2.3 template<typename T > bool mln::util::branch_iter< T >::is_valid () const [inline]

Test the iterator validity.

Referenced by mln::util::branch_iter< T >::deepness().

8.343.2.4 template<typename T > void mln::util::branch_iter< T >::next () [inline]

Go to the next point.

References mln::util::branch_iter< T >::invalidate().

8.343.2.5 template<typename T > mln::util::branch_iter< T >::operator util::tree_node< T > & () const [inline]

Conversion to node.

8.343.2.6 template<typename T > void mln::util::branch_iter< T >::start () [inline]

Start an iteration.

8.344 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.

8.344.1 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.

8.344.2 Member Function Documentation

8.344.2.1 template<typename T > unsigned mln::util::branch_iter_ind< T >::deepness () const [inline]

Give how deep is the iterator in the branch.

References mln::util::branch_iter_ind< T >::is_valid(), and mln::util::tree_node< T >::parent().

8.344.2.2 template<typename T > void mln::util::branch_iter_ind< T >::invalidate () [inline]

Invalidate the iterator.

Referenced by mln::util::branch_iter_ind< T >::next().

8.344.2.3 template<typename T > bool mln::util::branch_iter_ind< T >::is_valid () const [inline]

Test the iterator validity.

Referenced by mln::util::branch_iter_ind< T >::deepness().

8.344.2.4 template<typename T > void mln::util::branch_iter_ind< T >::next () [inline]

Go to the next point.

References mln::util::branch_iter_ind< T >::invalidate().

8.344.2.5 template<typename T > mln::util::branch_iter_ind< T >::operator util::tree_node< T > & () const [inline]

Conversion to node.

8.344.2.6 template<typename T > void mln::util::branch_iter_ind< T >::start () [inline]

Start an iteration.

8.345 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.

8.345.1 Detailed Description

template<typename T, typename U> class mln::util::couple< T, U >

Definition of a couple.

8.345.2 Member Function Documentation

8.345.2.1 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*.

8.345.2.2 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*.

8.345.2.3 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*.

8.345.2.4 template<typename T , typename U > const T & mln::util::couple< T, U >::first () const [inline]

Get the first member of the couple.

8.345.2.5 template<typename T , typename U > const U & mln::util::couple< T, U >::second () const [inline]

Get the second member of the couple.

8.346 mln::util::eat Struct Reference

Eat structure.

`#include <eat.hh>`

Inheritance diagram for mln::util::eat:

8.346.1 Detailed Description

Eat structure.

8.347 mln::util::edge< G > Class Template Reference

Edge of a graph G.

`#include <edge.hh>`

Inherits 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

- void change_graph (const G &g)
  Set g_ with g;.
- const G & graph () const
  Return a reference to the graph holding this edge.
- edge_id_t id () const
  Return the edge id.
- void invalidate ()
  Invalidate that vertex.
- bool is_valid () const
  Misc.
- operator edge_id_t () const
  Conversion to the edge id.
- void update_id (const edge_id_t &id)
  Set id_ with id;.
- edge ()
  Constructors.
- edge_id_t ith_nbh_edge (unsigned i) const

Return the `i` *th adjacent edge.*

- size_t nmax_nbh_edges () const
  Return the number max of adjacent edges.

- vertex_id_t v1 () const
  Edge oriented.

- vertex_id_t v2 () const
  Return the highest vertex id adjacent to this edge.

- vertex_id_t v_other (const vertex_id_t &id_v) const
  Vertex and edges oriented.

8.347.1 Detailed Description

template<typename G> class mln::util::edge< G >

Edge of a graph `G`.

8.347.2 Member Typedef Documentation

8.347.2.1 template<typename G> typedef Edge<void> mln::util::edge< G >::category

Object category.

8.347.2.2 template<typename G> typedef G mln::util::edge< G >::graph_t

Graph associated type.

8.347.2.3 template<typename G> typedef edge_id_t mln::util::edge< G >::id_t

The edge type id.

8.347.2.4 template<typename G> typedef edge_id_t::value_t mln::util::edge< G >::id_value_t

The underlying type used to store edge ids.

8.347.3 Constructor & Destructor Documentation

8.347.3.1 template<typename G > mln::util::edge< G >::edge () [inline]

Constructors.

References mln::util::edge< G >::invalidate().

8.347.4 Member Function Documentation

8.347.4.1 template<typename G > void mln::util::edge< G >::change_graph (const G & *g*) [inline]

Set g_ with g;.

8.347.4.2 template<typename G > const G & mln::util::edge< G >::graph () const [inline]

Return a reference to the graph holding this edge.

Referenced by mln::p_edges< G, F >::has(), and mln::util::line_graph< G >::has().

8.347.4.3 template<typename G > edge_id_t mln::util::edge< G >::id () const [inline]

Return the edge id.

Referenced by mln::util::line_graph< G >::has().

8.347.4.4 template<typename G > void mln::util::edge< G >::invalidate () [inline]

Invalidate that vertex.

Referenced by mln::util::edge< G >::edge().

8.347.4.5 template<typename G > bool mln::util::edge< G >::is_valid () const [inline]

Misc.

Return whether is points to a known edge.

Referenced by mln::p_edges< G, F >::has().

8.347.4.6 template<typename G > edge_id_t mln::util::edge< G >::ith_nbh_edge (unsigned *i*) const [inline]

Return the i th adjacent edge.

8.347.4.7 template<typename G > size_t mln::util::edge< G >::nmax_nbh_edges () const [inline]

Return the number max of adjacent edges.

8.347.4.8 template<typename G > mln::util::edge< G >::operator edge_id_t () const [inline]

Conversion to the edge id.

8.347.4.9 template<typename G > void mln::util::edge< G >::update_id (const edge_id_t & *id*) [inline]

Set id_ with `id`;.

8.347.4.10 template<typename G > vertex_id_t mln::util::edge< G >::v1 () const [inline]

Edge oriented.

Return the lowest vertex id adjacent to this edge.

Referenced by mln::util::edge< G >::v_other().

8.347.4.11 template<typename G > vertex_id_t mln::util::edge< G >::v2 () const [inline]

Return the highest vertex id adjacent to this edge.

Referenced by mln::util::edge< G >::v_other().

8.347.4.12 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`.

References mln::util::edge< G >::v1(), and mln::util::edge< G >::v2().

8.348 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 (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.*

- fibonacci_heap ()

  Default constructor.

- 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 (fibonacci_heap< P, T > &other_heap)

  Take `other_heap's` *elements and insert them in this heap.*

- void push (const P &priority, const T &value)

  Push a new element in the heap.

8.348.1 Detailed Description

template<typename P, typename T> class mln::util::fibonacci_heap< P, T >

Fibonacci heap.

8.348.2 Constructor & Destructor Documentation

8.348.2.1 template<typename P , typename T > mln::util::fibonacci_heap< P, T >::fibonacci_heap () [inline]

Default constructor.

8.348.2.2 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.

8.348.3 Member Function Documentation

8.348.3.1 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.

References mln::util::fibonacci_heap< P, T >::pop_front().

8.348.3.2 template<typename P , typename T > const T & mln::util::fibonacci_heap< P, T >::front () const [inline]

Return the minimum value in the heap.

8.348.3.3 template<typename P , typename T > bool mln::util::fibonacci_heap< P, T >::is_empty () const [inline]

Is it empty?

Referenced by mln::util::fibonacci_heap< P, T >::pop_front(), and mln::util::fibonacci_heap< P, T >::push().

8.348.3.4 template<typename P , typename T > bool mln::util::fibonacci_heap< P, T >::is_valid () const [inline]

return false if it is empty.

Referenced by mln::util::fibonacci_heap< P, T >::pop_front().

8.348.3.5 template<typename P , typename T > unsigned mln::util::fibonacci_heap< P, T >::nelements () const [inline]

Return the number of elements.

8.348.3.6 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.

8.348.3.7 template<typename P , typename T > T mln::util::fibonacci_heap< P, T >::pop_front () [inline]

Return and remove the minimum value in the heap.

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().

8.348.3.8 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.

References mln::util::fibonacci_heap< P, T >::is_empty().

8.348.3.9 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

Referenced by mln::util::fibonacci_heap< P, T >::pop_front().

8.349 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::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.

- 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.

Public Member Functions

- graph (unsigned nvertices)
  Construct a graph with `nvertices` vertices.
- graph ()
- bool has_v (const vertex_id_t &id_v) const
  Check whether a vertex id `id_v` exists in the graph.
- void invalidate ()
  Invalidate the graph.
- bool is_valid () const

Return true if this graph is valid.

- 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.

- edge_id_t add_edge (const vertex_id_t &id_v1, const vertex_id_t &id_v2)
  Edge oriented.
- 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`.
- size_t e_nmax () const
  Return the number of edges in the graph.
- edge_t edge (const vertex_t &v1, const vertex_t &v2) const
  @}
- 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.
- bool has_e (const edge_id_t &id_e) const
  Return whether `id_e` *is in the graph.*
- 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.
- 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`.

- 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.

8.349.1 Detailed Description

Undirected graph.

8.349.2 Member Typedef Documentation

8.349.2.1 typedef mln::internal::edge_fwd_iterator<graph> mln::util::graph::edge_fwd_iter

Edge iterators.

8.349.2.2 typedef mln::internal::edge_nbh_edge_fwd_iterator<graph> mln::util::graph::edge_nbh_edge_fwd_iter

Edge centered edge iterators.

8.349.2.3 typedef std::set<edge_data_t> mln::util::graph::edges_set_t

A set to test the presence of a given edge.

8.349.2.4 typedef std::vector<edge_data_t> mln::util::graph::edges_t

The type of the set of edges.

8.349.2.5 typedef mln::internal::vertex_nbh_edge_fwd_iterator<graph> mln::util::graph::vertex_nbh_edge_fwd_iter

Vertex centered edge iterators.

8.349.2.6 typedef mln::internal::vertex_nbh_vertex_fwd_iterator<graph> mln::util::graph::vertex_nbh_vertex_fwd_iter

Vertex centered vertex iterators.

8.349.2.7 typedef std::vector<vertex_data_t> mln::util::graph::vertices_t

The type of the set of vertices.

8.349.3 Constructor & Destructor Documentation

8.349.3.1 mln::util::graph::graph () [inline]

Constructor.

8.349.3.2 mln::util::graph::graph (unsigned *nvertices*) [inline]

Construct a graph with `nvertices` vertices.

8.349.4 Member Function Documentation

8.349.4.1 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)`.

References edge(), and has_v().

Referenced by mln::make::voronoi().

8.349.4.2 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.

References v_nmax().

Referenced by mln::make::voronoi().

8.349.4.3 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.

References v_nmax().

8.349.4.4 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`.

References e_nmax(), has_e(), v1(), v2(), and v_ith_nbh_edge().

8.349.4.5 size_t mln::util::graph::e_nmax () const [inline]

Return the number of edges in the graph.

Referenced by e_ith_nbh_edge(), and edge().

8.349.4.6 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.

References has_v().

8.349.4.7 graph::edge_t mln::util::graph::edge (const edge_id_t & *e*) const [inline]

Return the edge whose id is *e*.

References e_nmax().

Referenced by add_edge().

8.349.4.8 const std::vector< util::ord_pair< vertex_id_t > > & mln::util::graph::edges () const [inline]

Return the list of all edges.

8.349.4.9 bool mln::util::graph::has_e (const edge_id_t & *id_e*) const [inline]

Return whether `id_e` is in the graph.

@{

Referenced by e_ith_nbh_edge(), v1(), and v2().

8.349.4.10 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.

Referenced by add_edge(), edge(), v_ith_nbh_edge(), v_ith_nbh_vertex(), and vertex().

8.349.4.11 void mln::Graph< graph >::invalidate () [inherited]

Invalidate the graph.

FIXME: does nothing!

8.349.4.12 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.

8.349.4.13 bool mln::Graph< graph >::is_valid () const [inherited]

Return true if this graph is valid.

FIXME: currently it always returns true.

8.349.4.14 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`.

References has_e().

Referenced by e_ith_nbh_edge().

8.349.4.15 vertex_id_t mln::util::graph::v2 (const edge_id_t & *id_e*) const [inline]

Return the second vertex associated to edge `id_e`.

References has_e().

Referenced by e_ith_nbh_edge().

8.349.4.16 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`.

References has_v().

Referenced by e_ith_nbh_edge(), and v_ith_nbh_vertex().

8.349.4.17 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`.

References has_v(), and v_ith_nbh_edge().

8.349.4.18 size_t mln::util::graph::v_nmax () const [inline]

Return the number of vertices in the graph.

Referenced by add_vertex(), and add_vertices().

8.349.4.19 graph::vertex_t mln::util::graph::vertex (vertex_id_t *id_v*) const [inline]

Return the vertex whose id is *v*.

References has_v().

8.350 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*?*

8.350.1 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.

8.350.2 Member Function Documentation

8.350.2.1 template<typename I > bool mln::util::greater_point< I >::operator() (const point & *x*, const point & *y*) [inline]

Is *x* greater than *y*?

8.351 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*?*

8.351.1 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.

8.351.2 Member Function Documentation

8.351.2.1 template<typename I > bool mln::util::greater_psite< I >::operator() (const psite & *x*, const psite & *y*) [inline]

Is *x* greater than *y*?

8.352 mln::util::head< T, R > Class Template Reference

Top structure of the soft heap.

`#include <soft_heap.hh>`

8.352.1 Detailed Description

template<typename T, typename R> class mln::util::head< T, R >

Top structure of the soft heap.

8.353 mln::util::ignore Struct Reference

Ignore structure.

`#include <ignore.hh>`

Inheritance diagram for mln::util::ignore:

8.353.1 Detailed Description

Ignore structure.

8.354 mln::util::ilcell< T > Struct Template Reference

Element of an item list. Store the data (key) used in soft_heap.

`#include <soft_heap.hh>`

8.354.1 Detailed Description

template<typename T> struct mln::util::ilcell< T >

Element of an item list. Store the data (key) used in soft_heap.

8.355 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::edge_fwd_iterator< line_graph< G > > edge_fwd_iter

  Edge iterators.

- typedef mln::internal::vertex_nbh_edge_fwd_iterator< line_graph< G > > vertex_nbh_edge_fwd_iter

  Vertex nbh edge iterators.

- typedef mln::internal::vertex_nbh_vertex_fwd_iterator< line_graph< G > > vertex_nbh_vertex_fwd_iter

  Vertex nbh vertex iterators.

Public Member Functions

- template<typename G2 >
  bool has (const util::vertex< G2 > &v) const

  Check whether an edge v exists in the graph.

- bool has_v (const vertex_id_t &id_v) const

  Check whether a vertex id id_v exists in the graph.

- void invalidate ()

  Invalidate the graph.

- bool is_valid () const

  Return true if this graph is valid.

- 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.

- 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`.

- size_t e_nmax () const
  Return the number of edges in the graph.

- edge_t edge (const edge_id_t &e) const
  Edge oriented.

- const G & graph () const
  Return the underlying graph.

- template<typename G2 >
  bool has (const util::edge< G2 > &e) const
  Return whether `e` *is in the graph.*

- bool has_e (const util::edge_id_t &id_e) const
  Return whether `id_e` *is in the graph.*

- 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.

- 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`.

- vertex_t vertex (const vertex_id_t &id_v) const
  Vertex oriented.

8.355.1 Detailed Description

template<typename G> class mln::util::line_graph< G >

Undirected line graph of a graph of type `G`.

8.355.2 Member Typedef Documentation

8.355.2.1 template<typename G> typedef mln::internal::edge_fwd_iterator< line_graph<G> > mln::util::line_graph< G >::edge_fwd_iter

Edge iterators.

8.355.2.2 template<typename G> typedef std::vector<edge_data_t> mln::util::line_graph< G >::edges_t

The type of the set of edges.

8.355.2.3 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.

8.355.2.4 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.

8.355.2.5 template<typename G> typedef std::vector<vertex_data_t> mln::util::line_graph< G >::vertices_t

The type of the set of vertices.

8.355.3 Member Function Documentation

8.355.3.1 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`.

References mln::util::line_graph< G >::e_nmax(), 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_ith_nbh_edge().

8.355.3.2 template<typename G > size_t mln::util::line_graph< G >::e_nmax () const [inline]

Return the number of edges in the graph.

Referenced by mln::util::line_graph< G >::e_ith_nbh_edge(), and mln::util::line_graph< G >::edge().

8.355.3.3 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*.

References mln::util::line_graph< G >::e_nmax().

8.355.3.4 template<typename G > const G & mln::util::line_graph< G >::graph () const [inline]

Return the underlying graph.

8.355.3.5 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 graph.

References mln::util::edge< G >::graph(), mln::util::line_graph< G >::has_e(), and mln::util::edge< G >::id().

8.355.3.6 template<typename G > template<typename G2 > bool mln::util::line_graph< G >::has (const util::vertex< G2 > & *v*) const [inline]

Check whether an edge v exists in the graph.

References mln::util::vertex< G >::graph(), mln::util::line_graph< G >::has_v(), and mln::util::vertex< G >::id().

8.355.3.7 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 graph.

Referenced by mln::util::line_graph< G >::e_ith_nbh_edge(), mln::util::line_graph< G >::has(), mln::util::line_graph< G >::v1(), and mln::util::line_graph< G >::v2().

8.355.3.8 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 graph.

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(), and mln::util::line_graph< G >::vertex().

8.355.3.9 void mln::Graph< line_graph< G > >::invalidate () [inherited]

Invalidate the graph.

FIXME: does nothing!

8.355.3.10 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.

8.355.3.11 bool mln::Graph< line_graph< G > >::is_valid () const [inherited]

Return true if this graph is valid.

FIXME: currently it always returns true.

8.355.3.12 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`.

References mln::util::line_graph< G >::has_e().

Referenced by mln::util::line_graph< G >::e_ith_nbh_edge().

8.355.3.13 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`.

References mln::util::line_graph< G >::has_e().

Referenced by mln::util::line_graph< G >::e_ith_nbh_edge().

8.355.3.14 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`.

References mln::util::line_graph< G >::has_v(), and mln::util::line_graph< G >::v_nmax().

Referenced by mln::util::line_graph< G >::e_ith_nbh_edge(), and mln::util::line_graph< G >::v_ith_nbh_vertex().

8.355.3.15 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`.

References mln::util::line_graph< G >::has_v(), and mln::util::line_graph< G >::v_ith_nbh_edge().

8.355.3.16 template<typename G > size_t mln::util::line_graph< G >::v_nmax () const [inline]

Return the number of vertices in the graph.

Referenced by mln::util::line_graph< G >::v_ith_nbh_edge().

8.355.3.17 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*.

References mln::util::line_graph< G >::has_v().

8.356 mln::util::nil Struct Reference

Nil structure.

`#include <nil.hh>`

Inheritance diagram for mln::util::nil:

8.356.1 Detailed Description

Nil structure.

8.357 mln::util::node< T, R > Class Template Reference

Meta-data of an element in the heap.

`#include <soft_heap.hh>`

8.357.1 Detailed Description

template<typename T, typename R> class mln::util::node< T, R >

Meta-data of an element in the heap.

8.358 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.

8.358.1 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.

8.358.2 Member Typedef Documentation

8.358.2.1 template<typename Tag, typename V> typedef V mln::util::object_id< Tag, V >::value_t

The underlying type id.

8.358.3 Constructor & Destructor Documentation

8.358.3.1 template<typename Tag , typename V > mln::util::object_id< Tag, V >::object_id () [inline]

Constructors.

8.359 mln::util::ord< T > Struct Template Reference

Function-object that defines an ordering between objects with type `T`: *lhs* R *rhs*.

`#include <ord.hh>`

8.359.1 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."

8.360 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.

8.360.1 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.

8.360.2 Member Function Documentation

8.360.2.1 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).

References mln::util::ord_strict(), and mln::util::ord_weak().

8.360.2.2 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).

References mln::util::ord_strict(), and mln::util::ord_weak().

8.360.2.3 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).

References mln::util::ord_strict(), and mln::util::ord_weak().

8.360.2.4 template<typename T > const T & mln::util::ord_pair< T >::first () const [inline]

Get the first (lowest) member of the pair.

8.360.2.5 template<typename T > const T & mln::util::ord_pair< T >::second () const [inline]

Get the second (highest) member of the pair.

8.361 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.

8.361.1 Detailed Description

template<typename I> struct mln::util::pix< I >

Structure pix.

8.361.2 Member Typedef Documentation

8.361.2.1 template<typename I> typedef I ::psite mln::util::pix< I >::psite

Point_Site associated type.

8.361.2.2 template<typename I> typedef I ::value mln::util::pix< I >::value

Value associated type.

8.361.3 Constructor & Destructor Documentation

8.361.3.1 template<typename I > mln::util::pix< I >::pix (const Image< I > & *ima*, const typename I::psite & *p*) [inline]

Constructor.

Parameters:

← ***ima*** The image.

← ***p*** The p_site.

8.361.4 Member Function Documentation

8.361.4.1 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_.

8.361.4.2 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_.

8.361.4.3 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.

8.362 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.*

- template<typename U >
  set< T > & insert (const set< U > &other)

  Insert the elements of `other` *into the set.*

- set< T > & insert (const T &elt)

  Insert an element `elt` *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 `elt` *into the set.*

- set ()

  Constructor without arguments.

- const std::vector< T > & std_vector () const

  Give access to the set elements.

8.362.1 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

8.362.2 Member Typedef Documentation

8.362.2.1 template<typename T> typedef set_bkd_iter<T> mln::util::set< T >::bkd_eiter

Backward iterator associated type.

8.362.2.2 template<typename T> typedef fwd_eiter mln::util::set< T >::eiter

Iterator associated type.

8.362.2.3 template<typename T> typedef T mln::util::set< T >::element

Element associated type.

8.362.2.4 template<typename T> typedef set_fwd_iter<T> mln::util::set< T >::fwd_eiter

Forward iterator associated type.

8.362.3 Constructor & Destructor Documentation

8.362.3.1 template<typename T > mln::util::set< T >::set () [inline]

Constructor without arguments.

8.362.4 Member Function Documentation

8.362.4.1 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

References mln::util::set< T >::is_empty().

8.362.4.2 template<typename T > const T mln::util::set< T >::first_element () const [inline]

Return the first element of the set.

Precondition:

not is_empty()

References mln::util::set< T >::is_empty().

8.362.4.3 template<typename T > bool mln::util::set< T >::has (const T & *elt*) const [inline]

Test if the object `elt` belongs to the set.

Parameters:

← ***elt*** A possible element of the set.

Returns:

True is `elt` is in the set.

8.362.4.4 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:

← ***other*** The set containing the elements to be inserted.

Returns:

The set itself after insertion.

References mln::util::set< T >::is_empty(), and mln::util::set< T >::std_vector().

8.362.4.5 template<typename T > set< T > & mln::util::set< T >::insert (const T & *elt*) [inline]

Insert an element `elt` into the set.

Parameters:

← ***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.

Referenced by mln::p_key< K, P >::change_keys().

8.362.4.6 template<typename T > bool mln::util::set< T >::is_empty () const [inline]

Test if the set is empty.

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().

8.362.4.7 template<typename T > const T mln::util::set< T >::last_element () const [inline]

Return the last element of the set.

Precondition:

not is_empty()

References mln::util::set< T >::is_empty().

8.362.4.8 template<typename T > std::size_t mln::util::set< T >::memory_size () const [inline]

Return the size of this set in memory.

References mln::util::set< T >::nelements().

8.362.4.9 template<typename T > unsigned mln::util::set< T >::nelements () const [inline]

Return the number of elements of the set.

Referenced by mln::util::set< T >::is_empty(), mln::util::set< T >::memory_size(), and mln::util::set< T >::operator[ ]().

8.362.4.10 template<typename T > const T & mln::util::set< T >::operator[ ] (unsigned *i*) const [inline]

Return the i-th element of the set.

Parameters:

← ***i*** Index of the element to retrieve.

Precondition:

i < nelements()

The element is returned by reference and is constant.

References mln::util::set< T >::nelements().

8.362.4.11 template<typename T > set< T > & mln::util::set< T >::remove (const T & *elt*) [inline]

Remove an element `elt` into the set.

Parameters:

← ***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.

8.362.4.12 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.

Referenced by mln::util::set< T >::insert().

8.363 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.

8.363.1 Detailed Description

template<typename P> class mln::util::site_pair< P >

A pair of sites.

It can be used as site.

8.363.2 Member Function Documentation

8.363.2.1 template<typename P > const P & mln::util::site_pair< P >::first () const [inline]

Return the first site.

8.363.2.2 template<typename P > const util::ord_pair< P > & mln::util::site_pair< P >::pair () const [inline]

Return the underlying pair.

8.363.2.3 template<typename P > const P & mln::util::site_pair< P >::second () const [inline]

Return the second site.

8.364 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.

8.364.1 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

8.364.2 Member Typedef Documentation

8.364.2.1 template<typename T, typename R> typedef T mln::util::soft_heap< T, R >::element

Element associated type.

8.364.3 Constructor & Destructor Documentation

8.364.3.1 template<typename T , typename R > mln::util::soft_heap< T, R >::soft_heap (unsigned *r* = 20) [inline]

Default constructor.

A corruption threshold `r` 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.

8.364.3.2 template<typename T , typename R > mln::util::soft_heap< T, R >::~soft_heap () [inline]

Destructor.

8.364.4 Member Function Documentation

8.364.4.1 template<typename T , typename R > void mln::util::soft_heap< T, R >::clear () [inline]

Clear the heap.

8.364.4.2 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.

8.364.4.3 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.

Referenced by mln::util::soft_heap< T, R >::pop_front().

8.364.4.4 template<typename T , typename R > int mln::util::soft_heap< T, R >::nelements () const [inline]

Return the number of element in the heap.

Referenced by mln::util::soft_heap< T, R >::push().

8.364.4.5 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.

References mln::util::soft_heap< T, R >::is_valid().

8.364.4.6 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`.

References mln::util::soft_heap< T, R >::nelements().

8.364.4.7 template<typename T , typename R > void mln::util::soft_heap< T, R >::push (const T & *element*) [inline]

Add a new element `element`.

8.365 mln::util::timer Class Reference

Timer structure.

`#include <timer.hh>`

Inheritance diagram for mln::util::timer:

8.365.1 Detailed Description

Timer structure.

8.366 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 (const tracked_ptr< T > &rhs)

  Copy constructor.

- tracked_ptr ()

  Constructors.

8.366.1 Detailed Description

template<typename T> struct mln::util::tracked_ptr< T >

Smart pointer for shared data with tracking.

8.366.2 Constructor & Destructor Documentation

8.366.2.1 template<typename T > mln::util::tracked_ptr< T >::tracked_ptr () [inline]

Constructors.

8.366.2.2 template<typename T > mln::util::tracked_ptr< T >::tracked_ptr (const tracked_ptr< T > & *rhs*) [inline]

Copy constructor.

8.366.2.3 template<typename T > mln::util::tracked_ptr< T >::~tracked_ptr () [inline]

Destructor.

8.366.3 Member Function Documentation

8.366.3.1 template<typename T > mln::util::tracked_ptr< T >::operator bool () const [inline]

Coercion towards Boolean (for arithmetical tests).

8.366.3.2 template<typename T > bool mln::util::tracked_ptr< T >::operator! () const [inline]

Negation (for arithmetical tests).

8.366.3.3 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.

8.366.3.4 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.

8.366.3.5 template<typename T > tracked_ptr< T > & mln::util::tracked_ptr< T >::operator= (T * *ptr*) [inline]

Assignment.

8.366.3.6 template<typename T > tracked_ptr< T > & mln::util::tracked_ptr< T >::operator= (const tracked_ptr< T > & *rhs*) [inline]

Assignment.

8.367 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 (tree_node< T > *root)
  Constructor.
- tree ()
  Constructor.

8.367.1 Detailed Description

template<typename T> class mln::util::tree< T >

Class of generic tree.

8.367.2 Constructor & Destructor Documentation

8.367.2.1 template<typename T > mln::util::tree< T >::tree () [inline]

Constructor.

8.367.2.2 template<typename T > mln::util::tree< T >::tree (tree_node< T > * *root*) [inline]

Constructor.

Parameters:

← ***root*** The root of the tree.

8.367.3 Member Function Documentation

8.367.3.1 template<typename T > void mln::util::tree< T >::add_tree_down (T & *elt*) [inline]

Bind a new tree downer the current.

Parameters:

← ***elt*** The new value of the new tree_node of the new tree add downer the current.

8.367.3.2 template<typename T > void mln::util::tree< T >::add_tree_up (T & *elt*) [inline]

Bind a new tree upper the current.

Parameters:

← ***elt*** The new value of the new tree_node of the new tree add upper the current.

References mln::util::tree_node< T >::children().

8.367.3.3 template<typename T > bool mln::util::tree< T >::check_consistency () [inline]

Check the consistency of the tree.

Returns:

true if no error, else false.

References mln::util::tree< T >::root().

8.367.3.4 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.

References mln::util::tree< T >::root().

8.367.3.5 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.

Referenced by mln::util::tree< T >::check_consistency(), mln::util::display_tree(), mln::util::tree< T >::main_branch(), and mln::util::tree_to_fast().

8.368 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 (tree_node< T > *tree_node)

  Bind `tree_node` *to the current tree_node and become its child.*

- tree_node< T > * add_child (T elt)

  Create a tree_node with `elt` *which become the child of the current tree_node.*

- bool check_consistency ()

  Check the consistency of the tree_node.

- const children_t & children () const

  The getter of the children.

- children_t & children ()

  The getter of the children.

- tree_node< T > * delete_tree_node ()

  Delete the current tree_node.

- const T & elt () const

  The const getter of the element.

- T & elt ()

  The 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 (T elt)

  Constructor.

- tree_node ()
  Constructor.

8.368.1 Detailed Description

template<typename T> class mln::util::tree_node< T >

Class of generic tree_node for tree.

8.368.2 Constructor & Destructor Documentation

8.368.2.1 template<typename T > mln::util::tree_node< T >::tree_node () [inline]

Constructor.

8.368.2.2 template<typename T > mln::util::tree_node< T >::tree_node (T *elt*) [inline]

Constructor.

Parameters:

← ***elt*** The element of tree_node.

8.368.3 Member Function Documentation

8.368.3.1 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:

← ***tree_node*** The new child tree_node.

Returns:

The child tree_node.

References mln::util::tree_node< T >::children(), and mln::util::tree_node< T >::parent().

8.368.3.2 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:

← ***elt*** The element of the new child to add.

Returns:

The new tree_node created.

8.368.3.3 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.

8.368.3.4 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.

8.368.3.5 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.

Referenced by mln::util::tree_node< T >::add_child(), and mln::util::tree< T >::add_tree_up().

8.368.3.6 template<typename T > tree_node< T > * mln::util::tree_node< T >::delete_tree_node () [inline]

Delete the current tree_node.

8.368.3.7 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.

8.368.3.8 template<typename T > T & mln::util::tree_node< T >::elt () [inline]

The getter of the element.

Returns:

The element of the tree_node.

Referenced by mln::util::tree_node< T >::print().

8.368.3.9 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.

Referenced by mln::util::tree_node< T >::add_child(), mln::util::branch_iter_ind< T >::deepness(), and mln::util::branch_iter< T >::deepness().

8.368.3.10 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:

← ***ostr*** The output stream.

← ***level*** The deep level

References mln::util::tree_node< T >::elt().

8.368.3.11 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:

← ***elt*** The value of the searched tree_node.

Returns:

If not found 0 else the tree_node with elt value.

References mln::util::tree_node< T >::search_rec().

8.368.3.12 template<typename T > int mln::util::tree_node< T >::search_rec (tree_node< T > ** *res*, T & *elt*) [inline]

The using method for method search.

Referenced by mln::util::tree_node< T >::search().

8.368.3.13 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:

← ***parent*** The new parent tree_node.

8.369 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.

8.369.1 Detailed Description

template<typename G> class mln::util::vertex< G >

Vertex of a graph `G`.

8.369.2 Member Typedef Documentation

8.369.2.1 template<typename G> typedef Vertex<void> mln::util::vertex< G >::Category

Object category.

8.369.2.2 template<typename G> typedef G mln::util::vertex< G >::graph_t

Graph associated type.

8.369.2.3 template<typename G> typedef vertex_id_t mln::util::vertex< G >::id_t

The vertex type id.

8.369.2.4 template<typename G> typedef vertex_id_t::value_t mln::util::vertex< G >::id_value_t

The underlying type used to store vertex ids.

8.369.3 Constructor & Destructor Documentation

8.369.3.1 template<typename G > mln::util::vertex< G >::vertex () [inline]

Constructors.

References mln::util::vertex< G >::invalidate().

8.369.4 Member Function Documentation

8.369.4.1 template<typename G > void mln::util::vertex< G >::change_graph (const G & *g*) [inline]

Change the parent graph of that vertex.

8.369.4.2 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.

8.369.4.3 template<typename G > const G & mln::util::vertex< G >::graph () const [inline]

Returns the graph pointer this vertex belongs to.

Referenced by mln::p_vertices< G, F >::has(), mln::util::line_graph< G >::has(), and mln::util::operator==().

8.369.4.4 template<typename G > const vertex_id_t & mln::util::vertex< G >::id () const [inline]

Returns the vertex id.

Referenced by mln::util::line_graph< G >::has(), and mln::util::operator==().

8.369.4.5 template<typename G > void mln::util::vertex< G >::invalidate () [inline]

Invalidate that vertex.

Referenced by mln::util::vertex< G >::vertex().

8.369.4.6 template<typename G > bool mln::util::vertex< G >::is_valid () const [inline]

Check whether the vertex is still part of the graph.

Referenced by mln::p_vertices< G, F >::has().

8.369.4.7 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.

8.369.4.8 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.

8.369.4.9 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 retrived from the initial graph.

8.369.4.10 template<typename G > unsigned mln::util::vertex< G >::nmax_nbh_vertices () const [inline]

Returns the number max of vertices adjacent to this vertex.

8.369.4.11 template<typename G > mln::util::vertex< G >::operator vertex_id_t () const [inline]

Conversion to the vertex id.

FIXME: May cause ambiguities... :(

8.369.4.12 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`.

8.369.4.13 template<typename G > void mln::util::vertex< G >::update_id (const vertex_id_t & *id*) [inline]

Update the vertex id.

8.370 mln::util::yes Struct Reference

Object that always says "yes".

`#include <yes.hh>`

Inheritance diagram for mln::util::yes:

8.370.1 Detailed Description

Object that always says "yes".

8.371 mln::Value< E > Struct Template Reference

Base class for implementation classes of values.

`#include <value.hh>`

Inheritance diagram for mln::Value< E >:

8.371.1 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.

8.372 mln::value::float01 Class Reference

Class for floating values restricted to the interval [0.

`#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 (unsigned nbits, float val)

  Ctor.

- template<unsigned n>
  float01 (const float01_< n > &val)

  Ctor.

- float01 ()

  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.

8.372.1 Detailed Description

Class for floating values restricted to the interval [0.

.1] and discretized with n bits.

8.372.2 Member Typedef Documentation

8.372.2.1 typedef std::pair<unsigned, unsigned long> mln::value::float01::enc

Encoding associated type.

8.372.2.2 typedef float mln::value::float01::equiv

Equivalent associated type.

8.372.3 Constructor & Destructor Documentation

8.372.3.1 mln::value::float01::float01 () [inline]

Ctor.

8.372.3.2 template<unsigned n> mln::value::float01::float01 (const float01_< n > & *val*) [inline]

Ctor.

8.372.3.3 mln::value::float01::float01 (unsigned *nbits*, float *val*) [inline]

Ctor.

8.372.4 Member Function Documentation

8.372.4.1 unsigned mln::value::float01::nbits () const [inline]

Access to the encoding size.

8.372.4.2 mln::value::float01::operator float () const [inline]

Conversion to float.

8.372.4.3 float01 & mln::value::float01::set_nbits (unsigned *nbits*) [inline]

Set the encoding size to nbits.

Referenced by to_nbits().

8.372.4.4 const float01 mln::value::float01::to_nbits (unsigned *nbits*) const [inline]

Return an equivalent gray encoded on `nbits` bits.

References set_nbits().

8.372.4.5 float mln::value::float01::value () const [inline]

Access to std type.

8.372.4.6 unsigned long mln::value::float01::value_ind () const [inline]

Access to the position in the quantized interval.

8.373 mln::value::float01_f Struct Reference

Class for floating values restricted to the interval [0..1].

`#include <float01_f.hh>`

Inherits Floating< float01_f >, and value_like_< float,float,float,float01_f >.

Public Member Functions

- float01_f (float val)
  Constructor from a float.
- float01_f ()
  Constructor without argument.
- operator float () const
  Conversion to a float.
- float01_f & operator= (const float val)
  Assignment from a float.
- float value () const
  Access to float value.

8.373.1 Detailed Description

Class for floating values restricted to the interval [0..1].

8.373.2 Constructor & Destructor Documentation

8.373.2.1 mln::value::float01_f::float01_f () [inline]

Constructor without argument.

8.373.2.2 mln::value::float01_f::float01_f (float *val*) [inline]

Constructor from a float.

8.373.3 Member Function Documentation

8.373.3.1 mln::value::float01_f::operator float () const [inline]

Conversion to a float.

8.373.3.2 float01_f & mln::value::float01_f::operator= (const float *val*) [inline]

Assignment from a float.

8.373.3.3 float mln::value::float01_f::value () const [inline]

Access to float value.

8.374 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

- template<unsigned m>
  graylevel (const graylevel< m > &rhs)

  Constructor from any graylevel.

- graylevel (int val)

  Constructor from int.

- graylevel (const graylevel< n > &rhs)

  Copy constructor.

- graylevel ()

  Constructor without argument.

- template<unsigned m>
  graylevel< n > & operator= (const graylevel< m > &rhs)

  Assigment with any graylevel.

- graylevel< n > & operator= (int val)

  Assigment with int.

- graylevel< n > & operator= (const graylevel< n > &rhs)

  Assigment.

- 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.

8.374.1 Detailed Description

template<unsigned n> struct mln::value::graylevel< n >

General gray-level class on n bits.

8.374.2 Constructor & Destructor Documentation

8.374.2.1 template<unsigned n> mln::value::graylevel< n >::graylevel () [inline]

Constructor without argument.

8.374.2.2 template<unsigned n> mln::value::graylevel< n >::graylevel (const graylevel< n > & *rhs*) [inline]

Copy constructor.

8.374.2.3 template<unsigned n> mln::value::graylevel< n >::graylevel (int *val*) [inline]

Constructor from int.

8.374.2.4 template<unsigned n> template<unsigned m> mln::value::graylevel< n >::graylevel (const graylevel< m > & *rhs*) [inline]

Constructor from any graylevel.

References mln::value::graylevel< n >::value().

8.374.2.5 template<unsigned n> mln::value::graylevel< n >::graylevel (const mln::literal::black_t &) [inline]

Ctors with literals.

8.374.3 Member Function Documentation

8.374.3.1 template<unsigned n> graylevel< n > & mln::value::graylevel< n >::operator= (const mln::literal::black_t &) [inline]

Assigment with literals.

8.374.3.2 template<unsigned n> template<unsigned m> graylevel< n > & mln::value::graylevel< n >::operator= (const graylevel< m > & *rhs*) [inline]

Assigment with any graylevel.

References mln::value::graylevel< n >::value().

8.374.3.3 template<unsigned n> graylevel< n > & mln::value::graylevel< n >::operator= (int *val*) [inline]

Assigment with int.

8.374.3.4 template<unsigned n> graylevel< n > & mln::value::graylevel< n >::operator= (const graylevel< n > & *rhs*) [inline]

Assigment.

8.374.3.5 template<unsigned n> float mln::value::graylevel< n >::to_float () const [inline]

Conversion to float between 0 and 1.

Referenced by mln::value::graylevel_f::graylevel_f(), and mln::value::graylevel_f::operator=().

8.374.3.6 template<unsigned n> unsigned mln::value::graylevel< n >::value () const [inline]

Access to std type.

Referenced by mln::value::graylevel< n >::graylevel(), and mln::value::graylevel< n >::operator=().

8.375 mln::value::graylevel_f Struct Reference

General gray-level class on n bits.

`#include <graylevel_f.hh>`

Inherits mln::value::Floating< graylevel_f >, and value_like_< float01_f,float01_f::enc, internal::gray_f,graylevel_f >.

Public Member Functions

- template<unsigned n>
  graylevel_f (const graylevel< n > &rhs)

  Constructor from graylevel.

- graylevel_f (float val)

  Constructor from float.

- graylevel_f (const graylevel_f &rhs)

  Copy constructor.

- graylevel_f ()

  Constructor without argument.

- template<unsigned n>
  operator graylevel< n > () const

  Conversion to graylevel<n>.

- template<unsigned n>
  graylevel_f & operator= (const graylevel< n > &rhs)

  Assigment with graylevel.

- graylevel_f & operator= (float val)

  Assigment with float.

- graylevel_f & operator= (const graylevel_f &rhs)

  Assigment.

- 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 &)

  Assigment with literals.

8.375.1 Detailed Description

General gray-level class on n bits.

8.375.2 Constructor & Destructor Documentation

8.375.2.1 mln::value::graylevel_f::graylevel_f () [inline]

Constructor without argument.

8.375.2.2 mln::value::graylevel_f::graylevel_f (const graylevel_f & *rhs*) [inline]

Copy constructor.

8.375.2.3 mln::value::graylevel_f::graylevel_f (float *val*) [inline]

Constructor from float.

8.375.2.4 template<unsigned n> mln::value::graylevel_f::graylevel_f (const graylevel< n > & *rhs*) [inline]

Constructor from graylevel.

References mln::value::graylevel< n >::to_float().

8.375.2.5 mln::value::graylevel_f::graylevel_f (const mln::literal::black_t &) [inline]

Ctors with literals.

8.375.3 Member Function Documentation

8.375.3.1 template<unsigned n> mln::value::graylevel_f::operator graylevel< n > () const [inline]

Conversion to graylevel<n>.

8.375.3.2 graylevel_f & mln::value::graylevel_f::operator= (const mln::literal::black_t &) [inline]

Assigment with literals.

8.375.3.3 template<unsigned n> graylevel_f & mln::value::graylevel_f::operator= (const graylevel< n > & *rhs*) [inline]

Assigment with graylevel.

References mln::value::graylevel< n >::to_float().

8.375.3.4 graylevel_f & mln::value::graylevel_f::operator= (float *val*) [inline]

Assigment with float.

8.375.3.5 graylevel_f & mln::value::graylevel_f::operator= (const graylevel_f & *rhs*) [inline]

Assigment.

8.375.3.6 float mln::value::graylevel_f::value () const [inline]

Access to std type.

Referenced by mln::value::operator<<().

8.376 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 (int i)
  Constructor from an integer.
- int_s ()
  Constructor without argument.
- 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.

8.376.1 Detailed Description

template<unsigned n> struct mln::value::int_s< n >

Signed integer value class.

The parameter is `n` the number of encoding bits.

8.376.2 Constructor & Destructor Documentation

8.376.2.1 template<unsigned n> mln::value::int_s< n >::int_s () [inline]

Constructor without argument.

8.376.2.2 template<unsigned n> mln::value::int_s< n >::int_s (int *i*) [inline]

Constructor from an integer.

8.376.2.3 template<unsigned n> mln::value::int_s< n >::int_s (const mln::literal::zero_t &) [inline]

Constructors/assignments with literals.

8.376.3 Member Function Documentation

8.376.3.1 template<unsigned n> mln::value::int_s< n >::operator int () const [inline]

Conversion to an integer.

8.376.3.2 template<unsigned n> int_s< n > & mln::value::int_s< n >::operator= (int *i*) [inline]

Assignment from an integer.

8.376.4 Member Data Documentation

8.376.4.1 template<unsigned n> const int_s< n > mln::value::int_s< n >::one = 1 [inline, static]

Unit value.

8.376.4.2 template<unsigned n> const int_s< n > mln::value::int_s< n >::zero = 0 [inline, static]

Zero value.

8.377 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 (int i)

  Constructor from an integer.

- int_u ()

  Constructor without argument.

- 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.

8.377.1 Detailed Description

template<unsigned n> struct mln::value::int_u< n >

Unsigned integer value class.

The parameter is `n` the number of encoding bits.

8.377.2 Constructor & Destructor Documentation

8.377.2.1 template<unsigned n> mln::value::int_u< n >::int_u () [inline]

Constructor without argument.

8.377.2.2 template<unsigned n> mln::value::int_u< n >::int_u (int *i*) [inline]

Constructor from an integer.

8.377.2.3 template<unsigned n> mln::value::int_u< n >::int_u (const mln::literal::zero_t &) [inline]

Constructors/assignments with literals.

8.377.3 Member Function Documentation

8.377.3.1 template<unsigned n> int_u< n > mln::value::int_u< n >::next () const [inline]

Give the next value (i.e., i + 1).

8.377.3.2 template<unsigned n> mln::value::int_u< n >::operator unsigned () const [inline]

Conversion to an unsigned integer.

8.377.3.3 template<unsigned n> int mln::value::int_u< n >::operator- () const [inline]

Unary operator minus.

8.377.3.4 template<unsigned n> int_u< n > & mln::value::int_u< n >::operator= (int *i*) [inline]

Assignment from an integer.

8.378 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 (int i)
  Constructor from an integer.
- int_u_sat ()
  Constructor without argument.
- 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.

8.378.1 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.

8.378.2 Constructor & Destructor Documentation

8.378.2.1 template<unsigned n> mln::value::int_u_sat< n >::int_u_sat () [inline]

Constructor without argument.

8.378.2.2 template<unsigned n> mln::value::int_u_sat< n >::int_u_sat (int *i*) [inline]

Constructor from an integer.

8.378.3 Member Function Documentation

8.378.3.1 template<unsigned n> mln::value::int_u_sat< n >::operator int () const [inline]

Conversion to an integer.

8.378.3.2 template<unsigned n> int_u_sat< n > & mln::value::int_u_sat< n >::operator+= (int *i*) [inline]

Self addition.

8.378.3.3 template<unsigned n> int_u_sat< n > & mln::value::int_u_sat< n >::operator-= (int *i*) [inline]

Self subtraction.

8.378.3.4 template<unsigned n> int_u_sat< n > & mln::value::int_u_sat< n >::operator= (int *i*) [inline]

Assignment from an integer.

8.378.4 Member Data Documentation

8.378.4.1 template<unsigned n> const int_u_sat< n > mln::value::int_u_sat< n >::one = 1 [inline, static]

Unit value.

8.378.4.2 template<unsigned n> const int_u_sat< n > mln::value::int_u_sat< n >::zero = 0 [inline, static]

Zero value.

8.379 mln::value::Integer< E > Struct Template Reference

Concept of integer.

`#include <integer.hh>`

Inheritance diagram for mln::value::Integer< E >:

8.379.1 Detailed Description

template<typename E> struct mln::value::Integer< E >

Concept of integer.

8.380 mln::value::Integer< void > Struct Template Reference

Category flag type.

`#include <integer.hh>`

8.380.1 Detailed Description

template<> struct mln::value::Integer< void >

Category flag type.

8.381 mln::value::label< n > Struct Template Reference

Label value class.

`#include <label.hh>`

Inherits mln::value::Symbolic< label<n> >, and 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 (const literal::zero_t &v)

  Constructor from literal::zero.

- label (unsigned i)

  Constructor from an (unsigned) integer.

- label ()

  Constructor without argument.

- 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= (const literal::zero_t &v)

  Assignment from literal::zero.

- label< n > & operator= (unsigned i)

  Assignment from an (unsigned) integer.

- label< n > prev () const

  Return the previous value.

8.381.1 Detailed Description

template<unsigned n> struct mln::value::label< n >

Label value class.

The parameter `n` is the number of encoding bits.

8.381.2 Member Typedef Documentation

8.381.2.1 template<unsigned n> typedef internal::encoding_unsigned_<n>::ret mln::value::label< n >::enc

Encoding associated type.

8.381.3 Constructor & Destructor Documentation

8.381.3.1 template<unsigned n> mln::value::label< n >::label () [inline]

Constructor without argument.

8.381.3.2 template<unsigned n> mln::value::label< n >::label (unsigned *i*) [inline]

Constructor from an (unsigned) integer.

8.381.3.3 template<unsigned n> mln::value::label< n >::label (const literal::zero_t & *v*) [inline]

Constructor from literal::zero.

8.381.4 Member Function Documentation

8.381.4.1 template<unsigned n> label< n > mln::value::label< n >::next () const [inline]

Return the next value.

8.381.4.2 template<unsigned n> mln::value::label< n >::operator unsigned () const [inline]

Conversion to an unsigned integer.

8.381.4.3 template<unsigned n> label< n > & mln::value::label< n >::operator++ () [inline]

Self increment.

8.381.4.4 template<unsigned n> label< n > & mln::value::label< n >::operator– () [inline]

Self decrement.

8.381.4.5 template<unsigned n> label< n > & mln::value::label< n >::operator= (const literal::zero_t & *v*) [inline]

Assignment from literal::zero.

8.381.4.6 template<unsigned n> label< n > & mln::value::label< n >::operator= (unsigned *i*) [inline]

Assignment from an (unsigned) integer.

8.381.4.7 template<unsigned n> label< n > mln::value::label< n >::prev () const [inline]

Return the previous value.

8.382 mln::value::lut_vec< S, T > Struct Template Reference

Class that defines FIXME.

`#include <lut_vec.hh>`

Inherits Value_Set< 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 V >
  lut_vec (const S &vset, const Function_v2v< util::array< V > > &f)

  Constructor from a Site_set and any util::array.

- template<typename V >
  lut_vec (const S &vset, const Function_v2v< fun::i2v::array< V > > &f)

  Constructor from a Site_set and any fun::i2v::array.

- template<typename F >
  lut_vec (const S &vset, const Function_v2v< F > &f)

  Constructors
  Constructor from a Site_set and any Function_v2v.

8.382.1 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

8.382.2 Member Typedef Documentation

8.382.2.1 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.

8.382.2.2 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.

8.382.2.3 template<typename S , typename T > typedef T mln::value::lut_vec< S, T >::value

Value associated type.

8.382.3 Constructor & Destructor Documentation

8.382.3.1 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 Site_set and any Function_v2v.

8.382.3.2 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 Site_set and any fun::i2v::array.

8.382.3.3 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 Site_set and any util::array.

References mln::util::array< T >::size(), and mln::util::array< T >::std_vector().

8.382.4 Member Function Documentation

8.382.4.1 template<typename S , typename T > bool mln::value::lut_vec< S, T >::has (const value & *v*) const [inline]

Test if v belongs to this set.

8.382.4.2 template<typename S , typename T > unsigned mln::value::lut_vec< S, T >::index_of (const value & *v*) const [inline]

Give the index of value v in this set.

8.382.4.3 template<typename S , typename T > unsigned mln::value::lut_vec< S, T >::nvalues () const [inline]

Give the number of values.

Referenced by mln::value::lut_vec< S, T >::operator[ ]().

8.382.4.4 template<typename S , typename T > T mln::value::lut_vec< S, T >::operator[ ] (unsigned *i*) const [inline]

Give the i-th value.

References mln::value::lut_vec< S, T >::nvalues().

8.383 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

- operator typename I::value () const
  Conversion (read access); precise version.
- template<typename V >
  operator V () const
  Conversion (read access); general version.
- template<typename II >
  proxy< I > & operator= (const proxy< II > &rhs)

Assignment (write access); with other proxy.

- proxy< I > & operator= (const proxy< I > &rhs)

  Assignment (write access); replacement for default op.

- template<typename V >
  proxy< I > & operator= (const V &v)

  Assignment (write access); general version.

- proxy (I &ima, const typename I::psite &p)

  Constructor.

- I::value to_value () const

  Explicit read access.

- ~proxy ()

  Destructor.

8.383.1 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.

8.383.2 Member Typedef Documentation

8.383.2.1 template<typename I> typedef void mln::value::proxy< I >::enc

Encoding associated type.

8.383.2.2 template<typename I> typedef I ::value mln::value::proxy< I >::equiv

Equivalent associated type.

8.383.3 Constructor & Destructor Documentation

8.383.3.1 template<typename I > mln::value::proxy< I >::proxy (I & *ima*, const typename I::psite & *p*) [inline]

Constructor.

8.383.3.2 template<typename I > mln::value::proxy< I >::~proxy () [inline]

Destructor.

8.383.4 Member Function Documentation

8.383.4.1 template<typename I > mln::value::proxy< I >::operator typename I::value () const [inline]

Conversion (read access); precise version.

8.383.4.2 template<typename I > template<typename V > mln::value::proxy< I >::operator V () const [inline]

Conversion (read access); general version.

8.383.4.3 template<typename I > template<typename II > proxy< I > & mln::value::proxy< I >::operator= (const proxy< II > & *rhs*) [inline]

Assignment (write access); with other proxy.

References mln::value::proxy< I >::operator=(), and mln::value::proxy< I >::to_value().

8.383.4.4 template<typename I > proxy< I > & mln::value::proxy< I >::operator= (const proxy< I > & *rhs*) [inline]

Assignment (write access); replacement for default op.

References mln::value::proxy< I >::operator=(), and mln::value::proxy< I >::to_value().

8.383.4.5 template<typename I > template<typename V > proxy< I > & mln::value::proxy< I >::operator= (const V & *v*) [inline]

Assignment (write access); general version.

Referenced by mln::value::proxy< I >::operator=().

8.383.4.6 template<typename I > I::value mln::value::proxy< I >::to_value () const [inline]

Explicit read access.

Referenced by mln::value::proxy< I >::operator=().

8.384 mln::value::proxy< const I > Class Template Reference

Generic proxy class for an image pixel value.

`#include <proxy.hh>`

Inheritance diagram for mln::value::proxy< const I >:

Public Types

- typedef void enc
  Encoding associated type.
- typedef I::value equiv
  Equivalent associated type.

Public Member Functions

- operator typename I::value () const
  Conversion (read access); precise version.
- template<typename V >
  operator V () const
  Conversion (read access); general version.
- proxy (const I &ima, const typename I::psite &p)
  Constructor.

- I::value to_value () const
  Explicit read access.
- ~proxy ()
  Destructor.

8.384.1 Detailed Description

template<typename I> class mln::value::proxy< const I >

Generic proxy class for an image pixel value.

The parameter `I` is an image type.

8.384.2 Member Typedef Documentation

8.384.2.1 template<typename I > typedef void mln::value::proxy< const I >::enc

Encoding associated type.

8.384.2.2 template<typename I > typedef I ::value mln::value::proxy< const I >::equiv

Equivalent associated type.

8.384.3 Constructor & Destructor Documentation

8.384.3.1 template<typename I > mln::value::proxy< const I >::proxy (const I & *ima*, const typename I::psite & *p*) [inline]

Constructor.

8.384.3.2 template<typename I > mln::value::proxy< const I >::~proxy () [inline]

Destructor.

8.384.4 Member Function Documentation

8.384.4.1 template<typename I > mln::value::proxy< const I >::operator typename I::value () const [inline]

Conversion (read access); precise version.

8.384.4.2 template<typename I > template<typename V > mln::value::proxy< const I >::operator V () const [inline]

Conversion (read access); general version.

8.384.4.3 template<typename I > I::value mln::value::proxy< const I >::to_value () const [inline]

Explicit read access.

8.385 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 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.

8.385.1 Detailed Description

template<unsigned n> struct mln::value::rgb< n >

Color class for red-green-blue where every component is n-bit encoded.

8.385.2 Constructor & Destructor Documentation

8.385.2.1 template<unsigned n> mln::value::rgb< n >::rgb () [inline]

Constructor without argument.

8.385.2.2 template<unsigned n> mln::value::rgb< n >::rgb (int *r*, int *g*, int *b*) [inline]

Constructor from component values.

8.385.2.3 template<unsigned n> mln::value::rgb< n >::rgb (const algebra::vec< 3, int > & *rhs*) [inline]

Constructor from a algebra::vec.

8.385.2.4 template<unsigned n> mln::value::rgb< n >::rgb (const mln::literal::white_t &) [inline]

Constructors with literals.

8.385.3 Member Function Documentation

8.385.3.1 template<unsigned n> rgb< n > & mln::value::rgb< n >::operator= (const rgb< n > & *rhs*) [inline]

Assignment.

8.385.3.2 template<unsigned n> int_u<n> mln::value::rgb< n >::red () const [inline]

Acces to red/green/blue component.

8.385.4 Member Data Documentation

8.385.4.1 template<unsigned n> const rgb< n > mln::value::rgb< n >::zero [inline, static]

Zero value.

8.386 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_quant(T), mln::trait::value::quant::low >::value >.

Static Public Member Functions

- static const set< T > & the ()

  Return a singleton.

8.386.1 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`.

8.386.2 Member Function Documentation

8.386.2.1 template<typename T > const set< T > & mln::value::set< T >::the () [inline, static]

Return a singleton.

8.387 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 (int i)
  Constructor from an integer.
- sign ()
  Constructor without argument.
- 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.

8.387.1 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.

8.387.2 Member Typedef Documentation

8.387.2.1 typedef int mln::value::sign::enc

FIXME Are these typedefs correct?

Define the encoding type

8.387.2.2 typedef int mln::value::sign::equiv

Define the equivalent type.

8.387.3 Constructor & Destructor Documentation

8.387.3.1 mln::value::sign::sign () [inline]

Constructor without argument.

8.387.3.2 mln::value::sign::sign (int *i*) [inline]

Constructor from an integer.

8.387.3.3 mln::value::sign::sign (const mln::literal::zero_t &) [inline]

Constructors/assignments with literals.

8.387.4 Member Function Documentation

8.387.4.1 mln::value::sign::operator int () const [inline]

Conversion to an integer.

8.387.4.2 sign & mln::value::sign::operator= (int *i*) [inline]

Assignment from an integer.

8.387.5 Member Data Documentation

8.387.5.1 const sign mln::value::sign::one = 1 [static]

Unit value.

8.387.5.2 const sign mln::value::sign::zero = 0 [static]

Zero value.

8.388 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.

8.388.1 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)]

8.388.2 Member Typedef Documentation

8.388.2.1 template<unsigned n, typename I> typedef I ::domain_t mln::value::stack_image< n, I >::domain_t

Site_Set associated type.

8.388.2.2 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.

8.388.2.3 template<unsigned n, typename I> typedef I ::psite mln::value::stack_image< n, I >::psite

Point_Site associated type.

8.388.2.4 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).

8.388.2.5 template<unsigned n, typename I> typedef stack_image< n, tag::image_<I> > mln::value::stack_image< n, I >::skeleton

Skeleton.

8.388.2.6 template<unsigned n, typename I> typedef algebra::vec<n, typename I ::value> mln::value::stack_image< n, I >::value

Value associated type.

8.388.3 Constructor & Destructor Documentation

8.388.3.1 template<unsigned n, typename I> mln::value::stack_image< n, I >::stack_image (const algebra::vec< n, I > & *imas*) [inline]

Constructors.

8.388.4 Member Function Documentation

8.388.4.1 template<unsigned n, typename I > bool mln::value::stack_image< n, I >::is_valid () const [inline]

Test if this image has been initialized.

8.388.4.2 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.

8.388.4.3 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.

8.389 mln::Vertex< E > Struct Template Reference

Vertex category flag type.

`#include <vertex.hh>`

8.389.1 Detailed Description

template<typename E> struct mln::Vertex< E >

Vertex category flag type.

8.390 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 graph_elt_neighborhood< G, p_vertices< G, site_function_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_window< G, p_vertices< G, site_function_t > > win_t
  Window type.

Public Member Functions

- rvalue operator() (unsigned v_id) const
  Value accessors/operators overloads.
- vertex_image ()
  Constructors.

8.390.1 Detailed Description

template<typename P, typename V, typename G = util::graph> class mln::vertex_image< P, V, G >

Image based on graph vertices.

8.390.2 Member Typedef Documentation

8.390.2.1 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.

8.390.2.2 template<typename P, typename V, typename G = util::graph> typedef graph_elt_neighborhood<G,p_vertices<G,site_function_t> > mln::vertex_image< P, V, G >::nbh_t

Neighborhood type.

8.390.2.3 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.

8.390.2.4 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.

8.390.2.5 template<typename P, typename V, typename G = util::graph> typedef graph_elt_window<G,p_vertices<G,site_function_t> > mln::vertex_image< P, V, G >::win_t

Window type.

8.390.3 Constructor & Destructor Documentation

8.390.3.1 template<typename P , typename V , typename G > mln::vertex_image< P, V, G >::vertex_image () [inline]

Constructors.

8.390.4 Member Function Documentation

8.390.4.1 template<typename P , typename V , typename G > vertex_image< P, V, G >::rvalue mln::vertex_image< P, V, G >::operator() (unsigned *v_id*) const [inline]

Value accessors/operators overloads.

8.391 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`.

8.391.1 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.

8.391.2 Member Typedef Documentation

8.391.2.1 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.

8.391.2.2 template<typename D, typename W> typedef D mln::w_window< D, W >::dpsite

Dpsite associated type.

8.391.2.3 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.

8.391.2.4 template<typename D, typename W> typedef W mln::w_window< D, W >::weight

Weight associated type.

8.391.3 Constructor & Destructor Documentation

8.391.3.1 template<typename D , typename W > mln::w_window< D, W >::w_window () [inline]

Constructor without argument.

8.391.4 Member Function Documentation

8.391.4.1 template<typename D , typename W > void mln::w_window< D, W >::clear () [inline]

Clear this window.

8.391.4.2 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`.

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().

8.391.4.3 template<typename D , typename W > bool mln::w_window< D, W >::is_symmetric () const [inline]

Test if the window is symmetric.

References mln::w_window< D, W >::sym().

8.391.4.4 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.

8.391.4.5 template<typename D , typename W > void mln::w_window< D, W >::sym () [inline]

Apply a central symmetry to the window.

References mln::w_window< D, W >::insert().

Referenced by mln::w_window< D, W >::is_symmetric().

8.391.4.6 template<typename D , typename W > W mln::w_window< D, W >::w (unsigned *i*) const [inline]

Give the `i-th` weight.

8.391.4.7 template<typename D , typename W > const std::vector< W > & mln::w_window< D, W >::weights () const [inline]

Give access to the vector of weights.

Referenced by mln::w_window< D, W >::operator==().

8.391.4.8 template<typename D , typename W > const mln::window< D > & mln::w_window< D, W >::win () const [inline]

Give the corresponding window.

Referenced by mln::w_window< D, W >::operator==().

8.391.5 Friends And Related Function Documentation

8.391.5.1 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`.

8.391.5.2 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`.

References mln::w_window< D, W >::weights(), and mln::w_window< D, W >::win().

8.392 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`.

8.392.1 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.

8.392.2 Friends And Related Function Documentation

8.392.2.1 template<typename W > W operator- (const Weighted_Window< W > & *rhs*) [related]

Compute the symmetrical weighted window of `rhs`.

8.393 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.

8.393.1 Detailed Description

Diagonal line window defined on the 2D square grid.

An backdiag2d is centered and symmetric. its width (length) is odd.

For instance:

```
*  o
*    o
*      x
*        o
*          o
*
```

is defined with length = 5.

8.393.2 Constructor & Destructor Documentation

8.393.2.1 mln::win::backdiag2d::backdiag2d (unsigned *length*) [inline]

Constructor.

Parameters:

← ***length*** Length, thus width, of the diagonal line.

Precondition:

`length` is odd.

8.393.3 Member Function Documentation

8.393.3.1 unsigned mln::win::backdiag2d::length () const [inline]

Give the diagonal length, that is, its width.

8.394 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.

8.394.1 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.

8.394.2 Constructor & Destructor Documentation

8.394.2.1 template<typename G , typename C > mln::win::ball< G, C >::ball (unsigned *diameter*) [inline]

Constructor.

Parameters:

← ***diameter*** Diameter of the ball.

Precondition:

`diameter` is odd.

References mln::literal::origin.

8.394.3 Member Function Documentation

8.394.3.1 template<typename G , typename C > unsigned mln::win::ball< G, C >::diameter () const [inline]

Give the ball diameter.

8.395 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.

8.395.1 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.

8.395.2 Constructor & Destructor Documentation

8.395.2.1 mln::win::cube3d::cube3d (unsigned *length*) `[inline]`

Constructor.

Parameters:

← ***length*** Length, thus height, of the cube3d.

Precondition:

`length` is odd.

8.395.3 Member Function Documentation

8.395.3.1 unsigned mln::win::cube3d::length () const [inline]

Give the cube length, that is, its height.

8.396 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.

8.396.1 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

8.396.2 Constructor & Destructor Documentation

8.396.2.1 mln::win::cuboid3d::cuboid3d (unsigned *depth*, unsigned *height*, unsigned *width*) [inline]

Constructor.

Parameters:

← ***depth*** The depth of the cuboid3d.

← ***height*** The height of the cuboid3d.

← ***width*** The width of the cuboid3d.

Precondition:

Argument *depth*, *height* and *width* must be odd.

8.396.3 Member Function Documentation

8.396.3.1 unsigned mln::win::cuboid3d::depth () const [inline]

Accessors.

Return the depth of the cuboid.

8.396.3.2 unsigned mln::win::cuboid3d::height () const [inline]

Return the height of the cuboid.

8.396.3.3 unsigned mln::win::cuboid3d::volume () const [inline]

Return the volume of the cuboid.

8.396.3.4 unsigned mln::win::cuboid3d::width () const [inline]

Return the width of the cuboid.

8.397 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.

8.397.1 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
*
```

is defined with length = 5.

8.397.2 Constructor & Destructor Documentation

8.397.2.1 mln::win::diag2d::diag2d (unsigned *length*) [inline]

Constructor.

Parameters:

← ***length*** Length, thus width, of the diagonal line.

Precondition:

`length` is odd.

8.397.3 Member Function Documentation

8.397.3.1 unsigned mln::win::diag2d::length () const [inline]

Give the diagonal length, that is, its width.

8.398 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.

8.398.1 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.

8.398.2 Member Enumeration Documentation

8.398.2.1 template<typename M , unsigned i, typename C > anonymous enum

Direction.

8.398.3 Constructor & Destructor Documentation

8.398.3.1 template<typename M , unsigned i, typename C > mln::win::line< M, i, C >::line (unsigned *length*) [inline]

Constructor.

Parameters:

← ***length*** Length of the line.

Precondition:

`length` is odd.

References mln::dpoint< G, C >::set_all().

8.398.4 Member Function Documentation

8.398.4.1 template<typename M , unsigned i, typename C > unsigned mln::win::line< M, i, C >::length () const [inline]

Give the line length.

8.398.4.2 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.

8.399 mln::win::multiple< W, F > Class Template Reference

Multiple window.

`#include <multiple.hh>`

Inherits window_base< W::dpsite, multiple< W, F > >.

8.399.1 Detailed Description

template<typename W, typename F> class mln::win::multiple< W, F >

Multiple window.

8.400 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 > >.

8.400.1 Detailed Description

template<unsigned n, typename W, typename F> class mln::win::multiple_size< n, W, F >

Definition of a multiple-size window.

8.401 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.

8.401.1 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 >= 0.

For instance:

```
*     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
*
```

is defined with L = 7 (l = 1).

8.401.2 Constructor & Destructor Documentation

8.401.2.1 mln::win::octagon2d::octagon2d (unsigned *length*) [inline]

Constructor.

Parameters:

← ***length*** Length, of the octagon.

Precondition:

`length` is such as length = 6∗x + 1 where x >= 0.

8.401.3 Member Function Documentation

8.401.3.1 unsigned mln::win::octagon2d::area () const [inline]

Give the area.

8.401.3.2 unsigned mln::win::octagon2d::length () const [inline]

Give the octagon length, that is, its width.

8.402 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.

8.402.1 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.

8.402.2 Constructor & Destructor Documentation

8.402.2.1 mln::win::rectangle2d::rectangle2d (unsigned *height*, unsigned *width*) [inline]

Constructor.

Parameters:

← ***height*** Height of the rectangle2d.
← ***width*** Width of the rectangle2d.

Precondition:

Height and width are odd.

8.402.3 Member Function Documentation

8.402.3.1 unsigned mln::win::rectangle2d::area () const [inline]

Give the rectangle area.

8.402.3.2 unsigned mln::win::rectangle2d::height () const [inline]

Give the rectangle height.

8.402.3.3 const std::vector< dpoint2d > & mln::win::rectangle2d::std_vector () const [inline]

Give the std vector of delta-points.

8.402.3.4 unsigned mln::win::rectangle2d::width () const [inline]

Give the rectangle width.

8.403 mln::Window< E > Struct Template Reference

Base class for implementation classes that are windows.

`#include <window.hh>`

Inheritance diagram for mln::Window< E >:

8.403.1 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.

8.404 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.

- template<typename W >
  window< D > & insert (const Window< W > &win)

  Insert another window `win`.

- window< D > & insert (const D &dp)

  Insert a delta-point `dp`.

- 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`.

8.404.1 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.

8.404.2 Member Typedef Documentation

8.404.2.1 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.

8.404.2.2 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.

8.404.2.3 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.

8.404.2.4 template<typename D> typedef window<D> mln::window< D >::regular

Regular window associated type.

8.404.3 Constructor & Destructor Documentation

8.404.3.1 template<typename D > mln::window< D >::window () [inline]

Constructor without argument.

The constructed window is empty.

8.404.4 Member Function Documentation

8.404.4.1 template<typename D > void mln::window< D >::clear () [inline]

Clear the window.

8.404.4.2 template<typename D > unsigned mln::window< D >::delta () const [inline]

Give the maximum coordinate gap between the window center and a window point.

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

8.404.4.3 template<typename D > const D & mln::window< D >::dp (unsigned *i*) const [inline]

Give the `i-th` delta-point.

References mln::window< D >::size().

Referenced by mln::window< D >::delta(), and mln::window< D >::insert().

8.404.4.4 template<typename D > bool mln::window< D >::has (const D & *dp*) const [inline]

Test if `dp` is in this window definition.

8.404.4.5 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.

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

8.404.4.6 template<typename D > template<typename W > window< D > & mln::window< D >::insert (const Window< W > & *win*) [inline]

Insert another window `win`.

8.404.4.7 template<typename D > window< D > & mln::window< D >::insert (const D & *dp*) [inline]

Insert a delta-point `dp`.

Referenced by mln::c18(), mln::c26(), 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().

8.404.4.8 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.

References mln::literal::zero.

8.404.4.9 template<typename D > bool mln::window< D >::is_empty () const [inline]

Test if the window is empty (null size; no delta-point).

8.404.4.10 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.

References mln::window< D >::sym().

8.404.4.11 template<typename D > void mln::window< D >::print (std::ostream & *ostr*) const [inline]

Print the window definition into `ostr`.

8.404.4.12 template<typename D > unsigned mln::window< D >::size () const [inline]

Give the window size, i.e., the number of delta-sites.

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().

8.404.4.13 template<typename D > const std::vector< D > & mln::window< D >::std_vector () const [inline]

Give the std vector of delta-points.

8.404.4.14 template<typename D > void mln::window< D >::sym () [inline]

Apply a central symmetry to the target window.

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

Referenced by mln::window< D >::is_symmetric().

8.404.5 Friends And Related Function Documentation

8.404.5.1 template<typename D > bool operator== (const window< D > & *lhs*, const window< D > & *rhs*) [related]

Equality comparison between windows `lhs` and `rhs`.

8.405 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:

8.405.1 Detailed Description

Functor returning whether a site is a separator in an inter-pixel image.

Index

~proxy
  mln::value::proxy, 1201
  mln::value::proxy< const I >, 1204
~soft_heap
  mln::util::soft_heap, 1156
~tracked_ptr
  mln::util::tracked_ptr, 1160
_1
  mln::algebra::h_mat, 579
1D neighborhoods, 80
1D windows, 94
2D neighborhoods, 81
2D windows, 95
3D neighborhoods, 83
3D windows, 98

a_point_of
  mln, 130
abs
  mln::data, 193
  mln::math, 361
abs_inplace
  mln::data, 193
Accumulators, 74
add
  mln::topo::n_faces_set, 1085
add_child
  mln::util::tree_node, 1164
add_edge
  mln::util::graph, 1125
add_face
  mln::topo::complex, 1063
add_location
  mln::geom::complex_geometry, 780
add_tree_down
  mln::util::tree, 1162
add_tree_up
  mln::util::tree, 1162
add_vertex
  mln::util::graph, 1125
add_vertices
  mln::util::graph, 1125
addr
  mln::topo::complex, 1063
adj_higher_dim_connected_n_face_bkd_iter
  mln::topo::adj_higher_dim_connected_n_-face_bkd_iter, 1032
adj_higher_dim_connected_n_face_fwd_iter
  mln::topo::adj_higher_dim_connected_n_-face_fwd_iter, 1034
adj_higher_face_bkd_iter
  mln::topo::adj_higher_face_bkd_iter, 1036
adj_higher_face_fwd_iter
  mln::topo::adj_higher_face_fwd_iter, 1037
adj_lower_dim_connected_n_face_bkd_iter
  mln::topo::adj_lower_dim_connected_n_-face_bkd_iter, 1038
adj_lower_dim_connected_n_face_fwd_iter
  mln::topo::adj_lower_dim_connected_n_-face_fwd_iter, 1040
adj_lower_face_bkd_iter
  mln::topo::adj_lower_face_bkd_iter, 1042
adj_lower_face_fwd_iter
  mln::topo::adj_lower_face_fwd_iter, 1043
adj_lower_higher_face_bkd_iter
  mln::topo::adj_lower_higher_face_bkd_iter, 1044
adj_lower_higher_face_fwd_iter
  mln::topo::adj_lower_higher_face_fwd_iter, 1045
adj_m_face_bkd_iter
  mln::topo::adj_m_face_bkd_iter, 1046
adj_m_face_fwd_iter
  mln::topo::adj_m_face_fwd_iter, 1048
adjacency_matrix
  mln::util::adjacency_matrix, 1098
adjust
  mln::border, 174
  mln::extension, 231, 232
adjust_duplicate
  mln::extension, 232
adjust_fill
  mln::extension, 232
algebraic_face
  mln::topo::algebraic_face, 1051, 1052
algebraic_n_face
  mln::topo::algebraic_n_face, 1055
and_inplace
  mln::logical, 334
and_not

mln::logical, 334
and_not_inplace
mln::logical, 335
apex
mln::util::branch, 1106
append
mln::p_array, 905
mln::util::array, 1102
apply
mln::data, 193
apply_p2p
mln, 130
area
mln::accu::site_set::rectangularity, 540
mln::morpho::attribute::sharpness, 890
mln::morpho::attribute::volume, 894
mln::win::octagon2d, 1234
mln::win::rectangle2d, 1236
argument
mln::accu::shape::height, 536
mln::accu::shape::volume, 538
mln::doc::Accumulator, 642
array
mln::util::array, 1102
at
mln::opt, 401, 402
attachment
mln::make, 343

backdiag2d
mln::win::backdiag2d, 1222
background
mln::labeling, 307
ball
mln::win::ball, 1223
base_level
mln::morpho::attribute::height, 888
Basic types, 68, 87
bbox
mln::accu::site_set::rectangularity, 540
mln::Box, 596
mln::box, 591
mln::doc::Box, 645
mln::doc::Fastest_Image, 653
mln::doc::Image, 662
mln::geom, 256, 257
mln::image1d, 820
mln::image2d, 825
mln::image3d, 833
mln::labeled_image, 840
mln::p_line2d, 942
mln::p_run, 969
bbox_t
mln::labeled_image, 840
bboxes
mln::labeled_image, 840
before
mln, 141
begin
mln::p_line2d, 942
bin_2complex_image3df
mln, 127
binarization
mln::binarization, 173
bkd_citer
mln::topo::complex, 1063
bkd_eiter
mln::util::array, 1101
mln::util::set, 1149
bkd_niter
mln::doc::Neighborhood, 667
mln::graph_elt_neighborhood, 791
mln::graph_elt_neighborhood_if, 793
mln::neighb, 897
bkd_piter
mln::box, 590
mln::doc::Box, 645
mln::doc::Fastest_Image, 651
mln::doc::Image, 661
mln::doc::Site_Set, 679
mln::hexa, 811
mln::image2d_h, 829
mln::p_array, 904
mln::p_centered, 909
mln::p_complex, 912
mln::p_edges, 916
mln::p_faces, 921
mln::p_if, 926
mln::p_image, 929
mln::p_key, 937
mln::p_line2d, 941
mln::p_mutable_array_of, 945
mln::p_priority, 953
mln::p_queue, 959
mln::p_queue_fast, 963
mln::p_run, 968
mln::p_set, 973
mln::p_transformed, 977
mln::p_vaccess, 982
mln::p_vertices, 987
bkd_pixter1d
mln::bkd_pixter1d, 582
bkd_pixter2d
mln::bkd_pixter2d, 584
bkd_pixter3d
mln::bkd_pixter3d, 586
bkd_qiter
mln::doc::Weighted_Window, 685

mln::doc::Window, 687
mln::graph_elt_window, 796
mln::graph_elt_window_if, 800
mln::w_window, 1218
mln::window, 1239
bkd_viter
mln::doc::Value_Set, 683
mln::value::lut_vec, 1198
black
mln::literal, 331
blobs
mln::labeling, 308
blue
mln::literal, 331
border
mln::doc::Fastest_Image, 653
mln::image1d, 820
mln::image2d, 825
mln::image3d, 833
box
mln::box, 591
mln::draw, 227
box1d
mln, 127
mln::make, 344
box2d
mln, 127
mln::make, 344, 345
box2d_h
mln, 127
mln::make, 345
box3d
mln, 127
mln::make, 346
box_runstart_piter
mln::box_runstart_piter, 598
branch
mln::util::branch, 1106
brown
mln::literal, 331
buffer
mln::doc::Fastest_Image, 653
mln::image1d, 820
mln::image2d, 825
mln::image3d, 833

c18
modneighb3d, 83
c2
modneighb1d, 80
c26
modneighb3d, 84
c2_col
modneighb2d, 81
c2_row
modneighb2d, 81
c4
modneighb2d, 82
c4_3d
modneighb3d, 84
c6
modneighb3d, 85
c8
modneighb2d, 82
c8_3d
modneighb3d, 85
can_stop
mln::accu::logic::land_basic, 473
mln::accu::logic::lor_basic, 477
Canvas, 76
card
mln::set, 409
cast
mln::value, 448
Category
mln::util::vertex, 1169
category
mln::util::edge, 1116
cell
mln::make, 347
center
mln::box, 591
mln::p_centered, 910
center_only_iter
mln::topo::center_only_iter, 1058
center_val
mln::dpoints_bkd_pixter, 696
mln::dpoints_fwd_pixter, 699
centered_bkd_iter_adapter
mln::topo::centered_bkd_iter_adapter, 1060
centered_fwd_iter_adapter
mln::topo::centered_fwd_iter_adapter, 1061
chamfer
mln::geom, 257
change
mln::p_array, 905
change_both
mln::util::couple, 1113
mln::util::ord_pair, 1144
change_extension
mln::extension_val, 717
change_first
mln::util::couple, 1113
mln::util::ord_pair, 1144
change_graph
mln::util::edge, 1117
mln::util::vertex, 1169
change_key

mln::p_key, 937
change_keys
mln::p_key, 938
change_mask
mln::graph_elt_window_if, 802
change_second
mln::util::couple, 1113
mln::util::ord_pair, 1144
change_target
mln::complex_psite, 631
mln::p_transformed_piter, 980
change_to
mln::pixel, 992
change_window
mln::graph_elt_neighborhood, 791
mln::graph_elt_neighborhood_if, 793
mln::neighb, 897
check_consistency
mln::util::tree, 1162
mln::util::tree_node, 1164
children
mln::util::tree_node, 1165
clear
mln::p_array, 906
mln::p_image, 930
mln::p_key, 938
mln::p_mutable_array_of, 946
mln::p_priority, 954
mln::p_queue, 960
mln::p_queue_fast, 964
mln::p_set, 974
mln::util::array, 1102
mln::util::fibonacci_heap, 1120
mln::util::set, 1149
mln::util::soft_heap, 1156
mln::w_window, 1219
mln::window, 1240
closing
mln::morpho::elementary, 378
colorize
mln::labeling, 308
complementation
mln::morpho, 369
complementation_inplace
mln::morpho, 369
complex
mln::topo::complex, 1063
Complex based, 89
complex_geometry
mln::geom::complex_geometry, 779
complex_image
mln::complex_image, 624
complex_neighborhood_bkd_piter
mln::complex_neighborhood_bkd_piter, 627
complex_neighborhood_fwd_piter
mln::complex_neighborhood_fwd_piter, 629
complex_psite
mln::complex_psite, 631
complex_window_bkd_piter
mln::complex_window_bkd_piter, 634
complex_window_fwd_piter
mln::complex_window_fwd_piter, 636
compose
mln, 130
composed
mln::fun::x2x::composed, 753
compute
mln::accu, 144
mln::data, 193
mln::graph, 266
mln::histo, 270
mln::labeling, 309–311
mln::labeling::impl::generic, 319, 320
mln::set, 409
compute_attribute_image_from
mln::morpho::tree, 387
compute_has
mln::p_queue_fast, 964
compute_image
mln::labeling, 311, 312
compute_parent
mln::morpho::tree, 387
compute_with_weights
mln::set, 410
contrast
mln::morpho, 369
convert
mln::data, 194
mln::data::impl::generic, 209
convolve
mln::linear::local, 326
coord
mln::def, 221
mln::doc::Dpoint, 647
mln::doc::Fastest_Image, 651
mln::doc::Image, 661
mln::doc::Point_Site, 673
mln::dpoint, 690
mln::point, 1000
coordf
mln::def, 221
couple
mln::make, 347
cplx
mln::p_complex, 913
mln::p_faces, 922
mln::topo::algebraic_face, 1052
mln::topo::algebraic_n_face, 1056

mln::topo::face, 1066
mln::topo::n_face, 1078
crop_wrt
mln::box, 592
cube3d
mln::win::cube3d, 1224
cuboid3d
mln::win::cuboid3d, 1227
cyan
mln::literal, 331

D
mln::topo::is_simple_cell, 1076
dark_gray
mln::literal, 331
data
mln::topo::algebraic_face, 1052
mln::topo::algebraic_n_face, 1056
mln::topo::face, 1067
mln::topo::n_face, 1078
dec_face_id
mln::topo::algebraic_face, 1052
mln::topo::algebraic_n_face, 1056
mln::topo::face, 1067
mln::topo::n_face, 1078
dec_n
mln::topo::algebraic_face, 1052
mln::topo::face, 1067
decorated_image
mln::decorated_image, 638
decoration
mln::decorated_image, 638
deepness
mln::util::branch_iter, 1108
mln::util::branch_iter_ind, 1110
delete_tree_node
mln::util::tree_node, 1165
delta
mln::doc::Weighted_Window, 686
mln::geom, 257
mln::graph_elt_window, 797
mln::graph_elt_window_if, 802
mln::graph_window_base, 805
mln::point, 1000
mln::window, 1240
delta_index
mln::doc::Fastest_Image, 653
mln::image1d, 820
mln::image2d, 825
mln::image3d, 834
depth
mln::win::cuboid3d, 1227
detach
mln::topo, 421
detachment
mln::make, 347
diag2d
mln::win::diag2d, 1228
diameter
mln::win::ball, 1223
diff
mln::Site_Set, 1021
mln::win, 455
diff_abs
mln::arith, 161
dilation
mln::morpho, 369
dim
mln::complex_image, 625
mln::doc::Dpoint, 648
mln::doc::Point_Site, 674
mln::dpoint, 691
mln::point, 1001
direct
mln::morpho::tree::filter, 391
discrete_plane_1complex_geometry
mln, 127
discrete_plane_2complex_geometry
mln, 127
disk2d
modwin2d, 96
display_branch
mln::util, 437
display_tree
mln::util, 437
distance_and_closest_point_geodesic
mln::transform, 430
distance_and_influence_zone_geodesic
mln::transform, 431
distance_front
mln::canvas, 180
mln::transform, 431
distance_geodesic
mln::canvas, 180
mln::transform, 431
div
mln::arith, 161
div_cst
mln::arith, 161
div_inplace
mln::arith, 162
domain
mln::complex_image, 625
mln::doc::Fastest_Image, 653
mln::doc::Image, 662
mln::extended, 709
mln::flat_image, 720
mln::hexa, 812

mln::image1d, 820
mln::image2d, 826
mln::image2d_h, 830
mln::image3d, 834
mln::lazy_image, 843
mln::p2p_image, 902
mln::sub_image, 1026
mln::sub_image_if, 1028
mln::tr_image, 1093
mln::transformed_image, 1095
mln::unproject_image, 1096
Domain morphers, 71
domain_t
mln::value::stack_image, 1212
dp
mln::window, 1240
dpoint
mln::doc::Dpoint, 647
mln::doc::Fastest_Image, 651
mln::doc::Image, 661
mln::doc::Neighborhood, 667
mln::doc::Point_Site, 673
mln::doc::Weighted_Window, 685
mln::dpoint, 691, 692
dpoint1d
mln, 127
dpoint2d
mln, 128
dpoint2d_h
mln, 128
mln::make, 348
dpoint3d
mln, 128
dpoints_bkd_pixter
mln::dpoints_bkd_pixter, 695, 696
dpoints_fwd_pixter
mln::dpoints_fwd_pixter, 698, 699
dpsite
mln::point, 1000
mln::w_window, 1218
dpsites_bkd_piter
mln::dpsites_bkd_piter, 701
dpsites_fwd_piter
mln::dpsites_fwd_piter, 703
draw_graph
mln::debug, 216, 217
dummy_p_edges
mln::make, 348
dummy_p_vertices
mln::make, 348, 349
duplicate
mln, 130
mln::border, 175
mln::extension, 232
e_ith_nbh_edge
mln::util::graph, 1125
mln::util::line_graph, 1135
e_nmax
mln::util::graph, 1125
mln::util::line_graph, 1135
edge
mln::p_edges, 916
mln::topo, 421
mln::util::edge, 1116
mln::util::graph, 1125, 1126
mln::util::line_graph, 1135
edge_fwd_iter
mln::util::graph, 1124
mln::util::line_graph, 1134
edge_image
mln::edge_image, 707
mln::make, 349, 350
edge_nbh_edge_fwd_iter
mln::util::graph, 1124
edge_with
mln::util::vertex, 1169
edges
mln::util::graph, 1126
edges_set_t
mln::util::graph, 1124
edges_t
mln::util::graph, 1124
mln::util::line_graph, 1134
eiter
mln::util::array, 1101
mln::util::set, 1149
element
mln::box, 590
mln::graph_window_if_piter, 807
mln::graph_window_piter, 809
mln::image1d, 821
mln::image2d, 826
mln::image3d, 834
mln::p_array, 904
mln::p_centered, 909
mln::p_complex, 912
mln::p_edges, 917
mln::p_faces, 921
mln::p_if, 926
mln::p_image, 929
mln::p_key, 937
mln::p_line2d, 941
mln::p_mutable_array_of, 945
mln::p_priority, 953
mln::p_queue, 959
mln::p_queue_fast, 963
mln::p_run, 968
mln::p_set, 973

mln::p_transformed, 977
mln::p_vaccess, 982
mln::p_vertices, 987
mln::util::array, 1101
mln::util::set, 1149
mln::util::soft_heap, 1156
elt
mln::util::tree_node, 1165
enc
mln::value::float01, 1175
mln::value::label, 1195
mln::value::proxy, 1201
mln::value::proxy< const I >, 1204
mln::value::sign, 1210
end
mln::p_line2d, 942
mln::p_run, 969
enlarge
mln::box, 592
equalize
mln::border, 175
equiv
mln::value, 448
mln::value::float01, 1175
mln::value::proxy, 1201
mln::value::proxy< const I >, 1204
mln::value::sign, 1210
erosion
mln::morpho, 369
exists_key
mln::p_key, 938
exists_priority
mln::p_priority, 954
extend
mln, 131
extended
mln::extended, 709
extension
mln::extension_fun, 711
mln::extension_ima, 714
mln::extension_val, 717
extension_fun
mln::extension_fun, 711
extension_ima
mln::extension_ima, 714
extension_val
mln::extension_val, 717

f_hsi_to_rgb_3x8
mln::fun::v2v, 245
f_hsl_to_rgb_3x8
mln::fun::v2v, 245
f_rgb_to_hsi_f
mln::fun::v2v, 245
f_rgb_to_hsl_f
mln::fun::v2v, 245
face
mln::complex_psite, 631
mln::topo::face, 1066
face_bkd_iter
mln::topo::face_bkd_iter, 1069
face_fwd_iter
mln::topo::face_fwd_iter, 1071
face_id
mln::complex_psite, 631
mln::topo::algebraic_face, 1052
mln::topo::algebraic_n_face, 1056
mln::topo::face, 1067
mln::topo::n_face, 1078
faces
mln::topo::n_faces_set, 1085
faces_type
mln::topo::n_faces_set, 1085
fast_median
mln::data, 194
fibonacci_heap
mln::util::fibonacci_heap, 1120
filename
mln::debug, 217
fill
mln::border, 175
mln::data, 195
mln::extension, 232
mln::util::array, 1103
fill_holes
mln::labeling, 313
fill_with_image
mln::data, 195
mln::data::impl::generic, 209
fill_with_value
mln::data, 195
mln::data::impl::generic, 209
filter
mln::morpho::tree::filter, 391
find
mln::border, 176
first
mln::util::couple, 1113
mln::util::ord_pair, 1144
mln::util::site_pair, 1153
first_element
mln::util::set, 1149
flat_image
mln::flat_image, 720
flat_zones
mln::labeling, 313
float01
mln::value::float01, 1175

float01_16
mln::value, 446
float01_8
mln::value, 446
float01_f
mln::value::float01_f, 1177
float_2complex_image3df
mln, 128
flooding
mln::morpho::watershed, 394, 395
foreground
mln::labeling, 313
format
mln::debug, 217
from_to
mln::convert, 187
front
mln::p_priority, 954
mln::p_queue, 960
mln::p_queue_fast, 964
mln::util::fibonacci_heap, 1120
fun
mln::p2p_image, 902
fun_image
mln::fun_image, 763
fun_t
mln::p_edges, 917
mln::p_vertices, 987
Function
mln::Function, 764
function
mln::p_edges, 918
mln::p_transformed, 978
mln::p_vertices, 989
Functions, 77
fwd_citer
mln::topo::complex, 1063
fwd_eiter
mln::util::array, 1101
mln::util::set, 1149
fwd_niter
mln::doc::Neighborhood, 668
mln::graph_elt_neighborhood, 791
mln::graph_elt_neighborhood_if, 793
mln::neighb, 897
fwd_piter
mln::box, 590
mln::doc::Box, 645
mln::doc::Fastest_Image, 651
mln::doc::Image, 661
mln::doc::Site_Set, 679
mln::hexa, 811
mln::image2d_h, 829
mln::p_array, 905
mln::p_centered, 909
mln::p_complex, 912
mln::p_edges, 917
mln::p_faces, 921
mln::p_if, 926
mln::p_image, 929
mln::p_key, 937
mln::p_line2d, 941
mln::p_mutable_array_of, 945
mln::p_priority, 954
mln::p_queue, 959
mln::p_queue_fast, 964
mln::p_run, 968
mln::p_set, 973
mln::p_transformed, 977
mln::p_vaccess, 982
mln::p_vertices, 987
fwd_pixter1d
mln::fwd_pixter1d, 770
fwd_pixter2d
mln::fwd_pixter2d, 772
fwd_pixter3d
mln::fwd_pixter3d, 774
fwd_qiter
mln::doc::Weighted_Window, 685
mln::doc::Window, 687
mln::graph_elt_window, 796
mln::graph_elt_window_if, 800
mln::w_window, 1218
mln::window, 1239
fwd_viter
mln::doc::Value_Set, 683
mln::value::lut_vec, 1198

gaussian
mln::linear, 322
gaussian_1st_derivative
mln::linear, 322
gaussian_2nd_derivative
mln::linear, 323
gaussian_subsampling
mln::subsampling, 412
general
mln::morpho, 369
geom
mln::complex_image, 624
mln::p_complex, 913
get
mln::border, 176
mln::set, 411
get_rot
mln::registration, 406
gl16
mln::value, 446

gl8
  mln::value, 446
glf
  mln::value, 446
gradient
  mln::morpho, 370
gradient_external
  mln::morpho, 370
gradient_internal
  mln::morpho, 370
graph
  mln::p_edges, 918
  mln::p_graph_piter, 923
  mln::p_vertices, 989
  mln::util::edge, 1117
  mln::util::graph, 1124
  mln::util::line_graph, 1135
  mln::util::vertex, 1169
Graph based, 88
graph_element
  mln::p_edges, 917
  mln::p_vertices, 987
graph_elt_neighborhood_if
  mln::graph_elt_neighborhood_if, 793
graph_elt_window_if
  mln::graph_elt_window_if, 801
graph_t
  mln::edge_image, 706
  mln::p_edges, 917
  mln::p_vertices, 987
  mln::util::edge, 1116
  mln::util::vertex, 1169
  mln::vertex_image, 1215
graph_window_if_piter
  mln::graph_window_if_piter, 807
graph_window_piter
  mln::graph_window_piter, 808
Graphes, 66
graylevel
  mln::value::graylevel, 1180
graylevel_f
  mln::value::graylevel_f, 1183
green
  mln::literal, 331
grid
  mln::dpoint, 690
  mln::point, 1000

h_mat
  mln::algebra::h_mat, 578
  mln::make, 350
h_vec
  mln::algebra::h_vec, 581
  mln::point, 1000
has
  mln::box, 592
  mln::doc::Box, 645
  mln::doc::Fastest_Image, 653, 654
  mln::doc::Image, 662, 663
  mln::doc::Site_Set, 679
  mln::doc::Value_Set, 683
  mln::extension_fun, 711
  mln::extension_ima, 714
  mln::extension_val, 717
  mln::flat_image, 720
  mln::hexa, 812
  mln::image1d, 821
  mln::image2d, 826
  mln::image2d_h, 830
  mln::image3d, 834
  mln::interpolated, 837
  mln::lazy_image, 843
  mln::p_array, 906
  mln::p_centered, 910
  mln::p_complex, 913
  mln::p_edges, 918, 919
  mln::p_if, 927
  mln::p_image, 930
  mln::p_key, 938
  mln::p_line2d, 942
  mln::p_mutable_array_of, 946
  mln::p_priority, 955
  mln::p_queue, 960
  mln::p_queue_fast, 964, 965
  mln::p_run, 969, 970
  mln::p_set, 974
  mln::p_transformed, 978
  mln::p_vaccess, 983
  mln::p_vertices, 989
  mln::set, 411
  mln::tr_image, 1093
  mln::util::line_graph, 1135, 1136
  mln::util::set, 1149
  mln::value::lut_vec, 1199
  mln::window, 1240
has_e
  mln::util::graph, 1126
  mln::util::line_graph, 1136
has_index
  mln::p_run, 970
has_v
  mln::util::graph, 1126
  mln::util::line_graph, 1136
height
  mln::morpho::attribute::sharpness, 890
  mln::win::cuboid3d, 1227
  mln::win::rectangle2d, 1236
hexa

mln::hexa, 812
higher_dim_adj_faces
mln::topo::algebraic_face, 1052
mln::topo::algebraic_n_face, 1056
mln::topo::face, 1067
mln::topo::n_face, 1079
highest_priority
mln::p_priority, 955
hit_or_miss
mln::morpho, 370
mln::morpho::impl::generic, 381
hit_or_miss_background_closing
mln::morpho, 370
hit_or_miss_background_opening
mln::morpho, 371
hit_or_miss_closing
mln::morpho, 371
hit_or_miss_opening
mln::morpho, 371
hline2d
modwin2d, 96
hough
mln::transform, 431

i_element
mln::p_array, 905
mln::p_image, 929
mln::p_key, 937
mln::p_mutable_array_of, 945
mln::p_priority, 954
mln::p_queue, 959
mln::p_queue_fast, 964
mln::p_set, 973
mln::p_vaccess, 982
icp
mln::registration, 406
id
mln::graph_window_if_piter, 807
mln::graph_window_piter, 809
mln::p_graph_piter, 923
mln::util::edge, 1117
mln::util::vertex, 1169
id_t
mln::util::edge, 1116
mln::util::vertex, 1169
id_value_t
mln::util::edge, 1116
mln::util::vertex, 1169
identity
mln::literal, 331
Identity morphers, 72
ima
mln::doc::Generalized_Pixel, 658
mln::doc::Pixel_Iterator, 671
mln::fun::x2x::linear, 755
mln::util::pix, 1146
image
mln::bkd_pixter1d, 582
mln::bkd_pixter2d, 584
mln::bkd_pixter3d, 586
mln::doc::Generalized_Pixel, 657
mln::doc::Pixel_Iterator, 671
mln::fwd_pixter1d, 770
mln::fwd_pixter2d, 772
mln::fwd_pixter3d, 774
mln::make, 350, 351
mln::pw::image, 1008
Image morphers, 69
image1d
mln::image1d, 820
image2d
mln::image2d, 825
mln::make, 351
image2d_h
mln::image2d_h, 830
image3d
mln::image3d, 833
mln::make, 351, 352
Images, 67
implies
mln, 131
inc_face_id
mln::topo::algebraic_face, 1052
mln::topo::algebraic_n_face, 1056
mln::topo::face, 1067
mln::topo::n_face, 1079
inc_n
mln::topo::algebraic_face, 1052
mln::topo::face, 1067
index
mln::p_indexed_bkd_piter, 932
mln::p_indexed_fwd_piter, 933
index_of
mln::doc::Value_Set, 683
mln::value::lut_vec, 1199
influence_zone_adjacency_graph
mln::make, 352
influence_zone_front
mln::transform, 432
influence_zone_geodesic
mln::transform, 432
init
mln::accu::center, 458
mln::accu::convolve, 459
mln::accu::count_adjacent_vertices, 461
mln::accu::count_labels, 463
mln::accu::count_value, 465
mln::accu::label_used, 469

mln::accu::logic::land, 471
mln::accu::logic::land_basic, 473
mln::accu::logic::lor, 475
mln::accu::logic::lor_basic, 477
mln::accu::maj_h, 479
mln::accu::math::count, 481
mln::accu::math::inf, 483
mln::accu::math::sum, 485
mln::accu::math::sup, 487
mln::accu::max_site, 489
mln::accu::nil, 525
mln::accu::p, 527
mln::accu::pair, 530
mln::accu::rms, 531
mln::accu::shape::bbox, 533
mln::accu::shape::height, 536
mln::accu::shape::volume, 538
mln::accu::stat::deviation, 542
mln::accu::stat::max, 544
mln::accu::stat::max_h, 546
mln::accu::stat::mean, 548
mln::accu::stat::median_h, 552
mln::accu::stat::min, 555
mln::accu::stat::min_h, 557
mln::accu::stat::min_max, 560
mln::accu::stat::rank, 561
mln::accu::stat::rank< bool >, 563
mln::accu::stat::rank_high_quant, 565
mln::accu::stat::var, 568
mln::accu::stat::variance, 571
mln::accu::tuple, 573
mln::accu::val, 575
mln::doc::Accumulator, 642
mln::morpho::attribute::card, 884
mln::morpho::attribute::count_adjacent_vertices, 886
mln::morpho::attribute::height, 888
mln::morpho::attribute::sharpness, 891
mln::morpho::attribute::sum, 892
mln::morpho::attribute::volume, 894
mln::p_run, 970
initialize
mln, 131
insert
mln::p_array, 906
mln::p_image, 930
mln::p_key, 938
mln::p_mutable_array_of, 946
mln::p_priority, 955
mln::p_queue, 960
mln::p_queue_fast, 965
mln::p_set, 974
mln::p_vaccess, 983
mln::util::set, 1150
mln::w_window, 1219
mln::window, 1240, 1241
int_s
mln::value::int_s, 1186
int_s16
mln::value, 447
int_s32
mln::value, 447
int_s8
mln::value, 447
int_u
mln::value::int_u, 1188
int_u12
mln::value, 447
int_u16
mln::value, 447
int_u32
mln::value, 447
int_u8
mln::value, 447
int_u8_1complex_image2d
mln, 128
int_u8_2complex_image2d
mln, 128
int_u8_2complex_image3df
mln, 128
int_u_sat
mln::value::int_u_sat, 1190
inter
mln::Site_Set, 1021
interpolated
mln::interpolated, 837
inv
mln::fun::x2x::rotation, 758
mln::fun::x2x::translation, 760
invalidate
mln::complex_psite, 631
mln::doc::Iterator, 665
mln::doc::Pixel_Iterator, 671
mln::doc::Site_Iterator, 677
mln::doc::Value_Iterator, 681
mln::dpoints_bkd_pixter, 696
mln::dpoints_fwd_pixter, 699
mln::Graph, 786
mln::p_edges, 919
mln::p_vertices, 989
mln::topo::algebraic_face, 1053
mln::topo::algebraic_n_face, 1056
mln::topo::face, 1067
mln::topo::n_face, 1079
mln::util::branch_iter, 1108
mln::util::branch_iter_ind, 1110
mln::util::edge, 1117
mln::util::graph, 1126

mln::util::line_graph, 1136
mln::util::vertex, 1170
invert
mln::fun::x2x::rotation, 757
mln::fun::x2x::translation, 760
iota
mln::debug, 218
is_centered
mln::doc::Weighted_Window, 686
mln::graph_elt_window, 797
mln::graph_elt_window_if, 802
mln::graph_window_base, 805
mln::window, 1241
is_empty
mln::Box, 596
mln::box, 592
mln::doc::Weighted_Window, 686
mln::graph_elt_window, 797
mln::graph_elt_window_if, 802
mln::graph_window_base, 805
mln::util::array, 1103
mln::util::fibonacci_heap, 1120
mln::util::set, 1150
mln::util::soft_heap, 1156
mln::window, 1241
is_facet
mln::topo, 422
is_simple_2d
mln, 132
is_subgraph_of
mln::util::graph, 1126
mln::util::line_graph, 1136
is_symmetric
mln::graph_elt_window, 797
mln::graph_elt_window_if, 802
mln::graph_window_base, 805
mln::w_window, 1219
mln::window, 1241
is_valid
mln::accu::center, 458
mln::accu::convolve, 459
mln::accu::count_adjacent_vertices, 461
mln::accu::count_labels, 463
mln::accu::count_value, 465
mln::accu::histo, 467
mln::accu::label_used, 469
mln::accu::logic::land, 471
mln::accu::logic::land_basic, 473
mln::accu::logic::lor, 475
mln::accu::logic::lor_basic, 477
mln::accu::maj_h, 479
mln::accu::math::count, 481
mln::accu::math::inf, 483
mln::accu::math::sum, 485
mln::accu::math::sup, 487
mln::accu::max_site, 489
mln::accu::nil, 525
mln::accu::p, 527
mln::accu::pair, 530
mln::accu::rms, 531
mln::accu::shape::bbox, 533
mln::accu::shape::height, 536
mln::accu::shape::volume, 538
mln::accu::stat::deviation, 542
mln::accu::stat::max, 544
mln::accu::stat::max_h, 546
mln::accu::stat::mean, 548
mln::accu::stat::median_alt, 550
mln::accu::stat::median_h, 552
mln::accu::stat::min, 555
mln::accu::stat::min_h, 557
mln::accu::stat::min_max, 560
mln::accu::stat::rank, 561
mln::accu::stat::rank< bool >, 563
mln::accu::stat::rank_high_quant, 565
mln::accu::stat::var, 568
mln::accu::stat::variance, 571
mln::accu::tuple, 573
mln::accu::val, 575
mln::box, 592
mln::complex_psite, 631
mln::doc::Fastest_Image, 654
mln::doc::Image, 663
mln::doc::Iterator, 665
mln::doc::Pixel_Iterator, 671
mln::doc::Site_Iterator, 677
mln::doc::Value_Iterator, 681
mln::dpoints_bkd_pixter, 696
mln::dpoints_fwd_pixter, 699
mln::Graph, 786
mln::graph_elt_window, 797
mln::graph_elt_window_if, 802
mln::graph_window_base, 805
mln::interpolated, 837
mln::morpho::attribute::card, 884
mln::morpho::attribute::count_adjacent_-
vertices, 886
mln::morpho::attribute::height, 888
mln::morpho::attribute::sharpness, 891
mln::morpho::attribute::sum, 892
mln::morpho::attribute::volume, 894
mln::p_array, 906
mln::p_centered, 910
mln::p_complex, 913
mln::p_edges, 919
mln::p_faces, 922
mln::p_if, 927
mln::p_image, 930

mln::p_key, 938
mln::p_line2d, 942
mln::p_mutable_array_of, 946
mln::p_priority, 955
mln::p_queue, 960
mln::p_queue_fast, 965
mln::p_run, 970
mln::p_set, 974
mln::p_transformed, 978
mln::p_vaccess, 983
mln::p_vertices, 989
mln::pixel, 992
mln::topo::algebraic_face, 1053
mln::topo::algebraic_n_face, 1056
mln::topo::face, 1067
mln::topo::n_face, 1079
mln::tr_image, 1093
mln::util::branch_iter, 1108
mln::util::branch_iter_ind, 1110
mln::util::edge, 1117
mln::util::fibonacci_heap, 1120
mln::util::graph, 1126
mln::util::line_graph, 1136
mln::util::soft_heap, 1156
mln::util::vertex, 1170
mln::value::stack_image, 1213

iter
mln::complex_neighborhood_bkd_piter, 627
mln::complex_neighborhood_fwd_piter, 629
mln::complex_window_bkd_piter, 634
mln::complex_window_fwd_piter, 636

iter_type
mln::complex_neighborhood_bkd_piter, 626
mln::complex_neighborhood_fwd_piter, 628
mln::complex_window_bkd_piter, 633
mln::complex_window_fwd_piter, 635

ith_nbh_edge
mln::util::edge, 1117
mln::util::vertex, 1170

ith_nbh_vertex
mln::util::vertex, 1170

k
mln::accu::stat::rank, 561

key
mln::p_key, 938

keys
mln::p_key, 939

l1
mln::norm, 399

l1_distance
mln::norm, 399

l2
mln::norm, 399

l2_distance
mln::norm, 399

label
mln::value::label, 1195

label_16
mln::value, 447

label_8
mln::value, 447

labeled_image
mln::labeled_image, 840

labeling
mln::graph, 266

laplacian
mln::morpho, 371

larger_than
mln, 132

last_coord
mln::point, 1002

last_element
mln::util::set, 1150

lazy_image
mln::lazy_image, 843

ldlt_decomp
mln::algebra, 157

ldlt_solve
mln::algebra, 157

lemmings
mln::util, 438

len
mln::Box, 596
mln::box, 592

length
mln::p_run, 970
mln::win::backdiag2d, 1222
mln::win::cube3d, 1225
mln::win::diag2d, 1228
mln::win::line, 1230
mln::win::octagon2d, 1234

light_gray
mln::literal, 331

lime
mln::literal, 331

line
mln::accu, 144
mln::draw, 227
mln::win::line, 1230

line_gradient
mln::morpho, 371

linear
mln::fun::x2x::linear, 754

linfty
mln::norm, 399

linfty_distance

mln::norm, 399
load
mln::io::cloud, 276
mln::io::dicom, 277
mln::io::dump, 278
mln::io::fits, 279
mln::io::magick, 280
mln::io::off, 281
mln::io::pbm, 283
mln::io::pbms, 286
mln::io::pfm, 288
mln::io::pgm, 291
mln::io::pgms, 293
mln::io::plot, 294
mln::io::pnm, 296
mln::io::pnms, 299
mln::io::ppm, 300
mln::io::ppms, 302
mln::io::tiff, 303
load_raw_2d
mln::io::pnm, 296
lower_dim_adj_faces
mln::topo::algebraic_face, 1053
mln::topo::algebraic_n_face, 1056
mln::topo::face, 1067
mln::topo::n_face, 1079
lowest_priority
mln::p_priority, 955
lut_vec
mln::value::lut_vec, 1198
lvalue
mln::complex_image, 624
mln::decorated_image, 638
mln::doc::Fastest_Image, 651
mln::doc::Image, 661
mln::doc::Pixel_Iterator, 671
mln::flat_image, 720
mln::fun_image, 763
mln::hexa, 811
mln::image1d, 819
mln::image2d, 824
mln::image2d_h, 829
mln::image3d, 832
mln::interpolated, 836
mln::lazy_image, 843
mln::tr_image, 1092
mln::value::stack_image, 1212

magenta
mln::literal, 332
main_branch
mln::util::tree, 1162
make_algebraic_face
mln::topo, 422
make_algebraic_n_face
mln::topo, 422
make_debug_graph_image
mln, 132
make_greater_point
mln::util, 438
make_greater_psite
mln::util, 438
mask
mln::graph_elt_neighborhood_if, 793
mln::graph_elt_window_if, 802
mask_t
mln::graph_elt_window_if, 801
mat
mln::make, 352
max
mln::literal, 332
mln::morpho::tree::filter, 392
max_col
mln::geom, 257
max_component
mln::io::pnm, 297
max_ind
mln::geom, 258
max_row
mln::geom, 258
max_sli
mln::geom, 258
mean
mln::accu::stat::var, 568
mln::accu::stat::variance, 571
mln::estim, 229
mean_t
mln::accu::stat::var, 568
median
mln::data, 196
mln::data::approx, 203, 204
mln::data::impl::generic, 209
mln::data::naive, 213
medium_gray
mln::literal, 332
memory_size
mln::box, 593
mln::p_array, 906
mln::p_centered, 910
mln::p_edges, 919
mln::p_if, 927
mln::p_image, 931
mln::p_key, 939
mln::p_line2d, 943
mln::p_mutable_array_of, 946
mln::p_priority, 956
mln::p_queue, 961
mln::p_queue_fast, 965

mln::p_run, 970
mln::p_set, 975
mln::p_transformed, 978
mln::p_vaccess, 984
mln::p_vertices, 990
mln::util::array, 1103
mln::util::set, 1151
mesh
mln::doc::Point_Site, 674
mesh_corner_point_area
mln::geom, 258
mesh_curvature
mln::geom, 259
mesh_normal
mln::geom, 259
meyer_wst
mln::morpho, 371, 372
min
mln::arith, 162
mln::literal, 332
mln::morpho, 372
mln::morpho::tree::filter, 392
min_col
mln::geom, 259
min_ind
mln::geom, 259
min_inplace
mln::arith, 163
mln::morpho, 372
min_max
mln::estim, 230
min_row
mln::geom, 260
min_sli
mln::geom, 260
minus
mln::arith, 163
mln::morpho, 372
minus_cst
mln::arith, 164
minus_cst_inplace
mln::arith, 165
minus_infty
mln::point, 1002
minus_inplace
mln::arith, 165
mirror
mln::border, 176
mln, 105
a_point_of, 130
apply_p2p, 130
before, 141
bin_2complex_image3df, 127
box1d, 127
box2d, 127
box2d_h, 127
box3d, 127
compose, 130
discrete_plane_1complex_geometry, 127
discrete_plane_2complex_geometry, 127
dpoint1d, 127
dpoint2d, 128
dpoint2d_h, 128
dpoint3d, 128
duplicate, 130
extend, 131
float_2complex_image3df, 128
implies, 131
initialize, 131
int_u8_1complex_image2d, 128
int_u8_2complex_image2d, 128
int_u8_2complex_image3df, 128
is_simple_2d, 132
larger_than, 132
make_debug_graph_image, 132
mln_exact, 132
mln_gen_complex_neighborhood, 132, 133
mln_gen_complex_window, 133, 134
mln_regular, 134
mln_trait_op_geq, 134
mln_trait_op_greater, 134
mln_trait_op_leq, 134
mln_trait_op_neq, 135
operator<, 136, 137
operator<<, 137
operator<=, 138
operator∗, 136
operator++, 136
operator-, 136
operator–, 136
operator==, 138–140
p_run2d, 128
p_runs2d, 128
point1d, 128
point1df, 129
point2d, 129
point2d_h, 129
point2df, 129
point3d, 129
point3df, 129
primary, 141
ptransform, 141
rgb8_2complex_image3df, 129
sagittal_dec, 141
space_2complex_geometry, 129
up, 141
vec2d_d, 129
vec2d_f, 129

vec3d_d, 129
vec3d_f, 130
mln::accu, 142
compute, 144
line, 144
mln_meta_accu_result, 144
take, 145
mln::accu::center, 457
init, 458
is_valid, 458
take_as_init, 458
take_n_times, 458
to_result, 458
mln::accu::convolve, 459
init, 459
is_valid, 459
take_as_init, 459
take_n_times, 460
to_result, 460
mln::accu::count_adjacent_vertices, 461
init, 461
is_valid, 461
set_value, 462
take_as_init, 462
take_n_times, 462
to_result, 462
mln::accu::count_labels, 463
init, 463
is_valid, 463
set_value, 463
take_as_init, 464
take_n_times, 464
to_result, 464
mln::accu::count_value, 465
init, 465
is_valid, 465
set_value, 465
take_as_init, 466
take_n_times, 466
to_result, 466
mln::accu::histo, 467
is_valid, 467
take, 467
take_as_init, 467
take_n_times, 468
vect, 468
mln::accu::image, 146
mln::accu::impl, 147
mln::accu::label_used, 469
init, 469
is_valid, 469
take, 469
take_as_init, 470
take_n_times, 470
to_result, 470
mln::accu::logic, 148
mln::accu::logic::land, 471
init, 471
is_valid, 471
take_as_init, 471
take_n_times, 471
to_result, 472
mln::accu::logic::land_basic, 473
can_stop, 473
init, 473
is_valid, 473
to_result, 473
mln::accu::logic::lor, 475
init, 475
is_valid, 475
take_as_init, 475
take_n_times, 475
to_result, 476
mln::accu::logic::lor_basic, 477
can_stop, 477
init, 477
is_valid, 477
take_as_init, 478
take_n_times, 478
to_result, 478
mln::accu::maj_h, 479
init, 479
is_valid, 479
take_as_init, 479
take_n_times, 480
to_result, 480
mln::accu::math, 149
mln::accu::math::count, 481
init, 481
is_valid, 481
set_value, 481
take_as_init, 482
take_n_times, 482
to_result, 482
mln::accu::math::inf, 483
init, 483
is_valid, 483
take_as_init, 483
take_n_times, 484
to_result, 484
mln::accu::math::sum, 485
init, 485
is_valid, 485
take_as_init, 485
take_n_times, 486
to_result, 486
mln::accu::math::sup, 487
init, 487

is_valid, 487
take_as_init, 487
take_n_times, 488
to_result, 488
mln::accu::max_site, 489
init, 489
is_valid, 489
take_as_init, 489
take_n_times, 489
to_result, 490
mln::accu::meta::center, 491
mln::accu::meta::count_adjacent_vertices, 492
mln::accu::meta::count_labels, 493
mln::accu::meta::count_value, 494
mln::accu::meta::histo, 495
mln::accu::meta::label_used, 496
mln::accu::meta::logic, 150
mln::accu::meta::logic::land, 497
mln::accu::meta::logic::land_basic, 498
mln::accu::meta::logic::lor, 499
mln::accu::meta::logic::lor_basic, 500
mln::accu::meta::maj_h, 501
mln::accu::meta::math, 151
mln::accu::meta::math::count, 502
mln::accu::meta::math::inf, 503
mln::accu::meta::math::sum, 504
mln::accu::meta::math::sup, 505
mln::accu::meta::max_site, 506
mln::accu::meta::nil, 507
mln::accu::meta::p, 508
mln::accu::meta::pair, 509
mln::accu::meta::rms, 510
mln::accu::meta::shape, 152
mln::accu::meta::shape::bbox, 511
mln::accu::meta::shape::height, 512
mln::accu::meta::shape::volume, 513
mln::accu::meta::stat, 153
mln::accu::meta::stat::max, 514
mln::accu::meta::stat::max_h, 515
mln::accu::meta::stat::mean, 516
mln::accu::meta::stat::median_alt, 517
mln::accu::meta::stat::median_h, 518
mln::accu::meta::stat::min, 519
mln::accu::meta::stat::min_h, 520
mln::accu::meta::stat::rank, 521
mln::accu::meta::stat::rank_high_quant, 522
mln::accu::meta::tuple, 523
mln::accu::meta::val, 524
mln::accu::nil, 525
init, 525
is_valid, 525
take_as_init, 525
take_n_times, 525
to_result, 526
mln::accu::p, 527
init, 527
is_valid, 527
take_as_init, 527
take_n_times, 528
to_result, 528
mln::accu::pair, 529
init, 530
is_valid, 530
take_as_init, 530
take_n_times, 530
to_result, 530
mln::accu::rms, 531
init, 531
is_valid, 531
take_as_init, 531
take_n_times, 532
to_result, 532
mln::accu::shape, 154
mln::accu::shape::bbox, 533
init, 533
is_valid, 533
take_as_init, 533
take_n_times, 534
to_result, 534
mln::accu::shape::height, 535
argument, 536
init, 536
is_valid, 536
set_value, 536
take_as_init, 536
take_n_times, 536
to_result, 536
value, 536
mln::accu::shape::volume, 537
argument, 538
init, 538
is_valid, 538
set_value, 538
take_as_init, 538
take_n_times, 538
to_result, 538
value, 538
mln::accu::site_set::rectangularity, 540
area, 540
bbox, 540
rectangularity, 540
take_as_init, 541
take_n_times, 541
to_result, 541
mln::accu::stat, 155
mln::accu::stat::deviation, 542
init, 542
is_valid, 542

take_as_init, 542
take_n_times, 543
to_result, 543
mln::accu::stat::max, 544
init, 544
is_valid, 544
take_as_init, 544
take_n_times, 545
to_result, 545
mln::accu::stat::max_h, 546
init, 546
is_valid, 546
take_as_init, 546
take_n_times, 546
to_result, 547
mln::accu::stat::mean, 548
init, 548
is_valid, 548
take_as_init, 548
take_n_times, 549
to_result, 549
mln::accu::stat::median_alt, 550
is_valid, 550
take, 550
take_as_init, 551
take_n_times, 551
to_result, 551
mln::accu::stat::median_h, 552
init, 552
is_valid, 552
take_as_init, 553
take_n_times, 553
to_result, 553
mln::accu::stat::meta::deviation, 554
mln::accu::stat::min, 555
init, 555
is_valid, 555
take_as_init, 555
take_n_times, 556
to_result, 556
mln::accu::stat::min_h, 557
init, 557
is_valid, 557
take_as_init, 557
take_n_times, 557
to_result, 558
mln::accu::stat::min_max, 559
init, 560
is_valid, 560
take_as_init, 560
take_n_times, 560
to_result, 560
mln::accu::stat::rank, 561
init, 561
is_valid, 561
k, 561
take_as_init, 562
take_n_times, 562
to_result, 562
mln::accu::stat::rank< bool >, 563
init, 563
is_valid, 563
take_as_init, 563
take_n_times, 563
to_result, 564
mln::accu::stat::rank_high_quant, 565
init, 565
is_valid, 565
take_as_init, 565
take_n_times, 566
to_result, 566
mln::accu::stat::var, 567
init, 568
is_valid, 568
mean, 568
mean_t, 568
n_items, 568
take_as_init, 568
take_n_times, 568
to_result, 568
variance, 568
mln::accu::stat::variance, 570
init, 571
is_valid, 571
mean, 571
n_items, 571
standard_deviation, 571
sum, 571
take_as_init, 571
take_n_times, 571
to_result, 571
var, 572
mln::accu::tuple, 573
init, 573
is_valid, 573
take_as_init, 573
take_n_times, 574
to_result, 574
mln::accu::val, 575
init, 575
is_valid, 575
take_as_init, 575
take_n_times, 575
to_result, 576
mln::Accumulator, 577
take_as_init, 577
take_n_times, 577
mln::algebra, 157

ldlt_decomp, 157
ldlt_solve, 157
operator∗, 158
vprod, 158
mln::algebra::h_mat, 578
_1, 579
h_mat, 578
t, 579
mln::algebra::h_vec, 580
h_vec, 581
operator mat< n, 1, U >, 581
origin, 581
t, 581
to_vec, 581
zero, 581
mln::arith, 159
diff_abs, 161
div, 161
div_cst, 161
div_inplace, 162
min, 162
min_inplace, 163
minus, 163
minus_cst, 164
minus_cst_inplace, 165
minus_inplace, 165
plus, 165, 166
plus_cst, 166, 167
plus_cst_inplace, 167
plus_inplace, 167
revert, 168
revert_inplace, 168
times, 169
times_cst, 169
times_inplace, 169
mln::arith::impl, 171
mln::arith::impl::generic, 172
mln::binarization, 173
binarization, 173
threshold, 173
mln::bkd_pixter1d, 582
bkd_pixter1d, 582
image, 582
next, 582
mln::bkd_pixter2d, 584
bkd_pixter2d, 584
image, 584
next, 584
mln::bkd_pixter3d, 586
bkd_pixter3d, 586
image, 586
next, 586
mln::border, 174
adjust, 174
duplicate, 175
equalize, 175
fill, 175
find, 176
get, 176
mirror, 176
resize, 176
mln::border::impl, 178
mln::border::impl::generic, 179
mln::Box, 595
bbox, 596
is_empty, 596
len, 596
nsites, 596
operator<, 597
operator<=, 597
mln::box, 588
bbox, 591
bkd_piter, 590
box, 591
center, 591
crop_wrt, 592
element, 590
enlarge, 592
fwd_piter, 590
has, 592
is_empty, 592
is_valid, 592
len, 592
memory_size, 593
nsites, 593
operator<<, 594
piter, 590
pmax, 593
pmin, 593
psite, 591
site, 591
to_larger, 593
mln::box_runstart_piter, 598
box_runstart_piter, 598
next, 598
run_length, 598
mln::Browsing, 600
mln::canvas, 180
distance_front, 180
distance_geodesic, 180
mln::canvas::browsing, 182
mln::canvas::browsing::backdiagonal2d_t, 601
mln::canvas::browsing::breadth_first_search_t, 603
mln::canvas::browsing::depth_first_search_t, 604
mln::canvas::browsing::diagonal2d_t, 605
mln::canvas::browsing::dir_struct_elt_incr_-
update_t, 607
mln::canvas::browsing::directional_t, 609

mln::canvas::browsing::fwd_t, 611
mln::canvas::browsing::hyper_directional_t, 613
mln::canvas::browsing::snake_fwd_t, 615
mln::canvas::browsing::snake_generic_t, 617
mln::canvas::browsing::snake_vert_t, 619
mln::canvas::chamfer, 621
mln::canvas::impl, 183
mln::canvas::morpho, 184
mln::category< R(*)(A) >, 622
mln::complex_image, 623
  complex_image, 624
  dim, 625
  domain, 625
  geom, 624
  lvalue, 624
  operator(), 625
  rvalue, 624
  skeleton, 624
  value, 624
  values, 625
mln::complex_neighborhood_bkd_piter, 626
  complex_neighborhood_bkd_piter, 627
  iter, 627
  iter_type, 626
  next, 627
  psite, 626
mln::complex_neighborhood_fwd_piter, 628
  complex_neighborhood_fwd_piter, 629
  iter, 629
  iter_type, 628
  next, 629
  psite, 628
mln::complex_psite, 630
  change_target, 631
  complex_psite, 631
  face, 631
  face_id, 631
  invalidate, 631
  is_valid, 631
  n, 632
  site_set, 632
mln::complex_window_bkd_piter, 633
  complex_window_bkd_piter, 634
  iter, 634
  iter_type, 633
  next, 634
  psite, 633
mln::complex_window_fwd_piter, 635
  complex_window_fwd_piter, 636
  iter, 636
  iter_type, 635
  next, 636
  psite, 635
mln::convert, 185
  from_to, 187
  mln_image_from_grid, 187, 188
  mln_window, 188
  to, 188
  to_dpoint, 188
  to_fun, 188
  to_image, 188
  to_p_array, 188, 189
  to_p_set, 189
  to_upper_window, 189, 190
  to_window, 190
mln::data, 191
  abs, 193
  abs_inplace, 193
  apply, 193
  compute, 193
  convert, 194
  fast_median, 194
  fill, 195
  fill_with_image, 195
  fill_with_value, 195
  median, 196
  mln_meta_accu_result, 196
  paste, 196
  paste_without_localization, 197
  replace, 197
  saturate, 197, 198
  saturate_inplace, 198
  sort_offsets_increasing, 198
  sort_psites_decreasing, 198
  sort_psites_increasing, 199
  stretch, 199
  to_enc, 199
  transform, 200
  transform_inplace, 201
  update, 202
mln::data::approx, 203
  median, 203, 204
mln::data::approx::impl, 205
mln::data::impl, 206
  stretch, 206
  transform_inplace_lowq, 206
  update_fastest, 207
mln::data::impl::generic, 208
  convert, 209
  fill_with_image, 209
  fill_with_value, 209
  median, 209
  paste, 210
  sort_offsets_increasing, 210
  transform, 210
  transform_inplace, 211
  update, 211
mln::data::naive, 213

median, 213
mln::data::naive::impl, 214
mln::debug, 215
draw_graph, 216, 217
filename, 217
format, 217
iota, 218
println, 218
println_with_border, 218
put_word, 218
slices_2d, 218
superpose, 219
mln::debug::impl, 220
mln::decorated_image, 637
decorated_image, 638
decoration, 638
lvalue, 638
operator decorated_image< const I, D >, 638
operator(), 638, 639
psite, 638
rvalue, 638
skeleton, 638
mln::def, 221
coord, 221
coordf, 221
mln::Delta_Point_Site, 640
mln::Delta_Point_Site< void >, 641
mln::display, 222
mln::display::impl, 223
mln::display::impl::generic, 224
mln::doc, 225
mln::doc::Accumulator, 642
argument, 642
init, 642
take, 642
mln::doc::Box, 644
bbox, 645
bkd_piter, 645
fwd_piter, 645
has, 645
nsites, 646
pmax, 646
pmin, 646
psite, 645
site, 645
mln::doc::Dpoint, 647
coord, 647
dim, 648
dpoint, 647
point, 648
mln::doc::Fastest_Image, 649
bbox, 653
bkd_piter, 651
border, 653
buffer, 653
coord, 651
delta_index, 653
domain, 653
dpoint, 651
fwd_piter, 651
has, 653, 654
is_valid, 654
lvalue, 651
nelements, 654
nsites, 654
operator(), 654, 655
point, 652
point_at_index, 655
pset, 652
psite, 652
rvalue, 652
skeleton, 652
value, 652
values, 656
vset, 652
mln::doc::Generalized_Pixel, 657
ima, 658
image, 657
rvalue, 657
val, 658
value, 658
mln::doc::Image, 659
bbox, 662
bkd_piter, 661
coord, 661
domain, 662
dpoint, 661
fwd_piter, 661
has, 662, 663
is_valid, 663
lvalue, 661
nsites, 663
operator(), 663
point, 661
pset, 661
psite, 661
rvalue, 662
skeleton, 662
value, 662
values, 664
vset, 662
mln::doc::Iterator, 665
invalidate, 665
is_valid, 665
start, 665
mln::doc::Neighborhood, 667
bkd_niter, 667
dpoint, 667

fwd_niter, 668
niter, 668
point, 668
mln::doc::Object, 669
mln::doc::Pixel_Iterator, 670
ima, 671
image, 671
invalidate, 671
is_valid, 671
lvalue, 671
rvalue, 671
start, 671
val, 672
value, 671
mln::doc::Point_Site
dim, 674
mln::doc::Point_Site, 673
coord, 673
dpoint, 673
mesh, 674
point, 674
to_point, 674
mln::doc::Site_Iterator, 676
invalidate, 677
is_valid, 677
operator psite, 677
psite, 677
start, 677
mln::doc::Site_Set, 678
bkd_piter, 679
fwd_piter, 679
has, 679
psite, 679
site, 679
mln::doc::Value_Iterator, 680
invalidate, 681
is_valid, 681
operator value, 681
start, 681
value, 681
mln::doc::Value_Set, 682
bkd_viter, 683
fwd_viter, 683
has, 683
index_of, 683
nvalues, 683
value, 683
mln::doc::Weighted_Window, 684
bkd_qiter, 685
delta, 686
dpoint, 685
fwd_qiter, 685
is_centered, 686
is_empty, 686
point, 685
sym, 686
weight, 685
win, 686
window, 685
mln::doc::Window, 687
bkd_qiter, 687
fwd_qiter, 687
qiter, 687
mln::Dpoint, 694
to_dpoint, 694
mln::dpoint, 689
coord, 690
dim, 691
dpoint, 691, 692
grid, 690
operator mln::algebra::vec< dpoint< G, C >::dim, Q >, 692
psite, 691
set_all, 692
site, 691
to_vec, 693
vec, 691
mln::dpoints_bkd_pixter, 695
center_val, 696
dpoints_bkd_pixter, 695, 696
invalidate, 696
is_valid, 696
next, 696
start, 696
update, 697
mln::dpoints_fwd_pixter, 698
center_val, 699
dpoints_fwd_pixter, 698, 699
invalidate, 699
is_valid, 699
next, 699
start, 699
update, 700
mln::dpsites_bkd_piter, 701
dpsites_bkd_piter, 701
next, 701
mln::dpsites_fwd_piter, 703
dpsites_fwd_piter, 703
next, 703
mln::draw, 227
box, 227
line, 227
plot, 228
mln::Edge, 705
mln::edge_image, 706
edge_image, 707
graph_t, 706
nbh_t, 706

operator(), 707
site_function_t, 707
skeleton, 707
win_t, 707
mln::estim, 229
mean, 229
min_max, 230
sum, 230
mln::extended, 708
domain, 709
extended, 709
skeleton, 708
value, 708
mln::extension, 231
adjust, 231, 232
adjust_duplicate, 232
adjust_fill, 232
duplicate, 232
fill, 232
mln::extension_fun, 710
extension, 711
extension_fun, 711
has, 711
operator(), 711
rvalue, 711
skeleton, 711
value, 711
mln::extension_ima, 713
extension, 714
extension_ima, 714
has, 714
operator(), 714
rvalue, 714
skeleton, 714
value, 714
mln::extension_val, 716
change_extension, 717
extension, 717
extension_val, 717
has, 717
operator(), 717
rvalue, 717
skeleton, 717
value, 717
mln::flat_image, 719
domain, 720
flat_image, 720
has, 720
lvalue, 720
operator(), 720
rvalue, 720
skeleton, 720
value, 720
mln::fun, 234
mln::fun::access, 236
mln::fun::i2v, 237
mln::fun::p2b, 238
mln::fun::p2b::antilogy, 722
mln::fun::p2b::tautology, 723
mln::fun::p2p, 239
mln::fun::p2v, 240
mln::fun::stat, 241
mln::fun::v2b, 242
mln::fun::v2b::lnot, 724
mln::fun::v2b::threshold, 725
mln::fun::v2i, 243
mln::fun::v2v, 244
f_hsi_to_rgb_3x8, 245
f_hsl_to_rgb_3x8, 245
f_rgb_to_hsi_f, 245
f_rgb_to_hsl_f, 245
mln::fun::v2v::ch_function_value, 726
mln::fun::v2v::component, 727
mln::fun::v2v::l1_norm, 728
mln::fun::v2v::l2_norm, 729
mln::fun::v2v::linear, 730
mln::fun::v2v::linfty_norm, 731
mln::fun::v2w2v, 246
mln::fun::v2w2v::cos, 732
mln::fun::v2w_w2v, 247
mln::fun::v2w_w2v::l1_norm, 733
mln::fun::v2w_w2v::l2_norm, 734
mln::fun::v2w_w2v::linfty_norm, 735
mln::fun::vv2b, 248
mln::fun::vv2b::eq, 736
mln::fun::vv2b::ge, 737
mln::fun::vv2b::gt, 738
mln::fun::vv2b::implies, 739
mln::fun::vv2b::le, 740
mln::fun::vv2b::lt, 741
mln::fun::vv2v, 249
mln::fun::vv2v::diff_abs, 742
mln::fun::vv2v::land, 743
mln::fun::vv2v::land_not, 744
mln::fun::vv2v::lor, 745
mln::fun::vv2v::lxor, 746
mln::fun::vv2v::max, 747
mln::fun::vv2v::min, 748
mln::fun::vv2v::vec, 749
mln::fun::x2p, 250
mln::fun::x2p::closest_point, 750
mln::fun::x2v, 251
mln::fun::x2v::bilinear, 751
operator(), 751
mln::fun::x2v::trilinear, 752
mln::fun::x2x, 252
mln::fun::x2x::composed, 753
composed, 753

mln::fun::x2x::linear, 754
ima, 755
linear, 754
operator(), 754
mln::fun::x2x::rotation, 756
inv, 758
invert, 757
operator(), 758
rotation, 757
set_alpha, 758
set_axis, 758
mln::fun::x2x::translation, 759
inv, 760
invert, 760
operator(), 760
set_t, 761
t, 761
translation, 760
mln::fun_image, 762
fun_image, 763
lvalue, 763
operator(), 763
rvalue, 763
skeleton, 763
value, 763
mln::Function, 764
Function, 764
mln::Function< void >, 765
mln::Function_v2b, 766
mln::Function_v2v, 767
mln::Function_vv2b, 768
mln::Function_vv2v, 769
mln::fwd_pixter1d, 770
fwd_pixter1d, 770
image, 770
next, 771
mln::fwd_pixter2d, 772
fwd_pixter2d, 772
image, 772
next, 773
mln::fwd_pixter3d, 774
fwd_pixter3d, 774
image, 774
next, 775
mln::Gdpoint, 776
mln::Gdpoint< void >, 777
mln::Generalized_Pixel, 778
mln::geom, 253
bbox, 256, 257
chamfer, 257
delta, 257
max_col, 257
max_ind, 258
max_row, 258
max_sli, 258
mesh_corner_point_area, 258
mesh_curvature, 259
mesh_normal, 259
min_col, 259
min_ind, 259
min_row, 260
min_sli, 260
ncols, 260
ninds, 260
nrows, 260, 261
nsites, 261
nslis, 261
pmin_pmax, 261
rotate, 262
seeds2tiling, 262
seeds2tiling_roundness, 263
mln::geom::complex_geometry, 779
add_location, 780
complex_geometry, 779
operator(), 780
mln::geom::impl, 264
seeds2tiling, 264
seeds2tiling_roundness, 264
mln::Gpoint, 781
operator<<, 784
operator+, 782
operator+=, 782
operator-, 783
operator-=, 783
operator/, 784
operator==, 784
mln::Graph, 786
invalidate, 786
is_valid, 786
mln::graph, 266
compute, 266
labeling, 266
to_neighb, 267
to_win, 267
mln::graph::attribute::card_t, 788
result, 788
mln::graph::attribute::representative_t, 789
result, 789
mln::graph_elt_neighborhood, 790
bkd_niter, 791
change_window, 791
fwd_niter, 791
niter, 791
win, 791
mln::graph_elt_neighborhood_if, 792
bkd_niter, 793
change_window, 793
fwd_niter, 793

graph_elt_neighborhood_if, 793
mask, 793
niter, 793
win, 794
mln::graph_elt_window, 795
bkd_qiter, 796
delta, 797
fwd_qiter, 796
is_centered, 797
is_empty, 797
is_symmetric, 797
is_valid, 797
psite, 796
qiter, 796
site, 797
sym, 797
target, 797
mln::graph_elt_window_if, 799
bkd_qiter, 800
change_mask, 802
delta, 802
fwd_qiter, 800
graph_elt_window_if, 801
is_centered, 802
is_empty, 802
is_symmetric, 802
is_valid, 802
mask, 802
mask_t, 801
psite, 801
qiter, 801
site, 801
sym, 803
target, 801
mln::graph_window_base, 804
delta, 805
is_centered, 805
is_empty, 805
is_symmetric, 805
is_valid, 805
site, 804
sym, 805
mln::graph_window_if_piter, 806
element, 807
graph_window_if_piter, 807
id, 807
next, 807
P, 806
mln::graph_window_piter, 808
element, 809
graph_window_piter, 808
id, 809
next, 809
P, 808
mln::grid, 269
mln::hexa, 810
bkd_piter, 811
domain, 812
fwd_piter, 811
has, 812
hexa, 812
lvalue, 811
operator(), 812
psite, 811
rvalue, 812
skeleton, 812
value, 812
mln::histo, 270
compute, 270
mln::histo::array, 814
mln::histo::impl, 271
mln::histo::impl::generic, 272
mln::Image, 815
mln::image1d, 818
bbox, 820
border, 820
buffer, 820
delta_index, 820
domain, 820
element, 821
has, 821
image1d, 820
lvalue, 819
nelements, 821
ninds, 821
operator(), 821
point_at_index, 821
rvalue, 819
skeleton, 819
value, 819
mln::image2d, 823
bbox, 825
border, 825
buffer, 825
delta_index, 825
domain, 826
element, 826
has, 826
image2d, 825
lvalue, 824
ncols, 826
nelements, 826
nrows, 826
operator(), 826, 827
point_at_index, 827
rvalue, 824
skeleton, 824
value, 825

mln::image2d_h, 828
bkd_piter, 829
domain, 830
fwd_piter, 829
has, 830
image2d_h, 830
lvalue, 829
operator(), 830
psite, 829
rvalue, 829
skeleton, 829
value, 830
mln::image3d, 831
bbox, 833
border, 833
buffer, 833
delta_index, 834
domain, 834
element, 834
has, 834
image3d, 833
lvalue, 832
ncols, 834
nelements, 834
nrows, 834
nslices, 835
operator(), 835
point_at_index, 835
rvalue, 832
skeleton, 833
value, 833
mln::impl, 273
mln::interpolated, 836
has, 837
interpolated, 837
is_valid, 837
lvalue, 836
psite, 836
rvalue, 837
skeleton, 837
value, 837
mln::io, 274
mln::io::cloud, 276
load, 276
save, 276
mln::io::dicom, 277
load, 277
mln::io::dump, 278
load, 278
save, 278
mln::io::fits, 279
load, 279
mln::io::magick, 280
load, 280
save, 280
mln::io::off, 281
load, 281
save, 281
save_bin_alt, 282
mln::io::pbm, 283
load, 283
save, 284
mln::io::pbm::impl, 285
mln::io::pbms, 286
load, 286
mln::io::pbms::impl, 287
mln::io::pfm, 288
load, 288
save, 289
mln::io::pfm::impl, 290
mln::io::pgm, 291
load, 291
save, 292
mln::io::pgms, 293
load, 293
mln::io::plot, 294
load, 294
save, 294
mln::io::pnm, 296
load, 296
load_raw_2d, 296
max_component, 297
save, 297
mln::io::pnm::impl, 298
mln::io::pnms, 299
load, 299
mln::io::ppm, 300
load, 300
save, 301
mln::io::ppms, 302
load, 302
mln::io::tiff, 303
load, 303
mln::io::txt, 304
save, 304
mln::Iterator, 838
next, 838
mln::labeled_image, 839
bbox, 840
bbox_t, 840
bboxes, 840
labeled_image, 840
nlabels, 841
relabel, 841
skeleton, 840
subdomain, 841
mln::labeling, 305
background, 307

blobs, 308
colorize, 308
compute, 309–311
compute_image, 311, 312
fill_holes, 313
flat_zones, 313
foreground, 313
pack, 314
pack_inplace, 314
regional_maxima, 314
regional_minima, 315
relabel, 315
relabel_inplace, 316
value, 316
wrap, 317
mln::labeling::impl, 318
mln::labeling::impl::generic, 319
compute, 319, 320
mln::lazy_image, 842
domain, 843
has, 843
lazy_image, 843
lvalue, 843
operator(), 844
rvalue, 843
skeleton, 843
mln::linear, 321
gaussian, 322
gaussian_1st_derivative, 322
gaussian_2nd_derivative, 323
mln_ch_convolve, 323
mln_ch_convolve_grad, 324
mln::linear::impl, 325
mln::linear::local, 326
convolve, 326
mln::linear::local::impl, 327
mln::Literal, 845
mln::literal, 328
black, 331
blue, 331
brown, 331
cyan, 331
dark_gray, 331
green, 331
identity, 331
light_gray, 331
lime, 331
magenta, 332
max, 332
medium_gray, 332
min, 332
olive, 332
one, 332
orange, 332
origin, 332
pink, 332
purple, 332
red, 332
teal, 333
violet, 333
white, 333
yellow, 333
zero, 333
mln::literal::black_t, 847
mln::literal::blue_t, 848
mln::literal::brown_t, 849
mln::literal::cyan_t, 850
mln::literal::green_t, 851
mln::literal::identity_t, 852
mln::literal::light_gray_t, 853
mln::literal::lime_t, 854
mln::literal::magenta_t, 855
mln::literal::max_t, 856
mln::literal::min_t, 857
mln::literal::olive_t, 858
mln::literal::one_t, 859
mln::literal::orange_t, 860
mln::literal::origin_t, 861
mln::literal::pink_t, 862
mln::literal::purple_t, 863
mln::literal::red_t, 864
mln::literal::teal_t, 865
mln::literal::violet_t, 866
mln::literal::white_t, 867
mln::literal::yellow_t, 868
mln::literal::zero_t, 869
mln::logical, 334
and_inplace, 334
and_not, 334
and_not_inplace, 335
not_inplace, 335
or_inplace, 336
xor_inplace, 336
mln::logical::impl, 337
mln::logical::impl::generic, 338
mln::make, 339
attachment, 343
box1d, 344
box2d, 344, 345
box2d_h, 345
box3d, 346
cell, 347
couple, 347
detachment, 347
dpoint2d_h, 348
dummy_p_edges, 348
dummy_p_vertices, 348, 349
edge_image, 349, 350

h_mat, 350
image, 350, 351
image2d, 351
image3d, 351, 352
influence_zone_adjacency_graph, 352
mat, 352
ord_pair, 353
p_edges_with_mass_centers, 353
p_vertices_with_mass_centers, 353
pix, 353
pixel, 354
point2d_h, 354
rag_and_labeled_wsl, 354
region_adjacency_graph, 355
relabelfun, 355, 356
vec, 356, 357
vertex_image, 357, 358
voronoi, 358
w_window, 358
w_window1d, 359
w_window2d, 359
w_window3d, 359
w_window_directional, 360
mln::math, 361
abs, 361
mln::Mesh, 870
mln::Meta_Accumulator, 871
mln::Meta_Function, 873
mln::Meta_Function_v2v, 874
mln::Meta_Function_vv2v, 875
mln::metal, 362
mln::metal::ands, 876
mln::metal::converts_to, 877
mln::metal::equal, 878
mln::metal::goes_to, 879
mln::metal::impl, 363
mln::metal::is, 880
mln::metal::is_a, 881
mln::metal::is_not, 882
mln::metal::is_not_a, 883
mln::metal::math, 364
mln::metal::math::impl, 365
mln::morpho, 366
complementation, 369
complementation_inplace, 369
contrast, 369
dilation, 369
erosion, 369
general, 369
gradient, 370
gradient_external, 370
gradient_internal, 370
hit_or_miss, 370
hit_or_miss_background_closing, 370
hit_or_miss_background_opening, 371
hit_or_miss_closing, 371
hit_or_miss_opening, 371
laplacian, 371
line_gradient, 371
meyer_wst, 371, 372
min, 372
min_inplace, 372
minus, 372
plus, 373
rank_filter, 373
thick_miss, 373
thickening, 373
thin_fit, 373
thinning, 374
top_hat_black, 374
top_hat_self_complementary, 374
top_hat_white, 374
mln::morpho::approx, 375
mln::morpho::attribute, 376
mln::morpho::attribute::card, 884
init, 884
is_valid, 884
take_as_init, 884
take_n_times, 884
to_result, 885
mln::morpho::attribute::count_adjacent_vertices, 886
init, 886
is_valid, 886
take_as_init, 886
take_n_times, 887
to_result, 887
mln::morpho::attribute::height, 888
base_level, 888
init, 888
is_valid, 888
take_as_init, 889
take_n_times, 889
to_result, 889
mln::morpho::attribute::sharpness, 890
area, 890
height, 890
init, 891
is_valid, 891
take_as_init, 891
take_n_times, 891
to_result, 891
volume, 891
mln::morpho::attribute::sum, 892
init, 892
is_valid, 892
set_value, 893
take_as_init, 893

take_n_times, 893
to_result, 893
untake, 893
mln::morpho::attribute::volume, 894
area, 894
init, 894
is_valid, 894
take_as_init, 895
take_n_times, 895
to_result, 895
mln::morpho::closing::approx, 377
structural, 377
mln::morpho::elementary, 378
closing, 378
mln_trait_op_minus_twice, 379
opening, 379
top_hat_black, 379
top_hat_self_complementary, 379
top_hat_white, 379
mln::morpho::impl, 380
mln::morpho::impl::generic, 381
hit_or_miss, 381
rank_filter, 381
mln::morpho::opening::approx, 382
structural, 382
mln::morpho::reconstruction, 383
mln::morpho::reconstruction::by_dilation, 384
mln::morpho::reconstruction::by_erosion, 385
mln::morpho::tree, 386
compute_attribute_image_from, 387
compute_parent, 387
propagate_if, 388
propagate_if_value, 389
propagate_node_to_ancestors, 389
propagate_node_to_descendants, 390
propagate_representative, 390
mln::morpho::tree::filter, 391
direct, 391
filter, 391
max, 392
min, 392
subtractive, 392
mln::morpho::watershed, 394
flooding, 394, 395
superpose, 395
mln::morpho::watershed::watershed, 396
mln::morpho::watershed::watershed::generic, 397
mln::neighb, 896
bkd_niter, 897
change_window, 897
fwd_niter, 897
neighb, 897
niter, 897
win, 897
mln::Neighborhood, 898
mln::Neighborhood< void >, 899
mln::norm, 398
l1, 399
l1_distance, 399
l2, 399
l2_distance, 399
linfty, 399
linfty_distance, 399
sqr_l2, 399
mln::norm::impl, 400
mln::Object, 900
mln::opt, 401
at, 401, 402
mln::opt::impl, 403
mln::p2p_image, 901
domain, 902
fun, 902
operator(), 902
p2p_image, 902
skeleton, 901
mln::p_array, 903
append, 905
bkd_piter, 904
change, 905
clear, 906
element, 904
fwd_piter, 905
has, 906
i_element, 905
insert, 906
is_valid, 906
memory_size, 906
nsites, 906
p_array, 905
piter, 905
psite, 905
reserve, 907
std_vector, 907
mln::p_centered, 908
bkd_piter, 909
center, 910
element, 909
fwd_piter, 909
has, 910
is_valid, 910
memory_size, 910
p_centered, 909
piter, 909
psite, 909
site, 909
window, 910
mln::p_complex, 911
bkd_piter, 912

cplx, 913
element, 912
fwd_piter, 912
geom, 913
has, 913
is_valid, 913
nfaces, 913
nfaces_of_dim, 914
nsites, 914
p_complex, 913
piter, 912
psite, 912
mln::p_edges, 915
bkd_piter, 916
edge, 916
element, 917
fun_t, 917
function, 918
fwd_piter, 917
graph, 918
graph_element, 917
graph_t, 917
has, 918, 919
invalidate, 919
is_valid, 919
memory_size, 919
nedges, 919
nsites, 919
p_edges, 917, 918
piter, 917
psite, 917
mln::p_faces, 920
bkd_piter, 921
cplx, 922
element, 921
fwd_piter, 921
is_valid, 922
nfaces, 922
nsites, 922
p_faces, 921
piter, 921
psite, 921
mln::p_graph_piter, 923
graph, 923
id, 923
mln_q_subject, 924
next, 924
p_graph_piter, 923
mln::p_if, 925
bkd_piter, 926
element, 926
fwd_piter, 926
has, 927
is_valid, 927
memory_size, 927
overset, 927
p_if, 926
piter, 926
pred, 927
predicate, 927
psite, 926
mln::p_image, 928
bkd_piter, 929
clear, 930
element, 929
fwd_piter, 929
has, 930
i_element, 929
insert, 930
is_valid, 930
memory_size, 931
nsites, 931
operator typename internal::p_image_site_-set< I >::ret, 931
p_image, 930
piter, 929
psite, 929
r_element, 930
remove, 931
S, 930
toggle, 931
mln::p_indexed_bkd_piter, 932
index, 932
p_indexed_bkd_piter, 932
mln::p_indexed_fwd_piter, 933
index, 933
p_indexed_fwd_piter, 933
mln::p_indexed_psite, 934
mln::p_key, 935
bkd_piter, 937
change_key, 937
change_keys, 938
clear, 938
element, 937
exists_key, 938
fwd_piter, 937
has, 938
i_element, 937
insert, 938
is_valid, 938
key, 938
keys, 939
memory_size, 939
nsites, 939
operator(), 939
p_key, 937
piter, 937
psite, 937

r_element, 937
remove, 939
remove_key, 939
mln::p_line2d, 940
bbox, 942
begin, 942
bkd_piter, 941
element, 941
end, 942
fwd_piter, 941
has, 942
is_valid, 942
memory_size, 943
nsites, 943
p_line2d, 942
piter, 941
psite, 941
q_box, 941
std_vector, 943
mln::p_mutable_array_of, 944
bkd_piter, 945
clear, 946
element, 945
fwd_piter, 945
has, 946
i_element, 945
insert, 946
is_valid, 946
memory_size, 946
nelements, 946
p_mutable_array_of, 946
piter, 945
psite, 945
reserve, 947
mln::p_n_faces_bkd_piter, 948
n, 948
next, 948
p_n_faces_bkd_piter, 948
mln::p_n_faces_fwd_piter, 950
n, 950
next, 950
p_n_faces_fwd_piter, 950
mln::p_priority, 952
bkd_piter, 953
clear, 954
element, 953
exists_priority, 954
front, 954
fwd_piter, 954
has, 955
highest_priority, 955
i_element, 954
insert, 955
is_valid, 955
lowest_priority, 955
memory_size, 956
nsites, 956
operator(), 956
p_priority, 954
piter, 954
pop, 956
pop_front, 956
priorities, 956
psite, 954
push, 957
mln::p_queue, 958
bkd_piter, 959
clear, 960
element, 959
front, 960
fwd_piter, 959
has, 960
i_element, 959
insert, 960
is_valid, 960
memory_size, 961
nsites, 961
p_queue, 960
piter, 960
pop, 961
pop_front, 961
psite, 960
push, 961
std_deque, 961
mln::p_queue_fast, 962
bkd_piter, 963
clear, 964
compute_has, 964
element, 963
front, 964
fwd_piter, 964
has, 964, 965
i_element, 964
insert, 965
is_valid, 965
memory_size, 965
nsites, 965
p_queue_fast, 964
piter, 964
pop, 965
pop_front, 965
psite, 964
purge, 966
push, 966
reserve, 966
std_vector, 966
mln::p_run, 967
bbox, 969

bkd_piter, 968
element, 968
end, 969
fwd_piter, 968
has, 969, 970
has_index, 970
init, 970
is_valid, 970
length, 970
memory_size, 970
nsites, 970
p_run, 969
piter, 968
psite, 969
q_box, 969
start, 970
mln::p_set, 972
bkd_piter, 973
clear, 974
element, 973
fwd_piter, 973
has, 974
i_element, 973
insert, 974
is_valid, 974
memory_size, 975
nsites, 975
p_set, 974
piter, 974
psite, 974
r_element, 974
remove, 975
std_vector, 975
util_set, 975
mln::p_transformed, 976
bkd_piter, 977
element, 977
function, 978
fwd_piter, 977
has, 978
is_valid, 978
memory_size, 978
p_transformed, 977
piter, 977
primary_set, 978
psite, 977
mln::p_transformed_piter, 979
change_target, 980
p_transformed_piter, 979
mln::p_vaccess, 981
bkd_piter, 982
element, 982
fwd_piter, 982
has, 983
i_element, 982
insert, 983
is_valid, 983
memory_size, 984
operator(), 984
p_vaccess, 983
piter, 982
pset, 982
psite, 983
value, 983
values, 984
vset, 983
mln::p_vertices, 985
bkd_piter, 987
element, 987
fun_t, 987
function, 989
fwd_piter, 987
graph, 989
graph_element, 987
graph_t, 987
has, 989
invalidate, 989
is_valid, 989
memory_size, 990
nsites, 990
nvertices, 990
operator(), 990
p_vertices, 988
piter, 987
psite, 987
vertex, 987
mln::pixel, 991
change_to, 992
is_valid, 992
pixel, 991
mln::plain, 993
operator I, 994
operator=, 994
plain, 994
skeleton, 993
mln::Point, 995
operator+=, 996
operator-=, 996
operator/, 996
point, 996
to_point, 996
mln::point, 998
coord, 1000
delta, 1000
dim, 1001
dpsite, 1000
grid, 1000
h_vec, 1000

last_coord, 1002
minus_infty, 1002
operator+=, 1002
operator-=, 1002
origin, 1003
plus_infty, 1003
point, 1001
set_all, 1003
to_h_vec, 1003
to_vec, 1003
vec, 1001
mln::Proxy, 1004
mln::Proxy< void >, 1005
mln::Pseudo_Site, 1006
mln::Pseudo_Site< void >, 1007
mln::pw, 404
mln::pw::image, 1008
image, 1008
skeleton, 1008
mln::registration, 405
get_rot, 406
icp, 406
registration1, 407
registration2, 407
registration3, 407
mln::registration::closest_point_basic, 1009
mln::registration::closest_point_with_map, 1010
mln::Regular_Grid, 1011
mln::safe_image, 1012
operator safe_image< const I >, 1012
skeleton, 1012
mln::select, 408
mln::select::p_of, 1013
mln::set, 409
card, 409
compute, 409
compute_with_weights, 410
get, 411
has, 411
mln_meta_accu_result, 411
mln::Site, 1014
mln::Site< void >, 1015
mln::Site_Iterator, 1016
next, 1017
mln::Site_Proxy, 1018
mln::Site_Proxy< void >, 1019
mln::Site_Set, 1020
diff, 1021
inter, 1021
operator<, 1021
operator<<, 1022
operator<=, 1022
operator==, 1022
sym_diff, 1022
uni, 1022
unique, 1023
mln::Site_Set< void >, 1024
mln::sub_image, 1025
domain, 1026
operator sub_image< const I, S >, 1026
skeleton, 1025
sub_image, 1025
mln::sub_image_if, 1027
domain, 1028
skeleton, 1027
sub_image_if, 1027
mln::subsampling, 412
gaussian_subsampling, 412
subsampling, 412
mln::tag, 413
mln::test, 414
positive, 414
predicate, 414, 415
mln::test::impl, 416
mln::thru_image, 1029
operator thru_image< const I, F >, 1029
mln::thrubin_image, 1030
operator thrubin_image< const I1, const I2, F >, 1031
psite, 1030
rvalue, 1030
skeleton, 1030
value, 1031
mln::topo, 417
detach, 421
edge, 421
is_facet, 422
make_algebraic_face, 422
make_algebraic_n_face, 422
operator<, 423, 424
operator<<, 424, 425
operator+, 423
operator-, 423
operator==, 425
mln::topo::adj_higher_dim_connected_n_face_-bkd_iter, 1032
adj_higher_dim_connected_n_face_bkd_iter, 1032
next, 1032
mln::topo::adj_higher_dim_connected_n_face_-fwd_iter, 1034
adj_higher_dim_connected_n_face_fwd_iter, 1034
next, 1034
mln::topo::adj_higher_face_bkd_iter, 1036
adj_higher_face_bkd_iter, 1036
next, 1036
mln::topo::adj_higher_face_fwd_iter, 1037

adj_higher_face_fwd_iter, 1037
next, 1037
mln::topo::adj_lower_dim_connected_n_face_-bkd_iter, 1038
adj_lower_dim_connected_n_face_bkd_iter, 1038
next, 1038
mln::topo::adj_lower_dim_connected_n_face_-fwd_iter, 1040
adj_lower_dim_connected_n_face_fwd_iter, 1040
next, 1040
mln::topo::adj_lower_face_bkd_iter, 1042
adj_lower_face_bkd_iter, 1042
mln::topo::adj_lower_face_fwd_iter, 1043
adj_lower_face_fwd_iter, 1043
mln::topo::adj_lower_higher_face_bkd_iter, 1044
adj_lower_higher_face_bkd_iter, 1044
next, 1044
mln::topo::adj_lower_higher_face_fwd_iter, 1045
adj_lower_higher_face_fwd_iter, 1045
next, 1045
mln::topo::adj_m_face_bkd_iter, 1046
adj_m_face_bkd_iter, 1046
next, 1047
mln::topo::adj_m_face_fwd_iter, 1048
adj_m_face_fwd_iter, 1048
next, 1049
mln::topo::algebraic_face, 1050
algebraic_face, 1051, 1052
cplx, 1052
data, 1052
dec_face_id, 1052
dec_n, 1052
face_id, 1052
higher_dim_adj_faces, 1052
inc_face_id, 1052
inc_n, 1052
invalidate, 1053
is_valid, 1053
lower_dim_adj_faces, 1053
n, 1053
set_cplx, 1053
set_face_id, 1053
set_n, 1053
set_sign, 1053
sign, 1053
mln::topo::algebraic_n_face, 1054
algebraic_n_face, 1055
cplx, 1056
data, 1056
dec_face_id, 1056
face_id, 1056
higher_dim_adj_faces, 1056
inc_face_id, 1056
invalidate, 1056
is_valid, 1056
lower_dim_adj_faces, 1056
n, 1056
set_cplx, 1057
set_face_id, 1057
set_sign, 1057
sign, 1057
mln::topo::center_only_iter, 1058
center_only_iter, 1058
next, 1059
mln::topo::centered_bkd_iter_adapter, 1060
centered_bkd_iter_adapter, 1060
next, 1060
mln::topo::centered_fwd_iter_adapter, 1061
centered_fwd_iter_adapter, 1061
next, 1061
mln::topo::complex, 1062
add_face, 1063
addr, 1063
bkd_citer, 1063
complex, 1063
fwd_citer, 1063
nfaces, 1063
nfaces_of_dim, 1064
nfaces_of_static_dim, 1064
print, 1064
print_faces, 1064
mln::topo::face, 1065
cplx, 1066
data, 1067
dec_face_id, 1067
dec_n, 1067
face, 1066
face_id, 1067
higher_dim_adj_faces, 1067
inc_face_id, 1067
inc_n, 1067
invalidate, 1067
is_valid, 1067
lower_dim_adj_faces, 1067
n, 1068
set_cplx, 1068
set_face_id, 1068
set_n, 1068
mln::topo::face_bkd_iter, 1069
face_bkd_iter, 1069
next, 1069
start, 1069
mln::topo::face_fwd_iter, 1071
face_fwd_iter, 1071
next, 1071
start, 1071

mln::topo::is_n_face, 1073
mln::topo::is_simple_cell, 1074
D, 1076
mln_geom, 1075
operator(), 1075
psite, 1075
result, 1075
set_image, 1075
mln::topo::n_face, 1077
cplx, 1078
data, 1078
dec_face_id, 1078
face_id, 1078
higher_dim_adj_faces, 1079
inc_face_id, 1079
invalidate, 1079
is_valid, 1079
lower_dim_adj_faces, 1079
n, 1079
n_face, 1078
set_cplx, 1079
set_face_id, 1079
mln::topo::n_face_bkd_iter, 1081
n, 1081
next, 1081
start, 1082
mln::topo::n_face_fwd_iter, 1083
n, 1083
next, 1083
start, 1083
mln::topo::n_faces_set, 1085
add, 1085
faces, 1085
faces_type, 1085
reserve, 1086
mln::topo::static_n_face_bkd_iter, 1087
next, 1087
start, 1088
static_n_face_bkd_iter, 1087
mln::topo::static_n_face_fwd_iter, 1089
next, 1089
start, 1090
static_n_face_fwd_iter, 1089
mln::tr_image, 1091
domain, 1093
has, 1093
is_valid, 1093
lvalue, 1092
operator(), 1093
psite, 1092
rvalue, 1092
set_tr, 1093
site, 1092
skeleton, 1092
tr, 1093
tr_image, 1092
value, 1092
mln::trace, 427
mln::trait, 428
mln::transform, 429
distance_and_closest_point_geodesic, 430
distance_and_influence_zone_geodesic, 431
distance_front, 431
distance_geodesic, 431
hough, 431
influence_zone_front, 432
influence_zone_geodesic, 432
mln::transformed_image, 1094
domain, 1095
operator transformed_image< const I, F >, 1095
operator(), 1095
skeleton, 1094
transformed_image, 1095
mln::unproject_image, 1096
domain, 1096
operator(), 1096, 1097
unproject_image, 1096
mln::util, 434
display_branch, 437
display_tree, 437
lemmings, 438
make_greater_point, 438
make_greater_psite, 438
operator<, 438
operator<<, 438
operator==, 439
ord_strict, 439
ord_weak, 439
tree_fast_to_image, 439
tree_to_fast, 439
tree_to_image, 440
vertex_id_t, 437
mln::util::adjacency_matrix, 1098
adjacency_matrix, 1098
mln::util::array, 1099
append, 1102
array, 1102
bkd_eiter, 1101
clear, 1102
eiter, 1101
element, 1101
fill, 1103
fwd_eiter, 1101
is_empty, 1103
memory_size, 1103
nelements, 1103
operator(), 1103

reserve, 1104
resize, 1104
result, 1102
size, 1104
std_vector, 1105
mln::util::branch, 1106
apex, 1106
branch, 1106
util_tree, 1106
mln::util::branch_iter, 1108
deepness, 1108
invalidate, 1108
is_valid, 1108
next, 1109
operator util::tree_node< T > &, 1109
start, 1109
mln::util::branch_iter_ind, 1110
deepness, 1110
invalidate, 1110
is_valid, 1110
next, 1111
operator util::tree_node< T > &, 1111
start, 1111
mln::util::couple, 1112
change_both, 1113
change_first, 1113
change_second, 1113
first, 1113
second, 1113
mln::util::eat, 1114
mln::util::edge, 1115
category, 1116
change_graph, 1117
edge, 1116
graph, 1117
graph_t, 1116
id, 1117
id_t, 1116
id_value_t, 1116
invalidate, 1117
is_valid, 1117
ith_nbh_edge, 1117
nmax_nbh_edges, 1117
operator edge_id_t, 1117
update_id, 1117
v1, 1118
v2, 1118
v_other, 1118
mln::util::fibonacci_heap, 1119
clear, 1120
fibonacci_heap, 1120
front, 1120
is_empty, 1120
is_valid, 1120
nelements, 1121
operator=, 1121
pop_front, 1121
push, 1121
mln::util::graph, 1122
add_edge, 1125
add_vertex, 1125
add_vertices, 1125
e_ith_nbh_edge, 1125
e_nmax, 1125
edge, 1125, 1126
edge_fwd_iter, 1124
edge_nbh_edge_fwd_iter, 1124
edges, 1126
edges_set_t, 1124
edges_t, 1124
graph, 1124
has_e, 1126
has_v, 1126
invalidate, 1126
is_subgraph_of, 1126
is_valid, 1126
v1, 1126
v2, 1127
v_ith_nbh_edge, 1127
v_ith_nbh_vertex, 1127
v_nmax, 1127
vertex, 1127
vertex_nbh_edge_fwd_iter, 1124
vertex_nbh_vertex_fwd_iter, 1124
vertices_t, 1124
mln::util::greater_point, 1128
operator(), 1128
mln::util::greater_psite, 1129
operator(), 1129
mln::util::head, 1130
mln::util::ignore, 1131
mln::util::ilcell, 1132
mln::util::impl, 441
tree_fast_to_image, 441
mln::util::line_graph, 1133
e_ith_nbh_edge, 1135
e_nmax, 1135
edge, 1135
edge_fwd_iter, 1134
edges_t, 1134
graph, 1135
has, 1135, 1136
has_e, 1136
has_v, 1136
invalidate, 1136
is_subgraph_of, 1136
is_valid, 1136
v1, 1136

v2, 1137
v_ith_nbh_edge, 1137
v_ith_nbh_vertex, 1137
v_nmax, 1137
vertex, 1137
vertex_nbh_edge_fwd_iter, 1135
vertex_nbh_vertex_fwd_iter, 1135
vertices_t, 1135
mln::util::nil, 1138
mln::util::node, 1139
mln::util::object_id, 1140
object_id, 1141
value_t, 1141
mln::util::ord, 1142
mln::util::ord_pair, 1143
change_both, 1144
change_first, 1144
change_second, 1144
first, 1144
second, 1144
mln::util::pix, 1145
ima, 1146
p, 1146
pix, 1146
psite, 1145
v, 1146
value, 1145
mln::util::set, 1147
bkd_eiter, 1149
clear, 1149
eiter, 1149
element, 1149
first_element, 1149
fwd_eiter, 1149
has, 1149
insert, 1150
is_empty, 1150
last_element, 1150
memory_size, 1151
nelements, 1151
remove, 1151
set, 1149
std_vector, 1151
mln::util::site_pair, 1153
first, 1153
pair, 1153
second, 1154
mln::util::soft_heap, 1155
~soft_heap, 1156
clear, 1156
element, 1156
is_empty, 1156
is_valid, 1156
nelements, 1157
pop_front, 1157
push, 1157
soft_heap, 1156
mln::util::timer, 1158
mln::util::tracked_ptr, 1159
~tracked_ptr, 1160
operator bool, 1160
operator->, 1160
operator=, 1160
tracked_ptr, 1159
mln::util::tree, 1161
add_tree_down, 1162
add_tree_up, 1162
check_consistency, 1162
main_branch, 1162
root, 1162
tree, 1161
mln::util::tree_node, 1163
add_child, 1164
check_consistency, 1164
children, 1165
delete_tree_node, 1165
elt, 1165
parent, 1165
print, 1166
search, 1166
search_rec, 1166
set_parent, 1166
tree_node, 1164
mln::util::vertex, 1167
Category, 1169
change_graph, 1169
edge_with, 1169
graph, 1169
graph_t, 1169
id, 1169
id_t, 1169
id_value_t, 1169
invalidate, 1170
is_valid, 1170
ith_nbh_edge, 1170
ith_nbh_vertex, 1170
nmax_nbh_edges, 1170
nmax_nbh_vertices, 1170
operator vertex_id_t, 1170
other, 1170
update_id, 1171
vertex, 1169
mln::util::yes, 1172
mln::Value, 1173
mln::value, 442
cast, 448
equiv, 448
float01_16, 446

float01_8, 446
gl16, 446
gl8, 446
glf, 446
int_s16, 447
int_s32, 447
int_s8, 447
int_u12, 447
int_u16, 447
int_u32, 447
int_u8, 447
label_16, 447
label_8, 447
operator<<, 449–451
operator∗, 448
operator+, 448
operator-, 448
operator/, 449
operator==, 452
other, 452
rgb16, 447
rgb8, 447
stack, 452
mln::value::float01, 1174
enc, 1175
equiv, 1175
float01, 1175
nbits, 1175
operator float, 1175
set_nbits, 1175
to_nbits, 1175
value, 1176
value_ind, 1176
mln::value::float01_f, 1177
float01_f, 1177
operator float, 1177
operator=, 1177
value, 1177
mln::value::graylevel, 1179
graylevel, 1180
operator=, 1181
to_float, 1181
value, 1181
mln::value::graylevel_f, 1182
graylevel_f, 1183
operator graylevel< n >, 1183
operator=, 1183, 1184
value, 1184
mln::value::impl, 453
mln::value::int_s, 1185
int_s, 1186
one, 1186
operator int, 1186
operator=, 1186
zero, 1186
mln::value::int_u, 1187
int_u, 1188
next, 1188
operator unsigned, 1188
operator-, 1188
operator=, 1188
mln::value::int_u_sat, 1189
int_u_sat, 1190
one, 1191
operator int, 1190
operator+=, 1190
operator-=, 1190
operator=, 1190
zero, 1191
mln::value::Integer, 1192
mln::value::Integer< void >, 1193
mln::value::label, 1194
enc, 1195
label, 1195
next, 1195
operator unsigned, 1195
operator++, 1195
operator–, 1195
operator=, 1195, 1196
prev, 1196
mln::value::lut_vec, 1197
bkd_viter, 1198
fwd_viter, 1198
has, 1199
index_of, 1199
lut_vec, 1198
nvalues, 1199
value, 1198
mln::value::proxy, 1200
∼proxy, 1201
enc, 1201
equiv, 1201
operator typename I::value, 1202
operator V, 1202
operator=, 1202
proxy, 1201
to_value, 1202
mln::value::proxy< const I >, 1203
∼proxy, 1204
enc, 1204
equiv, 1204
operator typename I::value, 1204
operator V, 1204
proxy, 1204
to_value, 1204
mln::value::rgb, 1206
operator=, 1207
red, 1207

rgb, 1206, 1207
zero, 1207
mln::value::set, 1208
the, 1208
mln::value::sign, 1209
enc, 1210
equiv, 1210
one, 1210
operator int, 1210
operator=, 1210
sign, 1210
zero, 1210
mln::value::stack_image, 1211
domain_t, 1212
is_valid, 1213
lvalue, 1212
operator(), 1213
psite, 1212
rvalue, 1212
skeleton, 1212
stack_image, 1212
value, 1212
mln::Vertex, 1214
mln::vertex_image, 1215
graph_t, 1215
nbh_t, 1215
operator(), 1216
site_function_t, 1216
skeleton, 1216
vertex_image, 1216
win_t, 1216
mln::w_window, 1217
bkd_qiter, 1218
clear, 1219
dpsite, 1218
fwd_qiter, 1218
insert, 1219
is_symmetric, 1219
operator<<, 1220
operator==, 1220
std_vector, 1219
sym, 1219
w, 1219
w_window, 1218
weight, 1218
weights, 1219
win, 1219
mln::Weighted_Window, 1221
operator-, 1221
mln::win, 454
diff, 455
mln_regular, 455
sym, 456
mln::win::backdiag2d, 1222
backdiag2d, 1222
length, 1222
mln::win::ball, 1223
ball, 1223
diameter, 1223
mln::win::cube3d, 1224
cube3d, 1224
length, 1225
mln::win::cuboid3d, 1226
cuboid3d, 1227
depth, 1227
height, 1227
volume, 1227
width, 1227
mln::win::diag2d, 1228
diag2d, 1228
length, 1228
mln::win::line, 1229
length, 1230
line, 1230
size, 1230
mln::win::multiple, 1231
mln::win::multiple_size, 1232
mln::win::octagon2d, 1233
area, 1234
length, 1234
octagon2d, 1233
mln::win::rectangle2d, 1235
area, 1236
height, 1236
rectangle2d, 1235
std_vector, 1236
width, 1236
mln::Window, 1237
mln::window, 1238
bkd_qiter, 1239
clear, 1240
delta, 1240
dp, 1240
fwd_qiter, 1239
has, 1240
insert, 1240, 1241
is_centered, 1241
is_empty, 1241
is_symmetric, 1241
operator==, 1242
print, 1241
qiter, 1239
regular, 1240
size, 1241
std_vector, 1241
sym, 1242
window, 1240
mln::world::inter_pixel::is_separator, 1243

mln_ch_convolve
mln::linear, 323
mln_ch_convolve_grad
mln::linear, 324
mln_exact
mln, 132
mln_gen_complex_neighborhood
mln, 132, 133
mln_gen_complex_window
mln, 133, 134
mln_geom
mln::topo::is_simple_cell, 1075
mln_image_from_grid
mln::convert, 187, 188
mln_meta_accu_result
mln::accu, 144
mln::data, 196
mln::set, 411
mln_q_subject
mln::p_graph_piter, 924
mln_regular
mln, 134
mln::win, 455
mln_trait_op_geq
mln, 134
mln_trait_op_greater
mln, 134
mln_trait_op_leq
mln, 134
mln_trait_op_minus_twice
mln::morpho::elementary, 379
mln_trait_op_neq
mln, 135
mln_window
mln::convert, 188
modneighb1d
c2, 80
neighb1d, 80
modneighb2d
c2_col, 81
c2_row, 81
c4, 82
c8, 82
neighb2d, 81
modneighb3d
c18, 83
c26, 84
c4_3d, 84
c6, 85
c8_3d, 85
neighb3d, 83
modwin1d
segment1d, 94
window1d, 94
modwin2d
disk2d, 96
hline2d, 96
vline2d, 96
win_c4p, 96
win_c8p, 96
window2d, 96
modwin3d
sphere3d, 98
win_c4p_3d, 99
win_c8p_3d, 99
window3d, 98
Multiple accumulators, 65
Multiple windows, 101

n
mln::complex_psite, 632
mln::p_n_faces_bkd_piter, 948
mln::p_n_faces_fwd_piter, 950
mln::topo::algebraic_face, 1053
mln::topo::algebraic_n_face, 1056
mln::topo::face, 1068
mln::topo::n_face, 1079
mln::topo::n_face_bkd_iter, 1081
mln::topo::n_face_fwd_iter, 1083
N-D windows, 100
n_face
mln::topo::n_face, 1078
n_items
mln::accu::stat::var, 568
mln::accu::stat::variance, 571
nbh_t
mln::edge_image, 706
mln::vertex_image, 1215
nbits
mln::value::float01, 1175
ncols
mln::geom, 260
mln::image2d, 826
mln::image3d, 834
nedges
mln::p_edges, 919
neighb
mln::neighb, 897
neighb1d
modneighb1d, 80
neighb2d
modneighb2d, 81
neighb3d
modneighb3d, 83
Neighborhoods, 79
nelements
mln::doc::Fastest_Image, 654
mln::image1d, 821

mln::image2d, 826
mln::image3d, 834
mln::p_mutable_array_of, 946
mln::util::array, 1103
mln::util::fibonacci_heap, 1121
mln::util::set, 1151
mln::util::soft_heap, 1157
next
mln::bkd_pixter1d, 582
mln::bkd_pixter2d, 584
mln::bkd_pixter3d, 586
mln::box_runstart_piter, 598
mln::complex_neighborhood_bkd_piter, 627
mln::complex_neighborhood_fwd_piter, 629
mln::complex_window_bkd_piter, 634
mln::complex_window_fwd_piter, 636
mln::dpoints_bkd_pixter, 696
mln::dpoints_fwd_pixter, 699
mln::dpsites_bkd_piter, 701
mln::dpsites_fwd_piter, 703
mln::fwd_pixter1d, 771
mln::fwd_pixter2d, 773
mln::fwd_pixter3d, 775
mln::graph_window_if_piter, 807
mln::graph_window_piter, 809
mln::Iterator, 838
mln::p_graph_piter, 924
mln::p_n_faces_bkd_piter, 948
mln::p_n_faces_fwd_piter, 950
mln::Site_Iterator, 1017
mln::topo::adj_higher_dim_connected_n_face_bkd_iter, 1032
mln::topo::adj_higher_dim_connected_n_face_fwd_iter, 1034
mln::topo::adj_higher_face_bkd_iter, 1036
mln::topo::adj_higher_face_fwd_iter, 1037
mln::topo::adj_lower_dim_connected_n_face_bkd_iter, 1038
mln::topo::adj_lower_dim_connected_n_face_fwd_iter, 1040
mln::topo::adj_lower_higher_face_bkd_iter, 1044
mln::topo::adj_lower_higher_face_fwd_iter, 1045
mln::topo::adj_m_face_bkd_iter, 1047
mln::topo::adj_m_face_fwd_iter, 1049
mln::topo::center_only_iter, 1059
mln::topo::centered_bkd_iter_adapter, 1060
mln::topo::centered_fwd_iter_adapter, 1061
mln::topo::face_bkd_iter, 1069
mln::topo::face_fwd_iter, 1071
mln::topo::n_face_bkd_iter, 1081
mln::topo::n_face_fwd_iter, 1083
mln::topo::static_n_face_bkd_iter, 1087
mln::topo::static_n_face_fwd_iter, 1089
mln::util::branch_iter, 1109
mln::util::branch_iter_ind, 1111
mln::value::int_u, 1188
mln::value::label, 1195
nfaces
mln::p_complex, 913
mln::p_faces, 922
mln::topo::complex, 1063
nfaces_of_dim
mln::p_complex, 914
mln::topo::complex, 1064
nfaces_of_static_dim
mln::topo::complex, 1064
ninds
mln::geom, 260
mln::image1d, 821
niter
mln::doc::Neighborhood, 668
mln::graph_elt_neighborhood, 791
mln::graph_elt_neighborhood_if, 793
mln::neighb, 897
nlabels
mln::labeled_image, 841
nmax_nbh_edges
mln::util::edge, 1117
mln::util::vertex, 1170
nmax_nbh_vertices
mln::util::vertex, 1170
not_inplace
mln::logical, 335
nrows
mln::geom, 260, 261
mln::image2d, 826
mln::image3d, 834
nsites
mln::Box, 596
mln::box, 593
mln::doc::Box, 646
mln::doc::Fastest_Image, 654
mln::doc::Image, 663
mln::geom, 261
mln::p_array, 906
mln::p_complex, 914
mln::p_edges, 919
mln::p_faces, 922
mln::p_image, 931
mln::p_key, 939
mln::p_line2d, 943
mln::p_priority, 956
mln::p_queue, 961
mln::p_queue_fast, 965
mln::p_run, 970
mln::p_set, 975

mln::p_vertices, 990
nslices
mln::image3d, 835
nslis
mln::geom, 261
nvalues
mln::doc::Value_Set, 683
mln::value::lut_vec, 1199
nvertices
mln::p_vertices, 990

object_id
mln::util::object_id, 1141
octagon2d
mln::win::octagon2d, 1233
olive
mln::literal, 332
On images, 62
On site sets, 61
On values, 63
one
mln::literal, 332
mln::value::int_s, 1186
mln::value::int_u_sat, 1191
mln::value::sign, 1210
opening
mln::morpho::elementary, 379
operator bool
mln::util::tracked_ptr, 1160
operator decorated_image< const I, D >
mln::decorated_image, 638
operator edge_id_t
mln::util::edge, 1117
operator float
mln::value::float01, 1175
mln::value::float01_f, 1177
operator graylevel< n >
mln::value::graylevel_f, 1183
operator I
mln::plain, 994
operator int
mln::value::int_s, 1186
mln::value::int_u_sat, 1190
mln::value::sign, 1210
operator mat< n, 1, U >
mln::algebra::h_vec, 581
operator mln::algebra::vec< dpoint< G, C >::dim, Q >
mln::dpoint, 692
operator psite
mln::doc::Site_Iterator, 677
operator safe_image< const I >
mln::safe_image, 1012
operator sub_image< const I, S >
mln::sub_image, 1026
operator thru_image< const I, F >
mln::thru_image, 1029
operator thrubin_image< const I1, const I2, F >
mln::thrubin_image, 1031
operator transformed_image< const I, F >
mln::transformed_image, 1095
operator typename I::value
mln::value::proxy, 1202
mln::value::proxy< const I >, 1204
operator typename internal::p_image_site_set< I >::ret
mln::p_image, 931
operator unsigned
mln::value::int_u, 1188
mln::value::label, 1195
operator util::tree_node< T > &
mln::util::branch_iter, 1109
mln::util::branch_iter_ind, 1111
operator V
mln::value::proxy, 1202
mln::value::proxy< const I >, 1204
operator value
mln::doc::Value_Iterator, 681
operator vertex_id_t
mln::util::vertex, 1170
operator<
mln, 136, 137
mln::Box, 597
mln::Site_Set, 1021
mln::topo, 423, 424
mln::util, 438
operator<<
mln, 137
mln::box, 594
mln::Gpoint, 784
mln::Site_Set, 1022
mln::topo, 424, 425
mln::util, 438
mln::value, 449–451
mln::w_window, 1220
operator<=
mln, 138
mln::Box, 597
mln::Site_Set, 1022
operator∗
mln, 136
mln::algebra, 158
mln::value, 448
operator()
mln::complex_image, 625
mln::decorated_image, 638, 639
mln::doc::Fastest_Image, 654, 655
mln::doc::Image, 663

mln::edge_image, 707
mln::extension_fun, 711
mln::extension_ima, 714
mln::extension_val, 717
mln::flat_image, 720
mln::fun::x2v::bilinear, 751
mln::fun::x2x::linear, 754
mln::fun::x2x::rotation, 758
mln::fun::x2x::translation, 760
mln::fun_image, 763
mln::geom::complex_geometry, 780
mln::hexa, 812
mln::image1d, 821
mln::image2d, 826, 827
mln::image2d_h, 830
mln::image3d, 835
mln::lazy_image, 844
mln::p2p_image, 902
mln::p_key, 939
mln::p_priority, 956
mln::p_vaccess, 984
mln::p_vertices, 990
mln::topo::is_simple_cell, 1075
mln::tr_image, 1093
mln::transformed_image, 1095
mln::unproject_image, 1096, 1097
mln::util::array, 1103
mln::util::greater_point, 1128
mln::util::greater_psite, 1129
mln::value::stack_image, 1213
mln::vertex_image, 1216

operator+
mln::Gpoint, 782
mln::topo, 423
mln::value, 448

operator++
mln, 136
mln::value::label, 1195

operator+=
mln::Gpoint, 782
mln::Point, 996
mln::point, 1002
mln::value::int_u_sat, 1190

operator-
mln, 136
mln::Gpoint, 783
mln::topo, 423
mln::value, 448
mln::value::int_u, 1188
mln::Weighted_Window, 1221

operator->
mln::util::tracked_ptr, 1160

operator–
mln, 136
mln::value::label, 1195

operator-=
mln::Gpoint, 783
mln::Point, 996
mln::point, 1002
mln::value::int_u_sat, 1190

operator/
mln::Gpoint, 784
mln::Point, 996
mln::value, 449

operator=
mln::plain, 994
mln::util::fibonacci_heap, 1121
mln::util::tracked_ptr, 1160
mln::value::float01_f, 1177
mln::value::graylevel, 1181
mln::value::graylevel_f, 1183, 1184
mln::value::int_s, 1186
mln::value::int_u, 1188
mln::value::int_u_sat, 1190
mln::value::label, 1195, 1196
mln::value::proxy, 1202
mln::value::rgb, 1207
mln::value::sign, 1210

operator==
mln, 138–140
mln::Gpoint, 784
mln::Site_Set, 1022
mln::topo, 425
mln::util, 439
mln::value, 452
mln::w_window, 1220
mln::window, 1242

or_inplace
mln::logical, 336

orange
mln::literal, 332

ord_pair
mln::make, 353

ord_strict
mln::util, 439

ord_weak
mln::util, 439

origin
mln::algebra::h_vec, 581
mln::literal, 332
mln::point, 1003

other
mln::util::vertex, 1170
mln::value, 452

overset
mln::p_if, 927

P

mln::graph_window_if_piter, 806
mln::graph_window_piter, 808
p
mln::util::pix, 1146
p2p_image
mln::p2p_image, 902
p_array
mln::p_array, 905
p_centered
mln::p_centered, 909
p_complex
mln::p_complex, 913
p_edges
mln::p_edges, 917, 918
p_edges_with_mass_centers
mln::make, 353
p_faces
mln::p_faces, 921
p_graph_piter
mln::p_graph_piter, 923
p_if
mln::p_if, 926
p_image
mln::p_image, 930
p_indexed_bkd_piter
mln::p_indexed_bkd_piter, 932
p_indexed_fwd_piter
mln::p_indexed_fwd_piter, 933
p_key
mln::p_key, 937
p_line2d
mln::p_line2d, 942
p_mutable_array_of
mln::p_mutable_array_of, 946
p_n_faces_bkd_piter
mln::p_n_faces_bkd_piter, 948
p_n_faces_fwd_piter
mln::p_n_faces_fwd_piter, 950
p_priority
mln::p_priority, 954
p_queue
mln::p_queue, 960
p_queue_fast
mln::p_queue_fast, 964
p_run
mln::p_run, 969
p_run2d
mln, 128
p_runs2d
mln, 128
p_set
mln::p_set, 974
p_transformed
mln::p_transformed, 977
p_transformed_piter
mln::p_transformed_piter, 979
p_vaccess
mln::p_vaccess, 983
p_vertices
mln::p_vertices, 988
p_vertices_with_mass_centers
mln::make, 353
pack
mln::labeling, 314
pack_inplace
mln::labeling, 314
pair
mln::util::site_pair, 1153
parent
mln::util::tree_node, 1165
paste
mln::data, 196
mln::data::impl::generic, 210
paste_without_localization
mln::data, 197
pink
mln::literal, 332
piter
mln::box, 590
mln::p_array, 905
mln::p_centered, 909
mln::p_complex, 912
mln::p_edges, 917
mln::p_faces, 921
mln::p_if, 926
mln::p_image, 929
mln::p_key, 937
mln::p_line2d, 941
mln::p_mutable_array_of, 945
mln::p_priority, 954
mln::p_queue, 960
mln::p_queue_fast, 964
mln::p_run, 968
mln::p_set, 974
mln::p_transformed, 977
mln::p_vaccess, 982
mln::p_vertices, 987
pix
mln::make, 353
mln::util::pix, 1146
pixel
mln::make, 354
mln::pixel, 991
plain
mln::plain, 994
plot
mln::draw, 228
plus

mln::arith, 165, 166
mln::morpho, 373
plus_cst
mln::arith, 166, 167
plus_cst_inplace
mln::arith, 167
plus_infty
mln::point, 1003
plus_inplace
mln::arith, 167
pmax
mln::box, 593
mln::doc::Box, 646
pmin
mln::box, 593
mln::doc::Box, 646
pmin_pmax
mln::geom, 261
point
mln::doc::Dpoint, 648
mln::doc::Fastest_Image, 652
mln::doc::Image, 661
mln::doc::Neighborhood, 668
mln::doc::Point_Site, 674
mln::doc::Weighted_Window, 685
mln::Point, 996
mln::point, 1001
point1d
mln, 128
point1df
mln, 129
point2d
mln, 129
point2d_h
mln, 129
mln::make, 354
point2df
mln, 129
point3d
mln, 129
point3df
mln, 129
point_at_index
mln::doc::Fastest_Image, 655
mln::image1d, 821
mln::image2d, 827
mln::image3d, 835
pop
mln::p_priority, 956
mln::p_queue, 961
mln::p_queue_fast, 965
pop_front
mln::p_priority, 956
mln::p_queue, 961
mln::p_queue_fast, 965
mln::util::fibonacci_heap, 1121
mln::util::soft_heap, 1157
positive
mln::test, 414
pred
mln::p_if, 927
predicate
mln::p_if, 927
mln::test, 414, 415
prev
mln::value::label, 1196
primary
mln, 141
primary_set
mln::p_transformed, 978
print
mln::topo::complex, 1064
mln::util::tree_node, 1166
mln::window, 1241
print_faces
mln::topo::complex, 1064
println
mln::debug, 218
println_with_border
mln::debug, 218
priorities
mln::p_priority, 956
propagate_if
mln::morpho::tree, 388
propagate_if_value
mln::morpho::tree, 389
propagate_node_to_ancestors
mln::morpho::tree, 389
propagate_node_to_descendants
mln::morpho::tree, 390
propagate_representative
mln::morpho::tree, 390
proxy
mln::value::proxy, 1201
mln::value::proxy< const I >, 1204
pset
mln::doc::Fastest_Image, 652
mln::doc::Image, 661
mln::p_vaccess, 982
psite
mln::box, 591
mln::complex_neighborhood_bkd_piter, 626
mln::complex_neighborhood_fwd_piter, 628
mln::complex_window_bkd_piter, 633
mln::complex_window_fwd_piter, 635
mln::decorated_image, 638
mln::doc::Box, 645
mln::doc::Fastest_Image, 652

mln::doc::Image, 661
mln::doc::Site_Iterator, 677
mln::doc::Site_Set, 679
mln::dpoint, 691
mln::graph_elt_window, 796
mln::graph_elt_window_if, 801
mln::hexa, 811
mln::image2d_h, 829
mln::interpolated, 836
mln::p_array, 905
mln::p_centered, 909
mln::p_complex, 912
mln::p_edges, 917
mln::p_faces, 921
mln::p_if, 926
mln::p_image, 929
mln::p_key, 937
mln::p_line2d, 941
mln::p_mutable_array_of, 945
mln::p_priority, 954
mln::p_queue, 960
mln::p_queue_fast, 964
mln::p_run, 969
mln::p_set, 974
mln::p_transformed, 977
mln::p_vaccess, 983
mln::p_vertices, 987
mln::thrubin_image, 1030
mln::topo::is_simple_cell, 1075
mln::tr_image, 1092
mln::util::pix, 1145
mln::value::stack_image, 1212
ptransform
mln, 141
purge
mln::p_queue_fast, 966
purple
mln::literal, 332
push
mln::p_priority, 957
mln::p_queue, 961
mln::p_queue_fast, 966
mln::util::fibonacci_heap, 1121
mln::util::soft_heap, 1157
put_word
mln::debug, 218

q_box
mln::p_line2d, 941
mln::p_run, 969
qiter
mln::doc::Window, 687
mln::graph_elt_window, 796
mln::graph_elt_window_if, 801
mln::window, 1239
Queue based, 91

r_element
mln::p_image, 930
mln::p_key, 937
mln::p_set, 974
rag_and_labeled_wsl
mln::make, 354
rank_filter
mln::morpho, 373
mln::morpho::impl::generic, 381
rectangle2d
mln::win::rectangle2d, 1235
rectangularity
mln::accu::site_set::rectangularity, 540
red
mln::literal, 332
mln::value::rgb, 1207
region_adjacency_graph
mln::make, 355
regional_maxima
mln::labeling, 314
regional_minima
mln::labeling, 315
registration1
mln::registration, 407
registration2
mln::registration, 407
registration3
mln::registration, 407
regular
mln::window, 1240
relabel
mln::labeled_image, 841
mln::labeling, 315
relabel_inplace
mln::labeling, 316
relabelfun
mln::make, 355, 356
remove
mln::p_image, 931
mln::p_key, 939
mln::p_set, 975
mln::util::set, 1151
remove_key
mln::p_key, 939
replace
mln::data, 197
reserve
mln::p_array, 907
mln::p_mutable_array_of, 947
mln::p_queue_fast, 966
mln::topo::n_faces_set, 1086

mln::util::array, 1104
resize
mln::border, 176
mln::util::array, 1104
result
mln::graph::attribute::card_t, 788
mln::graph::attribute::representative_t, 789
mln::topo::is_simple_cell, 1075
mln::util::array, 1102
revert
mln::arith, 168
revert_inplace
mln::arith, 168
rgb
mln::value::rgb, 1206, 1207
rgb16
mln::value, 447
rgb8
mln::value, 447
rgb8_2complex_image3df
mln, 129
root
mln::util::tree, 1162
rotate
mln::geom, 262
rotation
mln::fun::x2x::rotation, 757
Routines, 75
run_length
mln::box_runstart_piter, 598
rvalue
mln::complex_image, 624
mln::decorated_image, 638
mln::doc::Fastest_Image, 652
mln::doc::Generalized_Pixel, 657
mln::doc::Image, 662
mln::doc::Pixel_Iterator, 671
mln::extension_fun, 711
mln::extension_ima, 714
mln::extension_val, 717
mln::flat_image, 720
mln::fun_image, 763
mln::hexa, 812
mln::image1d, 819
mln::image2d, 824
mln::image2d_h, 829
mln::image3d, 832
mln::interpolated, 837
mln::lazy_image, 843
mln::thrubin_image, 1030
mln::tr_image, 1092
mln::value::stack_image, 1212

S

mln::p_image, 930
sagittal_dec
mln, 141
saturate
mln::data, 197, 198
saturate_inplace
mln::data, 198
save
mln::io::cloud, 276
mln::io::dump, 278
mln::io::magick, 280
mln::io::off, 281
mln::io::pbm, 284
mln::io::pfm, 289
mln::io::pgm, 292
mln::io::plot, 294
mln::io::pnm, 297
mln::io::ppm, 301
mln::io::txt, 304
save_bin_alt
mln::io::off, 282
search
mln::util::tree_node, 1166
search_rec
mln::util::tree_node, 1166
second
mln::util::couple, 1113
mln::util::ord_pair, 1144
mln::util::site_pair, 1154
seeds2tiling
mln::geom, 262
mln::geom::impl, 264
seeds2tiling_roundness
mln::geom, 263
mln::geom::impl, 264
segment1d
modwin1d, 94
set
mln::util::set, 1149
set_all
mln::dpoint, 692
mln::point, 1003
set_alpha
mln::fun::x2x::rotation, 758
set_axis
mln::fun::x2x::rotation, 758
set_cplx
mln::topo::algebraic_face, 1053
mln::topo::algebraic_n_face, 1057
mln::topo::face, 1068
mln::topo::n_face, 1079
set_face_id
mln::topo::algebraic_face, 1053
mln::topo::algebraic_n_face, 1057

mln::topo::face, 1068
mln::topo::n_face, 1079
set_image
mln::topo::is_simple_cell, 1075
set_n
mln::topo::algebraic_face, 1053
mln::topo::face, 1068
set_nbits
mln::value::float01, 1175
set_parent
mln::util::tree_node, 1166
set_sign
mln::topo::algebraic_face, 1053
mln::topo::algebraic_n_face, 1057
set_t
mln::fun::x2x::translation, 761
set_tr
mln::tr_image, 1093
set_value
mln::accu::count_adjacent_vertices, 462
mln::accu::count_labels, 463
mln::accu::count_value, 465
mln::accu::math::count, 481
mln::accu::shape::height, 536
mln::accu::shape::volume, 538
mln::morpho::attribute::sum, 893
sign
mln::topo::algebraic_face, 1053
mln::topo::algebraic_n_face, 1057
mln::value::sign, 1210
site
mln::box, 591
mln::doc::Box, 645
mln::doc::Site_Set, 679
mln::dpoint, 691
mln::graph_elt_window, 797
mln::graph_elt_window_if, 801
mln::graph_window_base, 804
mln::p_centered, 909
mln::tr_image, 1092
Site sets, 86
site_function_t
mln::edge_image, 707
mln::vertex_image, 1216
site_set
mln::complex_psite, 632
size
mln::util::array, 1104
mln::win::line, 1230
mln::window, 1241
skeleton
mln::complex_image, 624
mln::decorated_image, 638
mln::doc::Fastest_Image, 652
mln::doc::Image, 662
mln::edge_image, 707
mln::extended, 708
mln::extension_fun, 711
mln::extension_ima, 714
mln::extension_val, 717
mln::flat_image, 720
mln::fun_image, 763
mln::hexa, 812
mln::image1d, 819
mln::image2d, 824
mln::image2d_h, 829
mln::image3d, 833
mln::interpolated, 837
mln::labeled_image, 840
mln::lazy_image, 843
mln::p2p_image, 901
mln::plain, 993
mln::pw::image, 1008
mln::safe_image, 1012
mln::sub_image, 1025
mln::sub_image_if, 1027
mln::thrubin_image, 1030
mln::tr_image, 1092
mln::transformed_image, 1094
mln::value::stack_image, 1212
mln::vertex_image, 1216
slices_2d
mln::debug, 218
soft_heap
mln::util::soft_heap, 1156
sort_offsets_increasing
mln::data, 198
mln::data::impl::generic, 210
sort_psites_decreasing
mln::data, 198
sort_psites_increasing
mln::data, 199
space_2complex_geometry
mln, 129
Sparse types, 90
sphere3d
modwin3d, 98
sqr_l2
mln::norm, 399
stack
mln::value, 452
stack_image
mln::value::stack_image, 1212
standard_deviation
mln::accu::stat::variance, 571
start
mln::doc::Iterator, 665
mln::doc::Pixel_Iterator, 671

mln::doc::Site_Iterator, 677
mln::doc::Value_Iterator, 681
mln::dpoints_bkd_pixter, 696
mln::dpoints_fwd_pixter, 699
mln::p_run, 970
mln::topo::face_bkd_iter, 1069
mln::topo::face_fwd_iter, 1071
mln::topo::n_face_bkd_iter, 1082
mln::topo::n_face_fwd_iter, 1083
mln::topo::static_n_face_bkd_iter, 1088
mln::topo::static_n_face_fwd_iter, 1090
mln::util::branch_iter, 1109
mln::util::branch_iter_ind, 1111
static_n_face_bkd_iter
mln::topo::static_n_face_bkd_iter, 1087
static_n_face_fwd_iter
mln::topo::static_n_face_fwd_iter, 1089
std_deque
mln::p_queue, 961
std_vector
mln::p_array, 907
mln::p_line2d, 943
mln::p_queue_fast, 966
mln::p_set, 975
mln::util::array, 1105
mln::util::set, 1151
mln::w_window, 1219
mln::win::rectangle2d, 1236
mln::window, 1241
stretch
mln::data, 199
mln::data::impl, 206
structural
mln::morpho::closing::approx, 377
mln::morpho::opening::approx, 382
sub_image
mln::sub_image, 1025
sub_image_if
mln::sub_image_if, 1027
subdomain
mln::labeled_image, 841
subsampling
mln::subsampling, 412
subtractive
mln::morpho::tree::filter, 392
sum
mln::accu::stat::variance, 571
mln::estim, 230
superpose
mln::debug, 219
mln::morpho::watershed, 395
sym
mln::doc::Weighted_Window, 686
mln::graph_elt_window, 797
mln::graph_elt_window_if, 803
mln::graph_window_base, 805
mln::w_window, 1219
mln::win, 456
mln::window, 1242
sym_diff
mln::Site_Set, 1022

t
mln::algebra::h_mat, 579
mln::algebra::h_vec, 581
mln::fun::x2x::translation, 761
take
mln::accu, 145
mln::accu::histo, 467
mln::accu::label_used, 469
mln::accu::stat::median_alt, 550
mln::doc::Accumulator, 642
take_as_init
mln::accu::center, 458
mln::accu::convolve, 459
mln::accu::count_adjacent_vertices, 462
mln::accu::count_labels, 464
mln::accu::count_value, 466
mln::accu::histo, 467
mln::accu::label_used, 470
mln::accu::logic::land, 471
mln::accu::logic::lor, 475
mln::accu::logic::lor_basic, 478
mln::accu::maj_h, 479
mln::accu::math::count, 482
mln::accu::math::inf, 483
mln::accu::math::sum, 485
mln::accu::math::sup, 487
mln::accu::max_site, 489
mln::accu::nil, 525
mln::accu::p, 527
mln::accu::pair, 530
mln::accu::rms, 531
mln::accu::shape::bbox, 533
mln::accu::shape::height, 536
mln::accu::shape::volume, 538
mln::accu::site_set::rectangularity, 541
mln::accu::stat::deviation, 542
mln::accu::stat::max, 544
mln::accu::stat::max_h, 546
mln::accu::stat::mean, 548
mln::accu::stat::median_alt, 551
mln::accu::stat::median_h, 553
mln::accu::stat::min, 555
mln::accu::stat::min_h, 557
mln::accu::stat::min_max, 560
mln::accu::stat::rank, 562
mln::accu::stat::rank< bool >, 563

mln::accu::stat::rank_high_quant, 565
mln::accu::stat::var, 568
mln::accu::stat::variance, 571
mln::accu::tuple, 573
mln::accu::val, 575
mln::Accumulator, 577
mln::morpho::attribute::card, 884
mln::morpho::attribute::count_adjacent_-vertices, 886
mln::morpho::attribute::height, 889
mln::morpho::attribute::sharpness, 891
mln::morpho::attribute::sum, 893
mln::morpho::attribute::volume, 895
take_n_times
mln::accu::center, 458
mln::accu::convolve, 460
mln::accu::count_adjacent_vertices, 462
mln::accu::count_labels, 464
mln::accu::count_value, 466
mln::accu::histo, 468
mln::accu::label_used, 470
mln::accu::logic::land, 471
mln::accu::logic::lor, 475
mln::accu::logic::lor_basic, 478
mln::accu::maj_h, 480
mln::accu::math::count, 482
mln::accu::math::inf, 484
mln::accu::math::sum, 486
mln::accu::math::sup, 488
mln::accu::max_site, 489
mln::accu::nil, 525
mln::accu::p, 528
mln::accu::pair, 530
mln::accu::rms, 532
mln::accu::shape::bbox, 534
mln::accu::shape::height, 536
mln::accu::shape::volume, 538
mln::accu::site_set::rectangularity, 541
mln::accu::stat::deviation, 543
mln::accu::stat::max, 545
mln::accu::stat::max_h, 546
mln::accu::stat::mean, 549
mln::accu::stat::median_alt, 551
mln::accu::stat::median_h, 553
mln::accu::stat::min, 556
mln::accu::stat::min_h, 557
mln::accu::stat::min_max, 560
mln::accu::stat::rank, 562
mln::accu::stat::rank< bool >, 563
mln::accu::stat::rank_high_quant, 566
mln::accu::stat::var, 568
mln::accu::stat::variance, 571
mln::accu::tuple, 574
mln::accu::val, 575
mln::Accumulator, 577
mln::morpho::attribute::card, 884
mln::morpho::attribute::count_adjacent_-vertices, 887
mln::morpho::attribute::height, 889
mln::morpho::attribute::sharpness, 891
mln::morpho::attribute::sum, 893
mln::morpho::attribute::volume, 895
target
mln::graph_elt_window, 797
mln::graph_elt_window_if, 801
teal
mln::literal, 333
the
mln::value::set, 1208
thick_miss
mln::morpho, 373
thickening
mln::morpho, 373
thin_fit
mln::morpho, 373
thinning
mln::morpho, 374
threshold
mln::binarization, 173
times
mln::arith, 169
times_cst
mln::arith, 169
times_inplace
mln::arith, 169
to
mln::convert, 188
to_dpoint
mln::convert, 188
mln::Dpoint, 694
to_enc
mln::data, 199
to_float
mln::value::graylevel, 1181
to_fun
mln::convert, 188
to_h_vec
mln::point, 1003
to_image
mln::convert, 188
to_larger
mln::box, 593
to_nbits
mln::value::float01, 1175
to_neighb
mln::graph, 267
to_p_array
mln::convert, 188, 189

to_p_set
mln::convert, 189
to_point
mln::doc::Point_Site, 674
mln::Point, 996
to_result
mln::accu::center, 458
mln::accu::convolve, 460
mln::accu::count_adjacent_vertices, 462
mln::accu::count_labels, 464
mln::accu::count_value, 466
mln::accu::label_used, 470
mln::accu::logic::land, 472
mln::accu::logic::land_basic, 473
mln::accu::logic::lor, 476
mln::accu::logic::lor_basic, 478
mln::accu::maj_h, 480
mln::accu::math::count, 482
mln::accu::math::inf, 484
mln::accu::math::sum, 486
mln::accu::math::sup, 488
mln::accu::max_site, 490
mln::accu::nil, 526
mln::accu::p, 528
mln::accu::pair, 530
mln::accu::rms, 532
mln::accu::shape::bbox, 534
mln::accu::shape::height, 536
mln::accu::shape::volume, 538
mln::accu::site_set::rectangularity, 541
mln::accu::stat::deviation, 543
mln::accu::stat::max, 545
mln::accu::stat::max_h, 547
mln::accu::stat::mean, 549
mln::accu::stat::median_alt, 551
mln::accu::stat::median_h, 553
mln::accu::stat::min, 556
mln::accu::stat::min_h, 558
mln::accu::stat::min_max, 560
mln::accu::stat::rank, 562
mln::accu::stat::rank< bool >, 564
mln::accu::stat::rank_high_quant, 566
mln::accu::stat::var, 568
mln::accu::stat::variance, 571
mln::accu::tuple, 574
mln::accu::val, 576
mln::morpho::attribute::card, 885
mln::morpho::attribute::count_adjacent_-vertices, 887
mln::morpho::attribute::height, 889
mln::morpho::attribute::sharpness, 891
mln::morpho::attribute::sum, 893
mln::morpho::attribute::volume, 895
to_upper_window
mln::convert, 189, 190
to_value
mln::value::proxy, 1202
mln::value::proxy< const I >, 1204
to_vec
mln::algebra::h_vec, 581
mln::dpoint, 693
mln::point, 1003
to_win
mln::graph, 267
to_window
mln::convert, 190
toggle
mln::p_image, 931
top_hat_black
mln::morpho, 374
mln::morpho::elementary, 379
top_hat_self_complementary
mln::morpho, 374
mln::morpho::elementary, 379
top_hat_white
mln::morpho, 374
mln::morpho::elementary, 379
tr
mln::tr_image, 1093
tr_image
mln::tr_image, 1092
tracked_ptr
mln::util::tracked_ptr, 1159
transform
mln::data, 200
mln::data::impl::generic, 210
transform_inplace
mln::data, 201
mln::data::impl::generic, 211
transform_inplace_lowq
mln::data::impl, 206
transformed_image
mln::transformed_image, 1095
translation
mln::fun::x2x::translation, 760
tree
mln::util::tree, 1161
tree_fast_to_image
mln::util, 439
mln::util::impl, 441
tree_node
mln::util::tree_node, 1164
tree_to_fast
mln::util, 439
tree_to_image
mln::util, 440
Types, 73

uni
mln::Site_Set, 1022
unique
mln::Site_Set, 1023
unproject_image
mln::unproject_image, 1096
untake
mln::morpho::attribute::sum, 893
up
mln, 141
update
mln::data, 202
mln::data::impl::generic, 211
mln::dpoints_bkd_pixter, 697
mln::dpoints_fwd_pixter, 700
update_fastest
mln::data::impl, 207
update_id
mln::util::edge, 1117
mln::util::vertex, 1171
util_set
mln::p_set, 975
util_tree
mln::util::branch, 1106
Utilities, 92

v
mln::util::pix, 1146
v1
mln::util::edge, 1118
mln::util::graph, 1126
mln::util::line_graph, 1136
v2
mln::util::edge, 1118
mln::util::graph, 1127
mln::util::line_graph, 1137
v2w2v functions, 102
v2w_w2v functions, 103
v_ith_nbh_edge
mln::util::graph, 1127
mln::util::line_graph, 1137
v_ith_nbh_vertex
mln::util::graph, 1127
mln::util::line_graph, 1137
v_nmax
mln::util::graph, 1127
mln::util::line_graph, 1137
v_other
mln::util::edge, 1118
val
mln::doc::Generalized_Pixel, 658
mln::doc::Pixel_Iterator, 672
value
mln::accu::shape::height, 536
mln::accu::shape::volume, 538
mln::complex_image, 624
mln::doc::Fastest_Image, 652
mln::doc::Generalized_Pixel, 658
mln::doc::Image, 662
mln::doc::Pixel_Iterator, 671
mln::doc::Value_Iterator, 681
mln::doc::Value_Set, 683
mln::extended, 708
mln::extension_fun, 711
mln::extension_ima, 714
mln::extension_val, 717
mln::flat_image, 720
mln::fun_image, 763
mln::hexa, 812
mln::image1d, 819
mln::image2d, 825
mln::image2d_h, 830
mln::image3d, 833
mln::interpolated, 837
mln::labeling, 316
mln::p_vaccess, 983
mln::thrubin_image, 1031
mln::tr_image, 1092
mln::util::pix, 1145
mln::value::float01, 1176
mln::value::float01_f, 1177
mln::value::graylevel, 1181
mln::value::graylevel_f, 1184
mln::value::lut_vec, 1198
mln::value::stack_image, 1212
value_ind
mln::value::float01, 1176
value_t
mln::util::object_id, 1141
values
mln::complex_image, 625
mln::doc::Fastest_Image, 656
mln::doc::Image, 664
mln::p_vaccess, 984
Values morphers, 70
var
mln::accu::stat::variance, 572
variance
mln::accu::stat::var, 568
vec
mln::dpoint, 691
mln::make, 356, 357
mln::point, 1001
vec2d_d
mln, 129
vec2d_f
mln, 129
vec3d_d

mln, 129
vec3d_f
mln, 130
vect
mln::accu::histo, 468
vertex
mln::p_vertices, 987
mln::util::graph, 1127
mln::util::line_graph, 1137
mln::util::vertex, 1169
vertex_id_t
mln::util, 437
vertex_image
mln::make, 357, 358
mln::vertex_image, 1216
vertex_nbh_edge_fwd_iter
mln::util::graph, 1124
mln::util::line_graph, 1135
vertex_nbh_vertex_fwd_iter
mln::util::graph, 1124
mln::util::line_graph, 1135
vertices_t
mln::util::graph, 1124
mln::util::line_graph, 1135
violet
mln::literal, 333
vline2d
modwin2d, 96
volume
mln::morpho::attribute::sharpness, 891
mln::win::cuboid3d, 1227
voronoi
mln::make, 358
vprod
mln::algebra, 158
vset
mln::doc::Fastest_Image, 652
mln::doc::Image, 662
mln::p_vaccess, 983
vv2b functions, 104

w
mln::w_window, 1219
w_window
mln::make, 358
mln::w_window, 1218
w_window1d
mln::make, 359
w_window2d
mln::make, 359
w_window3d
mln::make, 359
w_window_directional
mln::make, 360
weight
mln::doc::Weighted_Window, 685
mln::w_window, 1218
weights
mln::w_window, 1219
white
mln::literal, 333
width
mln::win::cuboid3d, 1227
mln::win::rectangle2d, 1236
win
mln::doc::Weighted_Window, 686
mln::graph_elt_neighborhood, 791
mln::graph_elt_neighborhood_if, 794
mln::neighb, 897
mln::w_window, 1219
win_c4p
modwin2d, 96
win_c4p_3d
modwin3d, 99
win_c8p
modwin2d, 96
win_c8p_3d
modwin3d, 99
win_t
mln::edge_image, 707
mln::vertex_image, 1216
window
mln::doc::Weighted_Window, 685
mln::p_centered, 910
mln::window, 1240
window1d
modwin1d, 94
window2d
modwin2d, 96
window3d
modwin3d, 98
Windows, 93
wrap
mln::labeling, 317

xor_inplace
mln::logical, 336

yellow
mln::literal, 333

zero
mln::algebra::h_vec, 581
mln::literal, 333
mln::value::int_s, 1186
mln::value::int_u_sat, 1191
mln::value::rgb, 1207
mln::value::sign, 1210