

**Milena (Olena)**  
User documentation 1.0a Id

Generated by Doxygen 1.7.1

Fri Oct 19 2012 04:15:33



# Contents

<b>1</b>	<b>Documentation of milena</b>	<b>1</b>
1.1	Introduction . . . . .	1
1.2	Overview of Milena. . . . .	1
1.3	Copyright and License. . . . .	2
<b>2</b>	<b>Quick Reference Guide</b>	<b>3</b>
<b>3</b>	<b>Tutorial</b>	<b>5</b>
<b>4</b>	<b>Module Index</b>	<b>7</b>
4.1	Modules . . . . .	7
<b>5</b>	<b>Namespace Index</b>	<b>9</b>
5.1	Namespace List . . . . .	9
<b>6</b>	<b>Class Index</b>	<b>13</b>
6.1	Class Hierarchy . . . . .	13
<b>7</b>	<b>Class Index</b>	<b>57</b>
7.1	Class List . . . . .	57
<b>8</b>	<b>Module Documentation</b>	<b>67</b>
8.1	On site sets . . . . .	67
8.1.1	Detailed Description . . . . .	67
8.2	On images . . . . .	67
8.2.1	Detailed Description . . . . .	68
8.3	On values . . . . .	68
8.3.1	Detailed Description . . . . .	70
8.4	Multiple accumulators . . . . .	70
8.4.1	Detailed Description . . . . .	70
8.5	Graphes . . . . .	70

8.5.1	Detailed Description	70
8.6	Images	71
8.6.1	Detailed Description	71
8.7	Basic types	71
8.7.1	Detailed Description	72
8.8	Image morphers	72
8.9	Values morphers	72
8.9.1	Detailed Description	72
8.10	Domain morphers	73
8.10.1	Detailed Description	73
8.11	Identity morphers	74
8.11.1	Detailed Description	74
8.12	Types	74
8.12.1	Detailed Description	75
8.13	Accumulators	75
8.13.1	Detailed Description	75
8.14	Routines	75
8.15	Canvas	75
8.16	Functions	76
8.16.1	Detailed Description	77
8.17	Neighborhoods	77
8.17.1	Detailed Description	77
8.18	1D neighborhoods	77
8.18.1	Detailed Description	78
8.18.2	Typedef Documentation	78
8.18.2.1	neighb1d	78
8.18.3	Function Documentation	78
8.18.3.1	c2	78
8.19	2D neighborhoods	78
8.19.1	Detailed Description	79
8.19.2	Typedef Documentation	79
8.19.2.1	neighb2d	79
8.19.3	Function Documentation	79
8.19.3.1	c2_col	79
8.19.3.2	c2_row	79
8.19.3.3	c4	79

---

8.19.3.4	c8	80
8.20	3D neighborhoods	80
8.20.1	Detailed Description	80
8.20.2	Typedef Documentation	81
8.20.2.1	neighb3d	81
8.20.3	Function Documentation	81
8.20.3.1	c18	81
8.20.3.2	c26	81
8.20.3.3	c2_3d_sli	82
8.20.3.4	c4_3d	82
8.20.3.5	c6	82
8.20.3.6	c8_3d	83
8.21	Site sets	83
8.21.1	Detailed Description	84
8.22	Basic types	84
8.22.1	Detailed Description	84
8.23	Graph based	84
8.23.1	Detailed Description	85
8.24	Complex based	85
8.24.1	Detailed Description	85
8.25	Sparse types	85
8.25.1	Detailed Description	86
8.26	Queue based	86
8.26.1	Detailed Description	86
8.27	Utilities	86
8.27.1	Detailed Description	87
8.28	Windows	87
8.28.1	Detailed Description	88
8.29	1D windows	88
8.29.1	Detailed Description	88
8.29.2	Typedef Documentation	88
8.29.2.1	segmentId	88
8.29.2.2	windowId	89
8.30	2D windows	89
8.30.1	Detailed Description	90
8.30.2	Typedef Documentation	90

8.30.2.1	disk2d	90
8.30.2.2	hline2d	90
8.30.2.3	vline2d	90
8.30.2.4	window2d	90
8.30.3	Function Documentation	90
8.30.3.1	win_c4p	90
8.30.3.2	win_c8p	91
8.31	3D windows	91
8.31.1	Detailed Description	92
8.31.2	Typedef Documentation	92
8.31.2.1	sline3d	92
8.31.2.2	sphere3d	92
8.31.2.3	window3d	92
8.31.3	Function Documentation	92
8.31.3.1	win_c4p_3d	92
8.31.3.2	win_c8p_3d	93
8.32	N-D windows	93
8.32.1	Detailed Description	94
8.33	Multiple windows	94
8.33.1	Detailed Description	94
8.34	v2w2v functions	94
8.35	v2w_w2v functions	94
8.36	vv2b functions	94
<b>9</b>	<b>Namespace Documentation</b>	<b>95</b>
9.1	mln Namespace Reference	95
9.1.1	Detailed Description	117
9.1.2	Typedef Documentation	119
9.1.2.1	bin_1complex_image2d	119
9.1.2.2	bin_2complex_image3df	119
9.1.2.3	box1d	119
9.1.2.4	box2d	120
9.1.2.5	box2d_h	120
9.1.2.6	box3d	120
9.1.2.7	discrete_plane_1complex_geometry	120
9.1.2.8	discrete_plane_2complex_geometry	120
9.1.2.9	dpoint1d	120

---

9.1.2.10	<a href="#">dpoint2d</a>	120
9.1.2.11	<a href="#">dpoint2d_h</a>	120
9.1.2.12	<a href="#">dpoint3d</a>	121
9.1.2.13	<a href="#">float_2complex_image3df</a>	121
9.1.2.14	<a href="#">int_u8_1complex_image2d</a>	121
9.1.2.15	<a href="#">int_u8_2complex_image2d</a>	121
9.1.2.16	<a href="#">int_u8_2complex_image3df</a>	121
9.1.2.17	<a href="#">p_run2d</a>	121
9.1.2.18	<a href="#">p_runs2d</a>	121
9.1.2.19	<a href="#">point1d</a>	121
9.1.2.20	<a href="#">point1df</a>	121
9.1.2.21	<a href="#">point2d</a>	121
9.1.2.22	<a href="#">point2d_h</a>	122
9.1.2.23	<a href="#">point2df</a>	122
9.1.2.24	<a href="#">point3d</a>	122
9.1.2.25	<a href="#">point3df</a>	122
9.1.2.26	<a href="#">rgb8_2complex_image3df</a>	122
9.1.2.27	<a href="#">space_2complex_geometry</a>	122
9.1.2.28	<a href="#">unsigned_2complex_image3df</a>	122
9.1.2.29	<a href="#">vec2d_d</a>	122
9.1.2.30	<a href="#">vec2d_f</a>	122
9.1.2.31	<a href="#">vec3d_d</a>	122
9.1.2.32	<a href="#">vec3d_f</a>	123
9.1.2.33	<a href="#">w_window1d_float</a>	123
9.1.2.34	<a href="#">w_window1d_int</a>	123
9.1.2.35	<a href="#">w_window2d_float</a>	123
9.1.2.36	<a href="#">w_window2d_int</a>	123
9.1.2.37	<a href="#">w_window3d_float</a>	123
9.1.2.38	<a href="#">w_window3d_int</a>	123
9.1.3	<a href="#">Function Documentation</a>	123
9.1.3.1	<a href="#">a_point_of</a>	123
9.1.3.2	<a href="#">apply_p2p</a>	123
9.1.3.3	<a href="#">apply_p2p</a>	124
9.1.3.4	<a href="#">compose</a>	124
9.1.3.5	<a href="#">duplicate</a>	124
9.1.3.6	<a href="#">extend</a>	124

9.1.3.7	<code>extend</code>	124
9.1.3.8	<code>extend</code>	125
9.1.3.9	<code>implies</code>	125
9.1.3.10	<code>initialize</code>	125
9.1.3.11	<code>is_simple_2d</code>	125
9.1.3.12	<code>larger_than</code>	125
9.1.3.13	<code>make_debug_graph_image</code>	126
9.1.3.14	<code>mln_exact</code>	126
9.1.3.15	<code>mln_gen_complex_neighborhood</code>	126
9.1.3.16	<code>mln_gen_complex_neighborhood</code>	126
9.1.3.17	<code>mln_gen_complex_neighborhood</code>	126
9.1.3.18	<code>mln_gen_complex_neighborhood</code>	126
9.1.3.19	<code>mln_gen_complex_neighborhood</code>	127
9.1.3.20	<code>mln_gen_complex_neighborhood</code>	127
9.1.3.21	<code>mln_gen_complex_window</code>	127
9.1.3.22	<code>mln_gen_complex_window</code>	127
9.1.3.23	<code>mln_gen_complex_window</code>	127
9.1.3.24	<code>mln_gen_complex_window</code>	127
9.1.3.25	<code>mln_gen_complex_window</code>	127
9.1.3.26	<code>mln_gen_complex_window</code>	127
9.1.3.27	<code>mln_gen_complex_window_p</code>	127
9.1.3.28	<code>mln_gen_complex_window_p</code>	128
9.1.3.29	<code>mln_gen_complex_window_p</code>	128
9.1.3.30	<code>mln_gen_complex_window_p</code>	128
9.1.3.31	<code>mln_gen_complex_window_p</code>	128
9.1.3.32	<code>mln_gen_complex_window_p</code>	128
9.1.3.33	<code>mln_regular</code>	128
9.1.3.34	<code>mln_trait_op_geq</code>	128
9.1.3.35	<code>mln_trait_op_greater</code>	129
9.1.3.36	<code>mln_trait_op_leq</code>	129
9.1.3.37	<code>mln_trait_op_neq</code>	129
9.1.3.38	<code>operator!=</code>	130
9.1.3.39	<code>operator!=</code>	130
9.1.3.40	<code>operator*</code>	130
9.1.3.41	<code>operator++</code>	130
9.1.3.42	<code>operator-</code>	130



---

9.1.3.43	operator-	130
9.1.3.44	operator--	131
9.1.3.45	operator<	131
9.1.3.46	operator<	131
9.1.3.47	operator<	131
9.1.3.48	operator<<	132
9.1.3.49	operator<<	132
9.1.3.50	operator<<	132
9.1.3.51	operator<<	132
9.1.3.52	operator<=	132
9.1.3.53	operator<=	132
9.1.3.54	operator<=	132
9.1.3.55	operator<=	132
9.1.3.56	operator<=	133
9.1.3.57	operator==	133
9.1.3.58	operator==	133
9.1.3.59	operator==	133
9.1.3.60	operator==	133
9.1.3.61	operator==	134
9.1.3.62	operator==	134
9.1.3.63	operator==	134
9.1.3.64	operator	134
9.1.3.65	operator	134
9.1.3.66	operator	135
9.1.3.67	operator	135
9.1.3.68	operator	135
9.1.3.69	operator	135
9.1.3.70	primary	135
9.1.3.71	ptransform	135
9.1.4	Variable Documentation	136
9.1.4.1	before	136
9.1.4.2	sagittal_dec	136
9.1.4.3	up	136
9.2	mln::accu Namespace Reference	136
9.2.1	Detailed Description	138
9.2.2	Function Documentation	138

9.2.2.1	compute	138
9.2.2.2	line	138
9.2.2.3	mln_meta_accu_result	139
9.2.2.4	take	139
9.3	mln::accu::image Namespace Reference	139
9.3.1	Detailed Description	139
9.4	mln::accu::impl Namespace Reference	139
9.4.1	Detailed Description	140
9.5	mln::accu::logic Namespace Reference	140
9.5.1	Detailed Description	140
9.6	mln::accu::math Namespace Reference	140
9.6.1	Detailed Description	141
9.7	mln::accu::meta::logic Namespace Reference	141
9.7.1	Detailed Description	141
9.8	mln::accu::meta::math Namespace Reference	141
9.8.1	Detailed Description	142
9.9	mln::accu::meta::shape Namespace Reference	142
9.9.1	Detailed Description	142
9.10	mln::accu::meta::stat Namespace Reference	142
9.10.1	Detailed Description	143
9.11	mln::accu::shape Namespace Reference	143
9.11.1	Detailed Description	143
9.12	mln::accu::stat Namespace Reference	143
9.12.1	Detailed Description	144
9.13	mln::algebra Namespace Reference	144
9.13.1	Detailed Description	145
9.13.2	Function Documentation	145
9.13.2.1	ldlt_decomp	145
9.13.2.2	ldlt_solve	146
9.13.2.3	operator*	146
9.13.2.4	vprod	146
9.14	mln::arith Namespace Reference	146
9.14.1	Detailed Description	148
9.14.2	Function Documentation	149
9.14.2.1	diff_abs	149
9.14.2.2	div	149

---

9.14.2.3	<code>div_cst</code>	149
9.14.2.4	<code>div_inplace</code>	150
9.14.2.5	<code>min</code>	150
9.14.2.6	<code>min_inplace</code>	150
9.14.2.7	<code>minus</code>	151
9.14.2.8	<code>minus</code>	151
9.14.2.9	<code>minus</code>	151
9.14.2.10	<code>minus_cst</code>	152
9.14.2.11	<code>minus_cst</code>	152
9.14.2.12	<code>minus_cst_inplace</code>	152
9.14.2.13	<code>minus_inplace</code>	153
9.14.2.14	<code>plus</code>	153
9.14.2.15	<code>plus</code>	154
9.14.2.16	<code>plus</code>	154
9.14.2.17	<code>plus_cst</code>	154
9.14.2.18	<code>plus_cst</code>	155
9.14.2.19	<code>plus_cst</code>	155
9.14.2.20	<code>plus_cst_inplace</code>	155
9.14.2.21	<code>plus_inplace</code>	156
9.14.2.22	<code>revert</code>	156
9.14.2.23	<code>revert_inplace</code>	156
9.14.2.24	<code>times</code>	157
9.14.2.25	<code>times_cst</code>	157
9.14.2.26	<code>times_inplace</code>	157
9.15	<code>mln::arith::impl</code> Namespace Reference	158
9.15.1	Detailed Description	158
9.16	<code>mln::arith::impl::generic</code> Namespace Reference	158
9.16.1	Detailed Description	158
9.17	<code>mln::binarization</code> Namespace Reference	158
9.17.1	Detailed Description	159
9.17.2	Function Documentation	159
9.17.2.1	<code>binarization</code>	159
9.17.2.2	<code>threshold</code>	159
9.18	<code>mln::border</code> Namespace Reference	159
9.18.1	Detailed Description	160
9.18.2	Function Documentation	160

9.18.2.1	adjust	160
9.18.2.2	duplicate	160
9.18.2.3	equalize	161
9.18.2.4	fill	161
9.18.2.5	find	161
9.18.2.6	get	162
9.18.2.7	mirror	162
9.18.2.8	resize	162
9.19	mln::border::impl Namespace Reference	163
9.19.1	Detailed Description	163
9.20	mln::border::impl::generic Namespace Reference	163
9.20.1	Detailed Description	163
9.21	mln::canvas Namespace Reference	163
9.21.1	Detailed Description	164
9.21.2	Function Documentation	164
9.21.2.1	distance_front	164
9.21.2.2	distance_geodesic	164
9.22	mln::canvas::browsing Namespace Reference	165
9.22.1	Detailed Description	165
9.23	mln::canvas::impl Namespace Reference	165
9.23.1	Detailed Description	166
9.24	mln::canvas::labeling Namespace Reference	166
9.24.1	Detailed Description	166
9.24.2	Function Documentation	166
9.24.2.1	blobs	166
9.25	mln::canvas::labeling::impl Namespace Reference	167
9.25.1	Detailed Description	167
9.26	mln::canvas::morpho Namespace Reference	167
9.26.1	Detailed Description	167
9.27	mln::convert Namespace Reference	167
9.27.1	Detailed Description	169
9.27.2	Function Documentation	170
9.27.2.1	from_to	170
9.27.2.2	from_to	170
9.27.2.3	from_to	170
9.27.2.4	from_to	170

---

9.27.2.5	<code>mln_image_from_grid</code>	170
9.27.2.6	<code>mln_image_from_grid</code>	170
9.27.2.7	<code>mln_image_from_grid</code>	170
9.27.2.8	<code>mln_image_from_grid</code>	170
9.27.2.9	<code>mln_window</code>	170
9.27.2.10	<code>to</code>	171
9.27.2.11	<code>to_dpoint</code>	171
9.27.2.12	<code>to_fun</code>	171
9.27.2.13	<code>to_image</code>	171
9.27.2.14	<code>to_p_array</code>	171
9.27.2.15	<code>to_p_array</code>	171
9.27.2.16	<code>to_p_array</code>	171
9.27.2.17	<code>to_p_set</code>	171
9.27.2.18	<code>to_p_set</code>	172
9.27.2.19	<code>to_p_set</code>	172
9.27.2.20	<code>to_p_set</code>	172
9.27.2.21	<code>to_p_set</code>	172
9.27.2.22	<code>to_qimage</code>	172
9.27.2.23	<code>to_upper_window</code>	172
9.27.2.24	<code>to_upper_window</code>	172
9.27.2.25	<code>to_window</code>	172
9.27.2.26	<code>to_window</code>	173
9.27.2.27	<code>to_window</code>	173
9.27.3	Variable Documentation	173
9.27.3.1	<code>to_fun</code>	173
9.28	<code>mln::data</code> Namespace Reference	173
9.28.1	Detailed Description	175
9.28.2	Function Documentation	175
9.28.2.1	<code>abs</code>	175
9.28.2.2	<code>abs_inplace</code>	176
9.28.2.3	<code>apply</code>	176
9.28.2.4	<code>compute</code>	176
9.28.2.5	<code>compute</code>	176
9.28.2.6	<code>convert</code>	177
9.28.2.7	<code>fast_median</code>	177
9.28.2.8	<code>fill</code>	177

9.28.2.9	<code>fill_with_image</code>	178
9.28.2.10	<code>fill_with_value</code>	178
9.28.2.11	<code>median</code>	178
9.28.2.12	<code>mln_meta_accu_result</code>	179
9.28.2.13	<code>paste</code>	179
9.28.2.14	<code>paste_without_localization</code>	179
9.28.2.15	<code>replace</code>	180
9.28.2.16	<code>saturate</code>	180
9.28.2.17	<code>saturate</code>	180
9.28.2.18	<code>saturate_inplace</code>	180
9.28.2.19	<code>sort_offsets_increasing</code>	181
9.28.2.20	<code>sort_psites_decreasing</code>	181
9.28.2.21	<code>sort_psites_increasing</code>	181
9.28.2.22	<code>stretch</code>	181
9.28.2.23	<code>to_enc</code>	182
9.28.2.24	<code>transform</code>	182
9.28.2.25	<code>transform</code>	182
9.28.2.26	<code>transform_inplace</code>	183
9.28.2.27	<code>transform_inplace</code>	183
9.28.2.28	<code>update</code>	183
9.28.2.29	<code>wrap</code>	183
9.29	<code>mln::data::approx</code> Namespace Reference	184
9.29.1	Detailed Description	184
9.29.2	Function Documentation	184
9.29.2.1	<code>median</code>	184
9.29.2.2	<code>median</code>	185
9.29.2.3	<code>median</code>	185
9.30	<code>mln::data::approx::impl</code> Namespace Reference	185
9.30.1	Detailed Description	185
9.31	<code>mln::data::impl</code> Namespace Reference	185
9.31.1	Detailed Description	186
9.31.2	Function Documentation	186
9.31.2.1	<code>paste_without_localization_fast</code>	186
9.31.2.2	<code>paste_without_localization_fastest</code>	187
9.31.2.3	<code>paste_without_localization_lines</code>	187
9.31.2.4	<code>stretch</code>	188

---

9.31.2.5	transform_inplace_lowq . . . . .	188
9.31.2.6	update_fastest . . . . .	188
9.32	mln::data::impl::generic Namespace Reference . . . . .	188
9.32.1	Detailed Description . . . . .	189
9.32.2	Function Documentation . . . . .	189
9.32.2.1	fill_with_image . . . . .	189
9.32.2.2	fill_with_value . . . . .	189
9.32.2.3	paste . . . . .	190
9.32.2.4	transform . . . . .	190
9.32.2.5	transform . . . . .	190
9.32.2.6	transform_inplace . . . . .	191
9.32.2.7	transform_inplace . . . . .	191
9.32.2.8	update . . . . .	191
9.33	mln::data::naive Namespace Reference . . . . .	191
9.33.1	Detailed Description . . . . .	192
9.33.2	Function Documentation . . . . .	192
9.33.2.1	median . . . . .	192
9.34	mln::data::naive::impl Namespace Reference . . . . .	192
9.34.1	Detailed Description . . . . .	192
9.35	mln::debug Namespace Reference . . . . .	192
9.35.1	Detailed Description . . . . .	194
9.35.2	Function Documentation . . . . .	194
9.35.2.1	draw_graph . . . . .	194
9.35.2.2	draw_graph . . . . .	195
9.35.2.3	draw_graph . . . . .	195
9.35.2.4	filename . . . . .	195
9.35.2.5	format . . . . .	195
9.35.2.6	format . . . . .	195
9.35.2.7	format . . . . .	195
9.35.2.8	format . . . . .	196
9.35.2.9	iota . . . . .	196
9.35.2.10	mosaic . . . . .	196
9.35.2.11	println . . . . .	196
9.35.2.12	println . . . . .	196
9.35.2.13	println_with_border . . . . .	196
9.35.2.14	put_word . . . . .	197

9.35.2.15	slices_2d . . . . .	197
9.35.2.16	slices_2d . . . . .	197
9.35.2.17	superpose . . . . .	197
9.35.2.18	superpose . . . . .	197
9.35.2.19	z_order . . . . .	198
9.36	mln::debug::impl Namespace Reference . . . . .	198
9.36.1	Detailed Description . . . . .	198
9.37	mln::def Namespace Reference . . . . .	198
9.37.1	Detailed Description . . . . .	198
9.37.2	Typedef Documentation . . . . .	199
9.37.2.1	coord . . . . .	199
9.37.2.2	coordf . . . . .	199
9.37.3	Enumeration Type Documentation . . . . .	199
9.37.3.1	"@21 . . . . .	199
9.38	mln::display Namespace Reference . . . . .	199
9.38.1	Detailed Description . . . . .	199
9.39	mln::display::impl Namespace Reference . . . . .	199
9.39.1	Detailed Description . . . . .	199
9.40	mln::display::impl::generic Namespace Reference . . . . .	200
9.40.1	Detailed Description . . . . .	200
9.41	mln::doc Namespace Reference . . . . .	200
9.41.1	Detailed Description . . . . .	201
9.42	mln::draw Namespace Reference . . . . .	201
9.42.1	Detailed Description . . . . .	201
9.42.2	Function Documentation . . . . .	202
9.42.2.1	box . . . . .	202
9.42.2.2	box_plain . . . . .	202
9.42.2.3	dashed_line . . . . .	202
9.42.2.4	line . . . . .	203
9.42.2.5	plot . . . . .	203
9.43	mln::estim Namespace Reference . . . . .	203
9.43.1	Detailed Description . . . . .	204
9.43.2	Function Documentation . . . . .	204
9.43.2.1	mean . . . . .	204
9.43.2.2	mean . . . . .	204
9.43.2.3	min_max . . . . .	205



---

9.43.2.4	sum	205
9.43.2.5	sum	205
9.44	mln::extension Namespace Reference	205
9.44.1	Detailed Description	206
9.44.2	Function Documentation	206
9.44.2.1	adjust	206
9.44.2.2	adjust	207
9.44.2.3	adjust	207
9.44.2.4	adjust	207
9.44.2.5	adjust_duplicate	207
9.44.2.6	adjust_fill	207
9.44.2.7	duplicate	207
9.44.2.8	fill	207
9.45	mln::fun Namespace Reference	208
9.45.1	Detailed Description	209
9.46	mln::fun::access Namespace Reference	209
9.46.1	Detailed Description	209
9.47	mln::fun::i2v Namespace Reference	209
9.47.1	Detailed Description	210
9.47.2	Function Documentation	210
9.47.2.1	operator<<	210
9.48	mln::fun::n2v Namespace Reference	210
9.48.1	Detailed Description	210
9.49	mln::fun::p2b Namespace Reference	210
9.49.1	Detailed Description	210
9.50	mln::fun::p2p Namespace Reference	210
9.50.1	Detailed Description	211
9.51	mln::fun::p2v Namespace Reference	211
9.51.1	Detailed Description	211
9.52	mln::fun::stat Namespace Reference	211
9.52.1	Detailed Description	211
9.53	mln::fun::v2b Namespace Reference	211
9.53.1	Detailed Description	211
9.54	mln::fun::v2i Namespace Reference	211
9.54.1	Detailed Description	212
9.55	mln::fun::v2v Namespace Reference	212

9.55.1	Detailed Description	213
9.55.2	Variable Documentation	213
9.55.2.1	f_hsi_to_rgb_3x8	213
9.55.2.2	f_hsl_to_rgb_3x8	213
9.55.2.3	f_rgb_to_hsi_f	213
9.55.2.4	f_rgb_to_hsl_f	213
9.56	mln::fun::v2w2v Namespace Reference	213
9.56.1	Detailed Description	213
9.57	mln::fun::v2w_w2v Namespace Reference	213
9.57.1	Detailed Description	214
9.58	mln::fun::vv2b Namespace Reference	214
9.58.1	Detailed Description	214
9.59	mln::fun::vv2v Namespace Reference	214
9.59.1	Detailed Description	215
9.60	mln::fun::x2p Namespace Reference	215
9.60.1	Detailed Description	215
9.61	mln::fun::x2v Namespace Reference	216
9.61.1	Detailed Description	216
9.62	mln::fun::x2x Namespace Reference	216
9.62.1	Detailed Description	216
9.63	mln::geom Namespace Reference	216
9.63.1	Detailed Description	221
9.63.2	Function Documentation	221
9.63.2.1	bbox	221
9.63.2.2	bbox	221
9.63.2.3	bbox	221
9.63.2.4	bbox	221
9.63.2.5	chamfer	221
9.63.2.6	delta	221
9.63.2.7	delta	222
9.63.2.8	delta	222
9.63.2.9	horizontal_symmetry	222
9.63.2.10	max_col	222
9.63.2.11	max_col	222
9.63.2.12	max_ind	222
9.63.2.13	max_row	222

9.63.2.14	<code>max_row</code>	222
9.63.2.15	<code>max_sli</code>	223
9.63.2.16	<code>mesh_corner_point_area</code>	223
9.63.2.17	<code>mesh_curvature</code>	223
9.63.2.18	<code>mesh_normal</code>	223
9.63.2.19	<code>min_col</code>	224
9.63.2.20	<code>min_col</code>	224
9.63.2.21	<code>min_ind</code>	224
9.63.2.22	<code>min_row</code>	224
9.63.2.23	<code>min_row</code>	224
9.63.2.24	<code>min_sli</code>	224
9.63.2.25	<code>ncols</code>	225
9.63.2.26	<code>ncols</code>	225
9.63.2.27	<code>ninds</code>	225
9.63.2.28	<code>nrows</code>	225
9.63.2.29	<code>nrows</code>	225
9.63.2.30	<code>nsites</code>	225
9.63.2.31	<code>nslices</code>	225
9.63.2.32	<code>pmin_pmax</code>	226
9.63.2.33	<code>pmin_pmax</code>	226
9.63.2.34	<code>pmin_pmax</code>	226
9.63.2.35	<code>pmin_pmax</code>	226
9.63.2.36	<code>rotate</code>	226
9.63.2.37	<code>rotate</code>	226
9.63.2.38	<code>rotate</code>	226
9.63.2.39	<code>rotate</code>	227
9.63.2.40	<code>rotate</code>	227
9.63.2.41	<code>seeds2tiling</code>	227
9.63.2.42	<code>seeds2tiling_roundness</code>	228
9.63.2.43	<code>translate</code>	228
9.63.2.44	<code>translate</code>	228
9.63.2.45	<code>translate</code>	229
9.63.2.46	<code>vertical_symmetry</code>	229
9.64	<code>mln::geom::impl</code> Namespace Reference	229
9.64.1	Detailed Description	229
9.64.2	Function Documentation	229

9.64.2.1	seeds2tiling	229
9.65	mln::graph Namespace Reference	230
9.65.1	Detailed Description	230
9.65.2	Function Documentation	230
9.65.2.1	compute	230
9.65.2.2	labeling	231
9.65.2.3	to_neighb	231
9.65.2.4	to_win	231
9.66	mln::grid Namespace Reference	232
9.66.1	Detailed Description	232
9.67	mln::histo Namespace Reference	232
9.67.1	Detailed Description	233
9.67.2	Function Documentation	233
9.67.2.1	compute	233
9.67.2.2	equalize	233
9.68	mln::histo::impl Namespace Reference	233
9.68.1	Detailed Description	233
9.69	mln::histo::impl::generic Namespace Reference	233
9.69.1	Detailed Description	233
9.70	mln::impl Namespace Reference	234
9.70.1	Detailed Description	234
9.71	mln::io Namespace Reference	234
9.71.1	Detailed Description	235
9.72	mln::io::cloud Namespace Reference	235
9.72.1	Detailed Description	236
9.72.2	Function Documentation	236
9.72.2.1	load	236
9.72.2.2	save	236
9.73	mln::io::dicom Namespace Reference	236
9.73.1	Detailed Description	236
9.73.2	Function Documentation	237
9.73.2.1	get_header	237
9.73.2.2	load	237
9.74	mln::io::dump Namespace Reference	237
9.74.1	Detailed Description	238
9.74.2	Function Documentation	238

---

9.74.2.1	get_header	238
9.74.2.2	load	238
9.74.2.3	save	238
9.75	mln::io::fits Namespace Reference	238
9.75.1	Detailed Description	238
9.75.2	Function Documentation	239
9.75.2.1	load	239
9.75.2.2	load	239
9.76	mln::io::fld Namespace Reference	239
9.76.1	Detailed Description	239
9.76.2	Function Documentation	240
9.76.2.1	load	240
9.76.2.2	read_header	240
9.76.2.3	write_header	240
9.77	mln::io::magick Namespace Reference	240
9.77.1	Detailed Description	241
9.77.2	Function Documentation	241
9.77.2.1	load	241
9.77.2.2	save	241
9.78	mln::io::off Namespace Reference	241
9.78.1	Detailed Description	241
9.78.2	Function Documentation	242
9.78.2.1	load	242
9.78.2.2	save	242
9.78.2.3	save_bin_alt	242
9.79	mln::io::pbm Namespace Reference	243
9.79.1	Detailed Description	243
9.79.2	Function Documentation	243
9.79.2.1	load	243
9.79.2.2	load	243
9.79.2.3	save	244
9.80	mln::io::pbm::impl Namespace Reference	244
9.80.1	Detailed Description	244
9.81	mln::io::pbms Namespace Reference	244
9.81.1	Detailed Description	244
9.81.2	Function Documentation	244

---

9.81.2.1	load	244
9.82	mln::io::pbms::impl Namespace Reference	245
9.82.1	Detailed Description	245
9.83	mln::io::pfm Namespace Reference	245
9.83.1	Detailed Description	245
9.83.2	Function Documentation	245
9.83.2.1	load	245
9.83.2.2	load	246
9.83.2.3	save	246
9.84	mln::io::pfm::impl Namespace Reference	246
9.84.1	Detailed Description	246
9.85	mln::io::pgm Namespace Reference	246
9.85.1	Detailed Description	247
9.85.2	Function Documentation	247
9.85.2.1	load	247
9.85.2.2	load	247
9.85.2.3	save	247
9.86	mln::io::pgms Namespace Reference	247
9.86.1	Detailed Description	248
9.86.2	Function Documentation	248
9.86.2.1	load	248
9.87	mln::io::plot Namespace Reference	248
9.87.1	Detailed Description	248
9.87.2	Function Documentation	249
9.87.2.1	load	249
9.87.2.2	save	249
9.87.2.3	save	249
9.87.2.4	save	249
9.88	mln::io::pnm Namespace Reference	250
9.88.1	Detailed Description	250
9.88.2	Function Documentation	250
9.88.2.1	load	250
9.88.2.2	load	251
9.88.2.3	load_ascii_builtin	251
9.88.2.4	load_ascii_value	251
9.88.2.5	load_raw_2d	251

---

9.88.2.6	max_component	251
9.88.2.7	save	251
9.89	mln::io::pnm::impl Namespace Reference	251
9.89.1	Detailed Description	252
9.90	mln::io::pnms Namespace Reference	252
9.90.1	Detailed Description	252
9.90.2	Function Documentation	252
9.90.2.1	load	252
9.90.2.2	load	252
9.91	mln::io::ppm Namespace Reference	253
9.91.1	Detailed Description	253
9.91.2	Function Documentation	253
9.91.2.1	load	253
9.91.2.2	load	253
9.91.2.3	save	254
9.92	mln::io::ppms Namespace Reference	254
9.92.1	Detailed Description	254
9.92.2	Function Documentation	254
9.92.2.1	load	254
9.93	mln::io::raw Namespace Reference	254
9.93.1	Detailed Description	255
9.93.2	Function Documentation	255
9.93.2.1	get_header	255
9.93.2.2	load	255
9.93.2.3	save	255
9.94	mln::io::tiff Namespace Reference	256
9.94.1	Detailed Description	256
9.94.2	Function Documentation	256
9.94.2.1	load	256
9.95	mln::io::txt Namespace Reference	256
9.95.1	Detailed Description	256
9.95.2	Function Documentation	256
9.95.2.1	save	256
9.96	mln::labeling Namespace Reference	257
9.96.1	Detailed Description	260
9.96.2	Function Documentation	260

9.96.2.1	background	260
9.96.2.2	blobs	260
9.96.2.3	blobs_and_compute	261
9.96.2.4	colorize	261
9.96.2.5	colorize	262
9.96.2.6	colorize	262
9.96.2.7	compute	262
9.96.2.8	compute	262
9.96.2.9	compute	263
9.96.2.10	compute	263
9.96.2.11	compute	264
9.96.2.12	compute_image	264
9.96.2.13	compute_image	264
9.96.2.14	compute_image	265
9.96.2.15	fill_holes	265
9.96.2.16	flat_zones	266
9.96.2.17	foreground	266
9.96.2.18	pack	267
9.96.2.19	pack	267
9.96.2.20	pack_inplace	267
9.96.2.21	pack_inplace	267
9.96.2.22	regional_maxima	268
9.96.2.23	regional_minima	268
9.96.2.24	relabel	268
9.96.2.25	relabel	269
9.96.2.26	relabel_inplace	269
9.96.2.27	relabel_inplace	269
9.96.2.28	superpose	270
9.96.2.29	value	270
9.96.2.30	value_and_compute	270
9.96.2.31	wrap	271
9.96.2.32	wrap	271
9.97	mln::labeling::impl Namespace Reference	271
9.97.1	Detailed Description	272
9.97.2	Function Documentation	272
9.97.2.1	compute_fastest	272



9.97.2.2	compute_fastest	272
9.98	mln::labeling::impl::generic Namespace Reference	273
9.98.1	Detailed Description	273
9.98.2	Function Documentation	273
9.98.2.1	compute	273
9.98.2.2	compute	274
9.98.2.3	compute	274
9.98.2.4	compute	275
9.99	mln::linear Namespace Reference	275
9.99.1	Detailed Description	276
9.99.2	Function Documentation	276
9.99.2.1	gaussian	276
9.99.2.2	gaussian	276
9.99.2.3	gaussian_1st_derivative	277
9.99.2.4	gaussian_1st_derivative	277
9.99.2.5	gaussian_2nd_derivative	277
9.99.2.6	gaussian_2nd_derivative	277
9.99.2.7	mln_ch_convolve	278
9.99.2.8	mln_ch_convolve	278
9.99.2.9	mln_ch_convolve	278
9.99.2.10	mln_ch_convolve	278
9.99.2.11	mln_ch_convolve_grad	279
9.100	mln::linear::impl Namespace Reference	279
9.100.1	Detailed Description	279
9.101	mln::linear::local Namespace Reference	279
9.101.1	Detailed Description	279
9.101.2	Function Documentation	279
9.101.2.1	convolve	279
9.101.2.2	convolve	280
9.102	mln::linear::local::impl Namespace Reference	280
9.102.1	Detailed Description	280
9.103	mln::literal Namespace Reference	280
9.103.1	Detailed Description	283
9.103.2	Variable Documentation	283
9.103.2.1	black	283
9.103.2.2	blue	284

9.103.2.3	brown	284
9.103.2.4	cyan	284
9.103.2.5	dark_gray	284
9.103.2.6	green	284
9.103.2.7	identity	284
9.103.2.8	light_gray	284
9.103.2.9	lime	284
9.103.2.10	magenta	284
9.103.2.11	lmax	284
9.103.2.12	medium_gray	284
9.103.2.13	min	285
9.103.2.14	olive	285
9.103.2.15	one	285
9.103.2.16	orange	285
9.103.2.17	origin	285
9.103.2.18	pink	285
9.103.2.19	purple	285
9.103.2.20	red	285
9.103.2.21	teal	285
9.103.2.22	violet	285
9.103.2.23	white	286
9.103.2.24	yellow	286
9.103.2.25	zero	286
9.104	mln::logical Namespace Reference	286
9.104.1	Detailed Description	287
9.104.2	Function Documentation	287
9.104.2.1	and_inplace	287
9.104.2.2	and_not	287
9.104.2.3	and_not_inplace	288
9.104.2.4	not_inplace	288
9.104.2.5	or_inplace	288
9.104.2.6	xor_inplace	289
9.105	mln::logical::impl Namespace Reference	289
9.105.1	Detailed Description	289
9.106	mln::logical::impl::generic Namespace Reference	289
9.106.1	Detailed Description	289

---

9.107mln::make Namespace Reference . . . . .	290
9.107.1 Detailed Description . . . . .	295
9.107.2 Function Documentation . . . . .	295
9.107.2.1 attachment . . . . .	295
9.107.2.2 box1d . . . . .	295
9.107.2.3 box1d . . . . .	295
9.107.2.4 box2d . . . . .	296
9.107.2.5 box2d . . . . .	296
9.107.2.6 box2d_h . . . . .	297
9.107.2.7 box2d_h . . . . .	297
9.107.2.8 box3d . . . . .	297
9.107.2.9 box3d . . . . .	298
9.107.2.10cell . . . . .	298
9.107.2.11couple . . . . .	299
9.107.2.12detachment . . . . .	299
9.107.2.13dpoint2d_h . . . . .	299
9.107.2.14dummy_p_edges . . . . .	299
9.107.2.15dummy_p_edges . . . . .	300
9.107.2.16dummy_p_vertices . . . . .	300
9.107.2.17dummy_p_vertices . . . . .	300
9.107.2.18edge_image . . . . .	301
9.107.2.19edge_image . . . . .	301
9.107.2.20edge_image . . . . .	301
9.107.2.21edge_image . . . . .	301
9.107.2.22edge_image . . . . .	302
9.107.2.23edge_image . . . . .	302
9.107.2.24h_mat . . . . .	302
9.107.2.25image . . . . .	303
9.107.2.26image . . . . .	303
9.107.2.27image . . . . .	303
9.107.2.28image2d . . . . .	303
9.107.2.29image3d . . . . .	304
9.107.2.30image3d . . . . .	304
9.107.2.31influence_zone_adjacency_graph . . . . .	304
9.107.2.32mat . . . . .	304
9.107.2.33ord_pair . . . . .	304

9.107.2.34	<code>p_edges_with_mass_centers</code>	305
9.107.2.35	<code>p_vertices_with_mass_centers</code>	305
9.107.2.36	<code>pix</code>	305
9.107.2.37	<code>pixel</code>	306
9.107.2.38	<code>pixel</code>	306
9.107.2.39	<code>point2d_h</code>	306
9.107.2.40	<code>rag_and_labeled_wsl</code>	306
9.107.2.41	<code>region_adjacency_graph</code>	307
9.107.2.42	<code>relabelfun</code>	307
9.107.2.43	<code>relabelfun</code>	308
9.107.2.44	<code>vec</code>	308
9.107.2.45	<code>vec</code>	308
9.107.2.46	<code>vec</code>	309
9.107.2.47	<code>vec</code>	309
9.107.2.48	<code>vertex_image</code>	309
9.107.2.49	<code>vertex_image</code>	309
9.107.2.50	<code>voronoi</code>	310
9.107.2.51	<code>w_window</code>	310
9.107.2.52	<code>w_window1d</code>	310
9.107.2.53	<code>w_window1d_int</code>	311
9.107.2.54	<code>w_window2d</code>	311
9.107.2.55	<code>w_window2d_int</code>	312
9.107.2.56	<code>w_window3d</code>	312
9.107.2.57	<code>w_window3d_int</code>	312
9.107.2.58	<code>w_window_directional</code>	313
9.108	<code>mln::math</code> Namespace Reference	313
9.108.1	Detailed Description	313
9.108.2	Function Documentation	313
9.108.2.1	<code>abs</code>	313
9.108.2.2	<code>abs</code>	314
9.108.2.3	<code>abs</code>	314
9.109	<code>mln::metal</code> Namespace Reference	314
9.109.1	Detailed Description	315
9.110	<code>mln::metal::impl</code> Namespace Reference	315
9.110.1	Detailed Description	315
9.111	<code>mln::metal::math</code> Namespace Reference	315

9.111.1 Detailed Description . . . . .	315
9.112 <code>mln::metal::math::impl</code> Namespace Reference . . . . .	315
9.112.1 Detailed Description . . . . .	315
9.113 <code>mln::morpho</code> Namespace Reference . . . . .	315
9.113.1 Detailed Description . . . . .	318
9.113.2 Function Documentation . . . . .	319
9.113.2.1 <code>complementation</code> . . . . .	319
9.113.2.2 <code>complementation_inplace</code> . . . . .	319
9.113.2.3 <code>contrast</code> . . . . .	319
9.113.2.4 <code>dilation</code> . . . . .	319
9.113.2.5 <code>erosion</code> . . . . .	319
9.113.2.6 <code>general</code> . . . . .	320
9.113.2.7 <code>gradient</code> . . . . .	320
9.113.2.8 <code>gradient_external</code> . . . . .	320
9.113.2.9 <code>gradient_internal</code> . . . . .	320
9.113.2.10 <code>hit_or_miss</code> . . . . .	320
9.113.2.11 <code>hit_or_miss_background_closing</code> . . . . .	320
9.113.2.12 <code>hit_or_miss_background_opening</code> . . . . .	321
9.113.2.13 <code>hit_or_miss_closing</code> . . . . .	321
9.113.2.14 <code>hit_or_miss_opening</code> . . . . .	321
9.113.2.15 <code>laplacian</code> . . . . .	321
9.113.2.16 <code>line_gradient</code> . . . . .	321
9.113.2.17 <code>meyer_wst</code> . . . . .	322
9.113.2.18 <code>meyer_wst</code> . . . . .	322
9.113.2.19 <code>min</code> . . . . .	322
9.113.2.20 <code>min_inplace</code> . . . . .	322
9.113.2.21 <code>minus</code> . . . . .	323
9.113.2.22 <code>plus</code> . . . . .	323
9.113.2.23 <code>rank_filter</code> . . . . .	323
9.113.2.24 <code>thick_miss</code> . . . . .	323
9.113.2.25 <code>thickening</code> . . . . .	323
9.113.2.26 <code>thin_fit</code> . . . . .	323
9.113.2.27 <code>thinning</code> . . . . .	324
9.113.2.28 <code>top_hat_black</code> . . . . .	324
9.113.2.29 <code>top_hat_self_complementary</code> . . . . .	324
9.113.2.30 <code>top_hat_white</code> . . . . .	324

9.114mln::morpho::approx Namespace Reference . . . . .	324
9.114.1 Detailed Description . . . . .	325
9.115mln::morpho::attribute Namespace Reference . . . . .	325
9.115.1 Detailed Description . . . . .	325
9.116mln::morpho::closing::approx Namespace Reference . . . . .	325
9.116.1 Detailed Description . . . . .	325
9.116.2 Function Documentation . . . . .	326
9.116.2.1 structural . . . . .	326
9.117mln::morpho::elementary Namespace Reference . . . . .	326
9.117.1 Detailed Description . . . . .	327
9.117.2 Function Documentation . . . . .	327
9.117.2.1 closing . . . . .	327
9.117.2.2 mln_trait_op_minus_twice . . . . .	327
9.117.2.3 opening . . . . .	327
9.117.2.4 top_hat_black . . . . .	327
9.117.2.5 top_hat_self_complementary . . . . .	327
9.117.2.6 top_hat_white . . . . .	328
9.118mln::morpho::impl Namespace Reference . . . . .	328
9.118.1 Detailed Description . . . . .	328
9.119mln::morpho::impl::generic Namespace Reference . . . . .	328
9.119.1 Detailed Description . . . . .	328
9.120mln::morpho::opening::approx Namespace Reference . . . . .	328
9.120.1 Detailed Description . . . . .	329
9.120.2 Function Documentation . . . . .	329
9.120.2.1 structural . . . . .	329
9.121mln::morpho::reconstruction Namespace Reference . . . . .	329
9.121.1 Detailed Description . . . . .	329
9.122mln::morpho::reconstruction::by_dilation Namespace Reference . . . . .	329
9.122.1 Detailed Description . . . . .	329
9.123mln::morpho::reconstruction::by_erosion Namespace Reference . . . . .	329
9.123.1 Detailed Description . . . . .	330
9.124mln::morpho::tree Namespace Reference . . . . .	330
9.124.1 Detailed Description . . . . .	331
9.124.2 Function Documentation . . . . .	331
9.124.2.1 compute_attribute_image . . . . .	331
9.124.2.2 compute_attribute_image_from . . . . .	332

---

9.124.2.3	compute_parent	332
9.124.2.4	dual_input_max_tree	333
9.124.2.5	max_tree	334
9.124.2.6	min_tree	334
9.124.2.7	propagate_if	334
9.124.2.8	propagate_if	335
9.124.2.9	propagate_if_value	335
9.124.2.10	propagate_if_value	335
9.124.2.11	propagate_node_to_ancestors	335
9.124.2.12	propagate_node_to_ancestors	336
9.124.2.13	propagate_node_to_descendants	336
9.124.2.14	propagate_node_to_descendants	336
9.124.2.15	propagate_representative	337
9.125	mln::morpho::tree::filter Namespace Reference	337
9.125.1	Detailed Description	337
9.125.2	Function Documentation	337
9.125.2.1	direct	337
9.125.2.2	filter	338
9.125.2.3	max	338
9.125.2.4	min	338
9.125.2.5	subtractive	339
9.126	mln::morpho::watershed Namespace Reference	339
9.126.1	Detailed Description	340
9.126.2	Function Documentation	340
9.126.2.1	flooding	340
9.126.2.2	flooding	340
9.126.2.3	superpose	341
9.126.2.4	superpose	341
9.126.2.5	topological	341
9.127	mln::morpho::watershed::watershed Namespace Reference	341
9.127.1	Detailed Description	341
9.128	mln::morpho::watershed::watershed::generic Namespace Reference	341
9.128.1	Detailed Description	341
9.129	mln::norm Namespace Reference	342
9.129.1	Detailed Description	343
9.129.2	Function Documentation	343

---

9.129.2.1	ll	343
9.129.2.2	ll_distance	343
9.129.2.3	l2	343
9.129.2.4	l2_distance	343
9.129.2.5	linfty	343
9.129.2.6	linfty_distance	343
9.129.2.7	sqr_l2	343
9.130	mln::norm::impl Namespace Reference	344
9.130.1	Detailed Description	344
9.131	mln::opt Namespace Reference	344
9.131.1	Detailed Description	344
9.131.2	Function Documentation	345
9.131.2.1	at	345
9.131.2.2	at	345
9.131.2.3	at	345
9.131.2.4	at	345
9.131.2.5	at	345
9.131.2.6	at	345
9.132	mln::opt::impl Namespace Reference	345
9.132.1	Detailed Description	345
9.133	mln::pw Namespace Reference	345
9.133.1	Detailed Description	346
9.134	mln::registration Namespace Reference	346
9.134.1	Detailed Description	347
9.134.2	Function Documentation	347
9.134.2.1	get_rot	347
9.134.2.2	icp	347
9.134.2.3	icp	348
9.134.2.4	registration1	348
9.134.2.5	registration2	348
9.134.2.6	registration3	348
9.135	mln::select Namespace Reference	348
9.135.1	Detailed Description	349
9.136	mln::set Namespace Reference	349
9.136.1	Detailed Description	349
9.136.2	Function Documentation	350



9.136.2.1	card	350
9.136.2.2	compute	350
9.136.2.3	compute_with_weights	350
9.136.2.4	get	350
9.136.2.5	has	350
9.136.2.6	mln_meta_accu_result	350
9.136.2.7	mln_meta_accu_result	351
9.137	mln::subsampling Namespace Reference	351
9.137.1	Detailed Description	351
9.137.2	Function Documentation	352
9.137.2.1	antialiased	352
9.137.2.2	antialiased	352
9.137.2.3	gaussian_subsampling	352
9.137.2.4	subsampling	352
9.138	mln::tag Namespace Reference	352
9.138.1	Detailed Description	352
9.139	mln::test Namespace Reference	353
9.139.1	Detailed Description	353
9.139.2	Function Documentation	353
9.139.2.1	positive	353
9.139.2.2	predicate	354
9.139.2.3	predicate	354
9.139.2.4	predicate	354
9.140	mln::test::impl Namespace Reference	354
9.140.1	Detailed Description	354
9.141	mln::topo Namespace Reference	354
9.141.1	Detailed Description	359
9.141.2	Function Documentation	359
9.141.2.1	detach	359
9.141.2.2	edge	359
9.141.2.3	is_facet	360
9.141.2.4	make_algebraic_face	360
9.141.2.5	operator!=	360
9.141.2.6	operator!=	360
9.141.2.7	operator!=	360
9.141.2.8	operator!=	360

9.141.2.9 operator+	361
9.141.2.10 operator-	361
9.141.2.11 operator-	361
9.141.2.12 operator-	361
9.141.2.13 operator<	361
9.141.2.14 operator<	361
9.141.2.15 operator<	361
9.141.2.16 operator<	362
9.141.2.17 operator<<	362
9.141.2.18 operator<<	362
9.141.2.19 operator<<	362
9.141.2.20 operator<<	362
9.141.2.21 operator<<	362
9.141.2.22 operator==	363
9.141.2.23 operator==	363
9.141.2.24 operator==	363
9.141.2.25 operator==	363
9.141.2.26 operator==	363
9.142 mln::trace Namespace Reference	364
9.142.1 Detailed Description	364
9.143 mln::trait Namespace Reference	364
9.143.1 Detailed Description	364
9.144 mln::transform Namespace Reference	364
9.144.1 Detailed Description	365
9.144.2 Function Documentation	365
9.144.2.1 distance_and_closest_point_geodesic	365
9.144.2.2 distance_and_closest_point_geodesic	366
9.144.2.3 distance_and_influence_zone_geodesic	366
9.144.2.4 distance_front	367
9.144.2.5 distance_geodesic	367
9.144.2.6 hough	367
9.144.2.7 influence_zone_front	367
9.144.2.8 influence_zone_front	368
9.144.2.9 influence_zone_geodesic	368
9.144.2.10 influence_zone_geodesic_saturated	368
9.144.2.11 influence_zone_geodesic_saturated	368

9.145mln::util Namespace Reference . . . . .	369
9.145.1 Detailed Description . . . . .	372
9.145.2 Typedef Documentation . . . . .	372
9.145.2.1 vertex_id_t . . . . .	372
9.145.3 Function Documentation . . . . .	372
9.145.3.1 display_branch . . . . .	372
9.145.3.2 display_tree . . . . .	373
9.145.3.3 lemmings . . . . .	373
9.145.3.4 make_greater_point . . . . .	373
9.145.3.5 make_greater_psite . . . . .	373
9.145.3.6 operator< . . . . .	373
9.145.3.7 operator<< . . . . .	373
9.145.3.8 operator<< . . . . .	374
9.145.3.9 operator== . . . . .	374
9.145.3.10operator== . . . . .	374
9.145.3.11ord_strict . . . . .	374
9.145.3.12ord_weak . . . . .	374
9.145.3.13tree_fast_to_image . . . . .	374
9.145.3.14tree_to_fast . . . . .	375
9.145.3.15tree_to_image . . . . .	375
9.146mln::util::impl Namespace Reference . . . . .	375
9.146.1 Detailed Description . . . . .	375
9.147mln::value Namespace Reference . . . . .	375
9.147.1 Detailed Description . . . . .	380
9.147.2 Typedef Documentation . . . . .	380
9.147.2.1 float01_16 . . . . .	380
9.147.2.2 float01_8 . . . . .	380
9.147.2.3 gl16 . . . . .	380
9.147.2.4 gl8 . . . . .	380
9.147.2.5 glf . . . . .	380
9.147.2.6 int_s16 . . . . .	380
9.147.2.7 int_s32 . . . . .	381
9.147.2.8 int_s8 . . . . .	381
9.147.2.9 int_u12 . . . . .	381
9.147.2.10int_u16 . . . . .	381
9.147.2.11int_u32 . . . . .	381

9.147.2.12	int_u8	381
9.147.2.13	label_16	381
9.147.2.14	label_32	381
9.147.2.15	label_8	381
9.147.2.16	rgb16	381
9.147.2.17	rgb8	381
9.147.3	Function Documentation	382
9.147.3.1	cast	382
9.147.3.2	equiv	382
9.147.3.3	operator*	382
9.147.3.4	operator*	382
9.147.3.5	operator+	382
9.147.3.6	operator+	382
9.147.3.7	operator-	382
9.147.3.8	operator-	382
9.147.3.9	operator/	383
9.147.3.10	operator/	383
9.147.3.11	operator<<	383
9.147.3.12	operator<<	383
9.147.3.13	operator<<	383
9.147.3.14	operator<<	383
9.147.3.15	operator<<	384
9.147.3.16	operator<<	384
9.147.3.17	operator<<	384
9.147.3.18	operator<<	384
9.147.3.19	operator<<	385
9.147.3.20	operator<<	385
9.147.3.21	operator<<	385
9.147.3.22	operator==	385
9.147.3.23	operator==	385
9.147.3.24	other	385
9.147.3.25	stack	386
9.148	mln::value::impl Namespace Reference	386
9.148.1	Detailed Description	386
9.149	mln::win Namespace Reference	386
9.149.1	Detailed Description	388

9.149.2 Function Documentation	388
9.149.2.1 diff	388
9.149.2.2 mln_regular	388
9.149.2.3 mln_regular	388
9.149.2.4 sym	388
9.149.2.5 sym	388
<b>10 Class Documentation</b>	<b>389</b>
10.1 mln::accu::center< P, V > Struct Template Reference	389
10.1.1 Detailed Description	389
10.1.2 Member Function Documentation	390
10.1.2.1 init	390
10.1.2.2 is_valid	390
10.1.2.3 nsites	390
10.1.2.4 take_as_init	390
10.1.2.5 take_n_times	390
10.1.2.6 to_result	390
10.2 mln::accu::convolve< T1, T2, R > Struct Template Reference	390
10.2.1 Detailed Description	391
10.2.2 Member Function Documentation	391
10.2.2.1 init	391
10.2.2.2 is_valid	391
10.2.2.3 take_as_init	392
10.2.2.4 take_n_times	392
10.2.2.5 to_result	392
10.3 mln::accu::count_adjacent_vertices< F, S > Struct Template Reference	392
10.3.1 Detailed Description	393
10.3.2 Member Function Documentation	393
10.3.2.1 init	393
10.3.2.2 is_valid	393
10.3.2.3 set_value	393
10.3.2.4 take_as_init	393
10.3.2.5 take_n_times	393
10.3.2.6 to_result	393
10.4 mln::accu::count_value< V > Struct Template Reference	393
10.4.1 Detailed Description	394
10.4.2 Member Function Documentation	394

10.4.2.1	init	394
10.4.2.2	is_valid	394
10.4.2.3	set_value	394
10.4.2.4	take_as_init	395
10.4.2.5	take_n_times	395
10.4.2.6	to_result	395
10.5	mln::accu::histo< V > Struct Template Reference	395
10.5.1	Detailed Description	396
10.5.2	Member Function Documentation	396
10.5.2.1	is_valid	396
10.5.2.2	take	396
10.5.2.3	take_as_init	396
10.5.2.4	take_n_times	396
10.5.2.5	vect	396
10.6	mln::accu::label_used< L > Struct Template Reference	396
10.6.1	Detailed Description	397
10.6.2	Member Function Documentation	397
10.6.2.1	init	397
10.6.2.2	is_valid	397
10.6.2.3	take	397
10.6.2.4	take_as_init	397
10.6.2.5	take_n_times	398
10.6.2.6	to_result	398
10.7	mln::accu::logic::land Struct Reference	398
10.7.1	Detailed Description	398
10.7.2	Member Function Documentation	398
10.7.2.1	init	398
10.7.2.2	is_valid	399
10.7.2.3	take_as_init	399
10.7.2.4	take_n_times	399
10.7.2.5	to_result	399
10.8	mln::accu::logic::land_basic Struct Reference	399
10.8.1	Detailed Description	400
10.8.2	Member Function Documentation	400
10.8.2.1	can_stop	400
10.8.2.2	init	400

10.8.2.3	is_valid	400
10.8.2.4	take_as_init	400
10.8.2.5	take_n_times	400
10.8.2.6	to_result	400
10.9	mln::accu::logic::lor Struct Reference	400
10.9.1	Detailed Description	401
10.9.2	Member Function Documentation	401
10.9.2.1	init	401
10.9.2.2	is_valid	401
10.9.2.3	take_as_init	401
10.9.2.4	take_n_times	401
10.9.2.5	to_result	401
10.10	mln::accu::logic::lor_basic Struct Reference	402
10.10.1	Detailed Description	402
10.10.2	Member Function Documentation	402
10.10.2.1	can_stop	402
10.10.2.2	init	402
10.10.2.3	is_valid	403
10.10.2.4	take_as_init	403
10.10.2.5	take_n_times	403
10.10.2.6	to_result	403
10.11	mln::accu::maj_h< T > Struct Template Reference	403
10.11.1	Detailed Description	404
10.11.2	Member Function Documentation	404
10.11.2.1	init	404
10.11.2.2	is_valid	404
10.11.2.3	take_as_init	404
10.11.2.4	take_n_times	404
10.11.2.5	to_result	404
10.12	mln::accu::math::count< T > Struct Template Reference	404
10.12.1	Detailed Description	405
10.12.2	Member Function Documentation	405
10.12.2.1	init	405
10.12.2.2	is_valid	405
10.12.2.3	set_value	405
10.12.2.4	take_as_init	405

10.12.2.5	take_n_times . . . . .	405
10.12.2.6	to_result . . . . .	406
10.13	mln::accu::math::inf< T > Struct Template Reference . . . . .	406
10.13.1	Detailed Description . . . . .	406
10.13.2	Member Function Documentation . . . . .	406
10.13.2.1	init . . . . .	406
10.13.2.2	is_valid . . . . .	407
10.13.2.3	take_as_init . . . . .	407
10.13.2.4	take_n_times . . . . .	407
10.13.2.5	to_result . . . . .	407
10.14	mln::accu::math::sum< T, S > Struct Template Reference . . . . .	407
10.14.1	Detailed Description . . . . .	408
10.14.2	Member Function Documentation . . . . .	408
10.14.2.1	init . . . . .	408
10.14.2.2	is_valid . . . . .	408
10.14.2.3	take_as_init . . . . .	408
10.14.2.4	take_n_times . . . . .	408
10.14.2.5	to_result . . . . .	408
10.15	mln::accu::math::sup< T > Struct Template Reference . . . . .	408
10.15.1	Detailed Description . . . . .	409
10.15.2	Member Function Documentation . . . . .	409
10.15.2.1	init . . . . .	409
10.15.2.2	is_valid . . . . .	409
10.15.2.3	take_as_init . . . . .	409
10.15.2.4	take_n_times . . . . .	409
10.15.2.5	to_result . . . . .	410
10.16	mln::accu::max_site< I > Struct Template Reference . . . . .	410
10.16.1	Detailed Description . . . . .	410
10.16.2	Member Function Documentation . . . . .	410
10.16.2.1	init . . . . .	410
10.16.2.2	is_valid . . . . .	410
10.16.2.3	take_as_init . . . . .	411
10.16.2.4	take_n_times . . . . .	411
10.16.2.5	to_result . . . . .	411
10.17	mln::accu::meta::center Struct Reference . . . . .	411
10.17.1	Detailed Description . . . . .	412



---

10.18	<code>mln::accu::meta::count_adjacent_vertices</code> Struct Reference	412
10.18.1	Detailed Description	413
10.19	<code>mln::accu::meta::count_labels</code> Struct Reference	413
10.19.1	Detailed Description	414
10.20	<code>mln::accu::meta::count_value</code> Struct Reference	414
10.20.1	Detailed Description	415
10.21	<code>mln::accu::meta::histo</code> Struct Reference	415
10.21.1	Detailed Description	416
10.22	<code>mln::accu::meta::label_used</code> Struct Reference	416
10.22.1	Detailed Description	417
10.23	<code>mln::accu::meta::logic::land</code> Struct Reference	417
10.23.1	Detailed Description	418
10.24	<code>mln::accu::meta::logic::land_basic</code> Struct Reference	418
10.24.1	Detailed Description	419
10.25	<code>mln::accu::meta::logic::lor</code> Struct Reference	419
10.25.1	Detailed Description	420
10.26	<code>mln::accu::meta::logic::lor_basic</code> Struct Reference	420
10.26.1	Detailed Description	421
10.27	<code>mln::accu::meta::maj_h</code> Struct Reference	421
10.27.1	Detailed Description	422
10.28	<code>mln::accu::meta::math::count</code> Struct Reference	422
10.28.1	Detailed Description	423
10.29	<code>mln::accu::meta::math::inf</code> Struct Reference	423
10.29.1	Detailed Description	424
10.30	<code>mln::accu::meta::math::sum</code> Struct Reference	424
10.30.1	Detailed Description	425
10.31	<code>mln::accu::meta::math::sup</code> Struct Reference	425
10.31.1	Detailed Description	426
10.32	<code>mln::accu::meta::max_site</code> Struct Reference	426
10.32.1	Detailed Description	427
10.33	<code>mln::accu::meta::nil</code> Struct Reference	427
10.33.1	Detailed Description	428
10.34	<code>mln::accu::meta::p&lt; mA &gt;</code> Struct Template Reference	428
10.34.1	Detailed Description	429
10.35	<code>mln::accu::meta::pair&lt; A1, A2 &gt;</code> Struct Template Reference	429
10.35.1	Detailed Description	430

10.36	mln::accu::meta::rms Struct Reference	430
10.36.1	Detailed Description	431
10.37	mln::accu::meta::shape::bbox Struct Reference	431
10.37.1	Detailed Description	432
10.38	mln::accu::meta::shape::height Struct Reference	432
10.38.1	Detailed Description	433
10.39	mln::accu::meta::shape::volume Struct Reference	433
10.39.1	Detailed Description	434
10.40	mln::accu::meta::stat::max Struct Reference	434
10.40.1	Detailed Description	435
10.41	mln::accu::meta::stat::max_h Struct Reference	435
10.41.1	Detailed Description	436
10.42	mln::accu::meta::stat::mean Struct Reference	436
10.42.1	Detailed Description	437
10.43	mln::accu::meta::stat::median_alt< T > Struct Template Reference	437
10.43.1	Detailed Description	438
10.44	mln::accu::meta::stat::median_h Struct Reference	438
10.44.1	Detailed Description	439
10.45	mln::accu::meta::stat::min Struct Reference	439
10.45.1	Detailed Description	440
10.46	mln::accu::meta::stat::min_h Struct Reference	440
10.46.1	Detailed Description	441
10.47	mln::accu::meta::stat::rank Struct Reference	441
10.47.1	Detailed Description	442
10.48	mln::accu::meta::stat::rank_high_quant Struct Reference	442
10.48.1	Detailed Description	443
10.49	mln::accu::meta::tuple< n, > Struct Template Reference	443
10.49.1	Detailed Description	444
10.50	mln::accu::meta::val< mA > Struct Template Reference	444
10.50.1	Detailed Description	445
10.51	mln::accu::nil< T > Struct Template Reference	445
10.51.1	Detailed Description	446
10.51.2	Member Function Documentation	446
10.51.2.1	init	446
10.51.2.2	is_valid	446
10.51.2.3	take_as_init	446

---

10.51.2.4	take_n_times	446
10.51.2.5	to_result	446
10.52	mln::accu::p< A > Struct Template Reference	447
10.52.1	Detailed Description	447
10.52.2	Member Function Documentation	447
10.52.2.1	init	447
10.52.2.2	is_valid	447
10.52.2.3	take_as_init	447
10.52.2.4	take_n_times	448
10.52.2.5	to_result	448
10.53	mln::accu::pair< A1, A2, T > Struct Template Reference	448
10.53.1	Detailed Description	449
10.53.2	Member Function Documentation	449
10.53.2.1	first	449
10.53.2.2	first_accu	449
10.53.2.3	init	449
10.53.2.4	is_valid	450
10.53.2.5	second	450
10.53.2.6	second_accu	450
10.53.2.7	take_as_init	450
10.53.2.8	take_n_times	450
10.53.2.9	to_result	450
10.54	mln::accu::rms< T, V > Struct Template Reference	450
10.54.1	Detailed Description	451
10.54.2	Member Function Documentation	451
10.54.2.1	init	451
10.54.2.2	is_valid	451
10.54.2.3	take_as_init	451
10.54.2.4	take_n_times	451
10.54.2.5	to_result	452
10.55	mln::accu::shape::bbox< P > Struct Template Reference	452
10.55.1	Detailed Description	452
10.55.2	Member Function Documentation	452
10.55.2.1	init	452
10.55.2.2	is_valid	452
10.55.2.3	take_as_init	453

10.55.2.4	take_n_times . . . . .	453
10.55.2.5	to_result . . . . .	453
10.56	mln::accu::shape::height< I > Struct Template Reference . . . . .	453
10.56.1	Detailed Description . . . . .	454
10.56.2	Member Typedef Documentation . . . . .	454
10.56.2.1	argument . . . . .	454
10.56.2.2	value . . . . .	454
10.56.3	Member Function Documentation . . . . .	454
10.56.3.1	init . . . . .	454
10.56.3.2	is_valid . . . . .	454
10.56.3.3	set_value . . . . .	454
10.56.3.4	take_as_init . . . . .	455
10.56.3.5	take_n_times . . . . .	455
10.56.3.6	to_result . . . . .	455
10.57	mln::accu::shape::volume< I > Struct Template Reference . . . . .	455
10.57.1	Detailed Description . . . . .	456
10.57.2	Member Typedef Documentation . . . . .	456
10.57.2.1	argument . . . . .	456
10.57.2.2	value . . . . .	456
10.57.3	Member Function Documentation . . . . .	456
10.57.3.1	init . . . . .	456
10.57.3.2	is_valid . . . . .	456
10.57.3.3	set_value . . . . .	456
10.57.3.4	take_as_init . . . . .	457
10.57.3.5	take_n_times . . . . .	457
10.57.3.6	to_result . . . . .	457
10.58	mln::accu::site_set::rectangularity< P > Class Template Reference . . . . .	457
10.58.1	Detailed Description . . . . .	458
10.58.2	Constructor & Destructor Documentation . . . . .	458
10.58.2.1	rectangularity . . . . .	458
10.58.3	Member Function Documentation . . . . .	458
10.58.3.1	area . . . . .	458
10.58.3.2	bbox . . . . .	458
10.58.3.3	take_as_init . . . . .	458
10.58.3.4	take_n_times . . . . .	458
10.58.3.5	to_result . . . . .	458

10.59	<code>mln::accu::stat::deviation&lt; T, S, M &gt;</code> Struct Template Reference . . . . .	459
10.59.1	Detailed Description . . . . .	459
10.59.2	Member Function Documentation . . . . .	459
10.59.2.1	<code>init</code> . . . . .	459
10.59.2.2	<code>is_valid</code> . . . . .	459
10.59.2.3	<code>take_as_init</code> . . . . .	460
10.59.2.4	<code>take_n_times</code> . . . . .	460
10.59.2.5	<code>to_result</code> . . . . .	460
10.60	<code>mln::accu::stat::max&lt; T &gt;</code> Struct Template Reference . . . . .	460
10.60.1	Detailed Description . . . . .	461
10.60.2	Member Function Documentation . . . . .	461
10.60.2.1	<code>init</code> . . . . .	461
10.60.2.2	<code>is_valid</code> . . . . .	461
10.60.2.3	<code>set_value</code> . . . . .	461
10.60.2.4	<code>take_as_init</code> . . . . .	461
10.60.2.5	<code>take_n_times</code> . . . . .	461
10.60.2.6	<code>to_result</code> . . . . .	461
10.61	<code>mln::accu::stat::max_h&lt; V &gt;</code> Struct Template Reference . . . . .	461
10.61.1	Detailed Description . . . . .	462
10.61.2	Member Function Documentation . . . . .	462
10.61.2.1	<code>init</code> . . . . .	462
10.61.2.2	<code>is_valid</code> . . . . .	462
10.61.2.3	<code>take_as_init</code> . . . . .	462
10.61.2.4	<code>take_n_times</code> . . . . .	462
10.61.2.5	<code>to_result</code> . . . . .	463
10.62	<code>mln::accu::stat::mean&lt; T, S, M &gt;</code> Struct Template Reference . . . . .	463
10.62.1	Detailed Description . . . . .	463
10.62.2	Member Function Documentation . . . . .	464
10.62.2.1	<code>count</code> . . . . .	464
10.62.2.2	<code>init</code> . . . . .	464
10.62.2.3	<code>is_valid</code> . . . . .	464
10.62.2.4	<code>sum</code> . . . . .	464
10.62.2.5	<code>take_as_init</code> . . . . .	464
10.62.2.6	<code>take_n_times</code> . . . . .	464
10.62.2.7	<code>to_result</code> . . . . .	464
10.63	<code>mln::accu::stat::median_alt&lt; S &gt;</code> Struct Template Reference . . . . .	464

10.63.1 Detailed Description	465
10.63.2 Member Function Documentation	465
10.63.2.1 is_valid	465
10.63.2.2 take	466
10.63.2.3 take_as_init	466
10.63.2.4 take_n_times	466
10.63.2.5 to_result	466
10.64mln::accu::stat::median_h< V > Struct Template Reference	466
10.64.1 Detailed Description	467
10.64.2 Member Function Documentation	467
10.64.2.1 init	467
10.64.2.2 is_valid	467
10.64.2.3 take_as_init	467
10.64.2.4 take_n_times	467
10.64.2.5 to_result	468
10.65mln::accu::stat::meta::deviation Struct Reference	468
10.65.1 Detailed Description	468
10.66mln::accu::stat::min< T > Struct Template Reference	468
10.66.1 Detailed Description	469
10.66.2 Member Function Documentation	469
10.66.2.1 init	469
10.66.2.2 is_valid	469
10.66.2.3 set_value	469
10.66.2.4 take_as_init	469
10.66.2.5 take_n_times	470
10.66.2.6 to_result	470
10.67mln::accu::stat::min_h< V > Struct Template Reference	470
10.67.1 Detailed Description	470
10.67.2 Member Function Documentation	470
10.67.2.1 init	470
10.67.2.2 is_valid	471
10.67.2.3 take_as_init	471
10.67.2.4 take_n_times	471
10.67.2.5 to_result	471
10.68mln::accu::stat::min_max< V > Struct Template Reference	471
10.68.1 Detailed Description	473

---

10.68.2 Member Function Documentation . . . . .	473
10.68.2.1 first . . . . .	473
10.68.2.2 first_accu . . . . .	473
10.68.2.3 init . . . . .	473
10.68.2.4 is_valid . . . . .	473
10.68.2.5 second . . . . .	473
10.68.2.6 second_accu . . . . .	473
10.68.2.7 take_as_init . . . . .	474
10.68.2.8 take_n_times . . . . .	474
10.68.2.9 to_result . . . . .	474
10.69 mln::accu::stat::rank< T > Struct Template Reference . . . . .	474
10.69.1 Detailed Description . . . . .	475
10.69.2 Member Function Documentation . . . . .	475
10.69.2.1 init . . . . .	475
10.69.2.2 is_valid . . . . .	475
10.69.2.3 k . . . . .	475
10.69.2.4 take_as_init . . . . .	475
10.69.2.5 take_n_times . . . . .	475
10.69.2.6 to_result . . . . .	475
10.70 mln::accu::stat::rank< bool > Struct Template Reference . . . . .	475
10.70.1 Detailed Description . . . . .	476
10.70.2 Member Function Documentation . . . . .	476
10.70.2.1 init . . . . .	476
10.70.2.2 is_valid . . . . .	476
10.70.2.3 take_as_init . . . . .	476
10.70.2.4 take_n_times . . . . .	476
10.70.2.5 to_result . . . . .	477
10.71 mln::accu::stat::rank_high_quant< T > Struct Template Reference . . . . .	477
10.71.1 Detailed Description . . . . .	477
10.71.2 Member Function Documentation . . . . .	477
10.71.2.1 init . . . . .	477
10.71.2.2 is_valid . . . . .	478
10.71.2.3 take_as_init . . . . .	478
10.71.2.4 take_n_times . . . . .	478
10.71.2.5 to_result . . . . .	478
10.72 mln::accu::stat::var< T > Struct Template Reference . . . . .	478

10.72.1 Detailed Description	479
10.72.2 Member Typedef Documentation	479
10.72.2.1 mean_t	479
10.72.3 Member Function Documentation	479
10.72.3.1 init	479
10.72.3.2 is_valid	479
10.72.3.3 mean	479
10.72.3.4 n_items	480
10.72.3.5 take_as_init	480
10.72.3.6 take_n_times	480
10.72.3.7 to_result	480
10.72.3.8 variance	480
10.73mln::accu::stat::variance< T, S, R > Struct Template Reference	480
10.73.1 Detailed Description	481
10.73.2 Member Function Documentation	481
10.73.2.1 init	481
10.73.2.2 is_valid	481
10.73.2.3 mean	482
10.73.2.4 n_items	482
10.73.2.5 standard_deviation	482
10.73.2.6 sum	482
10.73.2.7 take_as_init	482
10.73.2.8 take_n_times	482
10.73.2.9 to_result	482
10.73.2.10var	482
10.74mln::accu::tuple< A, n, > Struct Template Reference	483
10.74.1 Detailed Description	483
10.74.2 Member Function Documentation	483
10.74.2.1 init	483
10.74.2.2 is_valid	483
10.74.2.3 take_as_init	484
10.74.2.4 take_n_times	484
10.74.2.5 to_result	484
10.75mln::accu::val< A > Struct Template Reference	484
10.75.1 Detailed Description	485
10.75.2 Member Function Documentation	485



10.75.2.1	init	485
10.75.2.2	is_valid	485
10.75.2.3	take_as_init	485
10.75.2.4	take_n_times	485
10.75.2.5	to_result	485
10.76	mln::Accumulator< E > Struct Template Reference	485
10.76.1	Detailed Description	487
10.76.2	Member Function Documentation	487
10.76.2.1	take_as_init	487
10.76.2.2	take_n_times	487
10.77	mln::algebra::h_mat< d, T > Struct Template Reference	487
10.77.1	Detailed Description	488
10.77.2	Member Enumeration Documentation	488
10.77.2.1	"@7	488
10.77.3	Constructor & Destructor Documentation	488
10.77.3.1	h_mat	488
10.77.3.2	h_mat	488
10.77.4	Member Function Documentation	488
10.77.4.1	_1	488
10.77.4.2	t	489
10.78	mln::algebra::h_vec< d, C > Struct Template Reference	489
10.78.1	Detailed Description	490
10.78.2	Member Enumeration Documentation	490
10.78.2.1	"@8	490
10.78.3	Constructor & Destructor Documentation	490
10.78.3.1	h_vec	490
10.78.3.2	h_vec	490
10.78.4	Member Function Documentation	490
10.78.4.1	operator mat< n, 1, U >	490
10.78.4.2	t	490
10.78.4.3	to_vec	490
10.78.5	Member Data Documentation	490
10.78.5.1	origin	490
10.78.5.2	zero	491
10.79	mln::bkd_pixterId< I > Class Template Reference	491
10.79.1	Detailed Description	491

10.79.2 Member Typedef Documentation	491
10.79.2.1 image	491
10.79.3 Constructor & Destructor Documentation	491
10.79.3.1 bkd_pixter1d	491
10.79.4 Member Function Documentation	492
10.79.4.1 next	492
10.80mln::bkd_pixter2d< I > Class Template Reference	492
10.80.1 Detailed Description	492
10.80.2 Member Typedef Documentation	492
10.80.2.1 image	492
10.80.3 Constructor & Destructor Documentation	493
10.80.3.1 bkd_pixter2d	493
10.80.4 Member Function Documentation	493
10.80.4.1 next	493
10.81mln::bkd_pixter3d< I > Class Template Reference	493
10.81.1 Detailed Description	494
10.81.2 Member Typedef Documentation	494
10.81.2.1 image	494
10.81.3 Constructor & Destructor Documentation	494
10.81.3.1 bkd_pixter3d	494
10.81.4 Member Function Documentation	494
10.81.4.1 next	494
10.82mln::box< P > Struct Template Reference	494
10.82.1 Detailed Description	497
10.82.2 Member Typedef Documentation	497
10.82.2.1 bkd_piter	497
10.82.2.2 element	497
10.82.2.3 fwd_piter	498
10.82.2.4 piter	498
10.82.2.5 psite	498
10.82.2.6 site	498
10.82.3 Member Enumeration Documentation	498
10.82.3.1 "@31	498
10.82.4 Constructor & Destructor Documentation	498
10.82.4.1 box	498
10.82.4.2 box	498

---

10.82.4.3	box	498
10.82.5	Member Function Documentation	498
10.82.5.1	bbox	498
10.82.5.2	crop_wrt	499
10.82.5.3	enlarge	499
10.82.5.4	enlarge	499
10.82.5.5	has	499
10.82.5.6	is_empty	499
10.82.5.7	is_valid	499
10.82.5.8	len	500
10.82.5.9	memory_size	500
10.82.5.10	merge	500
10.82.5.11	nsites	500
10.82.5.12	pcenter	500
10.82.5.13	pmax	500
10.82.5.14	pmax	501
10.82.5.15	pmin	501
10.82.5.16	pmin	501
10.82.5.17	to_larger	501
10.82.6	Friends And Related Function Documentation	501
10.82.6.1	operator<<	501
10.83	mln::Box< E > Struct Template Reference	501
10.83.1	Detailed Description	503
10.83.2	Member Function Documentation	504
10.83.2.1	bbox	504
10.83.2.2	is_empty	504
10.83.2.3	len	504
10.83.2.4	nsites	504
10.83.3	Friends And Related Function Documentation	504
10.83.3.1	diff	504
10.83.3.2	inter	505
10.83.3.3	operator<	505
10.83.3.4	operator<	505
10.83.3.5	operator<<	505
10.83.3.6	operator<=	505
10.83.3.7	operator<=	506

10.83.3.8	operator==	506
10.83.3.9	sym_diff	506
10.83.3.10	uni	506
10.83.3.11	unique	506
10.84	mln::box_runend_piter< P > Class Template Reference	506
10.84.1	Detailed Description	507
10.84.2	Constructor & Destructor Documentation	507
10.84.2.1	box_runend_piter	507
10.84.3	Member Function Documentation	507
10.84.3.1	next	507
10.84.3.2	run_length	507
10.85	mln::box_runstart_piter< P > Class Template Reference	507
10.85.1	Detailed Description	508
10.85.2	Constructor & Destructor Documentation	508
10.85.2.1	box_runstart_piter	508
10.85.3	Member Function Documentation	508
10.85.3.1	next	508
10.85.3.2	run_length	508
10.86	mln::Browsing< E > Struct Template Reference	509
10.86.1	Detailed Description	509
10.87	mln::canvas::browsing::backdiagonal2d_t Struct Reference	509
10.87.1	Detailed Description	510
10.88	mln::canvas::browsing::breadth_first_search_t Struct Reference	511
10.88.1	Detailed Description	511
10.89	mln::canvas::browsing::depth_first_search_t Struct Reference	511
10.89.1	Detailed Description	511
10.90	mln::canvas::browsing::diagonal2d_t Struct Reference	511
10.90.1	Detailed Description	512
10.91	mln::canvas::browsing::dir_struct_elt_incr_update_t Struct Reference	513
10.91.1	Detailed Description	513
10.92	mln::canvas::browsing::directional_t Struct Reference	514
10.92.1	Detailed Description	515
10.93	mln::canvas::browsing::fwd_t Struct Reference	516
10.93.1	Detailed Description	517
10.94	mln::canvas::browsing::hyper_directional_t Struct Reference	517
10.94.1	Detailed Description	518

10.95	<code>mln::canvas::browsing::snake_fwd_t</code> Struct Reference	519
10.95.1	Detailed Description	519
10.96	<code>mln::canvas::browsing::snake_generic_t</code> Struct Reference	520
10.96.1	Detailed Description	521
10.97	<code>mln::canvas::browsing::snake_vert_t</code> Struct Reference	522
10.97.1	Detailed Description	522
10.98	<code>mln::canvas::chamfer&lt; F &gt;</code> Struct Template Reference	523
10.98.1	Detailed Description	523
10.99	<code>mln::category&lt; R(*) (A) &gt;</code> Struct Template Reference	523
10.99.1	Detailed Description	523
10.100	<code>mln::complex_image&lt; D, G, V &gt;</code> Class Template Reference	523
10.100.1	Detailed Description	525
10.100.2	Member Typedef Documentation	525
10.100.2.1	<code>geom</code>	525
10.100.2.2	<code>value</code>	525
10.100.2.3	<code>value</code>	525
10.100.2.4	<code>skeleton</code>	525
10.100.2.5	<code>value</code>	525
10.100.3	Constructor & Destructor Documentation	525
10.100.3.1	<code>complex_image</code>	525
10.100.4	Member Function Documentation	526
10.100.4.1	<code>domain</code>	526
10.100.4.2	<code>operator()</code>	526
10.100.4.3	<code>operator()</code>	526
10.100.4.4	<code>values</code>	526
10.100.5	Member Data Documentation	526
10.100.5.1	<code>dim</code>	526
10.101	<code>mln::complex_neighborhood_bkd_piter&lt; I, G, N &gt;</code> Class Template Reference	526
10.101.1	Detailed Description	527
10.101.2	Member Typedef Documentation	527
10.101.2.1	<code>liter_type</code>	527
10.101.2.2	<code>psite</code>	527
10.101.3	Constructor & Destructor Documentation	527
10.101.3.1	<code>complex_neighborhood_bkd_piter</code>	527
10.101.4	Member Function Documentation	528
10.101.4.1	<code>liter</code>	528

10.101.4.2next	528
10.102.1.1n::complex_neighborhood_fwd_piter< I, G, N > Class Template Reference	528
10.102.2.Detailed Description	529
10.102.3.Member Typedef Documentation	529
10.102.2.liter_type	529
10.102.2.2psite	529
10.102.4.Constructor & Destructor Documentation	529
10.102.3.lcomplex_neighborhood_fwd_piter	529
10.102.5.Member Function Documentation	529
10.102.4.liter	529
10.102.4.2next	529
10.103.1.1n::complex_psite< D, G > Class Template Reference	530
10.103.2.Detailed Description	530
10.103.3.Constructor & Destructor Documentation	531
10.103.2.lcomplex_psite	531
10.103.2.2complex_psite	531
10.103.4.Member Function Documentation	531
10.103.3.lchange_target	531
10.103.3.2face	531
10.103.3.3face_id	531
10.103.3.4invalidate	531
10.103.3.5s_valid	532
10.103.3.6n	532
10.103.3.7site_set	532
10.104.1.1n::complex_window_bkd_piter< I, G, W > Class Template Reference	532
10.104.2.Detailed Description	533
10.104.3.Member Typedef Documentation	533
10.104.2.liter_type	533
10.104.2.2psite	533
10.104.4.Constructor & Destructor Documentation	533
10.104.3.lcomplex_window_bkd_piter	533
10.104.5.Member Function Documentation	533
10.104.4.liter	533
10.104.4.2next	533
10.105.1.1n::complex_window_fwd_piter< I, G, W > Class Template Reference	534
10.105.2.Detailed Description	534

10.105.2	Member Typedef Documentation	535
10.105.2.1	liter_type	535
10.105.2.2	psite	535
10.105.3	Constructor & Destructor Documentation	535
10.105.3.1	lcomplex_window_fwd_piter	535
10.105.4	Member Function Documentation	535
10.105.4.1	liter	535
10.105.4.2	next	535
10.106	mln::decorated_image< I, D > Struct Template Reference	535
10.106.1	Detailed Description	536
10.106.2	Member Typedef Documentation	537
10.106.2.1	lvalue	537
10.106.2.2	psite	537
10.106.2.3	rvalue	537
10.106.2.4	skeleton	537
10.106.3	Constructor & Destructor Documentation	537
10.106.3.1	ldecorated_image	537
10.106.3.2	~decorated_image	537
10.106.4	Member Function Documentation	537
10.106.4.1	ldecoration	537
10.106.4.2	decoration	537
10.106.4.3	operator decorated_image< const I, D >	538
10.106.4.4	operator()	538
10.106.4.5	operator()	538
10.107	mln::Delta_Point_Site< E > Struct Template Reference	538
10.107.1	Detailed Description	538
10.108	mln::Delta_Point_Site< void > Struct Template Reference	539
10.108.1	Detailed Description	539
10.109	mln::doc::Accumulator< E > Struct Template Reference	539
10.109.1	Detailed Description	539
10.109.2	Member Typedef Documentation	540
10.109.2.1	largument	540
10.109.3	Member Function Documentation	540
10.109.3.1	linit	540
10.109.3.2	take	540
10.109.3.3	take	540

10.110	<a href="#">ln::doc::Box&lt; E &gt; Struct Template Reference</a>	540
10.110.1	<a href="#">Detailed Description</a>	541
10.110.2	<a href="#">Member Typedef Documentation</a>	541
10.110.2.1	<a href="#">bkd_piter</a>	541
10.110.2.2	<a href="#">fwd_piter</a>	542
10.110.2.3	<a href="#">psite</a>	542
10.110.2.4	<a href="#">site</a>	542
10.110.3	<a href="#">Member Function Documentation</a>	542
10.110.3.1	<a href="#">bbox</a>	542
10.110.3.2	<a href="#">has</a>	542
10.110.3.3	<a href="#">nsites</a>	542
10.110.3.4	<a href="#">pmax</a>	543
10.110.3.5	<a href="#">pmin</a>	543
10.111	<a href="#">ln::doc::Dpoint&lt; E &gt; Struct Template Reference</a>	543
10.111.1	<a href="#">Detailed Description</a>	544
10.111.2	<a href="#">Member Typedef Documentation</a>	544
10.111.2.1	<a href="#">lcoord</a>	544
10.111.2.2	<a href="#">dpoint</a>	544
10.111.2.3	<a href="#">point</a>	544
10.111.3	<a href="#">Member Enumeration Documentation</a>	544
10.111.3.1	<a href="#">l"@19</a>	544
10.111.4	<a href="#">Member Function Documentation</a>	544
10.111.4.1	<a href="#">operator[]</a>	544
10.112	<a href="#">ln::doc::Fastest_Image&lt; E &gt; Struct Template Reference</a>	545
10.112.1	<a href="#">Detailed Description</a>	547
10.112.2	<a href="#">Member Typedef Documentation</a>	547
10.112.2.1	<a href="#">bkd_piter</a>	547
10.112.2.2	<a href="#">coord</a>	547
10.112.2.3	<a href="#">dpoint</a>	548
10.112.2.4	<a href="#">fwd_piter</a>	548
10.112.2.5	<a href="#">value</a>	548
10.112.2.6	<a href="#">point</a>	548
10.112.2.7	<a href="#">pset</a>	548
10.112.2.8	<a href="#">psite</a>	548
10.112.2.9	<a href="#">value</a>	548
10.112.2.10	<a href="#">skelton</a>	549



10.112.2.1value	549
10.112.2.12set	549
10.112.3Member Function Documentation	549
10.112.3.1bbox	549
10.112.3.2border	549
10.112.3.3buffer	549
10.112.3.4delta_index	550
10.112.3.5domain	550
10.112.3.6has	550
10.112.3.7has	550
10.112.3.8s_valid	550
10.112.3.9elements	551
10.112.3.10sites	551
10.112.3.11operator()	551
10.112.3.12operator()	551
10.112.3.13operator[]	552
10.112.3.14operator[]	552
10.112.3.15point_at_index	552
10.112.3.16values	552
10.113Inl::doc::Generalized_Pixel< E > Struct Template Reference	553
10.113.1Detailed Description	553
10.113.2Member Typedef Documentation	554
10.113.2.1image	554
10.113.2.2value	554
10.113.2.3value	554
10.113.3Member Function Documentation	554
10.113.3.1ima	554
10.113.3.2val	554
10.114Inl::doc::Image< E > Struct Template Reference	554
10.114.1Detailed Description	556
10.114.2Member Typedef Documentation	557
10.114.2.1bkd_piter	557
10.114.2.2coord	557
10.114.2.3dpoint	557
10.114.2.4fwd_piter	557
10.114.2.5value	557

10.114.2.6point	557
10.114.2.7pset	557
10.114.2.8psite	558
10.114.2.9rvalue	558
10.114.2.10keleton	558
10.114.2.11alue	558
10.114.2.12set	558
10.114.3Member Function Documentation	558
10.114.3.1bbox	558
10.114.3.2domain	558
10.114.3.3has	559
10.114.3.4has	559
10.114.3.5is_valid	559
10.114.3.6nsites	559
10.114.3.7operator()	559
10.114.3.8operator()	560
10.114.3.9values	560
10.115In::doc::Iterator< E > Struct Template Reference	560
10.115.Detailed Description	561
10.115.2Member Function Documentation	561
10.115.2.1invalidate	561
10.115.2.2is_valid	561
10.115.2.3start	561
10.116In::doc::Neighborhood< E > Struct Template Reference	561
10.116.Detailed Description	562
10.116.2Member Typedef Documentation	562
10.116.2.1bkd_niter	562
10.116.2.2dpoint	563
10.116.2.3fwd_niter	563
10.116.2.4niter	563
10.116.2.5point	563
10.117In::doc::Object< E > Struct Template Reference	563
10.117.Detailed Description	564
10.118In::doc::Pixel_Iterator< E > Struct Template Reference	564
10.118.Detailed Description	565
10.118.2Member Typedef Documentation	565

10.118.2.1image	565
10.118.2.2value	565
10.118.2.3rvalue	565
10.118.2.4value	565
10.118.3Member Function Documentation	566
10.118.3.1ima	566
10.118.3.2invalidate	566
10.118.3.3is_valid	566
10.118.3.4start	566
10.118.3.5val	566
10.119mln::doc::Point_Site< E > Struct Template Reference	566
10.119.1Detailed Description	567
10.119.2Member Typedef Documentation	567
10.119.2.1coord	567
10.119.2.2dpoint	567
10.119.2.3mesh	567
10.119.2.4point	568
10.119.3Member Enumeration Documentation	568
10.119.3.1"@20	568
10.119.4Member Function Documentation	568
10.119.4.1operator[]	568
10.119.4.2to_point	568
10.120mln::doc::Site_Iterator< E > Struct Template Reference	568
10.120.1Detailed Description	569
10.120.2Member Typedef Documentation	570
10.120.2.1psite	570
10.120.3Member Function Documentation	570
10.120.3.1invalidate	570
10.120.3.2is_valid	570
10.120.3.3operator psite	570
10.120.3.4start	570
10.121mln::doc::Site_Set< E > Struct Template Reference	570
10.121.1Detailed Description	571
10.121.2Member Typedef Documentation	572
10.121.2.1bkd_piter	572
10.121.2.2fwd_piter	572

10.121.2.3psite . . . . .	572
10.121.2.4site . . . . .	572
10.121.3Member Function Documentation . . . . .	572
10.121.3.1has . . . . .	572
10.122In::doc::Value_Iterator< E > Struct Template Reference . . . . .	572
10.122.1Detailed Description . . . . .	573
10.122.2Member Typedef Documentation . . . . .	574
10.122.2.1value . . . . .	574
10.122.3Member Function Documentation . . . . .	574
10.122.3.1invalidate . . . . .	574
10.122.3.2is_valid . . . . .	574
10.122.3.3operator value . . . . .	574
10.122.3.4start . . . . .	574
10.123In::doc::Value_Set< E > Struct Template Reference . . . . .	574
10.123.1Detailed Description . . . . .	575
10.123.2Member Typedef Documentation . . . . .	576
10.123.2.1bkd_viter . . . . .	576
10.123.2.2fwd_viter . . . . .	576
10.123.2.3value . . . . .	576
10.123.3Member Function Documentation . . . . .	576
10.123.3.1has . . . . .	576
10.123.3.2index_of . . . . .	576
10.123.3.3nvalues . . . . .	576
10.123.3.4operator[] . . . . .	576
10.124In::doc::Weighted_Window< E > Struct Template Reference . . . . .	577
10.124.1Detailed Description . . . . .	578
10.124.2Member Typedef Documentation . . . . .	578
10.124.2.1bkd_qiter . . . . .	578
10.124.2.2dpoint . . . . .	578
10.124.2.3fwd_qiter . . . . .	578
10.124.2.4point . . . . .	578
10.124.2.5weight . . . . .	578
10.124.2.6window . . . . .	578
10.124.3Member Function Documentation . . . . .	579
10.124.3.1delta . . . . .	579
10.124.3.2is_centered . . . . .	579

10.124.3.3	<code>is_empty</code>	579
10.124.3.4	<code>sym</code>	579
10.124.3.5	<code>win</code>	579
10.125	<code>mln::doc::Window&lt; E &gt; Struct Template Reference</code>	579
10.125.1	Detailed Description	580
10.125.2	Member Typedef Documentation	580
10.125.2.1	<code>bkd_qiter</code>	580
10.125.2.2	<code>fwd_qiter</code>	580
10.125.2.3	<code>qiter</code>	580
10.126	<code>mln::Dpoint&lt; E &gt; Struct Template Reference</code>	580
10.126.1	Detailed Description	581
10.126.2	Member Function Documentation	581
10.126.2.1	<code>lto_dpoint</code>	581
10.127	<code>mln::dpoint&lt; G, C &gt; Struct Template Reference</code>	581
10.127.1	Detailed Description	583
10.127.2	Member Typedef Documentation	583
10.127.2.1	<code>lcoord</code>	583
10.127.2.2	<code>grid</code>	584
10.127.2.3	<code>psite</code>	584
10.127.2.4	<code>site</code>	584
10.127.2.5	<code>vec</code>	584
10.127.3	Member Enumeration Documentation	584
10.127.3.1	<code>"@22</code>	584
10.127.4	Constructor & Destructor Documentation	584
10.127.4.1	<code>ldpoint</code>	584
10.127.4.2	<code>2dpoint</code>	584
10.127.4.3	<code>3dpoint</code>	584
10.127.4.4	<code>4dpoint</code>	585
10.127.4.5	<code>5dpoint</code>	585
10.127.5	Member Function Documentation	585
10.127.5.1	<code>operator mln::algebra::vec&lt; dpoint&lt; G, C &gt;::dim, Q &gt;</code>	585
10.127.5.2	<code>operator[]</code>	585
10.127.5.3	<code>operator[]</code>	585
10.127.5.4	<code>set_all</code>	585
10.127.5.5	<code>to_vec</code>	586
10.128	<code>mln::dpoints_bkd_pixter&lt; I &gt; Class Template Reference</code>	586

10.128. Detailed Description	587
10.128. Constructor & Destructor Documentation	587
10.128.2.1dpoints_bkd_piter	587
10.128.2.2dpoints_bkd_piter	587
10.128. Member Function Documentation	587
10.128.3.1center_val	587
10.128.3.2invalidate	587
10.128.3.3is_valid	587
10.128.3.4next	588
10.128.3.5start	588
10.128.3.6update	588
10.129. <code>mln::dpoints_fwd_piter&lt; I &gt;</code> Class Template Reference	588
10.129. Detailed Description	589
10.129. Constructor & Destructor Documentation	589
10.129.2.1dpoints_fwd_piter	589
10.129.2.2dpoints_fwd_piter	589
10.129. Member Function Documentation	590
10.129.3.1center_val	590
10.129.3.2invalidate	590
10.129.3.3is_valid	590
10.129.3.4next	590
10.129.3.5start	590
10.129.3.6update	590
10.130. <code>mln::dpsites_bkd_piter&lt; V &gt;</code> Class Template Reference	590
10.130. Detailed Description	591
10.130. Constructor & Destructor Documentation	591
10.130.2.1dpsites_bkd_piter	591
10.130.2.2dpsites_bkd_piter	591
10.130. Member Function Documentation	591
10.130.3.1next	591
10.131. <code>mln::dpsites_fwd_piter&lt; V &gt;</code> Class Template Reference	592
10.131. Detailed Description	592
10.131. Constructor & Destructor Documentation	592
10.131.2.1dpsites_fwd_piter	592
10.131.2.2dpsites_fwd_piter	593
10.131. Member Function Documentation	593

10.131.3. <code>lnext</code> . . . . .	593
10.131. <code>ln::Edge&lt; E &gt;</code> Struct Template Reference . . . . .	593
10.132. Detailed Description . . . . .	593
10.132. <code>ln::edge_image&lt; P, V, G &gt;</code> Class Template Reference . . . . .	593
10.133. Detailed Description . . . . .	594
10.133.2. Member Typedef Documentation . . . . .	594
10.133.2.1. <code>ledge_nbh_t</code> . . . . .	594
10.133.2.2. <code>ledge_win_t</code> . . . . .	594
10.133.2.3. <code>lgraph_t</code> . . . . .	594
10.133.2.4. <code>lnbh_t</code> . . . . .	595
10.133.2.5. <code>lsite_function_t</code> . . . . .	595
10.133.2.6. <code>lskeleton</code> . . . . .	595
10.133.2.7. <code>lwin_t</code> . . . . .	595
10.133.3. Constructor & Destructor Documentation . . . . .	595
10.133.3.1. <code>ledge_image</code> . . . . .	595
10.133.4. Member Function Documentation . . . . .	595
10.133.4.1. <code>loperator()</code> . . . . .	595
10.134. <code>ln::extended&lt; I &gt;</code> Struct Template Reference . . . . .	595
10.134.1. Detailed Description . . . . .	596
10.134.2. Member Typedef Documentation . . . . .	596
10.134.2.1. <code>lskeleton</code> . . . . .	596
10.134.2.2. <code>lvalue</code> . . . . .	596
10.134.3. Constructor & Destructor Documentation . . . . .	596
10.134.3.1. <code>lextended</code> . . . . .	596
10.134.3.2. <code>2extended</code> . . . . .	596
10.134.4. Member Function Documentation . . . . .	597
10.134.4.1. <code>ldomain</code> . . . . .	597
10.135. <code>ln::extension_fun&lt; I, F &gt;</code> Class Template Reference . . . . .	597
10.135.1. Detailed Description . . . . .	598
10.135.2. Member Typedef Documentation . . . . .	598
10.135.2.1. <code>lrvalue</code> . . . . .	598
10.135.2.2. <code>2skeleton</code> . . . . .	598
10.135.2.3. <code>lvalue</code> . . . . .	598
10.135.3. Constructor & Destructor Documentation . . . . .	598
10.135.3.1. <code>lextension_fun</code> . . . . .	598
10.135.3.2. <code>2extension_fun</code> . . . . .	598

10.135.4	Member Function Documentation	598
10.135.4.1	extension	598
10.135.4.2	has	598
10.135.4.3	operator()	599
10.135.4.4	operator()	599
10.136	ln::extension_ima< I, J > Class Template Reference	599
10.136.1	Detailed Description	600
10.136.2	Member Typedef Documentation	600
10.136.2.1	rvalue	600
10.136.2.2	skeleton	600
10.136.2.3	value	600
10.136.3	Constructor & Destructor Documentation	600
10.136.3.1	extension_ima	600
10.136.3.2	extension_ima	600
10.136.4	Member Function Documentation	600
10.136.4.1	extension	600
10.136.4.2	has	601
10.136.4.3	operator()	601
10.136.4.4	operator()	601
10.137	ln::extension_val< I > Class Template Reference	601
10.137.1	Detailed Description	602
10.137.2	Member Typedef Documentation	602
10.137.2.1	rvalue	602
10.137.2.2	skeleton	602
10.137.2.3	value	602
10.137.3	Constructor & Destructor Documentation	602
10.137.3.1	extension_val	602
10.137.3.2	extension_val	602
10.137.4	Member Function Documentation	603
10.137.4.1	lchange_extension	603
10.137.4.2	extension	603
10.137.4.3	has	603
10.137.4.4	operator()	603
10.137.4.5	operator()	603
10.138	ln::faces_site< N, D, P > Class Template Reference	603
10.138.1	Detailed Description	604



10.138.1	Constructor & Destructor Documentation	604
10.138.2.1	ifaces_psite	604
10.138.2.2	faces_psite	604
10.138.3	Member Function Documentation	604
10.138.3.1	lchange_target	604
10.138.3.2	face	605
10.138.3.3	face_id	605
10.138.3.4	invalidate	605
10.138.3.5	is_valid	605
10.138.3.6		605
10.138.3.7	site_set	605
10.139	mln::flat_image< T, S > Struct Template Reference	606
10.139.1	Detailed Description	606
10.139.2	Member Typedef Documentation	607
10.139.2.1	lvalue	607
10.139.2.2	rvalue	607
10.139.2.3	skeleton	607
10.139.2.4	value	607
10.139.3	Constructor & Destructor Documentation	607
10.139.3.1	lflat_image	607
10.139.3.2	rflat_image	607
10.139.4	Member Function Documentation	607
10.139.4.1	ldomain	607
10.139.4.2	has	607
10.139.4.3	operator()	607
10.139.4.4	operator()	608
10.140	mln::fun::from_accu< A > Struct Template Reference	608
10.140.1	Detailed Description	608
10.141	mln::fun::n2v::white_gaussian< V > Struct Template Reference	608
10.141.1	Detailed Description	609
10.142	mln::fun::p2b::antilogy Struct Reference	609
10.142.1	Detailed Description	610
10.143	mln::fun::p2b::tautology Struct Reference	610
10.143.1	Detailed Description	611
10.144	mln::fun::v2b::lnot< V > Struct Template Reference	611
10.144.1	Detailed Description	612

10.145	<code>mln::fun::v2b::threshold&lt; V &gt;</code> Struct Template Reference	612
	10.145. Detailed Description	613
10.146	<code>mln::fun::v2v::ch_function_value&lt; F, V &gt;</code> Class Template Reference	613
	10.146. Detailed Description	614
10.147	<code>mln::fun::v2v::component&lt; T, i &gt;</code> Struct Template Reference	614
	10.147. Detailed Description	615
10.148	<code>mln::fun::v2v::l1_norm&lt; V, R &gt;</code> Struct Template Reference	615
	10.148. Detailed Description	616
10.149	<code>mln::fun::v2v::l2_norm&lt; V, R &gt;</code> Struct Template Reference	616
	10.149. Detailed Description	617
10.150	<code>mln::fun::v2v::linear&lt; V, T, R &gt;</code> Struct Template Reference	617
	10.150. Detailed Description	618
10.151	<code>mln::fun::v2v::linfty_norm&lt; V, R &gt;</code> Struct Template Reference	618
	10.151. Detailed Description	619
10.152	<code>mln::fun::v2w2v::cos&lt; V &gt;</code> Struct Template Reference	619
	10.152. Detailed Description	620
10.153	<code>mln::fun::v2w_w2v::l1_norm&lt; V, R &gt;</code> Struct Template Reference	620
	10.153. Detailed Description	621
10.154	<code>mln::fun::v2w_w2v::l2_norm&lt; V, R &gt;</code> Struct Template Reference	621
	10.154. Detailed Description	622
10.155	<code>mln::fun::v2w_w2v::linfty_norm&lt; V, R &gt;</code> Struct Template Reference	622
	10.155. Detailed Description	623
10.156	<code>mln::fun::vv2b::eq&lt; L, R &gt;</code> Struct Template Reference	623
	10.156. Detailed Description	624
10.157	<code>mln::fun::vv2b::ge&lt; L, R &gt;</code> Struct Template Reference	624
	10.157. Detailed Description	625
10.158	<code>mln::fun::vv2b::gt&lt; L, R &gt;</code> Struct Template Reference	625
	10.158. Detailed Description	626
10.159	<code>mln::fun::vv2b::implies&lt; L, R &gt;</code> Struct Template Reference	626
	10.159. Detailed Description	627
10.160	<code>mln::fun::vv2b::le&lt; L, R &gt;</code> Struct Template Reference	627
	10.160. Detailed Description	628
10.161	<code>mln::fun::vv2b::lt&lt; L, R &gt;</code> Struct Template Reference	628
	10.161. Detailed Description	629
10.162	<code>mln::fun::vv2v::diff_abs&lt; V &gt;</code> Struct Template Reference	629
	10.162. Detailed Description	630

10.163	<code>std::land&lt; L, R &gt;</code> Struct Template Reference . . . . .	630
	10.163. Detailed Description . . . . .	631
10.164	<code>std::land_not&lt; L, R &gt;</code> Struct Template Reference . . . . .	631
	10.164. Detailed Description . . . . .	632
10.165	<code>std::lor&lt; L, R &gt;</code> Struct Template Reference . . . . .	632
	10.165. Detailed Description . . . . .	633
10.166	<code>std::lxor&lt; L, R &gt;</code> Struct Template Reference . . . . .	633
	10.166. Detailed Description . . . . .	634
10.167	<code>std::max&lt; V &gt;</code> Struct Template Reference . . . . .	634
	10.167. Detailed Description . . . . .	635
10.168	<code>std::min&lt; L, R &gt;</code> Struct Template Reference . . . . .	635
	10.168. Detailed Description . . . . .	636
10.169	<code>std::vec&lt; V &gt;</code> Struct Template Reference . . . . .	636
	10.169. Detailed Description . . . . .	637
10.170	<code>std::closest_point&lt; P &gt;</code> Struct Template Reference . . . . .	637
	10.170. Detailed Description . . . . .	638
10.171	<code>std::bilinear&lt; I &gt;</code> Struct Template Reference . . . . .	638
	10.171. Detailed Description . . . . .	638
	10.171. Member Function Documentation . . . . .	638
	10.171.2. <code>operator()</code> . . . . .	638
	10.171.2.2. <code>operator()</code> . . . . .	638
10.172	<code>std::trilinear&lt; I &gt;</code> Struct Template Reference . . . . .	639
	10.172. Detailed Description . . . . .	639
10.173	<code>std::composed&lt; T2, T1 &gt;</code> Struct Template Reference . . . . .	639
	10.173. Detailed Description . . . . .	639
	10.173. Constructor & Destructor Documentation . . . . .	639
	10.173.2. <code>lcomposed</code> . . . . .	639
	10.173.2.2. <code>composed</code> . . . . .	640
10.174	<code>std::linear&lt; I &gt;</code> Struct Template Reference . . . . .	640
	10.174. Detailed Description . . . . .	640
	10.174. Constructor & Destructor Documentation . . . . .	640
	10.174.2. <code>llinear</code> . . . . .	640
	10.174. Member Function Documentation . . . . .	641
	10.174.3. <code>operator()</code> . . . . .	641
	10.174. Member Data Documentation . . . . .	641
	10.174.4. <code>lma</code> . . . . .	641

10.175	<code>mln::fun::x2x::rotation&lt; n, C &gt;</code> Struct Template Reference	641
10.175.1	Detailed Description	643
10.175.2	Member Typedef Documentation	643
10.175.2.1	<code>ldata_t</code>	643
10.175.2.2	<code>invert</code>	643
10.175.3	Constructor & Destructor Documentation	643
10.175.3.1	<code>rotation</code>	643
10.175.3.2	<code>rotation</code>	643
10.175.3.3	<code>rotation</code>	644
10.175.3.4	<code>rotation</code>	644
10.175.4	Member Function Documentation	644
10.175.4.1	<code>inv</code>	644
10.175.4.2	<code>operator()</code>	644
10.175.4.3	<code>set_alpha</code>	644
10.175.4.4	<code>set_axis</code>	644
10.176	<code>mln::fun::x2x::translation&lt; n, C &gt;</code> Struct Template Reference	644
10.176.1	Detailed Description	646
10.176.2	Member Typedef Documentation	646
10.176.2.1	<code>ldata_t</code>	646
10.176.2.2	<code>invert</code>	646
10.176.3	Constructor & Destructor Documentation	646
10.176.3.1	<code>translation</code>	646
10.176.3.2	<code>translation</code>	646
10.176.4	Member Function Documentation	647
10.176.4.1	<code>inv</code>	647
10.176.4.2	<code>operator()</code>	647
10.176.4.3	<code>set_t</code>	647
10.176.4.4	<code>t</code>	647
10.177	<code>mln::fun_image&lt; F, I &gt;</code> Struct Template Reference	647
10.177.1	Detailed Description	648
10.177.2	Member Typedef Documentation	648
10.177.2.1	<code>lvalue</code>	648
10.177.2.2	<code>rvalue</code>	648
10.177.2.3	<code>skeleton</code>	648
10.177.2.4	<code>value</code>	648
10.177.3	Constructor & Destructor Documentation	648

10.177.3.1fun_image . . . . .	648
10.177.3.2fun_image . . . . .	649
10.177.3.3fun_image . . . . .	649
10.177.4Member Function Documentation . . . . .	649
10.177.4.1operator() . . . . .	649
10.177.4.2operator() . . . . .	649
10.178mIn::Function< E > Struct Template Reference . . . . .	649
10.178.1Detailed Description . . . . .	649
10.178.2Constructor & Destructor Documentation . . . . .	650
10.178.2.1Function . . . . .	650
10.179mIn::Function< void > Struct Template Reference . . . . .	650
10.179.1Detailed Description . . . . .	650
10.180mIn::Function_n2v< E > Struct Template Reference . . . . .	650
10.180.1Detailed Description . . . . .	651
10.181mIn::Function_v2b< E > Struct Template Reference . . . . .	651
10.181.1Detailed Description . . . . .	652
10.182mIn::Function_v2v< E > Struct Template Reference . . . . .	652
10.182.1Detailed Description . . . . .	653
10.183mIn::Function_vv2b< E > Struct Template Reference . . . . .	653
10.183.1Detailed Description . . . . .	653
10.184mIn::Function_vv2v< E > Struct Template Reference . . . . .	653
10.184.1Detailed Description . . . . .	654
10.185mIn::fwd_pixter1d< I > Class Template Reference . . . . .	654
10.185.1Detailed Description . . . . .	655
10.185.2Member Typedef Documentation . . . . .	655
10.185.2.1image . . . . .	655
10.185.3Constructor & Destructor Documentation . . . . .	655
10.185.3.1fwd_pixter1d . . . . .	655
10.185.4Member Function Documentation . . . . .	655
10.185.4.1next . . . . .	655
10.186mIn::fwd_pixter2d< I > Class Template Reference . . . . .	655
10.186.1Detailed Description . . . . .	656
10.186.2Member Typedef Documentation . . . . .	656
10.186.2.1image . . . . .	656
10.186.3Constructor & Destructor Documentation . . . . .	656
10.186.3.1fwd_pixter2d . . . . .	656

10.186.	Member Function Documentation	656
10.186.4.	lnext	656
10.187.	ln::fwd_pixter3d< I > Class Template Reference	657
10.187.	Detailed Description	657
10.187.	Member Typedef Documentation	657
10.187.2.	limage	657
10.187.	Constructor & Destructor Documentation	657
10.187.3.	lfwd_pixter3d	657
10.187.	Member Function Documentation	658
10.187.4.	lnext	658
10.188.	ln::Gdpoint< E > Struct Template Reference	658
10.188.	Detailed Description	659
10.189.	ln::Gdpoint< void > Struct Template Reference	659
10.189.	Detailed Description	659
10.190.	ln::Generalized_Pixel< E > Struct Template Reference	659
10.190.	Detailed Description	659
10.191.	ln::geom::complex_geometry< D, P > Class Template Reference	660
10.191.	Detailed Description	660
10.191.	Constructor & Destructor Documentation	661
10.191.2.	lcomplex_geometry	661
10.191.	Member Function Documentation	661
10.191.3.	ladd_location	661
10.191.3.2.	operator()	661
10.192.	ln::Gpoint< E > Struct Template Reference	661
10.192.	Detailed Description	663
10.192.	Friends And Related Function Documentation	663
10.192.2.	loperator+	663
10.192.2.2.	operator+=	663
10.192.2.3.	operator-	664
10.192.2.4.	operator-=	664
10.192.2.5.	operator/	665
10.192.2.6.	operator<<	665
10.192.2.7.	operator==	665
10.193.	ln::Graph< E > Struct Template Reference	666
10.193.	Detailed Description	666
10.194.	ln::graph::attribute::card_t Struct Reference	666

---

10.194. Detailed Description	667
10.194. Member Typedef Documentation	667
10.194.2. lresult	667
10.195. <a href="#">Inl::graph::attribute::representative_t Struct Reference</a>	667
10.195. Detailed Description	667
10.195. Member Typedef Documentation	667
10.195.2. lresult	667
10.196. <a href="#">Inl::graph_elt_mixed_neighborhood&lt; G, S, S2 &gt; Struct Template Reference</a>	667
10.196. Detailed Description	668
10.196. Member Typedef Documentation	669
10.196.2. lbkd_niter	669
10.196.2. fwd_niter	669
10.196.2. niter	669
10.197. <a href="#">Inl::graph_elt_mixed_window&lt; G, S, S2 &gt; Class Template Reference</a>	669
10.197. Detailed Description	670
10.197. Member Typedef Documentation	671
10.197.2. lbkd_qiter	671
10.197.2. center_t	671
10.197.2. fwd_qiter	671
10.197.2. graph_element	671
10.197.2. psite	671
10.197.2. qiter	671
10.197.2. site	671
10.197.2. target	671
10.197. Member Function Documentation	672
10.197.3. ldelta	672
10.197.3. is_centered	672
10.197.3. is_empty	672
10.197.3. is_symmetric	672
10.197.3. is_valid	672
10.197.3. sym	672
10.198. <a href="#">Inl::graph_elt_neighborhood&lt; G, S &gt; Struct Template Reference</a>	672
10.198. Detailed Description	673
10.198. Member Typedef Documentation	674
10.198.2. lbkd_niter	674
10.198.2. fwd_niter	674

10.198.2.3niter . . . . .	674
10.199.1. <a href="#">In::graph_elt_neighborhood_if&lt; G, S, I &gt; Struct Template Reference</a> . . . . .	674
10.199.1.1 Detailed Description . . . . .	675
10.199.1.2 Member Typedef Documentation . . . . .	675
10.199.1.2.1 bkd_niter . . . . .	675
10.199.1.2.2 fwd_niter . . . . .	675
10.199.1.2.3 niter . . . . .	675
10.199.1.3 Constructor & Destructor Documentation . . . . .	675
10.199.1.3.1 graph_elt_neighborhood_if . . . . .	675
10.199.1.3.2 graph_elt_neighborhood_if . . . . .	676
10.199.1.4 Member Function Documentation . . . . .	676
10.199.1.4.1 lmask . . . . .	676
10.200.1. <a href="#">In::graph_elt_window&lt; G, S &gt; Class Template Reference</a> . . . . .	676
10.200.1.1 Detailed Description . . . . .	677
10.200.1.2 Member Typedef Documentation . . . . .	678
10.200.1.2.1 bkd_qiter . . . . .	678
10.200.1.2.2 center_t . . . . .	678
10.200.1.2.3 fwd_qiter . . . . .	678
10.200.1.2.4 graph_element . . . . .	678
10.200.1.2.5 psite . . . . .	678
10.200.1.2.6 qiter . . . . .	678
10.200.1.2.7 site . . . . .	678
10.200.1.2.8 target . . . . .	678
10.200.1.3 Member Function Documentation . . . . .	679
10.200.1.3.1 delta . . . . .	679
10.200.1.3.2 is_centered . . . . .	679
10.200.1.3.3 is_empty . . . . .	679
10.200.1.3.4 is_symmetric . . . . .	679
10.200.1.3.5 is_valid . . . . .	679
10.200.1.3.6 sym . . . . .	679
10.201.1. <a href="#">In::graph_elt_window_if&lt; G, S, I &gt; Class Template Reference</a> . . . . .	679
10.201.1.1 Detailed Description . . . . .	681
10.201.1.2 Member Typedef Documentation . . . . .	681
10.201.1.2.1 bkd_qiter . . . . .	681
10.201.1.2.2 fwd_qiter . . . . .	682
10.201.1.2.3 mask_t . . . . .	682



10.201.2.4	psite	682
10.201.2.5	qiter	682
10.201.2.6	site	682
10.201.2.7	target	682
10.201.3	Constructor & Destructor Documentation	682
10.201.3.1	graph_elt_window_if	682
10.201.3.2	graph_elt_window_if	683
10.201.4	Member Function Documentation	683
10.201.4.1	change_mask	683
10.201.4.2	delta	683
10.201.4.3	is_centered	683
10.201.4.4	is_empty	683
10.201.4.5	is_symmetric	683
10.201.4.6	is_valid	683
10.201.4.7	mask	684
10.201.4.8	sym	684
10.202	mln::graph_window_base< P, E > Class Template Reference	684
10.202.1	Detailed Description	685
10.202.2	Member Typedef Documentation	685
10.202.2.1	site	685
10.202.3	Member Function Documentation	685
10.202.3.1	delta	685
10.202.3.2	is_centered	685
10.202.3.3	is_empty	685
10.202.3.4	is_symmetric	685
10.202.3.5	is_valid	685
10.202.3.6	sym	685
10.203	mln::graph_window_if_piter< S, W, I > Class Template Reference	686
10.203.1	Detailed Description	686
10.203.2	Member Typedef Documentation	686
10.203.2.1	IP	686
10.203.3	Constructor & Destructor Documentation	687
10.203.3.1	graph_window_if_piter	687
10.203.4	Member Function Documentation	687
10.203.4.1	element	687
10.203.4.2	id	687

10.203.4.3next	687
10.204.1. <code>ln::graph_window_piter&lt; S, W, I &gt;</code> Class Template Reference	687
10.204.1.1 Detailed Description	688
10.204.1.2 Member Typedef Documentation	689
10.204.2.1 <code>lcenter_t</code>	689
10.204.2.2 <code>lgraph_element</code>	689
10.204.2.3 <code>lP</code>	689
10.204.1.3 Constructor & Destructor Documentation	689
10.204.3.1 <code>lgraph_window_piter</code>	689
10.204.3.2 <code>lgraph_window_piter</code>	689
10.204.3.3 <code>lgraph_window_piter</code>	689
10.204.1.4 Member Function Documentation	690
10.204.4.1 <code>lchange_target_site_set</code>	690
10.204.4.2 <code>lgraph_element</code>	690
10.204.4.3 <code>lgraph_id</code>	690
10.204.4.4 <code>lgraph_next</code>	690
10.204.4.5 <code>lgraph_target_site_set</code>	690
10.205.1. <code>ln::hexa&lt; I &gt;</code> Struct Template Reference	690
10.205.1.1 Detailed Description	692
10.205.1.2 Member Typedef Documentation	692
10.205.2.1 <code>lhexa_piter</code>	692
10.205.2.2 <code>lhexa_fwd_piter</code>	692
10.205.2.3 <code>lhexa_value</code>	692
10.205.2.4 <code>lhexa_site</code>	692
10.205.2.5 <code>lhexa_value</code>	693
10.205.2.6 <code>lhexa_skeleton</code>	693
10.205.2.7 <code>lhexa_value</code>	693
10.205.1.3 Constructor & Destructor Documentation	693
10.205.3.1 <code>lhexa</code>	693
10.205.3.2 <code>lhexa</code>	693
10.205.1.4 Member Function Documentation	693
10.205.4.1 <code>lhexa_domain</code>	693
10.205.4.2 <code>lhexa_has</code>	693
10.205.4.3 <code>lhexa_operator()</code>	693
10.205.4.4 <code>lhexa_operator()</code>	693
10.206.1. <code>ln::histo::array&lt; T &gt;</code> Struct Template Reference	694

10.206. Detailed Description	694
10.207. <code>Image&lt; E &gt;</code> Struct Template Reference	694
10.207. Detailed Description	696
10.208. <code>image1d&lt; T &gt;</code> Struct Template Reference	696
10.208. Detailed Description	697
10.208. Member Typedef Documentation	697
10.208.2. <code>lvalue</code>	697
10.208.2. <code>rvalue</code>	698
10.208.2. <code>skeleton</code>	698
10.208.2. <code>value</code>	698
10.208. Constructor & Destructor Documentation	698
10.208.3. <code>image1d</code>	698
10.208.3. <code>image1d</code>	698
10.208.3. <code>image1d</code>	698
10.208. Member Function Documentation	698
10.208.4. <code>lbbox</code>	698
10.208.4. <code>border</code>	698
10.208.4. <code>buffer</code>	698
10.208.4. <code>buffer</code>	699
10.208.4. <code>delta_index</code>	699
10.208.4. <code>domain</code>	699
10.208.4. <code>element</code>	699
10.208.4. <code>element</code>	699
10.208.4. <code>has</code>	699
10.208.4. <code>elements</code>	699
10.208.4. <code>lrinds</code>	699
10.208.4. <code>operator()</code>	699
10.208.4. <code>operator()</code>	700
10.208.4. <code>point_at_index</code>	700
10.209. <code>image2d&lt; T &gt;</code> Class Template Reference	700
10.209. Detailed Description	702
10.209. Member Typedef Documentation	702
10.209.2. <code>lvalue</code>	702
10.209.2. <code>rvalue</code>	702
10.209.2. <code>skeleton</code>	702
10.209.2. <code>value</code>	702

10.209.3	Constructor & Destructor Documentation	702
10.209.3.1	image2d	702
10.209.3.2	image2d	702
10.209.3.3	image2d	702
10.209.4	Member Function Documentation	703
10.209.4.1	bbox	703
10.209.4.2	border	703
10.209.4.3	buffer	703
10.209.4.4	buffer	703
10.209.4.5	delta_index	703
10.209.4.6	domain	703
10.209.4.7	element	703
10.209.4.8	element	703
10.209.4.9	has	703
10.209.4.10	cols	704
10.209.4.11	elements	704
10.209.4.12	rows	704
10.209.4.13	operator()	704
10.209.4.14	operator()	704
10.209.4.15	point_at_index	704
10.210.1	In::image2d_h< V > Struct Template Reference	704
10.210.1.1	Detailed Description	706
10.210.2	Member Typedef Documentation	706
10.210.2.1	bkd_piter	706
10.210.2.2	fwd_piter	706
10.210.2.3	value	706
10.210.2.4	psite	706
10.210.2.5	rvalue	706
10.210.2.6	skeleton	707
10.210.2.7	value	707
10.210.3	Constructor & Destructor Documentation	707
10.210.3.1	image2d_h	707
10.210.4	Member Function Documentation	707
10.210.4.1	domain	707
10.210.4.2	has	707
10.210.4.3	operator()	707

10.210.4.operator()	707
10.211.Inn::image3d< T > Struct Template Reference	707
10.211.1.Detailed Description	709
10.211.2.Member Typedef Documentation	709
10.211.2.1.lvalue	709
10.211.2.2.rvalue	709
10.211.2.3.skeleton	709
10.211.2.4.value	710
10.211.3.Constructor & Destructor Documentation	710
10.211.3.1.image3d	710
10.211.3.2.image3d	710
10.211.3.3.image3d	710
10.211.4.Member Function Documentation	710
10.211.4.1.lbbox	710
10.211.4.2.border	710
10.211.4.3.buffer	710
10.211.4.4.buffer	710
10.211.4.5.delta_index	710
10.211.4.6.domain	711
10.211.4.7.element	711
10.211.4.8.element	711
10.211.4.9.has	711
10.211.4.10.cols	711
10.211.4.11.elements	711
10.211.4.12.rows	711
10.211.4.13.slices	711
10.211.4.14.operator()	711
10.211.4.15.operator()	712
10.211.4.16.point_at_index	712
10.212.Inn::image_if< I, F > Struct Template Reference	712
10.212.1.Detailed Description	712
10.212.2.Member Typedef Documentation	713
10.212.2.1.skeleton	713
10.212.3.Constructor & Destructor Documentation	713
10.212.3.1.image_if	713
10.212.3.2.image_if	713

10.212.4	Member Function Documentation	713
10.212.4.1	domain	713
10.212.4.2	operator image_if< const I, F >	713
10.213	mln::interpolated< I, F > Struct Template Reference	713
10.213.1	Detailed Description	714
10.213.2	Member Typedef Documentation	714
10.213.2.1	lvalue	714
10.213.2.2	psite	714
10.213.2.3	rvalue	714
10.213.2.4	skeleton	714
10.213.2.5	value	715
10.213.3	Constructor & Destructor Documentation	715
10.213.3.1	interpolated	715
10.213.4	Member Function Documentation	715
10.213.4.1	lhas	715
10.213.4.2	is_valid	715
10.214	mln::io::dicom::dicom_header Struct Reference	715
10.214.1	Detailed Description	715
10.215	mln::io::dump::dump_header Struct Reference	715
10.215.1	Detailed Description	715
10.216	mln::io::fld::fld_header Struct Reference	716
10.216.1	Detailed Description	716
10.217	mln::io::raw::raw_header Struct Reference	716
10.217.1	Detailed Description	716
10.218	mln::Iterator< E > Struct Template Reference	716
10.218.1	Detailed Description	717
10.218.2	Member Function Documentation	718
10.218.2.1	lnext	718
10.219	mln::labeled_image< I > Class Template Reference	718
10.219.1	Detailed Description	719
10.219.2	Member Typedef Documentation	720
10.219.2.1	lbbox_t	720
10.219.2.2	skeleton	720
10.219.3	Constructor & Destructor Documentation	720
10.219.3.1	labeled_image	720
10.219.3.2	labeled_image	720

10.219.3.labeled_image . . . . .	720
10.219.4.Member Function Documentation . . . . .	720
10.219.4.1.lbbox . . . . .	720
10.219.4.2.lbboxes . . . . .	721
10.219.4.3.llabels . . . . .	721
10.219.4.4.relabel . . . . .	721
10.219.4.5.relabel . . . . .	721
10.219.4.6.subdomain . . . . .	721
10.219.4.7.update_data . . . . .	721
10.220.mln::labeled_image_base< I, E > Class Template Reference . . . . .	721
10.220.1.Detailed Description . . . . .	723
10.220.2.Member Typedef Documentation . . . . .	723
10.220.2.1.lbbox_t . . . . .	723
10.220.3.Constructor & Destructor Documentation . . . . .	723
10.220.3.1.labeled_image_base . . . . .	723
10.220.4.Member Function Documentation . . . . .	723
10.220.4.1.lbbox . . . . .	723
10.220.4.2.lbboxes . . . . .	724
10.220.4.3.llabels . . . . .	724
10.220.4.4.relabel . . . . .	724
10.220.4.5.relabel . . . . .	724
10.220.4.6.subdomain . . . . .	724
10.220.4.7.update_data . . . . .	724
10.221.mln::lazy_image< I, F, B > Struct Template Reference . . . . .	725
10.221.1.Detailed Description . . . . .	726
10.221.2.Member Typedef Documentation . . . . .	726
10.221.2.1.lvalue . . . . .	726
10.221.2.2.rvalue . . . . .	726
10.221.2.3.skeleton . . . . .	726
10.221.3.Constructor & Destructor Documentation . . . . .	726
10.221.3.1.lazy_image . . . . .	726
10.221.3.2.lazy_image . . . . .	726
10.221.4.Member Function Documentation . . . . .	726
10.221.4.1.ldomain . . . . .	726
10.221.4.2.has . . . . .	727
10.221.4.3.operator() . . . . .	727

10.221.4.4operator()	727
10.221.4.5operator()	727
10.221.4.6operator()	727
10.222mln::Literal< E > Struct Template Reference	727
10.222. Detailed Description	729
10.223mln::literal::black_t Struct Reference	729
10.223. Detailed Description	729
10.224mln::literal::blue_t Struct Reference	730
10.224. Detailed Description	730
10.225mln::literal::brown_t Struct Reference	730
10.225. Detailed Description	731
10.226mln::literal::cyan_t Struct Reference	731
10.226. Detailed Description	732
10.227mln::literal::green_t Struct Reference	732
10.227. Detailed Description	733
10.228mln::literal::identity_t Struct Reference	733
10.228. Detailed Description	734
10.229mln::literal::light_gray_t Struct Reference	734
10.229. Detailed Description	735
10.230mln::literal::lime_t Struct Reference	735
10.230. Detailed Description	736
10.231mln::literal::magenta_t Struct Reference	736
10.231. Detailed Description	737
10.232mln::literal::max_t Struct Reference	737
10.232. Detailed Description	738
10.233mln::literal::min_t Struct Reference	738
10.233. Detailed Description	739
10.234mln::literal::olive_t Struct Reference	739
10.234. Detailed Description	740
10.235mln::literal::one_t Struct Reference	740
10.235. Detailed Description	741
10.236mln::literal::orange_t Struct Reference	741
10.236. Detailed Description	742
10.237mln::literal::origin_t Struct Reference	742
10.237. Detailed Description	743
10.238mln::literal::pink_t Struct Reference	743



10.238. Detailed Description	744
10.239. <code>ln::literal::purple_t</code> Struct Reference	744
10.239. Detailed Description	745
10.240. <code>ln::literal::red_t</code> Struct Reference	745
10.240. Detailed Description	746
10.241. <code>ln::literal::teal_t</code> Struct Reference	746
10.241. Detailed Description	747
10.242. <code>ln::literal::violet_t</code> Struct Reference	747
10.242. Detailed Description	748
10.243. <code>ln::literal::white_t</code> Struct Reference	748
10.243. Detailed Description	749
10.244. <code>ln::literal::yellow_t</code> Struct Reference	749
10.244. Detailed Description	750
10.245. <code>ln::literal::zero_t</code> Struct Reference	750
10.245. Detailed Description	751
10.246. <code>ln::Mesh&lt; E &gt;</code> Struct Template Reference	751
10.246. Detailed Description	752
10.247. <code>ln::Meta_Accumulator&lt; E &gt;</code> Struct Template Reference	752
10.247. Detailed Description	753
10.248. <code>ln::Meta_Function&lt; E &gt;</code> Struct Template Reference	754
10.248. Detailed Description	754
10.249. <code>ln::Meta_Function_v2v&lt; E &gt;</code> Struct Template Reference	754
10.249. Detailed Description	755
10.250. <code>ln::Meta_Function_vv2v&lt; E &gt;</code> Struct Template Reference	755
10.250. Detailed Description	755
10.251. <code>ln::metal::ands&lt; E1, E2, E3, E4, E5, E6, E7, E8 &gt;</code> Struct Template Reference	756
10.251. Detailed Description	756
10.252. <code>ln::metal::converts_to&lt; T, U &gt;</code> Struct Template Reference	756
10.252. Detailed Description	756
10.253. <code>ln::metal::equal&lt; T1, T2 &gt;</code> Struct Template Reference	756
10.253. Detailed Description	757
10.254. <code>ln::metal::goes_to&lt; T, U &gt;</code> Struct Template Reference	757
10.254. Detailed Description	757
10.255. <code>ln::metal::is&lt; T, U &gt;</code> Struct Template Reference	757
10.255. Detailed Description	757
10.256. <code>ln::metal::is_a&lt; T, M &gt;</code> Struct Template Reference	757

10.256. Detailed Description	757
10.257. <code>ln::metal::is_not&lt; T, U &gt;</code> Struct Template Reference	757
10.257. Detailed Description	758
10.258. <code>ln::metal::is_not_a&lt; T, M &gt;</code> Struct Template Reference	758
10.258. Detailed Description	758
10.259. <code>ln::mixed_neighb&lt; W &gt;</code> Class Template Reference	758
10.259. Detailed Description	759
10.259. Member Typedef Documentation	759
10.259.2. <code>l_bkd_niter</code>	759
10.259.2.2. <code>2fwd_niter</code>	759
10.259.2.3. <code>3niter</code>	759
10.259. Constructor & Destructor Documentation	759
10.259.3. <code>l_mixed_neighb</code>	759
10.259.3.2. <code>2mixed_neighb</code>	759
10.260. <code>ln::morpho::attribute::card&lt; I &gt;</code> Class Template Reference	759
10.260. Detailed Description	760
10.260. Member Function Documentation	760
10.260.2. <code>linit</code>	760
10.260.2.2. <code>2is_valid</code>	760
10.260.2.3. <code>3take_as_init</code>	760
10.260.2.4. <code>4take_n_times</code>	760
10.260.2.5. <code>5to_result</code>	760
10.261. <code>ln::morpho::attribute::count_adjacent_vertices&lt; I &gt;</code> Struct Template Reference	761
10.261. Detailed Description	761
10.261. Member Function Documentation	761
10.261.2. <code>linit</code>	761
10.261.2.2. <code>2is_valid</code>	761
10.261.2.3. <code>3take_as_init</code>	762
10.261.2.4. <code>4take_n_times</code>	762
10.261.2.5. <code>5to_result</code>	762
10.262. <code>ln::morpho::attribute::height&lt; I &gt;</code> Struct Template Reference	762
10.262. Detailed Description	763
10.262. Member Function Documentation	763
10.262.2. <code>lbase_level</code>	763
10.262.2.2. <code>2nit</code>	763
10.262.2.3. <code>3is_valid</code>	763

10.262.2.4take_as_init . . . . .	763
10.262.2.5take_n_times . . . . .	763
10.262.2.6to_result . . . . .	763
10.263. <a href="#">mln::morpho::attribute::sharpness&lt; I &gt; Struct Template Reference</a> . . . . .	764
10.263. Detailed Description . . . . .	764
10.263. Member Function Documentation . . . . .	764
10.263.2.1area . . . . .	764
10.263.2.2height . . . . .	765
10.263.2.3init . . . . .	765
10.263.2.4is_valid . . . . .	765
10.263.2.5take_as_init . . . . .	765
10.263.2.6take_n_times . . . . .	765
10.263.2.7to_result . . . . .	765
10.263.2.8volume . . . . .	765
10.264. <a href="#">mln::morpho::attribute::sum&lt; I, S &gt; Class Template Reference</a> . . . . .	765
10.264. Detailed Description . . . . .	766
10.264. Member Function Documentation . . . . .	766
10.264.2.1init . . . . .	766
10.264.2.2is_valid . . . . .	766
10.264.2.3set_value . . . . .	767
10.264.2.4take_as_init . . . . .	767
10.264.2.5take_n_times . . . . .	767
10.264.2.6to_result . . . . .	767
10.264.2.7untake . . . . .	767
10.265. <a href="#">mln::morpho::attribute::volume&lt; I &gt; Struct Template Reference</a> . . . . .	767
10.265. Detailed Description . . . . .	768
10.265. Member Function Documentation . . . . .	768
10.265.2.1area . . . . .	768
10.265.2.2init . . . . .	768
10.265.2.3is_valid . . . . .	768
10.265.2.4take_as_init . . . . .	768
10.265.2.5take_n_times . . . . .	768
10.265.2.6to_result . . . . .	769
10.266. <a href="#">mln::neighb&lt; W &gt; Class Template Reference</a> . . . . .	769
10.266. Detailed Description . . . . .	769
10.266. Member Typedef Documentation . . . . .	770

10.266.2.1bkd_niter . . . . .	770
10.266.2.2fwd_niter . . . . .	770
10.266.2.3niter . . . . .	770
10.266.3Constructor & Destructor Documentation . . . . .	770
10.266.3.1neighb . . . . .	770
10.266.3.2neighb . . . . .	770
10.267In::Neighborhood< E > Struct Template Reference . . . . .	770
10.267.1Detailed Description . . . . .	771
10.268In::Neighborhood< void > Struct Template Reference . . . . .	771
10.268.1Detailed Description . . . . .	771
10.269In::Object< E > Struct Template Reference . . . . .	771
10.269.1Detailed Description . . . . .	771
10.270In::p2p_image< I, F > Struct Template Reference . . . . .	771
10.270.1Detailed Description . . . . .	772
10.270.2Member Typedef Documentation . . . . .	772
10.270.2.1skeleton . . . . .	772
10.270.3Constructor & Destructor Documentation . . . . .	772
10.270.3.1p2p_image . . . . .	772
10.270.3.2p2p_image . . . . .	773
10.270.4Member Function Documentation . . . . .	773
10.270.4.1domain . . . . .	773
10.270.4.2fun . . . . .	773
10.270.4.3operator() . . . . .	773
10.270.4.4operator() . . . . .	773
10.271In::p_array< P > Class Template Reference . . . . .	773
10.271.1Detailed Description . . . . .	775
10.271.2Member Typedef Documentation . . . . .	775
10.271.2.1bkd_piter . . . . .	775
10.271.2.2element . . . . .	775
10.271.2.3fwd_piter . . . . .	775
10.271.2.4element . . . . .	775
10.271.2.5piter . . . . .	775
10.271.2.6psite . . . . .	776
10.271.3Constructor & Destructor Documentation . . . . .	776
10.271.3.1p_array . . . . .	776
10.271.3.2p_array . . . . .	776

10.271.4	Member Function Documentation	776
10.271.4.1	append	776
10.271.4.2	append	776
10.271.4.3	change	776
10.271.4.4	clear	776
10.271.4.5	has	776
10.271.4.6	has	777
10.271.4.7	insert	777
10.271.4.8	is_valid	777
10.271.4.9	memory_size	777
10.271.4.10	sites	777
10.271.4.11	operator[]	777
10.271.4.12	operator[]	777
10.271.4.13	operator[]	777
10.271.4.14	reserve	778
10.271.4.15	size	778
10.271.4.16	id_vector	778
10.271	mln::p_centered< W > Class Template Reference	778
10.272.	Detailed Description	779
10.272.	Member Typedef Documentation	779
10.272.2.	l_bkd_piter	779
10.272.2.	element	779
10.272.2.	3fwd_piter	779
10.272.2.	4piter	780
10.272.2.	5psite	780
10.272.2.	6site	780
10.272.	Constructor & Destructor Documentation	780
10.272.3.	lp_centered	780
10.272.3.	2p_centered	780
10.272.	Member Function Documentation	780
10.272.4.	lcenter	780
10.272.4.	2has	780
10.272.4.	3is_valid	780
10.272.4.	4memory_size	780
10.272.4.	5window	781
10.271	mln::p_complex< D, G > Class Template Reference	781

10.273. Detailed Description	782
10.273. Member Typedef Documentation	782
10.273.2. l_bkd_piter	782
10.273.2. 2element	782
10.273.2. 3fwd_piter	782
10.273.2. 4piter	782
10.273.2. 5psite	783
10.273. Constructor & Destructor Documentation	783
10.273.3. lp_complex	783
10.273. Member Function Documentation	783
10.273.4. lcplx	783
10.273.4. 2cplx	783
10.273.4. 3geom	783
10.273.4. 4has	783
10.273.4. 5is_valid	784
10.273.4. 6faces	784
10.273.4. 7nfaces_of_dim	784
10.273.4. 8nsites	784
10.274. In::p_edges< G, F > Class Template Reference	784
10.274. Detailed Description	786
10.274. Member Typedef Documentation	786
10.274.2. l_bkd_piter	786
10.274.2. 2edge	786
10.274.2. 3element	786
10.274.2. 4fun_t	786
10.274.2. 5fwd_piter	786
10.274.2. 6graph_element	787
10.274.2. 7graph_t	787
10.274.2. 8piter	787
10.274.2. 9psite	787
10.274. Constructor & Destructor Documentation	787
10.274.3. lp_edges	787
10.274.3. 2p_edges	787
10.274.3. 3p_edges	787
10.274.3. 4p_edges	788
10.274. Member Function Documentation	788

10.274.4.1function	788
10.274.4.2graph	788
10.274.4.3has	788
10.274.4.4has	788
10.274.4.5invalidate	788
10.274.4.6is_valid	789
10.274.4.7memory_size	789
10.274.4.8nedges	789
10.274.4.9nsites	789
10.275mln::p_faces< N, D, P > Struct Template Reference	789
10.275.1Detailed Description	790
10.275.2Member Typedef Documentation	790
10.275.2.1bkd_piter	790
10.275.2.2element	790
10.275.2.3fwd_piter	790
10.275.2.4piter	791
10.275.2.5psite	791
10.275.3Constructor & Destructor Documentation	791
10.275.3.1p_faces	791
10.275.3.2p_faces	791
10.275.4Member Function Documentation	791
10.275.4.1cplx	791
10.275.4.2cplx	791
10.275.4.3is_valid	792
10.275.4.4nfaces	792
10.275.4.5nsites	792
10.276mln::p_graph_piter< S, I > Class Template Reference	792
10.276.1Detailed Description	793
10.276.2Constructor & Destructor Documentation	793
10.276.2.1p_graph_piter	793
10.276.3Member Function Documentation	793
10.276.3.1graph	793
10.276.3.2id	793
10.276.3.3mln_q_subject	793
10.276.3.4next	793
10.277mln::p_if< S, F > Class Template Reference	793

10.277. Detailed Description . . . . .	794
10.277. Member Typedef Documentation . . . . .	795
10.277.2. l_bkd_piter . . . . .	795
10.277.2. 2element . . . . .	795
10.277.2. 3fwd_piter . . . . .	795
10.277.2. 4piter . . . . .	795
10.277.2. 5psite . . . . .	795
10.277. Constructor & Destructor Documentation . . . . .	795
10.277.3. 1p_if . . . . .	795
10.277.3. 2p_if . . . . .	795
10.277. Member Function Documentation . . . . .	795
10.277.4. 1has . . . . .	795
10.277.4. 2is_valid . . . . .	796
10.277.4. 3memory_size . . . . .	796
10.277.4. 4overset . . . . .	796
10.277.4. 5pred . . . . .	796
10.277.4. 6predicate . . . . .	796
10.278. In::p_image< I > Class Template Reference . . . . .	796
10.278. Detailed Description . . . . .	798
10.278. Member Typedef Documentation . . . . .	798
10.278.2. l_bkd_piter . . . . .	798
10.278.2. 2element . . . . .	798
10.278.2. 3fwd_piter . . . . .	798
10.278.2. 4_element . . . . .	798
10.278.2. 5piter . . . . .	798
10.278.2. 6psite . . . . .	798
10.278.2. 7_element . . . . .	798
10.278.2. 8S . . . . .	798
10.278. Constructor & Destructor Documentation . . . . .	798
10.278.3. 1p_image . . . . .	798
10.278.3. 2p_image . . . . .	799
10.278. Member Function Documentation . . . . .	799
10.278.4. 1clear . . . . .	799
10.278.4. 2has . . . . .	799
10.278.4. 3insert . . . . .	799
10.278.4. 4is_valid . . . . .	799



10.278.4.5	memory_size	799
10.278.4.6	sites	799
10.278.4.7	operator typename internal::p_image_site_set< I >::ret	799
10.278.4.8	remove	800
10.278.4.9	toggle	800
10.279	mln::p_indexed_bkd_piter< S > Class Template Reference	800
10.279.1	Detailed Description	800
10.279.2	Constructor & Destructor Documentation	800
10.279.2.1	p_indexed_bkd_piter	800
10.279.2.2	p_indexed_bkd_piter	801
10.279.3	Member Function Documentation	801
10.279.3.1	index	801
10.279.3.2	next	801
10.280	mln::p_indexed_fwd_piter< S > Class Template Reference	801
10.280.1	Detailed Description	801
10.280.2	Constructor & Destructor Documentation	802
10.280.2.1	p_indexed_fwd_piter	802
10.280.2.2	p_indexed_fwd_piter	802
10.280.3	Member Function Documentation	802
10.280.3.1	index	802
10.281	mln::p_indexed_psite< S > Class Template Reference	802
10.281.1	Detailed Description	802
10.282	mln::p_key< K, P > Class Template Reference	802
10.282.1	Detailed Description	804
10.282.2	Member Typedef Documentation	804
10.282.2.1	bkd_piter	804
10.282.2.2	element	804
10.282.2.3	fwd_piter	804
10.282.2.4	element	805
10.282.2.5	piter	805
10.282.2.6	psite	805
10.282.2.7	r_element	805
10.282.3	Constructor & Destructor Documentation	805
10.282.3.1	p_key	805
10.282.4	Member Function Documentation	805
10.282.4.1	lchange_key	805

10.282.4.2	change_keys	805
10.282.4.3	clear	805
10.282.4.4	exists_key	805
10.282.4.5	has	806
10.282.4.6	has	806
10.282.4.7	insert	806
10.282.4.8	insert	806
10.282.4.9	is_valid	806
10.282.4.10	key	806
10.282.4.11	keys	806
10.282.4.12	memory_size	806
10.282.4.13	sites	807
10.282.4.14	operator()	807
10.282.4.15	move	807
10.282.4.16	move_key	807
10.283	mln::p_line2d Class Reference	807
10.283.1	Detailed Description	808
10.283.2	Member Typedef Documentation	809
10.283.2.1	l_bkd_piter	809
10.283.2.2	element	809
10.283.2.3	fwd_piter	809
10.283.2.4	piter	809
10.283.2.5	psite	809
10.283.2.6	q_box	809
10.283.3	Constructor & Destructor Documentation	809
10.283.3.1	l_p_line2d	809
10.283.3.2	p_line2d	809
10.283.4	Member Function Documentation	809
10.283.4.1	l_bbox	809
10.283.4.2	begin	810
10.283.4.3	end	810
10.283.4.4	has	810
10.283.4.5	has	810
10.283.4.6	is_valid	810
10.283.4.7	memory_size	810
10.283.4.8	sites	810

10.283.4.9operator[]	810
10.283.4.10d_vector	810
10.284ln::p_mutable_array_of< S > Class Template Reference	811
10.284.Detailed Description	812
10.284.Member Typedef Documentation	812
10.284.2.lbkd_piter	812
10.284.2.2element	812
10.284.2.3fwd_piter	812
10.284.2.4i_element	812
10.284.2.5piter	812
10.284.2.6psite	812
10.284.Constructor & Destructor Documentation	813
10.284.3.lp_mutable_array_of	813
10.284.Member Function Documentation	813
10.284.4.lclear	813
10.284.4.2has	813
10.284.4.3insert	813
10.284.4.4is_valid	813
10.284.4.5memory_size	813
10.284.4.6elements	813
10.284.4.7operator[]	813
10.284.4.8operator[]	814
10.284.4.9reserve	814
10.285ln::p_n_faces_bkd_piter< D, G > Class Template Reference	814
10.285.Detailed Description	814
10.285.Constructor & Destructor Documentation	814
10.285.2.lp_n_faces_bkd_piter	814
10.285.Member Function Documentation	814
10.285.3.ln	814
10.286ln::p_n_faces_fwd_piter< D, G > Class Template Reference	815
10.286.Detailed Description	815
10.286.Constructor & Destructor Documentation	815
10.286.2.lp_n_faces_fwd_piter	815
10.286.Member Function Documentation	815
10.286.3.ln	815
10.286.3.2next	816

10.287	ln::p_priority< P, Q > Class Template Reference	816
10.287.1	Detailed Description	818
10.287.2	Member Typedef Documentation	818
10.287.2.1	bkd_piter	818
10.287.2.2	element	818
10.287.2.3	fwd_piter	818
10.287.2.4	_element	818
10.287.2.5	piter	818
10.287.2.6	psite	818
10.287.3	Constructor & Destructor Documentation	818
10.287.3.1	p_priority	818
10.287.4	Member Function Documentation	819
10.287.4.1	clear	819
10.287.4.2	exists_priority	819
10.287.4.3	front	819
10.287.4.4	has	819
10.287.4.5	highest_priority	819
10.287.4.6	insert	819
10.287.4.7	insert	820
10.287.4.8	s_valid	820
10.287.4.9	lowest_priority	820
10.287.4.10	memory_size	820
10.287.4.11	sites	820
10.287.4.12	operator()	820
10.287.4.13	pop	820
10.287.4.14	pop_front	821
10.287.4.15	priorities	821
10.287.4.16	push	821
10.288	ln::p_queue< P > Class Template Reference	821
10.288.1	Detailed Description	822
10.288.2	Member Typedef Documentation	823
10.288.2.1	bkd_piter	823
10.288.2.2	element	823
10.288.2.3	fwd_piter	823
10.288.2.4	_element	823
10.288.2.5	piter	823

10.288.2.6psite . . . . .	823
10.288.3.Constructor & Destructor Documentation . . . . .	823
10.288.3.1p_queue . . . . .	823
10.288.4.Member Function Documentation . . . . .	823
10.288.4.1clear . . . . .	823
10.288.4.2front . . . . .	823
10.288.4.3has . . . . .	824
10.288.4.4has . . . . .	824
10.288.4.5insert . . . . .	824
10.288.4.6s_valid . . . . .	824
10.288.4.7memory_size . . . . .	824
10.288.4.8sites . . . . .	824
10.288.4.9operator[] . . . . .	824
10.288.4.10pop . . . . .	824
10.288.4.11pop_front . . . . .	825
10.288.4.12push . . . . .	825
10.288.4.13std_deque . . . . .	825
10.288.4.14In::p_queue_fast< P > Class Template Reference . . . . .	825
10.289.1.Detailed Description . . . . .	827
10.289.2.Member Typedef Documentation . . . . .	827
10.289.2.1bkd_piter . . . . .	827
10.289.2.2element . . . . .	827
10.289.2.3fwd_piter . . . . .	827
10.289.2.4_element . . . . .	827
10.289.2.5piter . . . . .	827
10.289.2.6psite . . . . .	827
10.289.3.Constructor & Destructor Documentation . . . . .	828
10.289.3.1p_queue_fast . . . . .	828
10.289.4.Member Function Documentation . . . . .	828
10.289.4.1clear . . . . .	828
10.289.4.2compute_has . . . . .	828
10.289.4.3empty . . . . .	828
10.289.4.4front . . . . .	828
10.289.4.5has . . . . .	828
10.289.4.6has . . . . .	828
10.289.4.7insert . . . . .	828

10.289.4.8s_valid . . . . .	829
10.289.4.9memory_size . . . . .	829
10.289.4.10sites . . . . .	829
10.289.4.11operator[] . . . . .	829
10.289.4.12op . . . . .	829
10.289.4.13op_front . . . . .	829
10.289.4.14urge . . . . .	829
10.289.4.15ush . . . . .	829
10.289.4.16serve . . . . .	830
10.289.4.17d_vector . . . . .	830
10.290.1.1n::p_run< P > Class Template Reference . . . . .	830
10.290.2.Detailed Description . . . . .	831
10.290.3.Member Typedef Documentation . . . . .	831
10.290.2.1bkd_piter . . . . .	831
10.290.2.2element . . . . .	832
10.290.2.3fwd_piter . . . . .	832
10.290.2.4piter . . . . .	832
10.290.2.5psite . . . . .	832
10.290.2.6q_box . . . . .	832
10.290.3.Constructor & Destructor Documentation . . . . .	832
10.290.3.1p_run . . . . .	832
10.290.3.2p_run . . . . .	832
10.290.3.3p_run . . . . .	832
10.290.4.Member Function Documentation . . . . .	832
10.290.4.1bbox . . . . .	832
10.290.4.2end . . . . .	833
10.290.4.3has . . . . .	833
10.290.4.4has . . . . .	833
10.290.4.5has_index . . . . .	833
10.290.4.6nit . . . . .	833
10.290.4.7s_valid . . . . .	833
10.290.4.8length . . . . .	833
10.290.4.9memory_size . . . . .	833
10.290.4.10sites . . . . .	834
10.290.4.11operator[] . . . . .	834
10.290.4.12art . . . . .	834

10.291. <code>std::p_set&lt; P &gt;</code> Class Template Reference . . . . .	834
10.291.1. Detailed Description . . . . .	835
10.291.2. Member Typedef Documentation . . . . .	836
10.291.2.1. <code>l_bkd_piter</code> . . . . .	836
10.291.2.2. <code>e_element</code> . . . . .	836
10.291.2.3. <code>fwd_piter</code> . . . . .	836
10.291.2.4. <code>i_element</code> . . . . .	836
10.291.2.5. <code>piter</code> . . . . .	836
10.291.2.6. <code>psite</code> . . . . .	836
10.291.2.7. <code>r_element</code> . . . . .	836
10.291.3. Constructor & Destructor Documentation . . . . .	836
10.291.3.1. <code>lp_set</code> . . . . .	836
10.291.4. Member Function Documentation . . . . .	836
10.291.4.1. <code>clear</code> . . . . .	836
10.291.4.2. <code>has</code> . . . . .	837
10.291.4.3. <code>has</code> . . . . .	837
10.291.4.4. <code>has</code> . . . . .	837
10.291.4.5. <code>insert</code> . . . . .	837
10.291.4.6. <code>is_valid</code> . . . . .	837
10.291.4.7. <code>memory_size</code> . . . . .	837
10.291.4.8. <code>nsites</code> . . . . .	837
10.291.4.9. <code>operator[]</code> . . . . .	837
10.291.4.10. <code>remove</code> . . . . .	837
10.291.4.11. <code>std_vector</code> . . . . .	838
10.291.4.12. <code>til_set</code> . . . . .	838
10.292. <code>std::p_set_of&lt; S &gt;</code> Class Template Reference . . . . .	838
10.292.1. Detailed Description . . . . .	839
10.292.2. Member Typedef Documentation . . . . .	839
10.292.2.1. <code>l_bkd_piter</code> . . . . .	839
10.292.2.2. <code>e_element</code> . . . . .	839
10.292.2.3. <code>fwd_piter</code> . . . . .	839
10.292.2.4. <code>i_element</code> . . . . .	839
10.292.2.5. <code>piter</code> . . . . .	839
10.292.2.6. <code>psite</code> . . . . .	840
10.292.3. Constructor & Destructor Documentation . . . . .	840
10.292.3.1. <code>lp_set_of</code> . . . . .	840

10.292.4	Member Function Documentation	840
10.292.4.1	clear	840
10.292.4.2	has	840
10.292.4.3	insert	840
10.292.4.4	is_valid	840
10.292.4.5	memory_size	840
10.292.4.6	elements	840
10.292.4.7	operator[]	840
10.293	mpl::p_transformed< S, F > Class Template Reference	841
10.293.1	Detailed Description	842
10.293.2	Member Typedef Documentation	842
10.293.2.1	bkd_piter	842
10.293.2.2	element	842
10.293.2.3	fwd_piter	842
10.293.2.4	piter	842
10.293.2.5	psite	842
10.293.3	Constructor & Destructor Documentation	842
10.293.3.1	lp_transformed	842
10.293.3.2	pp_transformed	842
10.293.4	Member Function Documentation	843
10.293.4.1	function	843
10.293.4.2	has	843
10.293.4.3	is_valid	843
10.293.4.4	memory_size	843
10.293.4.5	primary_set	843
10.294	mpl::p_transformed_piter< Pi, S, F > Struct Template Reference	843
10.294.1	Detailed Description	844
10.294.2	Constructor & Destructor Documentation	844
10.294.2.1	lp_transformed_piter	844
10.294.2.2	pp_transformed_piter	844
10.294.3	Member Function Documentation	844
10.294.3.1	lchange_target	844
10.294.3.2	next	844
10.295	mpl::p_vaccess< V, S > Class Template Reference	845
10.295.1	Detailed Description	846
10.295.2	Member Typedef Documentation	846



---

10.295.2.1bkd_piter . . . . .	846
10.295.2.2element . . . . .	846
10.295.2.3fwd_piter . . . . .	846
10.295.2.4i_element . . . . .	846
10.295.2.5piter . . . . .	846
10.295.2.6pset . . . . .	847
10.295.2.7psite . . . . .	847
10.295.2.8value . . . . .	847
10.295.2.9vset . . . . .	847
10.295.3Constructor & Destructor Documentation . . . . .	847
10.295.3.1p_vaccess . . . . .	847
10.295.4Member Function Documentation . . . . .	847
10.295.4.1has . . . . .	847
10.295.4.2has . . . . .	847
10.295.4.3insert . . . . .	847
10.295.4.4insert . . . . .	847
10.295.4.5is_valid . . . . .	848
10.295.4.6memory_size . . . . .	848
10.295.4.7operator() . . . . .	848
10.295.4.8values . . . . .	848
10.296In::p_vertices< G, F > Class Template Reference . . . . .	848
10.296.1Detailed Description . . . . .	850
10.296.2Member Typedef Documentation . . . . .	850
10.296.2.1bkd_piter . . . . .	850
10.296.2.2element . . . . .	850
10.296.2.3fun_t . . . . .	850
10.296.2.4fwd_piter . . . . .	850
10.296.2.5graph_element . . . . .	851
10.296.2.6graph_t . . . . .	851
10.296.2.7piter . . . . .	851
10.296.2.8psite . . . . .	851
10.296.2.9vertex . . . . .	851
10.296.3Constructor & Destructor Documentation . . . . .	851
10.296.3.1p_vertices . . . . .	851
10.296.3.2p_vertices . . . . .	851
10.296.3.3p_vertices . . . . .	851

10.296.3.4p_vertices	852
10.296.3.5p_vertices	852
10.296.4Member Function Documentation	852
10.296.4.1function	852
10.296.4.2graph	852
10.296.4.3has	852
10.296.4.4has	853
10.296.4.5invalidate	853
10.296.4.6is_valid	853
10.296.4.7memory_size	853
10.296.4.8sites	853
10.296.4.9vertices	853
10.296.4.10operator()	853
10.297In::pixel< I > Struct Template Reference	853
10.297.1Detailed Description	854
10.297.2Constructor & Destructor Documentation	854
10.297.2.1pixel	854
10.297.2.2pixel	855
10.297.3Member Function Documentation	855
10.297.3.1change_to	855
10.297.3.2is_valid	855
10.298In::Pixel_Iterator< E > Struct Template Reference	855
10.298.1Detailed Description	856
10.298.2Member Function Documentation	856
10.298.2.1next	856
10.299In::plain< I > Class Template Reference	856
10.299.1Detailed Description	857
10.299.2Member Typedef Documentation	857
10.299.2.1skeleton	857
10.299.3Constructor & Destructor Documentation	857
10.299.3.1plain	857
10.299.3.2plain	857
10.299.3.3plain	857
10.299.4Member Function Documentation	857
10.299.4.1operator I	857
10.299.4.2operator=	858

10.299.4.3operator=	858
10.300ln::Point< P > Struct Template Reference	858
10.300.1Detailed Description	859
10.300.2Member Typedef Documentation	859
10.300.2.1point	859
10.300.3Member Function Documentation	859
10.300.3.1to_point	859
10.300.4Friends And Related Function Documentation	859
10.300.4.1operator+=	859
10.300.4.2operator-=	860
10.300.4.3operator/	860
10.301ln::point< G, C > Struct Template Reference	860
10.301.1Detailed Description	863
10.301.2Member Typedef Documentation	863
10.301.2.1lcoord	863
10.301.2.2delta	863
10.301.2.3lpsite	863
10.301.2.4grid	863
10.301.2.5h_vec	863
10.301.2.6vec	864
10.301.3Member Enumeration Documentation	864
10.301.3.1"@30	864
10.301.4Constructor & Destructor Documentation	864
10.301.4.1point	864
10.301.4.2point	864
10.301.4.3point	864
10.301.4.4point	864
10.301.4.5point	864
10.301.5Member Function Documentation	864
10.301.5.1last_coord	864
10.301.5.2last_coord	865
10.301.5.3minus_infty	865
10.301.5.4operator+=	865
10.301.5.5operator-=	865
10.301.5.6operator[]	865
10.301.5.7operator[]	865

10.301.5.8plus_infty . . . . .	865
10.301.5.9set_all . . . . .	866
10.301.5.10h_vec . . . . .	866
10.301.5.11b_vec . . . . .	866
10.301.6Member Data Documentation . . . . .	866
10.301.6.1origin . . . . .	866
10.302mln::Point_Site< E > Struct Template Reference . . . . .	866
10.302.1Detailed Description . . . . .	868
10.302.2Friends And Related Function Documentation . . . . .	868
10.302.2.1operator+ . . . . .	868
10.302.2.2operator- . . . . .	868
10.302.2.3operator- . . . . .	869
10.302.2.4operator<< . . . . .	869
10.302.2.5operator== . . . . .	870
10.303mln::Point_Site< void > Struct Template Reference . . . . .	870
10.303.1Detailed Description . . . . .	870
10.304mln::Proxy< E > Struct Template Reference . . . . .	870
10.304.1Detailed Description . . . . .	870
10.305mln::Proxy< void > Struct Template Reference . . . . .	870
10.305.1Detailed Description . . . . .	871
10.306mln::Pseudo_Site< E > Struct Template Reference . . . . .	871
10.306.1Detailed Description . . . . .	871
10.307mln::Pseudo_Site< void > Struct Template Reference . . . . .	871
10.307.1Detailed Description . . . . .	871
10.308mln::pw::image< F, S > Class Template Reference . . . . .	871
10.308.1Detailed Description . . . . .	872
10.308.2Member Typedef Documentation . . . . .	872
10.308.2.1skeleton . . . . .	872
10.308.3Constructor & Destructor Documentation . . . . .	872
10.308.3.1image . . . . .	872
10.308.3.2image . . . . .	872
10.309mln::registration::closest_point_basic< P > Class Template Reference . . . . .	873
10.309.1Detailed Description . . . . .	873
10.310mln::registration::closest_point_with_map< P > Class Template Reference . . . . .	873
10.310.1Detailed Description . . . . .	873
10.311mln::Regular_Grid< E > Struct Template Reference . . . . .	873

10.311. Detailed Description	874
10.311. <code>mln::safe_image&lt; I &gt;</code> Class Template Reference	874
10.312. Detailed Description	874
10.312. Member Typedef Documentation	874
10.312.2. <code>Iskeleton</code>	874
10.312. Member Function Documentation	874
10.312.3. <code>operator safe_image&lt; const I &gt;</code>	874
10.313. <code>mln::select::p_of&lt; P &gt;</code> Struct Template Reference	875
10.313. Detailed Description	875
10.314. <code>mln::Site&lt; E &gt;</code> Struct Template Reference	875
10.314. Detailed Description	875
10.315. <code>mln::Site&lt; void &gt;</code> Struct Template Reference	876
10.315. Detailed Description	876
10.316. <code>mln::Site_Iterator&lt; E &gt;</code> Struct Template Reference	876
10.316. Detailed Description	878
10.316. Member Function Documentation	878
10.316.2. <code>Inext</code>	878
10.317. <code>mln::Site_Proxy&lt; E &gt;</code> Struct Template Reference	878
10.317. Detailed Description	878
10.318. <code>mln::Site_Proxy&lt; void &gt;</code> Struct Template Reference	878
10.318. Detailed Description	879
10.319. <code>mln::Site_Set&lt; E &gt;</code> Struct Template Reference	879
10.319. Detailed Description	880
10.319. Friends And Related Function Documentation	881
10.319.2. <code>ldiff</code>	881
10.319.2. <code>2inter</code>	881
10.319.2.3. <code>operator&lt;</code>	881
10.319.2.4. <code>operator&lt;&lt;</code>	881
10.319.2.5. <code>operator&lt;=</code>	881
10.319.2.6. <code>operator==</code>	882
10.319.2.7. <code>sym_diff</code>	882
10.319.2.8. <code>uni</code>	882
10.319.2.9. <code>unique</code>	882
10.320. <code>mln::Site_Set&lt; void &gt;</code> Struct Template Reference	882
10.320. Detailed Description	882
10.321. <code>mln::slice_image&lt; I &gt;</code> Struct Template Reference	882

10.321.1	Detailed Description	883
10.321.2	Member Typedef Documentation	883
10.321.2.1	Iskeleton	883
10.321.3	Constructor & Destructor Documentation	883
10.321.3.1	Islice_image	883
10.321.3.2	2slice_image	884
10.321.4	Member Function Documentation	884
10.321.4.1	Idomain	884
10.321.4.2	operator slice_image< const I >	884
10.321.4.3	operator()	884
10.321.4.4	operator()	884
10.321.4.5	5li	884
10.321	mln::sub_image< I, S > Struct Template Reference	884
10.322.1	Detailed Description	885
10.322.2	Member Typedef Documentation	885
10.322.2.1	Iskeleton	885
10.322.3	Constructor & Destructor Documentation	885
10.322.3.1	Isub_image	885
10.322.3.2	2sub_image	885
10.322.4	Member Function Documentation	885
10.322.4.1	Idomain	885
10.322.4.2	operator sub_image< const I, S >	886
10.322	mln::sub_image_if< I, S > Struct Template Reference	886
10.323.1	Detailed Description	886
10.323.2	Member Typedef Documentation	886
10.323.2.1	Iskeleton	886
10.323.3	Constructor & Destructor Documentation	887
10.323.3.1	Isub_image_if	887
10.323.3.2	2sub_image_if	887
10.323.4	Member Function Documentation	887
10.323.4.1	Idomain	887
10.324	mln::thru_image< I, F > Class Template Reference	887
10.324.1	Detailed Description	887
10.324.2	Member Function Documentation	887
10.324.2.1	operator thru_image< const I, F >	887
10.325	mln::thru_bin_image< I1, I2, F > Class Template Reference	887

10.325. Detailed Description	888
10.325. Member Typedef Documentation	888
10.325.2. lpsite	888
10.325.2. rvalue	888
10.325.2. skeleton	888
10.325.2. value	889
10.325. Member Function Documentation	889
10.325.3. loperator thrubin_image< const I1, const I2, F >	889
10.326. mln::topo::adj_higher_dim_connected_n_face_bkd_iter< D > Class Template Reference	889
10.326. Detailed Description	889
10.326. Constructor & Destructor Documentation	890
10.326.2. ladj_higher_dim_connected_n_face_bkd_iter	890
10.326. Member Function Documentation	890
10.326.3. lnext	890
10.327. mln::topo::adj_higher_dim_connected_n_face_fwd_iter< D > Class Template Reference	890
10.327. Detailed Description	890
10.327. Constructor & Destructor Documentation	891
10.327.2. ladj_higher_dim_connected_n_face_fwd_iter	891
10.327. Member Function Documentation	891
10.327.3. lnext	891
10.328. mln::topo::adj_higher_face_bkd_iter< D > Class Template Reference	891
10.328. Detailed Description	891
10.328. Constructor & Destructor Documentation	892
10.328.2. ladj_higher_face_bkd_iter	892
10.328. Member Function Documentation	892
10.328.3. lnext	892
10.329. mln::topo::adj_higher_face_fwd_iter< D > Class Template Reference	892
10.329. Detailed Description	892
10.329. Constructor & Destructor Documentation	893
10.329.2. ladj_higher_face_fwd_iter	893
10.329. Member Function Documentation	893
10.329.3. lnext	893
10.330. mln::topo::adj_lower_dim_connected_n_face_bkd_iter< D > Class Template Reference	893
10.330. Detailed Description	894
10.330. Constructor & Destructor Documentation	894
10.330.2. ladj_lower_dim_connected_n_face_bkd_iter	894

10.330.	Member Function Documentation	894
10.330.3.	lnext	894
10.331.	nl::topo::adj_lower_dim_connected_n_face_fwd_iter< D > Class Template Reference	894
10.331.	Detailed Description	895
10.331.	Constructor & Destructor Documentation	895
10.331.2.	ladj_lower_dim_connected_n_face_fwd_iter	895
10.331.	Member Function Documentation	895
10.331.3.	lnext	895
10.332.	nl::topo::adj_lower_face_bkd_iter< D > Class Template Reference	895
10.332.	Detailed Description	896
10.332.	Constructor & Destructor Documentation	896
10.332.2.	ladj_lower_face_bkd_iter	896
10.332.	Member Function Documentation	896
10.332.3.	lnext	896
10.333.	nl::topo::adj_lower_face_fwd_iter< D > Class Template Reference	896
10.333.	Detailed Description	897
10.333.	Constructor & Destructor Documentation	897
10.333.2.	ladj_lower_face_fwd_iter	897
10.333.	Member Function Documentation	897
10.333.3.	lnext	897
10.334.	nl::topo::adj_lower_higher_face_bkd_iter< D > Class Template Reference	897
10.334.	Detailed Description	898
10.334.	Constructor & Destructor Documentation	898
10.334.2.	ladj_lower_higher_face_bkd_iter	898
10.334.	Member Function Documentation	898
10.334.3.	lnext	898
10.335.	nl::topo::adj_lower_higher_face_fwd_iter< D > Class Template Reference	898
10.335.	Detailed Description	899
10.335.	Constructor & Destructor Documentation	899
10.335.2.	ladj_lower_higher_face_fwd_iter	899
10.335.	Member Function Documentation	899
10.335.3.	lnext	899
10.336.	nl::topo::adj_m_face_bkd_iter< D > Class Template Reference	899
10.336.	Detailed Description	900
10.336.	Constructor & Destructor Documentation	900
10.336.2.	ladj_m_face_bkd_iter	900



10.336.2.2adj_m_face_bkd_iter . . . . .	900
10.336.3Member Function Documentation . . . . .	900
10.336.3.1next . . . . .	900
10.337In::topo::adj_m_face_fwd_iter< D > Class Template Reference . . . . .	901
10.337.1Detailed Description . . . . .	901
10.337.2Constructor & Destructor Documentation . . . . .	901
10.337.2.1adj_m_face_fwd_iter . . . . .	901
10.337.2.2adj_m_face_fwd_iter . . . . .	902
10.337.3Member Function Documentation . . . . .	902
10.337.3.1next . . . . .	902
10.338In::topo::algebraic_face< D > Struct Template Reference . . . . .	902
10.338.1Detailed Description . . . . .	904
10.338.2Constructor & Destructor Documentation . . . . .	904
10.338.2.1algebraic_face . . . . .	904
10.338.2.2algebraic_face . . . . .	904
10.338.2.3algebraic_face . . . . .	904
10.338.2.4algebraic_face . . . . .	904
10.338.3Member Function Documentation . . . . .	904
10.338.3.1cplx . . . . .	904
10.338.3.2data . . . . .	905
10.338.3.3dec_face_id . . . . .	905
10.338.3.4dec_n . . . . .	905
10.338.3.5face_id . . . . .	905
10.338.3.6higher_dim_adj_faces . . . . .	905
10.338.3.7inc_face_id . . . . .	905
10.338.3.8inc_n . . . . .	905
10.338.3.9invalidate . . . . .	905
10.338.3.10valid . . . . .	905
10.338.3.11bwer_dim_adj_faces . . . . .	906
10.338.3.12 . . . . .	906
10.338.3.13set_cplx . . . . .	906
10.338.3.14set_face_id . . . . .	906
10.338.3.15set_n . . . . .	906
10.338.3.16set_sign . . . . .	906
10.338.3.17gn . . . . .	906
10.339In::topo::algebraic_n_face< N, D > Class Template Reference . . . . .	907

10.339. Detailed Description	908
10.339. Constructor & Destructor Documentation	908
10.339.2. 1algebraic_n_face	908
10.339.2. 2algebraic_n_face	908
10.339.2. 3algebraic_n_face	909
10.339. Member Function Documentation	909
10.339.3. 1cplx	909
10.339.3. 2data	909
10.339.3. 3dec_face_id	909
10.339.3. 4face_id	909
10.339.3. 5higher_dim_adj_faces	909
10.339.3. 6nc_face_id	909
10.339.3. 7invalidate	910
10.339.3. 8s_valid	910
10.339.3. 9lower_dim_adj_faces	910
10.339.3. 10	910
10.339.3. 11set_cplx	910
10.339.3. 12set_face_id	910
10.339.3. 13set_sign	910
10.339.3. 14gn	910
10.340. In::topo::center_only_iter< D > Class Template Reference	911
10.340. Detailed Description	911
10.340. Constructor & Destructor Documentation	911
10.340.2. 1center_only_iter	911
10.340. Member Function Documentation	912
10.340.3. 1next	912
10.341. In::topo::centered_bkd_iter_adapter< D, I > Class Template Reference	912
10.341. Detailed Description	912
10.341. Constructor & Destructor Documentation	912
10.341.2. 1centered_bkd_iter_adapter	912
10.341. Member Function Documentation	913
10.341.3. 1next	913
10.342. In::topo::centered_fwd_iter_adapter< D, I > Class Template Reference	913
10.342. Detailed Description	913
10.342. Constructor & Destructor Documentation	913
10.342.2. 1centered_fwd_iter_adapter	913

10.342.3	Member Function Documentation	914
10.342.3.1	next	914
10.343	nl::topo::complex< D > Class Template Reference	914
10.343.1	Detailed Description	915
10.343.2	Member Typedef Documentation	915
10.343.2.1	l_bkd_citer	915
10.343.2.2	l_fwd_citer	915
10.343.3	Constructor & Destructor Documentation	915
10.343.3.1	l_complex	915
10.343.4	Member Function Documentation	915
10.343.4.1	l_add_face	915
10.343.4.2	l_add_face	916
10.343.4.3	l_addr	916
10.343.4.4	l_faces	916
10.343.4.5	l_faces_of_dim	916
10.343.4.6	l_faces_of_static_dim	916
10.343.4.7	l_print	916
10.343.4.8	l_print_faces	917
10.344	nl::topo::face< D > Struct Template Reference	917
10.344.1	Detailed Description	918
10.344.2	Constructor & Destructor Documentation	918
10.344.2.1	l_face	918
10.344.2.2	l_face	918
10.344.2.3	l_face	919
10.344.3	Member Function Documentation	919
10.344.3.1	l_cplx	919
10.344.3.2	l_data	919
10.344.3.3	l_dec_face_id	919
10.344.3.4	l_dec_n	919
10.344.3.5	l_face_id	919
10.344.3.6	l_higher_dim_adj_faces	919
10.344.3.7	l_inc_face_id	919
10.344.3.8	l_inc_n	920
10.344.3.9	l_invalid	920
10.344.3.10	l_is_valid	920
10.344.3.11	l_lower_dim_adj_faces	920

10.344.3.1 <del>2</del>	920
10.344.3.1 <del>3</del> set_cplx	920
10.344.3.1 <del>4</del> set_face_id	920
10.344.3.1 <del>5</del> set_n	920
10.345.1mnl::topo::face_bkd_iter< D > Class Template Reference	920
10.345.1.Detailed Description	921
10.345.1.Constructor & Destructor Documentation	921
10.345.2.1face_bkd_iter	921
10.345.1.Member Function Documentation	921
10.345.3.1next	921
10.345.3.2start	922
10.346.1mnl::topo::face_fwd_iter< D > Class Template Reference	922
10.346.1.Detailed Description	922
10.346.1.Constructor & Destructor Documentation	922
10.346.2.1face_fwd_iter	922
10.346.1.Member Function Documentation	922
10.346.3.1next	922
10.346.3.2start	923
10.347.1mnl::topo::is_n_face< N > Struct Template Reference	923
10.347.1.Detailed Description	924
10.348.1mnl::topo::is_simple_cell< I > Class Template Reference	924
10.348.1.Detailed Description	926
10.348.1.Member Typedef Documentation	926
10.348.2.1psite	926
10.348.2.2result	926
10.348.1.Member Function Documentation	926
10.348.3.1mnl_geom	926
10.348.3.2operator()	926
10.348.3.3set_image	927
10.348.1.Member Data Documentation	927
10.348.4.1ID	927
10.349.1mnl::topo::n_face< N, D > Class Template Reference	927
10.349.1.Detailed Description	928
10.349.1.Constructor & Destructor Documentation	928
10.349.2.1n_face	928
10.349.2.2n_face	928

10.349.3	Member Function Documentation	929
10.349.3.1	lplx	929
10.349.3.2	data	929
10.349.3.3	dec_face_id	929
10.349.3.4	face_id	929
10.349.3.5	higher_dim_adj_faces	929
10.349.3.6	inc_face_id	929
10.349.3.7	invalidate	929
10.349.3.8	is_valid	930
10.349.3.9	lower_dim_adj_faces	930
10.349.3.10		930
10.349.3.11	set_cplx	930
10.349.3.12	set_face_id	930
10.350	mln::topo::n_face_bkd_iter< D > Class Template Reference	930
10.350.1	Detailed Description	931
10.350.2	Constructor & Destructor Documentation	931
10.350.2.1	ln_face_bkd_iter	931
10.350.3	Member Function Documentation	931
10.350.3.1	ln	931
10.350.3.2	next	931
10.350.3.3	start	932
10.351	mln::topo::n_face_fwd_iter< D > Class Template Reference	932
10.351.1	Detailed Description	932
10.351.2	Constructor & Destructor Documentation	932
10.351.2.1	ln_face_fwd_iter	932
10.351.3	Member Function Documentation	933
10.351.3.1	ln	933
10.351.3.2	next	933
10.351.3.3	start	933
10.352	mln::topo::n_faces_set< N, D > Class Template Reference	933
10.352.1	Detailed Description	934
10.352.2	Member Typedef Documentation	934
10.352.2.1	ifaces_type	934
10.352.3	Member Function Documentation	934
10.352.3.1	ladd	934
10.352.3.2	faces	934

10.352.3.reserve	934
10.352. Inl::topo::static_n_face_bkd_iter< N, D > Class Template Reference	934
10.353. Detailed Description	935
10.353. Constructor & Destructor Documentation	935
10.353.2. lstatic_n_face_bkd_iter	935
10.353. Member Function Documentation	935
10.353.3. lnext	935
10.353.3. 2start	935
10.354. Inl::topo::static_n_face_fwd_iter< N, D > Class Template Reference	936
10.354. Detailed Description	936
10.354. Constructor & Destructor Documentation	936
10.354.2. lstatic_n_face_fwd_iter	936
10.354. Member Function Documentation	936
10.354.3. lnext	936
10.354.3. 2start	937
10.355. Inl::tr_image< S, I, T > Struct Template Reference	937
10.355. Detailed Description	938
10.355. Member Typedef Documentation	938
10.355.2. llvalue	938
10.355.2. 2psite	938
10.355.2. 3rvalue	938
10.355.2. 4site	938
10.355.2. 5skeleton	938
10.355.2. 6value	939
10.355. Constructor & Destructor Documentation	939
10.355.3. ltr_image	939
10.355. Member Function Documentation	939
10.355.4. ldomain	939
10.355.4. 2has	939
10.355.4. 3is_valid	939
10.355.4. 4operator()	939
10.355.4. 5set_tr	939
10.355.4. 6tr	939
10.356. Inl::transformed_image< I, F > Struct Template Reference	940
10.356. Detailed Description	940
10.356. Member Typedef Documentation	940

---

10.356.2. Iskeleton	940
10.356.3. Constructor & Destructor Documentation	941
10.356.3.1 transformed_image	941
10.356.3.2 transformed_image	941
10.356.4. Member Function Documentation	941
10.356.4.1 domain	941
10.356.4.2 operator transformed_image< const I, F >	941
10.356.4.3 operator()	941
10.356.4.4 operator()	941
10.357. In::unproject_image< I, D, F > Struct Template Reference	941
10.357.1. Detailed Description	942
10.357.2. Constructor & Destructor Documentation	942
10.357.2.1 unproject_image	942
10.357.2.2 unproject_image	942
10.357.3. Member Function Documentation	942
10.357.3.1 domain	942
10.357.3.2 operator()	942
10.357.3.3 operator()	942
10.358. In::util::adjacency_matrix< V > Class Template Reference	943
10.358.1. Detailed Description	943
10.358.2. Constructor & Destructor Documentation	943
10.358.2.1 adjacency_matrix	943
10.358.2.2 adjacency_matrix	943
10.359. In::util::array< T > Class Template Reference	943
10.359.1. Detailed Description	946
10.359.2. Member Typedef Documentation	946
10.359.2.1 bkd_eiter	946
10.359.2.2 eiter	946
10.359.2.3 element	946
10.359.2.4 fwd_eiter	946
10.359.2.5 result	946
10.359.3. Constructor & Destructor Documentation	946
10.359.3.1 larray	946
10.359.3.2 array	946
10.359.3.3 array	947
10.359.4. Member Function Documentation	947

10.359.4.1	append	947
10.359.4.2	append	947
10.359.4.3	clear	947
10.359.4.4	fill	947
10.359.4.5	is_empty	947
10.359.4.6	memory_size	948
10.359.4.7	elements	948
10.359.4.8	operator()	948
10.359.4.9	operator()	948
10.359.4.10	operator[]	948
10.359.4.11	operator[]	948
10.359.4.12	reserve	949
10.359.4.13	size	949
10.359.4.14	size	949
10.359.4.15	size	949
10.359.4.16	id_vector	949
10.360	util::branch< T > Class Template Reference	949
10.360.1	Detailed Description	950
10.360.2	Constructor & Destructor Documentation	950
10.360.2.1	branch	950
10.360.3	Member Function Documentation	950
10.360.3.1	apex	950
10.360.3.2	util_tree	950
10.361	util::branch_iter< T > Class Template Reference	951
10.361.1	Detailed Description	951
10.361.2	Member Function Documentation	951
10.361.2.1	deepness	951
10.361.2.2	invalidate	951
10.361.2.3	is_valid	952
10.361.2.4	next	952
10.361.2.5	operator util::tree_node< T > &	952
10.361.2.6	start	952
10.362	util::branch_iter_ind< T > Class Template Reference	952
10.362.1	Detailed Description	953
10.362.2	Member Function Documentation	953
10.362.2.1	deepness	953



10.362.2.2	<code>invalidate</code>	953
10.362.2.3	<code>is_valid</code>	953
10.362.2.4	<code>next</code>	953
10.362.2.5	<code>operator util::tree_node&lt; T &gt; &amp;</code>	953
10.362.2.6	<code>start</code>	953
10.363	<code>util::couple&lt; T, U &gt;</code> Class Template Reference	953
10.363.1	Detailed Description	954
10.363.2	Member Function Documentation	955
10.363.2.1	<code>change_both</code>	955
10.363.2.2	<code>change_first</code>	955
10.363.2.3	<code>change_second</code>	955
10.363.2.4	<code>first</code>	955
10.363.2.5	<code>second</code>	955
10.364	<code>util::eat</code> Struct Reference	955
10.364.1	Detailed Description	956
10.365	<code>util::edge&lt; G &gt;</code> Class Template Reference	956
10.365.1	Detailed Description	957
10.365.2	Member Typedef Documentation	958
10.365.2.1	<code>category</code>	958
10.365.2.2	<code>graph_t</code>	958
10.365.2.3	<code>id_t</code>	958
10.365.2.4	<code>id_value_t</code>	958
10.365.3	Constructor & Destructor Documentation	958
10.365.3.1	<code>edge</code>	958
10.365.4	Member Function Documentation	958
10.365.4.1	<code>change_graph</code>	958
10.365.4.2	<code>graph</code>	958
10.365.4.3	<code>id</code>	958
10.365.4.4	<code>invalidate</code>	958
10.365.4.5	<code>is_valid</code>	959
10.365.4.6	<code>th_nbh_edge</code>	959
10.365.4.7	<code>nmax_nbh_edges</code>	959
10.365.4.8	<code>operator edge_id_t</code>	959
10.365.4.9	<code>update_id</code>	959
10.365.4.10	<code>1</code>	959
10.365.4.11	<code>2</code>	959

10.365.4.12	other	959
10.366	ln::util::fibonacci_heap< P, T > Class Template Reference	960
10.366	Detailed Description	961
10.366	Constructor & Destructor Documentation	961
10.366.2	1.fibonacci_heap	961
10.366.2	2.fibonacci_heap	961
10.366	3.Member Function Documentation	961
10.366.3	1.clear	961
10.366.3	2.front	961
10.366.3	3.is_empty	962
10.366.3	4.is_valid	962
10.366.3	5.elements	962
10.366.3	6.operator=	962
10.366.3	7.pop_front	962
10.366.3	8.push	962
10.366.3	9.push	963
10.367	ln::util::graph Class Reference	963
10.367	Detailed Description	965
10.367	2.Member Typedef Documentation	965
10.367.2	1.edge_fwd_iter	965
10.367.2	2.edge_nbh_edge_fwd_iter	965
10.367.2	3.edges_set_t	965
10.367.2	4.edges_t	965
10.367.2	5.vertex_fwd_iter	965
10.367.2	6.vertex_nbh_edge_fwd_iter	966
10.367.2	7.vertex_nbh_vertex_fwd_iter	966
10.367.2	8.vertices_t	966
10.367	3.Constructor & Destructor Documentation	966
10.367.3	1.graph	966
10.367.3	2.graph	966
10.367	4.Member Function Documentation	966
10.367.4	1.add_edge	966
10.367.4	2.add_vertex	966
10.367.4	3.add_vertices	967
10.367.4	4.e_ith_nbh_edge	967
10.367.4	5.e_nmax	967

10.367.4.6	<code>nmax_nbh_edges</code>	967
10.367.4.7	<code>edge</code>	967
10.367.4.8	<code>edge</code>	967
10.367.4.9	<code>edges</code>	968
10.367.4.10	<code>has_e</code>	968
10.367.4.11	<code>has_v</code>	968
10.367.4.12	<code>is_subgraph_of</code>	968
10.367.4.13	<code>is</code>	968
10.367.4.14	<code>is</code>	968
10.367.4.15	<code>ith_nbh_edge</code>	968
10.367.4.16	<code>ith_nbh_vertex</code>	968
10.367.4.17	<code>nmax</code>	969
10.367.4.18	<code>nmax_nbh_edges</code>	969
10.367.4.19	<code>nmax_nbh_vertices</code>	969
10.367.4.20	<code>vertex</code>	969
10.368	<code>util::greater_point&lt; I &gt; Class Template Reference</code>	969
10.368.1	Detailed Description	969
10.368.2	Member Function Documentation	970
10.368.2.1	<code>operator()</code>	970
10.369	<code>util::greater_site&lt; I &gt; Class Template Reference</code>	970
10.369.1	Detailed Description	970
10.369.2	Member Function Documentation	970
10.369.2.1	<code>operator()</code>	970
10.370	<code>util::head&lt; T, R &gt; Class Template Reference</code>	970
10.370.1	Detailed Description	970
10.371	<code>util::ignore Struct Reference</code>	971
10.371.1	Detailed Description	971
10.372	<code>util::icell&lt; T &gt; Struct Template Reference</code>	971
10.372.1	Detailed Description	971
10.373	<code>util::line_graph&lt; G &gt; Class Template Reference</code>	971
10.373.1	Detailed Description	973
10.373.2	Member Typedef Documentation	974
10.373.2.1	<code>edge_fwd_iter</code>	974
10.373.2.2	<code>edge_nbh_edge_fwd_iter</code>	974
10.373.2.3	<code>edges_t</code>	974
10.373.2.4	<code>vertex_fwd_iter</code>	974

10.373.2.5	vertex_nbh_edge_fwd_iter	974
10.373.2.6	vertex_nbh_vertex_fwd_iter	974
10.373.2.7	vertices_t	974
10.373.3	Member Function Documentation	974
10.373.3.1e	ith_nbh_edge	974
10.373.3.2e	nmax	975
10.373.3.3e	nmax_nbh_edges	975
10.373.3.4e	edge	975
10.373.3.5e	graph	975
10.373.3.6e	has	975
10.373.3.7e	has	975
10.373.3.8e	has_e	975
10.373.3.9e	has_v	976
10.373.3.10e	is_subgraph_of	976
10.373.3.11e	is	976
10.373.3.12e	is	976
10.373.3.13e	ith_nbh_edge	976
10.373.3.14e	ith_nbh_vertex	976
10.373.3.15e	nmax	977
10.373.3.16e	nmax_nbh_edges	977
10.373.3.17e	nmax_nbh_vertices	977
10.373.3.18e	vertex	977
10.374	mln::util::nil Struct Reference	977
10.374.1	Detailed Description	978
10.375	mln::util::node< T, R > Class Template Reference	978
10.375.1	Detailed Description	978
10.376	mln::util::object_id< Tag, V > Class Template Reference	978
10.376.1	Detailed Description	979
10.376.2	Member Typedef Documentation	980
10.376.2.1	lvalue_t	980
10.376.3	Constructor & Destructor Documentation	980
10.376.3.1	lobject_id	980
10.377	mln::util::ord< T > Struct Template Reference	980
10.377.1	Detailed Description	980
10.378	mln::util::ord_pair< T > Struct Template Reference	980
10.378.1	Detailed Description	981

10.378.2Member Function Documentation . . . . .	982
10.378.2.1change_both . . . . .	982
10.378.2.2change_first . . . . .	982
10.378.2.3change_second . . . . .	982
10.378.2.4first . . . . .	982
10.378.2.5second . . . . .	982
10.379Inn::util::pix< I > Struct Template Reference . . . . .	982
10.379.Detailed Description . . . . .	983
10.379.2Member Typedef Documentation . . . . .	983
10.379.2.1psite . . . . .	983
10.379.2.2value . . . . .	983
10.379.3Constructor & Destructor Documentation . . . . .	983
10.379.3.1pix . . . . .	983
10.379.4Member Function Documentation . . . . .	984
10.379.4.1ima . . . . .	984
10.379.4.2p . . . . .	984
10.379.4.3v . . . . .	984
10.380Inn::util::set< T > Class Template Reference . . . . .	984
10.380.Detailed Description . . . . .	986
10.380.2Member Typedef Documentation . . . . .	987
10.380.2.1bkd_eiter . . . . .	987
10.380.2.2eiter . . . . .	987
10.380.2.3element . . . . .	987
10.380.2.4fwd_eiter . . . . .	987
10.380.3Constructor & Destructor Documentation . . . . .	987
10.380.3.1set . . . . .	987
10.380.4Member Function Documentation . . . . .	987
10.380.4.1clear . . . . .	987
10.380.4.2first_element . . . . .	987
10.380.4.3has . . . . .	988
10.380.4.4insert . . . . .	988
10.380.4.5insert . . . . .	988
10.380.4.6is_empty . . . . .	988
10.380.4.7last_element . . . . .	989
10.380.4.8memory_size . . . . .	989
10.380.4.9elements . . . . .	989

---

10.380.4.10operator[]	989
10.380.4.11remove	989
10.380.4.12id_vector	990
10.381ln::util::site_pair< P > Class Template Reference	990
10.381.1Detailed Description	991
10.381.2Member Function Documentation	991
10.381.2.1first	991
10.381.2.2pair	991
10.381.2.3second	991
10.382ln::util::soft_heap< T, R > Class Template Reference	991
10.382.1Detailed Description	993
10.382.2Member Typedef Documentation	993
10.382.2.1element	993
10.382.3Constructor & Destructor Documentation	993
10.382.3.1soft_heap	993
10.382.3.2~soft_heap	993
10.382.4Member Function Documentation	993
10.382.4.1clear	993
10.382.4.2is_empty	993
10.382.4.3is_valid	994
10.382.4.4elements	994
10.382.4.5pop_front	994
10.382.4.6push	994
10.382.4.7push	994
10.383ln::util::timer Class Reference	994
10.383.1Detailed Description	995
10.384ln::util::tracked_ptr< T > Struct Template Reference	995
10.384.1Detailed Description	996
10.384.2Constructor & Destructor Documentation	996
10.384.2.1tracked_ptr	996
10.384.2.2tracked_ptr	996
10.384.2.3~tracked_ptr	996
10.384.3Member Function Documentation	996
10.384.3.1operator bool	996
10.384.3.2operator!	997
10.384.3.3operator->	997

10.384.3.4operator->	997
10.384.3.5operator=	997
10.384.3.6operator=	997
10.385mln::util::tree< T > Class Template Reference	997
10.385.1Detailed Description	998
10.385.2Constructor & Destructor Documentation	998
10.385.2.1tree	998
10.385.2.2tree	998
10.385.3Member Function Documentation	998
10.385.3.1add_tree_down	998
10.385.3.2add_tree_up	999
10.385.3.3check_consistency	999
10.385.3.4main_branch	999
10.385.3.5root	999
10.386mln::util::tree_node< T > Class Template Reference	999
10.386.1Detailed Description	1001
10.386.2Constructor & Destructor Documentation	1001
10.386.2.1tree_node	1001
10.386.2.2tree_node	1001
10.386.3Member Function Documentation	1001
10.386.3.1add_child	1001
10.386.3.2add_child	1001
10.386.3.3check_consistency	1002
10.386.3.4children	1002
10.386.3.5children	1002
10.386.3.6delete_tree_node	1002
10.386.3.7elt	1002
10.386.3.8elt	1002
10.386.3.9parent	1003
10.386.3.10print	1003
10.386.3.11search	1003
10.386.3.12search_rec	1003
10.386.3.13set_parent	1003
10.387mln::util::vertex< G > Class Template Reference	1004
10.387.1Detailed Description	1005
10.387.2Member Typedef Documentation	1006

10.387.2.1	Category	1006
10.387.2.2	graph_t	1006
10.387.2.3	id_t	1006
10.387.2.4	id_value_t	1006
10.387.3	Constructor & Destructor Documentation	1006
10.387.3.1	vertex	1006
10.387.4	Member Function Documentation	1006
10.387.4.1	change_graph	1006
10.387.4.2	edge_with	1006
10.387.4.3	graph	1006
10.387.4.4	id	1007
10.387.4.5	invalidate	1007
10.387.4.6	is_valid	1007
10.387.4.7	nth_nbh_edge	1007
10.387.4.8	nth_nbh_vertex	1007
10.387.4.9	max_nbh_edges	1007
10.387.4.10	max_nbh_vertices	1007
10.387.4.11	operator vertex_id_t	1007
10.387.4.12	other	1008
10.387.4.13	update_id	1008
10.388	in::util::yes Struct Reference	1008
10.388.1	Detailed Description	1008
10.389	in::Value< E > Struct Template Reference	1008
10.389.1	Detailed Description	1009
10.390	in::value::float01 Class Reference	1010
10.390.1	Detailed Description	1010
10.390.2	Member Typedef Documentation	1011
10.390.2.1	lenc	1011
10.390.2.2	equiv	1011
10.390.3	Constructor & Destructor Documentation	1011
10.390.3.1	float01	1011
10.390.3.2	float01	1011
10.390.3.3	float01	1011
10.390.4	Member Function Documentation	1011
10.390.4.1	Inbits	1011
10.390.4.2	operator float	1011



10.390.4.3set_nbits . . . . .	1011
10.390.4.4to_nbits . . . . .	1011
10.390.4.5value . . . . .	1012
10.390.4.6value_ind . . . . .	1012
10.391mln::value::float01_f Struct Reference . . . . .	1012
10.391.1.Detailed Description . . . . .	1012
10.391.2.Constructor & Destructor Documentation . . . . .	1012
10.391.2.1float01_f . . . . .	1012
10.391.2.2float01_f . . . . .	1012
10.391.3.Member Function Documentation . . . . .	1013
10.391.3.1operator float . . . . .	1013
10.391.3.2operator= . . . . .	1013
10.391.3.3value . . . . .	1013
10.392mln::value::graylevel< n > Struct Template Reference . . . . .	1013
10.392.1.Detailed Description . . . . .	1014
10.392.2.Constructor & Destructor Documentation . . . . .	1014
10.392.2.1graylevel . . . . .	1014
10.392.2.2graylevel . . . . .	1014
10.392.2.3graylevel . . . . .	1015
10.392.2.4graylevel . . . . .	1015
10.392.2.5graylevel . . . . .	1015
10.392.3.Member Function Documentation . . . . .	1015
10.392.3.1operator= . . . . .	1015
10.392.3.2operator= . . . . .	1015
10.392.3.3operator= . . . . .	1015
10.392.3.4operator= . . . . .	1015
10.392.3.5to_float . . . . .	1015
10.392.3.6value . . . . .	1016
10.393mln::value::graylevel_f Struct Reference . . . . .	1016
10.393.1.Detailed Description . . . . .	1017
10.393.2.Constructor & Destructor Documentation . . . . .	1017
10.393.2.1graylevel_f . . . . .	1017
10.393.2.2graylevel_f . . . . .	1017
10.393.2.3graylevel_f . . . . .	1017
10.393.2.4graylevel_f . . . . .	1017
10.393.2.5graylevel_f . . . . .	1017

10.393.3	Member Function Documentation	1017
10.393.3.1	operator graylevel< n >	1017
10.393.3.2	operator=	1017
10.393.3.3	operator=	1018
10.393.3.4	operator=	1018
10.393.3.5	operator=	1018
10.393.3.6	value	1018
10.394	ln::value::int_s< n > Struct Template Reference	1018
10.394.	Detailed Description	1019
10.394.	Constructor & Destructor Documentation	1019
10.394.2.	lint_s	1019
10.394.2.2	int_s	1019
10.394.2.3	int_s	1019
10.394.3	Member Function Documentation	1020
10.394.3.	operator int	1020
10.394.3.2	operator=	1020
10.394.4	Member Data Documentation	1020
10.394.4.	lone	1020
10.394.4.2	zero	1020
10.395	ln::value::int_u< n > Struct Template Reference	1020
10.395.	Detailed Description	1021
10.395.	Constructor & Destructor Documentation	1021
10.395.2.	lint_u	1021
10.395.2.2	int_u	1021
10.395.2.3	int_u	1021
10.395.3	Member Function Documentation	1022
10.395.3.	lnext	1022
10.395.3.2	operator unsigned	1022
10.395.3.3	operator-	1022
10.395.3.4	operator=	1022
10.396	ln::value::int_u_sat< n > Struct Template Reference	1022
10.396.	Detailed Description	1023
10.396.	Constructor & Destructor Documentation	1023
10.396.2.	lint_u_sat	1023
10.396.2.2	int_u_sat	1023
10.396.3	Member Function Documentation	1024

10.396.3.1operator int . . . . .	1024
10.396.3.2operator+= . . . . .	1024
10.396.3.3operator-= . . . . .	1024
10.396.3.4operator= . . . . .	1024
10.396.4Member Data Documentation . . . . .	1024
10.396.4.1one . . . . .	1024
10.396.4.2zero . . . . .	1024
10.397In::value::Integer< E > Struct Template Reference . . . . .	1024
10.397.Detailed Description . . . . .	1025
10.398In::value::Integer< void > Struct Template Reference . . . . .	1025
10.398.Detailed Description . . . . .	1025
10.399In::value::label< n > Struct Template Reference . . . . .	1025
10.399.Detailed Description . . . . .	1026
10.399.2Member Typedef Documentation . . . . .	1026
10.399.2.1enc . . . . .	1026
10.399.3Constructor & Destructor Documentation . . . . .	1027
10.399.3.1label . . . . .	1027
10.399.3.2label . . . . .	1027
10.399.3.3label . . . . .	1027
10.399.4Member Function Documentation . . . . .	1027
10.399.4.1next . . . . .	1027
10.399.4.2operator unsigned . . . . .	1027
10.399.4.3operator++ . . . . .	1027
10.399.4.4operator-- . . . . .	1027
10.399.4.5operator= . . . . .	1027
10.399.4.6operator= . . . . .	1027
10.399.4.7prev . . . . .	1028
10.400In::value::lut_vec< S, T > Struct Template Reference . . . . .	1028
10.400.Detailed Description . . . . .	1029
10.400.2Member Typedef Documentation . . . . .	1029
10.400.2.1bkd_viter . . . . .	1029
10.400.2.2fwd_viter . . . . .	1029
10.400.2.3value . . . . .	1030
10.400.3Constructor & Destructor Documentation . . . . .	1030
10.400.3.1lut_vec . . . . .	1030
10.400.3.2lut_vec . . . . .	1030

10.400.3.lut_vec . . . . .	1030
10.400.4.Member Function Documentation . . . . .	1030
10.400.4.1.has . . . . .	1030
10.400.4.2.index_of . . . . .	1030
10.400.4.3.values . . . . .	1030
10.400.4.4.operator[] . . . . .	1030
10.401.mn::value::proxy< I > Class Template Reference . . . . .	1031
10.401.1.Detailed Description . . . . .	1032
10.401.2.Member Typedef Documentation . . . . .	1032
10.401.2.1.enum . . . . .	1032
10.401.2.2.equival . . . . .	1032
10.401.3.Constructor & Destructor Documentation . . . . .	1032
10.401.3.1.proxy . . . . .	1032
10.401.3.2.proxy . . . . .	1032
10.401.3.3.~proxy . . . . .	1032
10.401.4.Member Function Documentation . . . . .	1033
10.401.4.1.operator= . . . . .	1033
10.401.4.2.operator= . . . . .	1033
10.401.4.3.to_value . . . . .	1033
10.402.mn::value::qt::rgb32 Struct Reference . . . . .	1033
10.402.1.Detailed Description . . . . .	1034
10.402.2.Constructor & Destructor Documentation . . . . .	1034
10.402.2.1.rgb32 . . . . .	1034
10.402.2.2.rgb32 . . . . .	1034
10.402.2.3.rgb32 . . . . .	1034
10.402.2.4.rgb32 . . . . .	1034
10.402.3.Member Function Documentation . . . . .	1034
10.402.3.1.operator= . . . . .	1034
10.402.3.2.red . . . . .	1034
10.402.4.Member Data Documentation . . . . .	1034
10.402.4.1.zero . . . . .	1034
10.403.mn::value::rgb< n > Struct Template Reference . . . . .	1035
10.403.1.Detailed Description . . . . .	1035
10.403.2.Constructor & Destructor Documentation . . . . .	1035
10.403.2.1.rgb . . . . .	1035
10.403.2.2.rgb . . . . .	1036

10.403.2.3rgb	1036
10.403.2.4rgb	1036
10.403.3Member Function Documentation	1036
10.403.3.1operator=	1036
10.403.3.2red	1036
10.403.4Member Data Documentation	1036
10.403.4.1zero	1036
10.404In::value::set< T > Struct Template Reference	1036
10.404.1Detailed Description	1037
10.404.2Member Function Documentation	1037
10.404.2.1the	1037
10.405In::value::sign Class Reference	1037
10.405.1Detailed Description	1038
10.405.2Member Typedef Documentation	1038
10.405.2.1enc	1038
10.405.2.2equiv	1038
10.405.3Constructor & Destructor Documentation	1038
10.405.3.1sign	1038
10.405.3.2sign	1038
10.405.3.3sign	1038
10.405.4Member Function Documentation	1038
10.405.4.1operator int	1038
10.405.4.2operator=	1039
10.405.5Member Data Documentation	1039
10.405.5.1one	1039
10.405.5.2zero	1039
10.406In::value::stack_image< n, I > Struct Template Reference	1039
10.406.1Detailed Description	1040
10.406.2Member Typedef Documentation	1040
10.406.2.1domain_t	1040
10.406.2.2value	1040
10.406.2.3psite	1040
10.406.2.4value	1040
10.406.2.5skeleton	1040
10.406.2.6value	1041
10.406.3Constructor & Destructor Documentation	1041

10.406.3. lstack_image . . . . .	1041
10.406.4. Member Function Documentation . . . . .	1041
10.406.4. lis_valid . . . . .	1041
10.406.4.2operator() . . . . .	1041
10.406.4.3operator() . . . . .	1041
10.407. In::value::super_value< sign > Struct Template Reference . . . . .	1041
10.407. Detailed Description . . . . .	1041
10.408. In::value::value_array< T, V > Struct Template Reference . . . . .	1041
10.408. Detailed Description . . . . .	1042
10.408. Constructor & Destructor Documentation . . . . .	1042
10.408.2. lvalue_array . . . . .	1042
10.408.3. Member Function Documentation . . . . .	1042
10.408.3. loperator() . . . . .	1042
10.408.3.2operator[] . . . . .	1042
10.408.3.3vset . . . . .	1042
10.409. In::Value_Iterator< E > Struct Template Reference . . . . .	1043
10.409. Detailed Description . . . . .	1043
10.409. Member Function Documentation . . . . .	1044
10.409.2. lnext . . . . .	1044
10.409. Friends And Related Function Documentation . . . . .	1044
10.409.3. loperator<< . . . . .	1044
10.410. In::Value_Set< E > Struct Template Reference . . . . .	1044
10.410. Detailed Description . . . . .	1045
10.411. In::Vertex< E > Struct Template Reference . . . . .	1045
10.411. Detailed Description . . . . .	1045
10.412. In::vertex_image< P, V, G > Class Template Reference . . . . .	1046
10.412. Detailed Description . . . . .	1046
10.412. Member Typedef Documentation . . . . .	1047
10.412.2. lgraph_t . . . . .	1047
10.412.2.2nbh_t . . . . .	1047
10.412.2.3site_function_t . . . . .	1047
10.412.2.4skeleton . . . . .	1047
10.412.2.5vertex_nbh_t . . . . .	1047
10.412.2.6vertex_win_t . . . . .	1047
10.412.2.7win_t . . . . .	1047
10.412. Constructor & Destructor Documentation . . . . .	1047

10.412.3. <code>lvertex_image</code> . . . . .	1047
10.412. <b>Member Function Documentation</b> . . . . .	1048
10.412.4. <code>operator()</code> . . . . .	1048
10.412. <b>ln::violent_cast_image&lt; T, I &gt; Struct Template Reference</b> . . . . .	1048
10.413. <b>Detailed Description</b> . . . . .	1048
10.413. <b>Member Typedef Documentation</b> . . . . .	1049
10.413.2. <code>lvalue</code> . . . . .	1049
10.413.2. <code>rvalue</code> . . . . .	1049
10.413.2. <code>skeleton</code> . . . . .	1049
10.413.2. <code>value</code> . . . . .	1049
10.413. <b>Constructor &amp; Destructor Documentation</b> . . . . .	1049
10.413.3. <code>lviolent_cast_image</code> . . . . .	1049
10.413. <b>Member Function Documentation</b> . . . . .	1049
10.413.4. <code>operator()</code> . . . . .	1049
10.413.4. <code>operator()</code> . . . . .	1049
10.412. <b>ln::w_window&lt; D, W &gt; Struct Template Reference</b> . . . . .	1049
10.414. <b>Detailed Description</b> . . . . .	1051
10.414. <b>Member Typedef Documentation</b> . . . . .	1051
10.414.2. <code>lbkd_qiter</code> . . . . .	1051
10.414.2. <code>dpsite</code> . . . . .	1051
10.414.2. <code>fwd_qiter</code> . . . . .	1051
10.414.2. <code>weight</code> . . . . .	1051
10.414. <b>Constructor &amp; Destructor Documentation</b> . . . . .	1051
10.414.3. <code>lw_window</code> . . . . .	1051
10.414. <b>Member Function Documentation</b> . . . . .	1052
10.414.4. <code>lclear</code> . . . . .	1052
10.414.4. <code>linsert</code> . . . . .	1052
10.414.4. <code>l3s_symmetric</code> . . . . .	1052
10.414.4. <code>l4std_vector</code> . . . . .	1052
10.414.4. <code>l5sym</code> . . . . .	1052
10.414.4. <code>l6w</code> . . . . .	1052
10.414.4. <code>l7weights</code> . . . . .	1052
10.414.4. <code>l8win</code> . . . . .	1053
10.414. <b>Friends And Related Function Documentation</b> . . . . .	1053
10.414.5. <code>loperator&lt;&lt;</code> . . . . .	1053
10.414.5. <code>l2operator==</code> . . . . .	1053

10.415	ln::Weighted_Window< E > Struct Template Reference	1053
10.415.1	Detailed Description	1054
10.415.2	Friends And Related Function Documentation	1054
10.415.2.1	operator-	1054
10.416	ln::win::backdiag2d Struct Reference	1054
10.416.1	Detailed Description	1054
10.416.2	Constructor & Destructor Documentation	1055
10.416.2.1	backdiag2d	1055
10.416.3	Member Function Documentation	1055
10.416.3.1	length	1055
10.417	ln::win::ball< G, C > Struct Template Reference	1055
10.417.1	Detailed Description	1055
10.417.2	Constructor & Destructor Documentation	1056
10.417.2.1	ball	1056
10.417.3	Member Function Documentation	1056
10.417.3.1	diameter	1056
10.418	ln::win::cube3d Struct Reference	1056
10.418.1	Detailed Description	1056
10.418.2	Constructor & Destructor Documentation	1057
10.418.2.1	cube3d	1057
10.418.3	Member Function Documentation	1057
10.418.3.1	length	1057
10.419	ln::win::cuboid3d Struct Reference	1057
10.419.1	Detailed Description	1058
10.419.2	Constructor & Destructor Documentation	1058
10.419.2.1	cuboid3d	1058
10.419.3	Member Function Documentation	1058
10.419.3.1	depth	1058
10.419.3.2	height	1059
10.419.3.3	volume	1059
10.419.3.4	width	1059
10.420	ln::win::diag2d Struct Reference	1059
10.420.1	Detailed Description	1059
10.420.2	Constructor & Destructor Documentation	1059
10.420.2.1	diag2d	1059
10.420.3	Member Function Documentation	1060



10.420.3. <code>length</code> . . . . .	1060
10.421 <code>ln::win::line&lt; M, i, C &gt;</code> Struct Template Reference . . . . .	1060
10.421.1. Detailed Description . . . . .	1060
10.421.2. Member Enumeration Documentation . . . . .	1061
10.421.2.1. <code>l"@87</code> . . . . .	1061
10.421.3. Constructor & Destructor Documentation . . . . .	1061
10.421.3.1. <code>lline</code> . . . . .	1061
10.421.4. Member Function Documentation . . . . .	1061
10.421.4.1. <code>length</code> . . . . .	1061
10.421.4.2. <code>size</code> . . . . .	1061
10.422 <code>ln::win::multiple&lt; W, F &gt;</code> Class Template Reference . . . . .	1061
10.422.1. Detailed Description . . . . .	1061
10.423 <code>ln::win::multiple_size&lt; n, W, F &gt;</code> Class Template Reference . . . . .	1062
10.423.1. Detailed Description . . . . .	1062
10.424 <code>ln::win::octagon2d</code> Struct Reference . . . . .	1062
10.424.1. Detailed Description . . . . .	1062
10.424.2. Constructor & Destructor Documentation . . . . .	1063
10.424.2.1. <code>loctagon2d</code> . . . . .	1063
10.424.3. Member Function Documentation . . . . .	1063
10.424.3.1. <code>larea</code> . . . . .	1063
10.424.3.2. <code>length</code> . . . . .	1063
10.425 <code>ln::win::rectangle2d</code> Struct Reference . . . . .	1063
10.425.1. Detailed Description . . . . .	1064
10.425.2. Constructor & Destructor Documentation . . . . .	1064
10.425.2.1. <code>lrectangle2d</code> . . . . .	1064
10.425.3. Member Function Documentation . . . . .	1064
10.425.3.1. <code>larea</code> . . . . .	1064
10.425.3.2. <code>height</code> . . . . .	1064
10.425.3.3. <code>lstd_vector</code> . . . . .	1064
10.425.3.4. <code>width</code> . . . . .	1064
10.426 <code>ln::Window&lt; E &gt;</code> Struct Template Reference . . . . .	1065
10.426.1. Detailed Description . . . . .	1065
10.427 <code>ln::window&lt; D &gt;</code> Class Template Reference . . . . .	1065
10.427.1. Detailed Description . . . . .	1067
10.427.2. Member Typedef Documentation . . . . .	1067
10.427.2.1. <code>lbkd_qiter</code> . . . . .	1067

10.427.2.2	2fwd_qiter . . . . .	1067
10.427.2.3	qiter . . . . .	1067
10.427.2.4	regular . . . . .	1067
10.427.3	Constructor & Destructor Documentation . . . . .	1067
10.427.3.1	window . . . . .	1067
10.427.4	Member Function Documentation . . . . .	1067
10.427.4.1	clear . . . . .	1067
10.427.4.2	delta . . . . .	1068
10.427.4.3	dp . . . . .	1068
10.427.4.4	has . . . . .	1068
10.427.4.5	insert . . . . .	1068
10.427.4.6	insert . . . . .	1068
10.427.4.7	insert . . . . .	1068
10.427.4.8	is_centered . . . . .	1068
10.427.4.9	is_empty . . . . .	1069
10.427.4.10	is_symmetric . . . . .	1069
10.427.4.11	print . . . . .	1069
10.427.4.12	size . . . . .	1069
10.427.4.13	std_vector . . . . .	1069
10.427.4.14	sym . . . . .	1069
10.427.5	Friends And Related Function Documentation . . . . .	1069
10.427.5.1	operator== . . . . .	1069
10.428	mln::world::inter_pixel::is_separator Struct Reference . . . . .	1069
10.428.1	Detailed Description . . . . .	1070
10.429	trait::graph< I > Struct Template Reference . . . . .	1070
10.429.1	Detailed Description . . . . .	1071
10.430	trait::graph< mln::complex_image< 1, G, V > > Struct Template Reference . . . . .	1071
10.430.1	Detailed Description . . . . .	1071
10.431	trait::graph< mln::image2d< T > > Struct Template Reference . . . . .	1071
10.431.1	Detailed Description . . . . .	1071

# Chapter 1

## Documentation of milena

### 1.1 Introduction

This is the documentation of Milena.

### 1.2 Overview of Milena.

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

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

### 1.3 Copyright and License.

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

This documentation is part of Olena.

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

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

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

## Chapter 2

# Quick Reference Guide

- installation
- foreword
- site
- siteset
- image
- winneigh
- sitesandco
- iterators
- imamemmgmt
- basicops
- inputoutput
- graphandima
- globalvars
- macros
- compilerrors



# Chapter 3

## Tutorial

- tuto1
- tuto2
- tuto3
- tuto4
- tuto5
- tuto6
- tuto7
- tuto8





# Chapter 4

## Module Index

### 4.1 Modules

Here is a list of all modules:

Types . . . . .	74
Graphes . . . . .	70
Images . . . . .	71
Basic types . . . . .	71
Image morphers . . . . .	72
Values morphers . . . . .	72
Domain morphers . . . . .	73
Identity morphers . . . . .	74
Neighborhoods . . . . .	77
1D neighborhoods . . . . .	77
2D neighborhoods . . . . .	78
3D neighborhoods . . . . .	80
Site sets . . . . .	83
Basic types . . . . .	84
Graph based . . . . .	84
Complex based . . . . .	85
Sparse types . . . . .	85
Queue based . . . . .	86
Utilities . . . . .	86
Windows . . . . .	87
1D windows . . . . .	88
2D windows . . . . .	89
3D windows . . . . .	91
N-D windows . . . . .	93
Multiple windows . . . . .	94
Accumulators . . . . .	75
On site sets . . . . .	67
On images . . . . .	67
On values . . . . .	68
Multiple accumulators . . . . .	70
Routines . . . . .	75
Canvas . . . . .	75

Functions . . . . .	76
v2w2v functions . . . . .	94
v2w_w2v functions . . . . .	94
vv2b functions . . . . .	94

# Chapter 5

## Namespace Index

### 5.1 Namespace List

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

<a href="#">mln</a> ( <a href="#">Mln/convert/to_image.hh</a> ) . . . . .	95
<a href="#">mln::accu</a> (Namespace of accumulators) . . . . .	136
<a href="#">mln::accu::image</a> (Namespace of accumulator image routines) . . . . .	139
<a href="#">mln::accu::impl</a> (Implementation namespace of accumulator namespace) . . . . .	139
<a href="#">mln::accu::logic</a> (Namespace of logical accumulators) . . . . .	140
<a href="#">mln::accu::math</a> (Namespace of mathematic accumulators) . . . . .	140
<a href="#">mln::accu::meta::logic</a> (Namespace of logical meta-accumulators) . . . . .	141
<a href="#">mln::accu::meta::math</a> (Namespace of mathematic meta-accumulators) . . . . .	141
<a href="#">mln::accu::meta::shape</a> (Namespace of shape meta-accumulators) . . . . .	142
<a href="#">mln::accu::meta::stat</a> (Namespace of statistical meta-accumulators) . . . . .	142
<a href="#">mln::accu::shape</a> (Namespace of shape accumulators) . . . . .	143
<a href="#">mln::accu::stat</a> (Namespace of statistical accumulators) . . . . .	143
<a href="#">mln::algebra</a> (Namespace of algebraic structure) . . . . .	144
<a href="#">mln::arith</a> (Namespace of arithmetic) . . . . .	146
<a href="#">mln::arith::impl</a> (Implementation namespace of arith namespace) . . . . .	158
<a href="#">mln::arith::impl::generic</a> (Generic implementation namespace of arith namespace) . . . . .	158
<a href="#">mln::binarization</a> (Namespace of "point-wise" expression tools) . . . . .	158
<a href="#">mln::border</a> (Namespace of routines related to image virtual (outer) border) . . . . .	159
<a href="#">mln::border::impl</a> (Implementation namespace of border namespace) . . . . .	163
<a href="#">mln::border::impl::generic</a> (Generic implementation namespace of border namespace) . . . . .	163
<a href="#">mln::canvas</a> (Namespace of canvas) . . . . .	163
<a href="#">mln::canvas::browsing</a> (Namespace of browsing canvas) . . . . .	165
<a href="#">mln::canvas::impl</a> (Implementation namespace of canvas namespace) . . . . .	165
<a href="#">mln::canvas::labeling</a> (Namespace of labeling canvas) . . . . .	166
<a href="#">mln::canvas::labeling::impl</a> (Implementation namespace of labeling canvas namespace) . . . . .	167
<a href="#">mln::canvas::morpho</a> (Namespace of morphological canvas) . . . . .	167
<a href="#">mln::convert</a> (Namespace of conversion routines) . . . . .	167
<a href="#">mln::data</a> (Namespace of image processing routines related to pixel data) . . . . .	173
<a href="#">mln::data::approx</a> (Namespace of image processing routines related to pixel levels with approximation) . . . . .	184
<a href="#">mln::data::approx::impl</a> (Implementation namespace of <a href="#">data::approx</a> namespace) . . . . .	185
<a href="#">mln::data::impl</a> (Implementation namespace of data namespace) . . . . .	185
<a href="#">mln::data::impl::generic</a> (Generic implementation namespace of data namespace) . . . . .	188

<code>mln::data::naive</code> (Namespace of image processing routines related to pixel levels with naive approach) . . . . .	191
<code>mln::data::naive::impl</code> (Implementation namespace of <code>data::naive</code> namespace) . . . . .	192
<code>mln::debug</code> (Namespace of routines that help to debug) . . . . .	192
<code>mln::debug::impl</code> (Implementation namespace of debug namespace) . . . . .	198
<code>mln::def</code> (Namespace for core definitions) . . . . .	198
<code>mln::display</code> (Namespace of routines that help to display images) . . . . .	199
<code>mln::display::impl</code> (Implementation namespace of display namespace) . . . . .	199
<code>mln::display::impl::generic</code> (Generic implementation namespace of display namespace) . . . . .	200
<code>mln::doc</code> (The namespace <code>mln::doc</code> is only for documentation purpose) . . . . .	200
<code>mln::draw</code> (Namespace of drawing routines) . . . . .	201
<code>mln::estim</code> (Namespace of estimation materials) . . . . .	203
<code>mln::extension</code> (Namespace of extension tools) . . . . .	205
<code>mln::fun</code> (Namespace of functions) . . . . .	208
<code>mln::fun::access</code> (Namespace for access functions) . . . . .	209
<code>mln::fun::i2v</code> (Namespace of integer-to-value functions) . . . . .	209
<code>mln::fun::n2v</code> (Namespace of functions from nil to value) . . . . .	210
<code>mln::fun::p2b</code> (Namespace of functions from point to boolean) . . . . .	210
<code>mln::fun::p2p</code> (Namespace of functions from grid point to grid point) . . . . .	210
<code>mln::fun::p2v</code> (Namespace of functions from point to value) . . . . .	211
<code>mln::fun::stat</code> (Namespace of statistical functions) . . . . .	211
<code>mln::fun::v2b</code> (Namespace of functions from value to logic value) . . . . .	211
<code>mln::fun::v2i</code> (Namespace of value-to-integer functions) . . . . .	211
<code>mln::fun::v2v</code> (Namespace of functions from value to value) . . . . .	212
<code>mln::fun::v2w2v</code> (Namespace of bijective functions) . . . . .	213
<code>mln::fun::v2w_w2v</code> (Namespace of functions from value to value) . . . . .	213
<code>mln::fun::vv2b</code> (Namespace of functions from value to value) . . . . .	214
<code>mln::fun::vv2v</code> (Namespace of functions from a couple of values to a value) . . . . .	214
<code>mln::fun::x2p</code> (Namespace of functions from point to value) . . . . .	215
<code>mln::fun::x2v</code> (Namespace of functions from vector to value) . . . . .	216
<code>mln::fun::x2x</code> (Namespace of functions from vector to vector) . . . . .	216
<code>mln::geom</code> (Namespace of all things related to geometry) . . . . .	216
<code>mln::geom::impl</code> (Implementation namespace of geom namespace) . . . . .	229
<code>mln::graph</code> (Namespace of graph related routines) . . . . .	230
<code>mln::grid</code> (Namespace of grids definitions) . . . . .	232
<code>mln::histo</code> (Namespace of histograms) . . . . .	232
<code>mln::histo::impl</code> (Implementation namespace of histo namespace) . . . . .	233
<code>mln::histo::impl::generic</code> (Generic implementation namespace of histo namespace) . . . . .	233
<code>mln::impl</code> (Implementation namespace of mln namespace) . . . . .	234
<code>mln::io</code> (Namespace of input/output handling) . . . . .	234
<code>mln::io::cloud</code> (Namespace of cloud input/output handling) . . . . .	235
<code>mln::io::dicom</code> (Namespace of DICOM input/output handling) . . . . .	236
<code>mln::io::dump</code> (Namespace of dump input/output handling) . . . . .	237
<code>mln::io::fits</code> (Namespace of fits input/output handling) . . . . .	238
<code>mln::io::fld</code> (Namespace of pgm input/output handling) . . . . .	239
<code>mln::io::magick</code> (Namespace of magick input/output handling) . . . . .	240
<code>mln::io::off</code> (Namespace of off input/output handling) . . . . .	241
<code>mln::io::pbm</code> (Namespace of pbm input/output handling) . . . . .	243
<code>mln::io::pbm::impl</code> (Namespace of pbm implementation details) . . . . .	244
<code>mln::io::pbms</code> (Namespace of pbms input/output handling) . . . . .	244
<code>mln::io::pbms::impl</code> (Namespace of pbms implementation details) . . . . .	245
<code>mln::io::pfm</code> (Namespace of pfm input/output handling) . . . . .	245
<code>mln::io::pfm::impl</code> (Implementation namespace of pfm namespace) . . . . .	246
<code>mln::io::pgm</code> (Namespace of pgm input/output handling) . . . . .	246

<a href="#">mln::io::pgms</a> (Namespace of pgms input/output handling ) . . . . .	247
<a href="#">mln::io::plot</a> (Namespace of plot input/output handling ) . . . . .	248
<a href="#">mln::io::pnm</a> (Namespace of pnm input/output handling ) . . . . .	250
<a href="#">mln::io::pnm::impl</a> (Namespace of pnm's implementation details ) . . . . .	251
<a href="#">mln::io::pnms</a> (Namespace of pnms input/output handling ) . . . . .	252
<a href="#">mln::io::ppm</a> (Namespace of ppm input/output handling ) . . . . .	253
<a href="#">mln::io::ppms</a> (Namespace of ppms input/output handling ) . . . . .	254
<a href="#">mln::io::raw</a> (Namespace of raw input/output handling ) . . . . .	254
<a href="#">mln::io::tiff</a> (Namespace of tiff input/output handling ) . . . . .	256
<a href="#">mln::io::txt</a> (Namespace of txt input/output handling ) . . . . .	256
<a href="#">mln::labeling</a> (Namespace of labeling routines ) . . . . .	257
<a href="#">mln::labeling::impl</a> (Implementation namespace of labeling namespace ) . . . . .	271
<a href="#">mln::labeling::impl::generic</a> (Generic implementation namespace of labeling namespace ) . . . . .	273
<a href="#">mln::linear</a> (Namespace of linear image processing routines ) . . . . .	275
<a href="#">mln::linear::impl</a> (Namespace of linear image processing routines implementation details ) . . . . .	279
<a href="#">mln::linear::local</a> (Specializations of local linear routines ) . . . . .	279
<a href="#">mln::linear::local::impl</a> (Namespace of local linear routines implementation details ) . . . . .	280
<a href="#">mln::literal</a> (Namespace of literals ) . . . . .	280
<a href="#">mln::logical</a> (Namespace of logic ) . . . . .	286
<a href="#">mln::logical::impl</a> (Implementation namespace of logical namespace ) . . . . .	289
<a href="#">mln::logical::impl::generic</a> (Generic implementation namespace of logical namespace ) . . . . .	289
<a href="#">mln::make</a> (Namespace of routines that help to make Milena's objects ) . . . . .	290
<a href="#">mln::math</a> (Namespace of mathematical routines ) . . . . .	313
<a href="#">mln::metal</a> (Namespace of meta-programming tools ) . . . . .	314
<a href="#">mln::metal::impl</a> (Implementation namespace of metal namespace ) . . . . .	315
<a href="#">mln::metal::math</a> (Namespace of static mathematical functions ) . . . . .	315
<a href="#">mln::metal::math::impl</a> (Implementation namespace of <a href="#">metal::math</a> namespace ) . . . . .	315
<a href="#">mln::morpho</a> (Namespace of mathematical morphology routines ) . . . . .	315
<a href="#">mln::morpho::approx</a> (Namespace of approximate mathematical morphology routines ) . . . . .	324
<a href="#">mln::morpho::attribute</a> (Namespace of attributes used in mathematical morphology ) . . . . .	325
<a href="#">mln::morpho::closing::approx</a> (Namespace of approximate mathematical morphology closing routines ) . . . . .	325
<a href="#">mln::morpho::elementary</a> (Namespace of image processing routines of elementary mathematical morphology ) . . . . .	326
<a href="#">mln::morpho::impl</a> (Namespace of mathematical morphology routines implementations ) . . . . .	328
<a href="#">mln::morpho::impl::generic</a> (Namespace of mathematical morphology routines generic implementations ) . . . . .	328
<a href="#">mln::morpho::opening::approx</a> (Namespace of approximate mathematical morphology opening routines ) . . . . .	328
<a href="#">mln::morpho::reconstruction</a> (Namespace of morphological reconstruction routines ) . . . . .	329
<a href="#">mln::morpho::reconstruction::by_dilation</a> (Namespace of morphological reconstruction by dilation routines ) . . . . .	329
<a href="#">mln::morpho::reconstruction::by_erosion</a> (Namespace of morphological reconstruction by erosion routines ) . . . . .	329
<a href="#">mln::morpho::tree</a> (Namespace of morphological tree-related routines ) . . . . .	330
<a href="#">mln::morpho::tree::filter</a> (Namespace for attribute filtering ) . . . . .	337
<a href="#">mln::morpho::watershed</a> (Namespace of morphological watershed routines ) . . . . .	339
<a href="#">mln::morpho::watershed::watershed</a> (Namespace of morphological watershed routines implementations ) . . . . .	341
<a href="#">mln::morpho::watershed::watershed::generic</a> (Namespace of morphological watershed routines generic implementations ) . . . . .	341
<a href="#">mln::norm</a> (Namespace of norms ) . . . . .	342
<a href="#">mln::norm::impl</a> (Implementation namespace of norm namespace ) . . . . .	344
<a href="#">mln::opt</a> (Namespace of optional routines ) . . . . .	344

---

<a href="#">mln::opt::impl</a> (Implementation namespace of opt namespace) . . . . .	345
<a href="#">mln::pw</a> (Namespace of "point-wise" expression tools) . . . . .	345
<a href="#">mln::registration</a> (Namespace of "point-wise" expression tools) . . . . .	346
<a href="#">mln::select</a> (Select namespace (FIXME doc)) . . . . .	348
<a href="#">mln::set</a> (Namespace of image processing routines related to pixel sets) . . . . .	349
<a href="#">mln::subsampling</a> (Namespace of "point-wise" expression tools) . . . . .	351
<a href="#">mln::tag</a> (Namespace of image processing routines related to tags) . . . . .	352
<a href="#">mln::test</a> (Namespace of image processing routines related to pixel tests) . . . . .	353
<a href="#">mln::test::impl</a> (Implementation namespace of test namespace) . . . . .	354
<a href="#">mln::topo</a> (Namespace of "point-wise" expression tools) . . . . .	354
<a href="#">mln::trace</a> (Namespace of routines related to the trace mechanism) . . . . .	364
<a href="#">mln::trait</a> (Namespace where traits are defined) . . . . .	364
<a href="#">mln::transform</a> (Namespace of transforms) . . . . .	364
<a href="#">mln::util</a> (Namespace of tools using for more complex algorithm) . . . . .	369
<a href="#">mln::util::impl</a> (Implementation namespace of util namespace) . . . . .	375
<a href="#">mln::value</a> (Namespace of materials related to pixel value types) . . . . .	375
<a href="#">mln::value::impl</a> (Implementation namespace of value namespace) . . . . .	386
<a href="#">mln::win</a> (Namespace of image processing routines related to win) . . . . .	386

# Chapter 6

## Class Index

### 6.1 Class Hierarchy

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

mln::Generalized_Pixel< bkd_pixter1d< I > > . . . . .	659
mln::Pixel_Iterator< bkd_pixter1d< I > > . . . . .	855
pixel_iterator_base_< I, bkd_pixter1d< I > >	
backward_pixel_iterator_base_< I, bkd_pixter1d< I > >	
mln::bkd_pixter1d< I > . . . . .	491
mln::Generalized_Pixel< bkd_pixter2d< I > > . . . . .	659
mln::Pixel_Iterator< bkd_pixter2d< I > > . . . . .	855
pixel_iterator_base_< I, bkd_pixter2d< I > >	
backward_pixel_iterator_base_< I, bkd_pixter2d< I > >	
mln::bkd_pixter2d< I > . . . . .	492
mln::Generalized_Pixel< bkd_pixter3d< I > > . . . . .	659
mln::Pixel_Iterator< bkd_pixter3d< I > > . . . . .	855
pixel_iterator_base_< I, bkd_pixter3d< I > >	
backward_pixel_iterator_base_< I, bkd_pixter3d< I > >	
mln::bkd_pixter3d< I > . . . . .	493
mln::Generalized_Pixel< dpoints_bkd_pixter< I > > . . . . .	659
mln::Pixel_Iterator< dpoints_bkd_pixter< I > > . . . . .	855
mln::dpoints_bkd_pixter< I > . . . . .	586
mln::Generalized_Pixel< dpoints_fwd_pixter< I > > . . . . .	659
mln::Pixel_Iterator< dpoints_fwd_pixter< I > > . . . . .	855
mln::dpoints_fwd_pixter< I > . . . . .	588
mln::Generalized_Pixel< fwd_pixter1d< I > > . . . . .	659
mln::Pixel_Iterator< fwd_pixter1d< I > > . . . . .	855
pixel_iterator_base_< I, fwd_pixter1d< I > >	
forward_pixel_iterator_base_< I, fwd_pixter1d< I > >	
mln::fwd_pixter1d< I > . . . . .	654
mln::Generalized_Pixel< fwd_pixter2d< I > > . . . . .	659
mln::Pixel_Iterator< fwd_pixter2d< I > > . . . . .	855
pixel_iterator_base_< I, fwd_pixter2d< I > >	
forward_pixel_iterator_base_< I, fwd_pixter2d< I > >	
mln::fwd_pixter2d< I > . . . . .	655

mln::Generalized_Pixel< fwd_pixter3d< I > > . . . . .	659
mln::Pixel_Iterator< fwd_pixter3d< I > > . . . . .	855
pixel_iterator_base_< I, fwd_pixter3d< I > >	
forward_pixel_iterator_base_< I, fwd_pixter3d< I > >	
mln::fwd_pixter3d< I > . . . . .	657
mln::Generalized_Pixel< pixel< I > > . . . . .	659
mln::pixel< I > . . . . .	853
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 > . . . . .	1039
mln::internal::image_base< const I::value, I::domain_t, E >	
image_morpher< const I, const I::value, I::domain_t, E >	
image_identity< const I, I::domain_t, E >	
mln::labeled_image_base< I, E > . . . . .	721
mln::internal::image_base< const I::value, I::domain_t, labeled_image< I > >	
image_morpher< const I, const I::value, I::domain_t, labeled_image< I > >	
image_identity< const I, I::domain_t, labeled_image< I > >	
mln::labeled_image_base< I, labeled_image< I > > . . . . .	721
mln::labeled_image< I > . . . . .	718
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 > . . . . .	887
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 > . . . . .	647
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 > . . . . .	871
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 > . . . . .	593
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 > . . . . .	1046
mln::internal::image_base< I::value, box2d, slice_image< I > >	
image_morpher< I, I::value, box2d, slice_image< I > >	
image_domain_morpher< I, box2d, slice_image< I > >	
mln::slice_image< I > . . . . .	882
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 > . . . . .	690
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 > . . . . .	595
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 > . . . . .	941
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 > . . . . .	535
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 > . . . . .	597
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 > . . . . .	599
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 > . . . . .	601
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 > . . . . .	713
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 > . . . . .	771
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 > . . . . .	856
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 > . . . . .	874
mln::internal::image_base< I::value, p_if< I::domain_t, F >, image_if< I, F > >	
image_morpher< I, I::value, p_if< I::domain_t, F >, image_if< I, F > >	
image_domain_morpher< I, p_if< I::domain_t, F >, image_if< I, F > >	
mln::image_if< I, F > . . . . .	712
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 > . . . . .	886
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 > . . . . . 940
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 > . . . . . 884
mln::internal::image_base< I::value, S, tr_image< S, I, T > >
    image_morpher< I, I::value, S, tr_image< S, I, T > >
    image_identity< I, S, tr_image< S, I, T > >
    mln::tr_image< S, I, T > . . . . . 937
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 > > . . . . . 690
    mln::image2d_h< V > . . . . . 704
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 > . . . . . 725
mln::internal::image_base< T, box1d, image1d< T > >
    image_primary< T, box1d, image1d< T > >
    mln::image1d< T > . . . . . 696
mln::internal::image_base< T, box3d, image3d< T > >
    image_primary< T, box3d, image3d< T > >
    mln::image3d< T > . . . . . 707
mln::internal::image_base< T, I::domain_t, E >
    image_morpher< I, T, I::domain_t, E >
mln::internal::image_base< T, I::domain_t, violent_cast_image< T, I > >
    image_morpher< I, T, I::domain_t, violent_cast_image< T, I > >
    image_value_morpher< I, T, violent_cast_image< T, I > >
    mln::violent_cast_image< T, I > . . . . . 1048
mln::internal::image_base< T, mln::box2d, image2d< T > >
    image_primary< T, mln::box2d, image2d< T > >
    mln::image2d< T > . . . . . 700
mln::internal::image_base< T, S, flat_image< T, S > >
    image_primary< T, S, flat_image< T, S > >
    mln::flat_image< T, S > . . . . . 606
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 > . . . . . 523
mln::internal::check::image_fastest< complex_image< D, G, V >, mln::metal::equal< mln_trait_
image_speed(complex_image< D, G, V >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest< decorated_image< I, D >, mln::metal::equal< mln_trait_
image_speed(decorated_image< I, D >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest< E, mln::metal::equal< mln_trait_image_speed(E),
trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest< edge_image< P, V, G >, mln::metal::equal< mln_trait_image_
speed(edge_image< P, V, G >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest< extended< I >, mln::metal::equal< mln_trait_image_

```

```

speed(extended< I >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< extension_fun< I, F >, mln::metal::equal< mln_trait_image_
speed(extension_fun< I, F >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< extension_ima< I, J >, mln::metal::equal< mln_trait_image_
speed(extension_ima< I, J >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< extension_val< I >, mln::metal::equal< mln_trait_image_
speed(extension_val< I >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< flat_image< T, S >, mln::metal::equal< mln_trait_image_
speed(flat_image< T, S >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< fun_image< F, I >, mln::metal::equal< mln_trait_image_
speed(fun_image< F, I >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< hexa< I >, mln::metal::equal< mln_trait_image_speed(hexa<
I >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< hexa< image2d< V > >, mln::metal::equal< mln_trait_
image_speed(hexa< image2d< V > >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< image1d< T >, mln::metal::equal< mln_trait_image_
speed(image1d< T >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< image2d< T >, mln::metal::equal< mln_trait_image_
speed(image2d< T >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< image3d< T >, mln::metal::equal< mln_trait_image_
speed(image3d< T >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< image< F, S >, mln::metal::equal< mln_trait_image_
speed(image< F, S >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< image_if< I, F >, mln::metal::equal< mln_trait_image_
speed(image_if< I, F >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< interpolated< I, F >, mln::metal::equal< mln_trait_image_
speed(interpolated< I, F >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< labeled_image< I >, mln::metal::equal< mln_trait_image_
speed(labeled_image< I >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< lazy_image< I, F, B >, mln::metal::equal< mln_trait_image_
speed(lazy_image< I, F, B >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< p2p_image< I, F >, mln::metal::equal< mln_trait_image_
speed(p2p_image< I, F >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< plain< I >, mln::metal::equal< mln_trait_image_speed(plain<
I >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< safe_image< I >, mln::metal::equal< mln_trait_image_
speed(safe_image< I >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< slice_image< I >, mln::metal::equal< mln_trait_image_
speed(slice_image< I >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< stack_image< n, I >, mln::metal::equal< mln_trait_image_
speed(stack_image< n, I >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< sub_image< I, S >, mln::metal::equal< mln_trait_image_
speed(sub_image< I, S >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< sub_image_if< I, S >, mln::metal::equal< mln_trait_image_
speed(sub_image_if< I, S >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< thrubin_image< I1, I2, F >, mln::metal::equal< mln_trait_
image_speed(thrubin_image< I1, I2, F >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< tr_image< S, I, T >, mln::metal::equal< mln_trait_image_
speed(tr_image< S, I, T >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< 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::internal::check::image_fastest_< violent_cast_image< T, I >, mln::metal::equal< mln_trait_	
image_speed(violent_cast_image< T, I >), trait::image::speed::fastest >::eval >	
mln::internal::impl_selector< W::center_t, W::psite, graph_window_piter< S, W, I > >	
mln::graph_window_piter< S, W, I > . . . . .	687
mln::value::Integer< graylevel< n > > . . . . .	1024
mln::value::graylevel< n > . . . . .	1013
mln::value::Integer< int_s< n > > . . . . .	1024
mln::value::int_s< n > . . . . .	1018
mln::value::Integer< int_u< n > > . . . . .	1024
mln::value::int_u< n > . . . . .	1020
mln::value::Integer< int_u_sat< n > > . . . . .	1024
mln::value::int_u_sat< n > . . . . .	1022
mln::value::Integer< object_id< Tag, V > > . . . . .	1024
mln::util::object_id< Tag, V > . . . . .	978
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 > . . . . .	686
mln::algebra::h_mat< d, T > . . . . .	487
mln::algebra::h_vec< d, C > . . . . .	489
mln::canvas::chamfer< F > . . . . .	523
mln::category< R(*) (A) > . . . . .	523
mln::Delta_Point_Site< void > . . . . .	539
mln::doc::Accumulator< E > . . . . .	539
mln::doc::Generalized_Pixel< E > . . . . .	553
mln::doc::Pixel_Iterator< E > . . . . .	564
mln::doc::Object< E > . . . . .	563
mln::doc::Dpoint< E > . . . . .	543
mln::doc::Image< E > . . . . .	554
mln::doc::Fastest_Image< E > . . . . .	545
mln::doc::Iterator< E > . . . . .	560
mln::doc::Pixel_Iterator< E > . . . . .	564
mln::doc::Site_Iterator< E > . . . . .	568
mln::doc::Value_Iterator< E > . . . . .	572
mln::doc::Neighborhood< E > . . . . .	561
mln::doc::Site_Set< E > . . . . .	570
mln::doc::Box< E > . . . . .	540
mln::doc::Value_Set< E > . . . . .	574
mln::doc::Weighted_Window< E > . . . . .	577
mln::doc::Window< E > . . . . .	579
mln::doc::Point_Site< E > . . . . .	566
mln::Edge< E > . . . . .	593
mln::fun::from_accu< A > . . . . .	608
mln::fun::internal::ch_function_value_impl< F, V >	
mln::fun::v2v::ch_function_value< F, V > . . . . .	613
mln::fun::x2p::closest_point< P > . . . . .	637
mln::fun::x2x::composed< T2, T1 > . . . . .	639
mln::Function< void > . . . . .	650
mln::Gdpoint< void > . . . . .	659
mln::Generalized_Pixel< E > . . . . .	659
mln::Pixel_Iterator< E > . . . . .	855

mln::internal::pixel_iterator_base_	
mln::geom::complex_geometry< D, P > . . . . .	660
mln::graph::attribute::card_t . . . . .	666
mln::graph::attribute::representative_t . . . . .	667
mln::histo::array< T > . . . . .	694
mln::internal::image_base< T, S, E >	
mln::internal::neighborhood_base< W, E >	
mln::internal::neighb_base	
mln::neighb< graph_elt_mixed_window< G, S, S2 > > . . . . .	769
mln::graph_elt_mixed_neighborhood< G, S, S2 > . . . . .	667
mln::neighb< graph_elt_window< G, S > > . . . . .	769
mln::graph_elt_neighborhood< G, S > . . . . .	672
mln::neighb< graph_elt_window_if< G, S, I > > . . . . .	769
mln::graph_elt_neighborhood_if< G, S, I > . . . . .	674
mln::internal::pixel_impl_< I, E >	
mln::internal::pixel_iterator_base_	
mln::io::dicom::dicom_header . . . . .	715
mln::io::dump::dump_header . . . . .	715
mln::io::fld::fld_header . . . . .	716
mln::io::raw::raw_header . . . . .	716
mln::metal::ands< E1, E2, E3, E4, E5, E6, E7, E8 > . . . . .	756
mln::metal::bool_< false >	
mln::metal::equal< T1::coord, T2::coord > . . . . .	756
mln::metal::equal< T1::point, T2::point > . . . . .	756
mln::metal::equal< T1, T2 > . . . . .	756
mln::metal::converts_to< T, U > . . . . .	756
mln::metal::goes_to< T, U > . . . . .	757
mln::metal::is< T, U > . . . . .	757
mln::metal::is_a< T, M > . . . . .	757
mln::metal::is_not< T, U > . . . . .	757
mln::metal::is_not_a< T, M > . . . . .	758
mln::Neighborhood< void > . . . . .	771
mln::Object< E > . . . . .	771
mln::Browsing< E > . . . . .	509
mln::Delta_Point_Site< E > . . . . .	538
mln::Dpoint< E > . . . . .	580
mln::Function< E > . . . . .	649
mln::Function_n2v< E > . . . . .	650
mln::Function_v2v< E > . . . . .	652
mln::fun::x2v::bilinear< I > . . . . .	638
mln::fun::x2v::trilinear< I > . . . . .	639
mln::fun::x2x::linear< I > . . . . .	640
mln::Function_v2b< E > . . . . .	651
mln::Function_vv2b< E > . . . . .	653
mln::Function_vv2v< E > . . . . .	653
mln::Gdpoint< E > . . . . .	658
mln::Graph< E > . . . . .	666
mln::Image< E > . . . . .	694
mln::Iterator< E > . . . . .	716
mln::Pixel_Iterator< E > . . . . .	855
mln::topo::internal::complex_iterator_base	
mln::topo::internal::complex_relative_iterator_base	
mln::Value_Iterator< E > . . . . .	1043

mln::Literal< E > . . . . .	727
mln::Mesh< E > . . . . .	751
mln::Regular_Grid< E > . . . . .	873
mln::Meta_Accumulator< E > . . . . .	752
mln::Meta_Function< E > . . . . .	754
mln::Meta_Function_v2v< E > . . . . .	754
mln::Meta_Function_vv2v< E > . . . . .	755
mln::Neighborhood< E > . . . . .	770
mln::Point_Site< E > . . . . .	866
mln::Proxy< E > . . . . .	870
mln::Accumulator< E > . . . . .	485
mln::accu::internal::base	
couple< accu::shape::bbox< P >, accu::math::count< P >, float, rectangularity< P >>	
mln::accu::site_set::rectangularity< P > . . . . .	457
mln::accu::pair< min< V >, max< V >> . . . . .	448
mln::accu::stat::min_max< V > . . . . .	471
mln::Site_Proxy< E > . . . . .	878
mln::Pseudo_Site< E > . . . . .	871
mln::Site_Iterator< E > . . . . .	876
mln::internal::site_iterator_base	
mln::internal::site_set_iterator_base	
mln::p_transformed_piter< Pi, S, F > . . . . .	843
mln::Site< E > . . . . .	875
mln::Gpoint< E > . . . . .	661
mln::Site_Set< E > . . . . .	879
mln::Box< E > . . . . .	501
mln::Value< E > . . . . .	1008
mln::Value_Set< E > . . . . .	1044
mln::Weighted_Window< E > . . . . .	1053
mln::Window< E > . . . . .	1065
mln::graph_window_base< P, E > . . . . .	684
mln::internal::window_base	
mln::Point_Site< void > . . . . .	870
mln::Proxy< void > . . . . .	870
mln::Pseudo_Site< void > . . . . .	871
mln::registration::closest_point_basic< P > . . . . .	873
mln::registration::closest_point_with_map< P > . . . . .	873
mln::select::p_of< P > . . . . .	875
mln::Site< void > . . . . .	876
mln::Site_Proxy< void > . . . . .	878
mln::Site_Set< void > . . . . .	882
mln::thru_image< I, F > . . . . .	887
mln::topo::complex< D > . . . . .	914
mln::topo::face< D > . . . . .	917
mln::topo::algebraic_face< D > . . . . .	902
mln::topo::n_face< N, D > . . . . .	927
mln::topo::algebraic_n_face< N, D > . . . . .	907
mln::topo::n_faces_set< N, D > . . . . .	933
mln::util::adjacency_matrix< V > . . . . .	943
mln::util::array< T > . . . . .	943
mln::util::branch< T > . . . . .	949

mln::util::branch_iter< T > . . . . .	951
mln::util::branch_iter_ind< T > . . . . .	952
mln::util::greater_point< I > . . . . .	969
mln::util::greater_psite< I > . . . . .	970
mln::util::head< T, R > . . . . .	970
mln::util::ilcell< T > . . . . .	971
mln::util::internal::edge_impl_< G >	
mln::util::edge< G > . . . . .	956
mln::util::internal::vertex_impl_< G >	
mln::util::vertex< G > . . . . .	1004
mln::util::node< T, R > . . . . .	978
mln::util::ord< T > . . . . .	980
mln::util::pix< I > . . . . .	982
mln::util::tracked_ptr< T > . . . . .	995
mln::util::tree< T > . . . . .	997
mln::util::tree_node< T > . . . . .	999
mln::value::float01 . . . . .	1010
mln::value::Integer< E > . . . . .	1024
mln::value::Integer< void > . . . . .	1025
mln::value::internal::value_like_< V, C, N, E >	
mln::value::float01_f . . . . .	1012
mln::value::graylevel< n > . . . . .	1013
mln::value::graylevel_f . . . . .	1016
mln::value::int_s< n > . . . . .	1018
mln::value::int_u< n > . . . . .	1020
mln::value::int_u_sat< n > . . . . .	1022
mln::value::label< n > . . . . .	1025
mln::value::qt::rgb32 . . . . .	1033
mln::value::rgb< n > . . . . .	1035
mln::value::set< T > . . . . .	1036
mln::value::sign . . . . .	1037
mln::value::super_value< sign > . . . . .	1041
mln::value::value_array< T, V > . . . . .	1041
mln::Vertex< E > . . . . .	1045
mln::internal::neighborhood_base< W, mixed_neighb< W > >	
neighb_base< W, mixed_neighb< W > >	
mln::mixed_neighb< W > . . . . .	758
mln::internal::neighborhood_base< W, neighb< W > >	
neighb_base< W, neighb< W > >	
mln::neighb< W > . . . . .	769
mln::Object< abs > . . . . .	771
mln::Meta_Function< abs > . . . . .	754
mln::Meta_Function_v2v< abs > . . . . .	754
mln::Object< abs< V > > . . . . .	771
mln::Function< abs< V > > . . . . .	649
mln::Function_v2v< abs< V > > . . . . .	652
mln::Object< accu_result > . . . . .	771
mln::Meta_Function< accu_result > . . . . .	754
mln::Meta_Function_v2v< accu_result > . . . . .	754
mln::Object< adj_higher_dim_connected_n_face_bkd_iter< D > >	
mln::Iterator< adj_higher_dim_connected_n_face_bkd_iter< D > >	716
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 >	889
mln::Object< adj_higher_dim_connected_n_face_fwd_iter< D >>	771
mln::Iterator< adj_higher_dim_connected_n_face_fwd_iter< D >>	716
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 >	890
mln::Object< adj_higher_face_bkd_iter< D >>	771
mln::Iterator< adj_higher_face_bkd_iter< D >>	716
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 >	891
mln::Object< adj_higher_face_fwd_iter< D >>	771
mln::Iterator< adj_higher_face_fwd_iter< D >>	716
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 >	892
mln::Object< adj_lower_dim_connected_n_face_bkd_iter< D >>	771
mln::Iterator< adj_lower_dim_connected_n_face_bkd_iter< D >>	716
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 >	893
mln::Object< adj_lower_dim_connected_n_face_fwd_iter< D >>	771
mln::Iterator< adj_lower_dim_connected_n_face_fwd_iter< D >>	716
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 >	894
mln::Object< adj_lower_face_bkd_iter< D >>	771
mln::Iterator< adj_lower_face_bkd_iter< D >>	716
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 > . . . . .	895
mln::Object< adj_lower_face_fwd_iter< D >> . . . . .	771
mln::Iterator< adj_lower_face_fwd_iter< D >> . . . . .	716
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 > . . . . .	896
mln::Object< adj_lower_higher_face_bkd_iter< D >> . . . . .	771
mln::Iterator< adj_lower_higher_face_bkd_iter< D >> . . . . .	716
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 > . . . . .	897
mln::Object< adj_lower_higher_face_fwd_iter< D >> . . . . .	771
mln::Iterator< adj_lower_higher_face_fwd_iter< D >> . . . . .	716
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 > . . . . .	898
mln::Object< adj_m_face_bkd_iter< D >> . . . . .	771
mln::Iterator< adj_m_face_bkd_iter< D >> . . . . .	716
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 > . . . . .	899
mln::Object< adj_m_face_fwd_iter< D >> . . . . .	771
mln::Iterator< adj_m_face_fwd_iter< D >> . . . . .	716
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 > . . . . .	901
mln::Object< all_to< T >> . . . . .	771
mln::Function< all_to< T >> . . . . .	649
mln::Function_v2v< all_to< T >> . . . . .	652
mln::Object< antilogy > . . . . .	771
mln::Function< antilogy > . . . . .	649
mln::Function_v2v< antilogy > . . . . .	652
mln::Function_v2b< antilogy > . . . . .	651
mln::fun::p2b::antilogy . . . . .	609
mln::Object< array1d< T, Size >> . . . . .	771
mln::Object< array2d< T, r, c >> . . . . .	771
mln::Object< array3d< T, s, r, c >> . . . . .	771
mln::Object< array_bkd_iter< T >> . . . . .	771

mln::Proxy< array_bkd_iter< T > > . . . . .	870
mln::Object< array_fwd_iter< T > > . . . . .	771
mln::Proxy< array_fwd_iter< T > > . . . . .	870
mln::Object< asc_propagation > . . . . .	771
mln::Object< backdiag2d > . . . . .	771
mln::Window< backdiag2d > . . . . .	1065
window_base< dpoint2d, backdiag2d >	
classical_window_base< dpoint2d, backdiag2d >	
mln::win::backdiag2d . . . . .	1054
mln::Object< backdiagonal2d_t > . . . . .	771
mln::Browsing< backdiagonal2d_t > . . . . .	509
mln::canvas::browsing::backdiagonal2d_t . . . . .	509
mln::Object< ball< G, C > > . . . . .	771
mln::Window< ball< G, C > > . . . . .	1065
window_base< dpoint< G, C >, ball< G, C > >	
classical_window_base< dpoint< G, C >, ball< G, C > >	
mln::win::ball< G, C > . . . . .	1055
mln::Object< bbox > . . . . .	771
mln::Meta_Accumulator< bbox > . . . . .	752
mln::accu::meta::shape::bbox . . . . .	431
mln::Object< bbox< P > > . . . . .	771
mln::Proxy< bbox< P > > . . . . .	870
mln::Accumulator< bbox< P > > . . . . .	485
base< const box< P > &, bbox< P > >	
mln::accu::shape::bbox< P > . . . . .	452
mln::Object< big_chess< B > > . . . . .	771
mln::Function< big_chess< B > > . . . . .	649
mln::Function_v2v< big_chess< B > > . . . . .	652
mln::Function_v2b< big_chess< B > > . . . . .	651
mln::Object< bin_off_loader > . . . . .	771
mln::Object< bin_off_saver > . . . . .	771
mln::Object< binary< Fun, T1, T2 > > . . . . .	771
mln::Function< binary< Fun, T1, T2 > > . . . . .	649
mln::Function_v2v< binary< Fun, T1, T2 > > . . . . .	652
mln::Object< bkd_pixter1d< I > > . . . . .	771
mln::Iterator< bkd_pixter1d< I > > . . . . .	716
mln::Pixel_Iterator< bkd_pixter1d< I > > . . . . .	855
mln::Object< bkd_pixter2d< I > > . . . . .	771
mln::Iterator< bkd_pixter2d< I > > . . . . .	716
mln::Pixel_Iterator< bkd_pixter2d< I > > . . . . .	855
mln::Object< bkd_pixter3d< I > > . . . . .	771
mln::Iterator< bkd_pixter3d< I > > . . . . .	716
mln::Pixel_Iterator< bkd_pixter3d< I > > . . . . .	855
mln::Object< black_t > . . . . .	771
mln::Literal< black_t > . . . . .	727
mln::literal::black_t . . . . .	729
mln::Object< blue > . . . . .	771
mln::Meta_Function< blue > . . . . .	754
mln::Meta_Function_v2v< blue > . . . . .	754

mln::Object< blue_t > . . . . .	771
mln::Literal< blue_t > . . . . .	727
mln::literal::blue_t . . . . .	730
mln::Object< box< P > > . . . . .	771
mln::Site_Set< box< P > > . . . . .	879
mln::Box< box< P > > . . . . .	501
mln::box< P > . . . . .	494
mln::Object< box_runend_piter< P > > . . . . .	771
mln::Proxy< box_runend_piter< P > > . . . . .	870
mln::Site_Proxy< box_runend_piter< P > > . . . . .	878
mln::Site_Iterator< box_runend_piter< P > > . . . . .	876
site_iterator_base< box< P >, box_runend_piter< P > >	
site_set_iterator_base< box< P >, box_runend_piter< P > >	
mln::box_runend_piter< P > . . . . .	506
mln::Object< box_runstart_piter< P > > . . . . .	771
mln::Proxy< box_runstart_piter< P > > . . . . .	870
mln::Site_Proxy< box_runstart_piter< P > > . . . . .	878
mln::Site_Iterator< box_runstart_piter< P > > . . . . .	876
site_iterator_base< box< P >, box_runstart_piter< P > >	
site_set_iterator_base< box< P >, box_runstart_piter< P > >	
mln::box_runstart_piter< P > . . . . .	507
mln::Object< breadth_first_search_t > . . . . .	771
mln::Browsing< breadth_first_search_t > . . . . .	509
graph_first_search_t< breadth_first_search_t, std::queue >	
mln::canvas::browsing::breadth_first_search_t . . . . .	511
mln::Object< brown_t > . . . . .	771
mln::Literal< brown_t > . . . . .	727
mln::literal::brown_t . . . . .	730
mln::Object< card< I > > . . . . .	771
mln::Proxy< card< I > > . . . . .	870
mln::Accumulator< card< I > > . . . . .	485
base< unsigned, card< I > >	
mln::morpho::attribute::card< I > . . . . .	759
mln::Object< cast< V > > . . . . .	771
mln::Function< cast< V > > . . . . .	649
mln::Function_v2v< cast< V > > . . . . .	652
mln::Object< center > . . . . .	771
mln::Meta_Accumulator< center > . . . . .	752
mln::accu::meta::center . . . . .	411
mln::Object< center< P, V > > . . . . .	771
mln::Proxy< center< P, V > > . . . . .	870
mln::Accumulator< center< P, V > > . . . . .	485
base< V, center< P, V > >	
mln::accu::center< P, V > . . . . .	389
mln::Object< center_only_iter< D > > . . . . .	771
mln::Iterator< center_only_iter< D > > . . . . .	716
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 >	911
mln::Object< centered_bkd_iter_adapter< D, I >>	771
mln::Iterator< centered_bkd_iter_adapter< D, I >>	716
complex_relative_iterator_sequence< I, center_only_iter< D >, centered_bkd_iter_adapter< D, I >>	
mln::topo::centered_bkd_iter_adapter< D, I >	912
mln::Object< centered_fwd_iter_adapter< D, I >>	771
mln::Iterator< centered_fwd_iter_adapter< D, I >>	716
complex_relative_iterator_sequence< center_only_iter< D >, I, centered_fwd_iter_adapter< D, I >>	
mln::topo::centered_fwd_iter_adapter< D, I >	913
mln::Object< ch_function_value< F, V >>	771
mln::Function< ch_function_value< F, V >>	649
mln::Function_v2v< ch_function_value< F, V >>	652
mln::fun::v2v::ch_function_value< F, V >	613
mln::Object< ch_piter_image< I, Fwd >>	771
mln::Image< ch_piter_image< I, Fwd >>	694
mln::Object< chess >	771
mln::Function< chess >	649
mln::Function_v2v< chess >	652
mln::Function_v2b< chess >	651
mln::Object< col >	771
mln::Meta_Function< col >	754
mln::Meta_Function_v2v< col >	754
mln::Object< colorize >	771
mln::Function< colorize >	649
mln::Function_v2v< colorize >	652
mln::Object< comp >	771
mln::Meta_Function< comp >	754
mln::Meta_Function_v2v< comp >	754
mln::Object< comp_count >	771
mln::Meta_Function< comp_count >	754
mln::Meta_Function_v2v< comp_count >	754
mln::Object< complex_image< D, G, V >>	771
mln::Image< complex_image< D, G, V >>	694
mln::Object< complex_neighborhood_bkd_piter< I, G, N >>	771
mln::Proxy< complex_neighborhood_bkd_piter< I, G, N >>	870
mln::Site_Proxy< complex_neighborhood_bkd_piter< I, G, N >>	878
mln::Site_Iterator< complex_neighborhood_bkd_piter< I, G, N >>	876
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 >	526
mln::Object< complex_neighborhood_fwd_piter< I, G, N >>	771
mln::Proxy< complex_neighborhood_fwd_piter< I, G, N >>	870
mln::Site_Proxy< complex_neighborhood_fwd_piter< I, G, N >>	878
mln::Site_Iterator< complex_neighborhood_fwd_piter< I, G, N >>	876
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 > . . . . .	528
mln::Object< complex_psite< D, G > > . . . . .	771
mln::Proxy< complex_psite< D, G > > . . . . .	870
mln::Site_Proxy< complex_psite< D, G > > . . . . .	878
mln::Pseudo_Site< complex_psite< D, G > > . . . . .	871
pseudo_site_base_< const G::site &, complex_psite< D, G > >	
mln::complex_psite< D, G > . . . . .	530
mln::Object< complex_window_bkd_piter< I, G, W > > . . . . .	771
mln::Proxy< complex_window_bkd_piter< I, G, W > > . . . . .	870
mln::Site_Proxy< complex_window_bkd_piter< I, G, W > > . . . . .	878
mln::Site_Iterator< complex_window_bkd_piter< I, G, W > > . . . . .	876
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 > . . . . .	532
mln::Object< complex_window_fwd_piter< I, G, W > > . . . . .	771
mln::Proxy< complex_window_fwd_piter< I, G, W > > . . . . .	870
mln::Site_Proxy< complex_window_fwd_piter< I, G, W > > . . . . .	878
mln::Site_Iterator< complex_window_fwd_piter< I, G, W > > . . . . .	876
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 > . . . . .	534
mln::Object< component< T, i > > . . . . .	771
mln::Function< component< T, i > > . . . . .	649
mln::Function_v2v< component< T, i > > . . . . .	652
mln::fun::v2v::component< T, i > . . . . .	614
mln::Object< compose > . . . . .	771
mln::Meta_Function< compose > . . . . .	754
mln::Meta_Function_vv2v< compose > . . . . .	755
mln::Object< composition< mln::Meta_Function_v2v, F, mln::Meta_Function_v2v, G > > . . . . .	771
mln::Meta_Function< composition< mln::Meta_Function_v2v, F, mln::Meta_Function_v2v, G > > . . . . .	754
mln::Meta_Function_v2v< composition< mln::Meta_Function_v2v, F, mln::Meta_Function_v2v, G > > . . . . .	754
mln::Object< composition< mln::Meta_Function_v2v, F, mln::Meta_Function_vv2v, G > > . . . . .	771
mln::Meta_Function< composition< mln::Meta_Function_v2v, F, mln::Meta_Function_vv2v, G > > . . . . .	754
mln::Meta_Function_vv2v< composition< mln::Meta_Function_v2v, F, mln::Meta_Function_vv2v, G > > . . . . .	755
mln::Object< concrete > . . . . .	771
mln::Object< convert< V > > . . . . .	771
mln::Function< convert< V > > . . . . .	649
mln::Function_v2v< convert< V > > . . . . .	652
mln::Object< convolve< T1, T2, R > > . . . . .	771
mln::Proxy< convolve< T1, T2, R > > . . . . .	870
mln::Accumulator< convolve< T1, T2, R > > . . . . .	485
base< R, convolve< T1, T2, R > >	
mln::accu::convolve< T1, T2, R > . . . . .	390
mln::Object< cos > . . . . .	771
mln::Meta_Function< cos > . . . . .	754

mln::Meta_Function_v2v< cos > . . . . .	754
mln::Object< cos< V > > . . . . .	771
mln::Function< cos< V > > . . . . .	649
mln::Function_v2v< cos< V > > . . . . .	652
mln::fun::v2w2v::cos< V > . . . . .	619
mln::Object< count > . . . . .	771
mln::Meta_Accumulator< count > . . . . .	752
mln::accu::meta::math::count . . . . .	422
mln::Object< count< T > > . . . . .	771
mln::Proxy< count< T > > . . . . .	870
mln::Accumulator< count< T > > . . . . .	485
base< unsigned, count< T > >	
mln::accu::math::count< T > . . . . .	404
mln::Object< count_adjacent_vertices > . . . . .	771
mln::Meta_Accumulator< count_adjacent_vertices > . . . . .	752
mln::accu::meta::count_adjacent_vertices . . . . .	412
mln::Object< count_adjacent_vertices< F, S > > . . . . .	771
mln::Proxy< count_adjacent_vertices< F, S > > . . . . .	870
mln::Accumulator< count_adjacent_vertices< F, S > > . . . . .	485
base< unsigned, count_adjacent_vertices< F, S > >	
mln::accu::count_adjacent_vertices< F, S > . . . . .	392
mln::Object< count_adjacent_vertices< I > > . . . . .	771
mln::Proxy< count_adjacent_vertices< I > > . . . . .	870
mln::Accumulator< count_adjacent_vertices< I > > . . . . .	485
base< unsigned, count_adjacent_vertices< I > >	
mln::morpho::attribute::count_adjacent_vertices< I > . . . . .	761
mln::Object< count_labels > . . . . .	771
mln::Meta_Accumulator< count_labels > . . . . .	752
mln::accu::meta::count_labels . . . . .	413
mln::Object< count_labels< L > > . . . . .	771
mln::Proxy< count_labels< L > > . . . . .	870
mln::Accumulator< count_labels< L > > . . . . .	485
mln::Object< count_value > . . . . .	771
mln::Meta_Accumulator< count_value > . . . . .	752
mln::accu::meta::count_value . . . . .	414
mln::Object< count_value< V > > . . . . .	771
mln::Proxy< count_value< V > > . . . . .	870
mln::Accumulator< count_value< V > > . . . . .	485
base< unsigned, count_value< V > >	
mln::accu::count_value< V > . . . . .	393
mln::Object< couple< T, U > > . . . . .	771
mln::util::couple< T, U > . . . . .	953
mln::Object< cube > . . . . .	771
mln::Mesh< cube > . . . . .	751
mln::Regular_Grid< cube > . . . . .	873
mln::Object< cube3d > . . . . .	771
mln::Window< cube3d > . . . . .	1065
window_base< dpoint3d, cube3d >	

classical_window_base< dpoint3d, cube3d >	
mln::win::cube3d	1056
mln::Object< cuboid3d >	771
mln::Window< cuboid3d >	1065
window_base< dpoint3d, cuboid3d >	
classical_window_base< dpoint3d, cuboid3d >	
mln::win::cuboid3d	1057
mln::Object< cyan_t >	771
mln::Literal< cyan_t >	727
mln::literal::cyan_t	731
mln::Object< d_t >	771
mln::Function< d_t >	649
mln::Function_vv2v< d_t >	653
mln::Object< dark_gray_t >	771
mln::Literal< dark_gray_t >	727
mln::Object< dashed_line_f< I, dim > >	771
mln::Function< dashed_line_f< I, dim > >	649
mln::Function_v2v< dashed_line_f< I, dim > >	652
mln::Function_v2b< dashed_line_f< I, dim > >	651
mln::Object< decorated_image< I, D > >	771
mln::Image< decorated_image< I, D > >	694
mln::Object< depth1st_piter< T > >	771
mln::Proxy< depth1st_piter< T > >	870
mln::Site_Proxy< depth1st_piter< T > >	878
mln::Site_Iterator< depth1st_piter< T > >	876
mln::Object< depth_first_search_t >	771
mln::Browsing< depth_first_search_t >	509
graph_first_search_t< depth_first_search_t, std::stack >	
mln::canvas::browsing::depth_first_search_t	511
mln::Object< desc_propagation >	771
mln::Object< deviation >	771
mln::Meta_Accumulator< deviation >	752
mln::accu::stat::meta::deviation	468
mln::Object< deviation< T, S, M > >	771
mln::Proxy< deviation< T, S, M > >	870
mln::Accumulator< deviation< T, S, M > >	485
base< M, deviation< T, S, M > >	
mln::accu::stat::deviation< T, S, M >	459
mln::Object< diag2d >	771
mln::Window< diag2d >	1065
window_base< dpoint2d, diag2d >	
classical_window_base< dpoint2d, diag2d >	
mln::win::diag2d	1059
mln::Object< diagonal2d_t >	771
mln::Browsing< diagonal2d_t >	509
mln::canvas::browsing::diagonal2d_t	511
mln::Object< diff_abs< V > >	771
mln::Function< diff_abs< V > >	649
mln::Function_vv2v< diff_abs< V > >	653

mln::fun::vv2v::diff_abs< V > . . . . .	629
mln::Object< dir_struct_elt_incr_update_t > . . . . .	771
mln::Browsing< dir_struct_elt_incr_update_t > . . . . .	509
mln::canvas::browsing::dir_struct_elt_incr_update_t . . . . .	513
mln::Object< directional_t > . . . . .	771
mln::Browsing< directional_t > . . . . .	509
mln::canvas::browsing::directional_t . . . . .	514
mln::Object< dist > . . . . .	771
mln::Function< dist > . . . . .	649
mln::Function_vv2v< dist > . . . . .	653
mln::Object< dist_t > . . . . .	771
mln::Function< dist_t > . . . . .	649
mln::Function_vv2v< dist_t > . . . . .	653
mln::Object< dn_leaf_piter< T > > . . . . .	771
mln::Proxy< dn_leaf_piter< T > > . . . . .	870
mln::Site_Proxy< dn_leaf_piter< T > > . . . . .	878
mln::Site_Iterator< dn_leaf_piter< T > > . . . . .	876
mln::Object< dn_node_piter< T > > . . . . .	771
mln::Proxy< dn_node_piter< T > > . . . . .	870
mln::Site_Proxy< dn_node_piter< T > > . . . . .	878
mln::Site_Iterator< dn_node_piter< T > > . . . . .	876
mln::Object< dn_site_piter< T > > . . . . .	771
mln::Proxy< dn_site_piter< T > > . . . . .	870
mln::Site_Proxy< dn_site_piter< T > > . . . . .	878
mln::Site_Iterator< dn_site_piter< T > > . . . . .	876
mln::Object< dpoint< G, C > > . . . . .	771
mln::Gdpoint< dpoint< G, C > > . . . . .	658
mln::dpoint< G, C > . . . . .	581
mln::Object< dpoints_bkd_pixter< I > > . . . . .	771
mln::Iterator< dpoints_bkd_pixter< I > > . . . . .	716
mln::Pixel_Iterator< dpoints_bkd_pixter< I > > . . . . .	855
mln::Object< dpoints_fwd_pixter< I > > . . . . .	771
mln::Iterator< dpoints_fwd_pixter< I > > . . . . .	716
mln::Pixel_Iterator< dpoints_fwd_pixter< I > > . . . . .	855
mln::Object< dpsites_bkd_piter< V > > . . . . .	771
mln::Proxy< dpsites_bkd_piter< V > > . . . . .	870
mln::Site_Proxy< dpsites_bkd_piter< V > > . . . . .	878
mln::Site_Iterator< dpsites_bkd_piter< V > > . . . . .	876
site_iterator_base< V, dpsites_bkd_piter< V > >	
site_relative_iterator_base< V, dpsites_bkd_piter< V > >	
mln::dpsites_bkd_piter< V > . . . . .	590
mln::Object< dpsites_fwd_piter< V > > . . . . .	771
mln::Proxy< dpsites_fwd_piter< V > > . . . . .	870
mln::Site_Proxy< dpsites_fwd_piter< V > > . . . . .	878
mln::Site_Iterator< dpsites_fwd_piter< V > > . . . . .	876
site_iterator_base< V, dpsites_fwd_piter< V > >	
site_relative_iterator_base< V, dpsites_fwd_piter< V > >	
mln::dpsites_fwd_piter< V > . . . . .	592



mln::Object< eat > . . . . .	771
mln::util::eat . . . . .	955
mln::Object< edge_bkd_iterator< G > > . . . . .	771
mln::Proxy< edge_bkd_iterator< G > > . . . . .	870
mln::Object< edge_fwd_iterator< G > > . . . . .	771
mln::Proxy< edge_fwd_iterator< G > > . . . . .	870
mln::Object< edge_image< P, V, G > > . . . . .	771
mln::Image< edge_image< P, V, G > > . . . . .	694
mln::Object< edge_nbh_edge_bkd_iterator< G > > . . . . .	771
mln::Proxy< edge_nbh_edge_bkd_iterator< G > > . . . . .	870
mln::Object< edge_nbh_edge_fwd_iterator< G > > . . . . .	771
mln::Proxy< edge_nbh_edge_fwd_iterator< G > > . . . . .	870
mln::Object< edge_to_color< I, V > > . . . . .	771
mln::Function< edge_to_color< I, V > > . . . . .	649
mln::Function_v2v< edge_to_color< I, V > > . . . . .	652
mln::Object< enc< V > > . . . . .	771
mln::Function< enc< V > > . . . . .	649
mln::Function_v2v< enc< V > > . . . . .	652
mln::Object< eq< L, R > > . . . . .	771
mln::Function< eq< L, R > > . . . . .	649
mln::Function_vv2b< eq< L, R > > . . . . .	653
mln::fun::vv2b::eq< L, R > . . . . .	623
mln::Object< extended< I > > . . . . .	771
mln::Image< extended< I > > . . . . .	694
mln::Object< extension_fun< I, F > > . . . . .	771
mln::Image< extension_fun< I, F > > . . . . .	694
mln::Object< extension_ima< I, J > > . . . . .	771
mln::Image< extension_ima< I, J > > . . . . .	694
mln::Object< extension_val< I > > . . . . .	771
mln::Image< extension_val< I > > . . . . .	694
mln::Object< f_16_to_8 > . . . . .	771
mln::Function< f_16_to_8 > . . . . .	649
mln::Function_v2v< f_16_to_8 > . . . . .	652
mln::Object< f_box1d_t > . . . . .	771
mln::Function< f_box1d_t > . . . . .	649
mln::Function_v2v< f_box1d_t > . . . . .	652
mln::Function_v2b< f_box1d_t > . . . . .	651
mln::Object< f_box2d_t > . . . . .	771
mln::Function< f_box2d_t > . . . . .	649
mln::Function_v2v< f_box2d_t > . . . . .	652
mln::Function_v2b< f_box2d_t > . . . . .	651
mln::Object< f_box3d_t > . . . . .	771
mln::Function< f_box3d_t > . . . . .	649
mln::Function_v2v< f_box3d_t > . . . . .	652
mln::Function_v2b< f_box3d_t > . . . . .	651
mln::Object< f_hsi_to_rgb_< T_rgb > > . . . . .	771
mln::Function< f_hsi_to_rgb_< T_rgb > > . . . . .	649

mln::Function_v2v< f_hsi_to_rgb_< T_rgb > > . . . . .	652
mln::Object< f_hsl_to_rgb_< T_rgb > > . . . . .	771
mln::Function< f_hsl_to_rgb_< T_rgb > > . . . . .	649
mln::Function_v2v< f_hsl_to_rgb_< T_rgb > > . . . . .	652
mln::Object< f_rgb_to_hsi_< T_hsi > > . . . . .	771
mln::Function< f_rgb_to_hsi_< T_hsi > > . . . . .	649
mln::Function_v2v< f_rgb_to_hsi_< T_hsi > > . . . . .	652
mln::Object< f_rgb_to_hsl_< T_hsl > > . . . . .	771
mln::Function< f_rgb_to_hsl_< T_hsl > > . . . . .	649
mln::Function_v2v< f_rgb_to_hsl_< T_hsl > > . . . . .	652
mln::Object< face_bkd_iter< D > > . . . . .	771
mln::Iterator< face_bkd_iter< D > > . . . . .	716
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 > . . . . .	920
mln::Object< face_fwd_iter< D > > . . . . .	771
mln::Iterator< face_fwd_iter< D > > . . . . .	716
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 > . . . . .	922
mln::Object< faces_psite< N, D, P > > . . . . .	771
mln::Proxy< faces_psite< N, D, P > > . . . . .	870
mln::Site_Proxy< faces_psite< N, D, P > > . . . . .	878
mln::Pseudo_Site< faces_psite< N, D, P > > . . . . .	871
pseudo_site_base_< const P &, faces_psite< N, D, P > >	
mln::faces_psite< N, D, P > . . . . .	603
mln::Object< fibonacci_heap< P, T > > . . . . .	771
mln::util::fibonacci_heap< P, T > . . . . .	960
mln::Object< flat_image< T, S > > . . . . .	771
mln::Image< flat_image< T, S > > . . . . .	694
mln::Object< float01 > . . . . .	771
mln::Value< float01 > . . . . .	1008
mln::Object< float01_f > . . . . .	771
mln::Value< float01_f > . . . . .	1008
mln::Object< float_off_loader > . . . . .	771
mln::Object< float_off_saver > . . . . .	771
mln::Object< fold< P, dir_0, dir_1, dir_2 > > . . . . .	771
mln::Function< fold< P, dir_0, dir_1, dir_2 > > . . . . .	649
mln::Function_v2v< fold< P, dir_0, dir_1, dir_2 > > . . . . .	652
mln::Object< from_accu< A > > . . . . .	771
mln::Meta_Function< from_accu< A > > . . . . .	754
mln::Meta_Function_v2v< from_accu< A > > . . . . .	754
mln::Object< fun_image< F, I > > . . . . .	771
mln::Image< fun_image< F, I > > . . . . .	694
mln::Object< function< meta::blue< value::rgb< n > > > > . . . . .	771
mln::Function< function< meta::blue< value::rgb< n > > > > . . . . .	649
mln::Function_v2v< function< meta::blue< value::rgb< n > > > > . . . . .	652
mln::Object< function< meta::first< util::couple< T, U > > > > . . . . .	771

mln::Function< function< meta::first< util::couple< T, U > > > > . . . . .	649
mln::Function_v2v< function< meta::first< util::couple< T, U > > > > . . . . .	652
mln::Object< function< meta::green< value::rgb< n > > > > . . . . .	771
mln::Function< function< meta::green< value::rgb< n > > > > . . . . .	649
mln::Function_v2v< function< meta::green< value::rgb< n > > > > . . . . .	652
mln::Object< function< meta::red< value::rgb< n > > > > . . . . .	771
mln::Function< function< meta::red< value::rgb< n > > > > . . . . .	649
mln::Function_v2v< function< meta::red< value::rgb< n > > > > . . . . .	652
mln::Object< function< meta::second< util::couple< T, U > > > > . . . . .	771
mln::Function< function< meta::second< util::couple< T, U > > > > . . . . .	649
mln::Function_v2v< function< meta::second< util::couple< T, U > > > > . . . . .	652
mln::Object< function< meta::to_enc< T > > > . . . . .	771
mln::Function< function< meta::to_enc< T > > > . . . . .	649
mln::Function_v2v< function< meta::to_enc< T > > > . . . . .	652
mln::Object< fwd_pixter1d< I > > . . . . .	771
mln::Iterator< fwd_pixter1d< I > > . . . . .	716
mln::Pixel_Iterator< fwd_pixter1d< I > > . . . . .	855
mln::Object< fwd_pixter2d< I > > . . . . .	771
mln::Iterator< fwd_pixter2d< I > > . . . . .	716
mln::Pixel_Iterator< fwd_pixter2d< I > > . . . . .	855
mln::Object< fwd_pixter3d< I > > . . . . .	771
mln::Iterator< fwd_pixter3d< I > > . . . . .	716
mln::Pixel_Iterator< fwd_pixter3d< I > > . . . . .	855
mln::Object< fwd_t > . . . . .	771
mln::Browsing< fwd_t > . . . . .	509
mln::canvas::browsing::fwd_t . . . . .	516
mln::Object< ge< L, R > > . . . . .	771
mln::Function< ge< L, R > > . . . . .	649
mln::Function_vv2b< ge< L, R > > . . . . .	653
mln::fun::vv2b::ge< L, R > . . . . .	624
mln::Object< graph > . . . . .	771
mln::Graph< graph > . . . . .	666
graph_base< graph > . . . . .	
mln::util::graph . . . . .	963
mln::Object< graph_elt_mixed_window< G, S, S2 > > . . . . .	771
mln::Window< graph_elt_mixed_window< G, S, S2 > > . . . . .	1065
mln::graph_window_base< S2::fun_t::result, graph_elt_mixed_window< G, S, S2 > > . . . . .	684
mln::graph_elt_mixed_window< G, S, S2 > . . . . .	669
mln::Object< graph_elt_window< G, S > > . . . . .	771
mln::Window< graph_elt_window< G, S > > . . . . .	1065
mln::graph_window_base< S::fun_t::result, graph_elt_window< G, S > > . . . . .	684
mln::graph_elt_window< G, S > . . . . .	676
mln::Object< graph_elt_window_if< G, S, I > > . . . . .	771
mln::Window< graph_elt_window_if< G, S, I > > . . . . .	1065
mln::graph_window_base< S::fun_t::result, graph_elt_window_if< G, S, I > > . . . . .	684
mln::graph_elt_window_if< G, S, I > . . . . .	679
mln::Object< graph_window_if_piter< S, W, I > > . . . . .	771
mln::Proxy< graph_window_if_piter< S, W, I > > . . . . .	870

mln::Site_Proxy< graph_window_if_piter< S, W, I > > . . . . .	878
mln::Site_Iterator< graph_window_if_piter< S, W, I > > . . . . .	876
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 > . . . . .	686
mln::Object< graph_window_piter< S, W, I > > . . . . .	771
mln::Proxy< graph_window_piter< S, W, I > > . . . . .	870
mln::Site_Proxy< graph_window_piter< S, W, I > > . . . . .	878
mln::Site_Iterator< graph_window_piter< S, W, I > > . . . . .	876
site_iterator_base< W, graph_window_piter< S, W, I > >	
site_relative_iterator_base< W, graph_window_piter< S, W, I >, W::center_t >	
mln::graph_window_piter< S, W, I > . . . . .	687
mln::Object< gray_f > . . . . .	771
mln::Value< gray_f > . . . . .	1008
mln::Object< graylevel< n > > . . . . .	771
mln::Value< graylevel< n > > . . . . .	1008
mln::Object< graylevel_f > . . . . .	771
mln::Value< graylevel_f > . . . . .	1008
mln::Object< green > . . . . .	771
mln::Meta_Function< green > . . . . .	754
mln::Meta_Function_v2v< green > . . . . .	754
mln::Object< green_t > . . . . .	771
mln::Literal< green_t > . . . . .	727
mln::literal::green_t . . . . .	732
mln::Object< gt< L, R > > . . . . .	771
mln::Function< gt< L, R > > . . . . .	649
mln::Function_vv2b< gt< L, R > > . . . . .	653
mln::fun::vv2b::gt< L, R > . . . . .	625
mln::Object< has< I > > . . . . .	771
mln::Function< has< I > > . . . . .	649
mln::Function_v2v< has< I > > . . . . .	652
mln::Function_v2b< has< I > > . . . . .	651
mln::Object< height > . . . . .	771
mln::Meta_Accumulator< height > . . . . .	752
mln::accu::meta::shape::height . . . . .	432
mln::Object< height< I > > . . . . .	771
mln::Proxy< height< I > > . . . . .	870
mln::Accumulator< height< I > >	
base< unsigned, height< I > >	
mln::accu::shape::height< I > . . . . .	453
mln::morpho::attribute::height< I > . . . . .	762
mln::Object< hexa > . . . . .	771
mln::Mesh< hexa > . . . . .	751
mln::Regular_Grid< hexa > . . . . .	873
mln::Object< hexa< I > > . . . . .	771
mln::Image< hexa< I > > . . . . .	694
mln::Object< hexa< image2d< V > > > . . . . .	771
mln::Image< hexa< image2d< V > > > . . . . .	694

mln::Object< histo > . . . . .	771
mln::Meta_Accumulator< histo > . . . . .	752
mln::accu::meta::histo . . . . .	415
mln::Object< histo< V > > . . . . .	771
mln::Proxy< histo< V > > . . . . .	870
mln::Accumulator< histo< V > > . . . . .	485
base< const std::vector< unsigned > &, histo< V > >	
mln::accu::histo< V > . . . . .	395
mln::Object< hyper_directional_t > . . . . .	771
mln::Browsing< hyper_directional_t > . . . . .	509
mln::canvas::browsing::hyper_directional_t . . . . .	517
mln::Object< id2element< G, Elt > > . . . . .	771
mln::Function< id2element< G, Elt > > . . . . .	649
mln::Function_v2v< id2element< G, Elt > > . . . . .	652
mln::Object< identity_t > . . . . .	771
mln::Literal< identity_t > . . . . .	727
mln::literal::identity_t . . . . .	733
mln::Object< ignore > . . . . .	771
mln::util::ignore . . . . .	971
mln::Object< image1d< T > > . . . . .	771
mln::Image< image1d< T > > . . . . .	694
mln::Object< image2d< T > > . . . . .	771
mln::Image< image2d< T > > . . . . .	694
mln::Object< image3d< T > > . . . . .	771
mln::Image< image3d< T > > . . . . .	694
mln::Object< image< F, S > > . . . . .	771
mln::Image< image< F, S > > . . . . .	694
mln::Object< image_if< I, F > > . . . . .	771
mln::Image< image_if< I, F > > . . . . .	694
mln::Object< implies< L, R > > . . . . .	771
mln::Function< implies< L, R > > . . . . .	649
mln::Function_vv2b< implies< L, R > > . . . . .	653
mln::fun::vv2b::implies< L, R > . . . . .	626
mln::Object< index_of_value< bool > > . . . . .	771
mln::Function< index_of_value< bool > > . . . . .	649
mln::Function_v2v< index_of_value< bool > > . . . . .	652
mln::Object< index_of_value< T > > . . . . .	771
mln::Function< index_of_value< T > > . . . . .	649
mln::Function_v2v< index_of_value< T > > . . . . .	652
mln::Object< inf > . . . . .	771
mln::Meta_Accumulator< inf > . . . . .	752
mln::accu::meta::math::inf . . . . .	423
mln::Meta_Function< inf > . . . . .	754
mln::Meta_Function_vv2v< inf > . . . . .	755
mln::Object< inf< T > > . . . . .	771
mln::Proxy< inf< T > > . . . . .	870
mln::Accumulator< inf< T > > . . . . .	485
base< const T &, inf< T > >	

mln::accu::math::inf< T > . . . . .	406
mln::Object< int_s< n > > . . . . .	771
mln::Value< int_s< n > > . . . . .	1008
mln::Object< int_u8_off_saver > . . . . .	771
mln::Object< int_u< n > > . . . . .	771
mln::Value< int_u< n > > . . . . .	1008
mln::Object< int_u_sat< n > > . . . . .	771
mln::Value< int_u_sat< n > > . . . . .	1008
mln::Object< interpolated< I, F > > . . . . .	771
mln::Image< interpolated< I, F > > . . . . .	694
mln::Object< iota > . . . . .	771
mln::Function< iota > . . . . .	649
mln::Function_v2v< iota > . . . . .	652
mln::Object< is_dot > . . . . .	771
mln::Function< is_dot > . . . . .	649
mln::Function_v2v< is_dot > . . . . .	652
mln::Function_v2b< is_dot > . . . . .	651
mln::Object< is_edge > . . . . .	771
mln::Function< is_edge > . . . . .	649
mln::Function_v2v< is_edge > . . . . .	652
mln::Function_v2b< is_edge > . . . . .	651
mln::Object< is_n_face< N > > . . . . .	771
mln::Function< is_n_face< N > > . . . . .	649
mln::Function_v2v< is_n_face< N > > . . . . .	652
mln::Function_v2b< is_n_face< N > > . . . . .	651
mln::topo::is_n_face< N > . . . . .	923
mln::Object< is_pixel > . . . . .	771
mln::Function< is_pixel > . . . . .	649
mln::Function_v2v< is_pixel > . . . . .	652
mln::Function_v2b< is_pixel > . . . . .	651
mln::Object< is_row_odd > . . . . .	771
mln::Function< is_row_odd > . . . . .	649
mln::Function_v2v< is_row_odd > . . . . .	652
mln::Function_v2b< is_row_odd > . . . . .	651
mln::Object< is_separator > . . . . .	771
mln::Function< is_separator > . . . . .	649
mln::Function_v2v< is_separator > . . . . .	652
mln::Function_v2b< is_separator > . . . . .	651
mln::world::inter_pixel::is_separator . . . . .	1069
mln::Object< is_simple_cell< I > > . . . . .	771
mln::Function< is_simple_cell< I > > . . . . .	649
mln::Function_v2v< is_simple_cell< I > > . . . . .	652
mln::Function_v2b< is_simple_cell< I > > . . . . .	651
mln::topo::is_simple_cell< I > . . . . .	924
mln::Object< ithcomp > . . . . .	771
mln::Meta_Function< ithcomp > . . . . .	754
mln::Meta_Function_vv2v< ithcomp > . . . . .	755
mln::Object< keep_specific_colors > . . . . .	771

mln::Function< keep_specific_colors > . . . . .	649
mln::Function_v2v< keep_specific_colors > . . . . .	652
mln::Function_v2b< keep_specific_colors > . . . . .	651
mln::Object< l1 > . . . . .	771
mln::Meta_Function< l1 > . . . . .	754
mln::Meta_Function_v2v< l1 > . . . . .	754
mln::Object< l1_norm< V > > . . . . .	771
mln::Function< l1_norm< V > > . . . . .	649
mln::Function_v2v< l1_norm< V > > . . . . .	652
mln::Object< l1_norm< V, R > > . . . . .	771
mln::Function< l1_norm< V, R > > . . . . .	649
mln::Function_v2v< l1_norm< V, R > > . . . . .	652
mln::fun::v2v::l1_norm< V, R > . . . . .	615
mln::fun::v2w_w2v::l1_norm< V, R > . . . . .	620
mln::Object< l2 > . . . . .	771
mln::Meta_Function< l2 > . . . . .	754
mln::Meta_Function_v2v< l2 > . . . . .	754
mln::Object< l2_norm< V, R > > . . . . .	771
mln::Function< l2_norm< V, R > > . . . . .	649
mln::Function_v2v< l2_norm< V, R > > . . . . .	652
mln::fun::v2v::l2_norm< V, R > . . . . .	616
mln::fun::v2w_w2v::l2_norm< V, R > . . . . .	621
mln::Object< label< n > > . . . . .	771
mln::Value< label< n > > . . . . .	1008
mln::Object< label_used > . . . . .	771
mln::Meta_Accumulator< label_used > . . . . .	752
mln::accu::meta::label_used . . . . .	416
mln::Object< label_used< L > > . . . . .	771
mln::Proxy< label_used< L > > . . . . .	870
mln::Accumulator< label_used< L > > . . . . .	485
base< const fun::i2v::array< bool > &, label_used< L > >	
mln::accu::label_used< L > . . . . .	396
mln::Object< labeled_image< I > > . . . . .	771
mln::Image< labeled_image< I > > . . . . .	694
mln::Object< land > . . . . .	771
mln::Meta_Accumulator< land > . . . . .	752
mln::accu::meta::logic::land . . . . .	417
mln::Proxy< land > . . . . .	870
mln::Accumulator< land > . . . . .	485
base< bool, land >	
mln::accu::logic::land . . . . .	398
mln::Object< land< L, R > > . . . . .	771
mln::Function< land< L, R > > . . . . .	649
mln::Function_vv2v< land< L, R > > . . . . .	653
mln::fun::vv2v::land< L, R > . . . . .	630
mln::Object< land_basic > . . . . .	771
mln::Meta_Accumulator< land_basic > . . . . .	752
mln::accu::meta::logic::land_basic . . . . .	418

mln::Proxy< land_basic > . . . . .	870
mln::Accumulator< land_basic > . . . . .	485
base< bool, land_basic >	
mln::accu::logic::land_basic . . . . .	399
mln::Object< land_not< L, R > > . . . . .	771
mln::Function< land_not< L, R > > . . . . .	649
mln::Function_vv2v< land_not< L, R > > . . . . .	653
mln::fun::vv2v::land_not< L, R > . . . . .	631
mln::Object< lazy_image< I, F, B > > . . . . .	771
mln::Image< lazy_image< I, F, B > > . . . . .	694
mln::Object< le< L, R > > . . . . .	771
mln::Function< le< L, R > > . . . . .	649
mln::Function_vv2b< le< L, R > > . . . . .	653
mln::fun::vv2b::le< L, R > . . . . .	627
mln::Object< light_gray_t > . . . . .	771
mln::Literal< light_gray_t > . . . . .	727
mln::literal::light_gray_t . . . . .	734
mln::Object< lime_t > . . . . .	771
mln::Literal< lime_t > . . . . .	727
mln::literal::lime_t . . . . .	735
mln::Object< line< M, i, C > > . . . . .	771
mln::Window< line< M, i, C > > . . . . .	1065
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 > . . . . .	1060
mln::Object< line_graph< G > > . . . . .	771
mln::Graph< line_graph< G > > . . . . .	666
graph_base< line_graph< G > >	
mln::util::line_graph< G > . . . . .	971
mln::Object< linear< V, T, R > > . . . . .	771
mln::Function< linear< V, T, R > > . . . . .	649
mln::Function_v2v< linear< V, T, R > > . . . . .	652
mln::fun::v2v::linear< V, T, R > . . . . .	617
mln::Object< linear_sat< V, T, R > > . . . . .	771
mln::Function< linear_sat< V, T, R > > . . . . .	649
mln::Function_v2v< linear_sat< V, T, R > > . . . . .	652
mln::Object< linfty > . . . . .	771
mln::Meta_Function< linfty > . . . . .	754
mln::Meta_Function_v2v< linfty > . . . . .	754
mln::Object< linfty_norm< V, R > > . . . . .	771
mln::Function< linfty_norm< V, R > > . . . . .	649
mln::Function_v2v< linfty_norm< V, R > > . . . . .	652
mln::fun::v2v::linfty_norm< V, R > . . . . .	618
mln::fun::v2w_w2v::linfty_norm< V, R > . . . . .	622
mln::Object< lnot< V > > . . . . .	771
mln::Function< lnot< V > > . . . . .	649
mln::Function_v2v< lnot< V > > . . . . .	652
mln::Function_v2b< lnot< V > > . . . . .	651
mln::fun::v2b::lnot< V > . . . . .	611



mln::Object< lor > . . . . .	771
mln::Meta_Accumulator< lor > . . . . .	752
mln::accu::meta::logic::lor . . . . .	419
mln::Proxy< lor > . . . . .	870
mln::Accumulator< lor > . . . . .	485
base< bool, lor >	
mln::accu::logic::lor . . . . .	400
mln::Object< lor< L, R > > . . . . .	771
mln::Function< lor< L, R > > . . . . .	649
mln::Function_vv2v< lor< L, R > > . . . . .	653
mln::fun::vv2v::lor< L, R > . . . . .	632
mln::Object< lor_basic > . . . . .	771
mln::Meta_Accumulator< lor_basic > . . . . .	752
mln::accu::meta::logic::lor_basic . . . . .	420
mln::Proxy< lor_basic > . . . . .	870
mln::Accumulator< lor_basic > . . . . .	485
base< bool, lor_basic >	
mln::accu::logic::lor_basic . . . . .	402
mln::Object< lt< L, R > > . . . . .	771
mln::Function< lt< L, R > > . . . . .	649
mln::Function_vv2b< lt< L, R > > . . . . .	653
mln::fun::vv2b::lt< L, R > . . . . .	628
mln::Object< lut_vec< S, T > > . . . . .	771
mln::Value_Set< lut_vec< S, T > > . . . . .	1044
mln::value::lut_vec< S, T > . . . . .	1028
mln::Object< lxor< L, R > > . . . . .	771
mln::Function< lxor< L, R > > . . . . .	649
mln::Function_vv2v< lxor< L, R > > . . . . .	653
mln::fun::vv2v::lxor< L, R > . . . . .	633
mln::Object< magenta_t > . . . . .	771
mln::Literal< magenta_t > . . . . .	727
mln::literal::magenta_t . . . . .	736
mln::Object< mahalanobis< V > > . . . . .	771
mln::Function< mahalanobis< V > > . . . . .	649
mln::Function_v2v< mahalanobis< V > > . . . . .	652
mln::Object< maj_h > . . . . .	771
mln::Meta_Accumulator< maj_h > . . . . .	752
mln::accu::meta::maj_h . . . . .	421
mln::Object< maj_h< T > > . . . . .	771
mln::Proxy< maj_h< T > > . . . . .	870
mln::Accumulator< maj_h< T > > . . . . .	485
base< const T &, maj_h< T > >	
mln::accu::maj_h< T > . . . . .	403
mln::Object< mat< n, m, T > > . . . . .	771
mln::Object< max > . . . . .	771
mln::Meta_Accumulator< max > . . . . .	752
mln::accu::meta::stat::max . . . . .	434
mln::Object< max< T > > . . . . .	771
mln::Proxy< max< T > > . . . . .	870

mln::Accumulator< max< T > > . . . . .	485
base< const T &, max< T > >	
mln::accu::stat::max< T > . . . . .	460
mln::Object< max< V > > . . . . .	771
mln::Function< max< V > > . . . . .	649
mln::Function_vv2v< max< V > > . . . . .	653
mln::fun::vv2v::max< V > . . . . .	634
mln::Object< max_h > . . . . .	771
mln::Meta_Accumulator< max_h > . . . . .	752
mln::accu::meta::stat::max_h . . . . .	435
mln::Object< max_h< V > > . . . . .	771
mln::Proxy< max_h< V > > . . . . .	870
mln::Accumulator< max_h< V > > . . . . .	485
base< const V &, max_h< V > >	
mln::accu::stat::max_h< V > . . . . .	461
mln::Object< max_site > . . . . .	771
mln::Meta_Accumulator< max_site > . . . . .	752
mln::accu::meta::max_site . . . . .	426
mln::Object< max_site< I > > . . . . .	771
mln::Proxy< max_site< I > > . . . . .	870
mln::Accumulator< max_site< I > > . . . . .	485
base< I::psite, max_site< I > >	
mln::accu::max_site< I > . . . . .	410
mln::Object< max_t > . . . . .	771
mln::Literal< max_t > . . . . .	727
mln::literal::max_t . . . . .	737
mln::Object< mean > . . . . .	771
mln::Meta_Accumulator< mean > . . . . .	752
mln::accu::meta::stat::mean . . . . .	436
mln::Meta_Function< mean > . . . . .	754
mln::Meta_Function_v2v< mean > . . . . .	754
mln::Object< mean< T, S, M > > . . . . .	771
mln::Proxy< mean< T, S, M > > . . . . .	870
mln::Accumulator< mean< T, S, M > > . . . . .	485
base< M, mean< T, S, M > >	
mln::accu::stat::mean< T, S, M > . . . . .	463
mln::Object< median_alt< S > > . . . . .	771
mln::Proxy< median_alt< S > > . . . . .	870
mln::Accumulator< median_alt< S > > . . . . .	485
base< const S::value &, median_alt< S > >	
mln::accu::stat::median_alt< S > . . . . .	464
mln::Object< median_alt< T > > . . . . .	771
mln::Meta_Accumulator< median_alt< T > > . . . . .	752
mln::accu::meta::stat::median_alt< T > . . . . .	437
mln::Object< median_alt< value::set< T > > > . . . . .	771
mln::Proxy< median_alt< value::set< T > > > . . . . .	870
mln::Accumulator< median_alt< value::set< T > > > . . . . .	485
base< const value::set< T >::value &, median_alt< value::set< T > > >	
mln::accu::stat::median_alt< value::set< T > > > . . . . .	464

mln::Object< median_h > . . . . .	771
mln::Meta_Accumulator< median_h > . . . . .	752
mln::accu::meta::stat::median_h . . . . .	438
mln::Object< median_h< V > > . . . . .	771
mln::Proxy< median_h< V > > . . . . .	870
mln::Accumulator< median_h< V > > . . . . .	485
base< const V &, median_h< V > >	
mln::accu::stat::median_h< V > . . . . .	466
mln::Object< medium_gray_t > . . . . .	771
mln::Literal< medium_gray_t > . . . . .	727
mln::Object< min > . . . . .	771
mln::Meta_Accumulator< min > . . . . .	752
mln::accu::meta::stat::min . . . . .	439
mln::Object< min< L, R > > . . . . .	771
mln::Function< min< L, R > > . . . . .	649
mln::Function_vv2v< min< L, R > > . . . . .	653
mln::fun::vv2v::min< L, R > . . . . .	635
mln::Object< min< T > > . . . . .	771
mln::Proxy< min< T > > . . . . .	870
mln::Accumulator< min< T > > . . . . .	485
base< const T &, min< T > >	
mln::accu::stat::min< T > . . . . .	468
mln::Object< min_h > . . . . .	771
mln::Meta_Accumulator< min_h > . . . . .	752
mln::accu::meta::stat::min_h . . . . .	440
mln::Object< min_h< V > > . . . . .	771
mln::Proxy< min_h< V > > . . . . .	870
mln::Accumulator< min_h< V > > . . . . .	485
base< const V &, min_h< V > >	
mln::accu::stat::min_h< V > . . . . .	470
mln::Object< min_t > . . . . .	771
mln::Literal< min_t > . . . . .	727
mln::literal::min_t . . . . .	738
mln::Object< mirror< B > > . . . . .	771
mln::Function< mirror< B > > . . . . .	649
mln::Function_v2v< mirror< B > > . . . . .	652
mln::Object< mixed_neighb< W > > . . . . .	771
mln::Neighborhood< mixed_neighb< W > > . . . . .	770
mln::Object< mln::util::set< T > > . . . . .	771
mln::util::set< T > . . . . .	984
mln::Object< multi_site< P > > . . . . .	771
mln::Object< multiple< W, F > > . . . . .	771
mln::Window< multiple< W, F > > . . . . .	1065
window_base< W::dpsite, multiple< W, F > >	
mln::win::multiple< W, F > . . . . .	1061
mln::Object< multiple_qiter< W, F > > . . . . .	771
mln::Proxy< multiple_qiter< W, F > > . . . . .	870
mln::Site_Proxy< multiple_qiter< W, F > > . . . . .	878

mln::Site_Iterator< multiple_qiter< W, F > > . . . . .	876
mln::Object< multiple_size< n, W, F > > . . . . .	771
mln::Window< multiple_size< n, W, F > > . . . . .	1065
window_base< W::dpsite, multiple_size< n, W, F > >	
mln::win::multiple_size< n, W, F > . . . . .	1062
mln::Object< multiple_size_qiter< n, W, F > > . . . . .	771
mln::Proxy< multiple_size_qiter< n, W, F > > . . . . .	870
mln::Site_Proxy< multiple_size_qiter< n, W, F > > . . . . .	878
mln::Site_Iterator< multiple_size_qiter< n, W, F > > . . . . .	876
mln::Object< my_box2d > . . . . .	771
mln::Function< my_box2d > . . . . .	649
mln::Function_v2v< my_box2d > . . . . .	652
mln::Function_v2b< my_box2d > . . . . .	651
mln::Object< my_ext > . . . . .	771
mln::Function< my_ext > . . . . .	649
mln::Function_v2v< my_ext > . . . . .	652
mln::Object< my_fun< G > > . . . . .	771
mln::Function< my_fun< G > > . . . . .	649
mln::Object< my_image2d< T > > . . . . .	771
mln::Image< my_image2d< T > > . . . . .	694
mln::Object< my_values_t > . . . . .	771
mln::Function< my_values_t > . . . . .	649
mln::Function_v2v< my_values_t > . . . . .	652
mln::Object< myfun > . . . . .	771
mln::Function< myfun > . . . . .	649
mln::Function_vv2v< myfun > . . . . .	653
mln::Object< mysqrt > . . . . .	771
mln::Function< mysqrt > . . . . .	649
mln::Function_v2v< mysqrt > . . . . .	652
mln::Object< n_face_bkd_iter< D > > . . . . .	771
mln::Iterator< n_face_bkd_iter< D > > . . . . .	716
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 > . . . . .	930
mln::Object< n_face_fwd_iter< D > > . . . . .	771
mln::Iterator< n_face_fwd_iter< D > > . . . . .	716
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 > . . . . .	932
mln::Object< neighb< graph_elt_mixed_window< G, S, S2 > > > . . . . .	771
mln::Neighborhood< neighb< graph_elt_mixed_window< G, S, S2 > > > . . . . .	770
mln::Object< neighb< graph_elt_window< G, S > > > . . . . .	771
mln::Neighborhood< neighb< graph_elt_window< G, S > > > . . . . .	770
mln::Object< neighb< graph_elt_window_if< G, S, I > > > . . . . .	771
mln::Neighborhood< neighb< graph_elt_window_if< G, S, I > > > . . . . .	770
mln::Object< neighb< W > > . . . . .	771
mln::Neighborhood< neighb< W > > . . . . .	770
mln::Object< neighb_bkd_niter< W > > . . . . .	771

mln::Proxy< neighb_bkd_niter< W > > . . . . .	870
mln::Site_Proxy< neighb_bkd_niter< W > > . . . . .	878
mln::Site_Iterator< neighb_bkd_niter< W > > . . . . .	876
mln::Object< neighb_fwd_niter< W > > . . . . .	771
mln::Proxy< neighb_fwd_niter< W > > . . . . .	870
mln::Site_Proxy< neighb_fwd_niter< W > > . . . . .	878
mln::Site_Iterator< neighb_fwd_niter< W > > . . . . .	876
mln::Object< nil > . . . . .	771
mln::Meta_Accumulator< nil > . . . . .	752
mln::accu::meta::nil . . . . .	427
mln::util::nil . . . . .	977
mln::Object< nil< T > > . . . . .	771
mln::Proxy< nil< T > > . . . . .	870
mln::Accumulator< nil< T > > . . . . .	485
base< util::ignore, nil< T > > . . . . .	
mln::accu::nil< T > . . . . .	445
mln::Object< not_to_remove > . . . . .	771
mln::Function< not_to_remove > . . . . .	649
mln::Function_v2v< not_to_remove > . . . . .	652
mln::Function_v2b< not_to_remove > . . . . .	651
mln::Object< object_id< Tag, V > > . . . . .	771
mln::Value< object_id< Tag, V > > . . . . .	1008
mln::Object< octagon2d > . . . . .	771
mln::Window< octagon2d > . . . . .	1065
window_base< dpoint2d, octagon2d > . . . . .	
classical_window_base< dpoint2d, octagon2d > . . . . .	
mln::win::octagon2d . . . . .	1062
mln::Object< olive_t > . . . . .	771
mln::Literal< olive_t > . . . . .	727
mln::literal::olive_t . . . . .	739
mln::Object< one_t > . . . . .	771
mln::Literal< one_t > . . . . .	727
mln::literal::one_t . . . . .	740
mln::Object< orange_t > . . . . .	771
mln::Literal< orange_t > . . . . .	727
mln::literal::orange_t . . . . .	741
mln::Object< ord_pair< T > > . . . . .	771
mln::util::ord_pair< T > . . . . .	980
mln::Object< origin_t > . . . . .	771
mln::Literal< origin_t > . . . . .	727
mln::literal::origin_t . . . . .	742
mln::Object< P > . . . . .	771
mln::Point_Site< P > . . . . .	866
mln::Point< P > . . . . .	858
mln::Object< p2p_image< I, F > > . . . . .	771
mln::Image< p2p_image< I, F > > . . . . .	694
mln::Object< p< A > > . . . . .	771
mln::Proxy< p< A > > . . . . .	870

mln::Accumulator< p< A > > . . . . .	485
base< const A::result &, p< A > >	
mln::accu::p< A > . . . . .	447
mln::Object< p< mA > > . . . . .	771
mln::Meta_Accumulator< p< mA > > . . . . .	752
mln::accu::meta::p< mA > . . . . .	428
mln::Object< p_array< P > > . . . . .	771
mln::Site_Set< p_array< P > > . . . . .	879
site_set_base_< P, p_array< P > >	
mln::p_array< P > . . . . .	773
mln::Object< p_centered< W > > . . . . .	771
mln::Site_Set< p_centered< W > > . . . . .	879
site_set_base_< W::psite, p_centered< W > >	
mln::p_centered< W > . . . . .	778
mln::Object< p_centered_piter< W > > . . . . .	771
mln::Proxy< p_centered_piter< W > > . . . . .	870
mln::Site_Proxy< p_centered_piter< W > > . . . . .	878
mln::Site_Iterator< p_centered_piter< W > > . . . . .	876
mln::Object< p_complex< D, G > > . . . . .	771
mln::Site_Set< p_complex< D, G > > . . . . .	879
site_set_base_< complex_psite< D, G >, p_complex< D, G > >	
mln::p_complex< D, G > . . . . .	781
mln::Object< p_double_piter< S, I1, I2 > > . . . . .	771
mln::Proxy< p_double_piter< S, I1, I2 > > . . . . .	870
mln::Site_Proxy< p_double_piter< S, I1, I2 > > . . . . .	878
mln::Site_Iterator< p_double_piter< S, I1, I2 > > . . . . .	876
mln::Object< p_double_psite< S, Sp > > . . . . .	771
mln::Proxy< p_double_psite< S, Sp > > . . . . .	870
mln::Site_Proxy< p_double_psite< S, Sp > > . . . . .	878
mln::Pseudo_Site< p_double_psite< S, Sp > > . . . . .	871
mln::Object< p_edges< G, F > > . . . . .	771
mln::Site_Set< p_edges< G, F > > . . . . .	879
site_set_base_< F::result, p_edges< G, F > >	
mln::p_edges< G, F > . . . . .	784
mln::Object< p_edges_psite< G, F > > . . . . .	771
mln::Proxy< p_edges_psite< G, F > > . . . . .	870
mln::Site_Proxy< p_edges_psite< G, F > > . . . . .	878
mln::Pseudo_Site< p_edges_psite< G, F > > . . . . .	871
mln::Object< p_faces< N, D, P > > . . . . .	771
mln::Site_Set< p_faces< N, D, P > > . . . . .	879
site_set_base_< faces_psite< N, D, P >, p_faces< N, D, P > >	
mln::p_faces< N, D, P > . . . . .	789
mln::Object< p_graph_piter< S, I > > . . . . .	771
mln::Proxy< p_graph_piter< S, I > > . . . . .	870
mln::Site_Proxy< p_graph_piter< S, I > > . . . . .	878
mln::Site_Iterator< p_graph_piter< S, I > > . . . . .	876
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 > . . . . .	792

mln::Object< p_if< S, F > > . . . . .	771
mln::Site_Set< p_if< S, F > > . . . . .	879
site_set_base_< S::psite, p_if< S, F > >	
mln::p_if< S, F > . . . . .	793
mln::Object< p_image< I > > . . . . .	771
mln::Site_Set< p_image< I > > . . . . .	879
site_set_base_< I::psite, p_image< I > >	
mln::p_image< I > . . . . .	796
mln::Object< p_indexed_bkd_piter< S > > . . . . .	771
mln::Proxy< p_indexed_bkd_piter< S > > . . . . .	870
mln::Site_Proxy< p_indexed_bkd_piter< S > > . . . . .	878
mln::Site_Iterator< p_indexed_bkd_piter< S > > . . . . .	876
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 > . . . . .	800
mln::Object< p_indexed_fwd_piter< S > > . . . . .	771
mln::Proxy< p_indexed_fwd_piter< S > > . . . . .	870
mln::Site_Proxy< p_indexed_fwd_piter< S > > . . . . .	878
mln::Site_Iterator< p_indexed_fwd_piter< S > > . . . . .	876
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 > . . . . .	801
mln::Object< p_indexed_psite< S > > . . . . .	771
mln::Proxy< p_indexed_psite< S > > . . . . .	870
mln::Site_Proxy< p_indexed_psite< S > > . . . . .	878
mln::Pseudo_Site< p_indexed_psite< S > > . . . . .	871
pseudo_site_base_< const S::element &, p_indexed_psite< S > >	
mln::p_indexed_psite< S > . . . . .	802
mln::Object< p_key< K, P > > . . . . .	771
mln::Site_Set< p_key< K, P > > . . . . .	879
site_set_base_< P, p_key< K, P > >	
mln::p_key< K, P > . . . . .	802
mln::Object< p_line2d > . . . . .	771
mln::Site_Set< p_line2d > . . . . .	879
site_set_base_< point2d, p_line2d >	
mln::p_line2d . . . . .	807
mln::Object< p_mutable_array_of< S > > . . . . .	771
mln::Site_Set< p_mutable_array_of< S > > . . . . .	879
site_set_base_< S::site, p_mutable_array_of< S > >	
mln::p_mutable_array_of< S > . . . . .	811
mln::Object< p_n_faces_bkd_piter< D, G > > . . . . .	771
mln::Proxy< p_n_faces_bkd_piter< D, G > > . . . . .	870
mln::Site_Proxy< p_n_faces_bkd_piter< D, G > > . . . . .	878
mln::Site_Iterator< p_n_faces_bkd_piter< D, G > > . . . . .	876
site_iterator_base< p_complex< D, G >, p_n_faces_bkd_piter< D, G > >	
site_set_iterator_base< p_complex< D, G >, p_n_faces_bkd_piter< D, G > >	
p_complex_piter_base_< topo::n_face_bkd_iter< D >, p_complex< D, G >, G::site, p_n_faces_bkd_piter< D, G > >	
mln::p_n_faces_bkd_piter< D, G > . . . . .	814
mln::Object< p_n_faces_fwd_piter< D, G > > . . . . .	771

mln::Proxy< p_n_faces_fwd_piter< D, G > > . . . . .	870
mln::Site_Proxy< p_n_faces_fwd_piter< D, G > > . . . . .	878
mln::Site_Iterator< p_n_faces_fwd_piter< D, G > > . . . . .	876
site_iterator_base< p_complex< D, G >, p_n_faces_fwd_piter< D, G > >	
site_set_iterator_base< p_complex< D, G >, p_n_faces_fwd_piter< D, G > >	
p_complex_piter_base_< topo::n_face_fwd_iter< D >, p_complex< D, G >, G::site, p_n_faces_fwd_piter< D, G > >	
mln::p_n_faces_fwd_piter< D, G > . . . . .	815
mln::Object< p_priority< P, Q > > . . . . .	771
mln::Site_Set< p_priority< P, Q > > . . . . .	879
site_set_base_< Q::site, p_priority< P, Q > >	
mln::p_priority< P, Q > . . . . .	816
mln::Object< p_queue< P > > . . . . .	771
mln::Site_Set< p_queue< P > > . . . . .	879
site_set_base_< P, p_queue< P > >	
mln::p_queue< P > . . . . .	821
mln::Object< p_queue_fast< P > > . . . . .	771
mln::Site_Set< p_queue_fast< P > > . . . . .	879
site_set_base_< P, p_queue_fast< P > >	
mln::p_queue_fast< P > . . . . .	825
mln::Object< p_run< P > > . . . . .	771
mln::Site_Set< p_run< P > > . . . . .	879
site_set_base_< P, p_run< P > >	
mln::p_run< P > . . . . .	830
mln::Object< p_run_psite< P > > . . . . .	771
mln::Proxy< p_run_psite< P > > . . . . .	870
mln::Site_Proxy< p_run_psite< P > > . . . . .	878
mln::Pseudo_Site< p_run_psite< P > > . . . . .	871
mln::Object< p_set< P > > . . . . .	771
mln::Site_Set< p_set< P > > . . . . .	879
site_set_base_< P, p_set< P > >	
mln::p_set< P > . . . . .	834
mln::Object< p_set_of< S > > . . . . .	771
mln::Site_Set< p_set_of< S > > . . . . .	879
site_set_base_< S::site, p_set_of< S > >	
mln::p_set_of< S > . . . . .	838
mln::Object< p_transformed< S, F > > . . . . .	771
mln::Site_Set< p_transformed< S, F > > . . . . .	879
site_set_base_< S::psite, p_transformed< S, F > >	
mln::p_transformed< S, F > . . . . .	841
mln::Object< p_transformed_piter< Pi, S, F > > . . . . .	771
mln::Proxy< p_transformed_piter< Pi, S, F > > . . . . .	870
mln::Site_Proxy< p_transformed_piter< Pi, S, F > > . . . . .	878
mln::Site_Iterator< p_transformed_piter< Pi, S, F > > . . . . .	876
mln::Object< p_vaccess< V, S > > . . . . .	771
mln::Site_Set< p_vaccess< V, S > > . . . . .	879
site_set_base_< S::site, p_vaccess< V, S > >	
mln::p_vaccess< V, S > . . . . .	845
mln::Object< p_vertices< G, F > > . . . . .	771



mln::Site_Set< p_vertices< G, F > > . . . . .	879
site_set_base_< F::result, p_vertices< G, F > >	
mln::p_vertices< G, F > . . . . .	848
mln::Object< p_vertices_psite< G, F > > . . . . .	771
mln::Proxy< p_vertices_psite< G, F > > . . . . .	870
mln::Site_Proxy< p_vertices_psite< G, F > > . . . . .	878
mln::Pseudo_Site< p_vertices_psite< G, F > > . . . . .	871
mln::Object< pair< A1, A2 > > . . . . .	771
mln::Meta_Accumulator< pair< A1, A2 > > . . . . .	752
mln::accu::meta::pair< A1, A2 > . . . . .	429
mln::Object< pair< A1, A2, T > > . . . . .	771
mln::Proxy< pair< A1, A2, T > > . . . . .	870
mln::Accumulator< pair< A1, A2, T > > . . . . .	485
base< std::pair< A1::result, A2::result >, pair< A1, A2, T > >	
mln::accu::pair< A1, A2, T > . . . . .	448
mln::Object< pair< min< V >, max< V >, mln_argument(min< V >) > > . . . . .	771
mln::Proxy< pair< min< V >, max< V >, mln_argument(min< V >) > > . . . . .	870
mln::Accumulator< pair< min< V >, max< V >, mln_argument(min< V >) > > . . . . .	485
mln::Object< pink_t > . . . . .	771
mln::Literal< pink_t > . . . . .	727
mln::literal::pink_t . . . . .	743
mln::Object< pixel< I > > . . . . .	771
mln::pixel< I > . . . . .	853
mln::Object< plain< I > > . . . . .	771
mln::Image< plain< I > > . . . . .	694
mln::Object< point< G, C > > . . . . .	771
mln::Site< point< G, C > > . . . . .	875
mln::Gpoint< point< G, C > > . . . . .	661
mln::point< G, C > . . . . .	860
mln::Object< point_from_value< T > > . . . . .	771
mln::Function< point_from_value< T > > . . . . .	649
mln::Function_v2v< point_from_value< T > > . . . . .	652
mln::Object< projection< P, dir > > . . . . .	771
mln::Function< projection< P, dir > > . . . . .	649
mln::Function_v2v< projection< P, dir > > . . . . .	652
mln::Object< proxy< I > > . . . . .	771
mln::Proxy< proxy< I > > . . . . .	870
mln::value::proxy< I > . . . . .	1031
mln::Object< purple_t > . . . . .	771
mln::Literal< purple_t > . . . . .	727
mln::literal::purple_t . . . . .	744
mln::Object< qrde > . . . . .	771
mln::Function< qrde > . . . . .	649
mln::Function_v2v< qrde > . . . . .	652
mln::Object< qt_rgb_to_int_u< n > > . . . . .	771
mln::Function< qt_rgb_to_int_u< n > > . . . . .	649
mln::Function_v2v< qt_rgb_to_int_u< n > > . . . . .	652
mln::Object< quat > . . . . .	771

mln::Value< quat > . . . . .	1008
mln::Object< rank > . . . . .	771
mln::Meta_Accumulator< rank > . . . . .	752
mln::accu::meta::stat::rank . . . . .	441
mln::Object< rank< bool > > . . . . .	771
mln::Proxy< rank< bool > > . . . . .	870
mln::Accumulator< rank< bool > > . . . . .	485
base< bool, rank< bool > >	
mln::accu::stat::rank< bool > . . . . .	475
mln::Object< rank< T > > . . . . .	771
mln::Proxy< rank< T > > . . . . .	870
mln::Accumulator< rank< T > > . . . . .	485
base< const T &, rank< T > >	
mln::accu::stat::rank< T > . . . . .	474
mln::Object< rank_high_quant > . . . . .	771
mln::Meta_Accumulator< rank_high_quant > . . . . .	752
mln::accu::meta::stat::rank_high_quant . . . . .	442
mln::Object< rank_high_quant< T > > . . . . .	771
mln::Proxy< rank_high_quant< T > > . . . . .	870
mln::Accumulator< rank_high_quant< T > > . . . . .	485
base< const T &, rank_high_quant< T > >	
mln::accu::stat::rank_high_quant< T > . . . . .	477
mln::Object< rectangle2d > . . . . .	771
mln::Window< rectangle2d > . . . . .	1065
window_base< dpoint2d, rectangle2d >	
classical_window_base< dpoint2d, rectangle2d >	
mln::win::rectangle2d . . . . .	1063
mln::Object< rectangularity< P > > . . . . .	771
mln::Proxy< rectangularity< P > > . . . . .	870
mln::Accumulator< rectangularity< P > > . . . . .	485
mln::Object< red > . . . . .	771
mln::Meta_Function< red > . . . . .	754
mln::Meta_Function_v2v< red > . . . . .	754
mln::Object< red_t > . . . . .	771
mln::Literal< red_t > . . . . .	727
mln::literal::red_t . . . . .	745
mln::Object< ref_data > . . . . .	771
mln::Function< ref_data > . . . . .	649
mln::Function_v2v< ref_data > . . . . .	652
mln::Object< rgb32 > . . . . .	771
mln::Value< rgb32 > . . . . .	1008
mln::Object< rgb8_off_loader > . . . . .	771
mln::Object< rgb8_off_saver > . . . . .	771
mln::Object< rgb< n > > . . . . .	771
mln::Value< rgb< n > > . . . . .	1008
mln::Object< rgb_to_int_u< n > > . . . . .	771
mln::Function< rgb_to_int_u< n > > . . . . .	649
mln::Function_v2v< rgb_to_int_u< n > > . . . . .	652

mln::Object< rms > . . . . .	771
mln::Meta_Accumulator< rms > . . . . .	752
mln::accu::meta::rms . . . . .	430
mln::Object< rms< T, V > > . . . . .	771
mln::Proxy< rms< T, V > > . . . . .	870
mln::Accumulator< rms< T, V > > . . . . .	485
base< V, rms< T, V > >	
mln::accu::rms< T, V > . . . . .	450
mln::Object< rotation< n, C > > . . . . .	771
mln::Function< rotation< n, C > > . . . . .	649
mln::Function_v2v< rotation< n, C > > . . . . .	652
mln::fun::x2x::rotation< n, C > . . . . .	641
mln::Object< round< R > > . . . . .	771
mln::Function< round< R > > . . . . .	649
mln::Function_v2v< round< R > > . . . . .	652
mln::Object< row > . . . . .	771
mln::Meta_Function< row > . . . . .	754
mln::Meta_Function_v2v< row > . . . . .	754
mln::Object< safe_image< I > > . . . . .	771
mln::Image< safe_image< I > > . . . . .	694
mln::Object< saturate< V > > . . . . .	771
mln::Function< saturate< V > > . . . . .	649
mln::Function_v2v< saturate< V > > . . . . .	652
mln::Object< saturate_rgb8 > . . . . .	771
mln::Function< saturate_rgb8 > . . . . .	649
mln::Function_v2v< saturate_rgb8 > . . . . .	652
mln::Object< scomp< ith > > . . . . .	771
mln::Meta_Function< scomp< ith > > . . . . .	754
mln::Meta_Function_v2v< scomp< ith > > . . . . .	754
mln::Object< set_bkd_iter< T > > . . . . .	771
mln::Proxy< set_bkd_iter< T > > . . . . .	870
mln::Object< set_fwd_iter< T > > . . . . .	771
mln::Proxy< set_fwd_iter< T > > . . . . .	870
mln::Object< sharpness< I > > . . . . .	771
mln::Proxy< sharpness< I > > . . . . .	870
mln::Accumulator< sharpness< I > > . . . . .	485
base< double, sharpness< I > >	
mln::morpho::attribute::sharpness< I > . . . . .	764
mln::Object< shell< F, I > > . . . . .	771
mln::Proxy< shell< F, I > > . . . . .	870
mln::Object< sign > . . . . .	771
mln::Value< sign > . . . . .	1008
mln::Object< site_pair< P > > . . . . .	771
mln::util::site_pair< P > . . . . .	990
mln::Object< sli > . . . . .	771
mln::Meta_Function< sli > . . . . .	754
mln::Meta_Function_v2v< sli > . . . . .	754
mln::Object< slice_image< I > > . . . . .	771

mln::Image< slice_image< I > > . . . . .	694
mln::Object< snake_fwd_t > . . . . .	771
mln::Browsing< snake_fwd_t > . . . . .	509
mln::canvas::browsing::snake_fwd_t . . . . .	519
mln::Object< snake_generic_t > . . . . .	771
mln::Browsing< snake_generic_t > . . . . .	509
mln::canvas::browsing::snake_generic_t . . . . .	520
mln::Object< snake_vert_t > . . . . .	771
mln::Browsing< snake_vert_t > . . . . .	509
mln::canvas::browsing::snake_vert_t . . . . .	522
mln::Object< soft_heap< T, R > > . . . . .	771
mln::util::soft_heap< T, R > . . . . .	991
mln::Object< sqrt > . . . . .	771
mln::Function< sqrt > . . . . .	649
mln::Function_v2v< sqrt > . . . . .	652
mln::Object< square > . . . . .	771
mln::Mesh< square > . . . . .	751
mln::Regular_Grid< square > . . . . .	873
mln::Object< stack_image< n, I > > . . . . .	771
mln::Image< stack_image< n, I > > . . . . .	694
mln::Object< static_n_face_bkd_iter< N, D > > . . . . .	771
mln::Iterator< static_n_face_bkd_iter< N, D > > . . . . .	716
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 > . . . . .	934
mln::Object< static_n_face_fwd_iter< N, D > > . . . . .	771
mln::Iterator< static_n_face_fwd_iter< N, D > > . . . . .	716
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 > . . . . .	936
mln::Object< sub_image< I, S > > . . . . .	771
mln::Image< sub_image< I, S > > . . . . .	694
mln::Object< sub_image_if< I, S > > . . . . .	771
mln::Image< sub_image_if< I, S > > . . . . .	694
mln::Object< sum > . . . . .	771
mln::Meta_Accumulator< sum > . . . . .	752
mln::accu::meta::math::sum . . . . .	424
mln::Object< sum< I, S > > . . . . .	771
mln::Proxy< sum< I, S > > . . . . .	870
mln::Accumulator< sum< I, S > > . . . . .	485
base< S, sum< I, S > >	
mln::morpho::attribute::sum< I, S > . . . . .	765
mln::Object< sum< T, S > > . . . . .	771
mln::Proxy< sum< T, S > > . . . . .	870
mln::Accumulator< sum< T, S > > . . . . .	485
base< const S &, sum< T, S > >	
mln::accu::math::sum< T, S > . . . . .	407
mln::Object< sup > . . . . .	771

mln::Meta_Accumulator< sup > . . . . .	752
mln::accu::meta::math::sup . . . . .	425
mln::Meta_Function< sup > . . . . .	754
mln::Meta_Function_vv2v< sup > . . . . .	755
mln::Object< sup< T > > . . . . .	771
mln::Proxy< sup< T > > . . . . .	870
mln::Accumulator< sup< T > > . . . . .	485
base< const T &, sup< T > >	
mln::accu::math::sup< T > . . . . .	408
mln::Object< tautology > . . . . .	771
mln::Function< tautology > . . . . .	649
mln::Function_v2v< tautology > . . . . .	652
mln::Function_v2b< tautology > . . . . .	651
mln::fun::p2b::tautology . . . . .	610
mln::Object< teal_t > . . . . .	771
mln::Literal< teal_t > . . . . .	727
mln::literal::teal_t . . . . .	746
mln::Object< test > . . . . .	771
mln::Function< test > . . . . .	649
mln::Function_v2v< test > . . . . .	652
mln::Object< threshold< V > > . . . . .	771
mln::Function< threshold< V > > . . . . .	649
mln::Function_v2v< threshold< V > > . . . . .	652
mln::Function_v2b< threshold< V > > . . . . .	651
mln::fun::v2b::threshold< V > . . . . .	612
mln::Object< thru_image< I, F > > . . . . .	771
mln::Image< thru_image< I, F > > . . . . .	694
mln::Object< thrubin_image< I1, I2, F > > . . . . .	771
mln::Image< thrubin_image< I1, I2, F > > . . . . .	694
mln::Object< tick > . . . . .	771
mln::Mesh< tick > . . . . .	751
mln::Regular_Grid< tick > . . . . .	873
mln::Object< timer > . . . . .	771
mln::Proxy< timer > . . . . .	870
mln::util::timer . . . . .	994
mln::Object< to16bits > . . . . .	771
mln::Function< to16bits > . . . . .	649
mln::Function_v2v< to16bits > . . . . .	652
mln::Object< to19bits > . . . . .	771
mln::Function< to19bits > . . . . .	649
mln::Function_v2v< to19bits > . . . . .	652
mln::Object< to23bits > . . . . .	771
mln::Function< to23bits > . . . . .	649
mln::Function_v2v< to23bits > . . . . .	652
mln::Object< to27bits > . . . . .	771
mln::Function< to27bits > . . . . .	649
mln::Function_v2v< to27bits > . . . . .	652
mln::Object< to8bits > . . . . .	771

mln::Function< to8bits > . . . . .	649
mln::Function_v2v< to8bits > . . . . .	652
mln::Object< tofloat01 > . . . . .	771
mln::Function< tofloat01 > . . . . .	649
mln::Function_v2v< tofloat01 > . . . . .	652
mln::Object< tr_image< S, I, T > > . . . . .	771
mln::Image< tr_image< S, I, T > > . . . . .	694
mln::Object< transformed_image< I, F > > . . . . .	771
mln::Image< transformed_image< I, F > > . . . . .	694
mln::Object< translation< n, C > > . . . . .	771
mln::Function< translation< n, C > > . . . . .	649
mln::Function_v2v< translation< n, C > > . . . . .	652
mln::fun::x2x::translation< n, C > . . . . .	644
mln::Object< translation_t< P > > . . . . .	771
mln::Function< translation_t< P > > . . . . .	649
mln::Function_v2v< translation_t< P > > . . . . .	652
mln::Object< tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T)> > . . . . .	771
mln::Proxy< tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T)> > . . . . .	870
mln::Accumulator< tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T)> > . . . . .	485
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, > . . . . .	483
mln::Object< tuple< n, BOOST_PP_ENUM_PARAMS(10, T)> > . . . . .	771
mln::Meta_Accumulator< tuple< n, BOOST_PP_ENUM_PARAMS(10, T)> > . . . . .	752
mln::accu::meta::tuple< n, > . . . . .	443
mln::Object< unary< Fun, T > > . . . . .	771
mln::Function< unary< Fun, T > > . . . . .	649
mln::Function_v2v< unary< Fun, T > > . . . . .	652
mln::Object< unproject_image< I, D, F > > . . . . .	771
mln::Image< unproject_image< I, D, F > > . . . . .	694
mln::Object< up_leaf_piter< T > > . . . . .	771
mln::Proxy< up_leaf_piter< T > > . . . . .	870
mln::Site_Proxy< up_leaf_piter< T > > . . . . .	878
mln::Site_Iterator< up_leaf_piter< T > > . . . . .	876
mln::Object< up_node_piter< T > > . . . . .	771
mln::Proxy< up_node_piter< T > > . . . . .	870
mln::Site_Proxy< up_node_piter< T > > . . . . .	878
mln::Site_Iterator< up_node_piter< T > > . . . . .	876
mln::Object< up_site_piter< T > > . . . . .	771
mln::Proxy< up_site_piter< T > > . . . . .	870
mln::Site_Proxy< up_site_piter< T > > . . . . .	878
mln::Site_Iterator< up_site_piter< T > > . . . . .	876
mln::Object< val< A > > . . . . .	771
mln::Proxy< val< A > > . . . . .	870
mln::Accumulator< val< A > > . . . . .	485
base< const A::result &, val< A > > . . . . .	
mln::accu::val< A > . . . . .	484
mln::Object< val< mA > > . . . . .	771

mln::Meta_Accumulator< val< mA > > . . . . .	752
mln::accu::meta::val< mA > . . . . .	444
mln::Object< value_at_index< bool > > . . . . .	771
mln::Function< value_at_index< bool > > . . . . .	649
mln::Function_v2v< value_at_index< bool > > . . . . .	652
mln::Object< value_at_index< T > > . . . . .	771
mln::Function< value_at_index< T > > . . . . .	649
mln::Function_v2v< value_at_index< T > > . . . . .	652
mln::Object< var< T > > . . . . .	771
mln::Proxy< var< T > > . . . . .	870
mln::Accumulator< var< T > > . . . . .	485
base< algebra::mat< T::dim, T::dim, float >, var< T > >	
mln::accu::stat::var< T > . . . . .	478
mln::Object< variance< T, S, R > > . . . . .	771
mln::Proxy< variance< T, S, R > > . . . . .	870
mln::Accumulator< variance< T, S, R > > . . . . .	485
base< R, variance< T, S, R > >	
mln::accu::stat::variance< T, S, R > . . . . .	480
mln::Object< vec< 1, T > > . . . . .	771
mln::Object< vec< 2, T > > . . . . .	771
mln::Object< vec< 3, T > > . . . . .	771
mln::Object< vec< 4, T > > . . . . .	771
mln::Object< vec< n, C > > . . . . .	771
mln::Object< vec< n, T > > . . . . .	771
mln::Object< vec< V > > . . . . .	771
mln::Function< vec< V > > . . . . .	649
mln::Function_vv2v< vec< V > > . . . . .	653
mln::fun::vv2v::vec< V > . . . . .	636
mln::Object< vertex< G > > . . . . .	771
mln::Site< vertex< G > > . . . . .	875
mln::util::vertex< G > . . . . .	1004
mln::Object< vertex_bkd_iterator< G > > . . . . .	771
mln::Proxy< vertex_bkd_iterator< G > > . . . . .	870
mln::Object< vertex_fwd_iterator< G > > . . . . .	771
mln::Proxy< vertex_fwd_iterator< G > > . . . . .	870
mln::Object< vertex_image< P, V, G > > . . . . .	771
mln::Image< vertex_image< P, V, G > > . . . . .	694
mln::Object< vertex_nbh_edge_bkd_iterator< G > > . . . . .	771
mln::Proxy< vertex_nbh_edge_bkd_iterator< G > > . . . . .	870
mln::Object< vertex_nbh_edge_fwd_iterator< G > > . . . . .	771
mln::Proxy< vertex_nbh_edge_fwd_iterator< G > > . . . . .	870
mln::Object< vertex_nbh_vertex_bkd_iterator< G > > . . . . .	771
mln::Proxy< vertex_nbh_vertex_bkd_iterator< G > > . . . . .	870
mln::Object< vertex_nbh_vertex_fwd_iterator< G > > . . . . .	771
mln::Proxy< vertex_nbh_vertex_fwd_iterator< G > > . . . . .	870
mln::Object< violent_cast_image< T, I > > . . . . .	771
mln::Image< violent_cast_image< T, I > > . . . . .	694
mln::Object< violet_t > . . . . .	771

mln::Literal< violet_t > . . . . .	727
mln::literal::violet_t . . . . .	747
mln::Object< viota_t > . . . . .	771
mln::Function< viota_t > . . . . .	649
mln::Function_v2v< viota_t > . . . . .	652
mln::Object< viota_t< S > > . . . . .	771
mln::Function< viota_t< S > > . . . . .	649
mln::Function_v2v< viota_t< S > > . . . . .	652
mln::Object< volume > . . . . .	771
mln::Meta_Accumulator< volume > . . . . .	752
mln::accu::meta::shape::volume . . . . .	433
mln::Object< volume< I > > . . . . .	771
mln::Proxy< volume< I > > . . . . .	870
mln::Accumulator< volume< I > > . . . . .	485
base< unsigned, volume< I > >	
mln::accu::shape::volume< I > . . . . .	455
mln::morpho::attribute::volume< I > . . . . .	767
mln::Object< W > . . . . .	771
mln::Object< w_window< D, W > > . . . . .	771
mln::Weighted_Window< w_window< D, W > > . . . . .	1053
weighted_window_base< mln::window< D >, w_window< D, W > >	
mln::w_window< D, W > . . . . .	1049
mln::Object< white_gaussian< V > > . . . . .	771
mln::Function< white_gaussian< V > > . . . . .	649
mln::Function_n2v< white_gaussian< V > > . . . . .	650
mln::fun::n2v::white_gaussian< V > . . . . .	608
mln::Object< white_t > . . . . .	771
mln::Literal< white_t > . . . . .	727
mln::literal::white_t . . . . .	748
mln::Object< window< D > > . . . . .	771
mln::Window< window< D > > . . . . .	1065
window_base< D, window< D > >	
mln::window< D > . . . . .	1065
mln::Object< wrap > . . . . .	771
mln::Function< wrap > . . . . .	649
mln::Function_v2v< wrap > . . . . .	652
mln::Object< wrap< L > > . . . . .	771
mln::Function< wrap< L > > . . . . .	649
mln::Function_v2v< wrap< L > > . . . . .	652
mln::Object< yellow_t > . . . . .	771
mln::Literal< yellow_t > . . . . .	727
mln::literal::yellow_t . . . . .	749
mln::Object< yes > . . . . .	771
mln::util::yes . . . . .	1008
mln::Object< zero_t > . . . . .	771
mln::Literal< zero_t > . . . . .	727
mln::literal::zero_t . . . . .	750
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 > . . . . .	586
mln::internal::pixel_impl_< I, dpoints_fwd_pixter< I > >	
mln::dpoints_fwd_pixter< I > . . . . .	588
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 > . . . . .	853
trait::graph< I > . . . . .	1070
trait::graph< mln::complex_image< I, G, V > > . . . . .	1071
trait::graph< mln::image2d< T > > . . . . .	1071



# Chapter 7

## Class Index

### 7.1 Class List

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

<a href="#">mln::accu::center&lt; P, V &gt;</a> (Mass center accumulator) . . . . .	389
<a href="#">mln::accu::convolve&lt; T1, T2, R &gt;</a> (Generic convolution accumulator class) . . . . .	390
<a href="#">mln::accu::count_adjacent_vertices&lt; F, S &gt;</a> ( <a href="#">Accumulator</a> class counting the number of vertices adjacent to a set of <a href="#">mln::p_edges_psite</a> (i.e., a set of edges)) . . . . .	392
<a href="#">mln::accu::count_value&lt; V &gt;</a> (Define an accumulator that counts the occurrence of a given value) . . . . .	393
<a href="#">mln::accu::histo&lt; V &gt;</a> (Generic histogram class over a value set with type $V$ ) . . . . .	395
<a href="#">mln::accu::label_used&lt; L &gt;</a> (References all the labels used) . . . . .	396
<a href="#">mln::accu::logic::land</a> ("Logical-and" accumulator) . . . . .	398
<a href="#">mln::accu::logic::land_basic</a> ("Logical-and" accumulator) . . . . .	399
<a href="#">mln::accu::logic::lor</a> ("Logical-or" accumulator) . . . . .	400
<a href="#">mln::accu::logic::lor_basic</a> ("Logical-or" accumulator class) . . . . .	402
<a href="#">mln::accu::maj_h&lt; T &gt;</a> (Compute the majority value) . . . . .	403
<a href="#">mln::accu::math::count&lt; T &gt;</a> (Generic counter accumulator) . . . . .	404
<a href="#">mln::accu::math::inf&lt; T &gt;</a> (Generic inf accumulator class) . . . . .	406
<a href="#">mln::accu::math::sum&lt; T, S &gt;</a> (Generic sum accumulator class) . . . . .	407
<a href="#">mln::accu::math::sup&lt; T &gt;</a> (Generic sup accumulator class) . . . . .	408
<a href="#">mln::accu::max_site&lt; I &gt;</a> (Define an accumulator that computes the first site with the maximum value in an image) . . . . .	410
<a href="#">mln::accu::meta::center</a> (Meta accumulator for center) . . . . .	411
<a href="#">mln::accu::meta::count_adjacent_vertices</a> (Meta accumulator for <a href="#">count_adjacent_vertices</a> ) . . . . .	412
<a href="#">mln::accu::meta::count_labels</a> (Meta accumulator for <a href="#">count_labels</a> ) . . . . .	413
<a href="#">mln::accu::meta::count_value</a> (FIXME: How to write a meta accumulator with a constructor taking a generic argument? Meta accumulator for <a href="#">count_value</a> ) . . . . .	414
<a href="#">mln::accu::meta::histo</a> (Meta accumulator for <a href="#">histo</a> ) . . . . .	415
<a href="#">mln::accu::meta::label_used</a> (Meta accumulator for <a href="#">label_used</a> ) . . . . .	416
<a href="#">mln::accu::meta::logic::land</a> (Meta accumulator for <a href="#">land</a> ) . . . . .	417
<a href="#">mln::accu::meta::logic::land_basic</a> (Meta accumulator for <a href="#">land_basic</a> ) . . . . .	418
<a href="#">mln::accu::meta::logic::lor</a> (Meta accumulator for <a href="#">lor</a> ) . . . . .	419
<a href="#">mln::accu::meta::logic::lor_basic</a> (Meta accumulator for <a href="#">lor_basic</a> ) . . . . .	420
<a href="#">mln::accu::meta::maj_h</a> (Meta accumulator for <a href="#">maj_h</a> ) . . . . .	421
<a href="#">mln::accu::meta::math::count</a> (Meta accumulator for <a href="#">count</a> ) . . . . .	422
<a href="#">mln::accu::meta::math::inf</a> (Meta accumulator for <a href="#">inf</a> ) . . . . .	423
<a href="#">mln::accu::meta::math::sum</a> (Meta accumulator for <a href="#">sum</a> ) . . . . .	424

<code>mln::accu::meta::math::sup</code> (Meta accumulator for <code>sup</code> ) . . . . .	425
<code>mln::accu::meta::max_site</code> (Meta accumulator for <code>max_site</code> ) . . . . .	426
<code>mln::accu::meta::nil</code> (Meta accumulator for <code>nil</code> ) . . . . .	427
<code>mln::accu::meta::p&lt; mA &gt;</code> (Meta accumulator for <code>p</code> ) . . . . .	428
<code>mln::accu::meta::pair&lt; A1, A2 &gt;</code> (Meta accumulator for <code>pair</code> ) . . . . .	429
<code>mln::accu::meta::rms</code> (Meta accumulator for <code>rms</code> ) . . . . .	430
<code>mln::accu::meta::shape::bbox</code> (Meta accumulator for <code>bbox</code> ) . . . . .	431
<code>mln::accu::meta::shape::height</code> (Meta accumulator for <code>height</code> ) . . . . .	432
<code>mln::accu::meta::shape::volume</code> (Meta accumulator for <code>volume</code> ) . . . . .	433
<code>mln::accu::meta::stat::max</code> (Meta accumulator for <code>max</code> ) . . . . .	434
<code>mln::accu::meta::stat::max_h</code> (Meta accumulator for <code>max</code> ) . . . . .	435
<code>mln::accu::meta::stat::mean</code> (Meta accumulator for <code>mean</code> ) . . . . .	436
<code>mln::accu::meta::stat::median_alt&lt; T &gt;</code> (Meta accumulator for <code>median_alt</code> ) . . . . .	437
<code>mln::accu::meta::stat::median_h</code> (Meta accumulator for <code>median_h</code> ) . . . . .	438
<code>mln::accu::meta::stat::min</code> (Meta accumulator for <code>min</code> ) . . . . .	439
<code>mln::accu::meta::stat::min_h</code> (Meta accumulator for <code>min</code> ) . . . . .	440
<code>mln::accu::meta::stat::rank</code> (Meta accumulator for <code>rank</code> ) . . . . .	441
<code>mln::accu::meta::stat::rank_high_quant</code> (Meta accumulator for <code>rank_high_quant</code> ) . . . . .	442
<code>mln::accu::meta::tuple&lt; n, &gt;</code> (Meta accumulator for <code>tuple</code> ) . . . . .	443
<code>mln::accu::meta::val&lt; mA &gt;</code> (Meta accumulator for <code>val</code> ) . . . . .	444
<code>mln::accu::nil&lt; T &gt;</code> (Define an accumulator that does nothing) . . . . .	445
<code>mln::accu::p&lt; A &gt;</code> (Generic <code>p</code> of accumulators) . . . . .	447
<code>mln::accu::pair&lt; A1, A2, T &gt;</code> (Generic pair of accumulators) . . . . .	448
<code>mln::accu::rms&lt; T, V &gt;</code> (Generic root mean square accumulator class) . . . . .	450
<code>mln::accu::shape::bbox&lt; P &gt;</code> (Generic bounding box accumulator class) . . . . .	452
<code>mln::accu::shape::height&lt; I &gt;</code> (Height accumulator) . . . . .	453
<code>mln::accu::shape::volume&lt; I &gt;</code> (Volume accumulator class) . . . . .	455
<code>mln::accu::site_set::rectangularity&lt; P &gt;</code> (Compute the rectangularity of a site set) . . . . .	457
<code>mln::accu::stat::deviation&lt; T, S, M &gt;</code> (Generic standard deviation accumulator class) . . . . .	459
<code>mln::accu::stat::max&lt; T &gt;</code> (Generic max accumulator class) . . . . .	460
<code>mln::accu::stat::max_h&lt; V &gt;</code> (Generic max function based on histogram over a value set with type <code>V</code> ) . . . . .	461
<code>mln::accu::stat::mean&lt; T, S, M &gt;</code> (Generic mean accumulator class) . . . . .	463
<code>mln::accu::stat::median_alt&lt; S &gt;</code> (Generic <code>median_alt</code> function based on histogram over a value set with type <code>S</code> ) . . . . .	464
<code>mln::accu::stat::median_h&lt; V &gt;</code> (Generic median function based on histogram over a value set with type <code>V</code> ) . . . . .	466
<code>mln::accu::stat::meta::deviation</code> (Meta accumulator for <code>deviation</code> ) . . . . .	468
<code>mln::accu::stat::min&lt; T &gt;</code> (Generic min accumulator class) . . . . .	468
<code>mln::accu::stat::min_h&lt; V &gt;</code> (Generic min function based on histogram over a value set with type <code>V</code> ) . . . . .	470
<code>mln::accu::stat::min_max&lt; V &gt;</code> (Generic min and max accumulator class) . . . . .	471
<code>mln::accu::stat::rank&lt; T &gt;</code> (Generic rank accumulator class) . . . . .	474
<code>mln::accu::stat::rank&lt; bool &gt;</code> (Rank accumulator class for Boolean) . . . . .	475
<code>mln::accu::stat::rank_high_quant&lt; T &gt;</code> (Generic rank accumulator class) . . . . .	477
<code>mln::accu::stat::var&lt; T &gt;</code> (Var accumulator class) . . . . .	478
<code>mln::accu::stat::variance&lt; T, S, R &gt;</code> (Variance accumulator class) . . . . .	480
<code>mln::accu::tuple&lt; A, n, &gt;</code> (Generic tuple of accumulators) . . . . .	483
<code>mln::accu::val&lt; A &gt;</code> (Generic <code>val</code> of accumulators) . . . . .	484
<code>mln::Accumulator&lt; E &gt;</code> (Base class for implementation of accumulators) . . . . .	485
<code>mln::algebra::h_mat&lt; d, T &gt;</code> (N-Dimensional matrix with homogeneous coordinates) . . . . .	487
<code>mln::algebra::h_vec&lt; d, C &gt;</code> (N-Dimensional vector with homogeneous coordinates) . . . . .	489
<code>mln::bkd_pixter1d&lt; I &gt;</code> (Backward pixel iterator on a 1-D image with border) . . . . .	491
<code>mln::bkd_pixter2d&lt; I &gt;</code> (Backward pixel iterator on a 2-D image with border) . . . . .	492

<code>mln::bkd_paxter3d&lt; I &gt;</code> (Backward pixel iterator on a 3-D image with border)	493
<code>mln::box&lt; P &gt;</code> (Generic box class: site set containing points of a regular grid)	494
<code>mln::Box&lt; E &gt;</code> (Base class for implementation classes of boxes)	501
<code>mln::box_runend_piter&lt; P &gt;</code> (A generic backward iterator on points by lines)	506
<code>mln::box_runstart_piter&lt; P &gt;</code> (A generic forward iterator on points by lines)	507
<code>mln::Browsing&lt; E &gt;</code> (Base class for implementation classes that are browsings)	509
<code>mln::canvas::browsing::backdiagonal2d_t</code> ( <code>Browsing</code> in a certain direction)	509
<code>mln::canvas::browsing::breadth_first_search_t</code> (Breadth-first search algorithm for graph, on vertices)	511
<code>mln::canvas::browsing::depth_first_search_t</code> (Breadth-first search algorithm for graph, on vertices)	511
<code>mln::canvas::browsing::diagonal2d_t</code> ( <code>Browsing</code> in a certain direction)	511
<code>mln::canvas::browsing::dir_struct_elt_incr_update_t</code> ( <code>Browsing</code> in a certain direction with a segment)	513
<code>mln::canvas::browsing::directional_t</code> ( <code>Browsing</code> in a certain direction)	514
<code>mln::canvas::browsing::fwd_t</code> (Canvas for forward browsing)	516
<code>mln::canvas::browsing::hyper_directional_t</code> ( <code>Browsing</code> in a certain direction)	517
<code>mln::canvas::browsing::snake_fwd_t</code> ( <code>Browsing</code> in a snake-way, forward)	519
<code>mln::canvas::browsing::snake_generic_t</code> (Multidimensional <code>Browsing</code> in a given-way)	520
<code>mln::canvas::browsing::snake_vert_t</code> ( <code>Browsing</code> in a snake-way, forward)	522
<code>mln::canvas::chamfer&lt; F &gt;</code> (Compute chamfer distance)	523
<code>mln::category&lt; R(*)&gt;(A)</code> (Category declaration for a unary C function)	523
<code>mln::complex_image&lt; D, G, V &gt;</code> (Image based on a complex)	523
<code>mln::complex_neighborhood_bkd_piter&lt; I, G, N &gt;</code> (Backward iterator on complex neighborhood)	526
<code>mln::complex_neighborhood_fwd_piter&lt; I, G, N &gt;</code> (Forward iterator on complex neighborhood)	528
<code>mln::complex_psite&lt; D, G &gt;</code> (Point site associated to a <code>mln::p_complex</code> )	530
<code>mln::complex_window_bkd_piter&lt; I, G, W &gt;</code> (Backward iterator on complex window)	532
<code>mln::complex_window_fwd_piter&lt; I, G, W &gt;</code> (Forward iterator on complex window)	534
<code>mln::decorated_image&lt; I, D &gt;</code> (Image that can have additional features)	535
<code>mln::Delta_Point_Site&lt; E &gt;</code> (FIXME: Doc!)	538
<code>mln::Delta_Point_Site&lt; void &gt;</code> (Delta point site category flag type)	539
<code>mln::doc::Accumulator&lt; E &gt;</code> (Documentation class for <code>mln::Accumulator</code> )	539
<code>mln::doc::Box&lt; E &gt;</code> (Documentation class for <code>mln::Box</code> )	540
<code>mln::doc::Dpoint&lt; E &gt;</code> (Documentation class for <code>mln::Dpoint</code> )	543
<code>mln::doc::Fastest_Image&lt; E &gt;</code> (Documentation class for the concept of images that have the speed property set to "fastest")	545
<code>mln::doc::Generalized_Pixel&lt; E &gt;</code> (Documentation class for <code>mln::Generalized_Pixel</code> )	553
<code>mln::doc::Image&lt; E &gt;</code> (Documentation class for <code>mln::Image</code> )	554
<code>mln::doc::Iterator&lt; E &gt;</code> (Documentation class for <code>mln::Iterator</code> )	560
<code>mln::doc::Neighborhood&lt; E &gt;</code> (Documentation class for <code>mln::Neighborhood</code> )	561
<code>mln::doc::Object&lt; E &gt;</code> (Documentation class for <code>mln::Object</code> )	563
<code>mln::doc::Pixel_Iterator&lt; E &gt;</code> (Documentation class for <code>mln::Iterator</code> )	564
<code>mln::doc::Point_Site&lt; E &gt;</code> (Documentation class for <code>mln::Point_Site</code> )	566
<code>mln::doc::Site_Iterator&lt; E &gt;</code> (Documentation class for <code>mln::Site_Iterator</code> )	568
<code>mln::doc::Site_Set&lt; E &gt;</code> (Documentation class for <code>mln::Site_Set</code> )	570
<code>mln::doc::Value_Iterator&lt; E &gt;</code> (Documentation class for <code>mln::Value_Iterator</code> )	572
<code>mln::doc::Value_Set&lt; E &gt;</code> (Documentation class for <code>mln::Value_Set</code> )	574
<code>mln::doc::Weighted_Window&lt; E &gt;</code> (Documentation class for <code>mln::Weighted_Window</code> )	577
<code>mln::doc::Window&lt; E &gt;</code> (Documentation class for <code>mln::Window</code> )	579
<code>mln::Dpoint&lt; E &gt;</code> (Base class for implementation of delta-point classes)	580
<code>mln::dpoint&lt; G, C &gt;</code> (Generic delta-point class)	581
<code>mln::dpoints_bkd_paxter&lt; I &gt;</code> (A generic backward iterator on the pixels of a dpoint-based window or neighborhood)	586

<a href="#">mln::dpoints_fwd_pixter&lt; I &gt;</a> (A generic forward iterator on the pixels of a dpoint-based window or neighborhood) . . . . .	588
<a href="#">mln::dpsites_bkd_piter&lt; V &gt;</a> (A generic backward iterator on points of windows and of neighborhoods) . . . . .	590
<a href="#">mln::dpsites_fwd_piter&lt; V &gt;</a> (A generic forward iterator on points of windows and of neighborhoods) . . . . .	592
<a href="#">mln::Edge&lt; E &gt;</a> (Edge category flag type) . . . . .	593
<a href="#">mln::edge_image&lt; P, V, G &gt;</a> (Image based on graph edges) . . . . .	593
<a href="#">mln::extended&lt; I &gt;</a> (Makes an image become restricted by a point set) . . . . .	595
<a href="#">mln::extension_fun&lt; I, F &gt;</a> (Extends the domain of an image with a function) . . . . .	597
<a href="#">mln::extension_ima&lt; I, J &gt;</a> (Extends the domain of an image with an image) . . . . .	599
<a href="#">mln::extension_val&lt; I &gt;</a> (Extends the domain of an image with a value) . . . . .	601
<a href="#">mln::faces_psite&lt; N, D, P &gt;</a> (Point site associated to a <a href="#">mln::p_faces</a> ) . . . . .	603
<a href="#">mln::flat_image&lt; T, S &gt;</a> (Image with a single value) . . . . .	606
<a href="#">mln::fun::from_accu&lt; A &gt;</a> (Wrap an accumulator into a function) . . . . .	608
<a href="#">mln::fun::n2v::white_gaussian&lt; V &gt;</a> (Generate a White Gaussian Noise) . . . . .	608
<a href="#">mln::fun::p2b::antilogy</a> (A <a href="#">p2b</a> function always returning <code>false</code> ) . . . . .	609
<a href="#">mln::fun::p2b::tautology</a> (A <a href="#">p2b</a> function always returning <code>true</code> ) . . . . .	610
<a href="#">mln::fun::v2b::lnot&lt; V &gt;</a> (Functor computing logical-not on a value) . . . . .	611
<a href="#">mln::fun::v2b::threshold&lt; V &gt;</a> (Threshold function) . . . . .	612
<a href="#">mln::fun::v2v::ch_function_value&lt; F, V &gt;</a> (Wrap a function <a href="#">v2v</a> and convert its result to another type) . . . . .	613
<a href="#">mln::fun::v2v::component&lt; T, i &gt;</a> (Functor that accesses the <i>i</i> -th component of a value) . . . . .	614
<a href="#">mln::fun::v2v::l1_norm&lt; V, R &gt;</a> (L1-norm) . . . . .	615
<a href="#">mln::fun::v2v::l2_norm&lt; V, R &gt;</a> (L2-norm) . . . . .	616
<a href="#">mln::fun::v2v::linear&lt; V, T, R &gt;</a> (Linear function. $f(v) = a * v + b$ . <i>V</i> is the type of input values; <i>T</i> is the type used to compute the result; <i>R</i> is the result type) . . . . .	617
<a href="#">mln::fun::v2v::linfty_norm&lt; V, R &gt;</a> (L-infty norm) . . . . .	618
<a href="#">mln::fun::v2w2v::cos&lt; V &gt;</a> (Cosinus bijective functor) . . . . .	619
<a href="#">mln::fun::v2w_w2v::l1_norm&lt; V, R &gt;</a> (L1-norm) . . . . .	620
<a href="#">mln::fun::v2w_w2v::l2_norm&lt; V, R &gt;</a> (L2-norm) . . . . .	621
<a href="#">mln::fun::v2w_w2v::linfty_norm&lt; V, R &gt;</a> (L-infty norm) . . . . .	622
<a href="#">mln::fun::vv2b::eq&lt; L, R &gt;</a> (Functor computing equal between two values) . . . . .	623
<a href="#">mln::fun::vv2b::ge&lt; L, R &gt;</a> (Functor computing "greater or equal than" between two values) . . . . .	624
<a href="#">mln::fun::vv2b::gt&lt; L, R &gt;</a> (Functor computing "greater than" between two values) . . . . .	625
<a href="#">mln::fun::vv2b::implies&lt; L, R &gt;</a> (Functor computing logical-implies between two values) . . . . .	626
<a href="#">mln::fun::vv2b::le&lt; L, R &gt;</a> (Functor computing "lower or equal than" between two values) . . . . .	627
<a href="#">mln::fun::vv2b::lt&lt; L, R &gt;</a> (Functor computing "lower than" between two values) . . . . .	628
<a href="#">mln::fun::vv2v::diff_abs&lt; V &gt;</a> (A functor computing the <code>diff_abs</code> imum of two values) . . . . .	629
<a href="#">mln::fun::vv2v::land&lt; L, R &gt;</a> (Functor computing logical-and between two values) . . . . .	630
<a href="#">mln::fun::vv2v::land_not&lt; L, R &gt;</a> (Functor computing logical and-not between two values) . . . . .	631
<a href="#">mln::fun::vv2v::lor&lt; L, R &gt;</a> (Functor computing logical-or between two values) . . . . .	632
<a href="#">mln::fun::vv2v::lxor&lt; L, R &gt;</a> (Functor computing logical-xor between two values) . . . . .	633
<a href="#">mln::fun::vv2v::max&lt; V &gt;</a> (A functor computing the maximum of two values) . . . . .	634
<a href="#">mln::fun::vv2v::min&lt; L, R &gt;</a> (A functor computing the minimum of two values) . . . . .	635
<a href="#">mln::fun::vv2v::vec&lt; V &gt;</a> (A functor computing the vecimum of two values) . . . . .	636
<a href="#">mln::fun::x2p::closest_point&lt; P &gt;</a> (FIXME: doxygen + concept checking) . . . . .	637
<a href="#">mln::fun::x2v::bilinear&lt; I &gt;</a> (Represent a bilinear interolation of values from an underlying image) . . . . .	638
<a href="#">mln::fun::x2v::trilinear&lt; I &gt;</a> (Represent a trilinear interolation of values from an underlying image) . . . . .	639
<a href="#">mln::fun::x2x::composed&lt; T2, T1 &gt;</a> (Represent a composition of two transformations) . . . . .	639
<a href="#">mln::fun::x2x::linear&lt; I &gt;</a> (Represent a linear interolation of values from an underlying image) . . . . .	640
<a href="#">mln::fun::x2x::rotation&lt; n, C &gt;</a> (Represent a rotation function) . . . . .	641

<a href="#">mln::fun::x2x::translation&lt; n, C &gt;</a> (Translation function-object) . . . . .	644
<a href="#">mln::fun_image&lt; F, I &gt;</a> ( <a href="#">Image</a> read through a function) . . . . .	647
<a href="#">mln::Function&lt; E &gt;</a> (Base class for implementation of function-objects) . . . . .	649
<a href="#">mln::Function&lt; void &gt;</a> ( <a href="#">Function</a> category flag type) . . . . .	650
<a href="#">mln::Function_n2v&lt; E &gt;</a> (Base class for implementation of function-objects from Nil to value) . . . . .	650
<a href="#">mln::Function_v2b&lt; E &gt;</a> (Base class for implementation of function-objects from a value to a Boolean) . . . . .	651
<a href="#">mln::Function_v2v&lt; E &gt;</a> (Base class for implementation of function-objects from value to value) . . . . .	652
<a href="#">mln::Function_vv2b&lt; E &gt;</a> (Base class for implementation of function-objects from a couple of values to a Boolean) . . . . .	653
<a href="#">mln::Function_vv2v&lt; E &gt;</a> (Base class for implementation of function-objects from a couple of values to a value) . . . . .	653
<a href="#">mln::fwd_pixter1d&lt; I &gt;</a> (Forward pixel iterator on a 1-D image with border) . . . . .	654
<a href="#">mln::fwd_pixter2d&lt; I &gt;</a> (Forward pixel iterator on a 2-D image with border) . . . . .	655
<a href="#">mln::fwd_pixter3d&lt; I &gt;</a> (Forward pixel iterator on a 3-D image with border) . . . . .	657
<a href="#">mln::Gdpoint&lt; E &gt;</a> (FIXME: Doc!) . . . . .	658
<a href="#">mln::Gdpoint&lt; void &gt;</a> (Delta point site category flag type) . . . . .	659
<a href="#">mln::Generalized_Pixel&lt; E &gt;</a> (Base class for implementation classes that are pixels or that have the behavior of pixels) . . . . .	659
<a href="#">mln::geom::complex_geometry&lt; D, P &gt;</a> (A functor returning the sites of the faces of a complex where the locations of each 0-face is stored) . . . . .	660
<a href="#">mln::Gpoint&lt; E &gt;</a> (Base class for implementation of point classes) . . . . .	661
<a href="#">mln::Graph&lt; E &gt;</a> (Base class for implementation of graph classes) . . . . .	666
<a href="#">mln::graph::attribute::card_t</a> (Compute the cardinality of every component in a graph) . . . . .	666
<a href="#">mln::graph::attribute::representative_t</a> (Compute the representative vertex of every component in a graph) . . . . .	667
<a href="#">mln::graph_elt_mixed_neighborhood&lt; G, S, S2 &gt;</a> (Elementary neighborhood on graph class) . . . . .	667
<a href="#">mln::graph_elt_mixed_window&lt; G, S, S2 &gt;</a> (Elementary window on graph class) . . . . .	669
<a href="#">mln::graph_elt_neighborhood&lt; G, S &gt;</a> (Elementary neighborhood on graph class) . . . . .	672
<a href="#">mln::graph_elt_neighborhood_if&lt; G, S, I &gt;</a> (Elementary neighborhood_if on graph class) . . . . .	674
<a href="#">mln::graph_elt_window&lt; G, S &gt;</a> (Elementary window on graph class) . . . . .	676
<a href="#">mln::graph_elt_window_if&lt; G, S, I &gt;</a> (Custom window on graph class) . . . . .	679
<a href="#">mln::graph_window_base&lt; P, E &gt;</a> . . . . .	684
<a href="#">mln::graph_window_if_piter&lt; S, W, I &gt;</a> (Forward iterator on line graph window) . . . . .	686
<a href="#">mln::graph_window_piter&lt; S, W, I &gt;</a> (Forward iterator on line graph window) . . . . .	687
<a href="#">mln::hexa&lt; I &gt;</a> (Hexagonal image class) . . . . .	690
<a href="#">mln::histo::array&lt; T &gt;</a> (Generic histogram class over a value set with type T) . . . . .	694
<a href="#">mln::Image&lt; E &gt;</a> (Base class for implementation of image classes) . . . . .	694
<a href="#">mln::image1d&lt; T &gt;</a> (Basic 1D image class) . . . . .	696
<a href="#">mln::image2d&lt; T &gt;</a> (Basic 2D image class) . . . . .	700
<a href="#">mln::image2d_h&lt; V &gt;</a> (2d image based on an hexagonal mesh) . . . . .	704
<a href="#">mln::image3d&lt; T &gt;</a> (Basic 3D image class) . . . . .	707
<a href="#">mln::image_if&lt; I, F &gt;</a> ( <a href="#">Image</a> which domain is restricted by a function 'site -> Boolean') . . . . .	712
<a href="#">mln::interpolated&lt; I, F &gt;</a> (Makes the underlying image being accessed with floating coordinates) . . . . .	713
<a href="#">mln::io::dicom::dicom_header</a> (Store dicom file header) . . . . .	715
<a href="#">mln::io::dump::dump_header</a> (Store dump file header) . . . . .	715
<a href="#">mln::io::fld::fld_header</a> (Define the header structure of an AVS field data file) . . . . .	716
<a href="#">mln::io::raw::raw_header</a> (Store raw file header) . . . . .	716
<a href="#">mln::Iterator&lt; E &gt;</a> (Base class for implementation classes that are iterators) . . . . .	716
<a href="#">mln::labeled_image&lt; I &gt;</a> (Morpher providing an improved interface for labeled image) . . . . .	718
<a href="#">mln::labeled_image_base&lt; I, E &gt;</a> (Base class Morpher providing an improved interface for labeled image) . . . . .	721
<a href="#">mln::lazy_image&lt; I, F, B &gt;</a> ( <a href="#">Image</a> values are computed on the fly) . . . . .	725
<a href="#">mln::Literal&lt; E &gt;</a> (Base class for implementation classes of literals) . . . . .	727

<code>mln::literal::black_t</code> (Type of literal black) . . . . .	729
<code>mln::literal::blue_t</code> (Type of literal blue) . . . . .	730
<code>mln::literal::brown_t</code> (Type of literal brown) . . . . .	730
<code>mln::literal::cyan_t</code> (Type of literal cyan) . . . . .	731
<code>mln::literal::green_t</code> (Type of literal green) . . . . .	732
<code>mln::literal::identity_t</code> (Type of literal identity) . . . . .	733
<code>mln::literal::light_gray_t</code> (Type of literal grays) . . . . .	734
<code>mln::literal::lime_t</code> (Type of literal lime) . . . . .	735
<code>mln::literal::magenta_t</code> (Type of literal magenta) . . . . .	736
<code>mln::literal::max_t</code> (Type of literal max) . . . . .	737
<code>mln::literal::min_t</code> (Type of literal min) . . . . .	738
<code>mln::literal::olive_t</code> (Type of literal olive) . . . . .	739
<code>mln::literal::one_t</code> (Type of literal one) . . . . .	740
<code>mln::literal::orange_t</code> (Type of literal orange) . . . . .	741
<code>mln::literal::origin_t</code> (Type of literal origin) . . . . .	742
<code>mln::literal::pink_t</code> (Type of literal pink) . . . . .	743
<code>mln::literal::purple_t</code> (Type of literal purple) . . . . .	744
<code>mln::literal::red_t</code> (Type of literal red) . . . . .	745
<code>mln::literal::teal_t</code> (Type of literal teal) . . . . .	746
<code>mln::literal::violet_t</code> (Type of literal violet) . . . . .	747
<code>mln::literal::white_t</code> (Type of literal white) . . . . .	748
<code>mln::literal::yellow_t</code> (Type of literal yellow) . . . . .	749
<code>mln::literal::zero_t</code> (Type of literal zero) . . . . .	750
<code>mln::Mesh&lt; E &gt;</code> (Base class for implementation classes of meshes) . . . . .	751
<code>mln::Meta_Accumulator&lt; E &gt;</code> (Base class for implementation of meta accumulators) . . . . .	752
<code>mln::Meta_Function&lt; E &gt;</code> (Base class for implementation of meta functions) . . . . .	754
<code>mln::Meta_Function_v2v&lt; E &gt;</code> (Base class for implementation of function-objects from value to value) . . . . .	754
<code>mln::Meta_Function_vv2v&lt; E &gt;</code> (Base class for implementation of function-objects from value to value) . . . . .	755
<code>mln::metal::ands&lt; E1, E2, E3, E4, E5, E6, E7, E8 &gt;</code> (Ands type) . . . . .	756
<code>mln::metal::converts_to&lt; T, U &gt;</code> ("converts-to" check) . . . . .	756
<code>mln::metal::equal&lt; T1, T2 &gt;</code> (Definition of a static 'equal' test) . . . . .	756
<code>mln::metal::goes_to&lt; T, U &gt;</code> ("goes-to" check) . . . . .	757
<code>mln::metal::is&lt; T, U &gt;</code> ("is" check) . . . . .	757
<code>mln::metal::is_a&lt; T, M &gt;</code> ("is_a" check) . . . . .	757
<code>mln::metal::is_not&lt; T, U &gt;</code> ("is_not" check) . . . . .	757
<code>mln::metal::is_not_a&lt; T, M &gt;</code> ("is_not_a" static Boolean expression) . . . . .	758
<code>mln::mixed_neighb&lt; W &gt;</code> (Adapter class from window to neighborhood) . . . . .	758
<code>mln::morpho::attribute::card&lt; I &gt;</code> (Cardinality accumulator class) . . . . .	759
<code>mln::morpho::attribute::count_adjacent_vertices&lt; I &gt;</code> (Count_Adjacent_Vertices accumulator class) . . . . .	761
<code>mln::morpho::attribute::height&lt; I &gt;</code> (Height accumulator class) . . . . .	762
<code>mln::morpho::attribute::sharpness&lt; I &gt;</code> (Sharpness accumulator class) . . . . .	764
<code>mln::morpho::attribute::sum&lt; I, S &gt;</code> (Suminality accumulator class) . . . . .	765
<code>mln::morpho::attribute::volume&lt; I &gt;</code> (Volume accumulator class) . . . . .	767
<code>mln::neighb&lt; W &gt;</code> (Adapter class from window to neighborhood) . . . . .	769
<code>mln::Neighborhood&lt; E &gt;</code> (Base class for implementation classes that are neighborhoods) . . . . .	770
<code>mln::Neighborhood&lt; void &gt;</code> (Neighborhood category flag type) . . . . .	771
<code>mln::Object&lt; E &gt;</code> (Base class for almost every class defined in Milena) . . . . .	771
<code>mln::p2p_image&lt; I, F &gt;</code> (FIXME: Doc!) . . . . .	771
<code>mln::p_array&lt; P &gt;</code> (Multi-set of sites) . . . . .	773
<code>mln::p_centered&lt; W &gt;</code> (Site set corresponding to a window centered on a site) . . . . .	778



<code>mln::p_complex&lt; D, G &gt;</code> (A complex psite set based on the N-faces of a complex of dimension D (a <code>D-complex</code> )) . . . . .	781
<code>mln::p_edges&lt; G, F &gt;</code> (Site set mapping graph edges and image sites) . . . . .	784
<code>mln::p_faces&lt; N, D, P &gt;</code> (A complex psite set based on a the N-faces of a complex of dimension D (a <code>D-complex</code> )) . . . . .	789
<code>mln::p_graph_piter&lt; S, I &gt;</code> (Generic iterator on point sites of a <code>mln::S</code> ) . . . . .	792
<code>mln::p_if&lt; S, F &gt;</code> (Site set restricted w.r.t) . . . . .	793
<code>mln::p_image&lt; I &gt;</code> (Site set based on an image of Booleans) . . . . .	796
<code>mln::p_indexed_bkd_piter&lt; S &gt;</code> (Backward iterator on sites of an indexed site set) . . . . .	800
<code>mln::p_indexed_fwd_piter&lt; S &gt;</code> (Forward iterator on sites of an indexed site set) . . . . .	801
<code>mln::p_indexed_psite&lt; S &gt;</code> (Psite class for indexed site sets such as <code>p_array</code> ) . . . . .	802
<code>mln::p_key&lt; K, P &gt;</code> (Priority queue class) . . . . .	802
<code>mln::p_line2d</code> (2D discrete line of points) . . . . .	807
<code>mln::p_mutable_array_of&lt; S &gt;</code> ( <code>P_mutable_array_of</code> is a mutable array of site sets) . . . . .	811
<code>mln::p_n_faces_bkd_piter&lt; D, G &gt;</code> (Backward iterator on the n-faces sites of an <code>mln::p-complex&lt;D, G&gt;</code> ) . . . . .	814
<code>mln::p_n_faces_fwd_piter&lt; D, G &gt;</code> (Forward iterator on the n-faces sites of an <code>mln::p-complex&lt;D, G&gt;</code> ) . . . . .	815
<code>mln::p_priority&lt; P, Q &gt;</code> (Priority queue) . . . . .	816
<code>mln::p_queue&lt; P &gt;</code> (Queue of sites (based on <code>std::deque</code> )) . . . . .	821
<code>mln::p_queue_fast&lt; P &gt;</code> (Queue of sites class (based on <code>p_array</code> )) . . . . .	825
<code>mln::p_run&lt; P &gt;</code> (Point set class in run) . . . . .	830
<code>mln::p_set&lt; P &gt;</code> (Mathematical set of sites (based on <code>util::set</code> )) . . . . .	834
<code>mln::p_set_of&lt; S &gt;</code> ( <code>P_set_of</code> is a set of site sets) . . . . .	838
<code>mln::p_transformed&lt; S, F &gt;</code> (Site set transformed through a function) . . . . .	841
<code>mln::p_transformed_piter&lt; Pi, S, F &gt;</code> (Iterator on <code>p_transformed&lt;S,F&gt;</code> ) . . . . .	843
<code>mln::p_vaccess&lt; V, S &gt;</code> (Site set in which sites are grouped by their associated value) . . . . .	845
<code>mln::p_vertices&lt; G, F &gt;</code> (Site set based mapping graph vertices to sites) . . . . .	848
<code>mln::pixel&lt; I &gt;</code> (Generic pixel class) . . . . .	853
<code>mln::Pixel_Iterator&lt; E &gt;</code> (Base class for the implementation of pixel iterator classes) . . . . .	855
<code>mln::plain&lt; I &gt;</code> (Prevents an image from sharing its data) . . . . .	856
<code>mln::Point&lt; P &gt;</code> (Base class for implementation of point classes) . . . . .	858
<code>mln::point&lt; G, C &gt;</code> (Generic point class) . . . . .	860
<code>mln::Point_Site&lt; E &gt;</code> (Base class for implementation classes of the notion of "point site") . . . . .	866
<code>mln::Point_Site&lt; void &gt;</code> (Point site category flag type) . . . . .	870
<code>mln::Proxy&lt; E &gt;</code> (Base class for implementation classes of the notion of "proxy") . . . . .	870
<code>mln::Proxy&lt; void &gt;</code> (Proxy category flag type) . . . . .	870
<code>mln::Pseudo_Site&lt; E &gt;</code> (Base class for implementation classes of the notion of "pseudo site") . . . . .	871
<code>mln::Pseudo_Site&lt; void &gt;</code> (Pseudo_Site category flag type) . . . . .	871
<code>mln::pw::image&lt; F, S &gt;</code> (A generic point-wise image implementation) . . . . .	871
<code>mln::registration::closest_point_basic&lt; P &gt;</code> (Closest point functor based on map distance) . . . . .	873
<code>mln::registration::closest_point_with_map&lt; P &gt;</code> (Closest point functor based on map distance) . . . . .	873
<code>mln::Regular_Grid&lt; E &gt;</code> (Base class for implementation classes of regular grids) . . . . .	873
<code>mln::safe_image&lt; I &gt;</code> (Makes an image accessible at undefined location) . . . . .	874
<code>mln::select::p_of&lt; P &gt;</code> (Structure <code>p_of</code> ) . . . . .	875
<code>mln::Site&lt; E &gt;</code> (Base class for classes that are explicitly sites) . . . . .	875
<code>mln::Site&lt; void &gt;</code> (Site category flag type) . . . . .	876
<code>mln::Site_Iterator&lt; E &gt;</code> (Base class for implementation of classes of iterator on points) . . . . .	876
<code>mln::Site_Proxy&lt; E &gt;</code> (Base class for implementation classes of the notion of "site proxy") . . . . .	878
<code>mln::Site_Proxy&lt; void &gt;</code> (Site_Proxy category flag type) . . . . .	878
<code>mln::Site_Set&lt; E &gt;</code> (Base class for implementation classes of site sets) . . . . .	879
<code>mln::Site_Set&lt; void &gt;</code> (Site_Set category flag type) . . . . .	882
<code>mln::slice_image&lt; I &gt;</code> (2D image extracted from a slice of a 3D image) . . . . .	882
<code>mln::sub_image&lt; I, S &gt;</code> (Image having its domain restricted by a site set) . . . . .	884

<code>mln::sub_image_if&lt; I, S &gt;</code> (Image having its domain restricted by a site set and a function ) . . .	886
<code>mln::thru_image&lt; I, F &gt;</code> (Morph image values through a function ) . . . . .	887
<code>mln::thru_bin_image&lt; I1, I2, F &gt;</code> (Morphes values from two images through a binary function ) .	887
<code>mln::topo::adj_higher_dim_connected_n_face_bkd_iter&lt; D &gt;</code> (Backward iterator on all the n-faces sharing an adjacent (n+1)-face with a (reference) n-face of an <code>mln::complex&lt;D&gt;</code> ) . . . . .	889
<code>mln::topo::adj_higher_dim_connected_n_face_fwd_iter&lt; D &gt;</code> (Forward iterator on all the n-faces sharing an adjacent (n+1)-face with a (reference) n-face of an <code>mln::complex&lt;D&gt;</code> ) . . . . .	890
<code>mln::topo::adj_higher_face_bkd_iter&lt; D &gt;</code> (Backward iterator on all the adjacent (n+1)-faces of the n-face of an <code>mln::complex&lt;D&gt;</code> ) . . . . .	891
<code>mln::topo::adj_higher_face_fwd_iter&lt; D &gt;</code> (Forward iterator on all the adjacent (n+1)-faces of the n-face of an <code>mln::complex&lt;D&gt;</code> ) . . . . .	892
<code>mln::topo::adj_lower_dim_connected_n_face_bkd_iter&lt; D &gt;</code> (Backward iterator on all the n-faces sharing an adjacent (n-1)-face with a (reference) n-face of an <code>mln::complex&lt;D&gt;</code> ) . . . . .	893
<code>mln::topo::adj_lower_dim_connected_n_face_fwd_iter&lt; D &gt;</code> (Forward iterator on all the n-faces sharing an adjacent (n-1)-face with a (reference) n-face of an <code>mln::complex&lt;D&gt;</code> ) . . . . .	894
<code>mln::topo::adj_lower_face_bkd_iter&lt; D &gt;</code> (Backward iterator on all the adjacent (n-1)-faces of the n-face of an <code>mln::complex&lt;D&gt;</code> ) . . . . .	895
<code>mln::topo::adj_lower_face_fwd_iter&lt; D &gt;</code> (Forward iterator on all the adjacent (n-1)-faces of the n-face of an <code>mln::complex&lt;D&gt;</code> ) . . . . .	896
<code>mln::topo::adj_lower_higher_face_bkd_iter&lt; D &gt;</code> (Forward iterator on all the adjacent (n-1)-faces and (n+1)-faces of the n-face of an <code>mln::complex&lt;D&gt;</code> ) . . . . .	897
<code>mln::topo::adj_lower_higher_face_fwd_iter&lt; D &gt;</code> (Forward iterator on all the adjacent (n-1)-faces and (n+1)-faces of the n-face of an <code>mln::complex&lt;D&gt;</code> ) . . . . .	898
<code>mln::topo::adj_m_face_bkd_iter&lt; D &gt;</code> (Backward iterator on all the m-faces transitively adjacent to a (reference) n-face in a complex ) . . . . .	899
<code>mln::topo::adj_m_face_fwd_iter&lt; D &gt;</code> (Forward iterator on all the m-faces transitively adjacent to a (reference) n-face in a complex ) . . . . .	901
<code>mln::topo::algebraic_face&lt; D &gt;</code> (Algebraic face handle in a complex; the face dimension is dynamic ) . . . . .	902
<code>mln::topo::algebraic_n_face&lt; N, D &gt;</code> (Algebraic N-face handle in a complex ) . . . . .	907
<code>mln::topo::center_only_iter&lt; D &gt;</code> (Iterator on all the adjacent (n-1)-faces of the n-face of an <code>mln::complex&lt;D&gt;</code> ) . . . . .	911
<code>mln::topo::centered_bkd_iter_adapter&lt; D, I &gt;</code> (Forward complex relative iterator adapters adding the central (reference) point to the set of iterated faces ) . . . . .	912
<code>mln::topo::centered_fwd_iter_adapter&lt; D, I &gt;</code> (Backward complex relative iterator adapters adding the central (reference) point to the set of iterated faces ) . . . . .	913
<code>mln::topo::complex&lt; D &gt;</code> (General complex of dimension D ) . . . . .	914
<code>mln::topo::face&lt; D &gt;</code> (Face handle in a complex; the face dimension is dynamic ) . . . . .	917
<code>mln::topo::face_bkd_iter&lt; D &gt;</code> (Backward iterator on all the faces of an <code>mln::complex&lt;D&gt;</code> ) .	920
<code>mln::topo::face_fwd_iter&lt; D &gt;</code> (Forward iterator on all the faces of an <code>mln::complex&lt;D&gt;</code> ) . .	922
<code>mln::topo::is_n_face&lt; N &gt;</code> (A functor testing wheter a <code>mln::complex_psite</code> is an N-face ) . . .	923
<code>mln::topo::is_simple_cell&lt; I &gt;</code> (A predicate for the simplicity of a point based on the collapse property of the attachment ) . . . . .	924
<code>mln::topo::n_face&lt; N, D &gt;</code> (N-face handle in a complex ) . . . . .	927
<code>mln::topo::n_face_bkd_iter&lt; D &gt;</code> (Backward iterator on all the faces of an <code>mln::complex&lt;D&gt;</code> )	930
<code>mln::topo::n_face_fwd_iter&lt; D &gt;</code> (Forward iterator on all the faces of an <code>mln::complex&lt;D&gt;</code> ) .	932
<code>mln::topo::n_faces_set&lt; N, D &gt;</code> (Set of face handles of dimension N ) . . . . .	933
<code>mln::topo::static_n_face_bkd_iter&lt; N, D &gt;</code> (Backward iterator on all the N-faces of a <code>mln::complex&lt;D&gt;</code> ) . . . . .	934

<code>mln::topo::static_n_face_fwd_iter&lt; N, D &gt;</code> (Forward iterator on all the N-faces of a <code>mln::complex&lt;D&gt;</code> ) . . . . .	936
<code>mln::tr_image&lt; S, I, T &gt;</code> (Transform an image by a given transformation) . . . . .	937
<code>mln::transformed_image&lt; I, F &gt;</code> ( <b>Image</b> having its domain restricted by a site set) . . . . .	940
<code>mln::unproject_image&lt; I, D, F &gt;</code> (Un-projects an image) . . . . .	941
<code>mln::util::adjacency_matrix&lt; V &gt;</code> (A class of adjacency matrix) . . . . .	943
<code>mln::util::array&lt; T &gt;</code> (A dynamic array class) . . . . .	943
<code>mln::util::branch&lt; T &gt;</code> (Class of generic branch) . . . . .	949
<code>mln::util::branch_iter&lt; T &gt;</code> (Basic 2D image class) . . . . .	951
<code>mln::util::branch_iter_ind&lt; T &gt;</code> (Basic 2D image class) . . . . .	952
<code>mln::util::couple&lt; T, U &gt;</code> (Definition of a couple) . . . . .	953
<code>mln::util::eat</code> (Eat structure) . . . . .	955
<code>mln::util::edge&lt; G &gt;</code> ( <b>Edge</b> of a graph <code>G</code> ) . . . . .	956
<code>mln::util::fibonacci_heap&lt; P, T &gt;</code> (Fibonacci heap) . . . . .	960
<code>mln::util::graph</code> (Undirected graph) . . . . .	963
<code>mln::util::greater_point&lt; I &gt;</code> (A "greater than" functor comparing points w.r.t) . . . . .	969
<code>mln::util::greater_psite&lt; I &gt;</code> (A "greater than" functor comparing psites w.r.t) . . . . .	970
<code>mln::util::head&lt; T, R &gt;</code> (Top structure of the soft heap) . . . . .	970
<code>mln::util::ignore</code> (Ignore structure) . . . . .	971
<code>mln::util::ilcell&lt; T &gt;</code> (Element of an item list. Store the data (key) used in <code>soft_heap</code> ) . . . . .	971
<code>mln::util::line_graph&lt; G &gt;</code> (Undirected line graph of a graph of type <code>G</code> ) . . . . .	971
<code>mln::util::nil</code> (Nil structure) . . . . .	977
<code>mln::util::node&lt; T, R &gt;</code> (Meta-data of an element in the heap) . . . . .	978
<code>mln::util::object_id&lt; Tag, V &gt;</code> (Base class of an object id) . . . . .	978
<code>mln::util::ord&lt; T &gt;</code> (Function-object that defines an ordering between objects with type <code>T</code> : <i>lhs</i> <code>R rhs</code> ) . . . . .	980
<code>mln::util::ord_pair&lt; T &gt;</code> (Ordered pair structure s.a) . . . . .	980
<code>mln::util::pix&lt; I &gt;</code> (Structure <code>pix</code> ) . . . . .	982
<code>mln::util::set&lt; T &gt;</code> (An "efficient" mathematical set class) . . . . .	984
<code>mln::util::site_pair&lt; P &gt;</code> (A pair of sites) . . . . .	990
<code>mln::util::soft_heap&lt; T, R &gt;</code> (Soft heap) . . . . .	991
<code>mln::util::timer</code> (Timer structure) . . . . .	994
<code>mln::util::tracked_ptr&lt; T &gt;</code> (Smart pointer for shared data with tracking) . . . . .	995
<code>mln::util::tree&lt; T &gt;</code> (Class of generic tree) . . . . .	997
<code>mln::util::tree_node&lt; T &gt;</code> (Class of generic <code>tree_node</code> for tree) . . . . .	999
<code>mln::util::vertex&lt; G &gt;</code> ( <b>Vertex</b> of a graph <code>G</code> ) . . . . .	1004
<code>mln::util::yes</code> (Object that always says "yes") . . . . .	1008
<code>mln::Value&lt; E &gt;</code> (Base class for implementation classes of values) . . . . .	1008
<code>mln::value::float01</code> (Class for floating values restricted to the interval <code>[0..1]</code> and discretized with <code>n</code> bits) . . . . .	1010
<code>mln::value::float01_f</code> (Class for floating values restricted to the interval <code>[0..1]</code> ) . . . . .	1012
<code>mln::value::graylevel&lt; n &gt;</code> (General gray-level class on <code>n</code> bits) . . . . .	1013
<code>mln::value::graylevel_f</code> (General gray-level class on <code>n</code> bits) . . . . .	1016
<code>mln::value::int_s&lt; n &gt;</code> (Signed integer value class) . . . . .	1018
<code>mln::value::int_u&lt; n &gt;</code> (Unsigned integer value class) . . . . .	1020
<code>mln::value::int_u_sat&lt; n &gt;</code> (Unsigned integer value class with saturation behavior) . . . . .	1022
<code>mln::value::Integer&lt; E &gt;</code> (Concept of integer) . . . . .	1024
<code>mln::value::Integer&lt; void &gt;</code> (Category flag type) . . . . .	1025
<code>mln::value::label&lt; n &gt;</code> (Label value class) . . . . .	1025
<code>mln::value::lut_vec&lt; S, T &gt;</code> (Class that defines <code>FIXME</code> ) . . . . .	1028
<code>mln::value::proxy&lt; I &gt;</code> (Generic proxy class for an image pixel value) . . . . .	1031
<code>mln::value::qt::rgb32</code> (Color class for red-green-blue where every component is <code>n</code> -bit encoded) . . . . .	1033
<code>mln::value::rgb&lt; n &gt;</code> (Color class for red-green-blue where every component is <code>n</code> -bit encoded) . . . . .	1035
<code>mln::value::set&lt; T &gt;</code> (Class that defines the set of values of type <code>T</code> ) . . . . .	1036

<code>mln::value::sign</code> (Value type composed by the set (-1, 0, 1) sign value type is a subset of the int value type) . . . . .	1037
<code>mln::value::stack_image&lt; n, I &gt;</code> (Stack image class) . . . . .	1039
<code>mln::value::super_value&lt; sign &gt;</code> (Specializations: ) . . . . .	1041
<code>mln::value::value_array&lt; T, V &gt;</code> (Generic array class over indexed by a value set with type T) . . . . .	1041
<code>mln::Value_Iterator&lt; E &gt;</code> (Base class for implementation of classes of iterator on values) . . . . .	1043
<code>mln::Value_Set&lt; E &gt;</code> (Base class for implementation classes of sets of values) . . . . .	1044
<code>mln::Vertex&lt; E &gt;</code> (Vertex category flag type) . . . . .	1045
<code>mln::vertex_image&lt; P, V, G &gt;</code> (Image based on graph vertices) . . . . .	1046
<code>mln::violent_cast_image&lt; T, I &gt;</code> (Violently cast image values to a given type) . . . . .	1048
<code>mln::w_window&lt; D, W &gt;</code> (Generic <code>w_window</code> class) . . . . .	1049
<code>mln::Weighted_Window&lt; E &gt;</code> (Base class for implementation classes that are weighted-windows) . . . . .	1053
<code>mln::win::backdiag2d</code> (Diagonal line window defined on the 2D square grid) . . . . .	1054
<code>mln::win::ball&lt; G, C &gt;</code> (Generic ball window defined on a given grid) . . . . .	1055
<code>mln::win::cube3d</code> (Cube window defined on the 3D grid) . . . . .	1056
<code>mln::win::cuboid3d</code> (Cuboid defined on the 3-D square grid) . . . . .	1057
<code>mln::win::diag2d</code> (Diagonal line window defined on the 2D square grid) . . . . .	1059
<code>mln::win::line&lt; M, i, C &gt;</code> (Generic line window defined on a given grid in the given dimension) . . . . .	1060
<code>mln::win::multiple&lt; W, F &gt;</code> (Multiple window) . . . . .	1061
<code>mln::win::multiple_size&lt; n, W, F &gt;</code> (Definition of a multiple-size window) . . . . .	1062
<code>mln::win::octagon2d</code> (Octagon window defined on the 2D square grid) . . . . .	1062
<code>mln::win::rectangle2d</code> (Rectangular window defined on the 2D square grid) . . . . .	1063
<code>mln::Window&lt; E &gt;</code> (Base class for implementation classes that are windows) . . . . .	1065
<code>mln::window&lt; D &gt;</code> (Generic window class) . . . . .	1065
<code>mln::world::inter_pixel::is_separator</code> (Functor returning whether a site is a separator in an inter-pixel image) . . . . .	1069
<code>trait::graph&lt; I &gt;</code> (Graph traits) . . . . .	1070
<code>trait::graph&lt; mln::complex_image&lt; 1, G, V &gt;&gt;</code> (Graph traits for 1-complexes images) . . . . .	1071
<code>trait::graph&lt; mln::image2d&lt; T &gt;&gt;</code> (Graph traits for <code>mln::image2d</code> ) . . . . .	1071

# Chapter 8

## Module Documentation

### 8.1 On site sets

Accumulators working on site sets.

#### Classes

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

#### 8.1.1 Detailed Description

Accumulators working on site sets.

### 8.2 On images

Accumulators working on images.

#### Classes

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

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

### 8.2.1 Detailed Description

Accumulators working on images.

## 8.3 On values

Accumulators working on image values.

### Classes

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

- struct `mln::accu::math::inf< T >`  
*Generic inf accumulator class.*
- struct `mln::accu::math::sum< T, S >`  
*Generic sum accumulator class.*
- struct `mln::accu::math::sup< T >`  
*Generic sup accumulator class.*
- struct `mln::accu::rms< T, V >`  
*Generic root mean square accumulator class.*
- struct `mln::accu::stat::deviation< T, S, M >`  
*Generic standard deviation accumulator class.*
- struct `mln::accu::stat::max< T >`  
*Generic max accumulator class.*
- struct `mln::accu::stat::max_h< V >`  
*Generic max function based on histogram over a value set with type V.*
- struct `mln::accu::stat::mean< T, S, M >`  
*Generic mean accumulator class.*
- struct `mln::accu::stat::median_alt< S >`  
*Generic `median_alt` function based on histogram over a value set with type S.*
- struct `mln::accu::stat::median_h< V >`  
*Generic median function based on histogram over a value set with type V.*
- struct `mln::accu::stat::min< T >`  
*Generic min accumulator class.*
- struct `mln::accu::stat::min_h< V >`  
*Generic min function based on histogram over a value set with type V.*
- struct `mln::accu::stat::min_max< V >`  
*Generic min and max accumulator class.*
- struct `mln::accu::stat::rank< T >`  
*Generic rank accumulator class.*
- struct `mln::accu::stat::rank< bool >`  
*rank accumulator class for Boolean.*
- struct `mln::accu::stat::rank_high_quant< T >`  
*Generic rank accumulator class.*
- struct `mln::accu::stat::var< T >`  
*Var accumulator class.*

- struct `mln::accu::stat::variance< T, S, R >`  
*Variance accumulator class.*

### 8.3.1 Detailed Description

Accumulators working on image values.

## 8.4 Multiple accumulators

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

### Classes

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

### 8.4.1 Detailed Description

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

## 8.5 Graphes

All graphes implementations.

### Classes

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

### 8.5.1 Detailed Description

All graphes implementations.



## 8.6 Images

All the generic image types provided in Olena.

### Modules

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

### 8.6.1 Detailed Description

All the generic image types provided in Olena.

## 8.7 Basic types

Concrete images.

### Classes

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

### 8.7.1 Detailed Description

Concrete images.

## 8.8 Image morphers

Morpher on both image values and domain.

Morpher on both image values and domain.

## 8.9 Values morphers

Morpher on image values.

### Classes

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

### 8.9.1 Detailed Description

Morpher on image values.

## 8.10 Domain morphers

Morpher on image domain.

### Classes

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

### 8.10.1 Detailed Description

Morpher on image domain.

## 8.11 Identity morphers

Morpher adding new fonctionnalités.

### Classes

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

### 8.11.1 Detailed Description

Morpher adding new fonctionnalités.

## 8.12 Types

Milena Object types.

### Modules

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

- [Utilities](#)

*Miscellaneous useful containers/structures.*

- [Windows](#)

*All the predefined generic windows.*

### 8.12.1 Detailed Description

Milena Object types.

## 8.13 Accumulators

All accumulator types.

### Modules

- [On site sets](#)

*Accumulators working on site sets.*

- [On images](#)

*Accumulators working on images.*

- [On values](#)

*Accumulators working on image values.*

- [Multiple accumulators](#)

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

### 8.13.1 Detailed Description

All accumulator types.

## 8.14 Routines

All algorithms/routines provided in Milena.

All algorithms/routines provided in Milena.

## 8.15 Canvas

All canvas.

All canvas.

## 8.16 Functions

All predefined functions.

### Classes

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

### Namespaces

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

### Modules

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

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

### 8.16.1 Detailed Description

All predefined functions.

## 8.17 Neighborhoods

All the predefined generic neighborhoods.

### Modules

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

### 8.17.1 Detailed Description

All the predefined generic neighborhoods.

## 8.18 1D neighborhoods

Predefined 1D neighborhoods.

### Typedefs

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

### Functions

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

### 8.18.1 Detailed Description

Predefined 1D neighborhoods.

### 8.18.2 Typedef Documentation

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

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

### 8.18.3 Function Documentation

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

2-connectivity neighborhood on the 1D grid.

○ × ○

#### Returns

A neighb1d.

## 8.19 2D neighborhoods

Predefined 2D neighborhoods.

### Typedefs

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

### Functions

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



## 8.19.1 Detailed Description

Predefined 2D neighborhoods.

## 8.19.2 Typedef Documentation

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

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

## 8.19.3 Function Documentation

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

Vertical 2-connectivity neighborhood on the 2D grid.

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

#### Returns

A neighb2d.

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

Horizontal 2-connectivity neighborhood on the 2D grid.

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

#### Returns

A neighb2d.

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

4-connectivity neighborhood on the 2D grid.

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

#### Returns

A neighb2d.

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

8-connectivity neighborhood on the 2D grid.

```

○ ○ ○
○ × ○
○ ○ ○

```

#### Returns

A `neighb2d`.

## 8.20 3D neighborhoods

Predefined 3D neighborhoods.

### Typedefs

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

### Functions

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

### 8.20.1 Detailed Description

Predefined 3D neighborhoods.

## 8.20.2 Typedef Documentation

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

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

## 8.20.3 Function Documentation

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

18-connectivity neighborhood on the 3D grid.

```

  . o .
  o o o
  . o .

  o o o
  o x o
  o o o

  . o .
  o o o
  . o .

```

#### Returns

A neighb3d.

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

Referenced by mln::c26().

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

26-connectivity neighborhood on the 3D grid.

```

  o o o
  o o o
  o o o

  o o o
  o x o
  o o o

  o o o
  o o o
  o o o

```

#### Returns

A neighb3d.

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

**8.20.3.3 const neighb3d & mln::c2\_3d\_sli ( ) [inline]**

depth 2-connectivity neighborhood on the 3D grid.

```

. . .
. o .
. . .

. . .
. x .
. . .

. . .
. o .
. . .

```

**Returns**

A neighb3d.

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

**8.20.3.4 const neighb3d & mln::c4\_3d ( ) [inline]**

4-connectivity neighborhood on the 3D grid.

```

. . .
. . .
. . .

. o .
o x o
. o .

. . .
. . .
. . .

```

**Returns**

A neighb3d.

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

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

6-connectivity neighborhood on the 3D grid.

```

. . .
. o .
. . .

```

```

. o .
o x o
. o .

. . .
. o .
. . .

```

**Returns**

A `neighb3d`.

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

Referenced by `mln::c18()`.

**8.20.3.6 const neighb3d & mln::c8\_3d ( ) [inline]**

8-connectivity neighborhood on the 3D grid.

```

. . .
. . .
. . .

o o o
o x o
o o o

. . .
. . .
. . .

```

**Returns**

A `neighb3d`.

**8.21 Site sets**

All Site set types.

**Modules**

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

- [Sparse types](#)  
*Sparse site sets.*

- [Queue based](#)  
*Site sets based on a queue.*

### 8.21.1 Detailed Description

All Site set types.

## 8.22 Basic types

Basic site sets.

### Classes

- 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.*

### 8.22.1 Detailed Description

Basic site sets.

## 8.23 Graph based

Site sets based on a graph.

### Classes

- class [mln::p\\_edges< G, F >](#)  
*Site set mapping graph edges and image sites.*
- struct [mln::p\\_faces< N, D, P >](#)  
*A complex psite set based on 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.*

### 8.23.1 Detailed Description

Site sets based on a graph.

## 8.24 Complex based

Site sets based on a complexes.

### Classes

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

### 8.24.1 Detailed Description

Site sets based on a complexes.

## 8.25 Sparse types

Sparse site sets.

### Classes

- class `mln::p_array< P >`  
*Multi-set of sites.*
- class `mln::p_centered< W >`  
*Site set corresponding to a window centered on a site.*
- class `mln::p_if< S, F >`  
*Site set restricted w.r.t.*
- class `mln::p_image< I >`  
*Site set based on an image of Booleans.*
- class `mln::p_set< P >`  
*Mathematical set of sites (based on `util::set`).*
- class `mln::p_transformed< S, F >`

*Site set transformed through a function.*

- class `mln::p_vaccess< V, S >`  
*Site set in which sites are grouped by their associated value.*

### 8.25.1 Detailed Description

Sparse site sets.

## 8.26 Queue based

Site sets based on a queue.

### Classes

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

### 8.26.1 Detailed Description

Site sets based on a queue.

## 8.27 Utilities

Miscellaneous useful containers/structures.

### Classes

- class `mln::util::adjacency_matrix< V >`  
*A class of adjacency matrix.*
- class `mln::util::array< T >`  
*A dynamic array class.*
- class `mln::util::couple< T, U >`



*Definition of a couple.*

- struct [mln::util::eat](#)  
*Eat structure.*
- class [mln::util::fibonacci\\_heap< P, T >](#)  
*Fibonacci heap.*
- struct [mln::util::ignore](#)  
*Ignore structure.*
- struct [mln::util::nil](#)  
*Nil structure.*
- struct [mln::util::ord\\_pair< T >](#)  
*Ordered pair structure s.a.*
- class [mln::util::set< T >](#)  
*An "efficient" mathematical set class.*
- class [mln::util::site\\_pair< P >](#)  
*A pair of sites.*
- class [mln::util::soft\\_heap< T, R >](#)  
*Soft heap.*
- struct [mln::util::tracked\\_ptr< T >](#)  
*Smart pointer for shared data with tracking.*
- struct [mln::util::yes](#)  
*Object that always says "yes".*

### 8.27.1 Detailed Description

Miscellaneous useful containers/structures.

## 8.28 Windows

All the predefined generic windows.

### Modules

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

- [3D windows](#)

*Predefined 3D windows.*

- [N-D windows](#)

*Predefined N-D windows.*

- [Multiple windows](#)

*Generic multiple windows.*

### 8.28.1 Detailed Description

All the predefined generic windows.

## 8.29 1D windows

Predefined 1D windows.

### Typedefs

- typedef line< grid::tick, 0, def::coord > [mln::win::segment1d](#)

*Segment window defined on the 1D grid.*

- typedef window< [mln::dpoint1d](#) > [mln::window1d](#)

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

### 8.29.1 Detailed Description

Predefined 1D windows.

### 8.29.2 Typedef Documentation

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

Segment window defined on the 1D grid.

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

For instance:

○ x ○

is defined with length = 3.

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

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

## 8.30 2D windows

Predefined 2D windows.

### Classes

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

### Typedefs

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

### Functions

- const window2d & [mln::win\\_c4p](#) ()  
*4-connectivity window on the 2D grid, including the center.*
- const window2d & [mln::win\\_c8p](#) ()  
*8-connectivity window on the 2D grid, including the center.*

### 8.30.1 Detailed Description

Predefined 2D windows.

### 8.30.2 Typedef Documentation

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

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

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

Horizontal line window defined on the 2D square grid.

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

For instance:

```
o o x o o
```

is defined with length = 5.

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

Vertical line window defined on the 2D square grid.

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

For instance:

```
o
x
o
```

is defined with length = 3.

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

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

### 8.30.3 Function Documentation

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

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

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

**Returns**

A `window2d`.

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

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

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

```

○ ○ ○
○ × ○
○ ○ ○

```

**Returns**

A `window2d`.

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

**8.31 3D windows**

Predefined 3D windows.

**Classes**

- struct `mln::win::cube3d`  
*Cube window defined on the 3D grid.*
- struct `mln::win::cuboid3d`  
*Cuboid defined on the 3-D square grid.*

**Typedefs**

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

## Functions

- `const window3d & mln::win_c4p_3d ()`  
*4-connectivity window on the 3D grid, including the center.*
- `const window3d & mln::win_c8p_3d ()`  
*8-connectivity window on the 3D grid, including the center.*

### 8.31.1 Detailed Description

Predefined 3D windows.

### 8.31.2 Typedef Documentation

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

Depth line window defined on the 3D cubic grid.

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

For instance:

```

. . .
. o .
. . .

. . .
. x .
. . .

. . .
. o .
. . .

```

is defined with `length = 3`.

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

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

#### 8.31.2.3 `typedef window<mln::dpoint3d> mln::window3d`

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

### 8.31.3 Function Documentation

#### 8.31.3.1 `const window3d & mln::win_c4p_3d ( ) [inline]`

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

```

- - -
- - -
- - -

- o -
o x o
- o -

- - -
- - -
- - -

```

**Returns**

A window3d.

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

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

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

```

- - -
- - -
- - -

o o o
o x o
o o o

- - -
- - -
- - -

```

**Returns**

A window3d.

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

**8.32 N-D windows**

Predefined N-D windows.

**Classes**

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

### 8.32.1 Detailed Description

Predefined N-D windows.

## 8.33 Multiple windows

Generic multiple windows.

### Classes

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

### 8.33.1 Detailed Description

Generic multiple windows.

## 8.34 v2w2v functions

All bijective functions.

All bijective functions.

## 8.35 v2w\_w2v functions

All bijective function.

All bijective function.

## 8.36 vv2b functions

All functions mapping two values to a logical value.

All functions mapping two values to a logical value.



# Chapter 9

## Namespace Documentation

### 9.1 mln Namespace Reference

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

#### Namespaces

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

*Namespace for core definitions.*

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

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

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

## Classes

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

- class [complex\\_neighborhood\\_fwd\\_piter](#)  
*Forward iterator on complex neighborhood.*
- class [complex\\_psite](#)  
*Point site associated to a `mln::p_complex`.*
- class [complex\\_window\\_bkd\\_piter](#)  
*Backward iterator on complex window.*
- class [complex\\_window\\_fwd\\_piter](#)  
*Forward iterator on complex window.*
- struct [decorated\\_image](#)  
*Image that can have additional features.*
- struct [Delta\\_Point\\_Site](#)  
*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.*
- class [faces\\_psite](#)  
*Point site associated to a `mln::p_faces`.*
- struct [flat\\_image](#)  
*Image with a single value.*
- struct [fun\\_image](#)  
*Image read through a function.*
- struct [Function](#)  
*Base class for implementation of function-objects.*
- struct [Function< void >](#)  
*Function category flag type.*
- struct [Function\\_n2v](#)  
*Base class for implementation of function-objects from Nil to value.*
- struct [Function\\_v2b](#)  
*Base class for implementation of function-objects from a value to a Boolean.*
- struct [Function\\_v2v](#)  
*Base class for implementation of function-objects from value to value.*
- struct [Function\\_vv2b](#)  
*Base class for implementation of function-objects from a couple of values to a Boolean.*
- struct [Function\\_vv2v](#)  
*Base class for implementation of function-objects from a couple of values to a value.*
- class [fwd\\_pixter1d](#)  
*Forward pixel iterator on a 1-D image with border.*
- class [fwd\\_pixter2d](#)  
*Forward pixel iterator on a 2-D image with border.*
- class [fwd\\_pixter3d](#)  
*Forward pixel iterator on a 3-D image with border.*
- struct [Gdpoint](#)  
*FIXME: Doc!*
- struct [Gdpoint< void >](#)

*Delta point site category flag type.*

- struct [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\\_mixed\\_neighborhood](#)  
*Elementary neighborhood on graph class.*
- class [graph\\_elt\\_mixed\\_window](#)  
*Elementary window on graph class.*
- struct [graph\\_elt\\_neighborhood](#)  
*Elementary neighborhood on graph class.*
- struct [graph\\_elt\\_neighborhood\\_if](#)  
*Elementary neighborhood\_if on graph class.*
- class [graph\\_elt\\_window](#)  
*Elementary window on graph class.*
- class [graph\\_elt\\_window\\_if](#)  
*Custom window on graph class.*
- class [graph\\_window\\_base](#)
- class [graph\\_window\\_if\\_piter](#)  
*Forward iterator on line graph window.*
- class [graph\\_window\\_piter](#)  
*Forward iterator on line graph window.*
- struct [hexa](#)  
*hexagonal image class.*
- struct [Image](#)  
*Base class for implementation of image classes.*
- struct [image1d](#)  
*Basic 1D image class.*
- class [image2d](#)  
*Basic 2D image class.*
- struct [image2d\\_h](#)  
*2d image based on an hexagonal mesh.*

- struct [image3d](#)  
*Basic 3D image class.*
- struct [image\\_if](#)  
*Image which domain is restricted by a function 'site -> Boolean'.*
- struct [interpolated](#)  
*Makes the underlying image being accessed with floating coordinates.*
- struct [Iterator](#)  
*Base class for implementation classes that are iterators.*
- class [labeled\\_image](#)  
*Morpher providing an improved interface for labeled image.*
- class [labeled\\_image\\_base](#)  
*Base class Morpher providing an improved interface for labeled image.*
- struct [lazy\\_image](#)  
*Image values are computed on the fly.*
- struct [Literal](#)  
*Base class for implementation classes of literals.*
- struct [Mesh](#)  
*Base class for implementation classes of meshes.*
- struct [Meta\\_Accumulator](#)  
*Base class for implementation of meta accumulators.*
- struct [Meta\\_Function](#)  
*Base class for implementation of meta functions.*
- struct [Meta\\_Function\\_v2v](#)  
*Base class for implementation of function-objects from value to value.*
- struct [Meta\\_Function\\_vv2v](#)  
*Base class for implementation of function-objects from value to value.*
- class [mixed\\_neighb](#)  
*Adapter class from window to neighborhood.*
- class [neighb](#)  
*Adapter class from window to neighborhood.*
- struct [Neighborhood](#)  
*Base class for implementation classes that are neighborhoods.*
- struct [Neighborhood< void >](#)



*Neighborhood* category flag type.

- struct [Object](#)  
*Base class for almost every class defined in Milena.*
- struct [p2p\\_image](#)  
*FIXME: Doc!*
- class [p\\_array](#)  
*Multi-set of sites.*
- class [p\\_centered](#)  
*Site set corresponding to a window centered on a site.*
- class [p\\_complex](#)  
*A complex psite set based on the N-faces of a complex of dimension D (a D-complex).*
- class [p\\_edges](#)  
*Site set mapping graph edges and image sites.*
- struct [p\\_faces](#)  
*A complex psite set based on a the N-faces of a complex of dimension D (a D-complex).*
- class [p\\_graph\\_piter](#)  
*Generic iterator on point sites of a `mln::S`.*
- class [p\\_if](#)  
*Site set restricted w.r.t.*
- class [p\\_image](#)  
*Site set based on an image of Booleans.*
- class [p\\_indexed\\_bkd\\_piter](#)  
*Backward iterator on sites of an indexed site set.*
- class [p\\_indexed\\_fwd\\_piter](#)  
*Forward iterator on sites of an indexed site set.*
- class [p\\_indexed\\_psite](#)  
*Psite class for indexed site sets such as [p\\_array](#).*
- class [p\\_key](#)  
*Priority queue class.*
- class [p\\_line2d](#)  
*2D discrete line of points.*
- class [p\\_mutable\\_array\\_of](#)  
*[p\\_mutable\\_array\\_of](#) is a mutable array of site sets.*

- class [p\\_n\\_faces\\_bkd\\_piter](#)  
*Backward iterator on the n-faces sites of an `mln::p_complex<D, G>`.*
- class [p\\_n\\_faces\\_fwd\\_piter](#)  
*Forward iterator on the n-faces sites of an `mln::p_complex<D, G>`.*
- class [p\\_priority](#)  
*Priority queue.*
- class [p\\_queue](#)  
*Queue of sites (based on `std::deque`).*
- class [p\\_queue\\_fast](#)  
*Queue of sites class (based on `p_array`).*
- class [p\\_run](#)  
*Point set class in run.*
- class [p\\_set](#)  
*Mathematical set of sites (based on `util::set`).*
- class [p\\_set\\_of](#)  
*`p_set_of` is a set of site sets.*
- class [p\\_transformed](#)  
*Site set transformed through a function.*
- struct [p\\_transformed\\_piter](#)  
*Iterator on `p_transformed<S,F>`.*
- class [p\\_vaccess](#)  
*Site set in which sites are grouped by their associated value.*
- class [p\\_vertices](#)  
*Site set based mapping graph vertices to sites.*
- struct [pixel](#)  
*Generic pixel class.*
- struct [Pixel\\_Iterator](#)  
*Base class for the implementation of pixel iterator classes.*
- class [plain](#)  
*Prevents an image from sharing its data.*
- struct [point](#)  
*Generic point class.*
- struct [Point](#)  
*Base class for implementation of point classes.*

- struct [Point\\_Site](#)  
*Base class for implementation classes of the notion of "point site".*
- struct [Point\\_Site< void >](#)  
*Point site category flag type.*
- struct [Proxy](#)  
*Base class for implementation classes of the notion of "proxy".*
- struct [Proxy< void >](#)  
*Proxy category flag type.*
- struct [Pseudo\\_Site](#)  
*Base class for implementation classes of the notion of "pseudo site".*
- struct [Pseudo\\_Site< void >](#)  
*Pseudo\_Site category flag type.*
- struct [Regular\\_Grid](#)  
*Base class for implementation classes of regular grids.*
- class [safe\\_image](#)  
*Makes an image accessible at undefined location.*
- struct [Site](#)  
*Base class for classes that are explicitly sites.*
- struct [Site< void >](#)  
*Site category flag type.*
- struct [Site\\_Iterator](#)  
*Base class for implementation of classes of iterator on points.*
- struct [Site\\_Proxy](#)  
*Base class for implementation classes of the notion of "site proxy".*
- struct [Site\\_Proxy< void >](#)  
*Site\_Proxy category flag type.*
- struct [Site\\_Set](#)  
*Base class for implementation classes of site sets.*
- struct [Site\\_Set< void >](#)  
*Site\_Set category flag type.*
- struct [slice\\_image](#)  
*2D image extracted from a slice of a 3D image.*
- 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 [Value\\_Iterator](#)  
*Base class for implementation of classes of iterator on values.*
- struct [Value\\_Set](#)  
*Base class for implementation classes of sets of values.*
- struct [Vertex](#)  
*Vertex category flag type.*
- class [vertex\\_image](#)  
*Image based on graph vertices.*
- struct [violent\\_cast\\_image](#)  
*Violently cast image values to a given type.*
- struct [w\\_window](#)  
*Generic `w_window` class.*
- struct [Weighted\\_Window](#)  
*Base class for implementation classes that are `weighted_windows`.*
- class [window](#)  
*Generic window class.*
- struct [Window](#)  
*Base class for implementation classes that are windows.*

## Typedefs

- typedef [mln::complex\\_image](#)< 1, [mln::discrete\\_plane\\_1complex\\_geometry](#), bool > [bin\\_1complex\\_image2d](#)  
*Type alias for a binary image based on a 1-complex, where 0-faces are located at discrete (integer) 2-dimensional points.*
- typedef [mln::complex\\_image](#)< 2, [mln::space\\_2complex\\_geometry](#), bool > [bin\\_2complex\\_image3df](#)  
*Type alias for a binary image based on a 2-complex, where 0-faces are located at floating-point 3-dimensional points.*
- typedef [box](#)< [mln::point1d](#) > [box1d](#)  
*Type alias for a box defined on the 1D square grid with integer coordinates.*
- typedef [box](#)< [mln::point2d](#) > [box2d](#)  
*Type alias for a box defined on the 2D square grid with integer coordinates.*
- typedef [box](#)< [point2d\\_h](#) > [box2d\\_h](#)  
*FIXME.*
- typedef [box](#)< [point3d](#) > [box3d](#)  
*Type alias for a box defined on the 3D square grid with integer coordinates.*
- typedef [mln::geom::complex\\_geometry](#)< 1, [point2d](#) > [discrete\\_plane\\_1complex\\_geometry](#)  
*Type alias for the geometry of a 1-complex (e.g., a graph) located in a discrete 2-dimensional plane (with integer coordinates).*
- typedef [mln::geom::complex\\_geometry](#)< 2, [point2d](#) > [discrete\\_plane\\_2complex\\_geometry](#)  
*Type alias for the geometry of a 2-complex located in a discrete 2-dimensional plane (with integer coordinates).*
- typedef [dpoint](#)< [mln::grid::tick](#), [def::coord](#) > [dpoint1d](#)  
*Type alias for a delta-point defined on the 1D square grid with integer coordinates.*
- typedef [dpoint](#)< [mln::grid::square](#), [mln::def::coord](#) > [dpoint2d](#)  
*Type alias for a delta-point defined on the 2D square grid with integer coordinates.*
- typedef [dpoint](#)< [mln::grid::hexa](#), [def::coord](#) > [dpoint2d\\_h](#)  
*Type alias for a delta-point defined on the 2D square grid with integer coordinates.*
- typedef [dpoint](#)< [mln::grid::cube](#), [def::coord](#) > [dpoint3d](#)  
*Type alias for a delta-point defined on the 3D square grid with integer coordinates.*
- typedef [mln::complex\\_image](#)< 2, [mln::space\\_2complex\\_geometry](#), float > [float\\_2complex\\_image3df](#)  
*Type alias for a floating-point image based on a 2-complex, where 0-faces are located at floating-point 3-dimensional points.*
- typedef [mln::complex\\_image](#)< 1, [mln::discrete\\_plane\\_1complex\\_geometry](#), [mln::value::int\\_u8](#) > [int\\_u8\\_1complex\\_image2d](#)

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

- typedef `mln::complex_image< 2, mln::discrete_plane_2complex_geometry, mln::value::int_u8 > int_u8_2complex_image2d`

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

- typedef `mln::complex_image< 2, mln::space_2complex_geometry, mln::value::int_u8 > int_u8_2complex_image3df`

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

- typedef `neighb< window1d > neighb1d`

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

- typedef `neighb< window2d > neighb2d`

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

- typedef `neighb< window3d > neighb3d`

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

- typedef `p_run< point2d > p_run2d`

*Type alias for a run of 2d points.*

- typedef `p_set_of< p_run2d > p_runs2d`

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

- typedef `point< grid::tick, def::coordf > point1df`

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

- typedef `point< mln::grid::square, mln::def::coordf > point2df`

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

- typedef `point< grid::cube, def::coordf > point3df`

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

- typedef `mln::complex_image< 2, mln::space_2complex_geometry, mln::value::rgb8 > rgb8_2complex_image3df`

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

- typedef `mln::geom::complex_geometry< 2, point3df > space_2complex_geometry`

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

- typedef `mln::complex_image< 2, mln::space_2complex_geometry, unsigned > unsigned_2complex_image3df`

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

- typedef algebra::vec< 2u, double > [vec2d\\_d](#)  
*2D vector with double coordinates.*
- typedef algebra::vec< 2u, float > [vec2d\\_f](#)  
*2D vector with float coordinates.*
- typedef algebra::vec< 3u, double > [vec3d\\_d](#)  
*3D vector with double coordinates.*
- typedef algebra::vec< 3u, float > [vec3d\\_f](#)  
*3D vector with float coordinates.*
- typedef [w\\_window](#)< [dpoint1d](#), float > [w\\_window1d\\_float](#)  
*Type alias for a [w\\_window](#) with arbitrary shape, defined on the 1D grid (with integer coordinates) and whose weights are floating values.*
- typedef [w\\_window](#)< [dpoint1d](#), int > [w\\_window1d\\_int](#)  
*Type alias for a [w\\_window](#) with arbitrary shape, defined on the 1D grid (with integer coordinates) and whose weights are integers.*
- typedef [w\\_window](#)< [dpoint2d](#), float > [w\\_window2d\\_float](#)  
*Type alias for a [w\\_window](#) with arbitrary shape, defined on the 2D square grid (with integer coordinates) and whose weights are floating values.*
- typedef [w\\_window](#)< [dpoint2d](#), int > [w\\_window2d\\_int](#)  
*Type alias for a [w\\_window](#) with arbitrary shape, defined on the 2D square grid (with integer coordinates) and whose weights are integers.*
- typedef [w\\_window](#)< [dpoint3d](#), float > [w\\_window3d\\_float](#)  
*Type alias for a [w\\_window](#) with arbitrary shape, defined on the 3D grid (with integer coordinates) and whose weights are floating values.*
- typedef [w\\_window](#)< [dpoint3d](#), int > [w\\_window3d\\_int](#)  
*Type alias for a [w\\_window](#) with arbitrary shape, defined on the 3D grid (with integer coordinates) and whose weights are integers.*
- typedef [window](#)< [mln::dpoint1d](#) > [window1d](#)  
*Type alias for a window with arbitrary shape, defined on the 1D square grid with integer coordinates.*
- typedef [window](#)< [mln::dpoint2d](#) > [window2d](#)  
*Type alias for a window with arbitrary shape, defined on the 2D square grid with integer coordinates.*
- typedef [window](#)< [mln::dpoint3d](#) > [window3d](#)  
*Type alias for a window with arbitrary shape, defined on the 3D square grid with integer coordinates.*
- typedef [point](#)< [grid::tick](#), [def::coord](#) > [point1d](#)  
*Type alias for a point defined on the 1D ruler with integer coordinates.*
- typedef [point](#)< [mln::grid::square](#), [mln::def::coord](#) > [point2d](#)  
*Type alias for a point defined on the 2D square grid with integer coordinates.*

- typedef [point](#)< [grid::hexa](#), [def::coord](#) > [point2d\\_h](#)  
*Type alias for a point defined on the 2D hexagonal grid with integer coordinates.*
- typedef [point](#)< [grid::cube](#), [def::coord](#) > [point3d](#)  
*Type alias for a point defined on the 3D square grid with integer coordinates.*

## Functions

- template<typename I >  
I::psite [a\\_point\\_of](#) (const [Image](#)< I > &ima)  
*Give a point of an image.*
- template<typename I, typename F >  
[p2p\\_image](#)< I, F > [apply\\_p2p](#) ([Image](#)< I > &ima, const [Function\\_v2v](#)< F > &f)  
*FIXME: Doc!*
- template<typename I, typename F >  
[p2p\\_image](#)< const I, F > [apply\\_p2p](#) (const [Image](#)< I > &ima, const [Function\\_v2v](#)< F > &f)  
*FIXME: Doc!*
- const [neighb3d](#) & [c18](#) ()  
*18-connectivity neighborhood on the 3D grid.*
- const [neighb1d](#) & [c2](#) ()  
*2-connectivity neighborhood on the 1D grid.*
- const [neighb3d](#) & [c26](#) ()  
*26-connectivity neighborhood on the 3D grid.*
- const [neighb3d](#) & [c2\\_3d\\_sli](#) ()  
*depth 2-connectivity neighborhood on the 3D grid.*
- const [neighb2d](#) & [c2\\_col](#) ()  
*Vertical 2-connectivity neighborhood on the 2D grid.*
- const [neighb2d](#) & [c2\\_row](#) ()  
*Horizontal 2-connectivity neighborhood on the 2D grid.*
- const [neighb2d](#) & [c4](#) ()  
*4-connectivity neighborhood on the 2D grid.*
- const [neighb3d](#) & [c4\\_3d](#) ()  
*4-connectivity neighborhood on the 3D grid.*
- const [neighb3d](#) & [c6](#) ()  
*6-connectivity neighborhood on the 3D grid.*
- const [neighb2d](#) & [c8](#) ()



*8-connectivity neighborhood on the 2D grid.*

- const [neighb3d](#) & [c8\\_3d](#) ()  
*8-connectivity neighborhood on the 3D grid.*
- template<typename T2 , typename T1 >  
[fun::x2x::composed](#)< T2, T1 > [compose](#) (T2 f, T1 g)  
*Do a composition of two transformations.*
- template<typename I >  
[mln::trait::concrete](#)< I >::ret [duplicate](#) (const [Image](#)< I > &model)  
*Duplicate the image model with the values of the image data.*
- template<typename I , typename F >  
[extension\\_fun](#)< const I, F > [extend](#) (const [Image](#)< I > &ima, const [Function\\_v2v](#)< F > &fun)  
*Routines for domain extension with a function.*
- template<typename I , typename J >  
[extension\\_ima](#)< const I, const J > [extend](#) (const [Image](#)< I > &ima, const [Image](#)< J > &ext)  
*Routines for domain extension with an image.*
- template<typename I >  
[extension\\_val](#)< const I > [extend](#) (const [Image](#)< I > &ima, const typename I::value &val)  
*Routines for domain extension with a value.*
- bool [implies](#) (bool lexpr, bool rexpr)  
*Implication.*
- template<typename I , typename J >  
void [initialize](#) ([Image](#)< I > &target, const [Image](#)< J > &model)
- template<typename 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\_higher\_dim\_connected\_n\_face\_neighborhood, complex\_higher\_dim\_connected\_n\_face\_window)  
*Neighborhood centered on an n-face of complex returning the n-faces sharing an (n+1)-face with the center n-face.*
- [mln\\_gen\\_complex\\_neighborhood](#) (complex\_m\_face\_neighborhood, complex\_m\_face\_window)

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

- [mln\\_gen\\_complex\\_neighborhood](#) (complex\_lower\_neighborhood, complex\_lower\_window)  
*Neighborhood* centered on an  $n$ -face of complex returning its adjacent  $(n-1)$ -faces.
- [mln\\_gen\\_complex\\_neighborhood](#) (complex\_higher\_neighborhood, complex\_higher\_window)  
*Neighborhood* centered on an  $n$ -face of complex returning its adjacent  $(n+1)$ -faces.
- [mln\\_gen\\_complex\\_neighborhood](#) (complex\_lower\_higher\_neighborhood, complex\_lower\_higher\_window)  
*Neighborhood* centered on an  $n$ -face of complex returning its adjacent  $(n-1)$ -faces and  $(n+1)$ -faces.
- [mln\\_gen\\_complex\\_neighborhood](#) (complex\_lower\_dim\_connected\_n\_face\_neighborhood, complex\_lower\_dim\_connected\_n\_face\_window)  
*Neighborhood* centered on an  $n$ -face of complex returning the  $n$ -faces sharing an  $(n-1)$ -face with the center  $n$ -face.
- [mln\\_gen\\_complex\\_window](#) (complex\_lower\_window, topo::adj\_lower\_face\_fwd\_iter, topo::adj\_lower\_face\_bkd\_iter)  
*Window* centered on an  $n$ -face of complex returning its adjacent  $(n-1)$ -faces.
- [mln\\_gen\\_complex\\_window](#) (complex\_higher\_window, topo::adj\_higher\_face\_fwd\_iter, topo::adj\_higher\_face\_bkd\_iter)  
*Window* centered on an  $n$ -face of complex returning its adjacent  $(n+1)$ -faces.
- [mln\\_gen\\_complex\\_window](#) (complex\_lower\_higher\_window, topo::adj\_lower\_higher\_face\_fwd\_iter, topo::adj\_lower\_higher\_face\_bkd\_iter)  
*Window* centered on an  $n$ -face of complex returning its adjacent  $(n-1)$ -faces and  $(n+1)$ -faces.
- [mln\\_gen\\_complex\\_window](#) (complex\_lower\_dim\_connected\_n\_face\_window, topo::adj\_lower\_dim\_connected\_n\_face\_fwd\_iter, topo::adj\_lower\_dim\_connected\_n\_face\_bkd\_iter)  
*Window* centered on an  $n$ -face of complex returning the  $n$ -faces sharing an  $(n-1)$ -face with the center  $n$ -face.
- [mln\\_gen\\_complex\\_window](#) (complex\_higher\_dim\_connected\_n\_face\_window, topo::adj\_higher\_dim\_connected\_n\_face\_fwd\_iter, topo::adj\_higher\_dim\_connected\_n\_face\_bkd\_iter)  
*Window* centered on an  $n$ -face of complex returning the  $n$ -faces sharing an  $(n+1)$ -face with the center  $n$ -face.
- [mln\\_gen\\_complex\\_window](#) (complex\_m\_face\_window, topo::adj\_m\_face\_fwd\_iter, topo::adj\_m\_face\_bkd\_iter)  
*Window* centered on an  $n$ -face of complex returning the  $m$ -faces transitively adjacent to this center  $n$ -face.
- [mln\\_gen\\_complex\\_window\\_p](#) (complex\_lower\_window\_p, topo::adj\_lower\_face\_fwd\_iter, topo::adj\_lower\_face\_bkd\_iter)  
*Window* centered on an  $n$ -face of complex returning its adjacent  $(n-1)$ -faces as well as the center  $n$ -face.
- [mln\\_gen\\_complex\\_window\\_p](#) (complex\_higher\_window\_p, topo::adj\_higher\_face\_fwd\_iter, topo::adj\_higher\_face\_bkd\_iter)  
*Window* centered on an  $n$ -face of complex returning its adjacent  $(n+1)$ -faces as well as the center  $n$ -face.

- [mln\\_gen\\_complex\\_window\\_p](#) ([complex\\_lower\\_higher\\_window\\_p](#), [topo::adj\\_lower\\_higher\\_face\\_fwd\\_iter](#), [topo::adj\\_lower\\_higher\\_face\\_bkd\\_iter](#))  
*Window centered on an n-face of complex returning its adjacent (n-1)-faces and (n+1)-faces as well as the center n-face.*
- [mln\\_gen\\_complex\\_window\\_p](#) ([complex\\_lower\\_dim\\_connected\\_n\\_face\\_window\\_p](#), [topo::adj\\_lower\\_dim\\_connected\\_n\\_face\\_fwd\\_iter](#), [topo::adj\\_lower\\_dim\\_connected\\_n\\_face\\_bkd\\_iter](#))  
*Window centered on an n-face of complex returning the n-faces sharing an (n-1)-face with the center n-face, as well as this center n-face.*
- [mln\\_gen\\_complex\\_window\\_p](#) ([complex\\_higher\\_dim\\_connected\\_n\\_face\\_window\\_p](#), [topo::adj\\_higher\\_dim\\_connected\\_n\\_face\\_fwd\\_iter](#), [topo::adj\\_higher\\_dim\\_connected\\_n\\_face\\_bkd\\_iter](#))  
*Window centered on an n-face of complex returning the n-faces sharing an (n+1)-face with the center n-face, as well as this center n-face.*
- [mln\\_gen\\_complex\\_window\\_p](#) ([complex\\_m\\_face\\_window\\_p](#), [topo::adj\\_m\\_face\\_fwd\\_iter](#), [topo::adj\\_m\\_face\\_bkd\\_iter](#))  
*Window centered on an n-face of complex returning the m-faces transitively adjacent to this center n-face, as well as this center n-face.*
- `template<typename W1 , typename W2 >`  
[mln\\_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`)  
*Subtract 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 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 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_fwd_piter< I, G, N > &p`)  
*Print an `mln::complex_neighborhood_fwd_piter`.*
- `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 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_vertices< G, F > &lhs`, const `p_vertices< G, F > &rhs`)  
*Inclusion of a `mln::p_vertices` 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 point-wise less than or equal to the pixel values of rhs.*
- `template<unsigned D, typename G >`  
`bool operator== (const p_complex< D, G > &lhs, const p_complex< D, G > &rhs)`  
*Comparison between two mln::p\_complex'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<typename L , typename R >`  
`bool operator== (const Image< L > &lhs, const Image< R > &rhs)`  
*Point-wise test if the pixel values of lhs are equal to the pixel values of rhs.*
- `template<unsigned N, unsigned D, typename P >`  
`bool operator== (const p_faces< N, D, P > &lhs, const p_faces< N, D, P > &rhs)`  
*Comparison between two mln::p\_faces's.*
- `template<typename G , typename F >`  
`bool operator== (const p_vertices< G, F > &lhs, const p_vertices< G, F > &rhs)`  
*Comparison between two mln::p\_vertices's.*
- `template<typename I , typename F >`  
`image_if< const I, F > operator| (const Image< I > &ima, const Function_v2b< F > &f)`  
*ima | f creates an image\_if with the image ima and the function f.*
- `template<typename V , typename G , typename P >`  
`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< 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 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 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 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 I >`  
`const internal::primary_type< I >::ret & primary (const Image< I > &input)`  
*FIXME: Doc!*
- `template<typename S, typename F >`  
`p\_transformed< S, F > ptransform (const Site\_Set< S > &s, const Function\_v2v< F > &f)`  
*Transform a site set  $s$  through the function  $f$ .*
- `const window2d & win\_c4p ()`  
*4-connectivity window on the 2D grid, including the center.*
- `const window3d & win\_c4p\_3d ()`  
*4-connectivity window on the 3D grid, including the center.*
- `const window2d & win\_c8p ()`  
*8-connectivity window on the 2D grid, including the center.*
- `const window3d & win\_c8p\_3d ()`  
*8-connectivity window on the 3D grid, including the center.*
  
- `template<unsigned N, unsigned D, typename P >`  
`bool operator== (const faces\_psite< N, D, P > &lhs, const faces\_psite< N, D, P > &rhs)`  
*Comparison of two instances of `mln::faces_psite`.*
- `template<unsigned N, unsigned D, typename P >`  
`bool operator!= (const faces\_psite< N, D, P > &lhs, const faces\_psite< N, D, P > &rhs)`  
*Is lhs equal to rhs?*
- `template<unsigned N, unsigned D, typename P >`  
`bool operator< (const faces\_psite< N, D, P > &lhs, const faces\_psite< N, D, P > &rhs)`  
*Is lhs "less" than rhs?*
  
- `template<typename T >`  
`mln\_exact (T)*exact(T *ptr)`  
*Exact cast routine for mln objects.*
  
- `template<unsigned D, typename G >`  
`bool operator== (const complex\_psite< D, G > &lhs, const complex\_psite< D, G > &rhs)`  
*Comparison of two instances of `mln::complex_psite`.*
- `template<unsigned D, typename G >`  
`bool operator!= (const complex\_psite< D, G > &lhs, const complex\_psite< D, G > &rhs)`  
*Is lhs not equal to rhs?*
- `template<unsigned D, typename G >`  
`bool operator< (const complex\_psite< D, G > &lhs, const complex\_psite< D, G > &rhs)`  
*Is lhs "less" than rhs?*

## Variables

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

### 9.1.1 Detailed Description

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

Generic class for hierarchical queues.

The generic dual input tree algorithm for high quantized image.

The dual input tree algorithm specialized for low quantized image.

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

Read AVS header from a file.

Define a function which aborts a process in io module.

Forward declaration.

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

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

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

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

#### See also

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

The functor should provide the following methods:

- `template <typename g>=""> void init(const Graph<G>& g)` Will be called at the beginning.
- `bool to_be_treated(unsigned id)` Return whether this vertex has already been marked or if it may be a component representative.
- `void new_component_from_vertex(unsigned id)` will be called for the first vertex encountered for each component.

- `void process_vertex(unsigned id)` Will be called for each vertex queued.
- `bool to_be_queued(unsigned id)` Return whether this vertex has already been marked or if it can be added to the current component.
- `void added_to_queue(unsigned id)` Will be called for every vertex encountered in each component, except the first one.
- `void next_component()` Will be called after all vertices from a component have been treated.
- `void final()` Will be called at the end;

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

FIXME: Re-write this description.

The contents of mln mimics the contents of the 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.

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

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

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

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

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

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

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

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

FIXME: consider hqueues as a site set ?

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

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

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

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

## 9.1.2 Typedef Documentation

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

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

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

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

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

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

**See also**

[mln::win::rectangle1d](#).

**9.1.2.4 typedef box<mln::point2d> mln::box2d**

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

**See also**

[mln::win::rectangle2d](#).

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

FIXME.

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

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

**See also**

[mln::win::rectangle3d](#).

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Type alias for a run of 2d points.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

2D vector with double coordinates.

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

2D vector with float coordinates.

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

3D vector with double coordinates.

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

3D vector with float coordinates.

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

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

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

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

**9.1.2.35 typedef w\_window<dpoint2d, float> mln::w\_window2d\_float**

Type alias for a [w\\_window](#) with arbitrary shape, defined on the 2D square grid (with integer coordinates) and whose weights are floating values.

**9.1.2.36 typedef w\_window<dpoint2d, int> mln::w\_window2d\_int**

Type alias for a [w\\_window](#) with arbitrary shape, defined on the 2D square grid (with integer coordinates) and whose weights are integers.

**9.1.2.37 typedef w\_window<dpoint3d, float> mln::w\_window3d\_float**

Type alias for a [w\\_window](#) with arbitrary shape, defined on the 3D grid (with integer coordinates) and whose weights are floating values.

**9.1.2.38 typedef w\_window<dpoint3d, int> mln::w\_window3d\_int**

Type alias for a [w\\_window](#) with arbitrary shape, defined on the 3D grid (with integer coordinates) and whose weights are integers.

**9.1.3 Function Documentation****9.1.3.1 template<typename I> I::psite mln::a\_point\_of( const Image< I > & ima ) [inline]**

Give a point of an image.

**9.1.3.2 template<typename I, typename F> p2p\_image< I, F > mln::apply\_p2p ( Image< I > & ima, const Function\_v2v< F > & f ) [inline]**

FIXME: Doc!

Referenced by mln::debug::mosaic(), and mln::debug::slices\_2d().

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

FIXME: Doc!

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

Do a composition of two transformations.

#### Parameters

[in] *f* The second transformation.

[in] *g* The first transformation.

#### Returns

The composed transformation fog.

References compose().

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

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

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

#### Parameters

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

#### Returns

The duplicate.

#### Precondition

*model.is\_valid*

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

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

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

Routines for domain extension with a function.

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

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

Routines for domain extension with an image.

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

Routines for domain extension with a value.

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

Implication.

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

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

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

#### Parameters

- [in, out] *target* The image to be initialized.
- [in] *model* The image to provide data for the initialization.

#### Precondition

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

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

**9.1.3.11** `template<typename 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 neighborhood, there is exactly one connected component of the object, and only one connected component of the background Examples : ( | == object, - = background)

- - | | P | Here p is simple in the c4 and c8 case. | | |
- | - | P | Here p is never simple. | | |

**9.1.3.12** `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()`.

**9.1.3.13** `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()`.

**9.1.3.14** `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()`, `mln::Accumulator< E >::take_as_init()`, `mln::Accumulator< E >::take_n_times()`, `mln::convert::to()`, and `mln::geom::translate()`.

**9.1.3.15** `mln::mln_gen_complex_neighborhood ( complex_lower_dim_connected_  
n_face_neighborhood , complex_lower_dim_connected_n_face_window  
)`

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

**9.1.3.16** `mln::mln_gen_complex_neighborhood ( complex_higher_dim_connected_  
n_face_neighborhood , complex_higher_dim_connected_n_face_window  
)`

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

**9.1.3.17** `mln::mln_gen_complex_neighborhood ( complex_higher_neighborhood ,  
complex_higher_window )`

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

**9.1.3.18** `mln::mln_gen_complex_neighborhood ( complex_lower_higher_neighborhood ,  
complex_lower_higher_window )`

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



**9.1.3.19** `mln::mln_gen_complex_neighborhood ( complex_m_face_neighborhood , complex_m_face_window )`

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

### Warning

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

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

General definition of the "greater than" operator.

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

**Warning**

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

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

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

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

**Warning**

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

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

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

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

template <typename O1
```

General definition of the "not equal to" operator.

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

**Warning**

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

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

Is *lhs* not equal to *rhs*?

#### Precondition

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

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

**9.1.3.39** `template<unsigned N, unsigned D, typename P > bool mln::operator!=( const faces_site< N, D, P > & lhs, const faces_site< N, D, P > & rhs )`

Is *lhs* equal to *rhs*?

#### Precondition

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

References `mln::faces_site< N, D, P >::face()`, and `mln::faces_site< N, D, P >::site_set()`.

**9.1.3.40** `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*.

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

Pre-incrementation for any scalar type.

References `mln::literal::one`.

**9.1.3.42** `template<typename N1, typename N2 > N2 neighb< typename N1::window::regular > mln::operator-( const Neighborhood< N1 > & nbh1, const Neighborhood< N2 > & nbh2 )`

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

It just calls [mln::win::diff](#).

References `mln::win::diff()`.

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

Subtract a delta-point *dp* to a grid point *p*.

#### Parameters

[in] *p* A grid point.

[in] *dp* A delta-point.

The type of *dp* has to compatible with the type of *p*.

#### Returns

A point (temporary object).

#### See also

[mln::Gdpoint](#)

[mln::Gdpoint](#)

#### 9.1.3.44 `template<typename S> S & mln::operator-- ( value::Scalar< S > & rhs ) [inline]`

Pre-decrementation for any scalar type.

References `mln::literal::one`.

#### 9.1.3.45 `template<unsigned D, typename G> bool mln::operator< ( const complex_psite< D, G > & lhs, const complex_psite< D, G > & rhs )`

Is *lhs* “less” than *rhs*?

This comparison is required by algorithms sorting psites.

#### Precondition

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

#### 9.1.3.46 `template<unsigned N, unsigned D, typename P> bool mln::operator< ( const faces_psite< N, D, P > & lhs, const faces_psite< N, D, P > & rhs )`

Is *lhs* “less” than *rhs*?

This comparison is required by algorithms sorting psites.

#### Precondition

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

#### 9.1.3.47 `template<typename L, typename R> bool mln::operator< ( const Image< L > & lhs, const Image< R > & rhs ) [inline]`

Point-wise test if the pixel values of *lhs* are point-wise less than the pixel values of *rhs*.

#### Parameters

[in] *lhs* A first image.

[in] *rhs* A second image.

#### Precondition

`lhs.domain == rhs.domain`

References `mln::test::predicate()`.

**9.1.3.48** `template<typename I , typename G , typename W > std::ostream & mln::operator<< ( std::ostream & ostr, const complex_window_fwd_piter< I, G, W > & p ) [inline]`

Print an [mln::complex\\_window\\_fwd\\_piter](#).

**9.1.3.49** `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](#).

**9.1.3.50** `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](#).

**9.1.3.51** `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](#).

**9.1.3.52** `template<typename G , typename F > bool mln::operator<= ( const p_edges< G, F > & lhs, const p_edges< G, F > & rhs )`

Inclusion of a [mln::p\\_edges](#) in another one.

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

Inclusion of a [mln::p\\_complex](#) in another one.

This inclusion relation is very strict for the moment, since our infrastructure for complexes is simple: a [mln::p\\_complex](#) is included in another one if their are equal.

**9.1.3.54** `template<unsigned N, unsigned D, typename P > bool mln::operator<= ( const p_faces< N, D, P > & lhs, const p_faces< N, D, P > & rhs )`

Inclusion of a [mln::p\\_faces](#) in another one.

This inclusion relation is very strict for the moment, since our infrastructure for complexes is simple: a [mln::p\\_faces](#) is included in another one if their are equal.

**9.1.3.55** `template<typename G , typename F > bool mln::operator<= ( const p_vertices< G, F > & lhs, const p_vertices< G, F > & rhs )`

Inclusion of a [mln::p\\_vertices](#) in another one.

This inclusion relation is very strict for the moment, since our infrastructure for graphs is simple: a [mln::p\\_vertices](#) is included in another one if their are equal.

**9.1.3.56** `template<typename L , typename R > bool mln::operator<= ( const Image< L > & lhs, const Image< R > & rhs ) [inline]`

Point-wise test if the pixel values of `lhs` are point-wise less than or equal to the pixel values of `rhs`.

#### Parameters

[in] *lhs* A first image.

[in] *rhs* A second image.

#### Precondition

`lhs.domain == rhs.domain`

References `mln::test::predicate()`.

**9.1.3.57** `template<typename G , typename F > bool mln::operator== ( const p_edges< G, F > & lhs, const p_edges< G, F > & rhs )`

Comparison between two `mln::p_edges`'s.

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

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

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

Comparison between two `mln::p_complex`'s.

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

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

**9.1.3.59** `template<typename L , typename R > bool mln::operator== ( const Image< L > & lhs, const Image< R > & rhs ) [inline]`

Point-wise test if the pixel values of `lhs` are equal to the pixel values of `rhs`.

#### Parameters

[in] *lhs* A first image.

[in] *rhs* A second image.

#### Precondition

`lhs.domain == rhs.domain`

References `mln::test::predicate()`.

**9.1.3.60** `template<typename G , typename F > bool mln::operator== ( const p_vertices< G, F > & lhs, const p_vertices< G, F > & rhs )`

Comparison between two `mln::p_vertices`'s.

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

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

**9.1.3.61** `template<unsigned N, unsigned D, typename P > bool mln::operator==( const faces_psite< N, D, P > & lhs, const faces_psite< N, D, P > & rhs )`

Comparison of two instances of `mln::faces_psite`.

Is *lhs* equal to *rhs*?

#### Precondition

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

References `mln::faces_psite< N, D, P >::face()`, and `mln::faces_psite< N, D, P >::site_set()`.

**9.1.3.62** `template<unsigned N, unsigned D, typename P > bool mln::operator==( const p_faces< N, D, P > & lhs, const p_faces< N, D, P > & rhs )`

Comparison between two `mln::p_faces`'s.

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

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

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

Comparison of two instances of `mln::complex_psite`.

Is *lhs* equal to *rhs*?

#### Precondition

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

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

**9.1.3.64** `template<typename I, typename F > image_if<const I,F> mln::operator|( const Image< I > & ima, const Function_v2b< F > & f )`

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

**9.1.3.65** `template<typename S, typename F > p_if<S, F> mln::operator|( const Site_Set< S > & s, const Function_v2b< F > & f )`

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

#### Parameters

[in] *s* A site set.

[in] *f* A function from point to Boolean.



**Returns**

A subset of points.

**9.1.3.66** `template<typename I , typename F > image_if<I,F> mln::operator| ( Image< I > & ima, const Function_v2b< F > & f )`

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

**9.1.3.67** `template<typename V , typename G , typename P > edge_image<P,V,G> mln::operator| ( const fun::i2v::array< V > & edge_values, const p_edges< G, fun::i2v::array< P > > & pe )`

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

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

**9.1.3.68** `template<typename V , typename G , typename P > vertex_image<P,V,G> mln::operator| ( const fun::i2v::array< V > & vertex_values, const p_vertices< G, fun::i2v::array< P > > & pv )`

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

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

**9.1.3.69** `template<typename F , typename S > pw::image<F,S> mln::operator| ( const Function_v2v< F > & f, const Site_Set< S > & ps )`

Construct an image from a function and a site set.

`image = function | site_set.`

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

FIXME: Doc!

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

**9.1.3.71** `template<typename S , typename F > p_transformed< S, F > mln::ptransform ( const Site_Set< S > & s, const Function_v2v< F > & f ) [inline]`

Transform a site set `s` through the function `f`.

**Parameters**

[in] `s` A site set.

[in] `f` A function from site to site.

**Returns**

The transformed site set.

## 9.1.4 Variable Documentation

### 9.1.4.1 `const dpoint1d mln::before = dpoint1d( -1 )`

Definition of a shortcut for delta point in 1d.

### 9.1.4.2 `const dpoint3d mln::sagittal_dec = dpoint3d( 0, 0, -1)`

Definition of a shortcut for delta point in 3d.

### 9.1.4.3 `const dpoint2d mln::up = dpoint2d( -1, 0 )`

Definition of a shortcut for delta point in 2d.

## 9.2 `mln::accu` Namespace Reference

Namespace of accumulators.

### Namespaces

- namespace [image](#)  
*Namespace of accumulator image routines.*
- namespace [impl](#)  
*Implementation namespace of accumulator namespace.*
- namespace [logic](#)  
*Namespace of logical accumulators.*
- namespace [math](#)  
*Namespace of mathematic accumulators.*
- namespace [shape](#)  
*Namespace of shape accumulators.*
- namespace [stat](#)  
*Namespace of statistical accumulators.*

### Classes

- struct [center](#)  
*Mass center accumulator.*
- struct [convolve](#)  
*Generic convolution accumulator class.*

- struct [count\\_adjacent\\_vertices](#)  
*Accumulator class counting the number of vertices adjacent to a set of mln::p\_edges\_psite (i.e., a set of edges).*
- struct [count\\_value](#)  
*Define an accumulator that counts the occurrence of a given value.*
- struct [histo](#)  
*Generic histogram class over a value set with type V.*
- struct [label\\_used](#)  
*References all the labels used.*
- struct [maj\\_h](#)  
*Compute the majority value.*
- struct [max\\_site](#)  
*Define an accumulator that computes the first site with the maximum value in an image.*
- struct [nil](#)  
*Define an accumulator that does nothing.*
- struct [p](#)  
*Generic p of accumulators.*
- struct [pair](#)  
*Generic pair of accumulators.*
- struct [rms](#)  
*Generic root mean square accumulator class.*
- struct [tuple](#)  
*Generic tuple of accumulators.*
- struct [val](#)  
*Generic val of accumulators.*

## Functions

- template<typename A , typename I >  
A::result [compute](#) (const [Accumulator](#)< A > &a, const [Image](#)< I > &input)  
*Make an accumulator compute the pixels of the image input.*
- template<typename Meta\_Accu , unsigned Dir, typename I , typename O >  
void [line](#) (const [Image](#)< I > &input, const typename I::site &p\_start, unsigned len, unsigned half\_length, [Image](#)< O > &output)
- template<typename A , typename I >  
[mln\\_meta\\_accu\\_result](#) (A, [util::pix](#)< I >) [compute](#)(const [Meta\\_Accumulator](#)< A > &a  
*Make an accumulator compute the pixels of the image input.*

- `template<typename A , typename I >`  
`void take (const Image< I > &input, Accumulator< A > &a)`  
*Make an accumulator take the pixels of the image input.*

## 9.2.1 Detailed Description

Namespace of accumulators.

## 9.2.2 Function Documentation

### 9.2.2.1 `template<typename A , typename I > A::result mln::accu::compute ( const Accumulator< A > & a, const Image< I > & input ) [inline]`

Make an accumulator compute the pixels of the image input.

#### Parameters

- [in] *input* The input image.
- [in] *a* An accumulator.

This routine runs:

```
a.take(make::pix(input, p)); on all pixels on the images.
```

#### Warning

This routine does not perform `a.init()`.

### 9.2.2.2 `template<typename Meta_Accu , unsigned Dir, typename I , typename O > void mln::accu::line ( const Image< I > & input, const typename I::site & p_start, unsigned len, unsigned half_length, Image< O > & output )`

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

#### Parameters

- [in] *input* The input image.
- [in] *p\_start* The starting site of the line.
- [in] *len* The line length.
- [in] *half\_length* The half length of the line.
- [in, out] *output* The resulting image.

This routine runs:

```
tmp = a
tmp.init()
accu::take(input, tmp)
return tmp.to_result()
```

**9.2.2.3** `template<typename A , typename I > mln::accu::mln_meta_accu_result ( A , util::pix< I > ) const [inline]`

Make an accumulator compute the pixels of the image `input`.

#### Parameters

[in] *input* The input image.

[in] *a* A meta accumulator.

This routine runs:

`a.take(make::pix(input, p));` on all pixels on the images.

#### Warning

This routine does not perform `a.init()`.

**9.2.2.4** `template<typename A , typename I > void mln::accu::take ( const Image< I > & input, Accumulator< A > & a ) [inline]`

Make an accumulator take the pixels of the image `input`.

#### Parameters

[in] *input* The input image.

[in, out] *a* The accumulator.

This routine runs:

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

#### Warning

This routine does not perform `a.init()`.

## 9.3 mln::accu::image Namespace Reference

Namespace of accumulator image routines.

### 9.3.1 Detailed Description

Namespace of accumulator image routines.

## 9.4 mln::accu::impl Namespace Reference

Implementation namespace of accumulator namespace.

### 9.4.1 Detailed Description

Implementation namespace of accumulator namespace.

## 9.5 mln::accu::logic Namespace Reference

Namespace of logical accumulators.

### Classes

- struct [land](#)  
*"Logical-and" accumulator.*
- struct [land\\_basic](#)  
*"Logical-and" accumulator.*
- struct [lor](#)  
*"Logical-or" accumulator.*
- struct [lor\\_basic](#)  
*"Logical-or" accumulator class.*

### 9.5.1 Detailed Description

Namespace of logical accumulators.

## 9.6 mln::accu::math Namespace Reference

Namespace of mathematic accumulators.

### Classes

- struct [count](#)  
*Generic counter accumulator.*
- struct [inf](#)  
*Generic inf accumulator class.*
- struct [sum](#)  
*Generic sum accumulator class.*
- struct [sup](#)  
*Generic sup accumulator class.*

### 9.6.1 Detailed Description

Namespace of mathematic accumulators.

## 9.7 mln::accu::meta::logic Namespace Reference

Namespace of logical meta-accumulators.

### Classes

- struct [land](#)  
*Meta accumulator for land.*
- struct [land\\_basic](#)  
*Meta accumulator for [land\\_basic](#).*
- struct [lor](#)  
*Meta accumulator for lor.*
- struct [lor\\_basic](#)  
*Meta accumulator for [lor\\_basic](#).*

### 9.7.1 Detailed Description

Namespace of logical meta-accumulators.

## 9.8 mln::accu::meta::math Namespace Reference

Namespace of mathematic meta-accumulators.

### Classes

- struct [count](#)  
*Meta accumulator for count.*
- struct [inf](#)  
*Meta accumulator for inf.*
- struct [sum](#)  
*Meta accumulator for sum.*
- struct [sup](#)  
*Meta accumulator for sup.*

### 9.8.1 Detailed Description

Namespace of mathematic meta-accumulators.

## 9.9 mln::accu::meta::shape Namespace Reference

Namespace of shape meta-accumulators.

### Classes

- struct [bbox](#)  
*Meta accumulator for bbox.*
- struct [height](#)  
*Meta accumulator for height.*
- struct [volume](#)  
*Meta accumulator for volume.*

### 9.9.1 Detailed Description

Namespace of shape meta-accumulators.

## 9.10 mln::accu::meta::stat Namespace Reference

Namespace of statistical meta-accumulators.

### Classes

- struct [max](#)  
*Meta accumulator for max.*
- struct [max\\_h](#)  
*Meta accumulator for max.*
- struct [mean](#)  
*Meta accumulator for mean.*
- struct [median\\_alt](#)  
*Meta accumulator for [median\\_alt](#).*
- struct [median\\_h](#)  
*Meta accumulator for [median\\_h](#).*
- struct [min](#)



*Meta accumulator for min.*

- struct [min\\_h](#)  
*Meta accumulator for min.*
- struct [rank](#)  
*Meta accumulator for rank.*
- struct [rank\\_high\\_quant](#)  
*Meta accumulator for [rank\\_high\\_quant](#).*

### 9.10.1 Detailed Description

Namespace of statistical meta-accumulators.

## 9.11 mln::accu::shape Namespace Reference

Namespace of shape accumulators.

### Classes

- struct [bbox](#)  
*Generic bounding box accumulator class.*
- struct [height](#)  
*Height accumulator.*
- struct [volume](#)  
*Volume accumulator class.*

### 9.11.1 Detailed Description

Namespace of shape accumulators.

## 9.12 mln::accu::stat Namespace Reference

Namespace of statistical accumulators.

### Classes

- struct [deviation](#)  
*Generic standard deviation accumulator class.*
- struct [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.*

### 9.12.1 Detailed Description

Namespace of statistical accumulators.

## 9.13 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  $Ax = B$  after [mln::algebra::ldlt\\_decomp](#).*
- `template<unsigned n, typename T, typename U >`  
`mln::trait::value_< typename mln::trait::op::times< T, U >::ret >::sum operator\* (const vec< n, T > &lhs, const vec< n, U > &rhs)`  
*Scalar product (dot product).*
- `template<typename T, typename U >`  
`vec< 3, typename mln::trait::op::times< T, U >::ret > vprod (const vec< 3, T > &lhs, const vec< 3, U > &rhs)`  
*Vectorial product (cross product).*

### 9.13.1 Detailed Description

Namespace of algebraic structure.

### 9.13.2 Function Documentation

#### 9.13.2.1 `template<unsigned N, typename T > bool mln::algebra::ldlt_decomp ( mat< N, N, T > & A, vec< N, T > & rdiag ) [inline]`

Perform  $LDL^T$  decomposition of a symmetric positive definite matrix.

Like Cholesky, but no square roots. Overwrites lower triangle of matrix.

From Trimesh's `ldltdc` routine.

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

**9.13.2.2** `template<unsigned N, typename T > void mln::algebra::ldlt_solve ( const mat< N, N, T > & A, const vec< N, T > & rdiag, const vec< N, T > & B, vec< N, T > & x )`  
**[inline]**

Solve  $Ax = B$  after [mln::algebra::ldlt\\_decomp](#).

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

**9.13.2.3** `template<unsigned n, typename T , typename U > mln::trait::value_< typename mln::trait::op::times< T, U >::ret >::sum mln::algebra::operator* ( const vec< n, T > & lhs, const vec< n, U > & rhs )` **[inline]**

Scalar product (dot product).

References `mln::literal::zero`.

**9.13.2.4** `template<typename T , typename U > vec< 3, typename mln::trait::op::times< T, U >::ret > mln::algebra::vprod ( const vec< 3, T > & lhs, const vec< 3, U > & rhs )`  
**[inline]**

Vectorial product (cross product).

References `vprod()`.

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

## 9.14 mln::arith Namespace Reference

Namespace of arithmetic.

### Namespaces

- namespace [impl](#)  
*Implementation namespace of arith namespace.*

### Functions

- `template<typename I > mln::trait::concrete< I >::ret diff_abs (const Image< I > &lhs, const Image< I > &rhs)`  
*Point-wise absolute difference of images lhs and rhs.*
- `template<typename L , typename R , typename O > void div (const Image< L > &lhs, const Image< R > &rhs, Image< O > &output)`  
*Point-wise division of images lhs and rhs.*
- `template<typename I , typename V , typename O > void div_cst (const Image< I > &input, const V &val, Image< O > &output)`  
*Point-wise division of the value val to image input.*

- `template<typename L , typename R >`  
`void div\_inplace (Image< L > &lhs, const Image< R > &rhs)`  
*Point-wise division of image rhs in image lhs.*
- `template<typename L , typename R >`  
`mln::trait::concrete< L >::ret min (const Image< L > &lhs, const Image< R > &rhs)`  
*Point-wise min of images lhs and rhs.*
- `template<typename L , typename R >`  
`void min\_inplace (Image< L > &lhs, const Image< R > &rhs)`  
*Point-wise min of image lhs in image rhs.*
- `template<typename L , typename R >`  
`mln::trait::op::minus< L, R >::ret minus (const Image< L > &lhs, const Image< R > &rhs)`  
*Point-wise addition of images lhs and rhs.*
- `template<typename L , typename R , typename F >`  
`mln::trait::ch_value< L, typename F::result >::ret minus (const Image< L > &lhs, const Image< R > &rhs, const Function\_v2v< F > &f)`  
*Point-wise addition of images lhs and rhs.*
- `template<typename V , typename L , typename R >`  
`mln::trait::ch_value< L, V >::ret minus (const Image< L > &lhs, const Image< R > &rhs)`  
*Point-wise addition of images lhs and rhs.*
- `template<typename I , typename V >`  
`mln::trait::op::minus< I, V >::ret minus\_cst (const Image< I > &input, const V &val)`  
*Point-wise addition of the value val to image input.*
- `template<typename I , typename V , typename F >`  
`mln::trait::ch_value< I, typename F::result >::ret minus\_cst (const Image< I > &input, const V &val, const Function\_v2v< F > &f)`  
*Point-wise addition of the value val to image input.*
- `template<typename I , typename V >`  
`I & minus\_cst\_inplace (Image< I > &input, const V &val)`  
*Point-wise addition of the value val to image input.*
- `template<typename L , typename R >`  
`void minus\_inplace (Image< L > &lhs, const Image< R > &rhs)`  
*Point-wise addition of image rhs in image lhs.*
- `template<typename L , typename R , typename F >`  
`mln::trait::ch_value< L, typename F::result >::ret plus (const Image< L > &lhs, const Image< R > &rhs, const Function\_v2v< F > &f)`  
*Point-wise addition of images lhs and rhs.*
- `template<typename V , typename L , typename R >`  
`mln::trait::ch_value< L, V >::ret plus (const Image< L > &lhs, const Image< R > &rhs)`  
*Point-wise addition of images lhs and rhs.*

- `template<typename L, typename R >`  
`mln::trait::op::plus< L, R >::ret plus (const Image< L > &lhs, const Image< R > &rhs)`  
*Point-wise addition of images lhs and rhs.*
- `template<typename I, typename V >`  
`mln::trait::op::plus< I, V >::ret plus_cst (const Image< I > &input, const V &val)`  
*Point-wise addition of the value val to image input.*
- `template<typename I, typename V, typename F >`  
`mln::trait::ch_value< I, typename F::result >::ret plus_cst (const Image< I > &input, const V &val, const Function_v2v< F > &f)`  
*Point-wise addition of the value val to image input.*
- `template<typename W, typename I, typename V >`  
`mln::trait::ch_value< I, W >::ret plus_cst (const Image< I > &input, const V &val)`  
*Point-wise addition of the value val to image input.*
- `template<typename I, typename V >`  
`I & plus_cst_inplace (Image< I > &input, const V &val)`  
*Point-wise addition of the value val to image input.*
- `template<typename L, typename R >`  
`void plus_inplace (Image< L > &lhs, const Image< R > &rhs)`  
*Point-wise addition of image rhs in image lhs.*
- `template<typename I >`  
`mln::trait::concrete< I >::ret revert (const Image< I > &input)`  
*Point-wise reversion of image input.*
- `template<typename I >`  
`void revert_inplace (Image< I > &input)`  
*Point-wise in-place reversion of image input.*
- `template<typename L, typename R, typename O >`  
`void times (const Image< L > &lhs, const Image< R > &rhs, Image< O > &output)`  
*Point-wise addition of images lhs and rhs.*
- `template<typename I, typename V, typename O >`  
`void times_cst (const Image< I > &input, const V &val, Image< O > &output)`  
*Point-wise addition of the value val to image input.*
- `template<typename L, typename R >`  
`void times_inplace (Image< L > &lhs, const Image< R > &rhs)`  
*Point-wise addition of image rhs in image lhs.*

### 9.14.1 Detailed Description

Namespace of arithmetic.

## 9.14.2 Function Documentation

### 9.14.2.1 `template<typename I> mln::trait::concrete< I >::ret mln::arith::diff_abs ( const Image< I > & lhs, const Image< I > & rhs ) [inline]`

Point-wise absolute difference of images `lhs` and `rhs`.

#### Parameters

- [in] *lhs* First operand image.
- [in] *rhs* Second operand image.

#### Returns

The result image.

#### Precondition

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

References `mln::data::transform()`.

### 9.14.2.2 `template<typename L , typename R , typename O > void mln::arith::div ( const Image< L > & lhs, const Image< R > & rhs, Image< O > & output ) [inline]`

Point-wise division of images `lhs` and `rhs`.

#### Parameters

- [in] *lhs* First operand image.
- [in] *rhs* Second operand image.
- [out] *output* The result image.

#### Precondition

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

### 9.14.2.3 `template<typename I , typename V , typename O > void mln::arith::div_cst ( const Image< I > & input, const V & val, Image< O > & output ) [inline]`

Point-wise division of the value `val` to image `input`.

#### Parameters

- [in] *input* The image.
- [in] *val* The value.
- [out] *output* The result image.

#### Precondition

```
output.domain == input.domain
```

References `div_cst()`.

Referenced by `div_cst()`.

#### 9.14.2.4 `template<typename L , typename R > void mln::arith::div_inplace ( Image< L > & lhs, const Image< R > & rhs ) [inline]`

Point-wise division of image `rhs` in image `lhs`.

##### Parameters

- [in] *lhs* First operand image (subject to division).
- [in, out] *rhs* Second operand image (to div `lhs`).

This addition performs:

for all `p` of `rhs.domain`

`lhs(p) /= rhs(p)`

##### Precondition

```
rhs.domain <= lhs.domain
```

References `div_inplace()`.

Referenced by `div_inplace()`.

#### 9.14.2.5 `template<typename L , typename R > mln::trait::concrete< L >::ret mln::arith::min ( const Image< L > & lhs, const Image< R > & rhs ) [inline]`

Point-wise min of images `lhs` and `rhs`.

##### Parameters

- [in] *lhs* First operand image.
- [in] *rhs* Second operand image.

##### Returns

The result image.

##### Precondition

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

References `mln::initialize()`.

#### 9.14.2.6 `template<typename L , typename R > void mln::arith::min_inplace ( Image< L > & lhs, const Image< R > & rhs ) [inline]`

Point-wise min of image `lhs` in image `rhs`.

##### Parameters

- [in, out] *lhs* First operand image.
- [in] *rhs* Second operand image.

##### Precondition

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



**9.14.2.7** `template<typename L , typename R > mln::trait::op::minus< L, R >::ret  
mln::arith::minus ( const Image< L > & lhs, const Image< R > & rhs ) [inline]`

Point-wise addition of images `lhs` and `rhs`.

#### Parameters

- [in] *lhs* First operand image.
- [in] *rhs* Second operand image.

#### Returns

The result image.

#### Precondition

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

References `mln::initialize()`.

**9.14.2.8** `template<typename L , typename R , typename F > mln::trait::ch_value< L, typename  
F::result >::ret mln::arith::minus ( const Image< L > & lhs, const Image< R > & rhs,  
const Function_v2v< F > & f ) [inline]`

Point-wise addition of images `lhs` and `rhs`.

#### Parameters

- [in] *lhs* First operand image.
- [in] *rhs* Second operand image.
- [in] *f* [Function](#).

#### Returns

The result image.

#### Precondition

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

References `mln::initialize()`.

**9.14.2.9** `template<typename V , typename L , typename R > mln::trait::ch_value< L, V >::ret  
mln::arith::minus ( const Image< L > & lhs, const Image< R > & rhs ) [inline]`

Point-wise addition of images `lhs` and `rhs`.

#### Parameters

- [in] *lhs* First operand image.
- [in] *rhs* Second operand image.

#### Returns

The result image.

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

#### Precondition

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

**9.14.2.10** `template<typename I, typename V > mln::trait::op::minus< I, V >::ret  
mln::arith::minus_cst ( const Image< I > & input, const V & val ) [inline]`

Point-wise addition of the value `val` to image `input`.

#### Parameters

[in] *input* The image.

[in] *val* The value.

#### Returns

The result image.

#### Precondition

```
input.is_valid
```

**9.14.2.11** `template<typename I, typename V, typename F > mln::trait::ch_value< I, typename  
F::result >::ret mln::arith::minus_cst ( const Image< I > & input, const V & val,  
const Function_v2v< F > & f ) [inline]`

Point-wise addition of the value `val` to image `input`.

#### Parameters

[in] *input* The image.

[in] *val* The value.

[in] *f* [Function](#).

#### Returns

The result image.

#### Precondition

```
input.is_valid
```

**9.14.2.12** `template<typename I, typename V > I & mln::arith::minus_cst_inplace ( Image< I >  
& input, const V & val ) [inline]`

Point-wise addition of the value `val` to image `input`.

#### Parameters

[in, out] *input* The image.

[in] *val* The value.

### Precondition

```
input.is_valid
```

References `minus_cst_inplace()`, and `minus_inplace()`.

Referenced by `minus_cst_inplace()`.

#### 9.14.2.13 `template<typename L , typename R > void mln::arith::minus_inplace ( Image< L > & lhs, const Image< R > & rhs ) [inline]`

Point-wise addition of image `rhs` in image `lhs`.

### Parameters

[in, out] *lhs* First operand image (subject to addition).

[in] *rhs* Second operand image (to be added to `lhs`).

This addition performs:

for all `p` of `rhs.domain`

`lhs(p) -= rhs(p)`

### Precondition

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

References `minus_inplace()`.

Referenced by `minus_cst_inplace()`, and `minus_inplace()`.

#### 9.14.2.14 `template<typename L , typename R , typename F > mln::trait::ch_value< L, typename F::result >::ret mln::arith::plus ( const Image< L > & lhs, const Image< R > & rhs, const Function_v2v< F > & f ) [inline]`

Point-wise addition of images `lhs` and `rhs`.

### Parameters

[in] *lhs* First operand image.

[in] *rhs* Second operand image.

[in] *f* [Function](#).

### Returns

The result image.

### Precondition

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

References `mln::initialize()`.

**9.14.2.15** `template<typename V , typename L , typename R > mln::trait::ch_value< L, V >::ret  
mln::arith::plus ( const Image< L > & lhs, const Image< R > & rhs ) [inline]`

Point-wise addition of images `lhs` and `rhs`.

#### Parameters

- [in] *lhs* First operand image.
- [in] *rhs* Second operand image.

#### Returns

The result image.

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

#### Precondition

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

**9.14.2.16** `template<typename L , typename R > mln::trait::op::plus< L, R >::ret  
mln::arith::plus ( const Image< L > & lhs, const Image< R > & rhs ) [inline]`

Point-wise addition of images `lhs` and `rhs`.

#### Parameters

- [in] *lhs* First operand image.
- [in] *rhs* Second operand image.

#### Returns

The result image.

#### Precondition

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

References `mln::initialize()`.

**9.14.2.17** `template<typename I , typename V > mln::trait::op::plus< I, V >::ret  
mln::arith::plus_cst ( const Image< I > & input, const V & val ) [inline]`

Point-wise addition of the value `val` to image `input`.

#### Parameters

- [in] *input* The image.
- [in] *val* The value.

#### Returns

The result image.

#### Precondition

```
input.is_valid
```

Referenced by `plus_cst()`.

**9.14.2.18** `template<typename I, typename V, typename F > mln::trait::ch_value< I, typename F::result >::ret mln::arith::plus_cst ( const Image< I > & input, const V & val, const Function_v2v< F > & f ) [inline]`

Point-wise addition of the value *val* to image *input*.

#### Parameters

- [in] *input* The image.
- [in] *val* The value.
- [in] *f* [Function](#).

#### Returns

The result image.

#### Precondition

`input.is_valid`

**9.14.2.19** `template<typename W, typename I, typename V > mln::trait::ch_value< I, W >::ret mln::arith::plus_cst ( const Image< I > & input, const V & val ) [inline]`

Point-wise addition of the value *val* to image *input*.

#### Parameters

- [in] *input* The image.
- [in] *val* The value.

#### Returns

The result image.

#### Precondition

`input.is_valid`

References `plus_cst()`.

**9.14.2.20** `template<typename I, typename V > I & mln::arith::plus_cst_inplace ( Image< I > & input, const V & val ) [inline]`

Point-wise addition of the value *val* to image *input*.

#### Parameters

- [in, out] *input* The image.
- [in] *val* The value.

#### Precondition

`input.is_valid`

References `plus_cst_inplace()`, and `plus_inplace()`.

Referenced by `plus_cst_inplace()`.

#### 9.14.2.21 `template<typename L , typename R > void mln::arith::plus_inplace ( Image< L > & lhs, const Image< R > & rhs ) [inline]`

Point-wise addition of image `rhs` in image `lhs`.

##### Parameters

[in, out] *lhs* First operand image (subject to addition).

[in] *rhs* Second operand image (to be added to `lhs`).

This addition performs:

for all `p` of `rhs.domain`

`lhs(p) += rhs(p)`

##### Precondition

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

References `plus_inplace()`.

Referenced by `plus_cst_inplace()`, and `plus_inplace()`.

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

Point-wise reversion of image `input`.

##### Parameters

[in] *input* the input image.

##### Returns

The result image.

##### Precondition

```
input.is_valid
```

It performs:

for all `p` of `input.domain`

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

References `mln::initialize()`.

#### 9.14.2.23 `template<typename I > void mln::arith::revert_inplace ( Image< I > & input ) [inline]`

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

##### Parameters

[in, out] *input* The target image.

**Precondition**

```
input.is_valid
```

It performs:

for all  $p$  of `input.domain`

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

**9.14.2.24** `template<typename L , typename R , typename O > void mln::arith::times ( const Image< L > & lhs, const Image< R > & rhs, Image< O > & output ) [inline]`

Point-wise addition of images `lhs` and `rhs`.

**Parameters**

[in] *lhs* First operand image.

[in] *rhs* Second operand image.

[out] *output* The result image.

**Precondition**

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

**9.14.2.25** `template<typename I , typename V , typename O > void mln::arith::times_cst ( const Image< I > & input, const V & val, Image< O > & output ) [inline]`

Point-wise addition of the value `val` to image `input`.

**Parameters**

[in] *input* The image.

[in] *val* The value.

[out] *output* The result image.

**Precondition**

```
output.domain == input.domain
```

References `times_cst()`.

Referenced by `times_cst()`.

**9.14.2.26** `template<typename L , typename R > void mln::arith::times_inplace ( Image< L > & lhs, const Image< R > & rhs ) [inline]`

Point-wise addition of image `rhs` in image `lhs`.

**Parameters**

[in] *lhs* First operand image (subject to addition).

[in, out] *rhs* Second operand image (to be added to `lhs`).

This addition performs:

for all `p` of `rhs.domain`

`lhs(p) *= rhs(p)`

#### Precondition

```
rhs.domain <= lhs.domain
```

References `times_inplace()`.

Referenced by `times_inplace()`.

## 9.15 `mln::arith::impl` Namespace Reference

Implementation namespace of `arith` namespace.

### Namespaces

- namespace [generic](#)  
*Generic implementation namespace of `arith` namespace.*

### 9.15.1 Detailed Description

Implementation namespace of `arith` namespace.

## 9.16 `mln::arith::impl::generic` Namespace Reference

Generic implementation namespace of `arith` namespace.

### 9.16.1 Detailed Description

Generic implementation namespace of `arith` namespace.

## 9.17 `mln::binarization` Namespace Reference

Namespace of "point-wise" expression tools.

### Functions

- `template<typename I, typename F > mln::trait::ch_value< I, bool >::ret binarization (const Image< I > &input, const Function\_v2b< F > &fun)`

*Thresholds the values of `input` so that they can be stored in the output binary image.*



- `template<typename I>`  
`mln::trait::ch_value< I, bool >::ret threshold (const Image< I > &input, const typename I::value threshold)`

*Thresholds the values of `input` so that they can be stored in the output binary image.*

### 9.17.1 Detailed Description

Namespace of "point-wise" expression tools.

### 9.17.2 Function Documentation

- 9.17.2.1** `template<typename I , typename F > mln::trait::ch_value< I, bool >::ret`  
`mln::binarization::binarization ( const Image< I > & input, const Function\_v2b< F >`  
`& fun ) [inline]`

Thresholds the values of `input` so that they can be stored in the output binary image.

#### Parameters

[in] *input* The input image.

[in] *fun* The thresholding function, from value(I) to bool.

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

Referenced by `threshold()`.

- 9.17.2.2** `template<typename I > mln::trait::ch_value< I, bool >::ret mln::binarization::threshold`  
`( const Image< I > & input, const typename I::value threshold ) [inline]`

Thresholds the values of `input` so that they can be stored in the output binary image.

#### Parameters

[in] *input* The input image.

[in] *threshold* The threshold.

If `input(p)` is greater or equal than the threshold, the value in the output image in the same point will be TRUE, else FALSE.

References `binarization()`.

## 9.18 mln::border Namespace Reference

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

### Namespaces

- namespace [impl](#)

*Implementation namespace of border namespace.*

## Functions

- `template<typename I >`  
void `adjust` (const `Image< I >` &ima, unsigned min\_thickness)
- `template<typename I >`  
void `duplicate` (const `Image< I >` &ima)
- `template<typename I, typename J >`  
void `equalize` (const `Image< I >` &ima1, const `Image< J >` &ima2, unsigned min\_thickness)
- `template<typename I >`  
void `fill` (const `Image< I >` &ima, const typename I::value &v)
- `template<typename I >`  
unsigned `find` (const `Image< I >` &ima)
- `template<typename I >`  
unsigned `get` (const `Image< I >` &ima)
- `template<typename I >`  
void `mirror` (const `Image< I >` &ima)
- `template<typename I >`  
void `resize` (const `Image< I >` &ima, unsigned thickness)

*Facade.*

### 9.18.1 Detailed Description

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

### 9.18.2 Function Documentation

#### 9.18.2.1 `template<typename I > void mln::border::adjust ( const Image< I > & ima, unsigned min_thickness ) [inline]`

Adjust the virtual (outer) border of image `ima` so that its size is at least `min_thickness`.

#### Parameters

- [in, out] *ima* The image whose border is to be adjusted.
- [in] *min\_thickness* The expected border minimum thickness.

#### Precondition

`ima` has to be initialized.

#### Warning

If the image border is already larger than `min_thickness`, this routine is a no-op.

References `get()`, and `resize()`.

#### 9.18.2.2 `template<typename I > void mln::border::duplicate ( const Image< I > & ima )`

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

**Parameters**

[in, out] *ima* The image whose border is to be duplicated.

**Precondition**

*ima* has to be initialized.

References get().

**9.18.2.3** `template<typename I, typename J > void mln::border::equalize ( const Image< I > & ima1, const Image< J > & ima2, unsigned min_thickness ) [inline]`

Equalize the virtual (outer) border of images *ima1* and *ima2* so that their size is equal and is at least *min\_thickness*.

**Parameters**

[in, out] *ima1* The first image whose border is to be equalized.

[in, out] *ima2* The second image whose border is to be equalized.

[in] *min\_thickness* The expected border minimum thickness of both images.

**Precondition**

*ima1* has to be initialized.

*ima2* has to be initialized.

**Warning**

If both image borders already have the same thickness and if this thickness is larger than *min\_thickness*, this routine is a no-op.

References get().

**9.18.2.4** `template<typename I > void mln::border::fill ( const Image< I > & ima, const typename I::value & v ) [inline]`

Fill the virtual (outer) border of image *ima* with the single value *v*.

**Parameters**

[in, out] *ima* The image whose border is to be filled.

[in] *v* The value to assign to all border pixels.

**Precondition**

*ima* has to be initialized.

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

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

**Parameters**

[in] *ima* The image.

**Returns**

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

**Precondition**

*ima* has to be initialized.

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

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

**Parameters**

[in] *ima* The image.

**Returns**

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

**Precondition**

*ima* has to be initialized.

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

**9.18.2.7** `template<typename I> void mln::border::mirror ( const Image< I > & ima )`  
`[inline]`

Mirror the virtual (outer) border of image *ima* with the (inner) level contents of this image.

**Parameters**

[in, out] *ima* The image whose border is to be mirrored.

**Precondition**

*ima* has to be initialized.

**9.18.2.8** `template<typename I> void mln::border::resize ( const Image< I > & ima, unsigned`  
`thickness ) [inline]`

Facade.

Resize the virtual (outer) border of image *ima* to exactly *thickness*.

**Parameters**

[in, out] *ima* The image whose border is to be resized.

[in] *thickness* The expected border thickness.

#### Precondition

`ima` has to be initialized.

#### Warning

If the image border already has the expected thickness, this routine is a no-op.

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

Referenced by `adjust()`, and `resize()`.

## 9.19 mln::border::impl Namespace Reference

Implementation namespace of border namespace.

### Namespaces

- namespace [generic](#)  
*Generic implementation namespace of border namespace.*

#### 9.19.1 Detailed Description

Implementation namespace of border namespace.

## 9.20 mln::border::impl::generic Namespace Reference

Generic implementation namespace of border namespace.

#### 9.20.1 Detailed Description

Generic implementation namespace of border namespace.

## 9.21 mln::canvas Namespace Reference

Namespace of canvas.

### Namespaces

- namespace [browsing](#)  
*Namespace of browsing canvas.*
- namespace [impl](#)

*Implementation namespace of canvas namespace.*

- namespace [labeling](#)  
*Namespace of labeling canvas.*
- namespace [morpho](#)  
*Namespace of morphological canvas.*

## Classes

- struct [chamfer](#)  
*Compute chamfer distance.*

## Functions

- `template<typename I, typename N, typename W, typename D, typename F > mln::trait::ch_value< I, D >::ret distance\_front (const Image< I > &input, const Neighborhood< N > &nbh, const Weighted\_Window< W > &w_win, D max, F &functor)`  
*Canvas of discrete distance computation by thick front propagation.*
- `template<typename I, typename N, typename D, typename F > mln::trait::ch_value< I, D >::ret distance\_geodesic (const Image< I > &input, const Neighborhood< N > &nbh, D max, F &functor)`  
*Discrete geodesic distance canvas.*

### 9.21.1 Detailed Description

Namespace of canvas.

#### 9.21.2 Function Documentation

**9.21.2.1** `template<typename I, typename N, typename W, typename D, typename F > mln::trait::ch_value< I, D >::ret mln::canvas::distance_front ( const Image< I > &input, const Neighborhood< N > & nbh, const Weighted\_Window< W > & w_win, D max, F & functor ) [inline]`

Canvas of discrete distance computation by thick front propagation.

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

**9.21.2.2** `template<typename I, typename N, typename D, typename F > mln::trait::ch_value< I, D >::ret mln::canvas::distance_geodesic ( const Image< I > & input, const Neighborhood< N > & nbh, D max, F & functor ) [inline]`

Discrete geodesic distance canvas.

Referenced by `mln::transform::influence_zone_geodesic_saturated()`.

## 9.22 mln::canvas::browsing Namespace Reference

Namespace of browsing canvas.

### Classes

- struct [backdiagonal2d\\_t](#)  
*Browsing in a certain direction.*
- struct [breadth\\_first\\_search\\_t](#)  
*Breadth-first search algorithm for graph, on vertices.*
- struct [depth\\_first\\_search\\_t](#)  
*Breadth-first search algorithm for graph, on vertices.*
- struct [diagonal2d\\_t](#)  
*Browsing in a certain direction.*
- struct [dir\\_struct\\_elt\\_incr\\_update\\_t](#)  
*Browsing in a certain direction with a segment.*
- struct [directional\\_t](#)  
*Browsing in a certain direction.*
- struct [fwd\\_t](#)  
*Canvas for forward browsing.*
- struct [hyper\\_directional\\_t](#)  
*Browsing in a certain direction.*
- struct [snake\\_fwd\\_t](#)  
*Browsing in a snake-way, forward.*
- struct [snake\\_generic\\_t](#)  
*Multidimensional Browsing in a given-way.*
- struct [snake\\_vert\\_t](#)  
*Browsing in a snake-way, forward.*

### 9.22.1 Detailed Description

Namespace of browsing canvas.

## 9.23 mln::canvas::impl Namespace Reference

Implementation namespace of canvas namespace.

### 9.23.1 Detailed Description

Implementation namespace of canvas namespace.

## 9.24 mln::canvas::labeling Namespace Reference

Namespace of labeling canvas.

### Namespaces

- namespace [impl](#)  
*Implementation namespace of labeling canvas namespace.*

### Functions

- `template<typename I, typename N, typename L, typename F > mln::trait::ch_value< I, L >::ret blobs (const Image< I > &input_, const Neighborhood< N > &nbh_, L &nlabels, F &functor)`  
*Canvas for connected component labeling of the binary objects of a binary image using a queue-based algorithm.*

### 9.24.1 Detailed Description

Namespace of labeling canvas.

### 9.24.2 Function Documentation

**9.24.2.1** `template<typename I, typename N, typename L, typename F > mln::trait::ch_value< I, L >::ret mln::canvas::labeling::blobs ( const Image< I > & input_, const Neighborhood< N > & nbh_, L & nlabels, F & functor ) [inline]`

Canvas for connected component labeling of the binary objects of a binary image using a queue-based algorithm.

#### Parameters

- [in] *input* The input image.
- [in] *nbh* The connexity of the objects.
- [out] *nlabels* The Number of labels. Its value is set in the algorithms.
- [in, out] *functor* A functor computing data while labeling.

#### Returns

The label image.

#### Precondition

The input image has to be binary (checked at compile-time).



A fast queue is used so that the algorithm is not recursive and can handle large binary objects (blobs).

## 9.25 mln::canvas::labeling::impl Namespace Reference

Implementation namespace of labeling canvas namespace.

### 9.25.1 Detailed Description

Implementation namespace of labeling canvas namespace.

## 9.26 mln::canvas::morpho Namespace Reference

Namespace of morphological canvas.

### 9.26.1 Detailed Description

Namespace of morphological canvas.

## 9.27 mln::convert Namespace Reference

Namespace of conversion routines.

### Functions

- `template<typename V >`  
`void from_to (const float &from, Value< V > &to)`  
*Conversion of a float from towards a value to.*
- `template<typename V >`  
`void from_to (const int &from, Value< V > &to)`  
*Conversion of a int from towards a value to.*
- `template<typename V >`  
`void from_to (const double &from, Value< V > &to)`  
*Conversion of a double from towards a value to.*
- `template<typename V >`  
`void from_to (const unsigned &from, Value< V > &to)`  
*Conversion of an unsigned from towards a value to.*
- `template<typename S >`  
`mln_image_from_grid (typename S::site::grid, bool) to_image(const Site_Set< S > &pset)`  
*Convert a point set pset into a binary image.*

- `template<typename W >`  
[mln\\_image\\_from\\_grid](#) (typename W::site::grid, bool) to\_image(const [Window](#)< W > &win)  
*Convert a window win into a binary image.*
- `template<typename W >`  
[mln\\_image\\_from\\_grid](#) (typename W::site::grid, mln\_weight(W)) to\_image(const [Weighted\\_Window](#)< W > &w\_win)  
*Convert a weighted window w\_win into an image.*
- `template<typename N >`  
[mln\\_image\\_from\\_grid](#) (typename N::site::grid, bool) to\_image(const [Neighborhood](#)< N > &nbh)  
*Convert a neighborhood nbh into a binary image.*
- `template<typename N >`  
[mln\\_window](#) (N) to\_window(const [Neighborhood](#)< N > &nbh)  
*Convert a neighborhood nbh into a window.*
- `template<typename T, typename O >`  
[T to](#) (const O &from)  
*Conversion of the object from towards an object with type T.*
- `template<typename P >`  
[P::dpoint to\\_dpoint](#) (const [Point\\_Site](#)< P > &p)  
*Convert a point site p into a delta-point.*
- `template<typename I >`  
[pw::value\\_< I > to\\_fun](#) (const [Image](#)< I > &ima)  
*Convert an image into a function.*
- `template<typename T >`  
[imageId< unsigned > to\\_image](#) (const [histo::array](#)< T > &h)  
*Convert an histo h into an imageId<unsigned>.*
- `template<typename I >`  
[p\\_array< typename I::psite > to\\_p\\_array](#) (const [Image](#)< I > &img)  
*Convert an image img into a p\_array.*
- `template<typename S >`  
[p\\_array< typename S::psite > to\\_p\\_array](#) (const [Site\\_Set](#)< S > &pset)  
*Convert a point set pset into a p\_array (point set vector).*
- `template<typename W >`  
[p\\_array< typename W::psite > to\\_p\\_array](#) (const [Window](#)< W > &win, const typename W::psite &p)  
*Convert a window win centered at point p into a p\_array (point set vector).*
- `template<typename N >`  
[p\\_set< typename N::psite > to\\_p\\_set](#) (const [Neighborhood](#)< N > &nbh)  
*Convert a neighborhood nbh into a site set.*

- `template<typename I >`  
`p_set< typename I::psite > to_p_set (const Image< I > &ima)`  
*Convert a binary image `ima` into a site set.*
- `template<typename P, typename C >`  
`p_set< P > to_p_set (const std::set< P, C > &s)`  
*Convert an `std::set` `s` of sites into a site set.*
- `template<typename S >`  
`p_set< typename S::psite > to_p_set (const Site_Set< S > &ps)`  
*Convert any site set `ps` into a 'mlnp\_set' site set.*
- `template<typename W >`  
`p_set< typename W::psite > to_p_set (const Window< W > &win)`  
*Convert a `Window` `win` into a site set.*
- `template<typename I >`  
`QImage to_qimage (const Image< I > &ima)`  
*Convert a Milena image to a `Qimage`.*
- `template<typename N >`  
`window< typename N::dpoint > to_upper_window (const Neighborhood< N > &nbh)`  
*Convert a neighborhood `nbh` into an upper window.*
- `template<typename W >`  
`window< typename W::dpsite > to_upper_window (const Window< W > &win)`  
*Convert a window `nbh` into an upper window.*
- `template<typename D, typename C >`  
`window< D > to_window (const std::set< D, C > &s)`  
*Convert an `std::set` `s` of delta-sites into a window.*
- `template<typename I >`  
`window< typename I::site::dpsite > to_window (const Image< I > &ima)`  
*Convert a binary image `ima` into a window.*
- `template<typename S >`  
`window< typename S::site::dpsite > to_window (const Site_Set< S > &pset)`  
*Convert a site set `pset` into a window.*

## Variables

- `fun::C< R(*)(>A)> to_fun (R(*f)(A))`  
*Convert a C unary function into an `mln::fun::C`.*

### 9.27.1 Detailed Description

Namespace of conversion routines.

## 9.27.2 Function Documentation

**9.27.2.1** `template<typename V > void mln::convert::from_to ( const float & from, Value< V > & to )`

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

**9.27.2.2** `template<typename V > void mln::convert::from_to ( const int & from, Value< V > & to )`

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

**9.27.2.3** `template<typename V > void mln::convert::from_to ( const double & from, Value< V > & to )`

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

**9.27.2.4** `template<typename V > void mln::convert::from_to ( const unsigned & from, Value< V > & to )`

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

**9.27.2.5** `template<typename S > mln::convert::mln_image_from_grid ( typename S::site::grid, bool ) const [inline]`

Convert a point set `pset` into a binary image.

Width of the converted image will be `pset.bbox + 2 * border`.

**9.27.2.6** `template<typename W > mln::convert::mln_image_from_grid ( typename W::site::grid, bool ) const`

Convert a window `win` into a binary image.

**9.27.2.7** `template<typename W > mln::convert::mln_image_from_grid ( typename W::site::grid, mln_weight(W) ) const`

Convert a weighted window `w_win` into an image.

**9.27.2.8** `template<typename N > mln::convert::mln_image_from_grid ( typename N::site::grid, bool ) const`

Convert a neighborhood `nbh` into a binary image.

**9.27.2.9** `template<typename N > mln::convert::mln_window ( N ) const [inline]`

Convert a neighborhood `nbh` into a window.

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

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

References `mln::mln_exact()`.

Referenced by `mln::make_debug_graph_image()`.

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

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

**9.27.2.12** `template<typename I > pw::value_<I> mln::convert::to_fun ( const Image< I > &`  
`ima )`

Convert an image into a function.

**9.27.2.13** `template<typename T > imageId<unsigned> mln::convert::to_image ( const`  
`histo::array< T > & h )`

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

**9.27.2.14** `template<typename I > p_array< typename I::psite > mln::convert::to_p_array (`  
`const Image< I > & img ) [inline]`

Convert an image `img` into a `p_array`.

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

**9.27.2.15** `template<typename S > p_array< typename S::psite > mln::convert::to_p_array (`  
`const Site_Set< S > & pset ) [inline]`

Convert a point set `pset` into a `p_array` (point set vector).

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

**9.27.2.16** `template<typename W > p_array< typename W::psite > mln::convert::to_p_array (`  
`const Window< W > & win, const typename W::psite & p ) [inline]`

Convert a window `win` centered at point `p` into a `p_array` (point set vector).

References `mln::p_array< P >::append()`, and `mln::p_array< P >::reserve()`.

**9.27.2.17** `template<typename N > p_set< typename N::psite > mln::convert::to_p_set ( const`  
`Neighborhood< N > & nbh ) [inline]`

Convert a neighborhood `nbh` into a site set.

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

**9.27.2.18** `template<typename I > p_set< typename I::psite > mln::convert::to_p_set ( const Image< I > & ima ) [inline]`

Convert a binary image `ima` into a site set.

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

**9.27.2.19** `template<typename P , typename C > p_set< P > mln::convert::to_p_set ( const std::set< P, C > & s ) [inline]`

Convert an `std::set` `s` of sites into a site set.

`C` is the comparison functor.

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

**9.27.2.20** `template<typename S > p_set< typename S::psite > mln::convert::to_p_set ( const Site_Set< S > & ps ) [inline]`

Convert any site set `ps` into a 'mlnp\_set' site set.

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

**9.27.2.21** `template<typename W > p_set< typename W::psite > mln::convert::to_p_set ( const Window< W > & win ) [inline]`

Convert a [Window](#) `win` into a site set.

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

**9.27.2.22** `template<typename I > QImage mln::convert::to_qimage ( const Image< I > & ima ) [inline]`

Convert a Milena image to a QImage.

**9.27.2.23** `template<typename N > window< typename N::dpoint > mln::convert::to_upper_window ( const Neighborhood< N > & nbh ) [inline]`

Convert a neighborhood `nbh` into an upper window.

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

**9.27.2.24** `template<typename W > window< typename W::dpsite > mln::convert::to_upper_window ( const Window< W > & win ) [inline]`

Convert a window `nbh` into an upper window.

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

**9.27.2.25** `template<typename D , typename C > window< D > mln::convert::to_window ( const std::set< D, C > & s ) [inline]`

Convert an `std::set` `s` of delta-sites into a window.

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

**9.27.2.26** `template<typename I > window< typename I::site::dpsite > mln::convert::to_window ( const Image< I > & ima ) [inline]`

Convert a binary image `ima` into a window.

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

Referenced by `to_window()`.

**9.27.2.27** `template<typename S > window< typename S::site::dpsite > mln::convert::to_window ( const Site_Set< S > & pset ) [inline]`

Convert a site set `pset` into a window.

References `to_window()`.

## 9.27.3 Variable Documentation

**9.27.3.1** `pw::value_< I > mln::convert::to_fun [inline]`

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

## 9.28 mln::data Namespace Reference

Namespace of image processing routines related to pixel data.

### Namespaces

- namespace [approx](#)  
*Namespace of image processing routines related to pixel levels with approximation.*
- namespace [impl](#)  
*Implementation namespace of data namespace.*
- namespace [naive](#)  
*Namespace of image processing routines related to pixel levels with naive approach.*

### Functions

- `template<typename I, typename O > void abs (const Image< I > &input, Image< O > &output)`
- `template<typename I > void abs\_inplace (Image< I > &input)`
- `template<typename I, typename F > void apply (Image< I > &input, const Function\_v2v< F > &f)`

- `template<typename A , typename I >`  
`A::result compute (const Accumulator< A > &a, const Image< I > &input)`  
*Compute an accumulator onto the pixel values of the image input.*
- `template<typename A , typename I >`  
`A::result compute (Accumulator< A > &a, const Image< I > &input)`  
*Compute an accumulator onto the pixel values of the image input.*
- `template<typename V , typename I >`  
`mln::trait::ch_value< I, V >::ret convert (const V &v, const Image< I > &input)`  
*Convert the image input by changing the value type.*
- `template<typename I , typename W , typename O >`  
`void fast_median (const Image< I > &input, const Window< W > &win, Image< O > &output)`
- `template<typename I , typename D >`  
`void fill (Image< I > &ima, const D &data)`
- `template<typename I , typename J >`  
`void fill_with_image (Image< I > &ima, const Image< J > &data)`  
*Fill the image ima with the values of the image data.*
- `template<typename I , typename W >`  
`mln::trait::concrete< I >::ret median (const Image< I > &input, const Window< W > &win)`
- `template<typename A , typename I >`  
`mln_meta_accu_result (A, typename I::value) compute(const Meta_Accumulator< A > &a)`  
*Compute an accumulator onto the pixel values of the image input.*
- `template<typename I , typename J >`  
`void paste (const Image< I > &input, Image< J > &output)`  
*Paste the contents of image input into the image output.*
- `template<typename I , typename J >`  
`void paste_without_localization (const Image< I > &input, Image< J > &output)`  
*Paste the contents of image input into the image output without taking into account the localization of sites.*
- `template<typename I >`  
`void replace (Image< I > &input, const typename I::value &old_value, const typename I::value &new_value)`
- `template<typename I , typename V >`  
`mln::trait::ch_value< I, V >::ret saturate (const Image< I > &input, const V &min, const V &max)`
- `template<typename V , typename I >`  
`mln::trait::ch_value< I, V >::ret saturate (V v, const Image< I > &input)`
- `template<typename I >`  
`void saturate_inplace (Image< I > &input, const typename I::value &min, const typename I::value &max)`
- `template<typename I >`  
`util::array< unsigned > sort_offsets_increasing (const Image< I > &input)`  
*Sort pixel offsets of the image input wrt increasing pixel values.*
- `template<typename I >`  
`p_array< typename I::psite > sort_psites_decreasing (const Image< I > &input)`



- `template<typename I >`  
`p_array< typename I::psite > sort_sites_increasing (const Image< I > &input)`
- `template<typename V , typename I >`  
`mln::trait::ch_value< I, V >::ret stretch (const V &v, const Image< I > &input)`
- `template<typename I , typename O >`  
`void to_enc (const Image< I > &input, Image< O > &output)`
- `template<typename I1 , typename I2 , typename F >`  
`mln::trait::ch_value< I1, typename F::result >::ret transform (const Image< I1 > &input1, const Image< I2 > &input2, const Function_vv2v< F > &f)`
- `template<typename I , typename F >`  
`mln::trait::ch_value< I, typename F::result >::ret transform (const Image< I > &input, const Function_v2v< F > &f)`
- `template<typename I1 , typename I2 , typename F >`  
`void transform_inplace (Image< I1 > &ima, const Image< I2 > &aux, const Function_vv2v< F > &f)`
- `template<typename I , typename F >`  
`void transform_inplace (Image< I > &ima, const Function_v2v< F > &f)`
- `template<typename A , typename I >`  
`A::result update (Accumulator< A > &a, const Image< I > &input)`
- `template<typename V , typename I >`  
`mln::trait::ch_value< I, V >::ret wrap (const V &v, const Image< I > &input)`

*Routine to wrap values such as 0 -> 0 and [1, lmax] maps to [1, lmax] (using modulus).*

- `template<typename I , typename V >`  
`void fill_with_value (Image< I > &ima, const V &val)`  
*Fill the whole image ima with the single value v.*

## 9.28.1 Detailed Description

Namespace of image processing routines related to pixel data.

## 9.28.2 Function Documentation

### 9.28.2.1 `template<typename I , typename O > void mln::data::abs ( const Image< I > & input, Image< O > & output ) [inline]`

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

#### Parameters

[in] *input* The input image.

[out] *output* The output image.

References transform().

### 9.28.2.2 `template<typename I > void mln::data::abs_inplace ( Image< I > & input ) [inline]`

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

#### Parameters

[in, out] *input* The input image.

References apply().

### 9.28.2.3 `template<typename I , typename F > void mln::data::apply ( Image< I > & input, const Function_v2v< F > & f ) [inline]`

Apply a function-object to the image *input*.

#### Parameters

[in, out] *input* The input image.

[in] *f* The function-object.

This routine runs:

for all *p* of *input*,  $input(p) = f(input(p))$

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

Referenced by `abs_inplace()`, and `saturate_inplace()`.

### 9.28.2.4 `template<typename A , typename I > A::result mln::data::compute ( const Accumulator< A > & a, const Image< I > & input ) [inline]`

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

Be ware that the given accumulator won't be modified and won't store any result.

#### Parameters

[in] *a* An accumulator.

[in] *input* The input image.

#### Returns

The accumulator result.

It fully relies on [data::update](#).

Referenced by `mln::labeled_image< I >::labeled_image()`, `mln::estim::mean()`, `mln::estim::min_max()`, and `mln::estim::sum()`.

### 9.28.2.5 `template<typename A , typename I > A::result mln::data::compute ( Accumulator< A > & a, const Image< I > & input ) [inline]`

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

**Parameters**

- [in, out] *a* An accumulator.
- [in] *input* The input image.

**Returns**

The accumulator result.

It fully relies on [data::update](#).

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

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

**Parameters**

- [in] *v* A value of the destination type.
- [in] *input* The input image.

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

**9.28.2.7** `template<typename I , typename W , typename O > void mln::data::fast_median (   
const Image< I > & input, const Window< W > & win, Image< O > & output )   
[inline]`

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

**Parameters**

- [in] *input* The image to be filtered.
- [in] *win* The window.
- [in, out] *output* The output image.

**Precondition**

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

**9.28.2.8** `template<typename I , typename D > void mln::data::fill ( Image< I > & ima, const D  
& data ) [inline]`

Fill the whole image *ima* with the data provided by *aux*.

**Parameters**

- [in, out] *ima* The image to be filled.
- [in] *data* The auxiliary data to fill the image *ima*.

**Precondition**

*ima* has to be initialized.

Referenced by `mln::draw::box_plain()`, `mln::draw::dashed_line()`, `mln::topo::detach()`, `mln::util::display_branch()`, `mln::transform::distance_and_closest_point_geodesic()`, `mln::duplicate()`, `mln::make::edge_image()`, `mln::labeling::fill_holes()`, `mln::morpho::tree::filter::filter()`, `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::mosaic()`, `mln::debug::slices_2d()`, `mln::morpho::watershed::superpose()`, `mln::debug::superpose()`, `mln::morpho::watershed::topological()`, and `mln::geom::translate()`.

### 9.28.2.9 `template<typename I, typename J > void mln::data::fill_with_image ( Image< I > & ima, const Image< J > & data ) [inline]`

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

#### Parameters

[in, out] *ima* The image to be filled.  
 [in] *data* The image.

#### Warning

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

#### Precondition

`ima.domain <= data.domain`.

### 9.28.2.10 `template<typename I, typename V > void mln::data::fill_with_value ( Image< I > & ima, const V & val ) [inline]`

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

#### Parameters

[in, out] *ima* The image to be filled.  
 [in] *val* The value to assign to all sites.

#### Precondition

`ima` has to be initialized.

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

### 9.28.2.11 `template<typename I, typename W > mln::trait::concrete< I >::ret mln::data::median ( const Image< I > & input, const Window< W > & win )`

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

#### Parameters

[in] *input* The image to be filtered.  
 [in] *win* The window.

#### Precondition

`input` have to be initialized.

**9.28.2.12** `template<typename A , typename I > mln::data::mln_meta_accu_result ( A ,  
typename I::value ) const [inline]`

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

#### Parameters

- [in] *a* A meta-accumulator.
- [in] *input* The input image.

#### Returns

The accumulator result.

**9.28.2.13** `template<typename I , typename J > void mln::data::paste ( const Image< I > &  
input, Image< J > & output ) [inline]`

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

#### Parameters

- [in] *input* The input image providing pixels values.
- [in, out] *output* The image in which values are assigned.

This routine runs:

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

#### Warning

The definition domain of `input` has to be included in the one of `output`; so using [mln::safe\\_image](#) does not make pasting outside the output domain work.

#### Precondition

```
input.domain <= output.domain
```

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

**9.28.2.14** `template<typename I , typename J > void mln::data::paste_without_localization (   
const Image< I > & input, Image< J > & output ) [inline]`

Paste the contents of image `input` into the image `output` without taking into account the localization of sites.

#### Parameters

- [in] *input* The input image providing pixels values.
- [in, out] *output* The image in which values are assigned.

**9.28.2.15** `template<typename I > void mln::data::replace ( Image< I > & input, const typename I::value & old_value, const typename I::value & new_value )`

Replace *old\_value* by *new\_value* in the image *input*

#### Parameters

- [in] *input* The input image.
- [in] *old\_value* The value to be replaced...
- [in] *new\_value* ...by this one.

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

Apply the saturate function to image pixel values.

#### Parameters

- [in] *v* A value of the output type.
- [in] *input* The input image.

The saturation is based on the min and max values of the output value type. This assumes that the range of values in the input image is larger than the one of the output image.

References transform().

**9.28.2.17** `template<typename I , typename V > mln::trait::ch_value< I, V >::ret mln::data::saturate ( const Image< I > & input, const V & min, const V & max ) [inline]`

Apply the saturate function to image pixel values.

#### Parameters

- [in] *input* The input image.
- [in] *min* The minimum output value.
- [in] *max* The maximum output value.

References transform().

**9.28.2.18** `template<typename I > void mln::data::saturate_inplace ( Image< I > & input, const typename I::value & min, const typename I::value & max ) [inline]`

Apply the saturate function to image pixel values.

#### Parameters

- [in, out] *input* The input image.
- [in] *min* The minimum output value.
- [in] *max* The maximum output value

References apply().

**9.28.2.19** `template<typename I> util::array< unsigned > mln::data::sort_offsets_increasing ( const Image< I> & input ) [inline]`

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

**9.28.2.20** `template<typename I> p_array< typename I::psite > mln::data::sort_psites_decreasing ( const Image< I> & input ) [inline]`

Sort psites the image `input` through a function `f` to set the `output` image in decreasing way.

#### Parameters

[in] *input* The input image.

#### Precondition

`input.is_valid`

Referenced by `mln::morpho::tree::min_tree()`.

**9.28.2.21** `template<typename I> p_array< typename I::psite > mln::data::sort_psites_increasing ( const Image< I> & input ) [inline]`

Sort psites the image `input` through a function `f` to set the `output` image in increasing way.

#### Parameters

[in] *input* The input image.

#### Precondition

`input.is_valid`

Referenced by `mln::morpho::tree::max_tree()`.

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

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

#### Parameters

[in] *v* A value to set the output value type.

[in] *input* The input image.

#### Returns

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

#### Precondition

`input.is_valid`

References `mln::data::impl::stretch()`.

**9.28.2.23** `template<typename I , typename O > void mln::data::to_enc ( const Image< I > & input, Image< O > & output ) [inline]`

Set the `output` image with the encoding values of the image `input` pixels.

#### Parameters

- [in] *input* The input image.
- [out] *output* The result image.

#### Precondition

`output.domain >= input.domain`

References `transform()`.

**9.28.2.24** `template<typename I , typename F > mln::trait::ch_value< I , typename F::result >::ret mln::data::transform ( const Image< I > & input, const Function_v2v< F > & f ) [inline]`

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

#### Parameters

- [in] *input* The input image.
- [in] *f* The function.

This routine runs:

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

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

**9.28.2.25** `template<typename I1 , typename I2 , typename F > mln::trait::ch_value< I1, typename F::result >::ret mln::data::transform ( const Image< I1 > & input1, const Image< I2 > & input2, const Function_vv2v< F > & f ) [inline]`

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

#### Parameters

- [in] *input1* The 1st input image.
- [in] *input2* The 2nd input image.
- [in] *f* The function.

This routine runs:

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



**9.28.2.26** `template<typename I1 , typename I2 , typename F > void mln::data::transform_inplace ( Image< I1 > & ima, const Image< I2 > & aux, const Function_vv2v< F > & f )`

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

#### Parameters

- [in] *ima* The image to be transformed.
- [in] *aux* The auxiliary image.
- [in] *f* The function.

This routine runs:

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

**9.28.2.27** `template<typename I , typename F > void mln::data::transform_inplace ( Image< I > & ima, const Function_v2v< F > & f )`

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

#### Parameters

- [in, out] *ima* The image to be transformed.
- [in] *f* The function.

This routine runs:

for all *p* of *ima*,  $ima(p) = f(ima(p))$ .

Referenced by `mln::logical::and_inplace()`, `mln::logical::and_not_inplace()`, `mln::logical::not_inplace()`, `mln::logical::or_inplace()`, `mln::labeling::relabel_inplace()`, and `mln::logical::xor_inplace()`.

**9.28.2.28** `template<typename A , typename I > A::result mln::data::update ( Accumulator< A > & a, const Image< I > & input ) [inline]`

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

#### Parameters

- [in] *a* The accumulator.
- [in] *input* The input image.

#### Returns

The accumulator result.

**9.28.2.29** `template<typename V , typename I > mln::trait::ch_value< I, V >::ret mln::data::wrap ( const V & v, const Image< I > & input )`

Routine to wrap values such as  $0 \rightarrow 0$  and  $[1, lmax]$  maps to  $[1, Lmax]$  (using modulus).

#### Parameters

- [in] *v* The target value type.

[in] *input* Input image.

### Returns

An image with wrapped values.

References transform().

## 9.29 mln::data::approx Namespace Reference

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

### Namespaces

- namespace [impl](#)  
*Implementation namespace of [data::approx](#) namespace.*

### Functions

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

#### 9.29.1 Detailed Description

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

#### 9.29.2 Function Documentation

##### 9.29.2.1 `template<typename I > mln::trait::concrete< I >::ret mln::data::approx::median (const Image< I > & input, const win::rectangle2d & win ) [inline]`

Compute in `output` an approximate of the median filter of image `input` by the 2D rectangle `win`.

#### Parameters

- [in] *input* The image to be filtered.  
[in] *win* The rectangle.

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

#### Precondition

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

Referenced by `median()`.

**9.29.2.2** `template<typename I> mln::trait::concrete< I >::ret mln::data::approx::median ( const Image< I > & input, const win::octagon2d & win ) [inline]`

Compute in `output` an approximate of the median filter of image `input` by the 2D octagon `win`.

#### Parameters

[in] *input* The image to be filtered.

[in] *win* The octagon.

The approximation is based on a vertical median and an horizontal median an two diagonal median.

#### Precondition

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

References `median()`.

**9.29.2.3** `template<typename I> mln::trait::concrete< I >::ret mln::data::approx::median ( const Image< I > & input, const win::disk2d & win ) [inline]`

Compute in `output` an approximate of the median filter of image `input` by the 2D disk `win`.

#### Parameters

[in] *input* The image to be filtered.

[in] *win* The disk.

The approximation is based on a vertical median and an horizontal median an two diagonal median.

#### Precondition

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

References `median()`.

## 9.30 mln::data::approx::impl Namespace Reference

Implementation namespace of [data::approx](#) namespace.

### 9.30.1 Detailed Description

Implementation namespace of [data::approx](#) namespace.

## 9.31 mln::data::impl Namespace Reference

Implementation namespace of `data` namespace.

## Namespaces

- namespace [generic](#)  
*Generic implementation namespace of data namespace.*

## Functions

- `template<typename I , typename J >`  
`void paste\_without\_localization\_fast (const Image< I > &input_, Image< J > &output_)`  
*Paste data to an image without using localization. Performs a point-wise copy.*
- `template<typename I , typename J >`  
`void paste\_without\_localization\_fastest (const Image< I > &input_, Image< J > &output_)`  
*Paste data to an image without using localization. Performs a one-block memory copy.*
- `template<typename I , typename J >`  
`void paste\_without\_localization\_lines (const Image< I > &input_, Image< J > &output_)`  
*Paste data to an image without using localization. Performs a line-per-line memory copy.*
- `template<typename V , typename I >`  
`mln::trait::ch_value< I, V >::ret stretch (const V &v, const Image< I > &input_)`  
*Generic implementation of [data::stretch](#).*
- `template<typename I , typename F >`  
`void transform\_inplace\_lowq (Image< I > &input_, const Function\_v2v< F > &f_)`  
*Specialized implementation.*
- `template<typename A , typename I >`  
`A::result update\_fastest (Accumulator< A > &a_, const Image< I > &input_)`  
*Fastest implementation of [data::update](#).*

### 9.31.1 Detailed Description

Implementation namespace of data namespace.

### 9.31.2 Function Documentation

#### 9.31.2.1 `template<typename I , typename J > void mln::data::impl::paste_without_localization_fast ( const Image< I > & input_ , Image< J > & output_ )` `[inline]`

Paste data to an image without using localization. Performs a point-wise copy.

`input` and `output` must have both the following properties:

- `mln::trait::image::value_alignment::with_grid`
- `mln::trait::image::value_storage::one_block`

- mln::trait::image::value\_access::direct
- mln::trait::image::ext\_domain::some

They must also fulfill the following conditions:

- Same domain size.

#### 9.31.2.2 `template<typename I , typename J > void mln::data::impl::paste_without_localization_fastest ( const Image< I > & input_ , Image< J > & output_ ) [inline]`

Paste data to an image without using localization. Performs a one-block memory copy.

`input` and `output` must have both the following properties:

- mln::trait::image::value\_alignment::with\_grid
- mln::trait::image::value\_storage::one\_block
- mln::trait::image::value\_access::direct
- mln::trait::image::ext\_domain::some

They must also fulfill the following conditions:

- Same border size.
- Same domain size.
- Same value type.

#### 9.31.2.3 `template<typename I , typename J > void mln::data::impl::paste_without_localization_lines ( const Image< I > & input_ , Image< J > & output_ ) [inline]`

Paste data to an image without using localization. Performs a line-per-line memory copy.

`input` and `output` must have both the following properties:

- mln::trait::image::value\_alignment::with\_grid
- mln::trait::image::value\_storage::one\_block
- mln::trait::image::value\_access::direct
- mln::trait::image::ext\_domain::some

They must also fulfill the following conditions:

- Same domain size.
- Same value type.

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

Generic implementation of [data::stretch](#).

#### Parameters

- [in] *v* A value to set the output value type.
- [in] *input* The input image.

#### Returns

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

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

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

**9.31.2.5** `template<typename I , typename F > void mln::data::impl::transform_inplace_lowq (   
Image< I > & input_, const Function_v2v< F > & f_ )`

Specialized implementation.

**9.31.2.6** `template<typename A , typename I > A ::result mln::data::impl::update_fastest (   
Accumulator< A > & a_, const Image< I > & input_ ) [inline]`

Fastest implementation of [data::update](#).

#### Parameters

- [in] *a\_* The accumulator.
- [in] *input\_* The input image.

#### Returns

The accumulator result.

## 9.32 mln::data::impl::generic Namespace Reference

Generic implementation namespace of data namespace.

### Functions

- `template<typename I , typename J >  
void fill\_with\_image (Image< I > &ima_, const Image< J > &data_)`  
*Generic implementation.*
- `template<typename I , typename V >  
void fill\_with\_value (Image< I > &ima_, const V &val)`  
*Fill the whole image ima with the single value v.*

- `template<typename I, typename J >`  
`void paste (const Image< I > &input_, Image< J > &output_)`  
*Generic implementation of `data::paste`.*
- `template<typename I, typename F >`  
`mln::trait::ch_value< I, typename F::result >::ret transform (const Image< I > &input_, const Function_v2v< F > &f_)`  
*Generic implementation of `data::transform`.*
- `template<typename I1, typename I2, typename F >`  
`mln::trait::ch_value< I1, typename F::result >::ret transform (const Image< I1 > &input1_, const Image< I2 > &input2_, const Function_vv2v< F > &f_)`  
*Generic implementation of `data::transform`.*
- `template<typename I1, typename I2, typename F >`  
`void transform_inplace (Image< I1 > &ima_, const Image< I2 > &aux_, const Function_vv2v< F > &f_)`  
*Generic implementation of `transform_inplace`.*
- `template<typename I, typename F >`  
`void transform_inplace (Image< I > &ima_, const Function_v2v< F > &f_)`  
*Generic implementation of `transform_inplace`.*
- `template<typename A, typename I >`  
`A::result update (Accumulator< A > &a_, const Image< I > &input_)`  
*Generic implementation of `data::update`.*

### 9.32.1 Detailed Description

Generic implementation namespace of data namespace.

### 9.32.2 Function Documentation

#### 9.32.2.1 `template<typename I, typename J > void mln::data::impl::generic::fill_with_image ( Image< I > & ima_, const Image< J > & data_ )`

Generic implementation.

#### Parameters

- `[in, out] ima_` The image to be filled.
- `[in] data_` The image.

#### 9.32.2.2 `template<typename I, typename V > void mln::data::impl::generic::fill_with_value ( Image< I > & ima_, const V & val )`

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

**Parameters**

[in, out] *ima\_* The image to be filled.

[in] *val* The value to assign to all sites.

**Precondition**

*ima* has to be initialized.

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

Generic implementation of [data::paste](#).

**Parameters**

[in] *input\_* The input image providing pixels values.

[in, out] *output\_* The image in which values are assigned.

**9.32.2.4** `template<typename I, typename F > mln::trait::ch_value< I, typename F ::result >::ret mln::data::impl::generic::transform ( const Image< I > & input_, const Function_v2v< F > & f_ )`

Generic implementation of [data::transform](#).

**Parameters**

[in] *input\_* The input image.

[in] *f\_* The function.

References `mln::initialize()`.

**9.32.2.5** `template<typename I1, typename I2, typename F > mln::trait::ch_value< I1, typename F ::result >::ret mln::data::impl::generic::transform ( const Image< I1 > & input1_, const Image< I2 > & input2_, const Function_vv2v< F > & f_ )`

Generic implementation of [data::transform](#).

**Parameters**

[in] *input1\_* The 1st input image.

[in] *input2\_* The 2nd input image.

[in] *f\_* The function.

References `mln::initialize()`.



**9.32.2.6** `template<typename I1 , typename I2 , typename F > void mln::data::impl::generic::transform_inplace ( Image< I1 > & ima_, const Image< I2 > & aux_, const Function_vv2v< F > & f_ )`

Generic implementation of `transform_inplace`.

#### Parameters

- [in] *ima\_* The image to be transformed.
- [in] *aux\_* The auxiliary image.
- [in] *f\_* The function.

**9.32.2.7** `template<typename I , typename F > void mln::data::impl::generic::transform_inplace ( Image< I > & ima_, const Function_v2v< F > & f_ )`

Generic implementation of `transform_inplace`.

#### Parameters

- [in, out] *ima\_* The image to be transformed.
- [in] *f\_* The function.

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

Generic implementation of `data::update`.

#### Parameters

- [in] *a\_* The accumulator.
- [in] *input\_* The input image.

#### Returns

The accumulator result.

## 9.33 mln::data::naive Namespace Reference

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

### Namespaces

- namespace `impl`

*Implementation namespace of `data::naive` namespace.*

## Functions

- `template<typename I , typename W , typename O >`  
`void median (const Image< I > &input, const Window< W > &win, Image< O > &output)`  
*Compute in output the median filter of image input by the window win.*

### 9.33.1 Detailed Description

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

### 9.33.2 Function Documentation

- 9.33.2.1 `template<typename I , typename W , typename O > void mln::data::naive::median (`  
`const Image< I > & input, const Window< W > & win, Image< O > & output )`  
`[inline]`

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

#### Parameters

- [in] *input* The image to be filtered.  
[in] *win* The window.  
[in, out] *output* The output image.

This is a NAIVE version for test / comparison purpose so do NOT use it.

#### Precondition

input and output have to be initialized.

#### See also

[mln::data::median](#)

## 9.34 mln::data::naive::impl Namespace Reference

Implementation namespace of [data::naive](#) namespace.

### 9.34.1 Detailed Description

Implementation namespace of [data::naive](#) namespace.

## 9.35 mln::debug Namespace Reference

Namespace of routines that help to debug.

## Namespaces

- namespace `impl`  
*Implementation namespace of debug namespace.*

## Functions

- template<typename I, typename G, typename F >  
void `draw_graph` (`Image`< I > &ima, const `p_vertices`< G, F > &pv, typename I::value vcolor, typename I::value ecolor)  
*Draw an image `ima` from a `mln::p_vertices` `pv`, with value `vcolor` for vertices, value `ecolor` for edges and 0 for the background.*
- template<typename I, typename G, typename F, typename V, typename E >  
void `draw_graph` (`Image`< I > &ima, const `p_vertices`< G, F > &pv, const `Function`< V > &vcolor\_f\_, const `Function`< E > &ecolor\_f\_)  
*Draw an image `ima` from a `mln::p_vertices` `pv`.*
- template<typename I, typename G, typename F, typename V, typename E >  
void `draw_graph` (`Image`< I > &ima, const `p_vertices`< `util::line_graph`< G >, F > &pv, const `Function`< V > &vcolor\_f\_, const `Function`< E > &ecolor\_f\_)  
*Draw an image `ima` from a `mln::p_vertices` `pv`.*
- std::string `filename` (const std::string &filename, int id)  
*Constructs and returns a formatted output file name.*
- signed short `format` (signed char v)  
*Format a signed char to print it properly, i.e., like an integer value.*
- unsigned short `format` (unsigned char v)  
*Format an unsigned char to print it properly, i.e., like an integer value.*
- template<typename T >  
const T & `format` (const T &v)  
*Default version for formatting a value is a no-op.*
- char `format` (bool v)  
*Format a Boolean to print it nicely: "|" for true and "-" for false.*
- template<typename I >  
void `iota` (`Image`< I > &input)
- template<typename I >  
mln::trait::concrete< I >::ret `mosaic` (const `util::array`< I > &input, unsigned n\_horizontal, const typename I::value &bg)  
*Create a single image from an array of image.*
- template<typename I >  
void `println` (const `Image`< I > &input)  
*Print the image `input` on the standard output.*

- `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\_with\_border (const Image< I > &input)`  
*Print the image `input` on the standard output.*
- `void put\_word (image2d< char > &inout, const point2d &word_start, const std::string &word)`  
*Put the word starting at location `word_start` in the image `inout`.*
- `template<typename I >`  
`image2d< typename I::value > slices\_2d (const Image< I > &input, unsigned n_horizontal, unsigned n_vertical, const typename I::value &bg)`  
*Create a 2D image of the slices of the 3D image `input`.*
- `template<typename I >`  
`image2d< typename I::value > slices\_2d (const Image< I > &input, float ratio_hv, const typename I::value &bg)`  
*Create a 2D image of the slices of the 3D image `input`.*
- `template<typename I , typename J >`  
`mln::trait::ch_value< I, value::rgb8 >::ret superpose (const Image< I > &input, const Image< J > &object)`
- `template<typename I , typename J >`  
`mln::trait::ch_value< I, value::rgb8 >::ret superpose (const Image< I > &input_, const Image< J > &object_, const value::rgb8 &object_color)`  
*Superpose two images.*
- `template<typename I >`  
`void z\_order (Image< I > &input)`

### 9.35.1 Detailed Description

Namespace of routines that help to debug.

### 9.35.2 Function Documentation

- 9.35.2.1** `template<typename I , typename G , typename F > void mln::debug::draw_graph ( Image< I > & ima, const p\_vertices< G, F > & pv, typename I::value vcolor, typename I::value ecolor ) [inline]`

Draw an image `ima` from a [mln::p\\_vertices](#) `pv`, with value `vcolor` for vertices, value `ecolor` for edges and 0 for the background.

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

Referenced by `mln::make_debug_graph_image()`.

**9.35.2.2** `template<typename I , typename G , typename F , typename V , typename E > void mln::debug::draw_graph ( Image< I > & ima, const p_vertices< G, F > & pv, const Function< V > & vcolor_f_, const Function< E > & ecolor_f_ ) [inline]`

Draw an image *ima* from a [mln::p\\_vertices](#) *pv*.

Colors for vertices are defined through *vcolor\_f\_*. Colors for edges are defined through *ecolor\_f\_*.

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

**9.35.2.3** `template<typename I , typename G , typename F , typename V , typename E > void mln::debug::draw_graph ( Image< I > & ima, const p_vertices< util::line_graph< G >, F > & pv, const Function< V > & vcolor_f_, const Function< E > & ecolor_f_ ) [inline]`

Draw an image *ima* from a [mln::p\\_vertices](#) *pv*.

Colors for vertices are defined through *vcolor\_f\_*. Colors for edges are defined through *ecolor\_f\_*.

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

**9.35.2.4** `std::string mln::debug::filename ( const std::string & filename, int id = -1 ) [inline]`

Constructs and returns a formatted output file name.

The file name is formatted as follow:

`'filename_prefix'_id_'filename'`

Where:

- `'filename_prefix'` can be set through the global variable `debug::internal::filename_prefix`.

`'postfix_id'` is autoincremented by default. Its value can be forced.

- `'filename'` is the given filename

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

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

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

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

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

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

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

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

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

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

Fill the image `input` with successive values.

**Parameters**

[in, out] *input* The image in which values are assigned.

**9.35.2.10 template<typename I> mln::trait::concrete< I >::ret mln::debug::mosaic ( const util::array< I > & input, unsigned n\_horizontal, const typename I::value & bg ) [inline]**

Create a single image from an array of image.

The size of the output image is defined by:

$\text{width} = n\_horizontal * \max(\text{input}[i].\text{ncols}())$   $\text{height} = (\text{input.size}() / n\_horizontal) * \max(\text{input}[i].\text{nrows}())$

**Returns**

a single image where all the input images are displayed as a mosaic.

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

**9.35.2.11 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()`.

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

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

References `println()`.

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

Print the image `input` on the standard output.

References `mln::geom::bbox()`.

**9.35.2.14** `void mln::debug::put_word ( image2d< char > & inout, const point2d & word_start, const std::string & word ) [inline]`

Put the `word` starting at location `word_start` in the image `inout`.

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

**9.35.2.15** `template<typename I > image2d< typename I::value > mln::debug::slices_2d ( const Image< I > & input, unsigned n_horizontal, unsigned n_vertical, const typename I::value & bg ) [inline]`

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

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

Referenced by `slices_2d()`.

**9.35.2.16** `template<typename I > image2d< typename I::value > mln::debug::slices_2d ( const Image< I > & input, float ratio_hv, const typename I::value & bg )`

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

References `slices_2d()`.

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

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

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

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

Superpose two images.

### Parameters

[in] `input_` An image. Its value type must be convertible toward `value::rgb8` thanks to a conversion operator or `convert::from_to`.

[in] `object_` A scalar or labeled image. Objects used for superposition. have their pixel values different from 0.

[in] `object_color` The color used to draw the objects in `object_`.

### Precondition

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

### Returns

A color image.

References `mln::data::convert()`, `mln::data::fill()`, and `mln::literal::zero`.

Referenced by `superpose()`.

#### 9.35.2.19 `template<typename I > void mln::debug::z_order ( Image< I > & input ) [inline]`

Fill the image `input` with Z-order (curve) values.

##### Parameters

[in, out] *input* The image in which values are assigned.

Reference: [http://en.wikipedia.org/wiki/Z-order\\_\(curve\)](http://en.wikipedia.org/wiki/Z-order_(curve))

## 9.36 `mln::debug::impl` Namespace Reference

Implementation namespace of debug namespace.

### 9.36.1 Detailed Description

Implementation namespace of debug namespace.

## 9.37 `mln::def` Namespace Reference

Namespace for core definitions.

### Typedefs

- typedef short [coord](#)  
*Definition of the default coordinate type: 'short'.*
- typedef float [coordf](#)  
*Definition of the floating coordinate type.*

### Enumerations

- enum  
*Definition of the number of bits of the low quantization threshold.*

### 9.37.1 Detailed Description

Namespace for core definitions.



## 9.37.2 Typedef Documentation

### 9.37.2.1 typedef short mln::def::coord

Definition of the default coordinate type: 'short'.

### 9.37.2.2 typedef float mln::def::coordf

Definition of the floating coordinate type.

## 9.37.3 Enumeration Type Documentation

### 9.37.3.1 anonymous enum

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

## 9.38 mln::display Namespace Reference

Namespace of routines that help to display images.

### Namespaces

- namespace [impl](#)  
*Implementation namespace of display namespace.*

### 9.38.1 Detailed Description

Namespace of routines that help to display images.

## 9.39 mln::display::impl Namespace Reference

Implementation namespace of display namespace.

### Namespaces

- namespace [generic](#)  
*Generic implementation namespace of display namespace.*

### 9.39.1 Detailed Description

Implementation namespace of display namespace.

## 9.40 `mln::display::impl::generic` Namespace Reference

Generic implementation namespace of display namespace.

### 9.40.1 Detailed Description

Generic implementation namespace of display namespace.

## 9.41 `mln::doc` Namespace Reference

The namespace `mln::doc` is only for documentation purpose.

### Classes

- struct [Accumulator](#)  
*Documentation class for `mln::Accumulator`.*
- struct [Box](#)  
*Documentation class for `mln::Box`.*
- struct [Dpoint](#)  
*Documentation class for `mln::Dpoint`.*
- struct [Fastest\\_Image](#)  
*Documentation class for the concept of images that have the speed property set to "fastest".*
- struct [Generalized\\_Pixel](#)  
*Documentation class for `mln::Generalized_Pixel`.*
- struct [Image](#)  
*Documentation class for `mln::Image`.*
- struct [Iterator](#)  
*Documentation class for `mln::Iterator`.*
- struct [Neighborhood](#)  
*Documentation class for `mln::Neighborhood`.*
- struct [Object](#)  
*Documentation class for `mln::Object`.*
- struct [Pixel\\_Iterator](#)  
*Documentation class for `mln::Iterator`.*
- struct [Point\\_Site](#)  
*Documentation class for `mln::Point_Site`.*
- struct [Site\\_Iterator](#)

*Documentation class for mln::Site\_Iterator.*

- struct [Site\\_Set](#)  
*Documentation class for mln::Site\_Set.*
- struct [Value\\_Iterator](#)  
*Documentation class for mln::Value\_Iterator.*
- struct [Value\\_Set](#)  
*Documentation class for mln::Value\_Set.*
- struct [Weighted\\_Window](#)  
*Documentation class for mln::Weighted\_Window.*
- struct [Window](#)  
*Documentation class for mln::Window.*

### 9.41.1 Detailed Description

The namespace [mln::doc](#) is only for documentation purpose. Since concepts are not yet part of the C++ Standard, they are not explicitly expressed in code. Their documentation is handled by their respective ghost class, located in this namespace.

#### Warning

The ghost classes located in [mln::doc](#) should not be used by the client.

## 9.42 mln::draw Namespace Reference

Namespace of drawing routines.

### Functions

- `template<typename I, typename B >`  
void [box](#) ([Image](#)< I > &ima, const [Box](#)< B > &b, const typename I::value &v)
- `template<typename I, typename B >`  
void [box\\_plain](#) ([Image](#)< I > &ima, const [Box](#)< B > &b, const typename I::value &v)
- `template<typename I >`  
void [dashed\\_line](#) ([Image](#)< I > &ima, const typename I::psite &beg, const typename I::psite &end, const typename I::value &v)
- `template<typename I >`  
void [line](#) ([Image](#)< I > &ima, const typename I::psite &beg, const typename I::psite &end, const typename I::value &v)
- `template<typename I >`  
void [plot](#) ([Image](#)< I > &ima, const typename I::point &p, const typename I::value &v)

### 9.42.1 Detailed Description

Namespace of drawing routines.

## 9.42.2 Function Documentation

**9.42.2.1** `template<typename I, typename B> void mln::draw::box ( Image< I > & ima, const Box< B > & b, const typename I::value & v ) [inline]`

Draw a box at value *v* in image *ima*

### Parameters

- [in, out] *ima* The image to be drawn.
- [in] *b* the box to draw.
- [in] *v* The value to assign to all drawn pixels.

### Precondition

- ima* has to be initialized.
- ima* has beg.
- ima* has end.

References `line()`.

**9.42.2.2** `template<typename I, typename B> void mln::draw::box_plain ( Image< I > & ima, const Box< B > & b, const typename I::value & v ) [inline]`

Draw a plain box at value *v* in image *ima*

### Parameters

- [in, out] *ima* The image to be drawn.
- [in] *b* the box to draw.
- [in] *v* The value to assign to all drawn pixels.

### Precondition

- ima* has to be initialized.
- ima* has beg.
- ima* has end.

References `mln::data::fill()`.

**9.42.2.3** `template<typename I> void mln::draw::dashed_line ( Image< I > & ima, const typename I::psite & beg, const typename I::psite & end, const typename I::value & v ) [inline]`

Draw a dashed line at level *v* in image *ima* between the points *beg* and *end*.

### Parameters

- [in, out] *ima* The image to be drawn.
- [in] *beg* The start point to drawn `dashed_line`.
- [in] *end* The end point to drawn `dashed_line`.

[in]  $\nu$  The value to assign to all drawn pixels.

#### Precondition

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

References mln::data::fill().

**9.42.2.4** `template<typename I> void mln::draw::line ( Image< I > & ima, const typename I::psite & beg, const typename I::psite & end, const typename I::value &  $\nu$  )`  
**[inline]**

Draw a line at level  $\nu$  in image ima between the points beg and end.

#### Parameters

[in, out] *ima* The image to be drawn.  
 [in] *beg* The start point to draw line.  
 [in] *end* The end point to draw line.  
 [in]  $\nu$  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().

**9.42.2.5** `template<typename I> void mln::draw::plot ( Image< I > & ima, const typename I::point & p, const typename I::value &  $\nu$  )`

Plot a point at level  $\nu$  in image ima

#### Parameters

[in, out] *ima* The image to be drawn.  
 [in] *p* The point to be plotted.  
 [in]  $\nu$  The value to assign to all drawn pixels.

#### Precondition

ima has to be initialized.  
 ima has p.

## 9.43 mln::estim Namespace Reference

Namespace of estimation materials.

## Functions

- `template<typename I >`  
`mln::value::props< typename I::value >::sum mean (const Image< I > &input)`  
*Compute the mean value of the pixels of image `input`.*
- `template<typename S , typename I , typename M >`  
`void mean (const Image< I > &input, M &result)`  
*Compute the mean value of the pixels of image `input`.*
- `template<typename I >`  
`void min_max (const Image< I > &input, typename I::value &min, typename I::value &max)`  
*Compute the min and max values of the pixels of image `input`.*
- `template<typename I >`  
`mln::value::props< typename I::value >::sum sum (const Image< I > &input)`  
*Compute the sum value of the pixels of image `input`.*
- `template<typename I , typename S >`  
`void sum (const Image< I > &input, S &result)`  
*Compute the sum value of the pixels of image `input`.*

### 9.43.1 Detailed Description

Namespace of estimation materials.

### 9.43.2 Function Documentation

#### 9.43.2.1 `template<typename I > mln::value::props< typename I::value >::sum mln::estim::mean ( const Image< I > & input ) [inline]`

Compute the mean value of the pixels of image `input`.

#### Parameters

[in] *input* The image.

#### Returns

The mean value.

References `mln::data::compute()`.

#### 9.43.2.2 `template<typename S , typename I , typename M > void mln::estim::mean ( const Image< I > & input, M & result ) [inline]`

Compute the mean value of the pixels of image `input`.

#### Parameters

[in] *input* The image.

[out] *result* The mean value.

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

References mln::data::compute().

**9.43.2.3** `template<typename I> void mln::estim::min_max ( const Image< I > & input, typename I::value & min, typename I::value & max ) [inline]`

Compute the min and max values of the pixels of image *input*.

#### Parameters

[in] *input* The image.

[out] *min* The minimum pixel value of *input*.

[out] *max* The maximum pixel value of *input*.

References mln::data::compute().

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

**9.43.2.4** `template<typename I> mln::value::props< typename I::value >::sum mln::estim::sum ( const Image< I > & input ) [inline]`

Compute the sum value of the pixels of image *input*.

#### Parameters

[in] *input* The image.

#### Returns

The sum value.

References mln::data::compute().

**9.43.2.5** `template<typename I, typename S> void mln::estim::sum ( const Image< I > & input, S & result ) [inline]`

Compute the sum value of the pixels of image *input*.

#### Parameters

[in] *input* The image.

[out] *result* The sum value.

References mln::data::compute().

## 9.44 mln::extension Namespace Reference

Namespace of extension tools.

## Functions

- `template<typename I, typename W >`  
`void adjust (const Image< I > &ima, const Window< W > &win)`  
*Adjust the domain extension of image `ima` with the size of the window `win`.*
- `template<typename I, typename W >`  
`void adjust (const Image< I > &ima, const Weighted_Window< W > &wwin)`  
*Adjust the domain extension of image `ima` with the size of the weighted window `wwin`.*
- `template<typename I >`  
`void adjust (const Image< I > &ima, unsigned delta)`  
*Adjust the domain extension of image `ima` with the size `delta`.*
- `template<typename I, typename N >`  
`void adjust (const Image< I > &ima, const Neighborhood< N > &nbh)`  
*Adjust the domain extension of image `ima` with the size of the neighborhood `nbh`.*
- `template<typename I, typename W >`  
`void adjust_duplicate (const Image< I > &ima, const Window< W > &win)`  
*Adjust then duplicate.*
- `template<typename I, typename W >`  
`void adjust_fill (const Image< I > &ima, const Window< W > &win, const typename I::value &val)`  
*Adjust then fill.*
- `template<typename I >`  
`void duplicate (const Image< I > &ima)`  
*Assign the contents of the domain extension by duplicating the values of the inner boundary of image `ima`.*
- `template<typename I >`  
`void fill (const Image< I > &ima, const typename I::value &val)`

### 9.44.1 Detailed Description

Namespace of extension tools.

### 9.44.2 Function Documentation

#### 9.44.2.1 `template<typename I, typename W > void mln::extension::adjust ( const Image< I > & ima, const Window< W > & win )`

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

References `mln::geom::delta()`.

Referenced by `adjust()`, `adjust_duplicate()`, and `adjust_fill()`.



**9.44.2.2** `template<typename I, typename W> void mln::extension::adjust ( const Image< I > & ima, const Weighted_Window< W > & wwin )`

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

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

**9.44.2.3** `template<typename I> void mln::extension::adjust ( const Image< I > & ima, unsigned delta )`

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

References `adjust()`.

**9.44.2.4** `template<typename I, typename N> void mln::extension::adjust ( const Image< I > & ima, const Neighborhood< N > & nbh )`

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

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

**9.44.2.5** `template<typename I, typename W> void mln::extension::adjust_duplicate ( const Image< I > & ima, const Window< W > & win )`

Adjust then duplicate.

References `adjust()`, and `duplicate()`.

**9.44.2.6** `template<typename I, typename W> void mln::extension::adjust_fill ( const Image< I > & ima, const Window< W > & win, const typename I::value & val )`

Adjust then fill.

References `adjust()`, and `fill()`.

**9.44.2.7** `template<typename I> void mln::extension::duplicate ( const Image< I > & ima )`

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

Referenced by `adjust_duplicate()`.

**9.44.2.8** `template<typename I> void mln::extension::fill ( const Image< I > & ima, const typename I::value & val )`

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

#### Parameters

[in, out] *ima* The image whose domain extension is to be filled.

[in] *val* The value to assign.

#### Precondition

*ima* has to be initialized.

Referenced by `adjust_fill()`.

## 9.45 `mln::fun` Namespace Reference

Namespace of functions.

### Namespaces

- namespace [access](#)  
*Namespace for access functions.*
- namespace [i2v](#)  
*Namespace of integer-to-value functions.*
- namespace [n2v](#)  
*Namespace of functions from nil to value.*
- namespace [p2b](#)  
*Namespace of functions from point to boolean.*
- namespace [p2p](#)  
*Namespace of functions from grid point to grid point.*
- namespace [p2v](#)  
*Namespace of functions from point to value.*
- namespace [stat](#)  
*Namespace of statistical functions.*
- namespace [v2b](#)  
*Namespace of functions from value to logic value.*
- namespace [v2i](#)  
*Namespace of value-to-integer functions.*
- namespace [v2v](#)  
*Namespace of functions from value to value.*
- namespace [v2w2v](#)  
*Namespace of bijective functions.*
- namespace [v2w\\_w2v](#)  
*Namespace of functions from value to value.*
- namespace [vv2b](#)  
*Namespace of functions from value to value.*
- namespace [vv2v](#)

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

- namespace [x2p](#)

*Namespace of functions from point to value.*

- namespace [x2v](#)

*Namespace of functions from vector to value.*

- namespace [x2x](#)

*Namespace of functions from vector to vector.*

## Classes

- struct [from\\_accu](#)

*Wrap an accumulator into a function.*

### 9.45.1 Detailed Description

Namespace of functions. Forward declarations.

fun::i2v::array

Forward declaration.

## 9.46 mln::fun::access Namespace Reference

Namespace for access functions.

### 9.46.1 Detailed Description

Namespace for access functions.

## 9.47 mln::fun::i2v Namespace Reference

Namespace of integer-to-value functions.

## Functions

- template<typename T >  
std::ostream & [operator<<](#) (std::ostream &ostr, const array< T > &a)

*Operator<<.*

### 9.47.1 Detailed Description

Namespace of integer-to-value functions.

### 9.47.2 Function Documentation

#### 9.47.2.1 `template<typename T > std::ostream & mln::fun::i2v::operator<< ( std::ostream & ostr, const array< T > & a )`

Operator<<.

## 9.48 `mln::fun::n2v` Namespace Reference

Namespace of functions from nil to value.

### Classes

- struct [white\\_gaussian](#)  
*Generate a White Gaussian Noise.*

### 9.48.1 Detailed Description

Namespace of functions from nil to value.

## 9.49 `mln::fun::p2b` Namespace Reference

Namespace of functions from point to boolean.

### Classes

- struct [antilogy](#)  
*A `p2b` function always returning `false`.*
- struct [tautology](#)  
*A `p2b` function always returning `true`.*

### 9.49.1 Detailed Description

Namespace of functions from point to boolean.

## 9.50 `mln::fun::p2p` Namespace Reference

Namespace of functions from grid point to grid point.

### 9.50.1 Detailed Description

Namespace of functions from grid point to grid point.

## 9.51 mln::fun::p2v Namespace Reference

Namespace of functions from point to value.

### 9.51.1 Detailed Description

Namespace of functions from point to value.

## 9.52 mln::fun::stat Namespace Reference

Namespace of statistical functions.

### 9.52.1 Detailed Description

Namespace of statistical functions.

## 9.53 mln::fun::v2b Namespace Reference

Namespace of functions from value to logic value.

### Classes

- struct [lnot](#)  
*Functor computing logical-not on a value.*
- struct [threshold](#)  
*Threshold function.*

### 9.53.1 Detailed Description

Namespace of functions from value to logic value.

## 9.54 mln::fun::v2i Namespace Reference

Namespace of value-to-integer functions.

### 9.54.1 Detailed Description

Namespace of value-to-integer functions.

## 9.55 mln::fun::v2v Namespace Reference

Namespace of functions from value to value.

### Classes

- class [ch\\_function\\_value](#)  
*Wrap a function  $v2v$  and convert its result to another type.*
- struct [component](#)  
*Functor that accesses the  $i$ -th component of a value.*
- struct [l1\\_norm](#)  
 *$L1$ -norm.*
- struct [l2\\_norm](#)  
 *$L2$ -norm.*
- struct [linear](#)  
*Linear function.  $f(v) = a * v + b$ .  $V$  is the type of input values;  $T$  is the type used to compute the result;  $R$  is the result type.*
- struct [linfty\\_norm](#)  
 *$L$ -infty norm.*

### Variables

- [f\\_hsi\\_to\\_rgb\\_3x8\\_t](#) [f\\_hsi\\_to\\_rgb\\_3x8](#)  
*Global variable.*
- [f\\_hsl\\_to\\_rgb\\_3x8\\_t](#) [f\\_hsl\\_to\\_rgb\\_3x8](#)  
*Global variables.*
- [f\\_rgb\\_to\\_hsi\\_f\\_t](#) [f\\_rgb\\_to\\_hsi\\_f](#)  
*Global variables.*
- [f\\_rgb\\_to\\_hsl\\_f\\_t](#) [f\\_rgb\\_to\\_hsl\\_f](#)  
*Global variables.*

### 9.55.1 Detailed Description

Namespace of functions from value to value.

### 9.55.2 Variable Documentation

#### 9.55.2.1 `f_hsi_to_rgb_3x8_t mln::fun::v2v::f_hsi_to_rgb_3x8`

Global variable.

#### 9.55.2.2 `f_hsl_to_rgb_3x8_t mln::fun::v2v::f_hsl_to_rgb_3x8`

Global variables.

#### 9.55.2.3 `f_rgb_to_hsi_f_t mln::fun::v2v::f_rgb_to_hsi_f`

Global variables.

#### 9.55.2.4 `f_rgb_to_hsl_f_t mln::fun::v2v::f_rgb_to_hsl_f`

Global variables.

## 9.56 `mln::fun::v2w2v` Namespace Reference

Namespace of bijective functions.

### Classes

- struct `cos`  
*Cosinus bijective functor.*

### 9.56.1 Detailed Description

Namespace of bijective functions.

## 9.57 `mln::fun::v2w_w2v` Namespace Reference

Namespace of functions from value to value.

### Classes

- struct `l1_norm`  
*L1-norm.*

- struct [l2\\_norm](#)  
*L2-norm.*
- struct [linfty\\_norm](#)  
*L-infty norm.*

### 9.57.1 Detailed Description

Namespace of functions from value to value.

## 9.58 mln::fun::vv2b Namespace Reference

Namespace of functions from value to value.

### Classes

- struct [eq](#)  
*Functor computing equal between two values.*
- struct [ge](#)  
*Functor computing "greater or equal than" between two values.*
- struct [gt](#)  
*Functor computing "greater than" between two values.*
- struct [implies](#)  
*Functor computing logical-implies between two values.*
- struct [le](#)  
*Functor computing "lower or equal than" between two values.*
- struct [lt](#)  
*Functor computing "lower than" between two values.*

### 9.58.1 Detailed Description

Namespace of functions from value to value.

## 9.59 mln::fun::vv2v Namespace Reference

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



## Classes

- struct [diff\\_abs](#)  
*A functor computing the diff\_absimum of two values.*
- struct [land](#)  
*Functor computing logical-and between two values.*
- struct [land\\_not](#)  
*Functor computing logical and-not between two values.*
- struct [lor](#)  
*Functor computing logical-or between two values.*
- struct [lxor](#)  
*Functor computing logical-xor between two values.*
- struct [max](#)  
*A functor computing the maximum of two values.*
- struct [min](#)  
*A functor computing the minimum of two values.*
- struct [vec](#)  
*A functor computing the vecimum of two values.*

### 9.59.1 Detailed Description

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

## 9.60 mln::fun::x2p Namespace Reference

Namespace of functions from point to value.

## Classes

- struct [closest\\_point](#)  
*FIXME: doxygen + concept checking.*

### 9.60.1 Detailed Description

Namespace of functions from point to value.

## 9.61 mln::fun::x2v Namespace Reference

Namespace of functions from vector to value.

### Classes

- struct [bilinear](#)  
*Represent a bilinear interolation of values from an underlying image.*
- struct [trilinear](#)  
*Represent a trilinear interolation of values from an underlying image.*

### 9.61.1 Detailed Description

Namespace of functions from vector to value.

## 9.62 mln::fun::x2x Namespace Reference

Namespace of functions from vector to vector.

### Classes

- struct [composed](#)  
*Represent a composition of two transformations.*
- struct [linear](#)  
*Represent a linear interolation of values from an underlying image.*
- struct [rotation](#)  
*Represent a rotation function.*
- struct [translation](#)  
*Translation function-object.*

### 9.62.1 Detailed Description

Namespace of functions from vector to vector.

## 9.63 mln::geom Namespace Reference

Namespace of all things related to geometry.

## Namespaces

- namespace [impl](#)  
*Implementation namespace of geom namespace.*

## Classes

- class [complex\\_geometry](#)  
*A functor returning the sites of the faces of a complex where the locations of each 0-face is stored.*

## Functions

- `template<typename S >`  
`box< typename S::site > bbox (const Site_Set< S > &pset)`  
*Compute the precise bounding box of a point set pset.*
- `template<typename I >`  
`box< typename I::site > bbox (const Image< I > &ima)`  
*Compute the precise bounding box of a point set pset.*
- `template<typename W >`  
`box< typename W::psite > bbox (const Window< W > &win)`  
*Compute the precise bounding box of a window win.*
- `template<typename W >`  
`box< typename W::psite > bbox (const Weighted_Window< W > &win)`  
*Compute the precise bounding box of a weighted window win.*
- `template<typename I, typename W >`  
`mln::trait::ch_value< I, unsigned >::ret chamfer (const Image< I > &input_, const W &w_win_, unsigned max=mln_max(unsigned))`  
*Apply chamfer algorithm to a binary image.*
- `template<typename W >`  
`unsigned delta (const Window< W > &win)`  
*Compute the delta of a window win.*
- `template<typename W >`  
`unsigned delta (const Weighted_Window< W > &wwin)`  
*Compute the delta of a weighted window wwin.*
- `template<typename N >`  
`unsigned delta (const Neighborhood< N > &nbh)`  
*Compute the delta of a neighborhood nbh.*
- `template<typename I >`  
`mln::trait::concrete< I >::ret horizontal_symmetry (const Image< I > &input)`  
*Performs a horizontal symmetry.*

- `template<typename I >`  
`I::site::coord max\_col (const Image< I > &ima)`  
*Give the maximum column of an image.*
- `template<typename B >`  
`B::site::coord max\_col (const Box< B > &b)`  
*Give the maximum col of an box 2d or 3d.*
- `template<typename I >`  
`I::site::coord max\_ind (const Image< I > &ima)`  
*Give the maximum ind of an image.*
- `template<typename I >`  
`I::site::coord max\_row (const Image< I > &ima)`  
*Give the maximum row of an image.*
- `template<typename B >`  
`B::site::coord max\_row (const Box< B > &b)`  
*Give the maximum row of an box 2d or 3d.*
- `template<typename I >`  
`I::site::coord max\_sli (const Image< I > &ima)`  
*Give the maximum sli of an image.*
- `std::pair< complex\_image< 2, mln::space\_2complex\_geometry, algebra::vec< 3, float > >, complex\_image< 2, mln::space\_2complex\_geometry, float > > mesh\_corner\_point\_area (const p\_complex< 2, space\_2complex\_geometry > &mesh)`  
*Compute the area “belonging” to normals at vertices.*
- `std::pair< complex\_image< 2, mln::space\_2complex\_geometry, float >, complex\_image< 2, mln::space\_2complex\_geometry, float > > mesh\_curvature (const p\_complex< 2, space\_2complex\_geometry > &mesh)`  
*Compute the principal curvatures of a surface at vertices.*
- `complex\_image< 2, mln::space\_2complex\_geometry, algebra::vec< 3, float > > mesh\_normal (const p\_complex< 2, space\_2complex\_geometry > &mesh)`  
*Compute normals at vertices.*
- `template<typename I >`  
`I::site::coord min\_col (const Image< I > &ima)`  
*Give the minimum column of an image.*
- `template<typename B >`  
`B::site::coord min\_col (const Box< B > &b)`  
*Give the minimum column of an box 2d or 3d.*
- `template<typename I >`  
`I::site::coord min\_ind (const Image< I > &ima)`  
*Give the minimum ind of an image.*

- `template<typename I >`  
`I::site::coord min_row (const Image< I > &ima)`  
*Give the minimum row of an image.*
- `template<typename B >`  
`B::site::coord min_row (const Box< B > &b)`  
*Give the minimum row of an box 2d or 3d.*
- `template<typename I >`  
`I::site::coord min_sli (const Image< I > &ima)`  
*Give the minimum sli of an image.*
- `template<typename I >`  
`unsigned ncols (const Image< I > &ima)`  
*Give the number of columns of an image.*
- `template<typename B >`  
`unsigned ncols (const Box< B > &b)`  
*Give the number of cols of a box 2d or 3d.*
- `template<typename I >`  
`unsigned ninds (const Image< I > &ima)`  
*Give the number of inds of an image.*
- `template<typename I >`  
`unsigned nrows (const Image< I > &ima)`  
*Give the number of rows of an image.*
- `template<typename B >`  
`unsigned nrows (const Box< B > &b)`  
*Give the number of rows of a box 2d or 3d.*
- `template<typename I >`  
`unsigned nsites (const Image< I > &input)`  
*Compute the number of sites of the image input.*
- `template<typename I >`  
`unsigned nslices (const Image< I > &ima)`  
*Give the number of slices of an image.*
- `template<typename S >`  
`void pmin_pmax (const Site_Set< S > &s, typename S::site &pmin, typename S::site &pmax)`  
*Compute the minimum and maximum points, pmin and max, of point set s.*
- `template<typename S >`  
`std::pair< typename S::site, typename S::site > pmin_pmax (const Site_Set< S > &s)`  
*Compute the minimum and maximum points of point set s.*
- `template<typename I >`  
`std::pair< typename I::site, typename I::site > pmin_pmax (const Site_Iterator< I > &p)`  
*Compute the minimum and maximum points when browsing with iterator p.*

- `template<typename I >`  
`void pmin_pmax (const Site_Iterator< I > &p, typename I::site &pmin, typename I::site &pmax)`  
*Compute the minimum and maximum points, `pmin` and `max`, when browsing with iterator `p`.*
- `template<typename I , typename Ext >`  
`mln::trait::concrete< I >::ret rotate (const Image< I > &input, double angle, const Ext &extension)`
- `template<typename B >`  
`B rotate (const Box< B > &box, double angle)`  
*This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts. The rotation center `ref` is set to `box.pcenter()`.*
- `template<typename I , typename Ext , typename S >`  
`mln::trait::concrete< I >::ret rotate (const Image< I > &input, double angle, const Ext &extension, const Site_Set< S > &output_domain)`  
*Perform a rotation from the center of an image.*
- `template<typename I >`  
`mln::trait::concrete< I >::ret rotate (const Image< I > &input, double angle)`  
*This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts. Use `literal::zero` as default value for the extension.*
- `template<typename B >`  
`B rotate (const Box< B > &box_, double angle, const typename B::site &ref)`  
*Rotate a box.*
- `template<typename I , typename N >`  
`mln::trait::concrete< I >::ret seeds2tiling (const Image< I > &ima_, const Neighborhood< N > &nbh)`  
*Take a labeled image `ima_` with seeds and extend them until creating tiles.*
- `template<typename I , typename V , typename Ext , typename S >`  
`mln::trait::concrete< I >::ret translate (const Image< I > &input, const algebra::vec< I::site::dim, V > &ref, const Ext &extension, const Site_Set< S > &output_domain)`  
*Perform a translation from the center of an image.*
- `template<typename I , typename V >`  
`mln::trait::concrete< I >::ret translate (const Image< I > &input, const algebra::vec< I::site::dim, V > &ref)`  
*This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts. Use `literal::zero` as default value for the extension.*
- `template<typename I , typename V , typename Ext >`  
`mln::trait::concrete< I >::ret translate (const Image< I > &input, const algebra::vec< I::site::dim, V > &ref, const Ext &extension)`
- `template<typename I >`  
`mln::trait::concrete< I >::ret vertical_symmetry (const Image< I > &input)`  
*Performs a vertical symmetry.*
- `template<typename I , typename N >`  
`I seeds2tiling_roundness (Image< I > &ima_, const w_window2d_int &w_win, unsigned max, const Neighborhood< N > &nbh_)`

Take a labeled image *ima\_* with seeds and extend them until creating tiles rounder than the primary version.

### 9.63.1 Detailed Description

Namespace of all things related to geometry. Namespace of essential things related to geometry.

### 9.63.2 Function Documentation

**9.63.2.1** `template<typename S > box< typename S::site > mln::geom::bbox ( const Site_Set< S > & pset ) [inline]`

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

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

**9.63.2.2** `template<typename I > box< typename I::site > mln::geom::bbox ( const Image< I > & ima )`

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

References `bbox()`.

**9.63.2.3** `template<typename W > box< typename W::psite > mln::geom::bbox ( const Window< W > & win )`

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

References `mln::literal::origin`.

**9.63.2.4** `template<typename W > box< typename W::psite > mln::geom::bbox ( const Weighted_Window< W > & win )`

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

References `bbox()`.

**9.63.2.5** `template<typename I, typename W > mln::trait::ch_value< I, unsigned >::ret mln::geom::chamfer ( const Image< I > & input_, const W & w_win_, unsigned max = mln_max(unsigned) )`

Apply chamfer algorithm to a binary image.

**9.63.2.6** `template<typename W > unsigned mln::geom::delta ( const Window< W > & win )`

Compute the delta of a window *win*.

Referenced by `mln::extension::adjust()`, and `delta()`.

**9.63.2.7** `template<typename W > unsigned mln::geom::delta ( const Weighted_Window< W > & wwin )`

Compute the delta of a weighted window `wwin`.

References `delta()`.

**9.63.2.8** `template<typename N > unsigned mln::geom::delta ( const Neighborhood< N > & nbh )`

Compute the delta of a neighborhood `nbh`.

References `delta()`.

**9.63.2.9** `template<typename I > mln::trait::concrete< I >::ret mln::geom::horizontal_symmetry ( const Image< I > & input )`

Performs a horizontal symmetry.

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

Give the maximum column of an image.

References `bbox()`.

Referenced by `ncols()`.

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

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

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

Give the maximum ind of an image.

Referenced by `ninds()`.

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

Give the maximum row of an image.

References `bbox()`.

Referenced by `nrows()`.

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

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



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

Give the maximum sli of an image.

References `bbox()`.

Referenced by `nslices()`.

**9.63.2.16** `std::pair< complex_image< 2, mln::space_2complex_geometry, algebra::vec<3, float> >, complex_image< 2, mln::space_2complex_geometry, float > > mln::geom::mesh_corner_point_area ( const p_complex< 2, space_2complex_geometry > & mesh ) [inline]`

Compute the area “belonging” to normals at vertices.

Inspired from the method `Trimesh::need_pointareas` of the Trimesh library.

**See also**

<http://www.cs.princeton.edu/gfx/proj/trimesh2/>

From the documentation of Trimesh:

“Compute the area “belonging” to each vertex or each corner of a triangle (defined as Voronoi area restricted to the 1-ring of a vertex, or to the triangle).”

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

Referenced by `mesh_curvature()`.

**9.63.2.17** `std::pair< complex_image< 2, mln::space_2complex_geometry, float >, complex_image< 2, mln::space_2complex_geometry, float > > mln::geom::mesh_curvature ( const p_complex< 2, space_2complex_geometry > & mesh ) [inline]`

Compute the principal curvatures of a surface at vertices.

These principal curvatures are names `kappa_1` and `kappa_2` in

Sylvie Philipp-Foliguet, Michel Jordan Laurent Najman and Jean Cousty. Artwork 3D Model Database Indexing and Classification.

**Parameters**

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

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

**9.63.2.18** `complex_image< 2, mln::space_2complex_geometry, algebra::vec<3, float> > mln::geom::mesh_normal ( const p_complex< 2, space_2complex_geometry > & mesh ) [inline]`

Compute normals at vertices.

Inspired from the method `Trimesh::need_normals` of the Trimesh library.

**See also**

<http://www.cs.princeton.edu/gfx/proj/trimesh2/>

For simplicity purpose, and contrary to Trimesh, this routine only compute normals from a mesh, not from a cloud of points.

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

Referenced by `mesh_curvature()`.

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

Give the minimum column of an image.

References `bbox()`.

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

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

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

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

Give the minimum ind of an image.

Referenced by `ninds()`.

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

Give the minimum row of an image.

References `bbox()`.

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

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

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

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

Give the minimum sli of an image.

References `bbox()`.

Referenced by `nslices()`.

**9.63.2.25** `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::labeling::impl::compute_fastest()`, `mln::subsampling::gaussian_subsampling()`, `mln::transform::hough()`, `ncols()`, and `mln::subsampling::subsampling()`.

**9.63.2.26** `template<typename B > unsigned mln::geom::ncols ( const Box< B > & b )`

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

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

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

Give the number of inds of an image.

References `max_ind()`, and `min_ind()`.

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

Give the number of rows of an image.

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

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

**9.63.2.29** `template<typename B > unsigned mln::geom::nrows ( const Box< B > & b )`

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

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

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

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

Referenced by `pmin_pmax()`.

**9.63.2.31** `template<typename I > unsigned mln::geom::nslices ( const Image< I > & ima )`  
`[inline]`

Give the number of slices of an image.

References `max_sli()`, and `min_sli()`.

**9.63.2.32** `template<typename S > std::pair< typename S::site, typename S::site > mln::geom::pmin_pmax ( const Site_Set< S > & s ) [inline]`

Compute the minimum and maximum points of point set *s*.

References `nsites()`.

Referenced by `pmin_pmax()`.

**9.63.2.33** `template<typename S > void mln::geom::pmin_pmax ( const Site_Set< S > & s, typename S::site & pmin, typename S::site & pmax ) [inline]`

Compute the minimum and maximum points, `pmin` and `max`, of point set *s*.

References `nsites()`.

**9.63.2.34** `template<typename I > std::pair< typename I::site, typename I::site > mln::geom::pmin_pmax ( const Site_Iterator< I > & p ) [inline]`

Compute the minimum and maximum points when browsing with iterator *p*.

References `pmin_pmax()`.

**9.63.2.35** `template<typename I > void mln::geom::pmin_pmax ( const Site_Iterator< I > & p, typename I::site & pmin, typename I::site & pmax ) [inline]`

Compute the minimum and maximum points, `pmin` and `max`, when browsing with iterator *p*.

**9.63.2.36** `template<typename I > mln::trait::concrete< I >::ret mln::geom::rotate ( const Image< I > & input, double angle )`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts. Use `literal::zero` as default value for the extension.

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

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

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

References `rotate()`.

**9.63.2.38** `template<typename B > B mln::geom::rotate ( const Box< B > & box, double angle )`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts. The rotation center `ref` is set to `box.pcenter()`.

References `rotate()`.

**9.63.2.39** `template<typename I , typename Ext , typename S > mln::trait::concrete< I >::ret  
mln::geom::rotate ( const Image< I > & input, double angle, const Ext & extension,  
const Site_Set< S > & output_domain )`

Perform a rotation from the center of an image.

#### Parameters

[in] *input* An image.

[in] *angle* An angle in degrees.

[in] *extension* [Function](#), image or value which will be used as extension. This extension allows to map values to sites which where not part of the domain before the rotation.

[in] *output\_domain* The domain of the output image. An invalid domain, causes the routine to use a domain large enough to display the whole original image.

#### Returns

An image with the same domain as *input*.

References [bbox\(\)](#), [mln::compose\(\)](#), [mln::duplicate\(\)](#), [mln::initialize\(\)](#), [mln::mln\\_exact\(\)](#), [mln::literal::origin](#), and [mln::data::paste\(\)](#).

Referenced by [rotate\(\)](#).

**9.63.2.40** `template<typename B > B mln::geom::rotate ( const Box< B > & box_, double angle,  
const typename B::site & ref )`

Rotate a box.

FIXME: the return type may be too generic and may lead to invalid covariance.

References [mln::compose\(\)](#), [mln::literal::origin](#), and [mln::accu::shape::bbox< P >::to\\_result\(\)](#).

**9.63.2.41** `template<typename I , typename N > mln::trait::concrete< I >::ret  
mln::geom::seeds2tiling ( const Image< I > & ima_, const Neighborhood< N > & nbh  
) [inline]`

Take a labeled image *ima\_* with seeds and extend them until creating tiles.

#### Parameters

[in, out] *ima\_* The labeled image with seed.

[in] *nbh* The neighborhood to use on this algorithm.

#### Returns

A tiled image.

#### Precondition

*ima\_* has to be initialized.

References [mln::geom::impl::seeds2tiling\(\)](#).

**9.63.2.42** `template<typename I , typename N > I mln::geom::seeds2tiling_roundness ( Image< I > & ima_ , const w_window2d_int & w_win , unsigned max , const Neighborhood< N > & nbh_ ) [inline]`

Take a labeled image `ima_` with seeds and extend them until creating tiles rounder than the primary version.

#### Parameters

- [in, out] *ima\_* The labeled image with seed.
- [in] *w\_win* The weight window using by [geom::chamfer](#) to compute distance.
- [in] *max* Unsigned using by [geom::chamfer](#) to compute the distance.
- [in] *nbh\_* The neighborhood to use on this algorithm.

#### Precondition

`ima_` has to be initialized.

**9.63.2.43** `template<typename I , typename V > mln::trait::concrete< I >::ret mln::geom::translate ( const Image< I > & input , const algebra::vec< I::site::dim, V > & ref )`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts. Use [literal::zero](#) as default value for the extension.

References `translate()`, and `mln::literal::zero`.

**9.63.2.44** `template<typename I , typename V , typename Ext , typename S > mln::trait::concrete< I >::ret mln::geom::translate ( const Image< I > & input , const algebra::vec< I::site::dim, V > & ref , const Ext & extension , const Site_Set< S > & output_domain )`

Perform a translation from the center of an image.

#### Parameters

- [in] *input* An image.
- [in] *ref* The translation vector.
- [in] *extension* [Function](#), image or value which will be used as extension. This extension allows to map values to sites which where not part of the domain before the translation.
- [in] *output\_domain* The domain of the output image. An invalid domain, causes the routine to use the translated `input_domain`.

#### Returns

An image with the same domain as `input`.

References `mln::extend()`, `mln::data::fill()`, and `mln::mln_exact()`.

Referenced by `translate()`.

**9.63.2.45** `template<typename I , typename V , typename Ext > mln::trait::concrete< I >::ret mln::geom::translate ( const Image< I > & input, const algebra::vec< I::site::dim, V > & ref, const Ext & extension )`

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

References translate().

**9.63.2.46** `template<typename I > mln::trait::concrete< I >::ret mln::geom::vertical_symmetry ( const Image< I > & input )`

Performs a vertical symmetry.

## 9.64 mln::geom::impl Namespace Reference

Implementation namespace of geom namespace.

### Functions

- `template<typename I , typename N > mln::trait::concrete< I >::ret seeds2tiling (const Image< I > &ima_, const Neighborhood< N > &nbh_)`

*Generic implementation of geom::seed2tiling.*

### 9.64.1 Detailed Description

Implementation namespace of geom namespace.

### 9.64.2 Function Documentation

**9.64.2.1** `template<typename I , typename N > mln::trait::concrete< I >::ret mln::geom::impl::seeds2tiling ( const Image< I > & ima_, const Neighborhood< N > & nbh_ ) [inline]`

Generic implementation of geom::seed2tiling.

#### Parameters

[in, out] *ima\_* The labeled image with seed.

[in] *nbh\_* The neighborhood to use on this algorithm.

References mln::duplicate(), mln::p\_queue< P >::front(), mln::p\_queue< P >::pop(), and mln::p\_queue< P >::push().

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

## 9.65 mln::graph Namespace Reference

Namespace of graph related routines.

### Functions

- `template<typename G , typename F > F::result compute (const Graph< G > &g_, F &functor)`  
*Base routine to compute attributes on a graph.*
- `template<typename I , typename N , typename L > mln::trait::ch_value< I, L >::ret labeling (const Image< I > &graph_image_, const Neighborhood< N > &nbh_, L &nlabels)`  
*Label graph components.*
- `template<typename I , typename M > graph_elt_neighborhood_if< mln_graph(I), typename I::domain_t, M > to_neighb (const Image< I > &graph_image_, const Image< M > &graph_mask_image_)`  
*Make a custom graph neighborhood from a mask image.*
- `template<typename I , typename M > graph_elt_window_if< mln_graph(I), typename I::domain_t, M > to_win (const Image< I > &graph_image_, const Image< M > &graph_mask_image_)`  
*Make a custom graph window from a mask image.*

### 9.65.1 Detailed Description

Namespace of graph related routines.

### 9.65.2 Function Documentation

#### 9.65.2.1 `template<typename G , typename F > F::result mln::graph::compute ( const Graph< G > & g_, F & functor )`

Base routine to compute attributes on a graph.

#### Parameters

- [in] `g_` A graph.
- [in] `functor` A functor implementing the right interface.

#### Returns

The computed data.

#### See also

`canvas::browsing::depth_first_search`



**9.65.2.2** `template<typename I, typename N, typename L > mln::trait::ch_value< I, L >::ret mln::graph::labeling ( const Image< I > & graph_image_, const Neighborhood< N > & nbh_, L & nlabels )`

Label graph components.

[Vertex](#) with id 0, usually used to represent the background component, will be labeled with an id different from 0. Therefore, the labeling starts from 1.

#### Parameters

[in] *graph\_image\_* A graph image (

#### See also

[vertex\\_image](#), [edge\\_image](#)).

#### Parameters

[in] *nbh\_* A graph neighborhood.

[in, out] *nlabels* The number of labels found.

#### Returns

a [Graph](#) image of labels.

References [mln::labeling::blobs\(\)](#), [mln::data::fill\(\)](#), and [mln::initialize\(\)](#).

**9.65.2.3** `template<typename I, typename M > graph_elt_neighborhood_if< mln_graph(I), typename I::domain_t, M > mln::graph::to_neighb ( const Image< I > & graph_image_, const Image< M > & graph_mask_image_ )`

Make a custom graph neighborhood from a mask image.

#### Parameters

[in] *graph\_image\_* A graph image (

#### See also

[vertex\\_image](#) and [edge\\_image](#)).

#### Parameters

[in] *graph\_mask\_image\_* A graph image of bool used as a mask.

#### Returns

A masked neighborhood on graph.

**9.65.2.4** `template<typename I, typename M > graph_elt_window_if< mln_graph(I), typename I::domain_t, M > mln::graph::to_win ( const Image< I > & graph_image_, const Image< M > & graph_mask_image_ )`

Make a custom graph window from a mask image.

**Parameters**

[in] *graph\_image\_* A graph image (

**See also**

[vertex\\_image](#) and [edge\\_image](#)).

**Parameters**

[in] *graph\_mask\_image\_* A graph image of bool used as a mask.

**Returns**

A masked window on graph.

## 9.66 mln::grid Namespace Reference

Namespace of grids definitions.

### 9.66.1 Detailed Description

Namespace of grids definitions. Compute the image::space trait from a point type.

## 9.67 mln::histo Namespace Reference

Namespace of histograms.

### Namespaces

- namespace [impl](#)  
*Implementation namespace of histo namespace.*

### Classes

- struct [array](#)  
*Generic histogram class over a value set with type T.*

### Functions

- `template<typename I >`  
`histo::array< typename I::value > compute (const Image< I > &input)`  
*Compute the histogram of image input.*
- `template<typename I >`  
`mln::trait::concrete< I >::ret equalize (const Image< I > &input)`  
*Equalizes the histogram of image input.*

## 9.67.1 Detailed Description

Namespace of histograms.

## 9.67.2 Function Documentation

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

Compute the histogram of image `input`.

Referenced by `equalize()`.

**9.67.2.2** `template<typename I> mln::trait::concrete< I >::ret mln::histo::equalize ( const Image< I> & input )`

Equalizes the histogram of image `input`.

### Author

J. Fabrizio, R. Levillain

References `compute()`, and `mln::initialize()`.

## 9.68 mln::histo::impl Namespace Reference

Implementation namespace of `histo` namespace.

### Namespaces

- namespace [generic](#)  
*Generic implementation namespace of `histo` namespace.*

## 9.68.1 Detailed Description

Implementation namespace of `histo` namespace.

## 9.69 mln::histo::impl::generic Namespace Reference

Generic implementation namespace of `histo` namespace.

## 9.69.1 Detailed Description

Generic implementation namespace of `histo` namespace.

## 9.70 mln::impl Namespace Reference

Implementation namespace of mln namespace.

### 9.70.1 Detailed Description

Implementation namespace of mln namespace.

## 9.71 mln::io Namespace Reference

Namespace of input/output handling.

### Namespaces

- namespace [cloud](#)  
*Namespace of cloud input/output handling.*
- namespace [dicom](#)  
*Namespace of DICOM input/output handling.*
- namespace [dump](#)  
*Namespace of dump input/output handling.*
- namespace [fits](#)  
*Namespace of fits input/output handling.*
- namespace [fld](#)  
*Namespace of pgm input/output handling.*
- namespace [magick](#)  
*Namespace of magick input/output handling.*
- namespace [off](#)  
*Namespace of off input/output handling.*
- namespace [pbm](#)  
*Namespace of pbm input/output handling.*
- namespace [pbms](#)  
*Namespace of pbms input/output handling.*
- namespace [pfm](#)  
*Namespace of pfm input/output handling.*
- namespace [pgm](#)  
*Namespace of pgm input/output handling.*
- namespace [pgms](#)

*Namespace of pgms input/output handling.*

- namespace [plot](#)  
*Namespace of plot input/output handling.*
- namespace [pnm](#)  
*Namespace of pnm input/output handling.*
- namespace [pnms](#)  
*Namespace of pnms input/output handling.*
- namespace [ppm](#)  
*Namespace of ppm input/output handling.*
- namespace [ppms](#)  
*Namespace of ppms input/output handling.*
- namespace [raw](#)  
*Namespace of raw input/output handling.*
- namespace [tiff](#)  
*Namespace of tiff input/output handling.*
- namespace [txt](#)  
*Namespace of txt input/output handling.*

### 9.71.1 Detailed Description

Namespace of input/output handling.

## 9.72 mln::io::cloud Namespace Reference

Namespace of cloud input/output handling.

### Functions

- `template<typename P >`  
`void load (p_array< P > &arr, const std::string &filename)`  
*Load a cloud of points.*
- `template<typename P >`  
`void save (const p_array< P > &arr, const std::string &filename)`  
*Load a cloud of points.*

### 9.72.1 Detailed Description

Namespace of cloud input/output handling.

### 9.72.2 Function Documentation

**9.72.2.1** `template<typename P > void mln::io::cloud::load ( p_array< P > & arr, const std::string & filename )`

Load a cloud of points.

#### Parameters

- [in, out] *arr* the site set where to load the data.
- [in] *filename* file to load.

**9.72.2.2** `template<typename P > void mln::io::cloud::save ( const p_array< P > & arr, const std::string & filename )`

Load a cloud of points.

#### Parameters

- [in] *arr* the cloud of points to save.
- [in] *filename* the destination.

## 9.73 mln::io::dicom Namespace Reference

Namespace of DICOM input/output handling.

### Classes

- struct [dicom\\_header](#)  
*Store dicom file header.*

### Functions

- [dicom\\_header get\\_header](#) (const std::string &filename)  
*Retrieve header in a dicom file.*
- `template<typename I > void load (Image< I > &ima, const std::string &filename)`

### 9.73.1 Detailed Description

Namespace of DICOM input/output handling.

## 9.73.2 Function Documentation

### 9.73.2.1 dicom\_header mln::io::dicom::get\_header ( const std::string & filename )

Retrieve header in a dicom file.

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

### 9.73.2.2 template<typename I > void mln::io::dicom::load ( Image< I > & ima, const std::string & filename ) [inline]

Load a DICOM file in a Milena image.

#### Parameters

[out] *ima* A reference to the image which will receive data.

[in] *filename* The source.

Common compilation flags to link to gdcm if this file is used:

-lgdcmCommon -lgdcmDICT -lgdcmDSED -lgdcmIOD -lgdcmMSFF -lgdcmexpat -lgdcmjpeg12 -lgdcmjpeg16 -lgdcmjpeg8 -lgdcmopenjpeg -lgdcmuuid -lgdcmzlib

References mln::initialize(), and mln::point< G, C >::to\_vec().

## 9.74 mln::io::dump Namespace Reference

Namespace of dump input/output handling.

### Classes

- struct [dump\\_header](#)  
*Store dump file header.*

### Functions

- [dump\\_header get\\_header](#) (const std::string &filename)  
*Retrieve header in a dump file.*
- template<typename I >  
void [load](#) (Image< I > &ima\_, const std::string &filename)  
*Load a Milena image by dumped into a file.*
- template<typename I >  
void [save](#) (const Image< I > &ima\_, const std::string &filename)  
*Save a Milena image by dumping its data to a file.*

### 9.74.1 Detailed Description

Namespace of dump input/output handling.

### 9.74.2 Function Documentation

#### 9.74.2.1 `dump_header mln::io::dump::get_header ( const std::string & filename )`

Retrieve header in a dump file.

References `mln::util::array< T >::resize()`.

#### 9.74.2.2 `template<typename I > void mln::io::dump::load ( Image< I > & ima_, const std::string & filename )`

Load a Milena image by dumped into a file.

##### Parameters

[in, out] *ima\_* The image to load.

[in] *filename* the destination.

#### 9.74.2.3 `template<typename I > void mln::io::dump::save ( const Image< I > & ima_, const std::string & filename )`

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

##### Parameters

[in] *ima\_* The image to save.

[in] *filename* the destination.

## 9.75 mln::io::fits Namespace Reference

Namespace of fits input/output handling.

### Functions

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

### 9.75.1 Detailed Description

Namespace of fits input/output handling.



## 9.75.2 Function Documentation

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

Load a fits image in a Milena image.

### Parameters

- `[out]` *ima* A reference to the image2d<float> which will receive data.
- `[in]` *filename* The source.

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

Load a fits image in a image2d<float>.

### Parameters

- `[in]` *filename* The image source.

### Returns

An image2d<float> which contains loaded data.

## 9.76 mln::io::fld Namespace Reference

Namespace of pgm input/output handling.

### Classes

- struct [fld\\_header](#)  
*Define the header structure of an AVS field data file.*

### Functions

- `template<typename I >`  
`void load (Image< I > &ima_, const char *filename)`  
*Load an image from an AVS field file.*
- `fld_header read_header (std::istream &ins)`  
*Read the header form an AVS field file.*
- `void write_header (std::ostream &file, const fld_header &h)`  
*Write the AVS header in a file.*

### 9.76.1 Detailed Description

Namespace of pgm input/output handling.

## 9.76.2 Function Documentation

### 9.76.2.1 `template<typename I> void mln::io::fld::load ( Image< I > & ima_, const char * filename ) [inline]`

Load an image from an AVS field file.

#### Parameters

- [in, out] *ima\_* The image to load.
- [in] *filename* The path to the AVS file.

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

### 9.76.2.2 `fld_header mln::io::fld::read_header ( std::istream & ins ) [inline]`

Read the header form an AVS field file.

#### Parameters

- ins* The file to read.

#### Returns

The header.

Referenced by `load()`.

### 9.76.2.3 `void mln::io::fld::write_header ( std::ostream & file, const fld_header & h ) [inline]`

Write the AVS header in a file.

#### Parameters

- file* The file to write.
- h* The AVS header.

## 9.77 `mln::io::magick` Namespace Reference

Namespace of magick input/output handling.

### Functions

- `template<typename I> void load (Image< I > &ima, const std::string &filename)`  
*Load data from a file into a Milena image using Magick++.*
- `template<typename I> void save (const Image< I > &ima, const std::string &filename)`  
*Save a Milena image into a file using Magick++.*

### 9.77.1 Detailed Description

Namespace of magick input/output handling.

### 9.77.2 Function Documentation

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

Load data from a file into a Milena image using Magick++.

#### Parameters

- [out] *ima* The image data are loaded into.
- [in] *filename* The name of the input file.

References mln::initialize().

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

Save a Milena image into a file using Magick++.

#### Parameters

- [out] *ima* The image to save.
- [in] *filename* The name of the output file.

## 9.78 mln::io::off Namespace Reference

Namespace of off input/output handling.

### Functions

- void `load (bin_2complex_image3df &ima, const std::string &filename)`  
*Load a (binary) OFF image into a complex image.*
- void `save (const bin_2complex_image3df &ima, const std::string &filename)`  
*Save a (binary) OFF image into a complex image.*
- `template<typename I >`  
void `save_bin_alt (const I &ima, const std::string &filename)`  
*FIXME: Similar to mln::io::off::save(const bin\_2complex\_image3df&, const std::string&), but does not save faces whose value is 'false'.*

### 9.78.1 Detailed Description

Namespace of off input/output handling.

## 9.78.2 Function Documentation

### 9.78.2.1 void mln::io::off::load ( bin\_2complex\_image3df & *ima*, const std::string & *filename* )

Load a (binary) OFF image into a complex image.

Load a 3x8-bit RGB (color) OFF image into a complex image.

Load a floating-point OFF image into a complex image.

#### Parameters

[out] *ima* A reference to the image to construct.

[in] *filename* The name of the file to load.

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

#### Parameters

[out] *ima* A reference to the image to construct.

[in] *filename* The name of the file to load.

Read floating-point data is attached to 2-faces only; 1-faces and 0-faces are set to 0.0f.

### 9.78.2.2 void mln::io::off::save ( const bin\_2complex\_image3df & *ima*, const std::string & *filename* )

Save a (binary) OFF image into a complex image.

Save a 3x8-bit RGB (color) OFF image into a complex image.

Save a floating-point value grey-level OFF image into a complex image.

Save an 8-bit grey-level OFF image into a complex image.

#### Parameters

[in] *ima* The image to save.

[in] *filename* The name of the file where to save the image.

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

#### Parameters

[in] *ima* The image to save.

[in] *filename* The name of the file where to save the image.

Only data is attached to 2-faces is saved; the OFF file cannot store data attached to faces of other dimensions.

### 9.78.2.3 template<typename I> void mln::io::off::save\_bin\_alt ( const I & *ima*, const std::string & *filename* )

FIXME: Similar to [mln::io::off::save\(const bin\\_2complex\\_image3df&, const std::string&\)](#), but does not save faces whose value is 'false'.

## 9.79 mln::io::pbm Namespace Reference

Namespace of pbm input/output handling.

### Namespaces

- namespace [impl](#)  
*Namespace of pbm implementation details.*

### Functions

- void [load](#) (image2d< bool > &ima, const std::string &filename)  
*Load a pbm image in a Milena image.*
- [image2d](#)< bool > [load](#) (const std::string &filename)  
*Load a pbm image in a image2d<float>.*
- template<typename I >  
void [save](#) (const [Image](#)< I > &ima, const std::string &filename)

#### 9.79.1 Detailed Description

Namespace of pbm input/output handling.

#### 9.79.2 Function Documentation

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

Load a pbm image in a Milena image.

##### Parameters

- [out] *ima* A reference to the image2d<bool> which will receive data.  
[in] *filename* The source.

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

Load a pbm image in a image2d<float>.

##### Parameters

- [in] *filename* The image source.

##### Returns

An image2d<float> which contains loaded data.

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

Save a Milena image as a pbm image.

#### Parameters

- [in] *ima* The image to save.
- [in, out] *filename* the destination.

## 9.80 mln::io::pbm::impl Namespace Reference

Namespace of pbm implementation details.

### 9.80.1 Detailed Description

Namespace of pbm implementation details.

## 9.81 mln::io::pbms Namespace Reference

Namespace of pbms input/output handling.

### Namespaces

- namespace [impl](#)  
*Namespace of pbms implementation details.*

### Functions

- void [load](#) ([image3d](#)< bool > &*ima*, const [util::array](#)< std::string > &*filenames*)  
*Load pbms images as slices of a 3D Milena image.*

### 9.81.1 Detailed Description

Namespace of pbms input/output handling.

### 9.81.2 Function Documentation

**9.81.2.1** `void mln::io::pbms::load ( image3d< bool > & ima, const util::array< std::string > & filenames ) [inline]`

Load pbms images as slices of a 3D Milena image.

### Parameters

- [out] *ima* A reference to the 3D image which will receive data.
- [in] *filenames* The list of 2D images to load..

## 9.82 mln::io::pbms::impl Namespace Reference

Namespace of pbms implementation details.

### 9.82.1 Detailed Description

Namespace of pbms implementation details.

## 9.83 mln::io::pfm Namespace Reference

Namespace of pfm input/output handling.

### Namespaces

- namespace [impl](#)  
*Implementation namespace of pfm namespace.*

### Functions

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

### 9.83.1 Detailed Description

Namespace of pfm input/output handling.

### 9.83.2 Function Documentation

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

Load a pfm image in a Milena image.

**Parameters**

- [out] *ima* A reference to the image2d<float> which will receive data.  
 [in] *filename* The source.

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

Load a pfm image in a image2d<float>.

**Parameters**

- [in] *filename* The image source.

**Returns**

An image2d<float> which contains loaded data.

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

Save a Milena image as a pfm image.

**Parameters**

- [in] *ima* The image to save.  
 [in, out] *filename* the destination.

**9.84 mln::io::pfm::impl Namespace Reference**

Implementation namespace of pfm namespace.

**9.84.1 Detailed Description**

Implementation namespace of pfm namespace.

**9.85 mln::io::pgm Namespace Reference**

Namespace of pgm input/output handling.

**Functions**

- template<typename I >  
void [load](#) (Image< I > &ima, const std::string &filename)  
*Load a pgm image in a Milena image.*
- template<typename V >  
[image2d](#)< V > [load](#) (const std::string &filename)



*Load a pgm image in a Milena image.*

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

### 9.85.1 Detailed Description

Namespace of pgm input/output handling.

### 9.85.2 Function Documentation

#### 9.85.2.1 `template<typename I > void mln::io::pgm::load ( Image< I > & ima, const std::string & filename ) \[inline\]`

Load a pgm image in a Milena image.

##### Parameters

- `[out]` *ima* A reference to the image which will receive data.
- `[in]` *filename* The source.

#### 9.85.2.2 `template<typename V > image2d< V > mln::io::pgm::load ( const std::string & filename ) \[inline\]`

Load a pgm image in a Milena image.

To use this routine, you should specialize the template with the value type of the image loaded. (ex : `load<value::int_u8>("...")`)

##### Parameters

- `[in]` *filename* The image source.

##### Returns

An [image2d](#) which contains loaded data.

#### 9.85.2.3 `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

- `[in]` *ima* The image to save.
- `[in, out]` *filename* the destination.

## 9.86 mln::io::pgms Namespace Reference

Namespace of pgms input/output handling.

## Functions

- `template<typename V >`  
`void load (image3d< V > &ima, const util::array< std::string > &filenames)`

*Load pgm images as slices of a 3D Milena image.*

### 9.86.1 Detailed Description

Namespace of pgms input/output handling.

### 9.86.2 Function Documentation

- 9.86.2.1** `template<typename V > void mln::io::pgms::load ( image3d< V > & ima, const util::array< std::string > & filenames ) [inline]`

Load pgm images as slices of a 3D Milena image.

#### Parameters

- [out] *ima* A reference to the 3D image which will receive data.  
 [in] *filenames* The list of 2D images to load..

## 9.87 mln::io::plot Namespace Reference

Namespace of plot input/output handling.

## Functions

- `template<typename I >`  
`void load (util::array< I > &arr, const std::string &filename)`
- `template<typename T >`  
`void save (const histo::array< T > &arr, const std::string &filename)`
- `template<typename T >`  
`void save (util::array< T > &arr, const std::string &filename, int start_value)`

*Save a Milena array in a plot file.*

- `template<typename I >`  
`void save (const image1d< I > &ima, const std::string &filename)`

*Save a Milena 1D image in a plot file.*

### 9.87.1 Detailed Description

Namespace of plot input/output handling.

## 9.87.2 Function Documentation

**9.87.2.1** `template<typename I> void mln::io::plot::load ( util::array< I > & arr, const std::string & filename ) [inline]`

Load a Milena 1D image from a plot file.

### Parameters

[in] *ima* A reference to the image to load.

[out] *filename* The output file.

[in] *start\_value* The start index value of the plot (optional).

Load a Milena array from a plot file.

### Parameters

[in] *arr* A reference to the array to load.

[out] *filename* The output file.

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

**9.87.2.2** `template<typename T> void mln::io::plot::save ( const histo::array< T > & arr, const std::string & filename ) [inline]`

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

**9.87.2.3** `template<typename T> void mln::io::plot::save ( util::array< T > & arr, const std::string & filename, int start_value )`

Save a Milena array in a plot file.

### Parameters

[in] *arr* A reference to the array to save.

[out] *filename* The output file.

[in] *start\_value* The start index value of the plot (optional).

**9.87.2.4** `template<typename I> void mln::io::plot::save ( const image1d< I > & ima, const std::string & filename )`

Save a Milena 1D image in a plot file.

### Parameters

[in] *ima* A reference to the image to save.

[out] *filename* The output file.

## 9.88 mln::io::pnm Namespace Reference

Namespace of pnm input/output handling.

### Namespaces

- namespace [impl](#)  
*Namespace of pnm's implementation details.*

### Functions

- `template<typename V > image2d< V > load (char type_, const std::string &filename)`  
*main function : load pnm format*
- `template<typename I > void load (char type_, Image< I > &ima_, const std::string &filename)`  
*An other way to load pnm files : the destination is an argument to check if the type match the file to load.*
- `template<typename I > void load\_ascii\_builtin (std::ifstream &file, I &ima)`  
*load\_ascii for builtin value types.*
- `template<typename I > void load\_ascii\_value (std::ifstream &file, I &ima)`  
*load\_ascii for Milena value types.*
- `template<typename I > void load\_raw\_2d (std::ifstream &file, I &ima)`  
*load\_raw\_2d.*
- `template<typename V > unsigned int max\_component (const V &)`  
*Give the maximum value which can be stored as a component value type V.*
- `template<typename I > void save (char type, const Image< I > &ima_, const std::string &filename)`

### 9.88.1 Detailed Description

Namespace of pnm input/output handling.

### 9.88.2 Function Documentation

**9.88.2.1** `template<typename V > image2d<V> mln::io::pnm::load ( char type_, const std::string & filename ) [inline]`

main function : load pnm format

References `load_raw_2d()`, and `max_component()`.

**9.88.2.2** `template<typename I> void mln::io::pnm::load ( char type_, Image< I > & ima_, const std::string & filename ) [inline]`

An other way to load pnm files : the destination is an argument to check if the type match the file to load.  
References `mln::make::box2d()`, `load_raw_2d()`, and `max_component()`.

**9.88.2.3** `template<typename I> void mln::io::pnm::load_ascii_builtin ( std::ifstream & file, I & ima ) [inline]`

`load_ascii` for builtin value types.

**9.88.2.4** `template<typename I> void mln::io::pnm::load_ascii_value ( std::ifstream & file, I & ima ) [inline]`

`load_ascii` for Milena value types.

**9.88.2.5** `template<typename I> void mln::io::pnm::load_raw_2d ( std::ifstream & file, I & ima ) [inline]`

`load_raw_2d`.

for all pnm 8/16 bits formats

Referenced by `load()`.

**9.88.2.6** `template<typename V> unsigned int mln::io::pnm::max_component ( const V & ) [inline]`

Give the maximum value which can be stored as a component value type V.

Referenced by `load()`.

**9.88.2.7** `template<typename I> void mln::io::pnm::save ( char type, const Image< I > & ima_, const std::string & filename ) [inline]`

Save a Milena image as a pnm image.

#### Parameters

[in] *type* The type of the image to save (can be PPM, PGM, PBM).

[in] *ima\_* The image to save.

[in, out] *filename* the destination.

## 9.89 mln::io::pnm::impl Namespace Reference

Namespace of pnm's implementation details.

### 9.89.1 Detailed Description

Namespace of pnm's implementation details.

## 9.90 mln::io::pnms Namespace Reference

Namespace of pnms input/output handling.

### Functions

- `template<typename V > void load (char type, image3d< V > &ima, const util::array< std::string > &filenames)`  
*Load pnm images as slices of a 3D Milena image.*
- `void load (char type, image3d< bool > &ima, const util::array< std::string > &filenames)`

### 9.90.1 Detailed Description

Namespace of pnms input/output handling.

### 9.90.2 Function Documentation

#### 9.90.2.1 `template<typename V > void mln::io::pnms::load ( char type, image3d< V > & ima, const util::array< std::string > & filenames ) [inline]`

Load pnm images as slices of a 3D Milena image.

#### Parameters

- [in] *type* The type of the pnm files.
- [out] *ima* A reference to the 3D image which will receive data.
- [in] *filenames* The list of 2D images to load..

References `mln::make::image3d()`, `mln::util::array< T >::is_empty()`, and `mln::util::array< T >::nelements()`.

Referenced by `load()`.

#### 9.90.2.2 `void mln::io::pnms::load ( char type, image3d< bool > & ima, const util::array< std::string > & filenames ) [inline]`

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

References `mln::make::image3d()`, `mln::util::array< T >::is_empty()`, `load()`, and `mln::util::array< T >::nelements()`.

## 9.91 mln::io::ppm Namespace Reference

Namespace of ppm input/output handling.

### Functions

- `template<typename I >`  
`void load (Image< I > &ima, const std::string &filename)`  
*Load a ppm image in a Milena image.*
- `template<typename V >`  
`image2d< V > load (const std::string &filename)`  
*Load a ppm image in a Milena image.*
- `template<typename I >`  
`void save (const Image< I > &ima, const std::string &filename)`

### 9.91.1 Detailed Description

Namespace of ppm input/output handling.

### 9.91.2 Function Documentation

#### 9.91.2.1 `template<typename I > void mln::io::ppm::load ( Image< I > & ima, const std::string & filename ) [inline]`

Load a ppm image in a Milena image.

#### Parameters

- [out] *ima* A reference to the image which will receive data.  
 [in] *filename* The source.

#### 9.91.2.2 `template<typename V > image2d< V > mln::io::ppm::load ( const std::string & filename ) [inline]`

Load a ppm image in a Milena image.

To use this routine, you should specialize the template with the value type of the image loaded. (ex : `load<value::int_u8>("...")`)

#### Parameters

- [in] *filename* The image source.

#### Returns

An `image2d` which contains loaded data.

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

Save a Milena image as a ppm image.

#### Parameters

- [in] *ima* The image to save.
- [in, out] *filename* the destination.

Referenced by `mln::registration::icp()`.

## 9.92 mln::io::ppms Namespace Reference

Namespace of ppms input/output handling.

### Functions

- `template<typename V > void load (image3d< V > &ima, const util::array< std::string > &filenames)`  
*Load ppm images as slices of a 3D Milena image.*

### 9.92.1 Detailed Description

Namespace of ppms input/output handling.

### 9.92.2 Function Documentation

**9.92.2.1** `template<typename V > void mln::io::ppms::load ( image3d< V > & ima, const util::array< std::string > & filenames ) [inline]`

Load ppm images as slices of a 3D Milena image.

#### Parameters

- [out] *ima* A reference to the 3D image which will receive data.
- [in] *filenames* The list of 2D images to load..

## 9.93 mln::io::raw Namespace Reference

Namespace of raw input/output handling.

### Classes

- struct `raw_header`  
*Store raw file header.*



## Functions

- `raw_header get_header` (const std::string &filename)  
*Retrieve header in a raw file.*
- `template<typename I > void load` (Image< I > &ima\_, const std::string &filename)  
*Load an image saved as a raw data file.*
- `template<typename I > void save` (const Image< I > &ima\_, const std::string &filename)  
*Save a Milena image as a raw data file.*

### 9.93.1 Detailed Description

Namespace of raw input/output handling.

### 9.93.2 Function Documentation

#### 9.93.2.1 raw\_header mln::io::raw::get\_header ( const std::string & filename )

Retrieve header in a raw file.

References mln::util::array< T >::resize().

#### 9.93.2.2 template<typename I > void mln::io::raw::load ( Image< I > & ima\_ , const std::string & filename )

Load an image saved as a raw data file.

##### Parameters

[in, out] *ima\_* The image to load.

[in] *filename* the destination.

This routine try to read two input files: 'filename' and 'filename.info'. 'filename' is the raw data. 'filename.info' store various information about the image.

#### 9.93.2.3 template<typename I > void mln::io::raw::save ( const Image< I > & ima\_ , const std::string & filename )

Save a Milena image as a raw data file.

##### Parameters

[in] *ima\_* The image to save.

[in] *filename* the destination.

This routine produce two output files: 'filename' and 'filename.info'. 'filename' is the raw data. 'filename.info' store various information about the image.

## 9.94 mln::io::tiff Namespace Reference

Namespace of tiff input/output handling.

### Functions

- `template<typename I > void load (Image< I > &ima_, const std::string &filename)`  
*Load a TIFF image to a Milena image.*

### 9.94.1 Detailed Description

Namespace of tiff input/output handling.

### 9.94.2 Function Documentation

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

Load a TIFF image to a Milena image.

## 9.95 mln::io::txt Namespace Reference

Namespace of txt input/output handling.

### Functions

- `void save (const image2d< char > &ima, const std::string &filename)`  
*Save an image as txt file.*

### 9.95.1 Detailed Description

Namespace of txt input/output handling.

### 9.95.2 Function Documentation

- 9.95.2.1** `void mln::io::txt::save ( const image2d< char > & ima, const std::string & filename ) [inline]`

Save an image as txt file.

### Parameters

- [in] *ima* The image to save. Must be an image of char.

[in] *filename* the destination.

References `mln::image2d< T >::domain()`.

## 9.96 mln::labeling Namespace Reference

Namespace of labeling routines.

### Namespaces

- namespace `impl`  
*Implementation namespace of labeling namespace.*

### Functions

- `template<typename I , typename N , typename L >`  
`mln::trait::ch_value< I, L >::ret background` (const `Image< I >` &input, const `Neighborhood< N >` &nbh, L &nlabels)
- `template<typename I , typename N , typename L >`  
`mln::trait::ch_value< I, L >::ret blobs` (const `Image< I >` &input, const `Neighborhood< N >` &nbh, L &nlabels)  
*Connected component labeling of the binary objects of a binary image.*
- `template<typename I , typename N , typename L , typename A >`  
`util::couple< mln::trait::ch_value< I, L >::ret, util::couple< util::array< typename A::result >, util::array< A > > >` `blobs_and_compute` (const `Image< I >` &input, const `Neighborhood< N >` &nbh, L &nlabels, const `Accumulator< A >` &accu)
- `template<typename V , typename L >`  
`mln::trait::ch_value< L, V >::ret colorize` (const V &value, const `Image< L >` &labeled\_image, const typename L::value &nlabels)  
*Create a new color image from a labeled image and fill each component with a random color.*
- `template<typename V , typename L >`  
`mln::trait::ch_value< L, V >::ret colorize` (const V &value, const `Image< L >` &labeled\_image)
- `template<typename L >`  
`mln::trait::ch_value< L, mln::value::rgb8 >::ret colorize` (const `Image< L >` &input, const typename L::value &nlabels)
- `template<typename A , typename I , typename L >`  
`util::array< mln_meta_accu_result(A, typename I::value)> compute` (const `Meta_Accumulator< A >` &a, const `Image< I >` &input, const `Image< L >` &label, const typename L::value &nlabels)  
*Compute an accumulator onto the pixel values of the image input.*
- `template<typename A , typename L >`  
`util::array< typename A::result > compute` (const `Accumulator< A >` &a, const `Image< L >` &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< mln_meta_accu_result(A, typename L::psite)> compute (const Meta_Accumulator< A`  
`> &a, const Image< L > &label, const typename L::value &nlabels)`  
*Compute an accumulator onto the pixel sites of each component domain of label.*
- `template<typename A , typename I , typename L >`  
`util::array< typename A::result > compute (util::array< A > &a, const Image< I > &input, const`  
`Image< L > &label, const typename L::value &nlabels)`  
*Compute an accumulator onto the pixel values of the image input.*
- `template<typename A , typename I , typename L >`  
`util::array< typename A::result > compute (const Accumulator< A > &a, const Image< I > &in-`  
`put, const Image< L > &label, const typename L::value &nlabels)`  
*Compute an accumulator onto the pixel values of the image input.*
- `template<typename A , typename I , typename L >`  
`mln::trait::ch_value< L, typename A::result >::ret compute_image (const util::array< typename`  
`A::result > &a, const Image< I > &input, const Image< L > &labels, const typename L::value`  
`&nlabels)`  
*Compute an accumulator onto the pixel values of the image input.*
- `template<typename A , typename I , typename L >`  
`mln::trait::ch_value< L, typename A::result >::ret compute_image (const Accumulator< A >`  
`&accu, const Image< I > &input, const Image< L > &labels, const typename L::value &nlabels)`  
*Compute an accumulator onto the pixel values of the image input.*
- `template<typename A , typename I , typename L >`  
`mln::trait::ch_value< L, typename mln::internal::meta_accu_ret_result_helper< A, typename`  
`I::value >::result >::ret compute_image (const Meta_Accumulator< A > &accu, const Image<`  
`I > &input, const Image< L > &labels, const typename L::value &nlabels)`  
*Compute an accumulator onto the pixel values of the image input.*
- `template<typename I , typename N , typename L >`  
`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,`  
`fun::i2v::array< typename I::value > &repack_fun)`  
*Relabel a labeled image in order to have a contiguous labeling.*
- `template<typename I >`  
`mln::trait::concrete< I >::ret pack (const Image< I > &label, typename I::value &new_nlabels)`

- `template<typename I >`  
`void pack\_inplace (Image< I > &label, typename I::value &new\_nlabels)`
- `template<typename I >`  
`void pack\_inplace (Image< I > &label, typename I::value &new\_nlabels, fun::i2v::array< type-  
name I::value > &repack\_fun)`  
*Relabel inplace a labeled image in order to have a contiguous labeling.*
- `template<typename I , typename N , typename L >`  
`mln::trait::ch_value< I, L >::ret regional\_maxima (const Image< I > &input, const Neighborhood<  
N > &nbh, L &nlabels)`
- `template<typename I , typename N , typename L >`  
`mln::trait::ch_value< I, L >::ret regional\_minima (const Image< I > &input, const Neighborhood<  
N > &nbh, L &nlabels)`
- `template<typename I , typename F >`  
`mln::trait::concrete< I >::ret relabel (const Image< I > &label, const typename I::value &nlabels,  
typename I::value &new\_nlabels, const Function\_v2b< F > &fv2b)`  
*Remove components and relabel a labeled image.*
- `template<typename I , typename F >`  
`mln::trait::concrete< I >::ret relabel (const Image< I > &label, const typename I::value &nlabels,  
const Function\_v2v< F > &fv2v)`  
*Remove components and relabel a labeled image.*
- `template<typename I , typename F >`  
`void relabel\_inplace (Image< I > &label, const typename I::value &nlabels, const Function\_v2v<  
F > &fv2v)`  
*Remove components and relabel a labeled image inplace.*
- `template<typename I , typename F >`  
`void relabel\_inplace (Image< I > &label, const typename I::value &nlabels, const Function\_v2b<  
F > &fv2b)`  
*Remove components and relabel a labeled image inplace.*
- `template<typename I , typename J >`  
`mln::trait::concrete< I >::ret superpose (const Image< I > &lhs, const typename I::value &lhs\_-  
nlabels, const Image< J > &rhs, const typename J::value &rhs\_nlabels, typename I::value &new\_-  
nlabels)`  
*Superpose two labeled image.*
- `template<typename I , typename N , typename L >`  
`mln::trait::ch_value< I, L >::ret value (const Image< I > &input, const typename I::value &val,  
const Neighborhood< N > &nbh, L &nlabels)`  
*Connected component labeling of the image sites at a given value.*
- `template<typename I , typename N , typename L , typename A >`  
`util::couple< mln::trait::ch_value< I, L >::ret, util::couple< util::array< typename A::result >,  
util::array< A > > > value\_and\_compute (const Image< I > &input, const typename I::value  
&val, const Neighborhood< N > &nbh, L &nlabels, const Accumulator< A > &accu)`  
*Connected component labeling of the image sites at a given value.*
- `template<typename V , typename I >`  
`mln::trait::ch_value< I, V >::ret wrap (const V &value\_type, const Image< I > &input)`

Wrap labels such as 0 -> 0 and [1, lmax] maps to [1, Lmax] (using modulus).

- `template<typename I >`  
`mln::trait::ch_value< I, mln::value::label_8 >::ret wrap (const Image< I > &input)`  
 Wrap labels such as 0 -> 0 and [1, lmax] maps to [1, Lmax] (using modulus).

## 9.96.1 Detailed Description

Namespace of labeling routines.

## 9.96.2 Function Documentation

### 9.96.2.1 `template<typename I, typename N, typename L > mln::trait::ch_value< I, L >::ret mln::labeling::background ( const Image< I > & input, const Neighborhood< N > & nbh, L & nlabels ) [inline]`

Connected component labeling of the background part in a binary image.

#### Parameters

- [in] *input* The input image.
- [in] *nbh* The connexity of the background.
- [out] *nlabels* The number of labels.

#### Returns

The label image.

#### Precondition

The input image has to be binary (checked at compile-time).

This routine actually calls `mln::labeling::value` with the value set to `false`.

#### See also

[mln::labeling::value](#)

References `value()`.

Referenced by `fill_holes()`.

### 9.96.2.2 `template<typename I, typename N, typename L > mln::trait::ch_value< I, L >::ret mln::labeling::blobs ( const Image< I > & input, const Neighborhood< N > & nbh, L & nlabels ) [inline]`

Connected component labeling of the binary objects of a binary image.

#### Parameters

- [in] *input* The input image.

[in] *nbh* The connexity of the objects.

[out] *nlabels* The Number of labels. Its value is set in the algorithms.

### Returns

The label image.

### Precondition

The input image has to be binary (checked at compile-time).

A fast queue is used so that the algorithm is not recursive and can handle large binary objects (blobs).

Referenced by `blobs_and_compute()`, and `mln::graph::labeling()`.

**9.96.2.3** `template<typename I , typename N , typename L , typename A > util::couple< mln::trait::ch_value< I, L >::ret, util::couple< util::array< typename A::result >, util::array< A > > > mln::labeling::blobs_and_compute ( const Image< I > & input, const Neighborhood< N > & nbh, L & nlabels, const Accumulator< A > & accu )`

Label an image and compute given accumulators.

### Parameters

[in] *input* A binary image.

[in] *nbh* A neighborhood used for labeling.

[in, out] *nlabels* The number of labels found.

[in] *accu* An accumulator to be computed while labeling.

### Returns

The labeled image, computed attributes for each regions and an array of the accumulators used to compute the attributes.

References `blobs()`, and `mln::make::couple()`.

**9.96.2.4** `template<typename V , typename L > mln::trait::ch_value< L, V >::ret mln::labeling::colorize ( const V & value, const Image< L > & labeled_image, const typename L::value & nlabels ) [inline]`

Create a new color image from a labeled image and fill each component with a random color.

`litera::black` is used for component 0, e.g. the background. Min and max values for RGB values can be set through the global variables `mln::labeling::colorize_::min_value` and `mln::labeling::colorize_::max_value`.

### Parameters

[in] *value* value type used in the returned image.

[in] *labeled\_image* A labeled image (

### See also

[labeling::blobs](#)).

**Parameters**

[in] *nlabels* Number of labels.

References `mln::literal::black`, and `mln::data::transform()`.

Referenced by `colorize()`.

**9.96.2.5** `template<typename V , typename L > mln::trait::ch_value< L, V >::ret  
mln::labeling::colorize ( const V & value, const Image< L > & labeled_image )  
[inline]`

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

References `colorize()`, and `compute()`.

**9.96.2.6** `template<typename L > mln::trait::ch_value< L, mln::value::rgb8 >::ret  
mln::labeling::colorize ( const Image< L > & input, const typename L::value & nlabels  
) [inline]`

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

References `colorize()`.

**9.96.2.7** `template<typename A , typename I , typename L > util::array<  
mln_meta_accu_result(A, typename I::value)> mln::labeling::compute ( const  
Meta_Accumulator< A > & a, const Image< I > & input, const Image< L > & label,  
const typename L::value & nlabels ) [inline]`

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

for each component of the image `label`.

**Parameters**

[in] *a* A meta-accumulator.

[in] *input* The input image.

[in] *label* The labeled image.

[in] *nlabels* The number of labels in `label`.

**Returns**

A `util::array` of accumulator result (one result per label).

References `compute()`.

**9.96.2.8** `template<typename A , typename L > util::array< typename A::result >  
mln::labeling::compute ( const Accumulator< A > & a, const Image< L > & label,  
const typename L::value & nlabels ) [inline]`

Compute an accumulator onto the pixel sites of each component domain of `label`.



**Parameters**

- [in] *a* An accumulator.
- [in] *label* The labeled image.
- [in] *nlabels* The number of labels in *label*.

**Returns**

A [util::array](#) of accumulator result (one result per label).

**9.96.2.9** `template<typename A , typename L > util::array< mln_meta_accu_result(A, typename L::psite)> mln::labeling::compute ( const Meta_Accumulator< A > & a, const Image< L > & label, const typename L::value & nlabels ) [inline]`

Compute an accumulator onto the pixel sites of each component domain of *label*.

**Parameters**

- [in] *a* A meta-accumulator.
- [in] *label* The labeled image.
- [in] *nlabels* The number of labels in *label*.

**Returns**

A [util::array](#) of accumulator result (one result per label).

References `compute()`.

**9.96.2.10** `template<typename A , typename I , typename L > util::array< typename A::result > mln::labeling::compute ( util::array< A > & a, const Image< I > & input, const Image< L > & label, const typename L::value & nlabels ) [inline]`

Compute an accumulator onto the pixel values of the image *input*.  
for each component of the image *label*.

**Parameters**

- [in] *a* An array of accumulator.
- [in] *input* The input image.
- [in] *label* The labeled image.
- [in] *nlabels* The number of labels in *label*.

**Returns**

A [util::array](#) of accumulator result (one result per label).

**9.96.2.11** `template<typename A , typename I , typename L > util::array< typename A::result > mln::labeling::compute ( const Accumulator< A > & a, const Image< I > & input, const Image< L > & label, const typename L::value & nlabels ) [inline]`

Compute an accumulator onto the pixel values of the image `input`.  
for each component of the image `label`.

#### Parameters

- [in] *a* An accumulator.
- [in] *input* The input image.
- [in] *label* The labeled image.
- [in] *nlabels* The number of labels in `label`.

#### Returns

A `util::array` of accumulator result (one result per label).

Referenced by `colorize()`, `compute()`, `compute_image()`, `fill_holes()`, `mln::make::p_edges_with_mass_centers()`, `mln::make::p_vertices_with_mass_centers()`, `pack()`, and `pack_inplace()`.

**9.96.2.12** `template<typename A , typename I , typename L > mln::trait::ch_value< L , typename A::result >::ret mln::labeling::compute_image ( const util::array< typename A::result > & a, const Image< I > & input, const Image< L > & labels, const typename L::value & nlabels )`

Compute an accumulator onto the pixel values of the image `input`.  
for each component of the image `label`.

#### Parameters

- [in] *a* The `mln::p_array` of accumulator result.
- [in] *input* The input image (values).
- [in] *labels* The label image.
- [in] *nlabels* The count of labels.

#### Returns

The image where labels are replaced by the result of the accumulator.

Referenced by `compute_image()`.

**9.96.2.13** `template<typename A , typename I , typename L > mln::trait::ch_value< L , typename A::result >::ret mln::labeling::compute_image ( const Accumulator< A > & accu, const Image< I > & input, const Image< L > & labels, const typename L::value & nlabels ) [inline]`

Compute an accumulator onto the pixel values of the image `input`.  
for each component of the image `label`.

**Parameters**

- [in] *accu* The accumulator.
- [in] *input* The input image (values).
- [in] *labels* The label image.
- [in] *nlabels* The count of labels.

**Returns**

The image where labels are replaced by the result of the accumulator.

References `compute()`, and `compute_image()`.

**9.96.2.14** `template<typename A , typename I , typename L > mln::trait::ch_value< L, typename mln::internal::meta_accu_ret_result_helper< A, typename I::value >::result >::ret mln::labeling::compute_image ( const Meta_Accumulator< A > & accu, const Image< I > & input, const Image< L > & labels, const typename L::value & nlabels ) [inline]`

Compute an accumulator onto the pixel values of the image `input`.  
for each component of the image `label`.

**Parameters**

- [in] *accu* The meta-accumulator.
- [in] *input* The input image (values).
- [in] *labels* The label image.
- [in] *nlabels* The count of labels.

**Returns**

The image where labels are replaced by the result of the accumulator.

References `compute()`, and `compute_image()`.

**9.96.2.15** `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**

- [in] *input* The input image.
- [in] *nbh* The connexity of the background.
- [out] *nlabels* The number of labels.

**Returns**

The binary image with a simple object without holes.

**Precondition**

The input image has to be binary (checked at compile-time).

This routine actually calls [mln::labeling::background](#)

#### See also

[mln::labeling::background](#)

References [background\(\)](#), [compute\(\)](#), [mln::data::fill\(\)](#), [mln::initialize\(\)](#), and [mln::util::array< T >::nelements\(\)](#).

**9.96.2.16** `template<typename I , typename N , typename L > mln::trait::ch_value< I, L >::ret mln::labeling::flat_zones ( const Image< I > & input, const Neighborhood< N > & nbh, L & nlabels )`

Connected component labeling of the flat zones of an image.

#### Parameters

- [in] *input* The input image.
- [in] *nbh* The connexity of the flat zones.
- [out] *nlabels* The number of labels.

#### Returns

The label image.

**9.96.2.17** `template<typename I , typename N , typename L > mln::trait::ch_value< I, L >::ret mln::labeling::foreground ( const Image< I > & input, const Neighborhood< N > & nbh, L & nlabels ) [inline]`

Connected component labeling of the object part in a binary image.

#### Parameters

- [in] *input* The input image.
- [in] *nbh* The connexity of the foreground.
- [out] *nlabels* The number of labels.

#### Returns

The label image.

#### Precondition

The input image has to be binary (checked at compile-time).

This routine actually calls [mln::labeling::value](#) with the value set to `true`.

#### See also

[mln::labeling::value](#)

References [value\(\)](#).

**9.96.2.18** `template<typename I > mln::trait::concrete< I >::ret mln::labeling::pack ( const Image< I > & label, typename I::value & new_nlabels, fun::i2v::array< typename I::value > & repack_fun )`

Relabel a labeled image in order to have a contiguous labeling.

#### Parameters

- [in] *label* The labeled image.
- [out] *new\_nlabels* The number of labels after relabeling.
- [out] *repack\_fun* The function used to repack the labels.

#### Returns

The relabeled image.

References `compute()`, `mln::make::relabelfun()`, and `mln::data::transform()`.

Referenced by `pack()`.

**9.96.2.19** `template<typename I > mln::trait::concrete< I >::ret mln::labeling::pack ( const Image< I > & label, typename I::value & new_nlabels )`

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

References `pack()`.

**9.96.2.20** `template<typename I > void mln::labeling::pack_inplace ( Image< I > & label, typename I::value & new_nlabels )`

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

References `pack_inplace()`.

**9.96.2.21** `template<typename I > void mln::labeling::pack_inplace ( Image< I > & label, typename I::value & new_nlabels, fun::i2v::array< typename I::value > & repack_fun )`

Relabel inplace a labeled image in order to have a contiguous labeling.

#### Parameters

- [in] *label* The labeled image.
- [out] *new\_nlabels* The number of labels after relabeling.
- [out] *repack\_fun* The function used to repack the labels.

References `compute()`, `mln::make::relabelfun()`, and `mln::data::transform()`.

Referenced by `pack_inplace()`.

**9.96.2.22** `template<typename I , typename N , typename L > mln::trait::ch_value< I, L >::ret mln::labeling::regional_maxima ( const Image< I > & input, const Neighborhood< N > & nbh, L & nlabels )`

Connected component labeling of the regional maxima of an image.

#### Parameters

- [in] *input* The input image.
- [in] *nbh* The connexity of the regional maxima.
- [out] *nlabels* The number of labeled regions.

#### Returns

The label image.

**9.96.2.23** `template<typename I , typename N , typename L > mln::trait::ch_value< I, L >::ret mln::labeling::regional_minima ( const Image< I > & input, const Neighborhood< N > & nbh, L & nlabels )`

Connected component labeling of the regional minima of an image.

#### Parameters

- [in] *input* The input image.
- [in] *nbh* The connexity of the regional minima.
- [out] *nlabels* The number of labeled regions.

#### Returns

The label image.

Referenced by `mln::morpho::meyer_wst()`.

**9.96.2.24** `template<typename I , typename F > mln::trait::concrete< I >::ret mln::labeling::relabel ( const Image< I > & label, const typename I::value & nlabels, typename I::value & new_nlabels, const Function_v2b< F > & fv2b ) [inline]`

Remove components and relabel a labeled image.

#### Parameters

- [in] *label* the labeled image.
- [in] *nlabels* the number of labels in *label*.
- [out] *new\_nlabels* the number of labels after relabeling.
- [in] *fv2b* function returning whether a label must be replaced by the background.

#### Returns

the relabeled image.

References `mln::make::relabelfun()`.

Referenced by `superpose()`.

**9.96.2.25** `template<typename I , typename F > mln::trait::concrete< I >::ret  
mln::labeling::relabel ( const Image< I > & label, const typename I::value & nlabels,  
const Function_v2v< F > & fv2v ) [inline]`

Remove components and relabel a labeled image.

#### Parameters

- [in] *label* the labeled image.
- [in] *nlabels* the number of labels in *label*.
- [in] *fv2v* function returning the new component id for each pixel value.

#### Returns

the relabeled image.

References `mln::data::transform()`.

**9.96.2.26** `template<typename I , typename F > void mln::labeling::relabel_inplace ( Image< I  
> & label, const typename I::value & nlabels, const Function_v2v< F > & fv2v )  
[inline]`

Remove components and relabel a labeled image inplace.

#### Parameters

- [in, out] *label* the labeled image.
- [in] *nlabels* the number of labels in *label*.
- [in] *fv2v* function returning the new component id for each pixel value.

References `mln::data::transform_inplace()`.

**9.96.2.27** `template<typename I , typename F > void mln::labeling::relabel_inplace ( Image< I >  
& label, const typename I::value & nlabels, const Function_v2b< F > & fv2b )  
[inline]`

Remove components and relabel a labeled image inplace.

#### Parameters

- [in, out] *label* the labeled image.
- [in] *nlabels* the number of labels in *label*.
- [in] *fv2b* function returning whether a label must be replaced by the background.

References `mln::make::relabelfun()`.

Referenced by `mln::labeled_image_base< I, E >::relabel()`.

**9.96.2.28** `template<typename I , typename J > mln::trait::concrete< I >::ret  
mln::labeling::superpose ( const Image< I > & lhs, const typename I::value &  
lhs_nlabels, const Image< J > & rhs, const typename J::value & rhs_nlabels,  
typename I::value & new_nlabels )`

Superpose two labeled image.

Labels in `lhs` are preserved in the output. Labels of `rhs` are renumbered from the last label value of `lhs`. It avoids duplicate label values in several components.

#### Parameters

- [in] *lhs* A labeled image.
- [in] *lhs\_nlabels* The number of labels in `lhs`.
- [in] *rhs* A labeled image.
- [in] *rhs\_nlabels* The number of labels in `rhs`.
- [out] *new\_nlabels* The number of labels in the output image.

#### Returns

An image with all the components of `rhs` and `lhs`.

#### Precondition

- `rhs` and `lhs` must have the same domain.
- The value type of `rhs` must be convertible towards `lhs`'s.

References `mln::duplicate()`, `mln::value::equiv()`, `mln::data::paste()`, `relabel()`, and `mln::literal::zero`.

**9.96.2.29** `template<typename I , typename N , typename L > mln::trait::ch_value< I, L >::ret  
mln::labeling::value ( const Image< I > & input, const typename I::value & val, const  
Neighborhood< N > & nbh, L & nlabels )`

Connected component labeling of the image sites at a given value.

#### Parameters

- [in] *input* The input image.
- [in] *val* The value to consider.
- [in] *nbh* The connectivity of components.
- [out] *nlabels* The number of labels.

#### Returns

The label image.

Referenced by `background()`, and `foreground()`.

**9.96.2.30** `template<typename I , typename N , typename L , typename A > util::couple<  
mln::trait::ch_value< I, L >::ret, util::couple< util::array< typename A::result >,  
util::array< A > > > mln::labeling::value_and_compute ( const Image< I > & input,  
const typename I::value & val, const Neighborhood< N > & nbh, L & nlabels, const  
Accumulator< A > & accu )`

Connected component labeling of the image sites at a given value.



**Parameters**

- [in] *input* The input image.
- [in] *val* The value to consider.
- [in] *nbh* The connectivity of components.
- [out] *nlabels* The number of labels.

**Returns**

The label image.

References mln::make::couple().

**9.96.2.31** `template<typename V , typename I > mln::trait::ch_value< I, V >::ret  
mln::labeling::wrap ( const V & value_type, const Image< I > & input ) [inline]`

Wrap labels such as 0 -> 0 and [1, lmax] maps to [1, Lmax] (using modulus).

**Parameters**

- [in] *value\_type* The type used to wrap the label type.
- [in] *input* The label image.

**Returns**

A new image with values wrapped with type V.

References mln::data::transform().

Referenced by wrap().

**9.96.2.32** `template<typename I > mln::trait::ch_value< I, mln::value::label_8 >::ret  
mln::labeling::wrap ( const Image< I > & input ) [inline]`

Wrap labels such as 0 -> 0 and [1, lmax] maps to [1, Lmax] (using modulus).

Use label\_8 as label type.

**Parameters**

- [in] *input* The label image.

**Returns**

A new image with values wrapped with type label\_8.

References wrap().

## 9.97 mln::labeling::impl Namespace Reference

Implementation namespace of labeling namespace.

## Namespaces

- namespace [generic](#)  
*Generic implementation namespace of labeling namespace.*

## Functions

- `template<typename A, typename I, typename L > util::array< typename A::result > compute_fastest (const Accumulator< A > &a_, const Image< I > &input_, const Image< L > &label_, const typename L::value &nlabels)`  
*Fastest implementation of [labeling::compute](#).*
- `template<typename A, typename I, typename L > util::array< typename A::result > compute_fastest (util::array< A > &accus, const Image< I > &input_, const Image< L > &label_, const typename L::value &nlabels)`  
*Fastest implementation of [labeling::compute](#).*

### 9.97.1 Detailed Description

Implementation namespace of labeling namespace.

### 9.97.2 Function Documentation

**9.97.2.1** `template<typename A, typename I, typename L > util::array<typename A ::result> mln::labeling::impl::compute_fastest ( const Accumulator< A > & a_, const Image< I > & input_, const Image< L > & label_, const typename L::value & nlabels )` `[inline]`

Fastest implementation of [labeling::compute](#).

#### Parameters

- `[in]` *a\_* An accumulator.
- `[in]` *input\_* The input image.
- `[in]` *label\_* The labeled image.
- `[in]` *nlabels* The number of labels in label.

#### Returns

A `util::array` of accumulator result (one result per label).

References `mln::geom::ncols()`.

**9.97.2.2** `template<typename A, typename I, typename L > util::array<typename A ::result> mln::labeling::impl::compute_fastest ( util::array< A > & accus, const Image< I > & input_, const Image< L > & label_, const typename L::value & nlabels )` `[inline]`

Fastest implementation of [labeling::compute](#).

**Parameters**

- [in] *accus* An array of accumulators.
- [in] *input\_* The input image.
- [in] *label\_* The labeled image.
- [in] *nlabels* The number of labels in *label\_*.

**Returns**

A `util::array` of accumulator result (one result per label).

References `mln::geom::ncols()`, `mln::util::array< T >::resize()`, and `mln::util::array< T >::size()`.

## 9.98 mln::labeling::impl::generic Namespace Reference

Generic implementation namespace of labeling namespace.

**Functions**

- `template<typename A , typename L >`  
`util::array< typename A::result > compute (const Accumulator< A > &a_, const Image< L > &label_, const typename L::value &nlabels)`  
*Generic implementation of `labeling::compute`.*
- `template<typename A , typename L >`  
`util::array< typename A::result > compute (util::array< A > &accus, const Image< L > &label_, const typename L::value &nlabels)`  
*Generic implementation of `labeling::compute`.*
- `template<typename A , typename I , typename L >`  
`util::array< typename A::result > compute (util::array< A > &accus, const Image< I > &input_, const Image< L > &label_, const typename L::value &nlabels)`  
*Generic implementation of `labeling::compute`.*
- `template<typename A , typename I , typename L >`  
`util::array< typename A::result > compute (const Accumulator< A > &a_, const Image< I > &input_, const Image< L > &label_, const typename L::value &nlabels)`  
*Generic implementation of `labeling::compute`.*

### 9.98.1 Detailed Description

Generic implementation namespace of labeling namespace.

### 9.98.2 Function Documentation

- 9.98.2.1** `template<typename A , typename L > util::array<typename A ::result>`  
`mln::labeling::impl::generic::compute ( const Accumulator< A > & a_ , const Image<`  
`L > & label_ , const typename L::value & nlabels ) [inline]`

Generic implementation of `labeling::compute`.

**Parameters**

- [in] *a\_* An accumulator.
- [in] *label\_* The labeled image.
- [in] *nlabels* The number of labels in *label*.

**Returns**

A [util::array](#) of accumulator result (one result per label).

```
9.98.2.2 template<typename A , typename L > util::array<typename A ::result>
mln::labeling::impl::generic::compute ( util::array< A > & accus, const Image< L >
& label_, const typename L::value & nlabels ) [inline]
```

Generic implementation of [labeling::compute](#).

**Parameters**

- [in] *accus\_* An array of accumulators. If the size is set to *nlabels* + 1, the accumulators are considered as initialized. Otherwise, the size is adjusted.
- [in] *label\_* The labeled image.
- [in] *nlabels* The number of labels in *label*.

**Returns**

A [util::array](#) of accumulator result (one result per label).

References [mln::util::array< T >::resize\(\)](#), and [mln::util::array< T >::size\(\)](#).

```
9.98.2.3 template<typename A , typename I , typename L > util::array<typename A ::result>
mln::labeling::impl::generic::compute ( util::array< A > & accus, const Image< I > &
input_, const Image< L > & label_, const typename L::value & nlabels ) [inline]
```

Generic implementation of [labeling::compute](#).

**Parameters**

- [in] *accus* An array of accumulators.
- [in] *input\_* The input image.
- [in] *label\_* The labeled image.
- [in] *nlabels* The number of labels in *label*.

**Returns**

A [util::array](#) of accumulator result (one result per label).

References [mln::util::array< T >::resize\(\)](#), and [mln::util::array< T >::size\(\)](#).

**9.98.2.4** `template<typename A , typename I , typename L > util::array<typename A ::result> mln::labeling::impl::generic::compute ( const Accumulator< A > & a_ , const Image< I > & input_ , const Image< L > & label_ , const typename L::value & nlabels )`  
**[inline]**

Generic implementation of [labeling::compute](#).

#### Parameters

- [in] *a\_* An accumulator.
- [in] *input\_* The input image.
- [in] *label\_* The labeled image.
- [in] *nlabels* The number of labels in label.

#### Returns

A [util::array](#) of accumulator result (one result per label).

## 9.99 mln::linear Namespace Reference

Namespace of linear image processing routines.

### Namespaces

- namespace [impl](#)  
*Namespace of linear image processing routines implementation details.*
- namespace [local](#)  
*Specializations of local linear routines.*

### Functions

- `template<typename I > mln::trait::concrete< I >::ret gaussian (const Image< I > &input, float sigma)`  
*Gaussian filter of an image input.*
- `template<typename I > mln::trait::concrete< I >::ret gaussian (const Image< I > &input, float sigma, int dir)`
- `template<typename I > mln::trait::concrete< I >::ret gaussian\_1st\_derivative (const Image< I > &input, float sigma)`
- `template<typename I > mln::trait::concrete< I >::ret gaussian\_1st\_derivative (const Image< I > &input, float sigma, int dir)`
- `template<typename I > mln::trait::concrete< I >::ret gaussian\_2nd\_derivative (const Image< I > &input, float sigma)`
- `template<typename I > mln::trait::concrete< I >::ret gaussian\_2nd\_derivative (const Image< I > &input, float sigma, int dir)`

- `template<typename I, typename W, unsigned Sh, unsigned Sv>`  
`mln_ch_convolve` (I, W) `convolve_2x1d`(const `Image`< I > &input)
- `template<typename I, typename W >`  
`mln_ch_convolve` (I, W) `convolve`(const `Image`< I > &input)
- `template<typename I, typename W, unsigned S>`  
`mln_ch_convolve` (I, W) `convolve_directional`(const `Image`< I > &input)
- `template<typename I >`  
`mln_ch_convolve_grad` (I, int) `sobel_2d`(const `Image`< I > &input)  
*Compute the vertical component of the 2D Sobel gradient.*
  
- `template<typename I >`  
`mln_ch_convolve` (I, int) `sobel_2d_h`(const `Image`< I > &input)  
*Sobel\_2d gradient components.*

### 9.99.1 Detailed Description

Namespace of linear image processing routines.

### 9.99.2 Function Documentation

#### 9.99.2.1 `template<typename I > mln::trait::concrete< I >::ret mln::linear::gaussian ( const Image< I > & input, float sigma ) [inline]`

Gaussian filter of an image `input`.

#### Precondition

`output.domain = input.domain`

Apply an approximated gaussian filter of `sigma` on `input`. This filter is applied in all the input image direction.

#### Precondition

`input.is_valid`

References `mln::initialize()`.

Referenced by `mln::subsampling::gaussian_subsampling()`.

#### 9.99.2.2 `template<typename I > mln::trait::concrete< I >::ret mln::linear::gaussian ( const Image< I > & input, float sigma, int dir ) [inline]`

Apply an approximated gaussian filter of `sigma` on `input`. on a specific direction `dir` if `dir = 0`, the filter is applied on the first image dimension. if `dir = 1`, the filter is applied on the second image dimension. And so on...

#### Precondition

`input.is_valid`  
`dir < dimension(input)`

References `mln::initialize()`.

**9.99.2.3** `template<typename I> mln::trait::concrete< I >::ret mln::linear::gaussian_1st_derivative ( const Image< I > & input, float sigma )`  
[inline]

Apply an approximated first derivative gaussian filter of `sigma` on `input`. This filter is applied in all the input image direction.

**Precondition**

`input.is_valid`

References `mln::initialize()`.

**9.99.2.4** `template<typename I> mln::trait::concrete< I >::ret mln::linear::gaussian_1st_derivative ( const Image< I > & input, float sigma, int dir )`  
[inline]

Apply an approximated first derivative gaussian filter of `sigma` on `input`. on a specific direction `dir` if `dir = 0`, the filter is applied on the first image dimension. if `dir = 1`, the filter is applied on the second image dimension. And so on...

**Precondition**

`input.is_valid`  
`dir < dimension(input)`

References `mln::initialize()`.

**9.99.2.5** `template<typename I> mln::trait::concrete< I >::ret mln::linear::gaussian_2nd_derivative ( const Image< I > & input, float sigma )`  
[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()`.

**9.99.2.6** `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()`.

**9.99.2.7** `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()`, `mln_ch_convolve_grad()`, and `mln::data::transform()`.

**9.99.2.8** `template<typename I , typename W , unsigned Sh, unsigned Sv> mln::linear::mln_ch_convolve ( I, W ) const`

Convolution of an image `input` by two weighted line-shapes windows.

**Warning**

The weighted window is used as-is, considering that its symmetrization is handled by the client.

**Precondition**

`input.is_valid`

**9.99.2.9** `template<typename I , typename W > mln::linear::mln_ch_convolve ( I, W ) const`

Convolution of an image `input` by the weighted window `w_win`.

**Warning**

Computation of `output (p)` is performed with the value type of `output`.

The weighted window is used as-is, considering that its symmetrization is handled by the client.

**Precondition**

`input.is_valid`

Referenced by `mln_ch_convolve()`, and `mln_ch_convolve_grad()`.

**9.99.2.10** `template<typename I , typename W , unsigned S> mln::linear::mln_ch_convolve ( I, W ) const [inline]`

Convolution of an image `input` by a line-shaped (directional) weighted window defined by the array of `weights`.

**Warning**

Computation of `output (p)` is performed with the value type of `output`.

The weighted window is used as-is, considering that its symmetrization is handled by the client.

**Precondition**

`input.is_valid`



**9.99.2.11** `template<typename I > mln::linear::mln_ch_convolve_grad ( I, int ) const`

Compute the vertical component of the 2D Sobel gradient.

References `mln_ch_convolve()`, and `mln::data::transform()`.

Referenced by `mln_ch_convolve()`.

**9.100 mln::linear::impl Namespace Reference**

Namespace of linear image processing routines implementation details.

**9.100.1 Detailed Description**

Namespace of linear image processing routines implementation details.

**9.101 mln::linear::local Namespace Reference**

Specializations of local linear routines.

**Namespaces**

- namespace [impl](#)

*Namespace of local linear routines implementation details.*

**Functions**

- `template<typename I , typename P , typename W , typename R >`  
void `convolve` (const `Image< I >` &input, const `Site< P >` &p, const `Weighted_Window< W >` &w\_win, R &result)
- `template<typename P , typename W , typename R >`  
void `convolve` (const `Generalized_Pixel< P >` &p, const `Weighted_Window< W >` &w\_win, R &result)

**9.101.1 Detailed Description**

Specializations of local linear routines.

**9.101.2 Function Documentation****9.101.2.1** `template<typename I , typename P , typename W , typename R > void`  
`mln::linear::local::convolve ( const Image< I > & input, const Site< P > & p, const`  
`Weighted_Window< W > & w_win, R & result ) [inline]`

Local convolution of image `input` at point `p` by the weighted window `w_win`.

**Warning**

Computation of the `result` is performed with the type `R`.  
The weighted window is used as-is, considering that its symmetrization is handled by the client.

Referenced by `convolve()`.

```
9.101.2.2 template<typename P , typename W , typename R > void mln::linear::local::convolve (
    const Generalized_Pixel< P > & p, const Weighted_Window< W > & w_win, R &
    result ) [inline]
```

Local convolution around (generalized) pixel by the weighted window `w_win`.

**Warning**

Computation of the `result` is performed with the type `R`.  
The weighted window is used as-is, considering that its symmetrization is handled by the client.

References `convolve()`.

## 9.102 mln::linear::local::impl Namespace Reference

Namespace of local linear routines implementation details.

### 9.102.1 Detailed Description

Namespace of local linear routines implementation details.

## 9.103 mln::literal Namespace Reference

Namespace of literals.

**Classes**

- struct [black\\_t](#)  
*Type of literal black.*
- struct [blue\\_t](#)  
*Type of literal blue.*
- struct [brown\\_t](#)  
*Type of literal brown.*
- struct [cyan\\_t](#)  
*Type of literal cyan.*
- struct [green\\_t](#)

*Type of literal green.*

- struct [identity\\_t](#)  
*Type of literal identity.*
- struct [light\\_gray\\_t](#)  
*Type of literal grays.*
- struct [lime\\_t](#)  
*Type of literal lime.*
- struct [magenta\\_t](#)  
*Type of literal magenta.*
- struct [max\\_t](#)  
*Type of literal max.*
- struct [min\\_t](#)  
*Type of literal min.*
- struct [olive\\_t](#)  
*Type of literal olive.*
- struct [one\\_t](#)  
*Type of literal one.*
- struct [orange\\_t](#)  
*Type of literal orange.*
- struct [origin\\_t](#)  
*Type of literal origin.*
- struct [pink\\_t](#)  
*Type of literal pink.*
- struct [purple\\_t](#)  
*Type of literal purple.*
- struct [red\\_t](#)  
*Type of literal red.*
- struct [teal\\_t](#)  
*Type of literal teal.*
- struct [violet\\_t](#)  
*Type of literal violet.*
- struct [white\\_t](#)  
*Type of literal white.*

- struct `yellow_t`  
*Type of literal yellow.*
- struct `zero_t`  
*Type of literal zero.*

## Variables

- const `black_t` & `black` = `black_t()`  
*Literal black.*
- const `blue_t` & `blue` = `blue_t()`  
*Literal blue.*
- const `brown_t` & `brown` = `brown_t()`  
*Literal brown.*
- const `cyan_t` & `cyan` = `cyan_t()`  
*Literal cyan.*
- const `dark_gray_t` & `dark_gray` = `dark_gray_t()`  
*Literal dark gray.*
- const `green_t` & `green` = `green_t()`  
*Literal green.*
- const `identity_t` & `identity` = `identity_t()`  
*Literal identity.*
- const `light_gray_t` & `light_gray` = `light_gray_t()`  
*Literal light gray.*
- const `lime_t` & `lime` = `lime_t()`  
*Literal lime.*
- const `magenta_t` & `magenta` = `magenta_t()`  
*Literal magenta.*
- const `max_t` & `max` = `max_t()`  
*Literal max.*
- const `medium_gray_t` & `medium_gray` = `medium_gray_t()`  
*Literal medium gray.*
- const `min_t` & `min` = `min_t()`  
*Literal min.*
- const `olive_t` & `olive` = `olive_t()`

*Literal olive.*

- const [one\\_t](#) & [one](#) = [one\\_t\(\)](#)

*Literal one.*

- const [orange\\_t](#) & [orange](#) = [orange\\_t\(\)](#)

*Literal orange.*

- const [origin\\_t](#) & [origin](#) = [origin\\_t\(\)](#)

*Literal origin.*

- const [pink\\_t](#) & [pink](#) = [pink\\_t\(\)](#)

*Literal pink.*

- const [purple\\_t](#) & [purple](#) = [purple\\_t\(\)](#)

*Literal purple.*

- const [red\\_t](#) & [red](#) = [red\\_t\(\)](#)

*Literal red.*

- const [teal\\_t](#) & [teal](#) = [teal\\_t\(\)](#)

*Literal teal.*

- const [violet\\_t](#) & [violet](#) = [violet\\_t\(\)](#)

*Literal violet.*

- const [white\\_t](#) & [white](#) = [white\\_t\(\)](#)

*Literal white.*

- const [yellow\\_t](#) & [yellow](#) = [yellow\\_t\(\)](#)

*Literal yellow.*

- const [zero\\_t](#) & [zero](#) = [zero\\_t\(\)](#)

*Literal zero.*

### 9.103.1 Detailed Description

Namespace of literals.

### 9.103.2 Variable Documentation

#### 9.103.2.1 const [black\\_t](#) & [mln::literal::black](#) = [black\\_t\(\)](#)

[Literal](#) [black](#).

Referenced by [mln::labeling::colorize\(\)](#), and [mln::registration::icp\(\)](#).

**9.103.2.2** `const blue_t & mln::literal::blue = blue_t()`

[Literal](#) blue.

**9.103.2.3** `const brown_t & mln::literal::brown = brown_t()`

[Literal](#) brown.

**9.103.2.4** `const cyan_t & mln::literal::cyan = cyan_t()`

[Literal](#) cyan.

**9.103.2.5** `const dark_gray_t & mln::literal::dark_gray = dark_gray_t()`

[Literal](#) dark gray.

**9.103.2.6** `const green_t & mln::literal::green = green_t()`

[Literal](#) green.

Referenced by `mln::registration::icp()`, and `mln::make_debug_graph_image()`.

**9.103.2.7** `const identity_t & mln::literal::identity = identity_t()`

[Literal](#) identity.

**9.103.2.8** `const light_gray_t & mln::literal::light_gray = light_gray_t()`

[Literal](#) light gray.

**9.103.2.9** `const lime_t & mln::literal::lime = lime_t()`

[Literal](#) lime.

**9.103.2.10** `const magenta_t & mln::literal::magenta = magenta_t()`

[Literal](#) magenta.

**9.103.2.11** `const max_t & mln::literal::max = max_t()`

[Literal](#) max.

**9.103.2.12** `const medium_gray_t & mln::literal::medium_gray = medium_gray_t()`

[Literal](#) medium\_gray.

**9.103.2.13** `const min_t & mln::literal::min = min_t()`

[Literal](#) min.

**9.103.2.14** `const olive_t & mln::literal::olive = olive_t()`

[Literal](#) olive.

**9.103.2.15** `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--()`.

**9.103.2.16** `const orange_t & mln::literal::orange = orange_t()`

[Literal](#) orange.

**9.103.2.17** `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()`.

**9.103.2.18** `const pink_t & mln::literal::pink = pink_t()`

[Literal](#) pink.

**9.103.2.19** `const purple_t & mln::literal::purple = purple_t()`

[Literal](#) purple.

**9.103.2.20** `const red_t & mln::literal::red = red_t()`

[Literal](#) red.

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

**9.103.2.21** `const teal_t & mln::literal::teal = teal_t()`

[Literal](#) teal.

**9.103.2.22** `const violet_t & mln::literal::violet = violet_t()`

[Literal](#) violet.

**9.103.2.23** `const white_t & mln::literal::white = white_t()`

[Literal](#) white.

Referenced by `mln::registration::icp()`.

**9.103.2.24** `const yellow_t & mln::literal::yellow = yellow_t()`

[Literal](#) yellow.

**9.103.2.25** `const zero_t & mln::literal::zero = zero_t()`

[Literal](#) zero.

Referenced by `mln::transform::influence_zone_geodesic_saturated()`, `mln::accu::shape::volume< I >::init()`, `mln::accu::stat::variance< T, S, R >::init()`, `mln::morpho::attribute::sum< I, S >::init()`, `mln::accu::math::sum< T, S >::init()`, `mln::accu::rms< T, V >::init()`, `mln::accu::convolve< T1, T2, R >::init()`, `mln::accu::center< P, V >::init()`, `mln::window< D >::is_centered()`, `mln::accu::stat::variance< T, S, R >::mean()`, `mln::accu::stat::var< T >::mean()`, `mln::geom::mesh_corner_point_area()`, `mln::geom::mesh_curvature()`, `mln::geom::mesh_normal()`, `mln::morpho::meyer_wst()`, `mln::algebra::operator*`, `mln::test::positive()`, `mln::make::relabelfun()`, `mln::geom::rotate()`, `mln::accu::shape::volume< I >::set_value()`, `mln::morpho::watershed::superpose()`, `mln::labeling::superpose()`, `mln::debug::superpose()`, `mln::accu::stat::var< T >::to_result()`, `mln::geom::translate()`, and `mln::make::w_window_directional()`.

## 9.104 mln::logical Namespace Reference

Namespace of logic.

### Namespaces

- namespace [impl](#)  
*Implementation namespace of logical namespace.*

### Functions

- `template<typename L, typename R >`  
`void and\_inplace (Image< L > &lhs, const Image< R > &rhs)`
- `template<typename L, typename R >`  
`mln::trait::ch_value< L, typename mln::fun::vv2v::land\_not< typename L::value, typename R::value >::result >::ret and\_not (const Image< L > &lhs, const Image< R > &rhs)`
- `template<typename L, typename R >`  
`void and\_not\_inplace (Image< L > &lhs, const Image< R > &rhs)`
- `template<typename I >`  
`void not\_inplace (Image< I > &input)`
- `template<typename L, typename R >`  
`void or\_inplace (Image< L > &lhs, const Image< R > &rhs)`
- `template<typename L, typename R >`  
`void xor\_inplace (Image< L > &lhs, const Image< R > &rhs)`



## 9.104.1 Detailed Description

Namespace of logic.

## 9.104.2 Function Documentation

### 9.104.2.1 `template<typename L , typename R > void mln::logical::and_inplace ( Image< L > & lhs, const Image< R > & rhs ) [inline]`

Point-wise in-place "logical and" of image `rhs` in image `lhs`.

#### Parameters

[in, out] *lhs* First operand image.

[in] *rhs* Second operand image.

It performs:

for all `p` of `rhs.domain`

`lhs(p) = lhs(p) and rhs(p)`

#### Precondition

```
rhs.domain >= lhs.domain
```

References `mln::data::transform_inplace()`.

### 9.104.2.2 `template<typename L , typename R > mln::trait::ch_value< L, typename mln::fun::vv2v::land_not< typename L::value, typename R::value >::result >::ret mln::logical::and_not ( const Image< L > & lhs, const Image< R > & rhs ) [inline]`

Point-wise "logical and-not" between images `lhs` and `rhs`.

#### Parameters

[in] *lhs* First operand image.

[in] *rhs* Second operand image.

#### Returns

The result image.

#### Precondition

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

References `mln::data::transform()`.

**9.104.2.3** `template<typename L , typename R > void mln::logical::and_not_inplace ( Image< L > & lhs, const Image< R > & rhs ) [inline]`

Point-wise in-place "logical and-not" of image *rhs* in image *lhs*.

#### Parameters

[in, out] *lhs* First operand image.

[in] *rhs* Second operand image.

It performs:

for all *p* of *rhs*.domain

$lhs(p) = lhs(p) \text{ and not } rhs(p)$

#### Precondition

`rhs.domain >= lhs.domain`

References `mln::data::transform_inplace()`.

**9.104.2.4** `template<typename I > void mln::logical::not_inplace ( Image< I > & input ) [inline]`

Point-wise in-place "logical not" of image *input*.

#### Parameters

[in, out] *input* The target image.

It performs:

for all *p* of *input*.domain

$input(p) = \text{not } input(p)$

#### Precondition

`input.is_valid`

References `mln::data::transform_inplace()`.

**9.104.2.5** `template<typename L , typename R > void mln::logical::or_inplace ( Image< L > & lhs, const Image< R > & rhs ) [inline]`

Point-wise in-place "logical or" of image *rhs* in image *lhs*.

#### Parameters

[in, out] *lhs* First operand image.

[in] *rhs* Second operand image.

It performs:

for all *p* of *rhs*.domain

$lhs(p) = lhs(p) \text{ or } rhs(p)$

**Precondition**

```
rhs.domain >= lhs.domain
```

References mln::data::transform\_inplace().

**9.104.2.6 template<typename L , typename R > void mln::logical::xor\_inplace ( Image< L > & lhs, const Image< R > & rhs ) [inline]**

Point-wise in-place "logical xor" of image rhs in image lhs.

**Parameters**

[in, out] *lhs* First operand image.

[in] *rhs* Second operand image.

It performs:

for all p of rhs.domain

lhs(p) = lhs(p) xor rhs(p)

**Precondition**

```
rhs.domain >= lhs.domain
```

References mln::data::transform\_inplace().

**9.105 mln::logical::impl Namespace Reference**

Implementation namespace of logical namespace.

**Namespaces**

- namespace [generic](#)  
*Generic implementation namespace of logical namespace.*

**9.105.1 Detailed Description**

Implementation namespace of logical namespace.

**9.106 mln::logical::impl::generic Namespace Reference**

Generic implementation namespace of logical namespace.

**9.106.1 Detailed Description**

Generic implementation namespace of logical namespace.

## 9.107 mln::make Namespace Reference

Namespace of routines that help to make Milena's objects.

### Functions

- `template<unsigned D, typename G, typename V >`  
`p_set< complex_psite< D, G > > attachment` (const `complex_psite< D, G > &f`, const `complex_image< D, G, V > &ima`)  
*Compute the attachment of the cell corresponding to the facet  $f$  to the image  $ima$ .*
- `mln::box1d box1d` (`def::coord min_ind`, `def::coord max_ind`)  
*Create an `mln::box1d`.*
- `mln::box1d box1d` (unsigned `ninds`)  
*Create an `mln::box1d`.*
- `mln::box2d box2d` (unsigned `nrows`, unsigned `ncols`)  
*Create an `mln::box2d`.*
- `mln::box2d box2d` (`def::coord min_row`, `def::coord min_col`, `def::coord max_row`, `def::coord max_col`)  
*Create an `mln::box2d`.*
- `mln::box2d_h box2d_h` (`def::coord min_row`, `def::coord min_col`, `def::coord max_row`, `def::coord max_col`)  
*Create an `mln::box2d_h`.*
- `mln::box2d_h box2d_h` (unsigned `nrows`, unsigned `ncols`)  
*Create an `mln::box2d_h`.*
- `mln::box3d box3d` (unsigned `nslices`, unsigned `nrows`, unsigned `ncols`)  
*Create an `mln::box3d`.*
- `mln::box3d box3d` (`def::coord min_sli`, `def::coord min_row`, `def::coord min_col`, `def::coord max_sli`, `def::coord max_row`, `def::coord max_col`)  
*Create an `mln::box3d`.*
- `template<unsigned D, typename G >`  
`p_set< complex_psite< D, G > > cell` (const `complex_psite< D, G > &f`)  
*Compute the set of faces of the cell corresponding to the facet  $f$ .*
- `template<typename T, typename U >`  
`util::couple< T, U > couple` (const `T &val1`, const `T &val2`)  
*Construct an `mln::util::couple` on-the-fly.*
- `template<unsigned D, typename G, typename V >`  
`p_set< complex_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 , typename P >](#)  
[p\\_edges](#)< G, pw::cst\_< P > > [dummy\\_p\\_edges](#) (const [Graph](#)< G > &g\_, const P &dummy\_site)  
*Create a [p\\_edges](#) which associate a graph element to a constant site.*
- [template<typename G >](#)  
[p\\_edges](#)< G > [dummy\\_p\\_edges](#) (const [Graph](#)< G > &g)  
*Create a [p\\_edges](#) which associate a graph element to a constant site.*
- [template<typename G , typename P >](#)  
[p\\_vertices](#)< G, pw::cst\_< P > > [dummy\\_p\\_vertices](#) (const [Graph](#)< G > &g\_, const P &dummy\_site)  
*Create a [p\\_vertices](#) which associate a graph element to a constant site.*
- [template<typename G >](#)  
[p\\_vertices](#)< G > [dummy\\_p\\_vertices](#) (const [Graph](#)< G > &g)  
*Create a [p\\_vertices](#) which associate a graph element to a constant site.*
- [template<typename V , typename G >](#)  
[mln::edge\\_image](#)< void, V, G > [edge\\_image](#) (const [Graph](#)< G > &g, const fun::i2v::array< V > &fv)  
*Construct an edge image.*
- [template<typename FV , typename G >](#)  
[mln::edge\\_image](#)< void, typename FV::result, G > [edge\\_image](#) (const [Graph](#)< G > &g, const [Function\\_v2v](#)< FV > &fv)  
*Construct an edge image.*
- [template<typename FP , typename FV , typename G >](#)  
[mln::edge\\_image](#)< typename FP::result, typename FV::result, G > [edge\\_image](#) (const [Graph](#)< G > &g\_, const [Function\\_v2v](#)< FP > &fp, const [Function\\_v2v](#)< FV > &fv)  
*Construct an edge image.*
- [template<typename P , typename V , typename G , typename FP , typename FV >](#)  
[mln::edge\\_image](#)< typename FP::result, typename FV::result, G > [edge\\_image](#) (const [mln::vertex\\_image](#)< P, V, G > &v\_ima\_, const [p\\_edges](#)< G, FP > pe, const [Function\\_vv2v](#)< FV > &fv\_)  
*Construct an edge image.*
- [template<typename P , typename V , typename G , typename FV >](#)  
[mln::edge\\_image](#)< void, typename FV::result, G > [edge\\_image](#) (const [mln::vertex\\_image](#)< P, V, G > &v\_ima\_, const [Function\\_vv2v](#)< FV > &fv\_)  
*Construct an edge image.*
- [template<typename P , typename V , typename G , typename F >](#)  
[mln::edge\\_image](#)< void, bool, G > [edge\\_image](#) (const [mln::vertex\\_image](#)< P, V, G > &v\_ima\_, const [Function\\_v2b](#)< F > &fv\_)  
*Construct an edge image.*

- `template<typename T, unsigned N>`  
`algebra::h_mat< mlc_sqrt_int(N), T > h_mat (const T(&tab)[N])`  
*Create an `mln::algebra::mat<n,n,T>`.*
- `template<typename V, unsigned L>`  
`mln::image1d< V > image (V(&values)[L])`  
*Create an `image1d` from an 1D array of values.*
- `template<typename V, unsigned R, unsigned C>`  
`mln::image2d< V > image (V(&values)[R][C])`  
*Create an `image2d` from an 2D array of values.*
- `template<typename V, unsigned S, unsigned R, unsigned C>`  
`mln::image3d< V > image (V(&values)[S][R][C])`  
*Create an `image3d` from an 3D array of values.*
- `template<typename V, unsigned S>`  
`mln::image2d< V > image2d (V(&values)[S])`  
*Create an `image2d` from an 2D array of values.*
- `template<typename I >`  
`mln::image3d< typename I::value > image3d (const Image< I > &ima)`  
*Create an `image3d` from a 2D image.*
- `template<typename I >`  
`mln::image3d< typename I::value > image3d (const util::array< I > &ima)`  
*Create an `image3d` from an array of 2D images.*
- `template<typename I, typename N >`  
`util::graph_influence_zone_adjacency_graph (const Image< I > &iz_, const Neighborhood< N > &nbh, const typename I::value &nlabels)`  
*Create a graph from an influence zone image.*
- `template<unsigned n, unsigned m, typename T >`  
`algebra::mat< n, m, T > mat (const T(&tab)[n *m])`  
*Create an `mln::algebra::mat<n,m,T>`.*
- `template<typename T >`  
`util::ord_pair< T > ord_pair (const T &val1, const T &val2)`  
*Construct an `mln::util::ord_pair` on-the-fly.*
- `template<typename W, typename G >`  
`p_edges< G, fun::i2v::array< util::site_pair< typename W::site > > > p_edges_with_mass_centers (const Image< W > &wst_, const Graph< G > &g_)`  
*Construct a `p_edges` from a watershed image and a region adjacency graph (RAG).*
- `template<typename W, typename G >`  
`p_vertices< G, fun::i2v::array< typename W::site > > p_vertices_with_mass_centers (const Image< W > &wst_, const Graph< G > &g_)`  
*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_v2b< F > &fv2b, const V &nlabels, V &new_nlabels)`  
*Create a `i2v` function from a `v2b` function.*
- `template<typename V, typename F >`  
`fun::i2v::array< V > relabelfun (const Function_v2v< F > &fv2v, const V &nlabels, V &new_nlabels)`  
*Create a `i2v` function from a `v2v` function.*
- `template<typename T >`  
`algebra::vec< 1, T > vec (const T &v_0)`  
*Create an `mln::algebra::vec<n,T>`.*
- `template<typename T >`  
`algebra::vec< 4, T > vec (const T &v_0, const T &v_1, const T &v_2, const T &v_3)`  
*Create an `mln::algebra::vec<4,T>`.*
- `template<typename T >`  
`algebra::vec< 3, T > vec (const T &v_0, const T &v_1, const T &v_2)`  
*Create an `mln::algebra::vec<3,T>`.*
- `template<typename T >`  
`algebra::vec< 2, T > vec (const T &v_0, const T &v_1)`  
*Create an `mln::algebra::vec<2,T>`.*

- `template<typename G , typename FV >`  
`mln::vertex_image< void, typename FV::result, G > vertex_image (const Graph< G > &g, const Function_v2v< FV > &fv)`  
*Construct a vertex image.*
- `template<typename FP , typename FV , typename G >`  
`mln::vertex_image< typename FP::result, typename FV::result, G > vertex_image (const Graph< G > &g_, const Function_v2v< FP > &fp, const Function_v2v< FV > &fv)`  
*Construct a vertex image.*
- `template<typename I , typename N >`  
`p_vertices< util::graph, fun::i2v::array< typename I::site > > voronoi (Image< I > &ima_, Image< I > &orig_, const Neighborhood< N > &nbh)`  
*Apply the Voronoi algorithm on ima\_ with the original image orig\_ for node computing with neighborhood nbh.*
- `template<typename W , typename F >`  
`mln::w_window< typename W::dpsite, typename F::result > w_window (const Window< W > &win, const Function_v2v< F > &wei)`  
*Create a mln::w\_window from a window and a weight function.*
- `template<typename W , unsigned M >`  
`mln::w_window< mln::dpoint1d, W > w_window1d (W(&weights)[M])`  
*Create a 1D mln::w\_window from an array of weights.*
- `template<unsigned M >`  
`mln::w_window1d_int w_window1d_int (int(&weights)[M])`  
*Create a mln::w\_window1d\_int.*
- `template<typename W , unsigned S >`  
`mln::w_window< mln::dpoint2d, W > w_window2d (W(&weights)[S])`  
*Create a 2D mln::w\_window from an array of weights.*
- `template<unsigned M >`  
`mln::w_window2d_int w_window2d_int (int(&weights)[M])`  
*Create a mln::w\_window2d\_int.*
- `template<typename W , unsigned M >`  
`mln::w_window< mln::dpoint3d, W > w_window3d (W(&weights)[M])`  
*Create a 3D mln::w\_window from an array of weights.*
- `template<unsigned M >`  
`mln::w_window3d_int w_window3d_int (int(&weights)[M])`  
*Create a mln::w\_window3d\_int.*
- `template<typename D , typename W , unsigned L >`  
`mln::w_window< D, W > w_window_directional (const Gdpoint< D > &dp, W(&weights)[L])`  
*Create a directional centered weighted window.*



## 9.107.1 Detailed Description

Namespace of routines that help to make Milena's objects.

## 9.107.2 Function Documentation

**9.107.2.1** `template<unsigned D, typename G, typename V> p_set< complex_psite< D, G > >  
mln::make::attachment ( const complex_psite< D, G > & f, const complex_image< D,  
G, V > & ima ) [inline]`

Compute the attachment of the cell corresponding to the facet *f* to the image *ima*.

### Precondition

*f* is a facet (it does not belong to any face of higher dimension).  
*ima* is an image of Boolean values.

### Returns

a set of faces containing the attachment.

We do not use the formal definition of the attachment here (see `coupric.08.pami`). We use the following (equivalent) definition: an N-face *F* in *CELL* is in the attachment of *CELL* to *IMA* if it is adjacent to at least an (N-1)-face or an (N+1)-face that does not belong to *CELL*.

References `cell()`, and `mln::topo::is_facet()`.

Referenced by `mln::topo::is_simple_cell< I >::operator()`.

**9.107.2.2** `mln::box1d mln::make::box1d ( def::coord min_ind, def::coord max_ind )  
[inline]`

Create an `mln::box1d`.

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

### Parameters

[in] *min\_ind* Minimum index.

[in] *max\_ind* Maximum index.

### Precondition

`max_ind >= min_ind`.

### Returns

A 1D box.

References `box1d()`.

**9.107.2.3** `mln::box1d mln::make::box1d ( unsigned ninds ) [inline]`

Create an `mln::box1d`.

**Parameters**

[in] *ninds* Number of indices.

**Precondition**

$ninds \neq 0$  and  $ncols \neq 0$ .

**Returns**

A 1D box.

Referenced by `box1d()`, and `mln::image1d< T >::image1d()`.

**9.107.2.4 mln::box2d mln::make::box2d ( unsigned *nrows*, unsigned *ncols* ) [inline]**

Create an `mln::box2d`.

**Parameters**

[in] *nrows* Number of rows.

[in] *ncols* Number of columns.

**Precondition**

$nrows \neq 0$  and  $ncols \neq 0$ .

**Returns**

A 2D box.

Referenced by `mln::image2d< T >::image2d()`, and `mln::io::pnm::load()`.

**9.107.2.5 mln::box2d mln::make::box2d ( def::coord *min\_row*, def::coord *min\_col*, def::coord *max\_row*, def::coord *max\_col* ) [inline]**

Create an `mln::box2d`.

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

**Parameters**

[in] *min\_row* Index of the top most row.

[in] *min\_col* Index of the left most column.

[in] *max\_row* Index of the bottom most row.

[in] *max\_col* Index of the right most column.

**Precondition**

$max\_row \geq min\_row$  and  $max\_col \geq min\_col$ .

**Returns**

A 2D box.

**9.107.2.6 mln::box2d\_h mln::make::box2d\_h ( def::coord *min\_row*, def::coord *min\_col*, def::coord *max\_row*, def::coord *max\_col* ) [inline]**

Create an [mln::box2d\\_h](#).

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

**Parameters**

- [in] *min\_row* Index of the top most row.
- [in] *min\_col* Index of the left most column.
- [in] *max\_row* Index of the bottom most row.
- [in] *max\_col* Index of the right most column.

**Precondition**

$max\_row \geq min\_row$  and  $max\_col \geq min\_col$ .

**Returns**

A 2D\_H box.

References [point2d\\_h\(\)](#).

**9.107.2.7 mln::box2d\_h mln::make::box2d\_h ( unsigned *nrows*, unsigned *ncols* ) [inline]**

Create an [mln::box2d\\_h](#).

**Parameters**

- [in] *nrows* Number of rows.
- [in] *ncols* Number of columns.

**Precondition**

$nrows \neq 0$  and  $ncols \neq 0$ .

**Returns**

A 2D\_H box.

References [point2d\\_h\(\)](#).

**9.107.2.8 mln::box3d mln::make::box3d ( unsigned *nslices*, unsigned *nrows*, unsigned *ncols* ) [inline]**

Create an [mln::box3d](#).

**Parameters**

- [in] *nslices* Number of slices.
- [in] *nrows* Number of rows.
- [in] *ncols* Number of columns.

**Precondition**

`ninds != 0` and `ncols != 0` and `nslices != 0`.

**Returns**

A 3D box.

Referenced by `image3d()`, and `mln::image3d< T >::image3d()`.

### 9.107.2.9 `mln::box3d mln::make::box3d ( def::coord min_sli, def::coord min_row, def::coord min_col, def::coord max_sli, def::coord max_row, def::coord max_col ) [inline]`

Create an [mln::box3d](#).

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

**Parameters**

- [in] *min\_sli* Index of the lowest slice.
- [in] *min\_row* Index of the top most row.
- [in] *min\_col* Index of the left most column.
- [in] *max\_sli* Index of the highest slice.
- [in] *max\_row* Index of the botton most row.
- [in] *max\_col* Index of the right most column.

**Precondition**

```
max_sli >= min_sli.
max_row >= min_row.
max_col >= min_col.
```

**Returns**

A 3D box.

### 9.107.2.10 `template<unsigned D, typename G > p_set< complex_psite< D, G > > mln::make::cell ( const complex_psite< D, G > & f ) [inline]`

Compute the set of faces of the cell corresponding to the facet *f*.

**Precondition**

*f* is a facet (it does not belong to any face of higher dimension).

**Returns**

An [mln::p\\_set](#) of sites (faces) containing the attachment.

References `mln::topo::is_facet()`, and `mln::complex_psite< D, G >::n()`.

Referenced by `attachment()`, and `detachment()`.

**9.107.2.11** `template<typename T , typename U > util::couple<T,U> mln::make::couple ( const T & val1, const T & val2 )`

Construct an [mln::util::couple](#) on-the-fly.

Referenced by `mln::labeling::blobs_and_compute()`, `mln::transform::distance_and_closest_point_geodesic()`, `mln::transform::distance_and_influence_zone_geodesic()`, and `mln::labeling::value_and_compute()`.

**9.107.2.12** `template<unsigned D, typename G , typename V > p_set< complex_psite< D, G > > mln::make::detachment ( const complex_psite< D, G > & f, const complex_image< D, G, V > & ima ) [inline]`

Compute the detachment of the cell corresponding to the facet *f* to the image *ima*.

#### Precondition

*f* is a facet (it does not belong to any face of higher dimension).  
*ima* is an image of Boolean values.

#### Returns

a set of faces containing the detachment.

We do not use the formal definition of the detachment here (see `couple.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()`.

**9.107.2.13** `mln::dpoint2d_h mln::make::dpoint2d_h ( def::coord row, def::coord col ) [inline]`

Create an [mln::dpoint2d\\_h](#).

#### Parameters

[in] *row* Row coordinate.  
[in] *col* Column coordinate.

#### Returns

A 2D dpoint.

**9.107.2.14** `template<typename G , typename P > p_edges< G, pw::cst_< P > > mln::make::dummy_p_edges ( const Graph< G > & g_, const P & dummy_site )`

Create a [p\\_edges](#) which associate a graph element to a constant site.

#### Parameters

[in] *g\_* A graph.

[in] *dummy\_site* The dummy site mapped to graph edges.

### Returns

A [p\\_edges](#).

**9.107.2.15** `template<typename G > p_edges< G > mln::make::dummy_p_edges ( const Graph< G > & g )`

Create a [p\\_edges](#) which associate a graph element to a constant site.

0 (int) is used as dummy site.

### Parameters

[in] *g* A graph.

### Returns

A [p\\_edges](#).

**9.107.2.16** `template<typename G , typename P > p_vertices< G, pw::cst_< P > > mln::make::dummy_p_vertices ( const Graph< G > & g_, const P & dummy_site )`

Create a [p\\_vertices](#) which associate a graph element to a constant site.

### Parameters

[in] *g\_* A graph.

[in] *dummy\_site* The dummy site mapped to graph vertices.

### Returns

A [p\\_vertices](#).

**9.107.2.17** `template<typename G > p_vertices< G > mln::make::dummy_p_vertices ( const Graph< G > & g )`

Create a [p\\_vertices](#) which associate a graph element to a constant site.

0 (int) is used as dummy site.

### Parameters

[in] *g* A graph.

### Returns

A [p\\_vertices](#).

**9.107.2.18** `template<typename V , typename G > mln::edge_image< void, V, G >  
mln::make::edge_image ( const Graph< G > & g, const fun::i2v::array< V > & fv )  
[inline]`

Construct an edge image.

#### Parameters

- [in] *g* A graph.
- [in] *fv* A function mapping edge ids to values.

#### Returns

an edge image.

**9.107.2.19** `template<typename FV , typename G > mln::edge_image< void, typename FV::result,  
G > mln::make::edge_image ( const Graph< G > & g, const Function_v2v< FV > &  
fv )`

Construct an edge image.

#### Parameters

- [in] *g* A graph.
- [in] *fv* A function mapping edge ids to values.

#### Returns

an edge image.

**9.107.2.20** `template<typename FP , typename FV , typename G > mln::edge_image< typename  
FP::result, typename FV::result, G > mln::make::edge_image ( const Graph< G > &  
g_, const Function_v2v< FP > & fp, const Function_v2v< FV > & fv ) [inline]`

Construct an edge image.

#### Parameters

- [in] *g\_* A graph.
- [in] *fp* A function mapping edge ids to sites.
- [in] *fv* A function mapping edge ids to values.

#### Returns

an edge image.

**9.107.2.21** `template<typename P , typename V , typename G , typename FP , typename  
FV > mln::edge_image< typename FP::result, typename FV::result, G >  
mln::make::edge_image ( const mln::vertex_image< P, V, G > & v_ima_, const  
p_edges< G, FP > pe, const Function_vv2v< FV > & fv_ ) [inline]`

Construct an edge image.

**Parameters**

- [in] *v\_ima\_* A vertex image.
- [in] *pe* A [p\\_edges](#) mapping graph elements to sites.
- [in] *fv\_* A function mapping two vertex ids to a value. The result is associated to the corresponding edge.

**Returns**

an edge image.

**9.107.2.22** `template<typename P , typename V , typename G , typename FV > mln::edge_image< void, typename FV::result, G > mln::make::edge_image ( const mln::vertex_image< P, V, G > & v_ima_ , const Function_vv2v< FV > & fv_ ) [inline]`

Construct an edge image.

**Parameters**

- [in] *v\_ima\_* A vertex image.
- [in] *fv\_* A function mapping two vertices' values to a value. The result is associated to the corresponding edge.

**Returns**

an edge image without localization information mapped to graph elements.

**9.107.2.23** `template<typename P , typename V , typename G , typename F > mln::edge_image< void, bool, G > mln::make::edge_image ( const mln::vertex_image< P, V, G > & v_ima_ , const Function_v2b< F > & fv_ ) [inline]`

Construct an edge image.

**Parameters**

- [in] *v\_ima\_* A vertex image.
- [in] *fv\_* A predicate on a vertex's value. The (Boolean) result is associated to the edges adjacent to the vertex.

**Returns**

an edge image without localization information mapped to graph elements.

References `mln::data::fill()`.

**9.107.2.24** `template<typename T , unsigned N> algebra::h_mat< mlc_sqrt_int(N), T > mln::make::h_mat ( const T(&) tab[N] ) [inline]`

Create an `mln::algebra::mat<n,n,T>`.

Referenced by `mln::fun::x2x::rotation< n, C >::rotation()`.



**9.107.2.25** `template<typename V , unsigned L> mln::image1d< V > mln::make::image ( V(& values[L] )`

Create an [image1d](#) from an 1D array of values.

#### Parameters

[in] *values* 1D array.

#### Returns

A 1D image.

**9.107.2.26** `template<typename V , unsigned R, unsigned C> mln::image2d< V > mln::make::image ( V(& values[R][C] )`

Create an [image2d](#) from an 2D array of values.

#### Parameters

[in] *values* 2D array.

#### Returns

A 2D image.

References `mln::opt::at()`.

**9.107.2.27** `template<typename V , unsigned S, unsigned R, unsigned C> mln::image3d< V > mln::make::image ( V(& values[S][R][C] )`

Create an [image3d](#) from an 3D array of values.

#### Parameters

[in] *values* 3D array.

#### Returns

A 3D image.

References `mln::opt::at()`.

**9.107.2.28** `template<typename V , unsigned S> mln::image2d< V > mln::make::image2d ( V(& values[S] )`

Create an [image2d](#) from an 2D array of values.

#### Parameters

[in] *values* 2D array.

#### Returns

A 2D image.

**9.107.2.29** `template<typename I> mln::image3d< typename I::value > mln::make::image3d ( const Image< I> & ima ) [inline]`

Create an [image3d](#) from a 2D image.

References [box3d\(\)](#), and [mln::data::paste\(\)](#).

**9.107.2.30** `template<typename I> mln::image3d< typename I::value > mln::make::image3d ( const util::array< I> & ima ) [inline]`

Create an [image3d](#) from an array of 2D images.

References [box3d\(\)](#), [mln::util::array< T >::is\\_empty\(\)](#), [mln::util::array< T >::nelements\(\)](#), [mln::data::paste\(\)](#), [mln::box< P >::pmax\(\)](#), and [mln::box< P >::pmin\(\)](#).

Referenced by [mln::io::pnms::load\(\)](#).

**9.107.2.31** `template<typename I, typename N> util::graph mln::make::influence_zone_adjacency_graph ( const Image< I> & iz_, const Neighborhood< N> & nbh, const typename I::value & nlabels ) [inline]`

Create a graph from an influence zone image.

#### Parameters

[in] *iz* influence zone image.

[in] *nbh* A neighborhood.

[in] *nlabels* number of influence zone in *iz*.

#### Returns

[util::graph Graph](#) based on the adjacency of the influence zones.

**9.107.2.32** `template<unsigned n, unsigned m, typename T> algebra::mat< n, m, T > mln::make::mat ( const T(&) tab[n *m] ) [inline]`

Create an [mln::algebra::mat<n,m,T>](#).

#### Parameters

[in] *tab* Array of values.

#### Precondition

The array dimension has to be  $n * m$ .

**9.107.2.33** `template<typename T> util::ord_pair< T> mln::make::ord_pair ( const T & val1, const T & val2 ) [inline]`

Construct an [mln::util::ord\\_pair](#) on-the-fly.

**9.107.2.34** `template<typename W , typename G > p_edges< G, fun::i2v::array< util::site_pair< typename W::site > > > mln::make::p_edges_with_mass_centers ( const Image< W > & wst_, const Graph< G > & g_ ) [inline]`

Construct a [p\\_edges](#) from a watershed image and a region adjacency graph (RAG).

Map each graph edge to a pair of mass centers of two adjacent regions.

#### Parameters

*wst\_* A watershed image.

*g\_* A region adjacency graph.

#### Returns

A [p\\_edges](#).

#### See also

[edge\\_image](#), [p\\_edges](#), [make::region\\_adjacency\\_graph](#)

References [mln::labeling::compute\(\)](#).

**9.107.2.35** `template<typename W , typename G > p_vertices< G, fun::i2v::array< typename W::site > > mln::make::p_vertices_with_mass_centers ( const Image< W > & wst_, const Graph< G > & g_ ) [inline]`

Construct a [p\\_vertices](#) from a watershed image and a region adjacency graph (RAG).

Map each graph vertex to the mass center of its corresponding region.

#### Parameters

*wst\_* A watershed image.

*g\_* A region adjacency graph.

#### Returns

A [p\\_vertices](#).

#### See also

[edge\\_image](#), [vertex\\_image](#), [p\\_vertices](#), [p\\_edges](#), [make::region\\_adjacency\\_graph](#)

References [mln::labeling::compute\(\)](#).

**9.107.2.36** `template<typename I > mln::util::pix< I > mln::make::pix ( const Image< I > & ima, const typename I::psite & p ) [inline]`

Create an [mln::util::pix](#) from an image *ima* and a psite *p*.

#### Parameters

[in] *ima* The input image.

[in] *p* The point site.

**Returns**

An [mln::util::pix](#).

**9.107.2.37** `template<typename I > mln::pixel< I > mln::make::pixel ( Image< I > & ima, const typename I::psite & p ) [inline]`

Create a [mln::pixel](#) from a mutable image `ima` and a point `p`.

**9.107.2.38** `template<typename I > mln::pixel< const I > mln::make::pixel ( const Image< I > & ima, const typename I::psite & p ) [inline]`

Create a [mln::pixel](#) from a constant image `ima` and a point `p`.

**9.107.2.39** `mln::point2d_h mln::make::point2d_h ( def::coord row, def::coord col ) [inline]`

Create an [mln::point2d\\_h](#).

**Parameters**

[in] *row* Row coordinate.

[in] *col* Column coordinate.

**Returns**

A 2D point.

Referenced by `box2d_h()`.

**9.107.2.40** `template<typename I , typename N > util::couple< util::graph, typename mln::trait::concrete< I >::ret > mln::make::rag_and_labeled_wsl ( const Image< I > & wshd_, const Neighborhood< N > & nbh_, const typename I::value & nbasins ) [inline]`

Create a region adjacency graph and a label image of the watershed line from a watershed image.

**Parameters**

[in] *wshd\_* Watershed image.

[in] *nbh\_* [Neighborhood](#)

[in] *nbasins* Number of influence zone in `wshd`.

**Returns**

A couple. First element is the graph, second element is an image with a labeled watershed line.

```
|-----|
| 1 1 1 0 2 2 0 3 |
| 1 1 0 2 2 2 0 3 |
| 1 0 4 0 2 0 3 3 |
| 0 4 4 4 0 5 0 3 |
|-----|
|-----|
| . . . 1 . . 2 . |
| . . 1 . . . 2 . |
| . 1 . 3 . 4 . . |
| 1 . . . 5 . 6 . |
|-----|
```

Watershed image            Labeled watershed line  
 (watershed line labeled with 0)

|  
 |  
 |  
 v

```
1 -- 2 - 3
 \ / /
  4 -- 5
```

Region Adjacency graph (RAG)

**9.107.2.41** `template<typename I, typename N > util::graph mln::make::region_adjacency_graph ( const Image< I > & wshd_, const Neighborhood< N > & nbh, const typename I::value & nbasins ) [inline]`

Create a region adjacency graph from a watershed image.

#### Parameters

- [in] *wshd\_* watershed image.
- [in] *nbh* A neighborhood.
- [in] *nbasins* number of influence zone in wshd.

#### Returns

[util::graph Graph](#) based on the adjacency of the influence zones.

**9.107.2.42** `template<typename V, typename F > fun::i2v::array< V > mln::make::relabelfun ( const Function_v2b< F > & fv2b, const V & nlabels, V & new_nlabels ) [inline]`

Create a i2v function from a v2b function.

This function can be used to relabel a labeled image.

#### Parameters

- [in] *fv2b* A v2b function.
- [in] *nlabels* The number of labels.
- [in] *new\_nlabels* The number of labels after relabeling.

#### Returns

a i2v function.

#### See also

[mln::labeling::relabel](#)

References `mln::literal::zero`.

Referenced by `mln::labeling::pack()`, `mln::labeling::pack_inplace()`, `mln::labeling::relabel()`, `mln::labeled_image_base< I, E >::relabel()`, and `mln::labeling::relabel_inplace()`.

**9.107.2.43** `template<typename V , typename F > fun::i2v::array< V > mln::make::relabelfun ( const Function_v2v< F > & fv2v, const V & nlabels, V & new_nlabels ) [inline]`

Create a i2v function from a v2v function.

This function can be used to relabel a labeled image.

#### Parameters

[in] *fv2v* A v2v function. This function maps an id to an already existing one.

[in] *nlabels* The number of labels.

[in] *new\_nlabels* The number of labels after relabeling.

#### Returns

a i2v function.

#### See also

[mln::labeling::relabel](#)

References `mln::literal::zero`.

**9.107.2.44** `template<typename T > algebra::vec< 1, T > mln::make::vec ( const T & v_0 ) [inline]`

Create an `mln::algebra::vec<n,T>`.

#### Parameters

[in] *v\_0* First coordinate.

#### Returns

A 1D vector.

**9.107.2.45** `template<typename T > algebra::vec< 4, T > mln::make::vec ( const T & v_0, const T & v_1, const T & v_2, const T & v_3 ) [inline]`

Create an `mln::algebra::vec<4,T>`.

#### Parameters

[in] *v\_0* First coordinate.

[in] *v\_1* Second coordinate.

[in] *v\_2* Third coordinate.

[in] *v\_3* Fourth coordinate.

#### Returns

A 4D vector.

**9.107.2.46** `template<typename T > algebra::vec< 3, T > mln::make::vec ( const T & v_0, const T & v_1, const T & v_2 ) [inline]`

Create an `mln::algebra::vec<3,T>`.

#### Parameters

- [in] `v_0` First coordinate.
- [in] `v_1` Second coordinate.
- [in] `v_2` Third coordinate.

#### Returns

A 3D vector.

**9.107.2.47** `template<typename T > algebra::vec< 2, T > mln::make::vec ( const T & v_0, const T & v_1 ) [inline]`

Create an `mln::algebra::vec<2,T>`.

#### Parameters

- [in] `v_0` First coordinate.
- [in] `v_1` Second coordinate.

#### Returns

A 2D vector.

**9.107.2.48** `template<typename G , typename FV > mln::vertex_image< void, typename FV::result, G > mln::make::vertex_image ( const Graph< G > & g, const Function_v2v< FV > & fv )`

Construct a vertex image.

#### Parameters

- [in] `g` A graph.
- [in] `fv` A function mapping vertex ids to values.

#### Returns

A vertex image.

**9.107.2.49** `template<typename FP , typename FV , typename G > mln::vertex_image< typename FP::result, typename FV::result, G > mln::make::vertex_image ( const Graph< G > & g, const Function_v2v< FP > & fp, const Function_v2v< FV > & fv )`

Construct a vertex image.

**Parameters**

- [in] *g\_* A graph.
- [in] *fp* A function mapping vertex ids to sites.
- [in] *fv* A function mapping vertex ids to values.

**Returns**

A vertex image.

**9.107.2.50** `template<typename I , typename N > p_vertices< util::graph, fun::i2v::array< typename I::site > > mln::make::voronoi ( Image< I > & ima_, Image< I > & orig_, const Neighborhood< N > & nbh ) [inline]`

Apply the Voronoi algorithm on *ima\_* with the original image *orig\_* for node computing with neighborhood *nbh*.

**Parameters**

- [in] *ima\_* The labeling image.
- [in] *orig\_* The original image.
- [in] *nbh* The neighborhood for computing algorithm.

**Returns**

The computed graph.

References `mln::util::graph::add_edge()`, `mln::util::graph::add_vertex()`, and `mln::estim::min_max()`.

**9.107.2.51** `template<typename W , typename F > mln::w_window< typename W::dpsite, typename F::result > mln::make::w_window ( const Window< W > & win, const Function_v2v< F > & wei ) [inline]`

Create a `mln::w_window` from a window and a weight function.

**Parameters**

- [in] *win* A simple window.
- [in] *wei* A weight function.

**Returns**

A weighted window.

References `mln::w_window< D, W >::insert()`, and `mln::literal::origin`.

**9.107.2.52** `template<typename W , unsigned M> mln::w_window< mln::dpoint1d, W > mln::make::w_window1d ( W(&) weights[M] ) [inline]`

Create a 1D `mln::w_window` from an array of weights.



**Parameters**

[in] *weights* Array.

**Precondition**

The array size, *M*, has to be a square of an odd integer.

**Returns**

A 1D weighted window.

References `mln::w_window< D, W >::insert()`.

Referenced by `w_window1d_int()`.

**9.107.2.53** `template<unsigned M> mln::w_window1d_int mln::make::w_window1d_int ( int(&)  
weights[M] ) [inline]`

Create a [mln::w\\_window1d\\_int](#).

**Parameters**

[in] *weights* Array of integers.

**Precondition**

The array size, *M*, has to be a square of an odd integer.

**Returns**

A 1D int-weighted window.

References `w_window1d()`.

**9.107.2.54** `template<typename W , unsigned S> mln::w_window< mln::dpoint2d, W >  
mln::make::w_window2d ( W(&) weights[S] ) [inline]`

Create a 2D [mln::w\\_window](#) from an array of weights.

**Parameters**

[in] *weights* Array.

**Precondition**

The array size, *S*, has to be a square of an odd integer.

**Returns**

A 2D weighted window.

Referenced by `w_window2d_int()`.

**9.107.2.55** `template<unsigned M> mln::w_window2d_int mln::make::w_window2d_int ( int(&)  
weights[M] ) [inline]`

Create a [mln::w\\_window2d\\_int](#).

#### Parameters

[in] *weights* Array of integers.

#### Precondition

The array size, M, has to be a square of an odd integer.

#### Returns

A 2D int-weighted window.

References `w_window2d()`.

**9.107.2.56** `template<typename W , unsigned M> mln::w_window< mln::dpoint3d, W >  
mln::make::w_window3d ( W(&) weights[M] ) [inline]`

Create a 3D [mln::w\\_window](#) from an array of weights.

#### Parameters

[in] *weights* Array.

#### Precondition

The array size, M, has to be a cube of an odd integer.

#### Returns

A 3D weighted window.

References `mln::w_window< D, W >::insert()`.

Referenced by `w_window3d_int()`.

**9.107.2.57** `template<unsigned M> mln::w_window3d_int mln::make::w_window3d_int ( int(&)  
weights[M] ) [inline]`

Create a [mln::w\\_window3d\\_int](#).

#### Parameters

[in] *weights* Array of integers.

#### Precondition

The array size, M, has to be a cube of an odd integer.

#### Returns

A 3D int-weighted window.

References `w_window3d()`.

**9.107.2.58** `template<typename D , typename W , unsigned L> mln::w_window< D, W >`  
`mln::make::w_window_directional ( const Gdpoint< D > & dp, W(&) weights[L] )`  
`[inline]`

Create a directional centered weighted window.

#### Parameters

- [in] *dp* A delta-point to set the orientation.
- [in] *weights* An array of weights.

#### Returns

A weighted window.

The window length L has to be odd.

References mln::w\_window< D, W >::insert(), and mln::literal::zero.

## 9.108 mln::math Namespace Reference

Namespace of mathematical routines.

### Functions

- `template<typename T >`  
`T abs (const T &v)`  
*Generic version.*
- `template<unsigned n>`  
`value::int_u< n > abs (const value::int_u< n > &v)`  
*Specialization for mln::value::int\_u.*
- `int abs (int v)`  
*Specializations for existing overloads of std::abs.*

### 9.108.1 Detailed Description

Namespace of mathematical routines.

### 9.108.2 Function Documentation

**9.108.2.1** `template<typename T > T mln::math::abs ( const T & v ) [inline]`

Generic version.

Referenced by abs(), and mln::morpho::line\_gradient().

### 9.108.2.2 `int mln::math::abs ( int v ) [inline]`

Specializations for existing overloads of `std::abs`.

Reference: ISO/IEC 14882:2003 C++ standard, section 26.5 (C Library, [lib.c.math]).

References `abs()`.

### 9.108.2.3 `template<unsigned n> value::int_u< n > mln::math::abs ( const value::int_u< n > & v ) [inline]`

Specialization for `mln::value::int_u`.

## 9.109 `mln::metal` Namespace Reference

Namespace of meta-programming tools.

### Namespaces

- namespace `impl`  
*Implementation namespace of metal namespace.*
- namespace `math`  
*Namespace of static mathematical functions.*

### Classes

- struct `ands`  
*Ands type.*
- struct `converts_to`  
*"converts-to" check.*
- struct `equal`  
*Definition of a static 'equal' test.*
- struct `goes_to`  
*"goes-to" check.*
- struct `is`  
*"is" check.*
- struct `is_a`  
*"is\_a" check.*
- struct `is_not`  
*"is\_not" check.*

- struct `is_not_a`  
*"is\_not\_a" static Boolean expression.*

### 9.109.1 Detailed Description

Namespace of meta-programming tools.

## 9.110 `mln::metal::impl` Namespace Reference

Implementation namespace of metal namespace.

### 9.110.1 Detailed Description

Implementation namespace of metal namespace.

## 9.111 `mln::metal::math` Namespace Reference

Namespace of static mathematical functions.

### Namespaces

- namespace `impl`  
*Implementation namespace of `metal::math` namespace.*

### 9.111.1 Detailed Description

Namespace of static mathematical functions.

## 9.112 `mln::metal::math::impl` Namespace Reference

Implementation namespace of `metal::math` namespace.

### 9.112.1 Detailed Description

Implementation namespace of `metal::math` namespace.

## 9.113 `mln::morpho` Namespace Reference

Namespace of mathematical morphology routines.

## Namespaces

- namespace [approx](#)  
*Namespace of approximate mathematical morphology routines.*
- namespace [attribute](#)  
*Namespace of attributes used in mathematical morphology.*
- namespace [elementary](#)  
*Namespace of image processing routines of elementary mathematical morphology.*
- namespace [impl](#)  
*Namespace of mathematical morphology routines implementations.*
- namespace [reconstruction](#)  
*Namespace of morphological reconstruction routines.*
- namespace [tree](#)  
*Namespace of morphological tree-related routines.*
- namespace [watershed](#)  
*Namespace of morphological watershed routines.*

## Functions

- `template<typename I >`  
`mln::trait::concrete< I >::ret complementation (const Image< I > &input)`
- `template<typename I >`  
`void complementation\_inplace (Image< I > &input)`
- `template<typename I , typename W >`  
`mln::trait::concrete< I >::ret contrast (const Image< I > &input, const Window< W > &win)`
- `template<typename I , typename W >`  
`mln::trait::concrete< I >::ret dilation (const Image< I > &input, const Window< W > &win)`  
*Morphological dilation.*
- `template<typename I , typename W >`  
`mln::trait::concrete< I >::ret erosion (const Image< I > &input, const Window< W > &win)`  
*Morphological erosion.*
- `template<typename Op , typename I , typename W >`  
`mln::trait::concrete< I >::ret general (const Op &op, const Image< I > &input, const Window< W > &win)`  
*Morphological general routine.*
- `template<typename I , typename W >`  
`mln::trait::concrete< I >::ret gradient (const Image< I > &input, const Window< W > &win)`  
*Morphological gradient.*

- `template<typename I, typename W >`  
`mln::trait::concrete< I >::ret gradient\_external (const Image< I > &input, const Window< W > &win)`  
*Morphological external gradient.*
- `template<typename I, typename W >`  
`mln::trait::concrete< I >::ret gradient\_internal (const Image< I > &input, const Window< W > &win)`  
*Morphological internal gradient.*
- `template<typename I, typename Wh, typename Wm >`  
`mln::trait::concrete< I >::ret hit\_or\_miss (const Image< I > &input, const Window< Wh > &win_  
hit, const Window< Wm > &win_miss)`  
*Morphological hit-or-miss.*
- `template<typename I, typename Wh, typename Wm >`  
`mln::trait::concrete< I >::ret hit\_or\_miss\_background\_closing (const Image< I > &input, const Window< Wh > &win_hit, const Window< Wm > &win_miss)`  
*Morphological hit-or-miss closing of the background.*
- `template<typename I, typename Wh, typename Wm >`  
`mln::trait::concrete< I >::ret hit\_or\_miss\_background\_opening (const Image< I > &input, const Window< Wh > &win_hit, const Window< Wm > &win_miss)`  
*Morphological hit-or-miss opening of the background.*
- `template<typename I, typename Wh, typename Wm >`  
`mln::trait::concrete< I >::ret hit\_or\_miss\_closing (const Image< I > &input, const Window< Wh > &win_hit, const Window< Wm > &win_miss)`  
*Morphological hit-or-miss closing.*
- `template<typename I, typename Wh, typename Wm >`  
`mln::trait::concrete< I >::ret hit\_or\_miss\_opening (const Image< I > &input, const Window< Wh > &win_hit, const Window< Wm > &win_miss)`  
*Morphological hit-or-miss opening.*
- `template<typename I, typename W, typename O >`  
`void laplacian (const Image< I > &input, const Window< W > &win, Image< O > &output)`
- `template<typename V >`  
`edge\_image< util::site\_pair< point2d >, V, util::graph > line\_gradient (const mln::image2d< V > &ima)`  
*Create a line graph image representing the gradient norm of a [mln::image2d](#).*
- `template<typename L, typename I, typename N >`  
`mln::trait::ch_value< I, L >::ret meyer\_wst (const Image< I > &input, const Neighborhood< N > &nbh, L &nbasins)`  
*Meyer's Watershed Transform (WST) algorithm.*
- `template<typename L, typename I, typename N >`  
`mln::trait::ch_value< I, L >::ret meyer\_wst (const Image< I > &input, const Neighborhood< N > &nbh)`  
*Meyer's Watershed Transform (WST) algorithm, with no count of basins.*

- `template<typename I , typename J >`  
`mln::trait::concrete< I >::ret min (const Image< I > &lhs, const Image< J > &rhs)`
- `template<typename I , typename J >`  
`void min_inplace (Image< I > &lhs, const Image< J > &rhs)`
- `template<typename I , typename J >`  
`mln::trait::concrete< I >::ret minus (const Image< I > &lhs, const Image< J > &rhs)`
- `template<typename I , typename J >`  
`mln::trait::concrete< I >::ret plus (const Image< I > &lhs, const Image< J > &rhs)`
- `template<typename I , typename W >`  
`mln::trait::concrete< I >::ret rank_filter (const Image< I > &input, const Window< W > &win, unsigned k)`

*Morphological rank\_filter.*

- `template<typename I , typename Wfg , typename Wbg >`  
`mln::trait::concrete< I >::ret thick_miss (const Image< I > &input, const Window< Wfg > &win_fg, const Window< Wbg > &win_bg)`
- `template<typename I , typename Wfg , typename Wbg >`  
`mln::trait::concrete< I >::ret thickening (const Image< I > &input, const Window< Wfg > &win_fg, const Window< Wbg > &win_bg)`
- `template<typename I , typename Wfg , typename Wbg >`  
`mln::trait::concrete< I >::ret thin_fit (const Image< I > &input, const Window< Wfg > &win_fg, const Window< Wbg > &win_bg)`
- `template<typename I , typename Wfg , typename Wbg >`  
`mln::trait::concrete< I >::ret thinning (const Image< I > &input, const Window< Wfg > &win_fg, const Window< Wbg > &win_bg)`

*Morphological thinning.*

- `template<typename I , typename W >`  
`mln::trait::concrete< I >::ret top_hat_black (const Image< I > &input, const Window< W > &win)`

*Morphological black top-hat (for background / dark objects).*

- `template<typename I , typename W >`  
`mln::trait::concrete< I >::ret top_hat_self_complementary (const Image< I > &input, const Window< W > &win)`

*Morphological self-complementary top-hat.*

- `template<typename I , typename W >`  
`mln::trait::concrete< I >::ret top_hat_white (const Image< I > &input, const Window< W > &win)`

*Morphological white top-hat (for object / light objects).*

### 9.113.1 Detailed Description

Namespace of mathematical morphology routines.



## 9.113.2 Function Documentation

**9.113.2.1** `template<typename I > mln::trait::concrete< I >::ret mln::morpho::complementation ( const Image< I > & input ) [inline]`

Morphological complementation: either a logical "not" (if morpho on sets) or an arithmetical complementation (if morpho on functions).

Referenced by `hit_or_miss_background_closing()`, `hit_or_miss_background_opening()`, `hit_or_miss_closing()`, and `thinning()`.

**9.113.2.2** `template<typename I > void mln::morpho::complementation_inplace ( Image< I > & input ) [inline]`

Morphological complementation, inplace version: either a logical "not" (if morpho on sets) or an arithmetical complementation (if morpho on functions).

**9.113.2.3** `template<typename I , typename W > mln::trait::concrete< I >::ret mln::morpho::contrast ( const Image< I > & input, const Window< W > & win ) [inline]`

Morphological contrast operator (based on top-hats).

This operator is  $I_d + w_{th\_B} - b_{th\_B}$ .

References `plus()`, `top_hat_black()`, and `top_hat_white()`.

**9.113.2.4** `template<typename I , typename W > mln::trait::concrete< I >::ret mln::morpho::dilation ( const Image< I > & input, const Window< W > & win ) [inline]`

Morphological dilation.

References `general()`.

Referenced by `gradient()`, `gradient_external()`, `hit_or_miss_background_opening()`, `hit_or_miss_opening()`, `laplacian()`, `mln::morpho::opening::approx::structural()`, and `mln::morpho::closing::approx::structural()`.

**9.113.2.5** `template<typename I , typename W > mln::trait::concrete< I >::ret mln::morpho::erosion ( const Image< I > & input, const Window< W > & win ) [inline]`

Morphological erosion.

References `general()`.

Referenced by `gradient()`, `gradient_internal()`, `laplacian()`, `mln::morpho::opening::approx::structural()`, and `mln::morpho::closing::approx::structural()`.

**9.113.2.6** `template<typename Op , typename I , typename W > mln::trait::concrete< I >::ret  
mln::morpho::general ( const Op & op, const Image< I > & input, const Window<  
W > & win ) [inline]`

Morphological general routine.

Referenced by dilation(), and erosion().

**9.113.2.7** `template<typename I , typename W > mln::trait::concrete< I >::ret  
mln::morpho::gradient ( const Image< I > & input, const Window< W > & win )  
[inline]`

Morphological gradient.

This operator is  $d_B - e_B$ .

References dilation(), erosion(), minus(), and mln::test::positive().

**9.113.2.8** `template<typename I , typename W > mln::trait::concrete< I >::ret  
mln::morpho::gradient_external ( const Image< I > & input, const Window< W > &  
win ) [inline]`

Morphological external gradient.

This operator is  $d_B - Id$ .

References dilation(), minus(), and mln::test::positive().

**9.113.2.9** `template<typename I , typename W > mln::trait::concrete< I >::ret  
mln::morpho::gradient_internal ( const Image< I > & input, const Window< W > &  
win ) [inline]`

Morphological internal gradient.

This operator is  $Id - e_B$ .

References erosion(), minus(), and mln::test::positive().

**9.113.2.10** `template<typename I , typename Wh , typename Wm > mln::trait::concrete< I >::ret  
mln::morpho::hit_or_miss ( const Image< I > & input, const Window< Wh > &  
win_hit, const Window< Wm > & win_miss ) [inline]`

Morphological hit-or-miss.

This operator is  $HMT_(B_h, B_m) = e_{B_h} \wedge (e_{B_m} \circ C)$ .

Referenced by thickening(), and thinning().

**9.113.2.11** `template<typename I , typename Wh , typename Wm > mln::trait::concrete< I >::ret  
mln::morpho::hit_or_miss_background_closing ( const Image< I > & input, const  
Window< Wh > & win_hit, const Window< Wm > & win_miss ) [inline]`

Morphological hit-or-miss closing of the background.

This operator is  $C \circ HMT_{\text{TopeBG}} \circ C$ .

References `complementation()`, `hit_or_miss_background_opening()`, and `hit_or_miss_closing()`.

**9.113.2.12** `template<typename I , typename Wh , typename Wm > mln::trait::concrete< I >::ret  
mln::morpho::hit_or_miss_background_opening ( const Image< I > & input, const  
Window< Wh > & win_hit, const Window< Wm > & win_miss ) [inline]`

Morphological hit-or-miss opening of the background.

This operator is  $HMT_{topeBG} = HMT_{tope}(B_m, B_h) \circ C = d_*(-B_m) \circ HMT_*(B_h, B_m)$ .

References `complementation()`, `dilation()`, `hit_or_miss_opening()`, and `mln::win::sym()`.

Referenced by `hit_or_miss_background_closing()`, and `thick_miss()`.

**9.113.2.13** `template<typename I , typename Wh , typename Wm > mln::trait::concrete< I >::ret  
mln::morpho::hit_or_miss_closing ( const Image< I > & input, const Window< Wh  
> & win_hit, const Window< Wm > & win_miss ) [inline]`

Morphological hit-or-miss closing.

This operator is  $C \circ HMT_{tope} \circ C$ .

References `complementation()`, and `hit_or_miss_opening()`.

Referenced by `hit_or_miss_background_closing()`.

**9.113.2.14** `template<typename I , typename Wh , typename Wm > mln::trait::concrete< I >::ret  
mln::morpho::hit_or_miss_opening ( const Image< I > & input, const Window< Wh  
> & win_hit, const Window< Wm > & win_miss ) [inline]`

Morphological hit-or-miss opening.

This operator is  $HMT_{tope}(B_h, B_m) = d_*(-B_h) \circ HMT_*(B_h, B_m)$ .

References `dilation()`, and `mln::win::sym()`.

Referenced by `hit_or_miss_background_opening()`, `hit_or_miss_closing()`, and `thin_fit()`.

**9.113.2.15** `template<typename I , typename W , typename O > void mln::morpho::laplacian (   
const Image< I > & input, const Window< W > & win, Image< O > & output )  
[inline]`

Morphological laplacian.

This operator is  $(d_B - Id) - (Id - e_B)$ .

References `dilation()`, `erosion()`, `mln::data::fill()`, and `minus()`.

**9.113.2.16** `template<typename V > edge_image< util::site_pair< point2d >, V, util::graph >  
mln::morpho::line_gradient ( const mln::image2d< V > & ima )`

Create a line graph image representing the gradient norm of a [mln::image2d](#).

References `mln::math::abs()`, `mln::image2d< T >::domain()`, `mln::box< P >::has()`, `mln::window< D >::insert()`, and `mln::Box< E >::nsites()`.

**9.113.2.17** `template<typename L , typename I , typename N > mln::trait::ch_value< I, L >::ret mln::morpho::meyer_wst ( const Image< I > & input, const Neighborhood< N > & nbh, L & nbasins )`

Meyer's Watershed Transform (WST) algorithm.

#### Parameters

- [in] *input* The input image.
- [in] *nbh* The connexity of markers.
- [out] *nbasins* The number of basins.

- L is the type of labels, used to number the watershed itself (with the minimal value), and the basins.
- I is the exact type of the input image.
- N is the exact type of the neighborhood used to express *input*'s connexity.

References `mln::data::fill()`, `mln::p_priority< P, Q >::front()`, `mln::initialize()`, `mln::p_priority< P, Q >::pop()`, `mln::p_priority< P, Q >::push()`, `mln::labeling::regional_minima()`, and `mln::literal::zero`.

**9.113.2.18** `template<typename L , typename I , typename N > mln::trait::ch_value< I, L >::ret mln::morpho::meyer_wst ( const Image< I > & input, const Neighborhood< N > & nbh )`

Meyer's Watershed Transform (WST) algorithm, with no count of basins.

#### Parameters

- [in] *input* The input image.
- [in] *nbh* The connexity of markers.

- L is the type of labels, used to number the watershed itself (with the minimal value), and the basins.
- I is the exact type of the input image.
- N is the exact type of the neighborhood used to express *input*'s connexity.

Note that the first parameter, L, is not automatically valued from the type of the actual argument during implicit instantiation: you have to explicitly pass this parameter at call sites.

**9.113.2.19** `template<typename I , typename J > mln::trait::concrete< I >::ret mln::morpho::min ( const Image< I > & lhs, const Image< J > & rhs ) [inline]`

Morphological min: either a logical "and" (if morpho on sets) or an arithmetical min (if morpho on functions).

**9.113.2.20** `template<typename I , typename J > void mln::morpho::min_inplace ( Image< I > & lhs, const Image< J > & rhs ) [inline]`

Morphological min, inplace version: either a logical "and" (if morpho on sets) or an arithmetical min (if morpho on functions).

**9.113.2.21** `template<typename I , typename J > mln::trait::concrete< I >::ret  
mln::morpho::minus ( const Image< I > & lhs, const Image< J > & rhs )  
[inline]`

Morphological minus: either a logical "and not" (if morpho on sets) or an arithmetical minus (if morpho on functions).

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()`.

**9.113.2.22** `template<typename I , typename J > mln::trait::concrete< I >::ret mln::morpho::plus  
( const Image< I > & lhs, const Image< J > & rhs ) [inline]`

Morphological plus: either a "logical or" (if morpho on sets) or an "arithmetical plus" (if morpho on functions).

Referenced by `contrast()`, `thick_miss()`, and `thickening()`.

**9.113.2.23** `template<typename I , typename W > mln::trait::concrete< I >::ret  
mln::morpho::rank_filter ( const Image< I > & input, const Window< W > & win,  
unsigned k ) [inline]`

Morphological `rank_filter`.

**9.113.2.24** `template<typename I , typename Wfg , typename Wbg > mln::trait::concrete< I  
>::ret mln::morpho::thick_miss ( const Image< I > & input, const Window< Wfg >  
& win_fg, const Window< Wbg > & win_bg ) [inline]`

Morphological thick-miss.

This operator is  $THICK\_B = Id + HMT_{TopeBG\_B}$ , where  $B = (Bfg, Bbg)$ .

References `hit_or_miss_background_opening()`, and `plus()`.

**9.113.2.25** `template<typename I , typename Wfg , typename Wbg > mln::trait::concrete< I  
>::ret mln::morpho::thickening ( const Image< I > & input, const Window< Wfg >  
& win_fg, const Window< Wbg > & win_bg ) [inline]`

Morphological thickening.

This operator is  $THICK\_B = Id + HMT\_B$ , where  $B = (Bfg, Bbg)$ .

References `hit_or_miss()`, and `plus()`.

Referenced by `thinning()`.

**9.113.2.26** `template<typename I , typename Wfg , typename Wbg > mln::trait::concrete< I  
>::ret mln::morpho::thin_fit ( const Image< I > & input, const Window< Wfg > &  
win_fg, const Window< Wbg > & win_bg ) [inline]`

Morphological thin-fit.

This operator is  $\text{THIN}_B = \text{Id} - \text{HMT}_{\text{Tope}_B}$  where  $B = (\text{Bfg}, \text{Bbg})$ .

References `hit_or_miss_opening()`, and `minus()`.

**9.113.2.27** `template<typename I , typename Wfg , typename Wbg > mln::trait::concrete< I >::ret mln::morpho::thinning ( const Image< I > & input, const Window< Wfg > & win_fg, const Window< Wbg > & win_bg ) [inline]`

Morphological thinning.

This operator is  $\text{THIN}_B = \text{Id} - \text{HMT}_B$ , where  $B = (\text{Bfg}, \text{Bbg})$ .

References `complementation()`, `hit_or_miss()`, `minus()`, and `thickening()`.

**9.113.2.28** `template<typename I , typename W > mln::trait::concrete< I >::ret mln::morpho::top_hat_black ( const Image< I > & input, const Window< W > & win ) [inline]`

Morphological black top-hat (for background / dark objects).

This operator is  $\text{clo}_B - \text{Id}$ .

References `minus()`, and `mln::test::positive()`.

Referenced by `contrast()`.

**9.113.2.29** `template<typename I , typename W > mln::trait::concrete< I >::ret mln::morpho::top_hat_self_complementary ( const Image< I > & input, const Window< W > & win ) [inline]`

Morphological self-complementary top-hat.

This operator is

$= \text{top\_hat\_white} + \text{top\_hat\_black}$

$= (\text{input} - \text{opening}) + (\text{closing} - \text{input})$

$= \text{closing} - \text{opening}$ .

References `minus()`, and `mln::test::positive()`.

**9.113.2.30** `template<typename I , typename W > mln::trait::concrete< I >::ret mln::morpho::top_hat_white ( const Image< I > & input, const Window< W > & win ) [inline]`

Morphological white top-hat (for object / light objects).

This operator is  $\text{Id} - \text{ope}_B$ .

References `minus()`, and `mln::test::positive()`.

Referenced by `contrast()`.

## 9.114 mln::morpho::approx Namespace Reference

Namespace of approximate mathematical morphology routines.

### 9.114.1 Detailed Description

Namespace of approximate mathematical morphology routines.

## 9.115 mln::morpho::attribute Namespace Reference

Namespace of attributes used in mathematical morphology.

### Classes

- class [card](#)  
*Cardinality accumulator class.*
- struct [count\\_adjacent\\_vertices](#)  
*Count\_Adjacent\_Vertices accumulator class.*
- struct [height](#)  
*Height accumulator class.*
- struct [sharpness](#)  
*Sharpness accumulator class.*
- class [sum](#)  
*Suminality accumulator class.*
- struct [volume](#)  
*Volume accumulator class.*

### 9.115.1 Detailed Description

Namespace of attributes used in mathematical morphology.

## 9.116 mln::morpho::closing::approx Namespace Reference

Namespace of approximate mathematical morphology closing routines.

### Functions

- `template<typename I, typename W >  
mln::trait::concrete< I >::ret structural (const Image< I > &input, const Window< W > &win)`  
*Approximate of morphological structural closing.*

### 9.116.1 Detailed Description

Namespace of approximate mathematical morphology closing routines.

## 9.116.2 Function Documentation

**9.116.2.1** `template<typename I , typename W > mln::trait::concrete< I >::ret  
mln::morpho::closing::approx::structural ( const Image< I > & input, const  
Window< W > & win ) [inline]`

Approximate of morphological structural closing.

This operator is  $e_{-B} \circ d_B$ .

References `mln::morpho::dilation()`, `mln::morpho::erosion()`, and `mln::win::sym()`.

## 9.117 mln::morpho::elementary Namespace Reference

Namespace of image processing routines of elementary mathematical morphology.

### Functions

- `template<typename I , typename N >  
mln::trait::concrete< I >::ret closing (const Image< I > &input, const Neighborhood< N >  
&nbh)`  
*Morphological elementary closing.*
- `template<typename I , typename N >  
mln_trait_op_minus_twice (typename mln::trait::concrete< I >::ret) laplacian(const Image< I >  
&input)`  
*Morphological elementary laplacian.*
- `template<typename I , typename N >  
mln::trait::concrete< I >::ret opening (const Image< I > &input, const Neighborhood< N >  
&nbh)`  
*Morphological elementary opening.*
- `template<typename I , typename N >  
mln::trait::concrete< I >::ret top_hat_black (const Image< I > &input, const Neighborhood< N >  
&nbh)`  
*Morphological elementary black top-hat (for background / dark objects).*
- `template<typename I , typename N >  
mln::trait::concrete< I >::ret top_hat_self_complementary (const Image< I > &input, const Neigh-  
borhood< 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).*



## 9.117.1 Detailed Description

Namespace of image processing routines of elementary mathematical morphology.

## 9.117.2 Function Documentation

**9.117.2.1** `template<typename I , typename N > mln::trait::concrete< I >::ret  
mln::morpho::elementary::closing ( const Image< I > & input, const Neighborhood<  
N > & nbh ) [inline]`

Morphological elementary closing.

This operator is e o d.

Referenced by top\_hat\_black(), and top\_hat\_self\_complementary().

**9.117.2.2** `template<typename I , typename N > mln::morpho::elementary::mln_  
trait_op_minus_twice ( typename mln::trait::concrete< I >::ret ) const  
[inline]`

Morphological elementary laplacian.

This operator is (d - id) - (id - e).

**9.117.2.3** `template<typename I , typename N > mln::trait::concrete< I >::ret  
mln::morpho::elementary::opening ( const Image< I > & input, const Neighborhood<  
N > & nbh ) [inline]`

Morphological elementary opening.

This operator is d o e.

Referenced by top\_hat\_self\_complementary(), and top\_hat\_white().

**9.117.2.4** `template<typename I , typename N > mln::trait::concrete< I >::ret  
mln::morpho::elementary::top_hat_black ( const Image< I > & input, const  
Neighborhood< N > & nbh ) [inline]`

Morphological elementary black top-hat (for background / dark objects).

This operator is clo - Id.

References closing(), mln::morpho::minus(), and mln::test::positive().

**9.117.2.5** `template<typename I , typename N > mln::trait::concrete< I >::ret  
mln::morpho::elementary::top_hat_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().

**9.117.2.6** `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().

## 9.118 mln::morpho::impl Namespace Reference

Namespace of mathematical morphology routines implementations.

### Namespaces

- namespace [generic](#)

*Namespace of mathematical morphology routines generic implementations.*

### 9.118.1 Detailed Description

Namespace of mathematical morphology routines implementations.

## 9.119 mln::morpho::impl::generic Namespace Reference

Namespace of mathematical morphology routines generic implementations.

### 9.119.1 Detailed Description

Namespace of mathematical morphology routines generic implementations.

## 9.120 mln::morpho::opening::approx Namespace Reference

Namespace of approximate mathematical morphology opening routines.

### Functions

- `template<typename I , typename W >  
mln::trait::concrete< I >::ret structural (const Image< I > &input, const Window< W > &win)`

*Approximate of morphological structural opening.*

### 9.120.1 Detailed Description

Namespace of approximate mathematical morphology opening routines.

### 9.120.2 Function Documentation

**9.120.2.1** `template<typename I , typename W > mln::trait::concrete< I >::ret  
mln::morpho::opening::approx::structural ( const Image< I > & input, const  
Window< W > & win ) [inline]`

Approximate of morphological structural opening.

This operator is  $d_{\{-B\}} \circ e_B$ .

References `mln::morpho::dilation()`, `mln::morpho::erosion()`, and `mln::win::sym()`.

## 9.121 mln::morpho::reconstruction Namespace Reference

Namespace of morphological reconstruction routines.

### Namespaces

- namespace [by\\_dilation](#)  
*Namespace of morphological reconstruction by dilation routines.*
- namespace [by\\_erosion](#)  
*Namespace of morphological reconstruction by erosion routines.*

### 9.121.1 Detailed Description

Namespace of morphological reconstruction routines.

## 9.122 mln::morpho::reconstruction::by\_dilation Namespace Reference

Namespace of morphological reconstruction by dilation routines.

### 9.122.1 Detailed Description

Namespace of morphological reconstruction by dilation routines.

## 9.123 mln::morpho::reconstruction::by\_erosion Namespace Reference

Namespace of morphological reconstruction by erosion routines.

### 9.123.1 Detailed Description

Namespace of morphological reconstruction by erosion routines.

## 9.124 mln::morpho::tree Namespace Reference

Namespace of morphological tree-related routines.

### Namespaces

- namespace [filter](#)  
*Namespace for attribute filtering.*

### Functions

- `template<typename A , typename T >`  
`mln::trait::ch_value< typename T::function, typename A::result >::ret compute\_attribute\_image`  
`(const Accumulator< A > &a, const T &t, mln::trait::ch_value< typename T::function, A >::ret`  
`*accu_image=0)`  
*Compute an attribute image using tree with a parent relationship between sites.*
- `template<typename A , typename T , typename V >`  
`mln::trait::ch_value< typename T::function, typename A::result >::ret compute\_attribute\_image\_`  
`from (const Accumulator< A > &a, const T &t, const Image< V > &values, mln::trait::ch_value<`  
`typename T::function, A >::ret *accu_image=0)`  
*The same as `compute_attribute_image` but uses the values stored by `values` image instead.*
- `template<typename I , typename N , typename S >`  
`mln::trait::ch_value< I, typename I::psite >::ret compute\_parent (const Image< I > &f, const`  
`Neighborhood< N > &nbh, const Site\_Set< S > &s)`  
*Compute a tree with a parent relationship between sites.*
- `template<typename I , typename N >`  
`data< I, p\_array< typename I::psite > > dual\_input\_max\_tree (const Image< I > &f, const Image<`  
`I > &m, const Neighborhood< N > &nbh)`  
*Compute the dual input max tree using mask-based connectivity.*
- `template<typename I , typename N >`  
`data< I, p\_array< typename I::psite > > max\_tree (const Image< I > &f, const Neighborhood< N`  
`> &nbh)`  
*Compute a canonized max-tree.*
- `template<typename I , typename N >`  
`data< I, p\_array< typename I::psite > > min\_tree (const Image< I > &f, const Neighborhood< N`  
`> &nbh)`  
*Compute a canonized min-tree.*

- `template<typename T, typename A, typename P, typename W >`  
`void propagate_if (const T &tree, Image< A > &a_, const way_of_propagation< W > &prop_,`  
`const Function_v2b< P > &pred_, const typename A::value &v)`
- `template<typename T, typename A, typename P >`  
`void propagate_if (const T &tree, Image< A > &a_, const desc_propagation &prop_, const`  
`Function_v2b< P > &pred_)`
- `template<typename T, typename A, typename W >`  
`void propagate_if_value (const T &tree, Image< A > &a_, const way_of_propagation< W >`  
`&prop_, const typename A::value &v)`
- `template<typename T, typename A, typename W >`  
`void propagate_if_value (const T &tree, Image< A > &a_, const way_of_propagation< W >`  
`&prop_, const typename A::value &v, const typename A::value &v_prop)`
- `template<typename T, typename A >`  
`void propagate_node_to_ancestors (typename A::psite n, const T &t, Image< A > &a_, const type-`  
`name A::value &v)`
- `template<typename T, typename A >`  
`void propagate_node_to_ancestors (typename A::psite n, const T &t, Image< A > &a_)`
- `template<typename T, typename A >`  
`void propagate_node_to_descendants (typename A::psite n, const T &t, Image< A > &a_, const`  
`typename A::value &v, unsigned *nb_leaves=0)`
- `template<typename T, typename A >`  
`void propagate_node_to_descendants (typename A::psite &n, const T &t, Image< A > &a_, un-`  
`signed *nb_leaves=0)`
- `template<typename T, typename F >`  
`void propagate_representative (const T &t, Image< F > &f_)`

*Propagate the representative node's value to non-representative points of the component.*

### 9.124.1 Detailed Description

Namespace of morphological tree-related routines.

### 9.124.2 Function Documentation

**9.124.2.1** `template<typename A, typename T > mln::trait::ch_value< typename T::function,`  
`typename A::result >::ret mln::morpho::tree::compute_attribute_image ( const`  
`Accumulator< A > & a, const T & t, mln::trait::ch_value< typename T::function, A`  
`>::ret * accu_image = 0 ) [inline]`

Compute an attribute image using tree with a parent relationship between sites.

In the attribute image, the resulting value at a node is the 'sum' of its sub-components value + the attribute value at this node.

Warning: `s` translates the ordering related to the "natural" childhood relationship. The parenthood is thus inverted w.r.t. to `s`.

It is very convenient since all processing upon the parent tree are performed following `s` (in the default "forward" way).

FIXME: Put it more clearly...

The parent result image verifies:

- $p$  is root iff  $\text{parent}(p) == p$
- $p$  is a node iff either  $p$  is root or  $f(\text{parent}(p)) != f(p)$ .

### Parameters

- [in]  $a$  Attribute.
- [in]  $t$  Component tree.
- [out]  $\text{accu\_image}$  Optional argument used to store image of attribute accumulator.

### Returns

The attribute image.

Referenced by `compute_attribute_image_from()`.

**9.124.2.2** `template<typename A , typename T , typename V > mln::trait::ch_value< typename T::function, typename A::result >::ret mln::morpho::tree::compute_attribute_image_from ( const Accumulator< A > & a, const T & t, const Image< V > & values, mln::trait::ch_value< typename T::function, A >::ret * accu_image = 0 ) [inline]`

The same as `compute_attribute_image` but uses the values stored by `values` image instead.

### Parameters

- [in]  $a$  Attribute.
- [in]  $t$  Component tree.
- [in]  $values$  Value image.
- [out]  $\text{accu\_image}$  Optional argument used to store image.

### Returns

References `compute_attribute_image()`.

**9.124.2.3** `template<typename I , typename N , typename S > mln::trait::ch_value< I, typename I::psite >::ret mln::morpho::tree::compute_parent ( const Image< I > & f, const Neighborhood< N > & nbh, const Site_Set< S > & s ) [inline]`

Compute a tree with a parent relationship between sites.

Warning:  $s$  translates the ordering related to the "natural" childhood relationship. The parenthood is thus inverted w.r.t. to  $s$ .

It is very convenient since most processing routines upon the parent tree are performed following  $s$  (in the default "forward" way). Indeed that is the way to propagate information from parents to children.

The parent result image verifies:

- $p$  is root iff  $\text{parent}(p) == p$
- $p$  is a node iff either  $p$  is root or  $f(\text{parent}(p)) != f(p)$ .

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 < .
```

**9.124.2.4** `template<typename I, typename N > morpho::tree::data< I, p_array< typename I::psite > > mln::morpho::tree::dual_input_max_tree ( const Image< I > & f, const Image< I > & m, const Neighborhood< N > & nbh ) [inline]`

Compute the dual input max tree using mask-based connectivity.

#### Parameters

[in] *f* The original image.

[in] *m* The connectivity mask.

[in] *nbh* The neighborhood of the mask.

### Returns

The computed tree.

**9.124.2.5** `template<typename I, typename N > data< I, p_array< typename I::psite > >  
mln::morpho::tree::max_tree ( const Image< I > & f, const Neighborhood< N > &  
nbh ) [inline]`

Compute a canonized max-tree.

### Parameters

[in] *f* The input image.

[in] *nbh* The neighborhood.

### Returns

The corresponding max-tree structure.

References mln::data::sort\_psites\_increasing().

**9.124.2.6** `template<typename I, typename N > data< I, p_array< typename I::psite > >  
mln::morpho::tree::min_tree ( const Image< I > & f, const Neighborhood< N > &  
nbh ) [inline]`

Compute a canonized min-tree.

### Parameters

[in] *f* The input image.

[in] *nbh* The neighborhood.

### Returns

The corresponding min-tree structure.

References mln::data::sort\_psites\_decreasing().

**9.124.2.7** `template<typename T, typename A, typename P, typename W > void  
mln::morpho::tree::propagate_if ( const T & tree, Image< A > & a_, const  
way_of_propagation< W > & prop_, const Function_v2b< P > & pred_, const  
typename A::value & v ) [inline]`

Propagate nodes checking the predicate *pred* in the way defined by *way\_of\_propagation*.

### Parameters

*tree* Component tree used for propagation.



- a\_* Attributed image where values are propagated.
- prop\_* Propagate node in ascendant or descendant way.
- pred\_* Predicate that node must check to be propagated.
- v* [Value](#) to be propagated. (By default *v* is the value at the node being propagated).

Referenced by `propagate_if()`, `propagate_if_value()`, and `mln::morpho::tree::filter::subtractive()`.

**9.124.2.8** `template<typename T , typename A , typename P > void  
mln::morpho::tree::propagate_if ( const T & tree, Image< A > & a_, const  
desc_propagation & prop_, const Function_v2b< P > & pred_ ) [inline]`

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

References `propagate_if()`.

**9.124.2.9** `template<typename T , typename A , typename W > void  
mln::morpho::tree::propagate_if_value ( const T & tree, Image< A > & a_, const  
way_of_propagation< W > & prop_, const typename A::value & v ) [inline]`

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

References `propagate_if()`.

**9.124.2.10** `template<typename T , typename A , typename W > void  
mln::morpho::tree::propagate_if_value ( const T & tree, Image< A > & a_, const  
way_of_propagation< W > & prop_, const typename A::value & v, const typename  
A::value & v_prop ) [inline]`

Propagate nodes having the value *v* in the way defined by `way_of_propagation`.

#### Parameters

- tree* Component tree used for propagation.
- a\_* Attributed image where values are propagated.
- prop\_* Propagate node in ascendant or descendant way.
- v* [Value](#) that node must have to be propagated.
- v\_prop* [Value](#) to propagate (By default it is the value at the node being propagated).

References `propagate_if()`.

**9.124.2.11** `template<typename T , typename A > void mln::morpho::tree::propagate_node_to_-  
ancestors ( typename A::psite n, const T & t, Image< A > & a_, const typename  
A::value & v )`

Propagate a value *v* from a node *n* to its ancestors.

#### Parameters

- [in] *n* Node to propagate.

- [in] *t* Component tree used for propagation.
- [in] *a\_* Attribute image where values are propagated.
- [in] *v* [Value](#) to propagate.

Referenced by `propagate_node_to_ancestors()`.

**9.124.2.12** `template<typename T , typename A > void mln::morpho::tree::propagate_node_to_ancestors ( typename A::psite n, const T & t, Image< A > & a_ ) [inline]`

Propagate the node's value to its ancestors.

#### Parameters

- [in] *n* Node to propagate.
- [in] *t* Component tree used for propagation.
- [in, out] *a\_* Attribute image where values are propagated.

References `propagate_node_to_ancestors()`.

**9.124.2.13** `template<typename T , typename A > void mln::morpho::tree::propagate_node_to_descendants ( typename A::psite n, const T & t, Image< A > & a_, const typename A::value & v, unsigned * nb_leaves = 0 ) [inline]`

Propagate a value *v* from a node *n* to its descendants.

#### Parameters

- [in] *n* Node to propagate.
- [in] *t* Component tree used for propagation.
- [in] *a\_* Attribute image where values are propagated.
- [in] *v* [Value](#) to propagate.
- [out] *nb\_leaves* Optional. Store the number of leaves in the component.

**9.124.2.14** `template<typename T , typename A > void mln::morpho::tree::propagate_node_to_descendants ( typename A::psite & n, const T & t, Image< A > & a_, unsigned * nb_leaves = 0 ) [inline]`

Propagate the node's value to its descendants.

#### Parameters

- [in] *n* Node to propagate.
- [in] *t* Component tree used for propagation.
- [in] *a\_* Attribute image where values are propagated.
- [out] *nb\_leaves* Optional. Store the number of leaves in the component.

**9.124.2.15** `template<typename T , typename F > void mln::morpho::tree::propagate_  
representative ( const T & t, Image< F > & f_ ) [inline]`

Propagate the representative node's value to non-representative points of the component.

#### Parameters

- t* Component tree.
- f\_* Value image.

## 9.125 mln::morpho::tree::filter Namespace Reference

Namespace for attribute filtering.

### Functions

- `template<typename T , typename F , typename P >  
void direct (const T &tree, Image< F > &f_, const Function_v2b< P > &pred_)`  
*Direct non-pruning strategy.*
- `template<typename T , typename F , typename P >  
void filter (const T &tree, Image< F > &f_, const Function_v2b< P > &pred_, const typename  
F::value &v)`  
*Filter the image f\_ with a given value.*
- `template<typename T , typename F , typename P >  
void max (const T &tree, Image< F > &f_, const Function_v2b< P > &pred_)`  
*Max pruning strategy.*
- `template<typename T , typename F , typename P >  
void min (const T &tree, Image< F > &f_, const Function_v2b< P > &pred_)`  
*Min pruning strategy.*
- `template<typename T , typename F , typename P >  
void subtractive (const T &tree, Image< F > &f_, const Function_v2b< P > &pred_)`  
*Subtractive pruning strategy.*

### 9.125.1 Detailed Description

Namespace for attribute filtering.

### 9.125.2 Function Documentation

**9.125.2.1** `template<typename T , typename F , typename P > void  
mln::morpho::tree::filter::direct ( const T & tree, Image< F > & f_, const  
Function_v2b< P > & pred_ ) [inline]`

Direct non-pruning strategy.

A node is removed if it does not verify the predicate. The sub-components remain intact.

#### Parameters

- [in] *tree* Component tree.
- [out] *f\_* [Image](#) to filter.
- [in] *pred\_* Filtering criterion.

**9.125.2.2** `template<typename T, typename F, typename P > void mln::morpho::tree::filter::filter ( const T & tree, Image< F > & f_, const Function_v2b< P > & pred_, const typename F::value & v ) [inline]`

Filter the image *f\_* with a given value.

The sub-components of nodes that does not match the predicate *pred\_* are filled with the given value *v*.

#### Parameters

- tree* Component tree.
- f\_* [Image](#) function.
- pred\_* Predicate.
- v* [Value](#) to propagate.

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

**9.125.2.3** `template<typename T, typename F, typename P > void mln::morpho::tree::filter::max ( const T & tree, Image< F > & f_, const Function_v2b< P > & pred_ ) [inline]`

Max pruning strategy.

A node is removed iif all of its children are removed or if it does not verify the predicate *pred\_*.

#### Parameters

- [in] *tree* Component tree.
- [out] *f\_* [Image](#) to filter.
- [in] *pred\_* Filtering criterion.

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

**9.125.2.4** `template<typename T, typename F, typename P > void mln::morpho::tree::filter::min ( const T & tree, Image< F > & f_, const Function_v2b< P > & pred_ ) [inline]`

Min pruning strategy.

A node is removed iif its parent is removed or if it does not verify the predicate *pred\_*.

#### Parameters

- [in] *tree* Component tree.
- [out] *f\_* [Image](#) to filter.
- [in] *pred\_* Filtering criterion.

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

**9.125.2.5** `template<typename T , typename F , typename P > void  
mln::morpho::tree::filter::subtractive ( const T & tree, Image< F > & f_, const  
Function_v2b< P > & pred_ ) [inline]`

Subtractive pruning strategy.

The node is removed if it does not verify the predicate. The sub-components values are set to the value of the removed component.

#### Parameters

- [in] *tree* Component tree.
- [out] *f\_* [Image](#) to filter.
- [in] *pred\_* Filtering criterion.

References `mln::morpho::tree::propagate_if()`.

## 9.126 mln::morpho::watershed Namespace Reference

Namespace of morphological watershed routines.

### Namespaces

- namespace [watershed](#)  
*Namespace of morphological watershed routines implementations.*

### Functions

- `template<typename L , typename I , typename N >  
mln::trait::ch_value< I , L >::ret flooding (const Image< I > &input, const Neighborhood< N >  
&nbh, L &n_basins)  
Meyer's Watershed Transform (WST) algorithm.`
- `template<typename L , typename I , typename N >  
mln::trait::ch_value< I , L >::ret flooding (const Image< I > &input, const Neighborhood< N >  
&nbh)  
Meyer's Watershed Transform (WST) algorithm, with no count of basins.`
- `template<typename I , typename J >  
mln::trait::ch_value< I , value::rgb8 >::ret superpose (const Image< I > &input, const Image< J >  
&ws_ima)  
Convert an image to a rgb8 image and draw the watershed lines.`
- `template<typename I , typename J >  
mln::trait::ch_value< I , value::rgb8 >::ret superpose (const Image< I > &input_, const Image< J >  
&ws_ima_, const value::rgb8 &wsl_color)  
Convert an image to a rgb8 image and draw the watershed lines.`

- `template<class T >`  
`T::image_t topological (T &tree)`  
*Compute a toological watershed transform from tree.*

### 9.126.1 Detailed Description

Namespace of morphological watershed routines.

### 9.126.2 Function Documentation

- 9.126.2.1** `template<typename L , typename I , typename N > mln::trait::ch_value< I, L >::ret  
mln::morpho::watershed::flooding ( const Image< I > & input, const Neighborhood<  
N > & nbh, L & n_basins ) [inline]`

Meyer's Watershed Transform (WST) algorithm.

#### Parameters

- [in] *input* The input image.
- [in] *nbh* The connexity of markers.
- [out] *n\_basins* The number of basins.

- L is the type of labels, used to number the watershed itself (with the minimal value), and the basins.
- I is the exact type of the input image.
- N is the exact type of the neighborhood used to express *input's* connexity.

- 9.126.2.2** `template<typename L , typename I , typename N > mln::trait::ch_value< I, L >::ret  
mln::morpho::watershed::flooding ( const Image< I > & input, const Neighborhood<  
N > & nbh )`

Meyer's Watershed Transform (WST) algorithm, with no count of basins.

#### Parameters

- [in] *input* The input image.
- [in] *nbh* The connexity of markers.

- L is the type of labels, used to number the watershed itself (with the minimal value), and the basins.
- I is the exact type of the input image.
- N is the exact type of the neighborhood used to express *input's* connexity.

Note that the first parameter, L, is not automatically valued from the type of the actual argument during implicit instantiation: you have to explicitly pass this parameter at call sites.

**9.126.2.3** `template<typename I, typename J> mln::trait::ch_value< I, value::rgb8 >::ret  
mln::morpho::watershed::superpose ( const Image< I > & input, const Image< J > &  
ws_ima ) [inline]`

Convert an image to a rgb8 image and draw the watershed lines.

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

**9.126.2.4** `template<typename I, typename J> mln::trait::ch_value< I, value::rgb8 >::ret  
mln::morpho::watershed::superpose ( const Image< I > & input_, const Image< J >  
& ws_ima_, const value::rgb8 & wsl_color ) [inline]`

Convert an image to a rgb8 image and draw the watershed lines.

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

Referenced by superpose().

**9.126.2.5** `template<class T> T::image_t mln::morpho::watershed::topological ( T & tree )`

Compute a toological watershed transform from *tree*.

References mln::data::fill(), mln::p\_priority< P, Q >::front(), mln::initialize(), mln::p\_priority< P, Q >::pop(), mln::p\_priority< P, Q >::push(), and topological().

Referenced by topological().

## 9.127 mln::morpho::watershed::watershed Namespace Reference

Namespace of morphological watershed routines implementations.

### Namespaces

- namespace [generic](#)

*Namespace of morphological watershed routines generic implementations.*

### 9.127.1 Detailed Description

Namespace of morphological watershed routines implementations.

## 9.128 mln::morpho::watershed::watershed::generic Namespace Reference

Namespace of morphological watershed routines generic implementations.

### 9.128.1 Detailed Description

Namespace of morphological watershed routines generic implementations.

## 9.129 mln::norm Namespace Reference

Namespace of norms.

### Namespaces

- namespace [impl](#)  
*Implementation namespace of norm namespace.*

### Functions

- `template<unsigned n, typename C >`  
`mln::trait::value_< typename mln::trait::op::times< C, C >::ret >::sum l2 (const C(&vec)[n])`  
*L2-norm of a vector vec.*
- `template<unsigned n, typename C >`  
`mln::trait::value_< typename mln::trait::op::times< C, C >::ret >::sum l1 (const C(&vec)[n])`  
*L1-norm of a vector vec.*
- `template<unsigned n, typename C >`  
`mln::trait::value_< typename mln::trait::op::times< C, C >::ret >::sum l1\_distance (const C(&vec1)[n], const C(&vec2)[n])`  
*L1-norm distance between vectors vec1 and vec2.*
- `template<unsigned n, typename C >`  
`mln::trait::value_< typename mln::trait::op::times< C, C >::ret >::sum sqr\_l2 (const C(&vec)[n])`  
*Squared L2-norm of a vector vec.*
- `template<unsigned n, typename C >`  
`mln::trait::value_< typename mln::trait::op::times< C, C >::ret >::sum l2\_distance (const C(&vec1)[n], const C(&vec2)[n])`  
*L2-norm distance between vectors vec1 and vec2.*
- `template<unsigned n, typename C >`  
`C lifty (const C(&vec)[n])`  
*L-infinity-norm of a vector vec.*
- `template<unsigned n, typename C >`  
`C lifty\_distance (const C(&vec1)[n], const C(&vec2)[n])`  
*L-infinity-norm distance between vectors vec1 and vec2.*



## 9.129.1 Detailed Description

Namespace of norms.

## 9.129.2 Function Documentation

**9.129.2.1** `template<unsigned n, typename C > mln::trait::value_< typename  
mln::trait::op::times< C, C >::ret >::sum mln::norm::l1 ( const C(&) vec[n] )  
[inline]`

L1-norm of a vector *vec*.

**9.129.2.2** `template<unsigned n, typename C > mln::trait::value_< typename  
mln::trait::op::times< C, C >::ret >::sum mln::norm::l1_distance ( const C(&)  
vec1[n], const C(&) vec2[n] ) [inline]`

L1-norm distance between vectors *vec1* and *vec2*.

**9.129.2.3** `template<unsigned n, typename C > mln::trait::value_< typename  
mln::trait::op::times< C, C >::ret >::sum mln::norm::l2 ( const C(&) vec[n] )  
[inline]`

L2-norm of a vector *vec*.

**9.129.2.4** `template<unsigned n, typename C > mln::trait::value_< typename  
mln::trait::op::times< C, C >::ret >::sum mln::norm::l2_distance ( const C(&)  
vec1[n], const C(&) vec2[n] ) [inline]`

L2-norm distance between vectors *vec1* and *vec2*.

**9.129.2.5** `template<unsigned n, typename C > C mln::norm::linfty ( const C(&) vec[n] )  
[inline]`

L-infinity-norm of a vector *vec*.

**9.129.2.6** `template<unsigned n, typename C > C mln::norm::linfty_distance ( const C(&)  
vec1[n], const C(&) vec2[n] ) [inline]`

L-infinity-norm distance between vectors *vec1* and *vec2*.

**9.129.2.7** `template<unsigned n, typename C > mln::trait::value_< typename  
mln::trait::op::times< C, C >::ret >::sum mln::norm::sqr_l2 ( const C(&) vec[n] )  
[inline]`

Squared L2-norm of a vector *vec*.

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

## 9.130 mln::norm::impl Namespace Reference

Implementation namespace of norm namespace.

### 9.130.1 Detailed Description

Implementation namespace of norm namespace.

## 9.131 mln::opt Namespace Reference

Namespace of optional routines.

### Namespaces

- namespace [impl](#)  
*Implementation namespace of opt namespace.*

### Functions

- `template<typename I >`  
`I::rvalue at (const Image< I > &ima, def::coord ind)`  
*One dimension Read-only access to the ima value located at (ind).*
- `template<typename I >`  
`I::lvalue at (Image< I > &ima, def::coord ind)`  
*Read-write access to the ima value located at (ind).*
- `template<typename I >`  
`I::lvalue at (Image< I > &ima, def::coord row, def::coord col)`  
*Read-write access to the ima value located at (row, col).*
- `template<typename I >`  
`I::rvalue at (const Image< I > &ima, def::coord sli, def::coord row, def::coord col)`  
*Three dimensions Read-only access to the ima value located at (sli, row, col).*
- `template<typename I >`  
`I::rvalue at (const Image< I > &ima, def::coord row, def::coord col)`  
*Two dimensions Read-only access to the ima value located at (row, col).*
- `template<typename I >`  
`I::lvalue at (Image< I > &ima, def::coord sli, def::coord row, def::coord col)`  
*Read-write access to the ima value located at (sli, row, col).*

### 9.131.1 Detailed Description

Namespace of optional routines.

## 9.131.2 Function Documentation

**9.131.2.1** `template<typename I> I::rvalue mln::opt::at ( const Image< I> & ima, def::coord ind ) [inline]`

One dimension Read-only access to the `ima` value located at (`ind`).

Referenced by `mln::transform::hough()`, and `mln::make::image()`.

**9.131.2.2** `template<typename I> I::lvalue mln::opt::at ( Image< I> & ima, def::coord ind )`

Read-write access to the `ima` value located at (`ind`).

**9.131.2.3** `template<typename I> I::lvalue mln::opt::at ( Image< I> & ima, def::coord row, def::coord col )`

Read-write access to the `ima` value located at (`row`, `col`).

**9.131.2.4** `template<typename I> I::rvalue mln::opt::at ( const Image< I> & ima, def::coord sli, def::coord row, def::coord col ) [inline]`

Three dimensions Read-only access to the `ima` value located at (`sli`, `row`, `col`).

**9.131.2.5** `template<typename I> I::rvalue mln::opt::at ( const Image< I> & ima, def::coord row, def::coord col ) [inline]`

Two dimensions Read-only access to the `ima` value located at (`row`, `col`).

**9.131.2.6** `template<typename I> I::lvalue mln::opt::at ( Image< I> & ima, def::coord sli, def::coord row, def::coord col )`

Read-write access to the `ima` value located at (`sli`, `row`, `col`).

## 9.132 mln::opt::impl Namespace Reference

Implementation namespace of `opt` namespace.

### 9.132.1 Detailed Description

Implementation namespace of `opt` namespace. Three dimensions.

Two dimensions.

One dimension.

## 9.133 mln::pw Namespace Reference

Namespace of "point-wise" expression tools.

## Classes

- class [image](#)  
*A generic point-wise image implementation.*

### 9.133.1 Detailed Description

Namespace of "point-wise" expression tools.

## 9.134 mln::registration Namespace Reference

Namespace of "point-wise" expression tools.

## Classes

- class [closest\\_point\\_basic](#)  
*Closest point functor based on map distance.*
- class [closest\\_point\\_with\\_map](#)  
*Closest point functor based on map distance.*

## Functions

- `template<typename P, typename F >  
algebra::quat get\_rot (const p\_array< P > &P_, const vec3d\_f &mu_P, const vec3d\_f &mu_Yk,  
const F &closest_point, const algebra::quat &qR, const vec3d\_f &qT)  
FIXME: work only for 3d images.`
- `template<typename P, typename F >  
std::pair< algebra::quat, mln\_vec(P)> icp (const p\_array< P > &P_, const p\_array< P > &X, const  
F &closest_point, const algebra::quat &initial_rot, const mln\_vec(P)&initial_translation)  
Base version of the ICP algorithm. It is called in other variants.`
- `template<typename P, typename F >  
composed< translation< P::dim, float >, rotation< P::dim, float > > icp (const p\_array< P > &P_,  
const p\_array< P > &X, const F &closest_point)`
- `template<typename P >  
composed< translation< P::dim, float >, rotation< P::dim, float > > registration1 (const box< P  
> &domain, const p\_array< P > &P_, const p\_array< P > &X)  
Call ICP once and return the resulting transformation.`
- `template<typename P >  
composed< translation< P::dim, float >, rotation< P::dim, float > > registration2 (const box< P  
> &domain, const p\_array< P > &P_, const p\_array< P > &X)  
Call ICP 10 times.`

- `template<typename P >`  
`composed< translation< P::dim, float >, rotation< P::dim, float > > registration3 (const box< P`  
`> &domain, const p_array< P > &P_, const p_array< P > &X)`  
*Call ICP 10 times.*

### 9.134.1 Detailed Description

Namespace of "point-wise" expression tools.

### 9.134.2 Function Documentation

- 9.134.2.1** `template<typename P , typename F > algebra::quat mln::registration::get_rot ( const`  
`p_array< P > & P_, const vec3d_f & mu_P, const vec3d_f & mu_Yk, const F &`  
`closest_point, const algebra::quat & qR, const vec3d_f & qT )`

FIXME: work only for 3d images.

References `mln::p_array< P >::nsites()`.

- 9.134.2.2** `template<typename P , typename F > std::pair< algebra::quat, mln_vec(P)>`  
`mln::registration::icp ( const p_array< P > & P_, const p_array< P > & X, const F &`  
`closest_point, const algebra::quat & initial_rot, const mln_vec(P)& initial_translation )`  
`[inline]`

Base version of the ICP algorithm. It is called in other variants.

Register point in `c` using a function of closest points `closest_point`. This overload allows to specify initial transformations.

#### Parameters

- [in] `P_` The cloud of points.
- [in] `X` the reference surface.
- [in] `closest_point` The function of closest points.
- [in] `initial_rot` An initial rotation.
- [in] `initial_translation` An initial translation.

#### Returns

the rigid transformation which may be use later to create a registered image.

WARNING: the function `closest_point` \*MUST\* take float/double vector as arguments. Otherwise the resulting transformation may be wrong due to the truncation of the vector coordinate values.

#### Precondition

`P_` and `X` must not be empty.

Reference article: "A Method for Registration of 3-D Shapes", Paul J. Besl and Neil D. McKay, IEEE, 2, February 1992.

References `mln::geom::bbox()`, `mln::literal::black`, `mln::set::compute()`, `mln::duplicate()`, `mln::box< P >::enlarge()`, `mln::data::fill()`, `mln::literal::green`, `mln::io::ppm::save()`, and `mln::literal::white`.

**9.134.2.3** `template<typename P , typename F > composed< translation<P::dim,float>,rotation<P::dim,float> > mln::registration::icp ( const p_array< P > & P_, const p_array< P > & X, const F & closest_point )`

Register point in `c` using a function of closest points `closest_point`.

#### Parameters

- [in] `P_` The cloud of points.
- [in] `X` the reference surface.
- [in] `closest_point` The function of closest points.

#### Returns

the rigid transformation which may be use later to create a registered image.

**9.134.2.4** `template<typename P > composed< translation< P::dim, float >, rotation< P::dim, float > > mln::registration::registration1 ( const box< P > & domain, const p_array< P > & P_, const p_array< P > & X ) [inline]`

Call ICP once and return the resulting transformation.

**9.134.2.5** `template<typename P > composed< translation< P::dim, float >, rotation< P::dim, float > > mln::registration::registration2 ( const box< P > & domain, const p_array< P > & P_, const p_array< P > & X ) [inline]`

Call ICP 10 times.

Do the first call to ICP with all sites then work on a subset of which size is decreasing. For each call, a distance criterion is computed on a subset. Sites part of the subset which are too far or too close are removed. Removed sites are *\*NOT\** reused later in the subset.

**9.134.2.6** `template<typename P > composed< translation< P::dim, float >, rotation< P::dim, float > > mln::registration::registration3 ( const box< P > & domain, const p_array< P > & P_, const p_array< P > & X ) [inline]`

Call ICP 10 times.

Do the first call to ICP with all sites then work on a subset. For each call, a distance criterion is computed on a subset. A new subset is computed from the whole set of points according to this distance. It will be used in the next call. Removed Sites *\*MAY\** be reintegrated.

## 9.135 mln::select Namespace Reference

Select namespace (FIXME doc).

### Classes

- struct `p_of`  
*Structure `p_of`.*

### 9.135.1 Detailed Description

Select namespace (FIXME doc).

## 9.136 mln::set Namespace Reference

Namespace of image processing routines related to pixel sets.

### Functions

- `template<typename S >`  
`unsigned card (const Site_Set< S > &s)`  
*Compute the cardinality of the site set *s*.*
- `template<typename A , typename S >`  
`A::result compute (const Accumulator< A > &a, const Site_Set< S > &s)`  
*Compute an accumulator onto a site set.*
- `template<typename A , typename I >`  
`A::result compute_with_weights (const Accumulator< A > &a, const Image< I > &w)`  
*Compute an accumulator on a site set described by an image.*
- `template<typename S >`  
`S::site get (const Site_Set< S > &s, size_t index)`  
*FIXME.*
- `template<typename S >`  
`bool has (const Site_Set< S > &s, const typename S::site &e)`  
*FIXME.*
- `template<typename A , typename I >`  
`mln_meta_accu_result (A, typename I::site) compute_with_weights(const Meta_Accumulator< A > &a`  
`> &a`  
*Compute an accumulator on a site set described by an image.*
- `template<typename A , typename S >`  
`mln_meta_accu_result (A, typename S::site) compute(const Meta_Accumulator< A > &a`  
`> &a`  
*Compute an accumulator onto a site set.*

### 9.136.1 Detailed Description

Namespace of image processing routines related to pixel sets.

## 9.136.2 Function Documentation

**9.136.2.1** `template<typename S > unsigned mln::set::card ( const Site_Set< S > & s )`  
`[inline]`

Compute the cardinality of the site set *s*.

**9.136.2.2** `template<typename A , typename S > A::result mln::set::compute ( const`  
`Accumulator< A > & a, const Site_Set< S > & s ) [inline]`

Compute an accumulator onto a site set.

### Parameters

[in] *a* An accumulator.

[in] *s* A site set.

### Returns

The accumulator result.

Referenced by `mln::registration::icp()`.

**9.136.2.3** `template<typename A , typename I > A::result mln::set::compute_with_weights ( const`  
`Accumulator< A > & a, const Image< I > & w ) [inline]`

Compute an accumulator on a site set described by an image.

### Parameters

[in] *a* An accumulator.

[in] *w* An image of weights (a site -> a weight).

### Returns

The accumulator result.

**9.136.2.4** `template<typename S > S::site mln::set::get ( const Site_Set< S > & s, size_t index )`

FIXME.

**9.136.2.5** `template<typename S > bool mln::set::has ( const Site_Set< S > & s, const typename`  
`S::site & e )`

FIXME.

**9.136.2.6** `template<typename A , typename I > mln::set::mln_meta_accu_result ( A , typename`  
`I::site ) const [inline]`

Compute an accumulator on a site set described by an image.



**Parameters**

- [in] *a* A meta-accumulator.
- [in] *w* An image of weights (a site -> a weight).

**Returns**

The accumulator result.

### 9.136.2.7 `template<typename A , typename S > mln::set::mln_meta_accu_result ( A , typename S::site ) const`

Compute an accumulator onto a site set.

**Parameters**

- [in] *a* A meta-accumulator.
- [in] *s* A site set.

## 9.137 mln::subsampling Namespace Reference

Namespace of "point-wise" expression tools.

**Functions**

- `template<typename I > mln::trait::concrete< I >::ret antialiased (const Image< I > &input, unsigned factor, const typename I::domain_t &output_domain, unsigned border_thickness)`  
*Antialiased subsampling.*
- `template<typename I > mln::trait::concrete< I >::ret antialiased (const Image< I > &input, unsigned factor)`
- `template<typename I > mln::trait::concrete< I >::ret gaussian_subsampling (const Image< I > &input, float sigma, const typename I::dpsite &first_p, const typename I::site::coord &gap)`  
*Gaussian subsampling FIXME : doxy.*
- `template<typename I > mln::trait::concrete< I >::ret subsampling (const Image< I > &input, const typename I::site::delta &first_p, const typename I::site::coord &gap)`  
*Subsampling FIXME : doxy.*

### 9.137.1 Detailed Description

Namespace of "point-wise" expression tools.

## 9.137.2 Function Documentation

**9.137.2.1** `template<typename I > mln::trait::concrete< I >::ret mln::subsampling::antialiased ( const Image< I > & input, unsigned factor, const typename I::domain_t & output_domain, unsigned border_thickness ) [inline]`

Antialiased subsampling.

### Parameters

- [in] *input* A gray-level image.
- [in] *factor* Subsampling ratio. Must be divisible by 2 or 3.
- [in] *output\_domain* Force output domain.
- [in] *border\_thickness* Force output border thickness.

Referenced by `antialiased()`.

**9.137.2.2** `template<typename I > mln::trait::concrete< I >::ret mln::subsampling::antialiased ( const Image< I > & input, unsigned factor ) [inline]`

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

References `antialiased()`.

**9.137.2.3** `template<typename I > mln::trait::concrete< I >::ret mln::subsampling::gaussian_subsampling ( const Image< I > & input, float sigma, const typename I::dpsite & first_p, const typename I::site::coord & gap ) [inline]`

Gaussian subsampling FIXME : doxy.

References `mln::linear::gaussian()`, `mln::geom::ncols()`, and `mln::geom::nrows()`.

**9.137.2.4** `template<typename I > mln::trait::concrete< I >::ret mln::subsampling::subsampling ( const Image< I > & input, const typename I::site::delta & first_p, const typename I::site::coord & gap ) [inline]`

Subsampling FIXME : doxy.

References `mln::geom::ncols()`, and `mln::geom::nrows()`.

## 9.138 mln::tag Namespace Reference

Namespace of image processing routines related to tags.

### 9.138.1 Detailed Description

Namespace of image processing routines related to tags.

## 9.139 mln::test Namespace Reference

Namespace of image processing routines related to pixel tests.

### Namespaces

- namespace [impl](#)

*Implementation namespace of test namespace.*

### Functions

- `template<typename I >`  
`bool positive (const Image< I > &input)`  
*Test if an image only contains positive values.*
- `template<typename S , typename F >`  
`bool predicate (const Site\_Set< S > &pset, const Function\_v2b< F > &f)`  
*Test if all points of pset verify the predicate f.*
- `template<typename I , typename J , typename F >`  
`bool predicate (const Image< I > &lhs, const Image< J > &rhs, const Function\_vv2b< F > &f)`  
*Test if all pixel values of lhs and rhs verify the predicate f.*
- `template<typename I , typename F >`  
`bool predicate (const Image< I > &ima, const Function\_v2b< F > &f)`  
*Test if all pixel values of ima verify the predicate f.*

### 9.139.1 Detailed Description

Namespace of image processing routines related to pixel tests.

### 9.139.2 Function Documentation

#### 9.139.2.1 `template<typename I > bool mln::test::positive ( const Image< I > & input )` `[inline]`

Test if an image only contains positive values.

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()`.

**9.139.2.2** `template<typename S, typename F> bool mln::test::predicate ( const Site_Set< S > & pset, const Function_v2b< F > & f ) [inline]`

Test if all points of `pset` verify the predicate `f`.

#### Parameters

[in] *pset* The point set.

[in] *f* The predicate.

**9.139.2.3** `template<typename I, typename J, typename F> bool mln::test::predicate ( const Image< I > & lhs, const Image< J > & rhs, const Function_vv2b< F > & f ) [inline]`

Test if all pixel values of `lhs` and `rhs` verify the predicate `f`.

#### Parameters

[in] *lhs* The image.

[in] *rhs* The image.

[in] *f* The predicate.

**9.139.2.4** `template<typename I, typename F> bool mln::test::predicate ( const Image< I > & ima, const Function_v2b< F > & f ) [inline]`

Test if all pixel values of `ima` verify the predicate `f`.

#### Parameters

[in] *ima* The image.

[in] *f* The predicate.

Referenced by `mln::operator<()`, `mln::operator<=()`, `mln::operator==()`, and `positive()`.

## 9.140 mln::test::impl Namespace Reference

Implementation namespace of test namespace.

### 9.140.1 Detailed Description

Implementation namespace of test namespace.

## 9.141 mln::topo Namespace Reference

Namespace of "point-wise" expression tools.

## Classes

- class [adj\\_higher\\_dim\\_connected\\_n\\_face\\_bkd\\_iter](#)  
*Backward iterator on all the  $n$ -faces sharing an adjacent  $(n+1)$ -face with a (reference)  $n$ -face of an `mln::complex<D>`.*
- class [adj\\_higher\\_dim\\_connected\\_n\\_face\\_fwd\\_iter](#)  
*Forward iterator on all the  $n$ -faces sharing an adjacent  $(n+1)$ -face with a (reference)  $n$ -face of an `mln::complex<D>`.*
- class [adj\\_higher\\_face\\_bkd\\_iter](#)  
*Backward iterator on all the adjacent  $(n+1)$ -faces of the  $n$ -face of an `mln::complex<D>`.*
- class [adj\\_higher\\_face\\_fwd\\_iter](#)  
*Forward iterator on all the adjacent  $(n+1)$ -faces of the  $n$ -face of an `mln::complex<D>`.*
- class [adj\\_lower\\_dim\\_connected\\_n\\_face\\_bkd\\_iter](#)  
*Backward iterator on all the  $n$ -faces sharing an adjacent  $(n-1)$ -face with a (reference)  $n$ -face of an `mln::complex<D>`.*
- class [adj\\_lower\\_dim\\_connected\\_n\\_face\\_fwd\\_iter](#)  
*Forward iterator on all the  $n$ -faces sharing an adjacent  $(n-1)$ -face with a (reference)  $n$ -face of an `mln::complex<D>`.*
- class [adj\\_lower\\_face\\_bkd\\_iter](#)  
*Backward iterator on all the adjacent  $(n-1)$ -faces of the  $n$ -face of an `mln::complex<D>`.*
- class [adj\\_lower\\_face\\_fwd\\_iter](#)  
*Forward iterator on all the adjacent  $(n-1)$ -faces of the  $n$ -face of an `mln::complex<D>`.*
- class [adj\\_lower\\_higher\\_face\\_bkd\\_iter](#)  
*Forward iterator on all the adjacent  $(n-1)$ -faces and  $(n+1)$ -faces of the  $n$ -face of an `mln::complex<D>`.*
- class [adj\\_lower\\_higher\\_face\\_fwd\\_iter](#)  
*Forward iterator on all the adjacent  $(n-1)$ -faces and  $(n+1)$ -faces of the  $n$ -face of an `mln::complex<D>`.*
- class [adj\\_m\\_face\\_bkd\\_iter](#)  
*Backward iterator on all the  $m$ -faces transitively adjacent to a (reference)  $n$ -face in a complex.*
- class [adj\\_m\\_face\\_fwd\\_iter](#)  
*Forward iterator on all the  $m$ -faces transitively adjacent to a (reference)  $n$ -face in a complex.*
- 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_site< D, G > &f)`  
*Is f a facet, i.e., a face not “included in” (adjacent to) a face of higher dimension?*
- `template<unsigned D>`  
`algebraic_face< D > make_algebraic_face (const face< D > &f, bool sign)`  
*Create an algebraic face handle of a D-complex.*
- `template<unsigned D>`  
`std::ostream & operator<< (std::ostream &ostr, const face< D > &f)`  
*Print an mln::topo::face.*
- `template<unsigned N, unsigned D>`  
`std::ostream & operator<< (std::ostream &ostr, const n_face< N, D > &f)`  
*Print an mln::topo::n\_face.*
- `template<unsigned D>`  
`std::ostream & operator<< (std::ostream &ostr, const complex< D > &c)`  
*Pretty print a complex.*
- `template<unsigned D>`  
`std::ostream & operator<< (std::ostream &ostr, const algebraic_face< D > &f)`  
*Print an mln::topo::algebraic\_face.*
- `template<unsigned N, unsigned D>`  
`std::ostream & operator<< (std::ostream &ostr, const algebraic_n_face< N, D > &f)`  
*Print an mln::topo::algebraic\_n\_face.*
- `template<unsigned D>`  
`bool operator== (const complex< D > &lhs, const complex< D > &rhs)`  
*Compare two complexes for equality.*
- `template<unsigned D>`  
`algebraic_face< D > operator- (const face< D > &f)`  
*Inversion operators.*
- `template<unsigned D>`  
`bool operator== (const algebraic_face< D > &lhs, const algebraic_face< D > &rhs)`  
*Comparison of two instances of mln::topo::algebraic\_face.*
- `template<unsigned D>`  
`bool operator!= (const algebraic_face< D > &lhs, const algebraic_face< D > &rhs)`  
*Is lhs different from rhs?*
- `template<unsigned D>`  
`bool operator< (const algebraic_face< D > &lhs, const algebraic_face< D > &rhs)`  
*Is lhs “less” than rhs?*

- `template<unsigned N, unsigned D>`  
`algebraic_n_face< N, D > operator-` (const `n_face< N, D >` &f)  
*Inversion operators.*
  
- `template<unsigned N, unsigned D>`  
`bool operator==` (const `algebraic_n_face< N, D >` &lhs, const `algebraic_n_face< N, D >` &rhs)  
*Comparison of two instances of `mln::topo::algebraic_n_face`.*
  
- `template<unsigned N, unsigned D>`  
`bool operator!=` (const `algebraic_n_face< N, D >` &lhs, const `algebraic_n_face< N, D >` &rhs)  
*Is lhs different from rhs?*
  
- `template<unsigned N, unsigned D>`  
`bool operator<` (const `algebraic_n_face< N, D >` &lhs, const `algebraic_n_face< N, D >` &rhs)  
*Is lhs "less" than rhs?*
  
  
- `template<unsigned D>`  
`algebraic_n_face< 1, D > edge` (const `n_face< 0, D >` &f1, const `n_face< 0, D >` &f2)  
*Helpers.*
  
  
- `template<unsigned D>`  
`bool operator==` (const `face< D >` &lhs, const `face< D >` &rhs)  
*Comparison of two instances of `mln::topo::face`.*
  
- `template<unsigned D>`  
`bool operator!=` (const `face< D >` &lhs, const `face< D >` &rhs)  
*Is lhs different from rhs?*
  
- `template<unsigned D>`  
`bool operator<` (const `face< D >` &lhs, const `face< D >` &rhs)  
*Is lhs "less" than rhs?*
  
  
- `template<unsigned N, unsigned D>`  
`bool operator==` (const `n_face< N, D >` &lhs, const `n_face< N, D >` &rhs)  
*Comparison of two instances of `mln::topo::n_face`.*
  
- `template<unsigned N, unsigned D>`  
`bool operator!=` (const `n_face< N, D >` &lhs, const `n_face< N, D >` &rhs)  
*Is lhs different from rhs?*
  
- `template<unsigned N, unsigned D>`  
`bool operator<` (const `n_face< N, D >` &lhs, const `n_face< N, D >` &rhs)  
*Is lhs "less" than rhs?*
  
  
- `template<unsigned N, unsigned D>`  
`n_faces_set< N, D > operator+` (const `algebraic_n_face< N, D >` &f1, const `algebraic_n_face< N, D >` &f2)  
*Addition.*



- `template<unsigned N, unsigned D>`  
`n_faces_set< N, D > operator-` (const `algebraic_n_face< N, D >` &f1, const `algebraic_n_face<`  
`N, D >` &f2)  
*Subtraction.*

### 9.141.1 Detailed Description

Namespace of "point-wise" expression tools.

### 9.141.2 Function Documentation

**9.141.2.1** `template<unsigned D, typename G > void mln::topo::detach ( const complex_site< 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()`.

**9.141.2.2** `template<unsigned D> algebraic_n_face< 1, D > mln::topo::edge ( const n_face< 0, D`  
`> & f1, const n_face< 0, D > & f2 )`

Helpers.

Return the algebraic 1-face (edge) linking the 0-faces (vertices) *f1* and *f2*. If there is no 1-face between *f1* and *f2*, return an invalid 1-face.

#### Precondition

*f1* and *f2* must belong to the same complex.

Note: this routine assumes the complex is not degenerated, i.e.,

- it does not check that *f1* and *f2* are the only 0-faces adjacent to an hypothetical 1-face; it just checks that *f1* and *f2* share a common 1-face;
- if there are several adjacent 1-faces shared by *f1* and *f2* (if the complex is ill-formed), there is no guarantee on the returned 1-face (the current implementation return the first 1-face found, but client code should not rely on this implementation-defined behavior).

References `mln::topo::n_face< N, D >::higher_dim_adj_faces()`.

**9.141.2.3** `template<unsigned D, typename G > bool mln::topo::is_facet ( const complex_psite< D, G > & f ) [inline]`

Is *f* a facet, i.e., a face not “included in” (adjacent to) a face of higher dimension?

Referenced by `mln::make::attachment()`, `mln::make::cell()`, `detach()`, and `mln::make::detachment()`.

**9.141.2.4** `template<unsigned D> algebraic_face< D > mln::topo::make_algebraic_face ( const face< D > & f, bool sign )`

Create an algebraic face handle of a `D-complex`.

**9.141.2.5** `template<unsigned D> bool mln::topo::operator!= ( const algebraic_face< D > & lhs, const algebraic_face< D > & rhs ) [inline]`

Is *lhs* different from *rhs*?

#### Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

References `mln::topo::face< D >::cplx()`.

**9.141.2.6** `template<unsigned D> bool mln::topo::operator!= ( const face< D > & lhs, const face< D > & rhs ) [inline]`

Is *lhs* different from *rhs*?

#### Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

References `mln::topo::face< D >::cplx()`.

**9.141.2.7** `template<unsigned N, unsigned D> bool mln::topo::operator!= ( const algebraic_n_face< N, D > & lhs, const algebraic_n_face< N, D > & rhs ) [inline]`

Is *lhs* different from *rhs*?

#### Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

References `mln::topo::n_face< N, D >::cplx()`.

**9.141.2.8** `template<unsigned N, unsigned D> bool mln::topo::operator!= ( const n_face< N, D > & lhs, const n_face< N, D > & rhs ) [inline]`

Is *lhs* different from *rhs*?

#### Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

References `mln::topo::n_face< N, D >::cplx()`.

**9.141.2.9** `template<unsigned N, unsigned D> n_faces_set< N, D > mln::topo::operator+ ( const algebraic_n_face< N, D > & f1, const algebraic_n_face< N, D > & f2 ) [inline]`

Addition.

References `mln::topo::n_faces_set< N, D >::add()`.

**9.141.2.10** `template<unsigned N, unsigned D> algebraic_n_face< N, D > mln::topo::operator- ( const n_face< N, D > & f )`

Inversion operators.

**9.141.2.11** `template<unsigned N, unsigned D> n_faces_set< N, D > mln::topo::operator- ( const algebraic_n_face< N, D > & f1, const algebraic_n_face< N, D > & f2 ) [inline]`

Subtraction.

References `mln::topo::n_faces_set< N, D >::add()`.

**9.141.2.12** `template<unsigned D> algebraic_face< D > mln::topo::operator- ( const face< D > & f )`

Inversion operators.

**9.141.2.13** `template<unsigned N, unsigned D> bool mln::topo::operator< ( const n_face< N, D > & lhs, const n_face< N, D > & rhs ) [inline]`

Is *lhs* “less” than *rhs*?

This comparison is required by algorithms sorting face handles.

#### Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

**9.141.2.14** `template<unsigned D> bool mln::topo::operator< ( const face< D > & lhs, const face< D > & rhs ) [inline]`

Is *lhs* “less” than *rhs*?

This comparison is required by algorithms sorting face handles.

#### Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

Arguments *lhs* and *rhs* must have the same dimension.

**9.141.2.15** `template<unsigned D> bool mln::topo::operator< ( const algebraic_face< D > & lhs, const algebraic_face< D > & rhs ) [inline]`

Is *lhs* “less” than *rhs*?

This comparison is required by algorithms sorting algebraic face handles.

#### Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).  
Arguments *lhs* and *rhs* must have the same dimension.

**9.141.2.16** `template<unsigned N, unsigned D> bool mln::topo::operator<( const algebraic_n_face< N, D > & lhs, const algebraic_n_face< N, D > & rhs ) [inline]`

Is *lhs* “less” than *rhs*?

This comparison is required by algorithms sorting algebraic face handles.

#### Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

**9.141.2.17** `template<unsigned N, unsigned D> std::ostream & mln::topo::operator<<( std::ostream & ostr, const algebraic_n_face< N, D > & f ) [inline]`

Print an [mln::topo::algebraic\\_n\\_face](#).

**9.141.2.18** `template<unsigned D> std::ostream & mln::topo::operator<<( std::ostream & ostr, const face< D > & f ) [inline]`

Print an [mln::topo::face](#).

**9.141.2.19** `template<unsigned N, unsigned D> std::ostream & mln::topo::operator<<( std::ostream & ostr, const n_face< N, D > & f ) [inline]`

Print an [mln::topo::n\\_face](#).

**9.141.2.20** `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()`.

**9.141.2.21** `template<unsigned D> std::ostream & mln::topo::operator<<( std::ostream & ostr, const algebraic_face< D > & f ) [inline]`

Print an [mln::topo::algebraic\\_face](#).

**9.141.2.22** `template<unsigned N, unsigned D> bool mln::topo::operator==( const algebraic_n_face< N, D > & lhs, const algebraic_n_face< N, D > & rhs ) [inline]`

Comparison of two instances of [mln::topo::algebraic\\_n\\_face](#).

Is *lhs* equal to *rhs*?

#### Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

References [mln::topo::n\\_face< N, D >::cplx\(\)](#), [mln::topo::n\\_face< N, D >::face\\_id\(\)](#), and [mln::topo::algebraic\\_n\\_face< N, D >::sign\(\)](#).

**9.141.2.23** `template<unsigned D> bool mln::topo::operator==( const complex< D > & lhs, const complex< D > & rhs ) [inline]`

Compare two complexes for equality.

**9.141.2.24** `template<unsigned D> bool mln::topo::operator==( const face< D > & lhs, const face< D > & rhs ) [inline]`

Comparison of two instances of [mln::topo::face](#).

Is *lhs* equal to *rhs*?

#### Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

References [mln::topo::face< D >::cplx\(\)](#), [mln::topo::face< D >::face\\_id\(\)](#), and [mln::topo::face< D >::n\(\)](#).

**9.141.2.25** `template<unsigned D> bool mln::topo::operator==( const algebraic_face< D > & lhs, const algebraic_face< D > & rhs ) [inline]`

Comparison of two instances of [mln::topo::algebraic\\_face](#).

Is *lhs* equal to *rhs*?

#### Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

References [mln::topo::face< D >::cplx\(\)](#), [mln::topo::face< D >::face\\_id\(\)](#), [mln::topo::face< D >::n\(\)](#), and [mln::topo::algebraic\\_face< D >::sign\(\)](#).

**9.141.2.26** `template<unsigned N, unsigned D> bool mln::topo::operator==( const n_face< N, D > & lhs, const n_face< N, D > & rhs ) [inline]`

Comparison of two instances of [mln::topo::n\\_face](#).

Is *lhs* equal to *rhs*?

**Precondition**

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

References [mln::topo::n\\_face< N, D >::cplx\(\)](#), and [mln::topo::n\\_face< N, D >::face\\_id\(\)](#).

**9.142 mln::trace Namespace Reference**

Namespace of routines related to the trace mechanism.

**9.142.1 Detailed Description**

Namespace of routines related to the trace mechanism.

**9.143 mln::trait Namespace Reference**

Namespace where traits are defined.

**9.143.1 Detailed Description**

Namespace where traits are defined. Namespace for image traits.

**9.144 mln::transform Namespace Reference**

Namespace of transforms.

**Functions**

- `template<typename I , typename N , typename D >`  
[util::couple](#)< [mln::trait::ch\\_value](#)< I, D >::ret, [mln::trait::ch\\_value](#)< I, typename I::psite >::ret >  
[distance\\_and\\_closest\\_point\\_geodesic](#) (const [Image](#)< I > &input, const [Neighborhood](#)< N > &nbh,  
 D max)  
*Discrete geodesic distance transform.*
- `template<typename P , typename N , typename D >`  
[util::couple](#)< [mln\\_image\\_from\\_grid](#)([mln\\_grid](#)(P), D), [mln\\_image\\_from\\_grid](#)([mln\\_grid](#)(P),  
 unsigned)> [distance\\_and\\_closest\\_point\\_geodesic](#) (const [p\\_array](#)< P > &pset, const [box](#)< P  
 > &closest\_point\_domain, const [Neighborhood](#)< N > &nbh, D max)  
*Discrete geodesic distance transform.*
- `template<typename I , typename N , typename D >`  
[util::couple](#)< [mln::trait::ch\\_value](#)< I, D >::ret, I > [distance\\_and\\_influence\\_zone\\_geodesic](#) (const  
[Image](#)< I > &input, const [Neighborhood](#)< N > &nbh, D max)  
*Discrete geodesic distance transform.*

- `template<typename I , typename N , typename W , typename D >`  
`mln::trait::ch_value< I , D >::ret distance_front (const Image< I > &input, const Neighborhood< N > &nbh, const Weighted_Window< W > &w_win, D max)`  
*Discrete front distance transform.*
- `template<typename I , typename N , typename D >`  
`mln::trait::ch_value< I , D >::ret distance_geodesic (const Image< I > &input, const 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)`  
*Geodesic influence zone transform.*
- `template<typename I , typename N , typename D >`  
`mln::trait::concrete< I >::ret influence_zone_geodesic_saturated (const Image< I > &input, const Neighborhood< N > &nbh, const D &max, const typename I::value &background_value)`  
*Geodesic influence zone transform.*
- `template<typename I , typename N , typename D >`  
`mln::trait::concrete< I >::ret influence_zone_geodesic_saturated (const Image< I > &input, const Neighborhood< N > &nbh, const D &max)`

### 9.144.1 Detailed Description

Namespace of transforms.

### 9.144.2 Function Documentation

- 9.144.2.1** `template<typename I , typename N , typename D > util::couple<`  
`mln::trait::ch_value< I , D >::ret, mln::trait::ch_value< I , typename I::psite >::ret >`  
`mln::transform::distance_and_closest_point_geodesic ( const Image< I > & input,`  
`const Neighborhood< N > & nbh, D max ) [inline]`

Discrete geodesic distance transform.

**Parameters**

- [in] *input* [Image](#) from which the geodesic distance is computed.
- [in] *nbh* [Neighborhood](#)
- [in] *max* Max distance of propagation.

**Returns**

a couple of images. The first one is the distance map and the second one is the closest point image. The closest point image contains sites.

**Postcondition**

The returned images have the same domain as `input`.

References `mln::make::couple()`, and `distance_geodesic()`.

**9.144.2.2** `template<typename P , typename N , typename D > util::couple< mln_image_from_grid(mln_grid(P), D), mln_image_from_grid(mln_grid(P), unsigned)> mln::transform::distance_and_closest_point_geodesic ( const p_array< P > & pset, const box< P > & closest_point_domain, const Neighborhood< N > & nbh, D max ) [inline]`

Discrete geodesic distance transform.

**Parameters**

- [in] *pset* an array of sites.
- [in] *closest\_point\_domain* domain of the returned image.
- [in] *nbh* neighborhood
- [in] *max* max distance of propagation.

**Returns**

A couple of images. The first one is the distance map and the second one is the closest point image. The closest point image contains site indexes.

**Postcondition**

The returned image domains are defined on `closest_point_domain`.

References `mln::geom::bbox()`, `mln::make::couple()`, `distance_geodesic()`, `mln::data::fill()`, and `mln::box< P >::is_valid()`.

**9.144.2.3** `template<typename I , typename N , typename D > util::couple< mln::trait::ch_value< I, D >::ret, I > mln::transform::distance_and_influence_zone_geodesic ( const Image< I > & input, const Neighborhood< N > & nbh, D max ) [inline]`

Discrete geodesic distance transform.

**Parameters**

- [in] *input* [Image](#) from which the geodesic distance is computed.



[in] *nbh* Neighborhood  
 [in] *max* Max distance of propagation.

### Returns

a couple of images. The first one is the distance map and the second one is the closest point image. The closest point image contains sites.

### Postcondition

The returned images have the same domain as *input*.

References `mln::make::couple()`, and `distance_geodesic()`.

**9.144.2.4** `template<typename I, typename N, typename W, typename D > mln::trait::ch_value< I, D >::ret mln::transform::distance_front ( const Image< I > & input, const Neighborhood< N > & nbh, const Weighted_Window< W > & w_win, D max ) [inline]`

Discrete front distance transform.

**9.144.2.5** `template<typename I, typename N, typename D > mln::trait::ch_value< I, D >::ret mln::transform::distance_geodesic ( const Image< I > & input, const Neighborhood< N > & nbh, D max ) [inline]`

Discrete geodesic distance transform.

Referenced by `distance_and_closest_point_geodesic()`, and `distance_and_influence_zone_geodesic()`.

**9.144.2.6** `template<typename I > image2d< float > mln::transform::hough ( const Image< I > & input_ )`

Compute the hough transform from a binary image.

Objects used for computation must be set to 'true'.

### Parameters

[in] *input\_* A binary image.

### Returns

A 2D image of float. Rows are used for the distance and columns are used for the angles. Angles go from 0 to 359. Distance goes from 0 to the maximum distance between the center and a corner. The site having the maximum value indicates through its column index the document inclination.

References `mln::opt::at()`, `mln::data::fill()`, `mln::geom::min_col()`, `mln::geom::min_row()`, `mln::geom::ncols()`, and `mln::geom::nrows()`.

**9.144.2.7** `template<typename I, typename N, typename W > mln::trait::concrete< I >::ret mln::transform::influence_zone_front ( const Image< I > & input, const Neighborhood< N > & nbh, const Weighted_Window< W > & w_win )`

Influence zone transform.

References `influence_zone_front()`.

**9.144.2.8** `template<typename I , typename N , typename W , typename D > mln::trait::concrete< I >::ret mln::transform::influence_zone_front ( const Image< I > & input, const Neighborhood< N > & nbh, const Weighted_Window< W > & w_win, D max )`

Influence zone transform.

References `mln::canvas::distance_front()`.

Referenced by `influence_zone_front()`.

**9.144.2.9** `template<typename I , typename N > mln::trait::concrete< I >::ret mln::transform::influence_zone_geodesic ( const Image< I > & input, const Neighborhood< N > & nbh )`

Geodesic influence zone transform.

#### Parameters

[in] *input* An image.

[in] *nbh* A neighborhood.

#### Returns

An image of influence zone.

**9.144.2.10** `template<typename I , typename N , typename D > mln::trait::concrete< I >::ret mln::transform::influence_zone_geodesic_saturated ( const Image< I > & input, const Neighborhood< N > & nbh, const D & max, const typename I::value & background_value )`

Geodesic influence zone transform.

#### Parameters

[in] *input* An image.

[in] *nbh* A neighborhood.

[in] *max* The maximum influence zone distance.

[in] *background\_value* The value used as background (i.e. not propagated).

#### Returns

An image of influence zone.

References `mln::canvas::distance_geodesic()`.

Referenced by `influence_zone_geodesic_saturated()`.

**9.144.2.11** `template<typename I , typename N , typename D > mln::trait::concrete< I >::ret mln::transform::influence_zone_geodesic_saturated ( const Image< I > & input, const Neighborhood< N > & nbh, const D & max )`

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

References `influence_zone_geodesic_saturated()`, and `mln::literal::zero`.

## 9.145 mln::util Namespace Reference

Namespace of tools using for more complex algorithm.

### Namespaces

- namespace [impl](#)  
*Implementation namespace of util namespace.*

### Classes

- class [adjacency\\_matrix](#)  
*A class of adjacency matrix.*
- class [array](#)  
*A dynamic array class.*
- class [branch](#)  
*Class of generic branch.*
- class [branch\\_iter](#)  
*Basic 2D image class.*
- class [branch\\_iter\\_ind](#)  
*Basic 2D image class.*
- class [couple](#)  
*Definition of a couple.*
- struct [eat](#)  
*Eat structure.*
- class [edge](#)  
*Edge of a graph  $G$ .*
- class [fibonacci\\_heap](#)  
*Fibonacci heap.*
- class [graph](#)  
*Undirected graph.*
- class [greater\\_point](#)  
*A “greater than” functor comparing points w.r.t.*
- class [greater\\_psite](#)  
*A “greater than” functor comparing psites w.r.t.*
- class [head](#)

*Top structure of the soft heap.*

- struct [ignore](#)  
*Ignore structure.*
- struct [ilcell](#)  
*Element of an item list. Store the data (key) used in [soft\\_heap](#).*
- class [line\\_graph](#)  
*Undirected line graph of a graph of type  $G$ .*
- struct [nil](#)  
*Nil structure.*
- class [node](#)  
*Meta-data of an element in the heap.*
- class [object\\_id](#)  
*Base class of an object id.*
- struct [ord](#)  
*Function-object that defines an ordering between objects with type  $T$ : lhs  $R$  rhs.*
- struct [ord\\_pair](#)  
*Ordered pair structure s.a.*
- struct [pix](#)  
*Structure pix.*
- class [set](#)  
*An "efficient" mathematical set class.*
- class [site\\_pair](#)  
*A pair of sites.*
- class [soft\\_heap](#)  
*Soft heap.*
- class [timer](#)  
*Timer structure.*
- struct [tracked\\_ptr](#)  
*Smart pointer for shared data with tracking.*
- class [tree](#)  
*Class of generic tree.*
- class [tree\\_node](#)  
*Class of generic [tree\\_node](#) for tree.*

- class [vertex](#)  
*Vertex of a graph G.*
- struct [yes](#)  
*Object that always says "yes".*

## Typedefs

- typedef [object\\_id](#)< [vertex\\_tag](#), unsigned > [vertex\\_id\\_t](#)  
*Vertex id type.*

## Functions

- template<typename I, typename J >  
void [display\\_branch](#) (const [Image](#)< J > &ima\_, [tree\\_node](#)< I > \*tree\_node)  
*Display an arborescence from tree\_node.*
- template<typename I, typename J >  
void [display\\_tree](#) (const [Image](#)< J > &ima\_, [tree](#)< I > &tree)  
*Display a tree.*
- template<typename I >  
I::psite [lemmings](#) (const [Image](#)< I > &ima, const typename I::psite &pt, const typename I::psite::delta &dpt, const typename I::value &val)  
*Launch a lemmings on an image.*
- template<typename I >  
[greater\\_point](#)< I > [make\\_greater\\_point](#) (const [Image](#)< I > &ima)  
*Helper to build a mln::util::greater\_point.*
- template<typename I >  
[greater\\_psite](#)< I > [make\\_greater\\_psite](#) (const [Image](#)< I > &ima)  
*Helper to build a mln::util::greater\_psite.*
- template<typename G >  
bool [operator](#)<< (const [vertex](#)< G > &lhs, const [vertex](#)< G > &rhs)  
*Less operator. Test whether lhs.id() < rhs.id().*
- template<typename G >  
std::ostream & [operator](#)<< (std::ostream &ostr, const [vertex](#)< G > &v)  
*Push the vertex v in the output stream ostr.*
- template<typename T >  
std::ostream & [operator](#)<< (std::ostream &ostr, const [array](#)< T > &a)  
*Operator<<.*
- template<typename G >  
bool [operator](#)== (const [vertex](#)< G > &v1, const [vertex](#)< G > &v2)

*Equality operator.*

- `template<typename T >`  
`bool operator== (const array< T > &lhs, const array< T > &rhs)`  
*Operator==.*
- `template<typename T >`  
`bool ord_strict (const T &lhs, const T &rhs)`  
*Routine to test if lhs is strictly "less-than" rhs.*
- `template<typename T >`  
`bool ord_weak (const T &lhs, const T &rhs)`  
*Routine to test if lhs is "less-than or equal-to" rhs.*
- `template<typename T , typename I >`  
`void tree_fast_to_image (tree_fast< T > &tree, Image< I > &output_)`
- `template<typename T >`  
`tree_fast< T > tree_to_fast (tree< T > &input)`  
*Facade.*
- `template<typename T , typename I >`  
`void tree_to_image (tree< T > &tree, Image< I > &output_)`  
*Convert a tree into an image.*

### 9.145.1 Detailed Description

Namespace of tools using for more complex algorithm. Forward declaration.

### 9.145.2 Typedef Documentation

#### 9.145.2.1 typedef object\_id<vertex\_tag, unsigned> mln::util::vertex\_id\_t

[Vertex](#) id type.

### 9.145.3 Function Documentation

#### 9.145.3.1 `template<typename I , typename J > void mln::util::display_branch ( const Image< J > & ima_ , tree_node< I > * tree_node ) [inline]`

Display an arborescence from [tree\\_node](#).

#### Parameters

- [in] *ima\_* The domain of output image.
- [in] *tree\_node* The root [tree\\_node](#) to display.

References [mln::data::fill\(\)](#).

**9.145.3.2** `template<typename I, typename J> void mln::util::display_tree ( const Image< J > & ima_, tree< I > & tree ) [inline]`

Display a tree.

#### Parameters

[in] *ima\_* The domain of output image.

[in] *tree* The tree to display.

References `mln::util::tree< T >::root()`.

**9.145.3.3** `template<typename I> I::psite mln::util::lemmings ( const Image< I > & ima, const typename I::psite & pt, const typename I::psite::delta & dpt, const typename I::value & val )`

Launch a lemmings on an image.

A lemmings is the point `pt` that you put on an image `ima`. This point will move through the image using the delta-point `dpt` while consider his value on the given image.

#### Returns

The first point that is not in the domain `domain` or which value on the given image is different to the value `val`.

#### Precondition

The domain `domain` must be contained in the domain of `ima`.

**9.145.3.4** `template<typename I> greater_point< I > mln::util::make_greater_point ( const Image< I > & ima )`

Helper to build a [mln::util::greater\\_point](#).

**9.145.3.5** `template<typename I> greater_psite< I > mln::util::make_greater_psite ( const Image< I > & ima )`

Helper to build a [mln::util::greater\\_psite](#).

**9.145.3.6** `template<typename G> bool mln::util::operator< ( const vertex< G > & lhs, const vertex< G > & rhs ) [inline]`

Less operator. Test whether `lhs.id() < rhs.id()`.

**9.145.3.7** `template<typename G> std::ostream & mln::util::operator<<< ( std::ostream & ostr, const vertex< G > & v ) [inline]`

Push the vertex `v` in the output stream `ostr`.

**9.145.3.8** `template<typename T> std::ostream & mln::util::operator<< ( std::ostream & ostr, const array< T > & a )`

Operator<<.

References `mln::util::array< T >::nelements()`.

**9.145.3.9** `template<typename G> bool mln::util::operator==( const vertex< G > & v1, const vertex< G > & v2 ) [inline]`

Equality operator.

Test whether two vertices have the same id.

References `mln::util::vertex< G >::graph()`, and `mln::util::vertex< G >::id()`.

**9.145.3.10** `template<typename T> bool mln::util::operator==( const array< T > & lhs, const array< T > & rhs )`

Operator==.

References `mln::util::array< T >::std_vector()`.

**9.145.3.11** `template<typename T> bool mln::util::ord_strict ( const T & lhs, const T & rhs ) [inline]`

Routine to test if *lhs* is strictly "less-than" *rhs*.

Referenced by `mln::util::ord_pair< T >::change_both()`, `mln::util::ord_pair< T >::change_first()`, and `mln::util::ord_pair< T >::change_second()`.

**9.145.3.12** `template<typename T> bool mln::util::ord_weak ( const T & lhs, const T & rhs ) [inline]`

Routine to test if *lhs* is "less-than or equal-to" *rhs*.

Referenced by `mln::util::ord_pair< T >::change_both()`, `mln::util::ord_pair< T >::change_first()`, `mln::util::ord_pair< T >::change_second()`, and `mln::box< P >::is_valid()`.

**9.145.3.13** `template<typename T, typename I> void mln::util::tree_fast_to_image ( tree_fast< T > & tree, Image< I > & output_ ) [inline]`

Convert a `tree_fast` into an image.

#### Parameters

[in] *tree* The tree to convert.

[out] *output\_* The image containing tree informations.



**9.145.3.14** `template<typename T > tree_fast< T > mln::util::tree_to_fast ( tree< T > & input )`  
`[inline]`

Facade.

Convert a tree into an tree\_fast.

#### Parameters

[in] *input* The tree to convert.

#### Returns

The tree\_fast containing tree informations.

References mln::util::tree< T >::root().

**9.145.3.15** `template<typename T , typename I > void mln::util::tree_to_image ( tree< T > & tree, Image< I > & output_ )`  
`[inline]`

Convert a tree into an image.

#### Parameters

[in] *tree* The tree to convert.

[out] *output\_* The image containing tree information.

## 9.146 mln::util::impl Namespace Reference

Implementation namespace of util namespace.

### 9.146.1 Detailed Description

Implementation namespace of util namespace.

## 9.147 mln::value Namespace Reference

Namespace of materials related to pixel value types.

### Namespaces

- namespace [impl](#)

*Implementation namespace of value namespace.*

## Classes

- class [float01](#)  
*Class for floating values restricted to the interval [0..1] and discretized with n bits.*
- struct [float01\\_f](#)  
*Class for floating values restricted to the interval [0..1].*
- struct [graylevel](#)  
*General gray-level class on n bits.*
- struct [graylevel\\_f](#)  
*General gray-level class on n bits.*
- struct [int\\_s](#)  
*Signed integer value class.*
- struct [int\\_u](#)  
*Unsigned integer value class.*
- struct [int\\_u\\_sat](#)  
*Unsigned integer value class with saturation behavior.*
- struct [Integer](#)  
*Concept of integer.*
- struct [Integer< void >](#)  
*Category flag type.*
- struct [label](#)  
*Label value class.*
- struct [lut\\_vec](#)  
*Class that defines FIXME.*
- class [proxy](#)  
*Generic proxy class for an image pixel value.*
- struct [rgb](#)  
*Color class for red-green-blue where every component is n-bit encoded.*
- struct [set](#)  
*Class that defines the set of values of type T.*
- class [sign](#)  
*The sign class represents the value type composed by the set (-1, 0, 1) sign value type is a subset of the int value type.*
- struct [stack\\_image](#)  
*Stack image class.*

- struct [super\\_value](#)< sign >  
*Specializations:*
- struct [value\\_array](#)  
*Generic array class over indexed by a value set with type T.*

## Typedefs

- typedef [float01\\_](#)< 16 > [float01\\_16](#)  
*Alias for 16 bit [float01](#).*
- typedef [float01\\_](#)< 8 > [float01\\_8](#)  
*Alias for 8 bit [float01](#).*
- typedef [graylevel](#)< 16 > [gl16](#)  
*Alias for 16 bit [graylevel](#).*
- typedef [graylevel](#)< 8 > [gl8](#)  
*Alias for 8 bit [graylevel](#).*
- typedef [graylevel\\_f](#) [glf](#)  
*Alias for graylevels encoded by float.*
- typedef [int\\_s](#)< 16 > [int\\_s16](#)  
*Alias for signed 16-bit integers.*
- typedef [int\\_s](#)< 32 > [int\\_s32](#)  
*Alias for signed 32-bit integers.*
- typedef [int\\_s](#)< 8 > [int\\_s8](#)  
*Alias for signed 8-bit integers.*
- typedef [int\\_u](#)< 12 > [int\\_u12](#)  
*Alias for unsigned 12-bit integers.*
- typedef [int\\_u](#)< 16 > [int\\_u16](#)  
*Alias for unsigned 16-bit integers.*
- typedef [mln::value::int\\_u](#)< 32 > [int\\_u32](#)  
*Alias for unsigned 32-bit integers.*
- typedef [mln::value::int\\_u](#)< 8 > [int\\_u8](#)  
*Alias for unsigned 8-bit integers.*
- typedef [label](#)< 16 > [label\\_16](#)  
*Alias for 16-bit integers.*

- typedef `label< 32 > label_32`  
*Alias for 32-bit integers.*
- typedef `mln::value::label< 8 > label_8`  
*Alias for 8-bit labels.*
- typedef `rgb< 16 > rgb16`  
*Color class for red-green-blue where every component is 16-bit encoded.*
- typedef `rgb< 8 > rgb8`  
*Color class for red-green-blue where every component is 8-bit encoded.*

## Functions

- template<typename Dest , typename Src >  
Dest `cast` (const Src &src)  
*Cast a value `src` from type `Src` to type `Dest`.*
- template<typename V >  
internal::equiv\_< V >::ret `equiv` (const mln::Value< V > &v)  
*Access to the equivalent value.*
- template<unsigned n>  
`rgb< n >::interop operator+` (const `rgb< n >` &lhs, const `rgb< n >` &rhs)  
*Addition.*
- template<typename H , typename S , typename L >  
`hsl_< H, S, L > operator+` (const `hsl_< H, S, L >` &lhs, const `hsl_< H, S, L >` &rhs)  
*Addition.*
- template<unsigned n>  
std::ostream & `operator<<` (std::ostream &ostr, const `label< n >` &l)  
*Print a label `l` into the output stream `ostr`.*
- template<unsigned n>  
std::ostream & `operator<<` (std::ostream &ostr, const `rgb< n >` &c)  
*Print an rgb `c` into the output stream `ostr`.*
- std::ostream & `operator<<` (std::ostream &ostr, const `graylevel_f` &g)  
*Op<<.*
- template<typename T >  
std::ostream & `operator<<` (std::ostream &ostr, const scalar\_< T > &s)  
*Print a scalar `s` in an output stream `ostr`.*
- template<typename H , typename S , typename L >  
std::ostream & `operator<<` (std::ostream &ostr, const `hsl_< H, S, L >` &c)  
*Print an hsl `c` into the output stream `ostr`.*

- template<unsigned n>  
std::ostream & **operator<<** (std::ostream &ostr, const **graylevel**< n > &g)  
*Op<<*.
- template<unsigned n>  
std::ostream & **operator<<** (std::ostream &ostr, const float01\_< n > &f)  
*Op<<*.
- std::ostream & **operator<<** (std::ostream &ostr, const **sign** &i)  
*Print an signed integer i into the output stream ostr.*
- template<unsigned n>  
std::ostream & **operator<<** (std::ostream &ostr, const **int\_u**< n > &i)  
*Print an unsigned integer i into the output stream ostr.*
- template<unsigned n>  
std::ostream & **operator<<** (std::ostream &ostr, const **int\_s**< n > &i)  
*Print an signed integer i into the output stream ostr.*
- template<unsigned n>  
std::ostream & **operator<<** (std::ostream &ostr, const **int\_u\_sat**< n > &i)  
*Print a saturated unsigned integer i into the output stream ostr.*
- bool **operator==** (const **sign** &lhs, const **sign** &rhs)  
*Comparison operator.*
- template<typename V >  
V **other** (const V &val)  
*Give an other value than val.*
- template<typename H , typename S , typename L >  
hsl\_< H, S, L > **operator-** (const hsl\_< H, S, L > &lhs, const hsl\_< H, S, L > &rhs)  
*Subtraction.*
- template<typename H , typename S , typename L , typename S2 >  
hsl\_< H, S, L > **operator\*** (const hsl\_< H, S, L > &lhs, const mln::value::scalar\_< S2 > &s)  
*Product.*
- template<typename H , typename S , typename L , typename S2 >  
hsl\_< H, S, L > **operator/** (const hsl\_< H, S, L > &lhs, const mln::value::scalar\_< S2 > &s)  
*Division.*
- template<typename H , typename S , typename L >  
bool **operator==** (const hsl\_< H, S, L > &lhs, const hsl\_< H, S, L > &rhs)  
*Comparison.*

- `template<unsigned n>`  
`rgb< n >::interop operator-` (const `rgb< n >` &lhs, const `rgb< n >` &rhs)  
*Subtraction.*
- `template<unsigned n, typename S >`  
`rgb< n >::interop operator*` (const `rgb< n >` &lhs, const `mln::value::scalar_< S >` &s)  
*Product.*
- `template<unsigned n, typename S >`  
`rgb< n >::interop operator/` (const `rgb< n >` &lhs, const `mln::value::scalar_< S >` &s)  
*Division.*
- `template<typename I >`  
`stack_image< 2, const I > stack` (const `Image< I >` &ima1, const `Image< I >` &ima2)  
*Shortcut to build a stack with two images.*

### 9.147.1 Detailed Description

Namespace of materials related to pixel value types.

### 9.147.2 Typedef Documentation

#### 9.147.2.1 `typedef float01_<16> mln::value::float01_16`

Alias for 16 bit `float01`.

#### 9.147.2.2 `typedef float01_<8> mln::value::float01_8`

Alias for 8 bit `float01`.

#### 9.147.2.3 `typedef graylevel<16> mln::value::gl16`

Alias for 16 bit graylevel.

#### 9.147.2.4 `typedef graylevel<8> mln::value::gl8`

Alias for 8 bit graylevel.

#### 9.147.2.5 `typedef graylevel_f mln::value::glf`

Alias for graylevels encoded by float.

#### 9.147.2.6 `typedef int_s<16> mln::value::int_s16`

Alias for signed 16-bit integers.

**9.147.2.7 typedef int\_s<32> mln::value::int\_s32**

Alias for signed 32-bit integers.

**9.147.2.8 typedef int\_s<8> mln::value::int\_s8**

Alias for signed 8-bit integers.

**9.147.2.9 typedef int\_u<12> mln::value::int\_u12**

Alias for unsigned 12-bit integers.

**9.147.2.10 typedef int\_u<16> mln::value::int\_u16**

Alias for unsigned 16-bit integers.

**9.147.2.11 typedef mln::value::int\_u<32> mln::value::int\_u32**

Alias for unsigned 32-bit integers.

**9.147.2.12 typedef mln::value::int\_u<8> mln::value::int\_u8**

Alias for unsigned 8-bit integers.

**9.147.2.13 typedef label<16> mln::value::label\_16**

Alias for 16-bit integers.

**9.147.2.14 typedef label<32> mln::value::label\_32**

Alias for 32-bit integers.

**9.147.2.15 typedef mln::value::label<8> mln::value::label\_8**

Alias for 8-bit labels.

**9.147.2.16 typedef rgb<16> mln::value::rgb16**

Color class for red-green-blue where every component is 16-bit encoded.

**9.147.2.17 typedef rgb<8> mln::value::rgb8**

Color class for red-green-blue where every component is 8-bit encoded.

### 9.147.3 Function Documentation

**9.147.3.1** `template<typename Dest , typename Src > Dest mln::value::cast ( const Src & src ) [inline]`

Cast a value `src` from type `Src` to type `Dest`.

**9.147.3.2** `template<typename V > internal::equiv_< V >::ret mln::value::equiv ( const mln::Value< V > & v ) [inline]`

Access to the equivalent value.

Referenced by `mln::labeling::superpose()`.

**9.147.3.3** `template<unsigned n, typename S > rgb< n >::interop mln::value::operator* ( const rgb< n > & lhs, const mln::value::scalar_< S > & s ) [inline]`

Product.

**9.147.3.4** `template<typename H , typename S , typename L , typename S2 > hsl_< H, S, L > mln::value::operator* ( const hsl_< H, S, L > & lhs, const mln::value::scalar_< S2 > & s )`

Product.

**9.147.3.5** `template<unsigned n> rgb< n >::interop mln::value::operator+ ( const rgb< n > & lhs, const rgb< n > & rhs ) [inline]`

Addition.

{

**9.147.3.6** `template<typename H , typename S , typename L > hsl_< H, S, L > mln::value::operator+ ( const hsl_< H, S, L > & lhs, const hsl_< H, S, L > & rhs )`

Addition.

{

**9.147.3.7** `template<unsigned n> rgb< n >::interop mln::value::operator- ( const rgb< n > & lhs, const rgb< n > & rhs ) [inline]`

Subtraction.

**9.147.3.8** `template<typename H , typename S , typename L > hsl_< H, S, L > mln::value::operator- ( const hsl_< H, S, L > & lhs, const hsl_< H, S, L > & rhs )`

Subtraction.



**9.147.3.9** `template<unsigned n, typename S > rgb< n >::interop mln::value::operator/ ( const rgb< n > & lhs, const mln::value::scalar_< S > & s ) [inline]`

Division.

**9.147.3.10** `template<typename H , typename S , typename L , typename S2 > hsl_< H, S, L > mln::value::operator/ ( const hsl_< H, S, L > & lhs, const mln::value::scalar_< S2 > & s )`

Division.

**9.147.3.11** `template<typename T > std::ostream & mln::value::operator<< ( std::ostream & ostr, const scalar_< T > & s ) [inline]`

Print a scalar *s* in an output stream *ostr*.

**9.147.3.12** `std::ostream & mln::value::operator<< ( std::ostream & ostr, const sign & i ) [inline]`

Print an signed integer *i* into the output stream *ostr*.

#### Parameters

[in, out] *ostr* An output stream.

[in] *i* An sign value

#### Returns

The modified output stream *ostr*.

References `mln::debug::format()`.

**9.147.3.13** `template<unsigned n> std::ostream & mln::value::operator<< ( std::ostream & ostr, const int_s< n > & i ) [inline]`

Print an signed integer *i* into the output stream *ostr*.

#### Parameters

[in, out] *ostr* An output stream.

[in] *i* An signed integer.

#### Returns

The modified output stream *ostr*.

References `mln::debug::format()`.

**9.147.3.14** `template<unsigned n> std::ostream & mln::value::operator<< ( std::ostream & ostr, const graylevel< n > & g ) [inline]`

Op<<.

**9.147.3.15** `template<unsigned n> std::ostream & mln::value::operator<<< ( std::ostream & ostr, const int_u<n> & i ) [inline]`

Print an unsigned integer `i` into the output stream `ostr`.

#### Parameters

[in, out] `ostr` An output stream.

[in] `i` An unsigned integer.

#### Returns

The modified output stream `ostr`.

References `mln::debug::format()`.

**9.147.3.16** `template<unsigned n> std::ostream & mln::value::operator<<< ( std::ostream & ostr, const int_u_sat<n> & i ) [inline]`

Print a saturated unsigned integer `i` into the output stream `ostr`.

#### Parameters

[in, out] `ostr` An output stream.

[in] `i` A saturated unsigned integer.

#### Returns

The modified output stream `ostr`.

References `mln::debug::format()`.

**9.147.3.17** `template<unsigned n> std::ostream & mln::value::operator<<< ( std::ostream & ostr, const rgb<n> & c ) [inline]`

Print an rgb `c` into the output stream `ostr`.

#### Parameters

[in, out] `ostr` An output stream.

[in] `c` An rgb.

#### Returns

The modified output stream `ostr`.

References `mln::debug::format()`.

**9.147.3.18** `template<unsigned n> std::ostream & mln::value::operator<<< ( std::ostream & ostr, const float01_<n> & f ) [inline]`

Op<<<.

**9.147.3.19** `template<typename H , typename S , typename L > std::ostream & mln::value::operator<< ( std::ostream & ostr, const hsl_< H, S, L > & c ) [inline]`

Print an hsl *c* into the output stream *ostr*.

#### Parameters

[in, out] *ostr* An output stream.

[in] *c* An rgb.

#### Returns

The modified output stream *ostr*.

References mln::debug::format().

**9.147.3.20** `template<unsigned n> std::ostream & mln::value::operator<< ( std::ostream & ostr, const label< n > & l ) [inline]`

Print a label *l* into the output stream *ostr*.

#### Parameters

[in, out] *ostr* An output stream.

[in] *l* A label.

#### Returns

The modified output stream *ostr*.

References mln::debug::format().

**9.147.3.21** `std::ostream & mln::value::operator<< ( std::ostream & ostr, const graylevel_f & g ) [inline]`

Op<<.

References mln::value::graylevel\_f::value().

**9.147.3.22** `template<typename H , typename S , typename L > bool mln::value::operator==( const hsl_< H, S, L > & lhs, const hsl_< H, S, L > & rhs )`

Comparison.

**9.147.3.23** `bool mln::value::operator==( const sign & lhs, const sign & rhs ) [inline]`

Comparison operator.

**9.147.3.24** `template<typename V > V mln::value::other ( const V & val ) [inline]`

Give an other value than *val*.

**9.147.3.25** `template<typename I> stack_image< 2, const I> mln::value::stack ( const Image< I> & ima1, const Image< I> & ima2 ) [inline]`

Shortcut to build a stack with two images.

## 9.148 mln::value::impl Namespace Reference

Implementation namespace of value namespace.

### 9.148.1 Detailed Description

Implementation namespace of value namespace.

## 9.149 mln::win Namespace Reference

Namespace of image processing routines related to win.

### Classes

- struct [backdiag2d](#)  
*Diagonal line window defined on the 2D square grid.*
- struct [ball](#)  
*Generic ball window defined on a given grid.*
- struct [cube3d](#)  
*Cube window defined on the 3D grid.*
- struct [cuboid3d](#)  
*Cuboid defined on the 3-D square grid.*
- struct [diag2d](#)  
*Diagonal line window defined on the 2D square grid.*
- struct [line](#)  
*Generic line window defined on a given grid in the given dimension.*
- class [multiple](#)  
*Multiple window.*
- class [multiple\\_size](#)  
*Definition of a multiple-size window.*
- struct [octagon2d](#)  
*Octagon window defined on the 2D square grid.*
- struct [rectangle2d](#)

*Rectangular window defined on the 2D square grid.*

## Typedefs

- typedef `ball`< `grid::square`, `def::coord` > `disk2d`  
*2D disk window; precisely, ball-shaped window defined on the 2D square grid.*
- typedef `line`< `grid::square`, 1, `def::coord` > `hline2d`  
*Horizontal line window defined on the 2D square grid.*
- typedef `line`< `grid::tick`, 0, `def::coord` > `segment1d`  
*Segment window defined on the 1D grid.*
- typedef `line`< `grid::cube`, 0, `def::coord` > `sline3d`  
*Depth line window defined on the 3D cubic grid.*
- typedef `ball`< `grid::cube`, `def::coord` > `sphere3d`  
*3D sphere window; precisely, ball-shaped window defined on the 3D cubic grid.*
- typedef `line`< `grid::square`, 0, `def::coord` > `vline2d`  
*Vertical line window defined on the 2D square grid.*

## Functions

- template<typename N1 , typename N2 >  
`neighb`< typename N1::window::regular > `diff` (const `Neighborhood`< N1 > &nbh1, const `Neighborhood`< N2 > &nbh2)  
*Set difference between a couple of neighborhoods nbh1 and nbh2.*
- template<typename W1 , typename W2 >  
`mln_regular` (W1) `diff`(const `Window`< W1 > &win1  
*Set difference between a couple of windows win1 and win2.*
- template<typename W >  
`mln_regular` (W) `shift`(const `Window`< W > &win  
*Shift a window win with a delta-point dp.*
- template<typename W >  
W `sym` (const `Window`< W > &win)  
*Give the symmetrical window of win.*
- template<typename W >  
W `sym` (const `Weighted_Window`< W > &w\_win)  
*Give the symmetrical weighted window of w\_win.*

### 9.149.1 Detailed Description

Namespace of image processing routines related to `win`.

### 9.149.2 Function Documentation

**9.149.2.1** `template<typename N1 , typename N2 > N2 neighb< typename N1::window::regular > mln::win::diff ( const Neighborhood< N1 > & nbh1, const Neighborhood< N2 > & nbh2 )`

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

Referenced by `mln::operator()`.

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

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

**9.149.2.3** `template<typename W > mln::win::mln_regular ( W ) const [inline]`

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

**9.149.2.4** `template<typename W > W mln::win::sym ( const Window< W > & win ) [inline]`

Give the symmetrical window of `win`.

Referenced by `mln::c18()`, `mln::c26()`, `mln::c4_3d()`, `mln::c6()`, `mln::morpho::hit_or_miss_background_opening()`, `mln::morpho::hit_or_miss_opening()`, `mln::morpho::opening::approx::structural()`, and `mln::morpho::closing::approx::structural()`.

**9.149.2.5** `template<typename W > W mln::win::sym ( const Weighted_Window< W > & w_win ) [inline]`

Give the symmetrical weighted window of `w_win`.

# Chapter 10

## Class Documentation

### 10.1 mln::accu::center< P, V > Struct Template Reference

Mass center accumulator.

```
#include <center.hh>
```

Inherits base< V, center< P, V > >.

#### Public Member Functions

- bool [is\\_valid](#) () const  
*Check whether this accu is able to return a result.*
- unsigned [nsites](#) () const  
*Return the number of sites taken in consideration.*
- void [take\\_as\\_init](#) (const T &t)  
*Take as initialization the value  $t$ .*
- void [take\\_n\\_times](#) (unsigned n, const T &t)  
*Take  $n$  times the value  $t$ .*
- V [to\\_result](#) () const  
*Get the value of the accumulator.*
  
- void [init](#) ()  
*Manipulators.*

#### 10.1.1 Detailed Description

```
template<typename P, typename V = typename P::vec> struct mln::accu::center< P, V >
```

Mass center accumulator.

## Template Parameters

*P* the type of site.

*V* the type of vector to be used as result. The default vector type is the one provided by *P*.

## 10.1.2 Member Function Documentation

**10.1.2.1** `template<typename P , typename V > void mln::accu::center< P, V >::init ( )`  
**[inline]**

Manipulators.

References `mln::literal::zero`.

**10.1.2.2** `template<typename P , typename V > bool mln::accu::center< P, V >::is_valid ( )`  
**const [inline]**

Check whether this `accu` is able to return a result.

Referenced by `mln::accu::center< P, V >::to_result()`.

**10.1.2.3** `template<typename P , typename V > unsigned mln::accu::center< P, V >::nsites ( )`  
**const [inline]**

Return the number of sites taken in consideration.

**10.1.2.4** `void mln::Accumulator< center< P, V > >::take_as_init ( const T & t )`  
**[inherited]**

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.1.2.5** `void mln::Accumulator< center< P, V > >::take_n_times ( unsigned n, const T & t )`  
**[inherited]**

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.1.2.6** `template<typename P , typename V > V mln::accu::center< P, V >::to_result ( ) const`  
**[inline]**

Get the value of the accumulator.

References `mln::accu::center< P, V >::is_valid()`.

## 10.2 mln::accu::convolve< T1, T2, R > Struct Template Reference

Generic convolution accumulator class.



```
#include <convolve.hh>
Inherits base< R, convolve< T1, T2, R > >.
```

## Public Member Functions

- bool [is\\_valid](#) () const  
*Check whether this accu is able to return a result.*
- void [take\\_as\\_init](#) (const T &t)  
*Take as initialization the value  $\tau$ .*
- void [take\\_n\\_times](#) (unsigned n, const T &t)  
*Take  $n$  times the value  $\tau$ .*
- R [to\\_result](#) () const  
*Get the value of the accumulator.*
- void [init](#) ()  
*Manipulators.*

### 10.2.1 Detailed Description

```
template<typename T1, typename T2, typename R = typename mln::trait::value_< typename
mln::trait::op::times< T1 , T2 >::ret >::sum> struct mln::accu::convolve< T1, T2, R >
```

Generic convolution accumulator class. Parameters T1 and T2 are the type of values to be convolved. Parameter R is the result type.

### 10.2.2 Member Function Documentation

**10.2.2.1** `template<typename T1 , typename T2 , typename R > void mln::accu::convolve< T1, T2, R >::init ( ) [inline]`

Manipulators.

References mln::literal::zero.

**10.2.2.2** `template<typename T1 , typename T2 , typename R > bool mln::accu::convolve< T1, T2, R >::is_valid ( ) const [inline]`

Check whether this accu is able to return a result.

Always true here.

**10.2.2.3** `void mln::Accumulator< convolve< T1, T2, R > >::take_as_init ( const T & t )`  
**[inherited]**

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.2.2.4** `void mln::Accumulator< convolve< T1, T2, R > >::take_n_times ( unsigned n, const T & t )` **[inherited]**

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.2.2.5** `template<typename T1 , typename T2 , typename R > R mln::accu::convolve< T1, T2, R >::to_result ( ) const` **[inline]**

Get the value of the accumulator.

## 10.3 mln::accu::count\_adjacent\_vertices< F, S > Struct Template Reference

[Accumulator](#) class counting the number of vertices adjacent to a set of `mln::p_edges_psite` (i.e., a set of edges).

```
#include <count_adjacent_vertices.hh>
```

Inherits `base< unsigned, count_adjacent_vertices< F, S > >`.

### Public Member Functions

- `bool is_valid () const`  
*Return whether this `accu` can return a result.*
- `void take_as_init (const T &t)`  
*Take as initialization the value  $t$ .*
- `void take_n_times (unsigned n, const T &t)`  
*Take  $n$  times the value  $t$ .*
- `unsigned to_result () const`  
*Get the value of the accumulator.*
- `void init ()`  
*Manipulators.*
- `void set_value (unsigned c)`  
*Force the value of the counter to  $c$ .*

### 10.3.1 Detailed Description

`template<typename F, typename S> struct mln::accu::count_adjacent_vertices< F, S >`

[Accumulator](#) class counting the number of vertices adjacent to a set of `mln::p_edges_psite` (i.e., a set of edges). The type to be count is `mln::util::pix< pw::image<F, S> >` where `F` and `S` are the parameters of this class.

This accumulator is used by `mln::closing_area_on_vertices` and `mln::opening_area_on_vertices`.

### 10.3.2 Member Function Documentation

**10.3.2.1** `template<typename F , typename S > void mln::accu::count_adjacent_vertices< F, S >::init ( ) [inline]`

Manipulators.

**10.3.2.2** `template<typename F , typename S > bool mln::accu::count_adjacent_vertices< F, S >::is_valid ( ) const [inline]`

Return whether this accu can return a result.

**10.3.2.3** `template<typename F , typename S > void mln::accu::count_adjacent_vertices< F, S >::set_value ( unsigned c ) [inline]`

Force the value of the counter to `c`.

**10.3.2.4** `void mln::Accumulator< count_adjacent_vertices< F, S > >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.3.2.5** `void mln::Accumulator< count_adjacent_vertices< F, S > >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.3.2.6** `template<typename F , typename S > unsigned mln::accu::count_adjacent_vertices< F, S >::to_result ( ) const [inline]`

Get the value of the accumulator.

## 10.4 mln::accu::count\_value< V > Struct Template Reference

Define an accumulator that counts the occurrence of a given value.

```
#include <count_value.hh>
```

Inherits base< unsigned, count\_value< V > >.

## Public Member Functions

- bool `is_valid` () const  
*Check whether this accu is able to return a result.*
- void `take_as_init` (const T &t)  
*Take as initialization the value  $\tau$ .*
- void `take_n_times` (unsigned n, const T &t)  
*Take  $n$  times the value  $\tau$ .*
- unsigned `to_result` () const  
*Get the value of the accumulator.*
- void `init` ()  
*Manipulators.*
- void `set_value` (unsigned c)  
*Force the value of the counter to  $c$ .*

### 10.4.1 Detailed Description

```
template<typename V> struct mln::accu::count_value< V >
```

Define an accumulator that counts the occurrence of a given value.

### 10.4.2 Member Function Documentation

**10.4.2.1** `template<typename V > void mln::accu::count_value< V >::init ( ) [inline]`

Manipulators.

**10.4.2.2** `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.

**10.4.2.3** `template<typename V > void mln::accu::count_value< V >::set_value ( unsigned c ) [inline]`

Force the value of the counter to  $c$ .

#### 10.4.2.4 void mln::Accumulator< count\_value< V > >::take\_as\_init ( const T & t ) [inherited]

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by take\_as\_init\_ (ending with '\_').

#### 10.4.2.5 void mln::Accumulator< count\_value< V > >::take\_n\_times ( unsigned n, const T & t ) [inherited]

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by take\_as\_init\_ (ending with '\_').

#### 10.4.2.6 template<typename V > unsigned mln::accu::count\_value< V >::to\_result ( ) const [inline]

Get the value of the accumulator.

## 10.5 mln::accu::histo< V > Struct Template Reference

Generic histogram class over a value set with type  $V$ .

```
#include <histo.hh>
```

Inherits base< const std::vector< unsigned > &, histo< V > >.

### Public Member Functions

- bool [is\\_valid](#) () const  
*Check whether this accu is able to return a result.*
- void [take\\_as\\_init](#) (const T &t)  
*Take as initialization the value  $t$ .*
- void [take\\_n\\_times](#) (unsigned n, const T &t)  
*Take  $n$  times the value  $t$ .*
- void [take](#) (const argument &t)  
*Manipulators.*
- const std::vector< unsigned > & [vect](#) () const  
*Get the value of the accumulator.*

## 10.5.1 Detailed Description

```
template<typename V> struct mln::accu::histo< V >
```

Generic histogram class over a value set with type `V`.

## 10.5.2 Member Function Documentation

```
10.5.2.1 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.

```
10.5.2.2 template<typename V > void mln::accu::histo< V >::take ( const argument & t )
[inline]
```

Manipulators.

```
10.5.2.3 void mln::Accumulator< histo< V > >::take_as_init ( const T & t ) [inherited]
```

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

```
10.5.2.4 void mln::Accumulator< histo< V > >::take_n_times ( unsigned n, const T & t )
[inherited]
```

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

```
10.5.2.5 template<typename V > const std::vector< unsigned > & mln::accu::histo< V >::vect (
) const [inline]
```

Get the value of the accumulator.

## 10.6 mln::accu::label\_used< L > Struct Template Reference

References all the labels used.

```
#include <label_used.hh>
```

Inherits `base< const fun::i2v::array< bool > &, label_used< L > >`.

### Public Member Functions

- void `init` ()

*Initialize accumulator attributes.*

- bool `is_valid` () const  
*Check whether this accu is able to return a result.*
  - void `take_as_init` (const T &t)  
*Take as initialization the value t.*
  - void `take_n_times` (unsigned n, const T &t)  
*Take n times the value t.*
  - const fun::i2v::array< bool > & `to_result` () const  
*Get the value of the accumulator.*
- 
- void `take` (const argument &)  
*Manipulators.*

### 10.6.1 Detailed Description

`template<typename L> struct mln::accu::label_used< L >`

References all the labels used. The parameter *L* is the label type.

### 10.6.2 Member Function Documentation

**10.6.2.1** `template<typename L > void mln::accu::label_used< L >::init ( ) [inline]`

Initialize accumulator attributes.

**10.6.2.2** `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.

**10.6.2.3** `template<typename L > void mln::accu::label_used< L >::take ( const argument & l ) [inline]`

Manipulators.

**10.6.2.4** `void mln::Accumulator< label_used< L > >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value t.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

### 10.6.2.5 void mln::Accumulator< label\_used< L > >::take\_n\_times ( unsigned n, const T & t ) [inherited]

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by take\_as\_init\_ (ending with '\_').

### 10.6.2.6 template<typename L > const fun::i2v::array< bool > & mln::accu::label\_used< L >::to\_result ( ) const [inline]

Get the value of the accumulator.

## 10.7 mln::accu::logic::land Struct Reference

"Logical-and" accumulator.

```
#include <land.hh>
```

Inherits base< bool, land >.

### Public Member Functions

- bool [is\\_valid](#) () const  
*Check whether this accu is able to return a result.*
- void [take\\_as\\_init](#) (const T &t)  
*Take as initialization the value  $t$ .*
- void [take\\_n\\_times](#) (unsigned n, const T &t)  
*Take  $n$  times the value  $t$ .*
- bool [to\\_result](#) () const  
*Get the value of the accumulator.*
- void [init](#) ()  
*Manipulators.*

### 10.7.1 Detailed Description

"Logical-and" accumulator.

### 10.7.2 Member Function Documentation

#### 10.7.2.1 void mln::accu::logic::land::init ( ) [inline]

Manipulators.



**10.7.2.2 bool mln::accu::logic::land::is\_valid ( ) const [inline]**

Check whether this accu is able to return a result.

Always true here.

**10.7.2.3 void mln::Accumulator< land >::take\_as\_init ( const T & t ) [inherited]**

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by take\_as\_init\_ (ending with '\_').

**10.7.2.4 void mln::Accumulator< land >::take\_n\_times ( unsigned n, const T & t ) [inherited]**

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by take\_as\_init\_ (ending with '\_').

**10.7.2.5 bool mln::accu::logic::land::to\_result ( ) const [inline]**

Get the value of the accumulator.

**10.8 mln::accu::logic::land\_basic Struct Reference**

"Logical-and" accumulator.

```
#include <land_basic.hh>
```

Inherits base< bool, land\_basic >.

**Public Member Functions**

- bool [can\\_stop](#) () const  
*Test if it is worth for this accumulator to take extra data.*
- bool [is\\_valid](#) () const  
*Check whether this accu is able to return a result.*
- void [take\\_as\\_init](#) (const T &t)  
*Take as initialization the value  $t$ .*
- void [take\\_n\\_times](#) (unsigned n, const T &t)  
*Take  $n$  times the value  $t$ .*
- bool [to\\_result](#) () const  
*Get the value of the accumulator.*
- void [init](#) ()  
*Manipulators.*

## 10.8.1 Detailed Description

"Logical-and" accumulator. Conversely to [accu::logic::land](#), this version does not have the 'untake' method but features the 'can\_stop' method.

## 10.8.2 Member Function Documentation

### 10.8.2.1 `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.

### 10.8.2.2 `void mln::accu::logic::land_basic::init ( ) [inline]`

Manipulators.

### 10.8.2.3 `bool mln::accu::logic::land_basic::is_valid ( ) const [inline]`

Check whether this accu is able to return a result.

Always true here.

### 10.8.2.4 `void mln::Accumulator< land_basic >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '\_').

### 10.8.2.5 `void mln::Accumulator< land_basic >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '\_').

### 10.8.2.6 `bool mln::accu::logic::land_basic::to_result ( ) const [inline]`

Get the value of the accumulator.

## 10.9 mln::accu::logic::lor Struct Reference

"Logical-or" accumulator.

```
#include <lor.hh>
```

Inherits base< bool, lor >.

## Public Member Functions

- bool `is_valid` () const  
*Check whether this accu is able to return a result.*
- void `take_as_init` (const T &t)  
*Take as initialization the value  $t$ .*
- void `take_n_times` (unsigned n, const T &t)  
*Take  $n$  times the value  $t$ .*
- bool `to_result` () const  
*Get the value of the accumulator.*
  
- void `init` ()  
*Manipulators.*

### 10.9.1 Detailed Description

"Logical-or" accumulator.

### 10.9.2 Member Function Documentation

#### 10.9.2.1 void mln::accu::logic::lor::init ( ) [inline]

Manipulators.

#### 10.9.2.2 bool mln::accu::logic::lor::is\_valid ( ) const [inline]

Check whether this accu is able to return a result.

Always true here.

#### 10.9.2.3 void mln::Accumulator< lor >::take\_as\_init ( const T & t ) [inherited]

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

#### 10.9.2.4 void mln::Accumulator< lor >::take\_n\_times ( unsigned n, const T & t ) [inherited]

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

#### 10.9.2.5 bool mln::accu::logic::lor::to\_result ( ) const [inline]

Get the value of the accumulator.

## 10.10 mln::accu::logic::lor\_basic Struct Reference

"Logical-or" accumulator class.

```
#include <lor_basic.hh>
```

Inherits base< bool, lor\_basic >.

### Public Member Functions

- bool [can\\_stop](#) () const  
*Test if it is worth for this accumulator to take extra data.*
- bool [is\\_valid](#) () const  
*Check whether this accu is able to return a result.*
- void [take\\_as\\_init](#) (const T &t)  
*Take as initialization the value t.*
- void [take\\_n\\_times](#) (unsigned n, const T &t)  
*Take n times the value t.*
- bool [to\\_result](#) () const  
*Get the value of the accumulator.*
- void [init](#) ()  
*Manipulators.*

### 10.10.1 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.

### 10.10.2 Member Function Documentation

#### 10.10.2.1 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.

#### 10.10.2.2 void mln::accu::logic::lor\_basic::init ( ) [inline]

Manipulators.

**10.10.2.3** `bool mln::accu::logic::lor_basic::is_valid ( ) const [inline]`

Check whether this accu is able to return a result.

Always true here.

**10.10.2.4** `void mln::Accumulator< lor_basic >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.10.2.5** `void mln::Accumulator< lor_basic >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.10.2.6** `bool mln::accu::logic::lor_basic::to_result ( ) const [inline]`

Get the value of the accumulator.

**10.11 mln::accu::maj\_h< T > Struct Template Reference**

Compute the majority value.

```
#include <maj_h.hh>
```

Inherits `base< const T &, maj_h< T > >`.

**Public Member Functions**

- `bool is_valid () const`  
*Check whether this accu is able to return a result.*
- `void take_as_init (const T &t)`  
*Take as initialization the value  $t$ .*
- `void take_n_times (unsigned n, const T &t)`  
*Take  $n$  times the value  $t$ .*
- `const T & to_result () const`  
*Get the value of the accumulator.*
- `void init ()`  
*Manipulators.*

### 10.11.1 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.

### 10.11.2 Member Function Documentation

**10.11.2.1** `template<typename T > void mln::accu::maj_h< T >::init ( ) [inline]`

Manipulators.

**10.11.2.2** `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.

**10.11.2.3** `void mln::Accumulator< maj_h< T > >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value t.

Dev note: this is a final method; override if needed by take\_as\_init\_ (ending with '\_').

**10.11.2.4** `void mln::Accumulator< maj_h< T > >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take n times the value t.

Dev note: this is a final method; override if needed by take\_as\_init\_ (ending with '\_').

**10.11.2.5** `template<typename T > const T & mln::accu::maj_h< T >::to_result ( ) const [inline]`

Get the value of the accumulator.

## 10.12 mln::accu::math::count< T > Struct Template Reference

Generic counter accumulator.

```
#include <count.hh>
```

Inherits base< unsigned, count< T > >.

### Public Member Functions

- bool [is\\_valid](#) () const  
*Check whether this accu is able to return a result.*
- void [take\\_as\\_init](#) (const T &t)

Take as initialization the value  $t$ .

- void [take\\_n\\_times](#) (unsigned  $n$ , const T & $t$ )  
Take  $n$  times the value  $t$ .
- unsigned [to\\_result](#) () const  
Get the value of the accumulator.
- void [init](#) ()  
Manipulators.
- void [set\\_value](#) (unsigned  $c$ )  
Force the value of the counter to  $c$ .

### 10.12.1 Detailed Description

`template<typename T> struct mln::accu::math::count< T >`

Generic counter accumulator. The parameter  $T$  is the type to be count.

### 10.12.2 Member Function Documentation

**10.12.2.1** `template<typename T > void mln::accu::math::count< T >::init ( ) [inline]`

Manipulators.

**10.12.2.2** `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.

**10.12.2.3** `template<typename T > void mln::accu::math::count< T >::set_value ( unsigned c ) [inline]`

Force the value of the counter to  $c$ .

**10.12.2.4** `void mln::Accumulator< count< T > >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.12.2.5** `void mln::Accumulator< count< T > >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.12.2.6** `template<typename T> unsigned mln::accu::math::count< T >::to_result ( ) const`  
**[inline]**

Get the value of the accumulator.

## 10.13 mln::accu::math::inf< T > Struct Template Reference

Generic inf accumulator class.

```
#include <inf.hh>
```

Inherits `base< const T &, inf< T > >`.

### Public Member Functions

- `bool is_valid () const`  
*Check whether this accu is able to return a result.*
- `void take_as_init (const T &t)`  
*Take as initialization the value  $t$ .*
- `void take_n_times (unsigned n, const T &t)`  
*Take  $n$  times the value  $t$ .*
- `const T & to_result () const`  
*Get the value of the accumulator.*
- `void init ()`  
*Manipulators.*

### 10.13.1 Detailed Description

```
template<typename T> struct mln::accu::math::inf< T >
```

Generic inf accumulator class. The parameter `T` is the type of values.

### 10.13.2 Member Function Documentation

**10.13.2.1** `template<typename T> void mln::accu::math::inf< T >::init ( )` **[inline]**

Manipulators.



**10.13.2.2** `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.

**10.13.2.3** `void mln::Accumulator< inf< T > >::take_as_init ( const T & t )` **[inherited]**

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.13.2.4** `void mln::Accumulator< inf< T > >::take_n_times ( unsigned n, const T & t )`  
**[inherited]**

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.13.2.5** `template<typename T> const T & mln::accu::math::inf< T >::to_result ( ) const`  
**[inline]**

Get the value of the accumulator.

## 10.14 mln::accu::math::sum< T, S > Struct Template Reference

Generic sum accumulator class.

```
#include <sum.hh>
```

Inherits base< const S &, sum< T, S > >.

### Public Member Functions

- `bool is_valid () const`  
*Check whether this accu is able to return a result.*
- `void take_as_init (const T &t)`  
*Take as initialization the value  $t$ .*
- `void take_n_times (unsigned n, const T &t)`  
*Take  $n$  times the value  $t$ .*
- `const S & to_result () const`  
*Get the value of the accumulator.*
- `void init ()`  
*Manipulators.*

### 10.14.1 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`.

### 10.14.2 Member Function Documentation

**10.14.2.1** `template<typename T, typename S > void mln::accu::math::sum< T, S >::init ( )`  
**[inline]**

Manipulators.

References `mln::literal::zero`.

**10.14.2.2** `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.

**10.14.2.3** `void mln::Accumulator< sum< T, S > >::take_as_init ( const T & t )` **[inherited]**

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.14.2.4** `void mln::Accumulator< sum< T, S > >::take_n_times ( unsigned n, const T & t )`  
**[inherited]**

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.14.2.5** `template<typename T, typename S > const S & mln::accu::math::sum< T, S >::to_result ( ) const` **[inline]**

Get the value of the accumulator.

## 10.15 mln::accu::math::sup< T > Struct Template Reference

Generic sup accumulator class.

```
#include <sup.hh>
```

Inherits `base< const T &, sup< T > >`.

## Public Member Functions

- bool `is_valid` () const  
*Check whether this accu is able to return a result.*
- void `take_as_init` (const T &t)  
*Take as initialization the value t.*
- void `take_n_times` (unsigned n, const T &t)  
*Take n times the value t.*
- const T & `to_result` () const  
*Get the value of the accumulator.*
  
- void `init` ()  
*Manipulators.*

### 10.15.1 Detailed Description

`template<typename T> struct mln::accu::math::sup< T >`

Generic sup accumulator class. The parameter T is the type of values.

### 10.15.2 Member Function Documentation

**10.15.2.1** `template<typename T> void mln::accu::math::sup< T >::init ( ) [inline]`

Manipulators.

**10.15.2.2** `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.

**10.15.2.3** `void mln::Accumulator< sup< T > >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value t.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '\_').

**10.15.2.4** `void mln::Accumulator< sup< T > >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take n times the value t.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '\_').

**10.15.2.5** `template<typename T > const T & mln::accu::math::sup< T >::to_result ( ) const [inline]`

Get the value of the accumulator.

## 10.16 mln::accu::max\_site< I > Struct Template Reference

Define an accumulator that computes the first site with the maximum value in an image.

```
#include <max_site.hh>
```

Inherits base< I::psite, max\_site< I > >.

### Public Member Functions

- bool `is_valid` () const  
*Check whether this accu is able to return a result.*
- void `take_as_init` (const T &t)  
*Take as initialization the value t.*
- void `take_n_times` (unsigned n, const T &t)  
*Take n times the value t.*
- I::psite `to_result` () const  
*Get the value of the accumulator.*
- void `init` ()  
*Manipulators.*

### 10.16.1 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.

### 10.16.2 Member Function Documentation

**10.16.2.1** `template<typename I > void mln::accu::max_site< I >::init ( ) [inline]`

Manipulators.

**10.16.2.2** `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.

**10.16.2.3** `void mln::Accumulator< max_site< I > >::take_as_init ( const T & t )`  
[`inherited`]

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.16.2.4** `void mln::Accumulator< max_site< I > >::take_n_times ( unsigned n, const T & t )`  
[`inherited`]

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.16.2.5** `template<typename I > I::psite mln::accu::max_site< I >::to_result ( ) const`  
[`inline`]

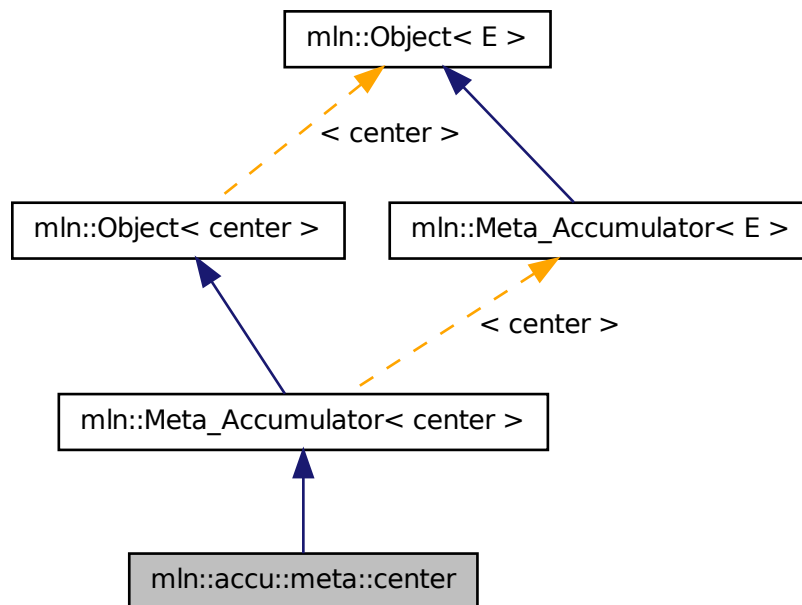
Get the value of the accumulator.

## 10.17 mln::accu::meta::center Struct Reference

Meta accumulator for center.

```
#include <center.hh>
```

Inheritance diagram for mln::accu::meta::center:



### 10.17.1 Detailed Description

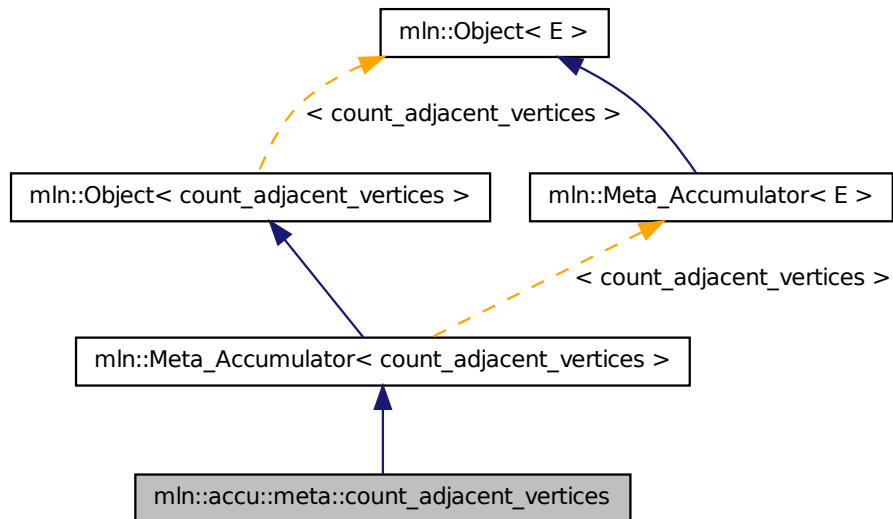
Meta accumulator for center.

## 10.18 mln::accu::meta::count\_adjacent\_vertices Struct Reference

Meta accumulator for [count\\_adjacent\\_vertices](#).

```
#include <count_adjacent_vertices.hh>
```

Inheritance diagram for mln::accu::meta::count\_adjacent\_vertices:



### 10.18.1 Detailed Description

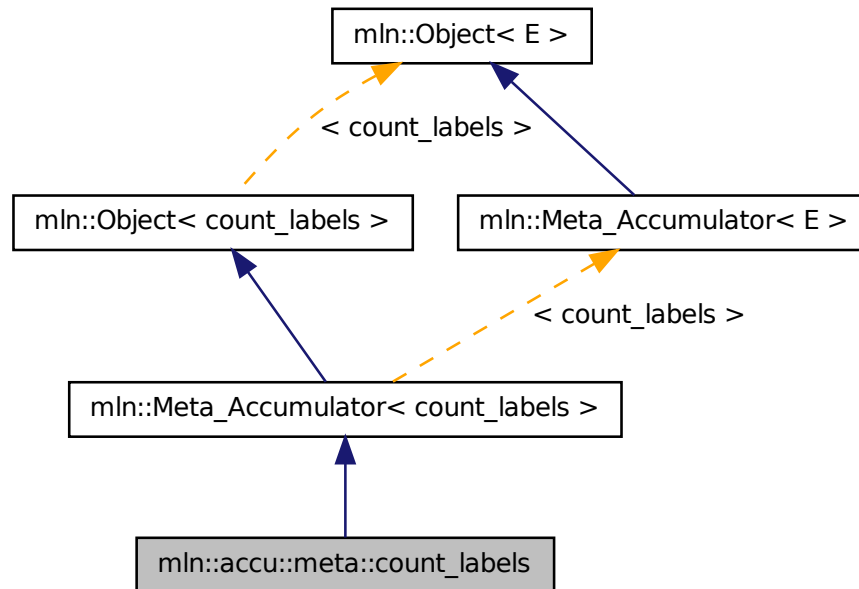
Meta accumulator for [count\\_adjacent\\_vertices](#).

## 10.19 mln::accu::meta::count\_labels Struct Reference

Meta accumulator for [count\\_labels](#).

```
#include <count_labels.hh>
```

Inheritance diagram for mln::accu::meta::count\_labels:



### 10.19.1 Detailed Description

Meta accumulator for [count\\_labels](#).

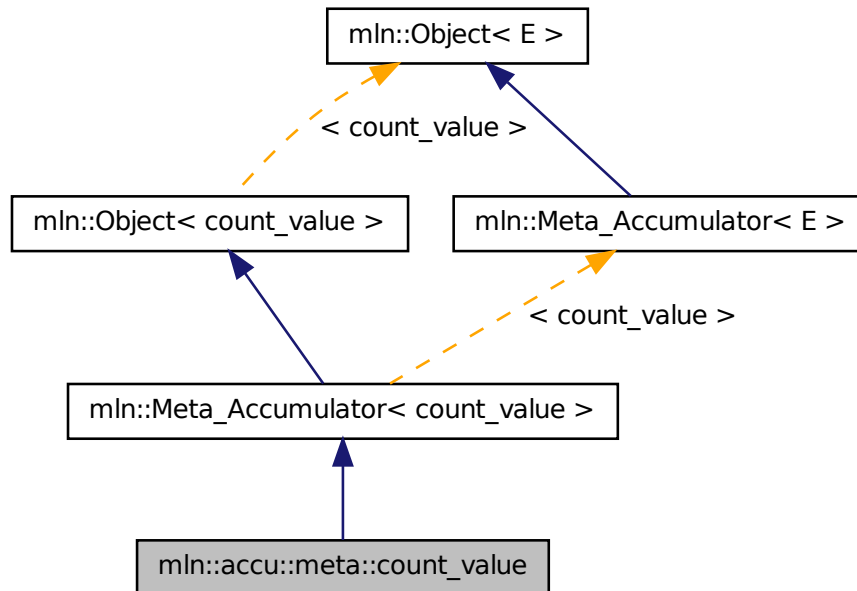
## 10.20 mln::accu::meta::count\_value Struct Reference

FIXME: How to write a meta accumulator with a constructor taking a generic argument? Meta accumulator for [count\\_value](#).

```
#include <count_value.hh>
```



Inheritance diagram for mln::accu::meta::count\_value:



### 10.20.1 Detailed Description

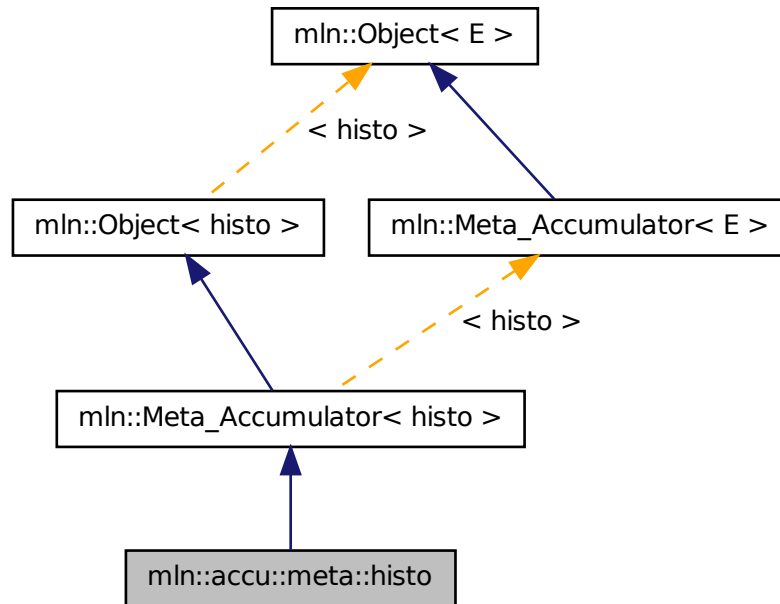
FIXME: How to write a meta accumulator with a constructor taking a generic argument? Meta accumulator for `count_value`.

## 10.21 mln::accu::meta::histo Struct Reference

Meta accumulator for histo.

```
#include <histo.hh>
```

Inheritance diagram for mln::accu::meta::histo:



### 10.21.1 Detailed Description

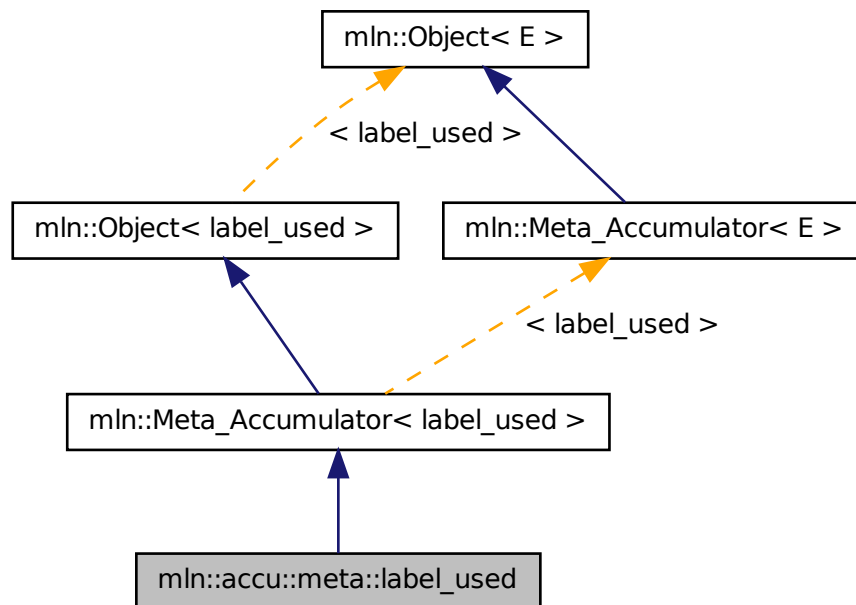
Meta accumulator for histo.

## 10.22 mln::accu::meta::label\_used Struct Reference

Meta accumulator for [label\\_used](#).

```
#include <label_used.hh>
```

Inheritance diagram for mln::accu::meta::label\_used:



### 10.22.1 Detailed Description

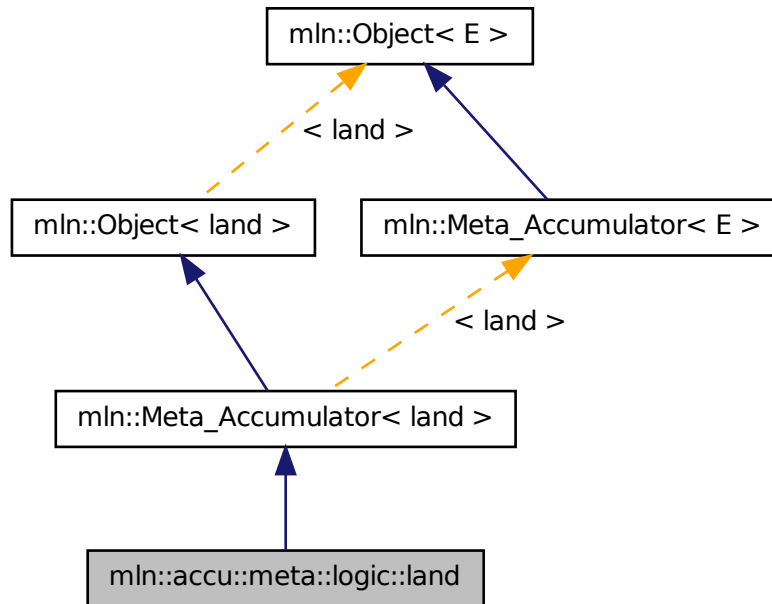
Meta accumulator for [label\\_used](#).

## 10.23 mln::accu::meta::logic::land Struct Reference

Meta accumulator for land.

```
#include <land.hh>
```

Inheritance diagram for mln::accu::meta::logic::land:



### 10.23.1 Detailed Description

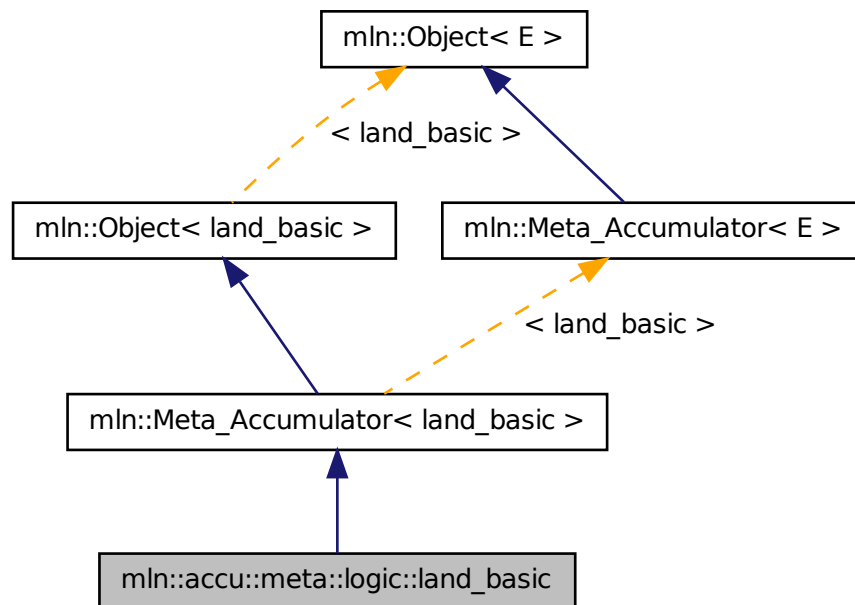
Meta accumulator for land.

## 10.24 mln::accu::meta::logic::land\_basic Struct Reference

Meta accumulator for [land\\_basic](#).

```
#include <land_basic.hh>
```

Inheritance diagram for mln::accu::meta::logic::land\_basic:



### 10.24.1 Detailed Description

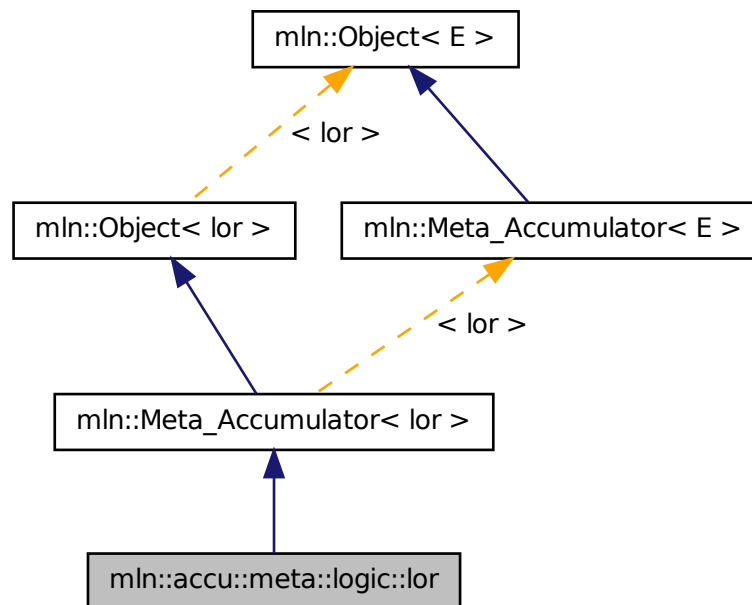
Meta accumulator for [land\\_basic](#).

## 10.25 mln::accu::meta::logic::lor Struct Reference

Meta accumulator for lor.

```
#include <lor.hh>
```

Inheritance diagram for mln::accu::meta::logic::lor:



### 10.25.1 Detailed Description

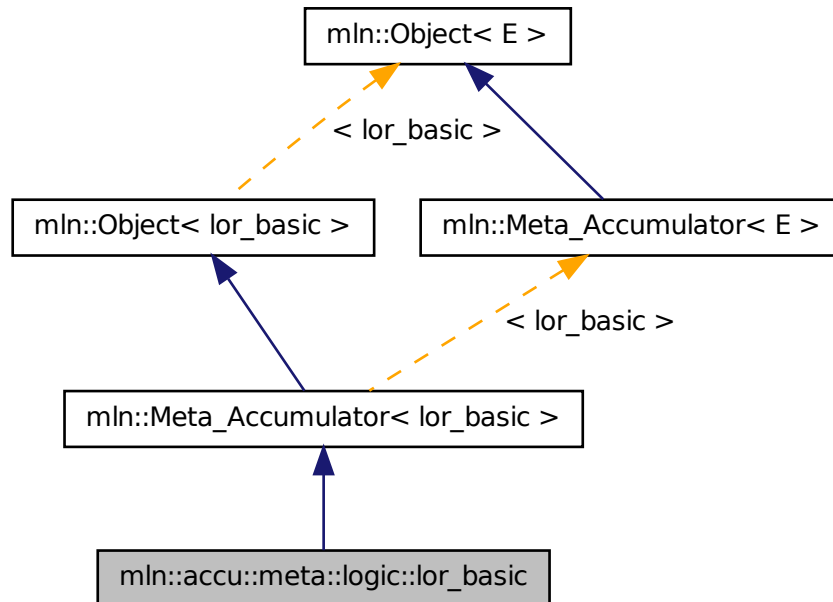
Meta accumulator for lor.

## 10.26 mln::accu::meta::logic::lor\_basic Struct Reference

Meta accumulator for [lor\\_basic](#).

```
#include <lor_basic.hh>
```

Inheritance diagram for mln::accu::meta::logic::lor\_basic:



### 10.26.1 Detailed Description

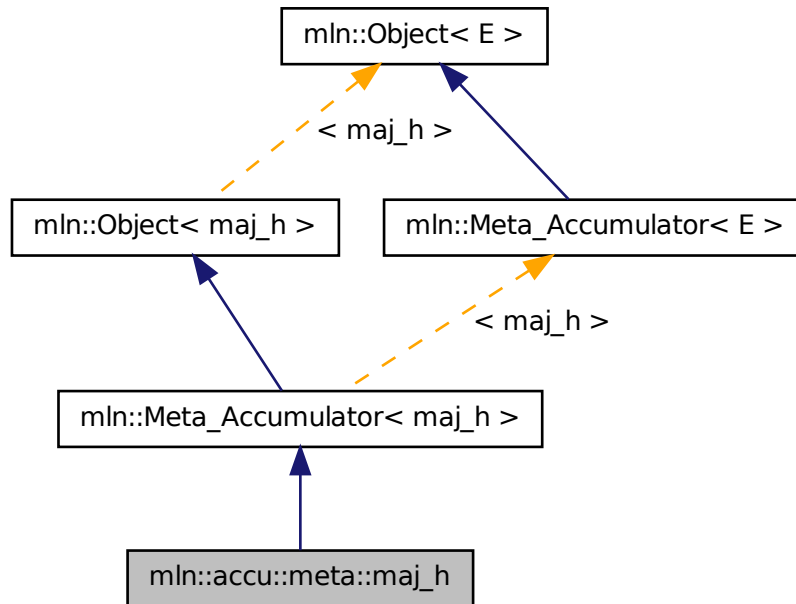
Meta accumulator for [lor\\_basic](#).

## 10.27 mln::accu::meta::maj\_h Struct Reference

Meta accumulator for [maj\\_h](#).

```
#include <maj_h.hh>
```

Inheritance diagram for mln::accu::meta::maj\_h:



### 10.27.1 Detailed Description

Meta accumulator for [maj\\_h](#).

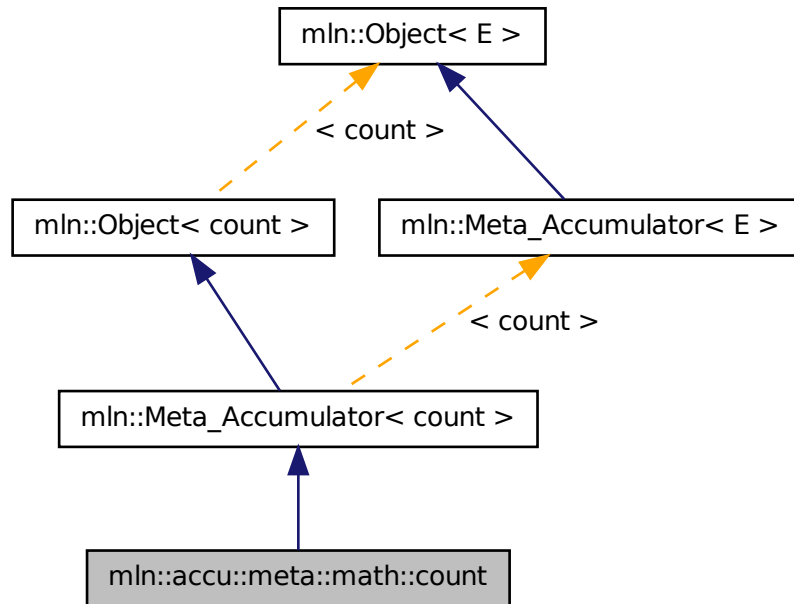
## 10.28 mln::accu::meta::math::count Struct Reference

Meta accumulator for count.

```
#include <count.hh>
```



Inheritance diagram for mln::accu::meta::math::count:



### 10.28.1 Detailed Description

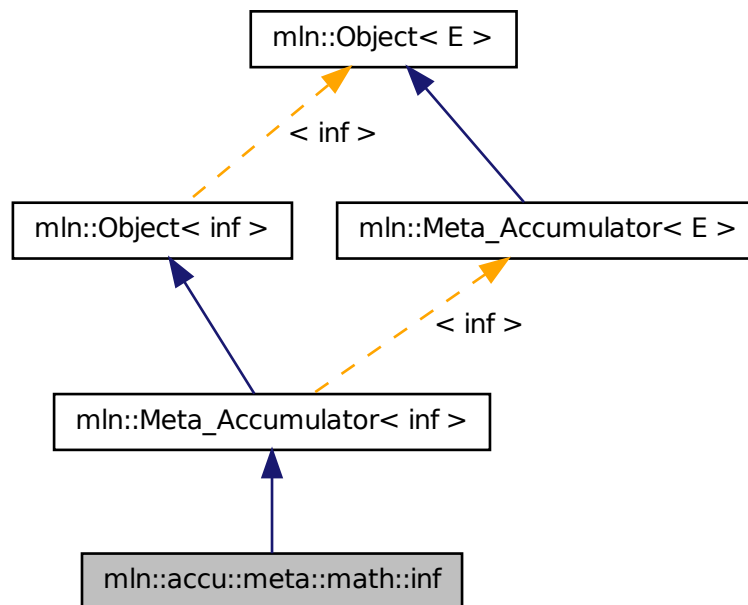
Meta accumulator for count.

## 10.29 mln::accu::meta::math::inf Struct Reference

Meta accumulator for inf.

```
#include <inf.hh>
```

Inheritance diagram for mln::accu::meta::math::inf:



### 10.29.1 Detailed Description

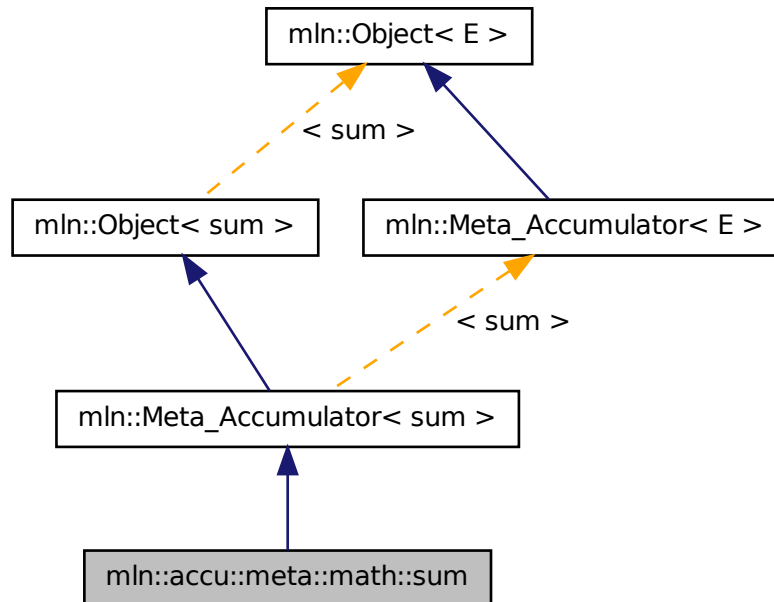
Meta accumulator for inf.

## 10.30 mln::accu::meta::math::sum Struct Reference

Meta accumulator for sum.

```
#include <sum.hh>
```

Inheritance diagram for mln::accu::meta::math::sum:



### 10.30.1 Detailed Description

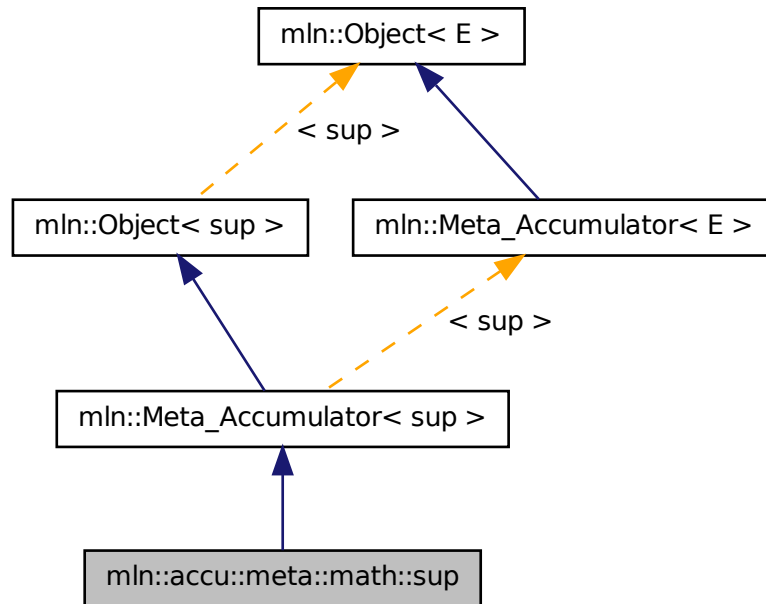
Meta accumulator for sum.

## 10.31 mln::accu::meta::math::sup Struct Reference

Meta accumulator for sup.

```
#include <sup.hh>
```

Inheritance diagram for mln::accu::meta::math::sup:



### 10.31.1 Detailed Description

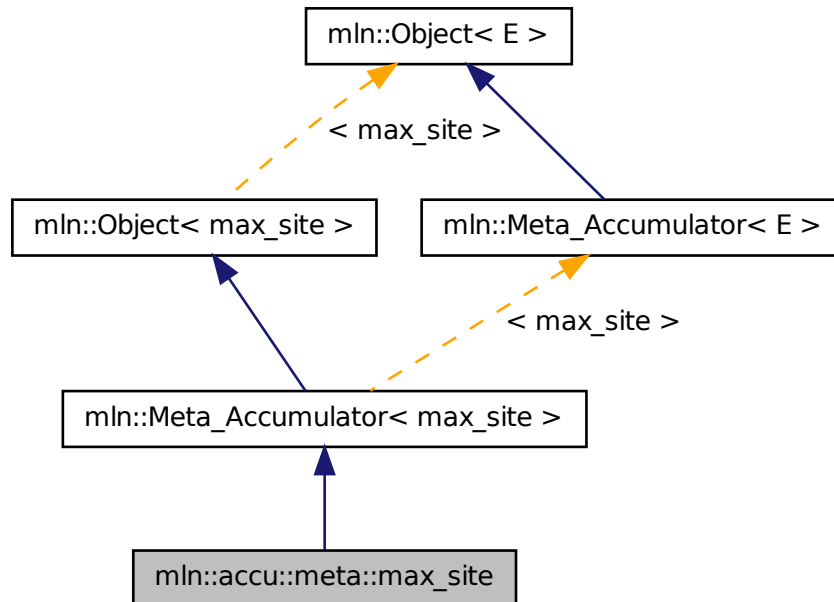
Meta accumulator for sup.

## 10.32 mln::accu::meta::max\_site Struct Reference

Meta accumulator for [max\\_site](#).

```
#include <max_site.hh>
```

Inheritance diagram for mln::accu::meta::max\_site:



### 10.32.1 Detailed Description

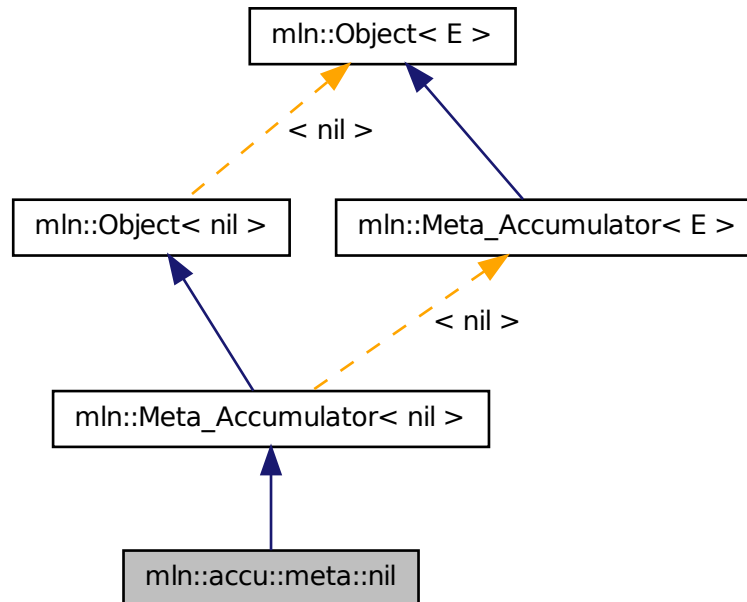
Meta accumulator for [max\\_site](#).

## 10.33 mln::accu::meta::nil Struct Reference

Meta accumulator for nil.

```
#include <nil.hh>
```

Inheritance diagram for mln::accu::meta::nil:



### 10.33.1 Detailed Description

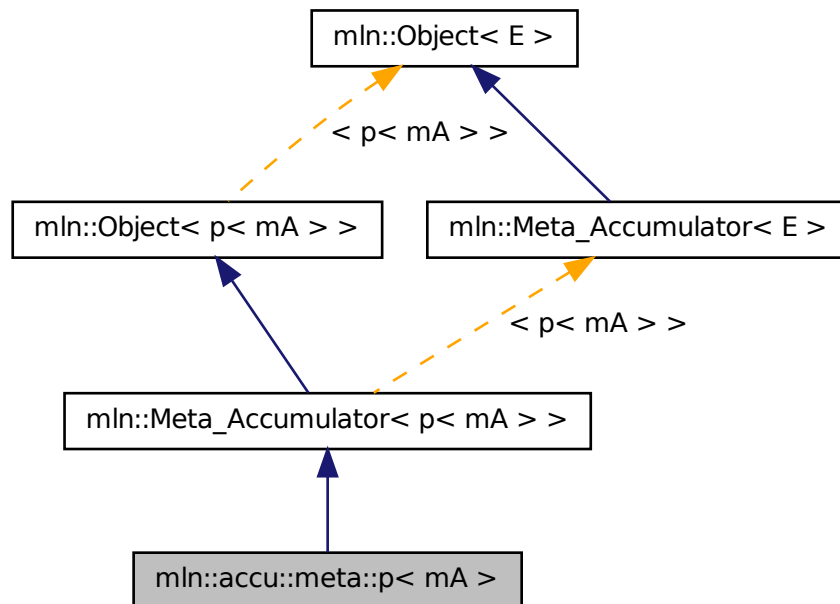
Meta accumulator for nil.

## 10.34 mln::accu::meta::p< mA > Struct Template Reference

Meta accumulator for p.

```
#include <p.hh>
```

Inheritance diagram for mln::accu::meta::p< mA >:



### 10.34.1 Detailed Description

```
template<typename mA> struct mln::accu::meta::p< mA >
```

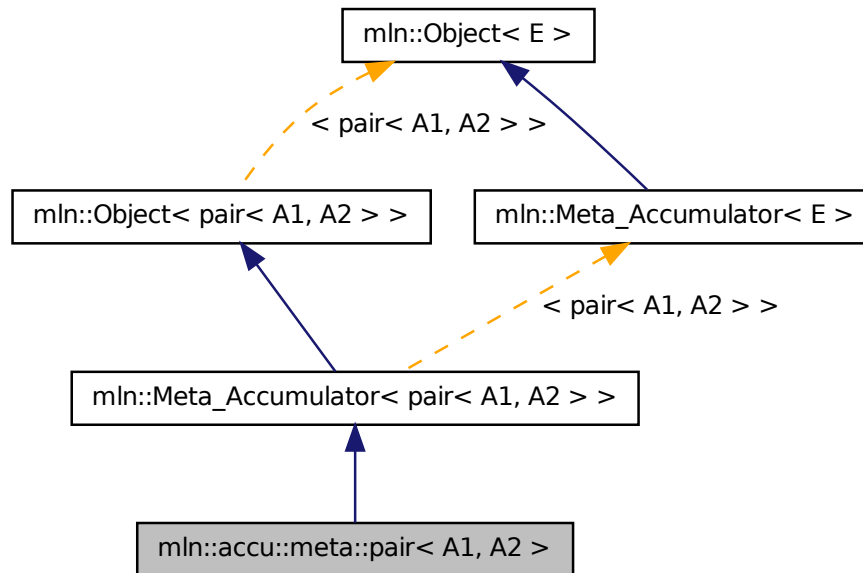
Meta accumulator for p.

## 10.35 mln::accu::meta::pair< A1, A2 > Struct Template Reference

Meta accumulator for pair.

```
#include <pair.hh>
```

Inheritance diagram for `mln::accu::meta::pair< A1, A2 >`:



### 10.35.1 Detailed Description

```
template<typename A1, typename A2> struct mln::accu::meta::pair< A1, A2 >
```

Meta accumulator for pair.

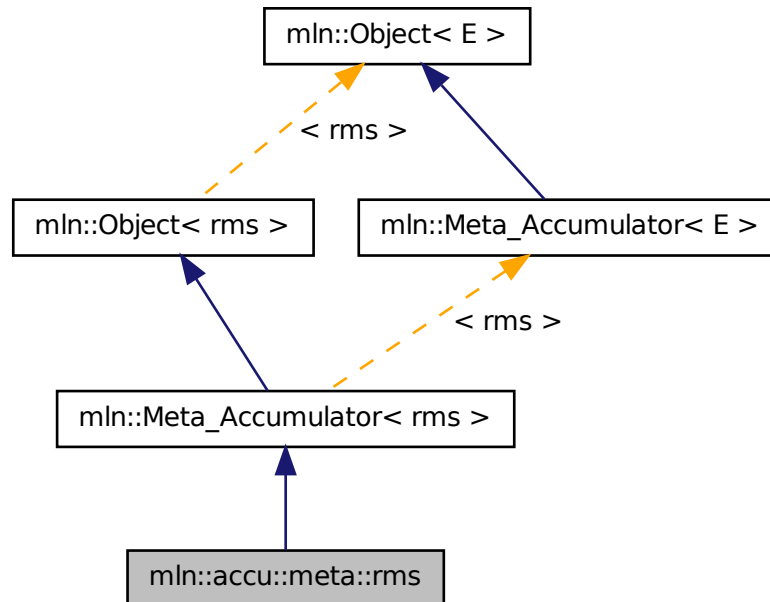
## 10.36 mln::accu::meta::rms Struct Reference

Meta accumulator for rms.

```
#include <rms.hh>
```



Inheritance diagram for mln::accu::meta::rms:



### 10.36.1 Detailed Description

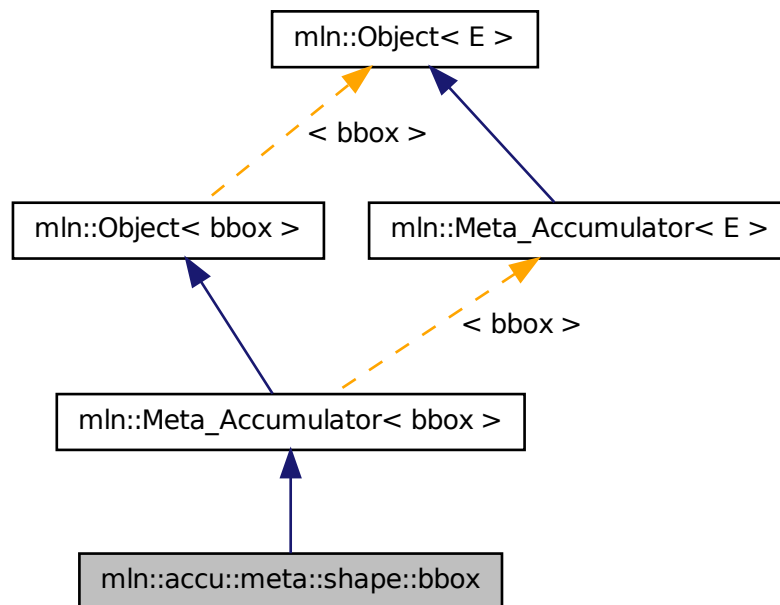
Meta accumulator for rms.

## 10.37 mln::accu::meta::shape::bbox Struct Reference

Meta accumulator for bbox.

```
#include <bbox.hh>
```

Inheritance diagram for mln::accu::meta::shape::bbox:



### 10.37.1 Detailed Description

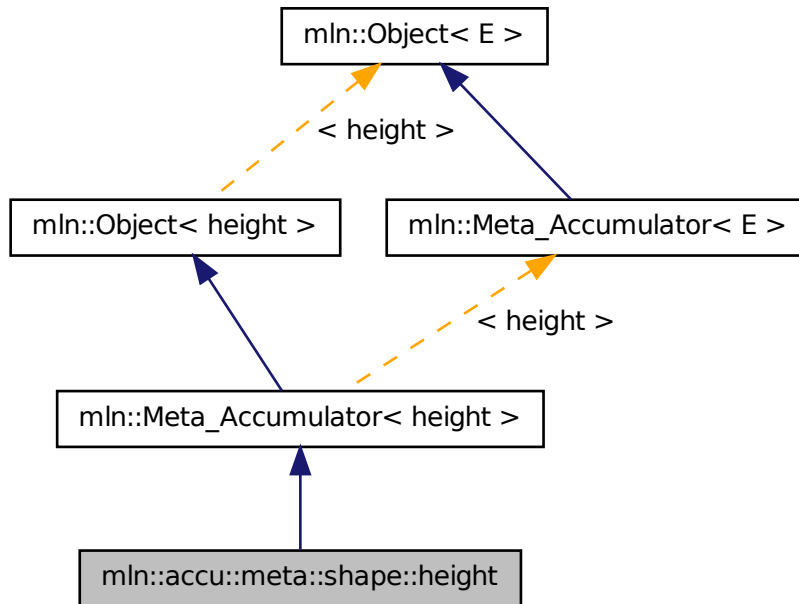
Meta accumulator for bbox.

## 10.38 mln::accu::meta::shape::height Struct Reference

Meta accumulator for height.

```
#include <height.hh>
```

Inheritance diagram for mln::accu::meta::shape::height:



### 10.38.1 Detailed Description

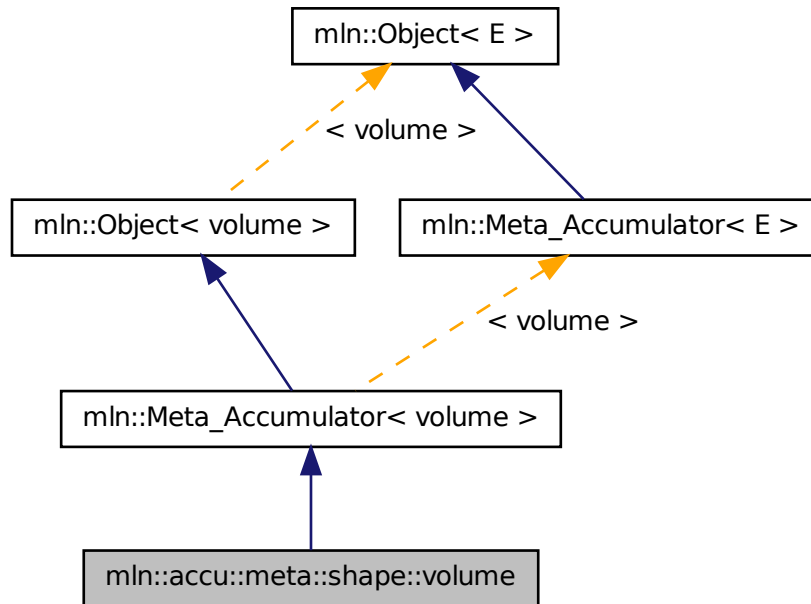
Meta accumulator for height.

## 10.39 mln::accu::meta::shape::volume Struct Reference

Meta accumulator for volume.

```
#include <volume.hh>
```

Inheritance diagram for mln::accu::meta::shape::volume:



### 10.39.1 Detailed Description

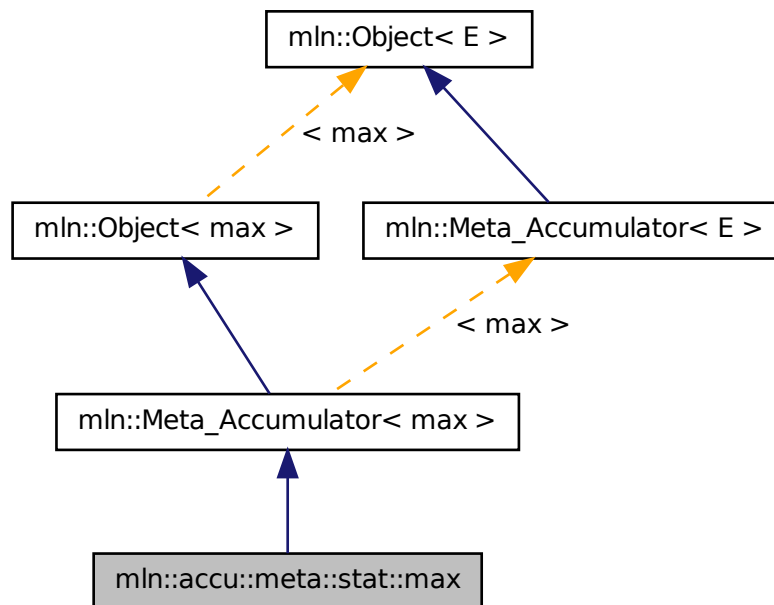
Meta accumulator for volume.

## 10.40 mln::accu::meta::stat::max Struct Reference

Meta accumulator for max.

```
#include <max.hh>
```

Inheritance diagram for mln::accu::meta::stat::max:



### 10.40.1 Detailed Description

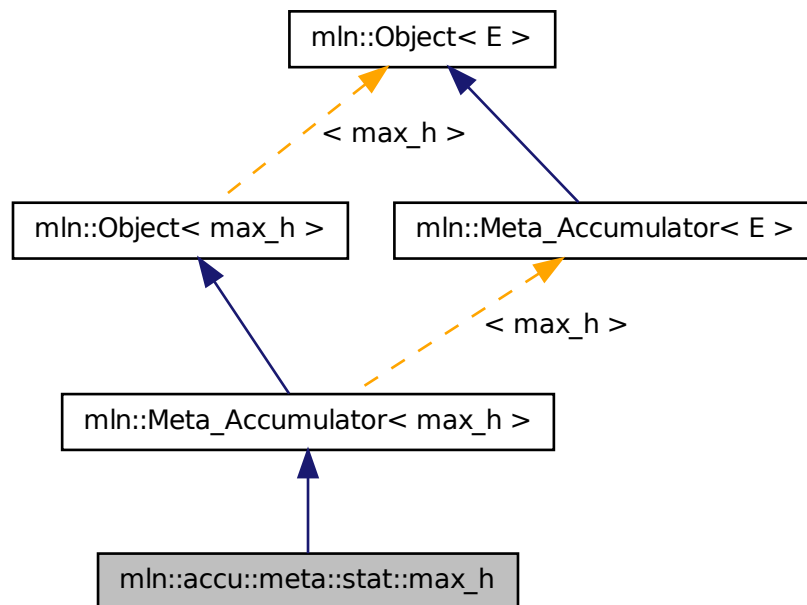
Meta accumulator for max.

## 10.41 mln::accu::meta::stat::max\_h Struct Reference

Meta accumulator for max.

```
#include <max_h.hh>
```

Inheritance diagram for mln::accu::meta::stat::max\_h:



### 10.41.1 Detailed Description

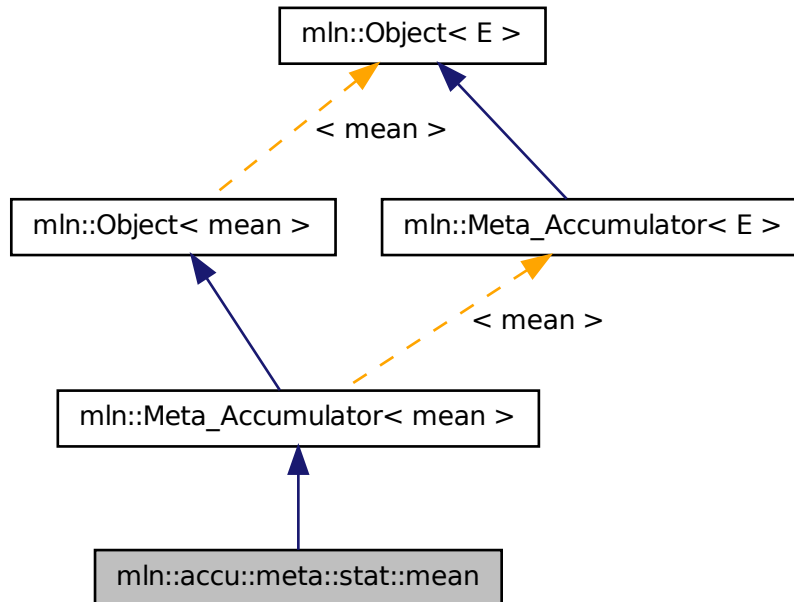
Meta accumulator for max.

## 10.42 mln::accu::meta::stat::mean Struct Reference

Meta accumulator for mean.

```
#include <mean.hh>
```

Inheritance diagram for mln::accu::meta::stat::mean:



### 10.42.1 Detailed Description

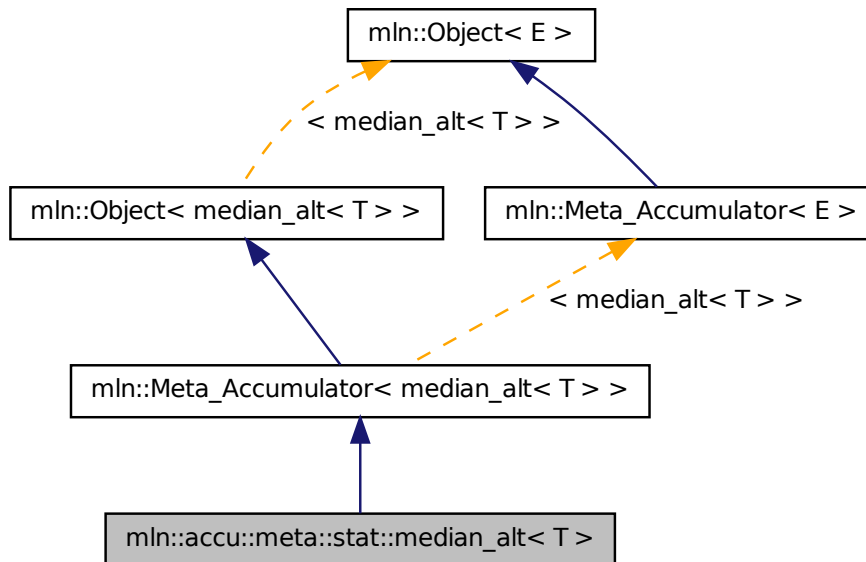
Meta accumulator for mean.

## 10.43 mln::accu::meta::stat::median\_alt< T > Struct Template Reference

Meta accumulator for [median\\_alt](#).

```
#include <median_alt.hh>
```

Inheritance diagram for `mln::accu::meta::stat::median_alt< T >`:



### 10.43.1 Detailed Description

```
template<typename T> struct mln::accu::meta::stat::median_alt< T >
```

Meta accumulator for [median\\_alt](#).

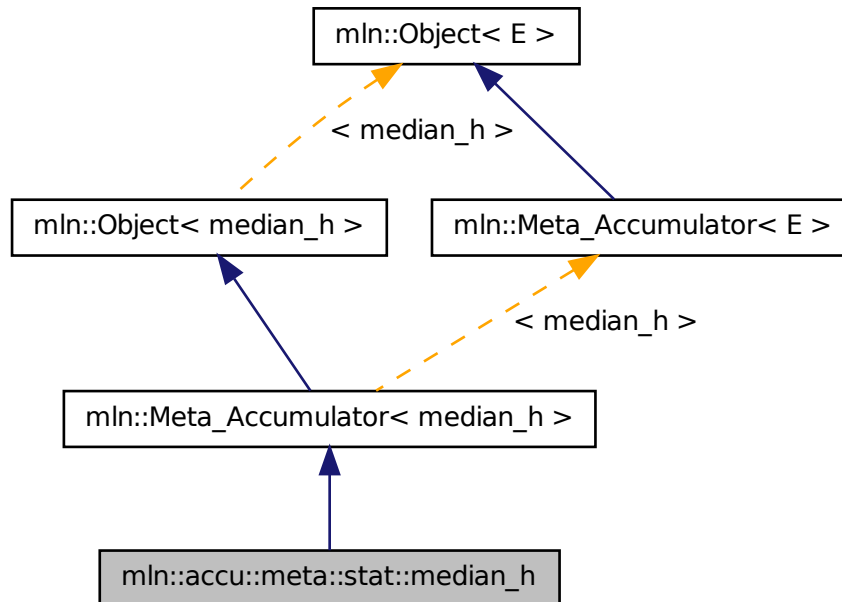
## 10.44 mln::accu::meta::stat::median\_h Struct Reference

Meta accumulator for [median\\_h](#).

```
#include <median_h.hh>
```



Inheritance diagram for mln::accu::meta::stat::median\_h:



### 10.44.1 Detailed Description

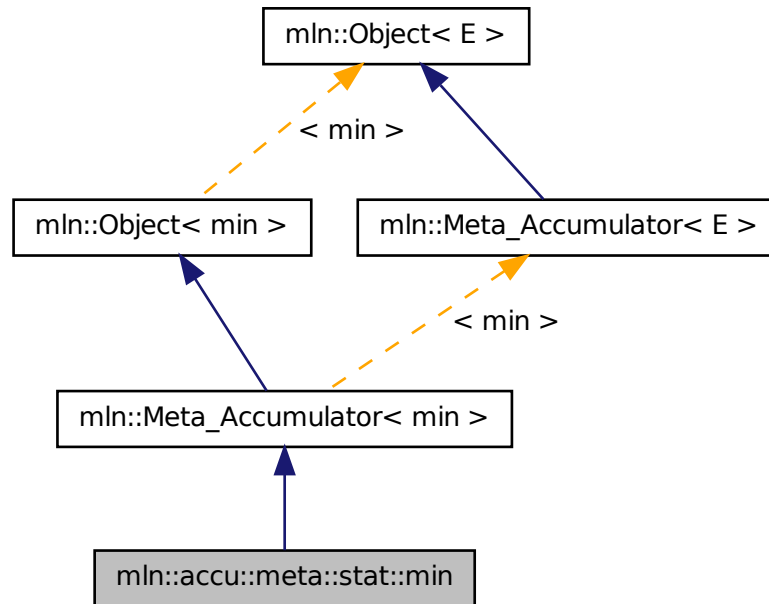
Meta accumulator for [median\\_h](#).

## 10.45 mln::accu::meta::stat::min Struct Reference

Meta accumulator for min.

```
#include <min.hh>
```

Inheritance diagram for mln::accu::meta::stat::min:



### 10.45.1 Detailed Description

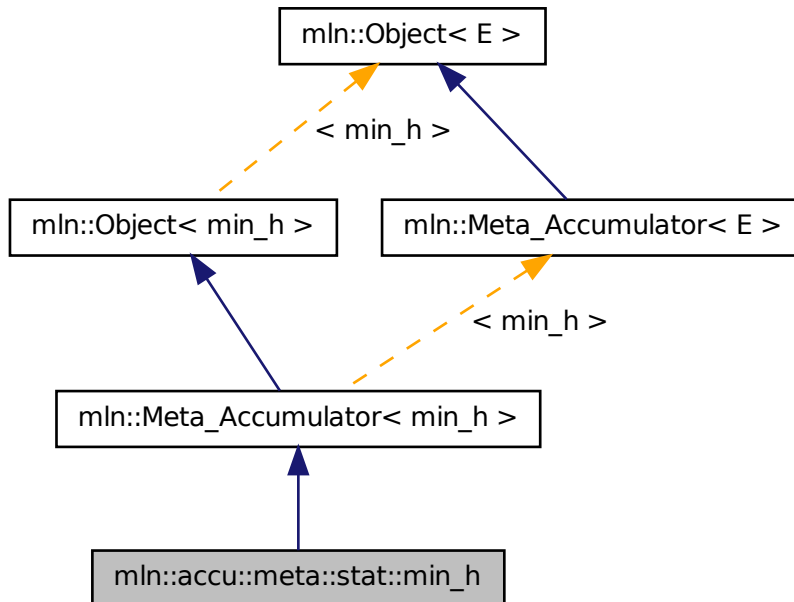
Meta accumulator for min.

## 10.46 mln::accu::meta::stat::min\_h Struct Reference

Meta accumulator for min.

```
#include <min_h.hh>
```

Inheritance diagram for mln::accu::meta::stat::min\_h:



### 10.46.1 Detailed Description

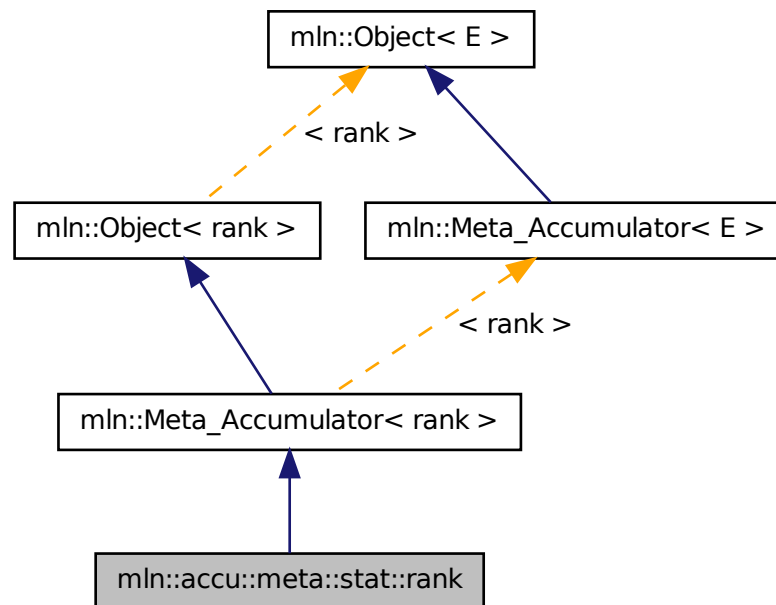
Meta accumulator for min.

## 10.47 mln::accu::meta::stat::rank Struct Reference

Meta accumulator for rank.

```
#include <rank.hh>
```

Inheritance diagram for mln::accu::meta::stat::rank:



### 10.47.1 Detailed Description

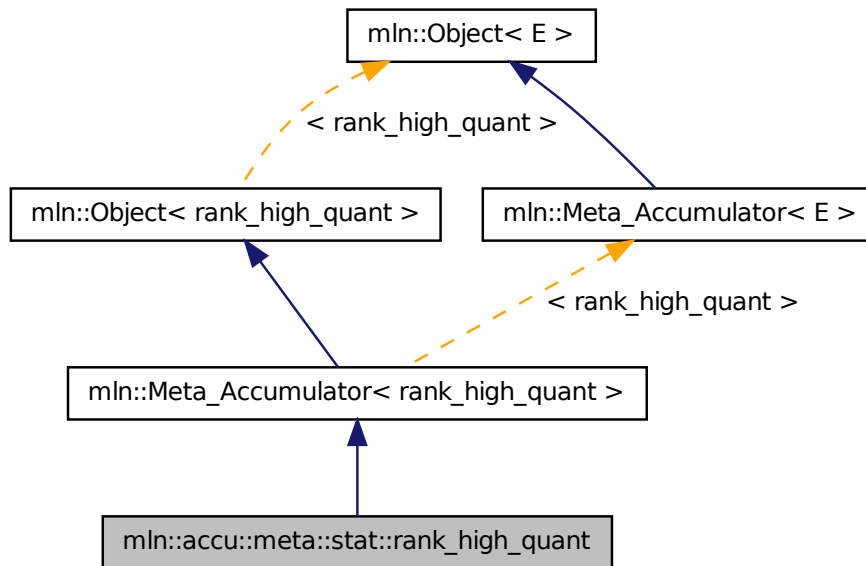
Meta accumulator for rank.

## 10.48 mln::accu::meta::stat::rank\_high\_quant Struct Reference

Meta accumulator for [rank\\_high\\_quant](#).

```
#include <rank_high_quant.hh>
```

Inheritance diagram for mln::accu::meta::stat::rank\_high\_quant:



### 10.48.1 Detailed Description

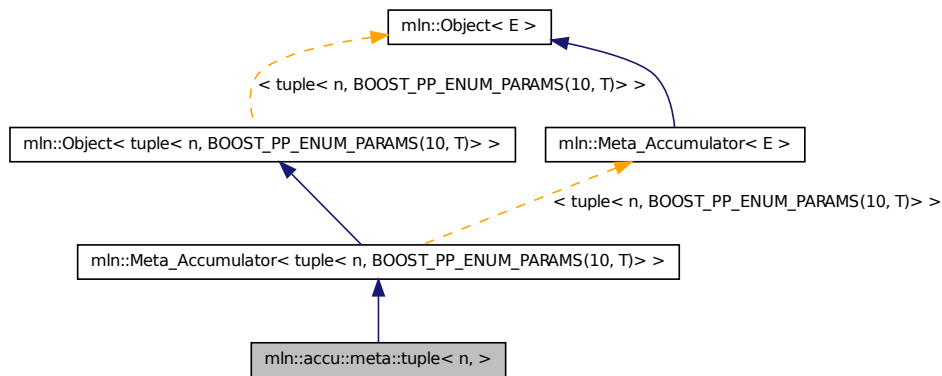
Meta accumulator for [rank\\_high\\_quant](#).

## 10.49 mln::accu::meta::tuple< n, > Struct Template Reference

Meta accumulator for tuple.

```
#include <tuple.hh>
```

Inheritance diagram for `mln::accu::meta::tuple< n, >`:



### 10.49.1 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, >
```

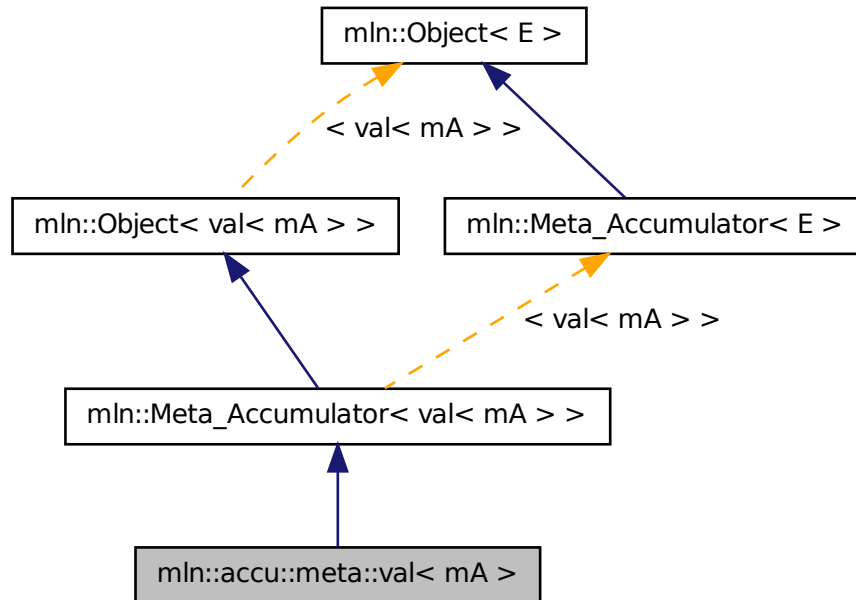
Meta accumulator for tuple.

## 10.50 `mln::accu::meta::val< mA >` Struct Template Reference

Meta accumulator for val.

```
#include <v.hh>
```

Inheritance diagram for mln::accu::meta::val< mA > :



### 10.50.1 Detailed Description

```
template<typename mA> struct mln::accu::meta::val< mA >
```

Meta accumulator for val.

## 10.51 mln::accu::nil< T > Struct Template Reference

Define an accumulator that does nothing.

```
#include <nil.hh>
```

Inherits base< util::ignore, nil< T > >.

### Public Member Functions

- bool `is_valid` () const  
*Check whether this accu is able to return a result.*
- void `take_as_init` (const T &t)  
*Take as initialization the value t.*

- void `take_n_times` (unsigned `n`, const `T` &`t`)  
*Take  $n$  times the value  $t$ .*
- `util::ignore to_result` () const  
*Get the value of the accumulator.*
- void `init` ()  
*Manipulators.*

### 10.51.1 Detailed Description

`template<typename T> struct mln::accu::nil< T >`

Define an accumulator that does nothing.

### 10.51.2 Member Function Documentation

**10.51.2.1** `template<typename T > void mln::accu::nil< T >::init ( ) [inline]`

Manipulators.

**10.51.2.2** `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.

**10.51.2.3** `void mln::Accumulator< nil< T > >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.51.2.4** `void mln::Accumulator< nil< T > >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.51.2.5** `template<typename T > util::ignore mln::accu::nil< T >::to_result ( ) const [inline]`

Get the value of the accumulator.



## 10.52 mln::accu::p< A > Struct Template Reference

Generic p of accumulators.

```
#include <p.hh>
```

Inherits base< const A::result &, p< A > >.

### Public Member Functions

- bool [is\\_valid](#) () const  
*Check whether this accu is able to return a result.*
- void [take\\_as\\_init](#) (const T &t)  
*Take as initialization the value t.*
- void [take\\_n\\_times](#) (unsigned n, const T &t)  
*Take n times the value t.*
- const A::result & [to\\_result](#) () const  
*Get the value of the accumulator.*
- void [init](#) ()  
*Manipulators.*

### 10.52.1 Detailed Description

```
template<typename A> struct mln::accu::p< A >
```

Generic p of accumulators. The parameter V is the type of values.

### 10.52.2 Member Function Documentation

**10.52.2.1** `template<typename A> void mln::accu::p< A >::init ( ) [inline]`

Manipulators.

**10.52.2.2** `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.

**10.52.2.3** `void mln::Accumulator< p< A > >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value t.

Dev note: this is a final method; override if needed by take\_as\_init\_ (ending with '\_').

**10.52.2.4** `void mln::Accumulator< p< A > >::take_n_times ( unsigned n, const T & t )`  
**[inherited]**

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.52.2.5** `template<typename A > const A::result & mln::accu::p< A >::to_result ( ) const`  
**[inline]**

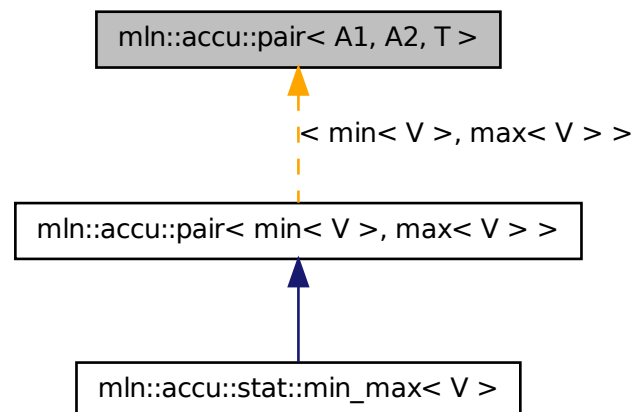
Get the value of the accumulator.

## 10.53 `mln::accu::pair< A1, A2, T >` Struct Template Reference

Generic pair of accumulators.

```
#include <pair.hh>
```

Inheritance diagram for `mln::accu::pair< A1, A2, T >`:



### Public Member Functions

- `A1::result first () const`  
*Return the result of the first accumulator.*
- `A1 first_accu () const`  
*Return the first accumulator.*
- `bool is_valid () const`  
*Check whether this accu is able to return a result.*

- A2::result `second` () const  
*Return the result of the second accumulator.*
- A2 `second_accu` () const  
*Return the second accumulator.*
- void `take_as_init` (const T &t)  
*Take as initialization the value t.*
- void `take_n_times` (unsigned n, const T &t)  
*Take n times the value t.*
  
- void `init` ()  
*Manipulators.*
  
- std::pair< typename A1::result, typename A2::result > `to_result` () const  
*Get the value of the accumulator.*

### 10.53.1 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.

### 10.53.2 Member Function Documentation

**10.53.2.1** `template<typename A1 , typename A2 , typename T > A1::result mln::accu::pair< A1, A2, T >::first ( ) const [inline]`

Return the result of the first accumulator.

**10.53.2.2** `template<typename A1 , typename A2 , typename T > A1 mln::accu::pair< A1, A2, T >::first_accu ( ) const [inline]`

Return the first accumulator.

**10.53.2.3** `template<typename A1 , typename A2 , typename T > void mln::accu::pair< A1, A2, T >::init ( ) [inline]`

Manipulators.

**10.53.2.4** `template<typename A1 , typename A2 , typename T > bool mln::accu::pair< A1, A2, T >::is_valid ( ) const [inline]`

Check whether this accu is able to return a result.

Always true here.

**10.53.2.5** `template<typename A1 , typename A2 , typename T > A2::result mln::accu::pair< A1, A2, T >::second ( ) const [inline]`

Return the result of the second accumulator.

**10.53.2.6** `template<typename A1 , typename A2 , typename T > A2 mln::accu::pair< A1, A2, T >::second_accu ( ) const [inline]`

Return the second accumulator.

**10.53.2.7** `void mln::Accumulator< pair< A1, A2, T > >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.53.2.8** `void mln::Accumulator< pair< A1, A2, T > >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.53.2.9** `template<typename A1 , typename A2 , typename T > std::pair< typename A1::result, typename A2::result > mln::accu::pair< A1, A2, T >::to_result ( ) const [inline]`

Get the value of the accumulator.

## 10.54 mln::accu::rms< T, V > Struct Template Reference

Generic root mean square accumulator class.

```
#include <rms.hh>
```

Inherits base< V, rms< T, V > >.

### Public Member Functions

- bool `is_valid` () const  
*Check whether this accu is able to return a result.*
- void `take_as_init` (const T &t)

Take as initialization the value  $t$ .

- void `take_n_times` (unsigned  $n$ , const T & $t$ )

Take  $n$  times the value  $t$ .

- V `to_result` () const

Get the value of the accumulator.

- void `init` ()

Manipulators.

### 10.54.1 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.

### 10.54.2 Member Function Documentation

**10.54.2.1** `template<typename T , typename V > void mln::accu::rms< T, V >::init ( )`  
**[inline]**

Manipulators.

References `mln::literal::zero`.

**10.54.2.2** `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.

**10.54.2.3** `void mln::Accumulator< rms< T, V > >::take_as_init ( const T & t )` **[inherited]**

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.54.2.4** `void mln::Accumulator< rms< T, V > >::take_n_times ( unsigned n, const T & t )`  
**[inherited]**

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.54.2.5** `template<typename T , typename V > V mln::accu::rms< T, V >::to_result ( ) const [inline]`

Get the value of the accumulator.

## 10.55 mln::accu::shape::bbox< P > Struct Template Reference

Generic bounding box accumulator class.

```
#include <bbox.hh>
```

Inherits `base< const box< P > &, bbox< P > >`.

### Public Member Functions

- `bool is_valid () const`  
*Check whether this accu is able to return a result.*
- `void take_as_init (const T &t)`  
*Take as initialization the value  $t$ .*
- `void take_n_times (unsigned n, const T &t)`  
*Take  $n$  times the value  $t$ .*
- `const box< P > & to_result () const`  
*Get the value of the accumulator.*
- `void init ()`  
*Manipulators.*

### 10.55.1 Detailed Description

`template<typename P> struct mln::accu::shape::bbox< P >`

Generic bounding box accumulator class. The parameter `P` is the type of points.

### 10.55.2 Member Function Documentation

**10.55.2.1** `template<typename P > void mln::accu::shape::bbox< P >::init ( ) [inline]`

Manipulators.

**10.55.2.2** `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.

**10.55.2.3 void mln::Accumulator< bbox< P > >::take\_as\_init ( const T & t ) [inherited]**

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.55.2.4 void mln::Accumulator< bbox< P > >::take\_n\_times ( unsigned n, const T & t ) [inherited]**

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.55.2.5 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()`.

**10.56 mln::accu::shape::height< I > Struct Template Reference**

Height accumulator.

```
#include <height.hh>
```

Inherits `base< unsigned, height< I > >`.

**Public Types**

- typedef `util::pix< I > argument`  
*The accumulated data type.*
- typedef `argument::value value`  
*The value type associated to the pixel type.*

**Public Member Functions**

- bool `is_valid () const`  
*Check whether this accu is able to return a result.*
- void `take_as_init (const T &t)`  
*Take as initialization the value  $t$ .*
- void `take_n_times (unsigned n, const T &t)`  
*Take  $n$  times the value  $t$ .*
- unsigned `to_result () const`  
*Get the value of the accumulator.*

- void `init()`  
*Manipulators.*
- void `set_value(unsigned h)`  
*Force the value of the counter to  $h$ .*

### 10.56.1 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.

### 10.56.2 Member Typedef Documentation

**10.56.2.1** `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](#)

**10.56.2.2** `template<typename I> typedef argument::value mln::accu::shape::height< I >::value`

The value type associated to the pixel type.

### 10.56.3 Member Function Documentation

**10.56.3.1** `template<typename I> void mln::accu::shape::height< I >::init( ) [inline]`

Manipulators.

**10.56.3.2** `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.

**10.56.3.3** `template<typename I> void mln::accu::shape::height< I >::set_value( unsigned h ) [inline]`

Force the value of the counter to  $h$ .



**10.56.3.4** `void mln::Accumulator< height< I > >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.56.3.5** `void mln::Accumulator< height< I > >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.56.3.6** `template<typename I > unsigned mln::accu::shape::height< I >::to_result ( ) const [inline]`

Get the value of the accumulator.

**10.57** `mln::accu::shape::volume< I >` Struct Template Reference

Volume accumulator class.

```
#include <volume.hh>
```

Inherits `base< unsigned, volume< I > >`.

**Public Types**

- typedef `util::pix< I > argument`  
*The accumulated data type.*
- typedef `argument::value value`  
*The value type associated to the pixel type.*

**Public Member Functions**

- bool `is_valid () const`  
*Check whether this `accu` is able to return a result.*
- void `take_as_init (const T &t)`  
*Take as initialization the value `t`.*
- void `take_n_times (unsigned n, const T &t)`  
*Take `n` times the value `t`.*
- unsigned `to_result () const`  
*Get the value of the accumulator.*

- void `init()`  
*Manipulators.*
- void `set_value(unsigned v)`  
*Force the value of the counter to v.*

### 10.57.1 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.

### 10.57.2 Member Typedef Documentation

**10.57.2.1 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](#)

**10.57.2.2 template<typename I> typedef argument::value mln::accu::shape::volume< I >::value**

The value type associated to the pixel type.

### 10.57.3 Member Function Documentation

**10.57.3.1 template<typename I> void mln::accu::shape::volume< I >::init( ) [inline]**

Manipulators.

References `mln::literal::zero`.

**10.57.3.2 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.

**10.57.3.3 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`.

**10.57.3.4 void mln::Accumulator< volume< I > >::take\_as\_init ( const T & t ) [inherited]**

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by take\_as\_init\_ (ending with '\_').

**10.57.3.5 void mln::Accumulator< volume< I > >::take\_n\_times ( unsigned n, const T & t ) [inherited]**

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by take\_as\_init\_ (ending with '\_').

**10.57.3.6 template<typename I > unsigned mln::accu::shape::volume< I >::to\_result ( ) const [inline]**

Get the value of the accumulator.

**10.58 mln::accu::site\_set::rectangularity< P > Class Template Reference**

Compute the rectangularity of a site set.

```
#include <rectangularity.hh>
```

Inherits couple< accu::shape::bbox< P >, accu::math::count< P >, float, rectangularity< P > >.

**Public Member Functions**

- A2::result [area](#) () const  
*Return the site set area.*
- A1::result [bbox](#) () const  
*Return the site set bounding box.*
- [rectangularity](#) ()  
*Constructor.*
- template<typename T >  
void [take\\_as\\_init](#) (const T &t)  
*Take as initialization the value  $t$ .*
- template<typename T >  
void [take\\_n\\_times](#) (unsigned n, const T &t)  
*Take  $n$  times the value  $t$ .*
- result [to\\_result](#) () const  
*Return the rectangularity value.*

### 10.58.1 Detailed Description

`template<typename P> class mln::accu::site_set::rectangularity< P >`

Compute the rectangularity of a site set.

### 10.58.2 Constructor & Destructor Documentation

**10.58.2.1** `template<typename P > mln::accu::site_set::rectangularity< P >::rectangularity ( )`  
`[inline]`

Constructor.

### 10.58.3 Member Function Documentation

**10.58.3.1** `template<typename P > rectangularity< P >::A2::result`  
`mln::accu::site_set::rectangularity< P >::area ( ) const [inline]`

Return the site set area.

**10.58.3.2** `template<typename P > rectangularity< P >::A1::result`  
`mln::accu::site_set::rectangularity< P >::bbox ( ) const [inline]`

Return the site set bounding box.

**10.58.3.3** `template<typename E > template<typename T > void mln::Accumulator< E`  
`>::take_as_init ( const T & t ) [inherited]`

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

**10.58.3.4** `template<typename E > template<typename T > void mln::Accumulator< E`  
`>::take_n_times ( unsigned n, const T & t ) [inherited]`

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

**10.58.3.5** `template<typename P > rectangularity< P >::result mln::accu::site_`  
`set::rectangularity< P >::to_result ( ) const [inline]`

Return the rectangularity value.

## 10.59 mln::accu::stat::deviation< T, S, M > Struct Template Reference

Generic standard deviation accumulator class.

```
#include <deviation.hh>
```

Inherits base< M, deviation< T, S, M > >.

### Public Member Functions

- bool `is_valid` () const  
*Check whether this accu is able to return a result.*
- void `take_as_init` (const T &t)  
*Take as initialization the value t.*
- void `take_n_times` (unsigned n, const T &t)  
*Take n times the value t.*
- M `to_result` () const  
*Get the value of the accumulator.*
- void `init` ()  
*Manipulators.*

### 10.59.1 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.

### 10.59.2 Member Function Documentation

**10.59.2.1** `template<typename T, typename S, typename M > void mln::accu::stat::deviation< T, S, M >::init ( ) [inline]`

Manipulators.

**10.59.2.2** `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.

**10.59.2.3** `void mln::Accumulator< deviation< T, S, M > >::take_as_init ( const T & t )`  
**[inherited]**

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.59.2.4** `void mln::Accumulator< deviation< T, S, M > >::take_n_times ( unsigned n, const T & t )` **[inherited]**

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.59.2.5** `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.

## 10.60 mln::accu::stat::max< T > Struct Template Reference

Generic max accumulator class.

```
#include <max.hh>
```

Inherits base< const T &, max< T > >.

### Public Member Functions

- bool [is\\_valid](#) () const  
*Check whether this accu is able to return a result.*
- void [set\\_value](#) (const T &t)  
*Force the value of the min to  $t$ .*
- void [take\\_as\\_init](#) (const T &t)  
*Take as initialization the value  $t$ .*
- void [take\\_n\\_times](#) (unsigned n, const T &t)  
*Take  $n$  times the value  $t$ .*
- const T & [to\\_result](#) () const  
*Get the value of the accumulator.*
- void [init](#) ()  
*Manipulators.*

### 10.60.1 Detailed Description

```
template<typename T> struct mln::accu::stat::max< T >
```

Generic max accumulator class. The parameter `T` is the type of values.

### 10.60.2 Member Function Documentation

**10.60.2.1** `template<typename T> void mln::accu::stat::max< T >::init ( ) [inline]`

Manipulators.

**10.60.2.2** `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.

**10.60.2.3** `template<typename T> void mln::accu::stat::max< T >::set_value ( const T & t ) [inline]`

Force the value of the min to `t`.

**10.60.2.4** `void mln::Accumulator< max< T > >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.60.2.5** `void mln::Accumulator< max< T > >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.60.2.6** `template<typename T> const T & mln::accu::stat::max< T >::to_result ( ) const [inline]`

Get the value of the accumulator.

## 10.61 mln::accu::stat::max\_h< V > Struct Template Reference

Generic max function based on histogram over a value set with type `V`.

```
#include <max_h.hh>
```

Inherits `base< const V &, max_h< V > >`.

## Public Member Functions

- `bool is_valid () const`  
*Check whether this `accu` is able to return a result.*
- `void take_as_init (const T &t)`  
*Take as initialization the value `t`.*
- `void take_n_times (unsigned n, const T &t)`  
*Take `n` times the value `t`.*
- `const argument & to_result () const`  
*Get the value of the accumulator.*
  
- `void init ()`  
*Manipulators.*

### 10.61.1 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`.

### 10.61.2 Member Function Documentation

**10.61.2.1** `template<typename V > void mln::accu::stat::max_h< V >::init ( ) [inline]`

Manipulators.

**10.61.2.2** `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.

**10.61.2.3** `void mln::Accumulator< max_h< V > >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.61.2.4** `void mln::Accumulator< max_h< V > >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).



**10.61.2.5** `template<typename V > const max_h< V >::argument & mln::accu::stat::max_h< V >::to_result ( ) const [inline]`

Get the value of the accumulator.

## 10.62 mln::accu::stat::mean< T, S, M > Struct Template Reference

Generic mean accumulator class.

```
#include <mean.hh>
```

Inherits base< M, mean< T, S, M > >.

### Public Member Functions

- `accu::math::count< T >::result count ( ) const`  
*Get the cardinality.*
- `bool is_valid ( ) const`  
*Check whether this accu is able to return a result.*
- `accu::math::sum< T >::result sum ( ) const`  
*Get the sum of values.*
- `void take_as_init (const T &t)`  
*Take as initialization the value  $t$ .*
- `void take_n_times (unsigned n, const T &t)`  
*Take  $n$  times the value  $t$ .*
- `M to_result ( ) const`  
*Get the value of the accumulator.*
- `void init ( )`  
*Manipulators.*

### 10.62.1 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`.

## 10.62.2 Member Function Documentation

**10.62.2.1** `template<typename T , typename S , typename M > accu::math::count< T >::result  
mln::accu::stat::mean< T, S, M >::count ( ) const [inline]`

Get the cardinality.

**10.62.2.2** `template<typename T , typename S , typename M > void mln::accu::stat::mean< T, S,  
M >::init ( ) [inline]`

Manipulators.

**10.62.2.3** `template<typename T , typename S , typename M > bool mln::accu::stat::mean< T, S,  
M >::is_valid ( ) const [inline]`

Check whether this accu is able to return a result.

Always true here.

**10.62.2.4** `template<typename T , typename S , typename M > accu::math::sum< T >::result  
mln::accu::stat::mean< T, S, M >::sum ( ) const [inline]`

Get the sum of values.

**10.62.2.5** `void mln::Accumulator< mean< T, S, M > >::take_as_init ( const T & t )  
[inherited]`

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.62.2.6** `void mln::Accumulator< mean< T, S, M > >::take_n_times ( unsigned n, const T & t  
) [inherited]`

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.62.2.7** `template<typename T , typename S , typename M > M mln::accu::stat::mean< T, S, M  
>::to_result ( ) const [inline]`

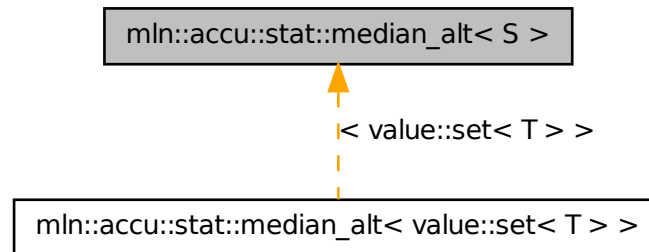
Get the value of the accumulator.

## 10.63 mln::accu::stat::median\_alt< S > Struct Template Reference

Generic `median_alt` function based on histogram over a value set with type  $S$ .

```
#include <median_alt.hh>
```

Inheritance diagram for mln::accu::stat::median\_alt< S >:



## Public Member Functions

- bool [is\\_valid](#) () const  
*Check whether this accu is able to return a result.*
- void [take\\_as\\_init](#) (const T &t)  
*Take as initialization the value t.*
- void [take\\_n\\_times](#) (unsigned n, const T &t)  
*Take n times the value t.*
- const argument & [to\\_result](#) () const  
*Get the value of the accumulator.*
- void [take](#) (const argument &t)  
*Manipulators.*

### 10.63.1 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.

### 10.63.2 Member Function Documentation

**10.63.2.1** `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.

**10.63.2.2** `template<typename S > void mln::accu::stat::median_alt< S >::take ( const argument & t ) [inline]`

Manipulators.

**10.63.2.3** `void mln::Accumulator< median_alt< S > >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.63.2.4** `void mln::Accumulator< median_alt< S > >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.63.2.5** `template<typename S > const median_alt< S >::argument & mln::accu::stat::median_alt< S >::to_result ( ) const [inline]`

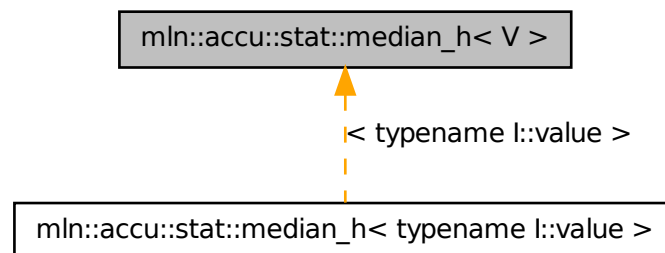
Get the value of the accumulator.

## 10.64 `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.*

### 10.64.1 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.

### 10.64.2 Member Function Documentation

**10.64.2.1 template<typename V > void mln::accu::stat::median\_h< V >::init ( ) [inline]**

Manipulators.

**10.64.2.2 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.

**10.64.2.3 void mln::Accumulator< median\_h< V > >::take\_as\_init ( const T & t ) [inherited]**

Take as initialization the value t.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '\_').

**10.64.2.4 void mln::Accumulator< median\_h< V > >::take\_n\_times ( unsigned n, const T & t ) [inherited]**

Take n times the value t.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '\_').

**10.64.2.5** `template<typename V > const median_h< V >::argument & mln::accu::stat::median_h< V >::to_result ( )const [inline]`

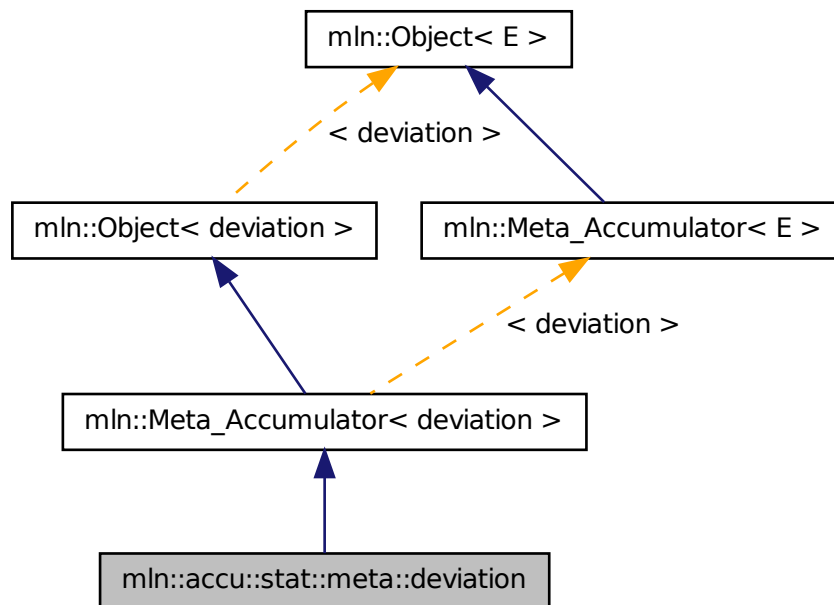
Get the value of the accumulator.

## 10.65 mln::accu::stat::meta::deviation Struct Reference

Meta accumulator for deviation.

```
#include <deviation.hh>
```

Inheritance diagram for mln::accu::stat::meta::deviation:



### 10.65.1 Detailed Description

Meta accumulator for deviation.

## 10.66 mln::accu::stat::min< T > Struct Template Reference

Generic min accumulator class.

```
#include <min.hh>
```

Inherits base< const T &, min< T > >.

## Public Member Functions

- `bool is_valid () const`  
*Check whether this accu is able to return a result.*
- `void set_value (const T &t)`  
*Force the value of the min to t.*
- `void take_as_init (const T &t)`  
*Take as initialization the value t.*
- `void take_n_times (unsigned n, const T &t)`  
*Take n times the value t.*
- `const T & to_result () const`  
*Get the value of the accumulator.*
- `void init ()`  
*Manipulators.*

### 10.66.1 Detailed Description

`template<typename T> struct mln::accu::stat::min< T >`

Generic min accumulator class. The parameter T is the type of values.

### 10.66.2 Member Function Documentation

**10.66.2.1** `template<typename T > void mln::accu::stat::min< T >::init ( ) [inline]`

Manipulators.

**10.66.2.2** `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.

**10.66.2.3** `template<typename T > void mln::accu::stat::min< T >::set_value ( const T & t ) [inline]`

Force the value of the min to t.

**10.66.2.4** `void mln::Accumulator< min< T > >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value t.

Dev note: this is a final method; override if needed by take\_as\_init\_ (ending with '\_').

**10.66.2.5** `void mln::Accumulator< min< T > >::take_n_times ( unsigned n, const T & t )`  
**[inherited]**

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.66.2.6** `template<typename T > const T & mln::accu::stat::min< T >::to_result ( ) const`  
**[inline]**

Get the value of the accumulator.

## 10.67 mln::accu::stat::min\_h< V > Struct Template Reference

Generic min function based on histogram over a value set with type  $V$ .

```
#include <min_h.hh>
```

Inherits `base< const V &, min_h< V > >`.

### Public Member Functions

- `bool is_valid () const`  
*Check whether this accu is able to return a result.*
- `void take_as_init (const T &t)`  
*Take as initialization the value  $t$ .*
- `void take_n_times (unsigned n, const T &t)`  
*Take  $n$  times the value  $t$ .*
- `const argument & to_result () const`  
*Get the value of the accumulator.*
- `void init ()`  
*Manipulators.*

### 10.67.1 Detailed Description

`template<typename V> struct mln::accu::stat::min_h< V >`

Generic min function based on histogram over a value set with type  $V$ .

### 10.67.2 Member Function Documentation

**10.67.2.1** `template<typename V > void mln::accu::stat::min_h< V >::init ( )` **[inline]**

Manipulators.



**10.67.2.2** `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.

**10.67.2.3** `void mln::Accumulator< min_h< V > >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.67.2.4** `void mln::Accumulator< min_h< V > >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.67.2.5** `template<typename V > const min_h< V >::argument & mln::accu::stat::min_h< V >::to_result ( ) const [inline]`

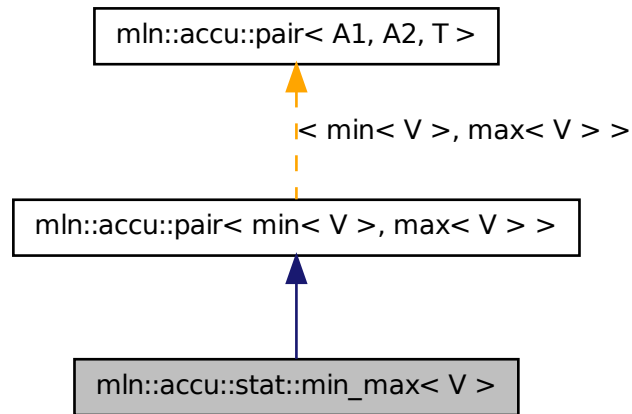
Get the value of the accumulator.

## 10.68 mln::accu::stat::min\_max< V > Struct Template Reference

Generic min and max accumulator class.

```
#include <min_max.hh>
```

Inheritance diagram for `mln::accu::stat::min_max< V >`:



## Public Member Functions

- `min< V >::result first () const`  
*Return the result of the first accumulator.*
- `min< V > first_accu () const`  
*Return the first accumulator.*
- `bool is_valid () const`  
*Check whether this accu is able to return a result.*
- `max< V >::result second () const`  
*Return the result of the second accumulator.*
- `max< V > second_accu () const`  
*Return the second accumulator.*
- `template<typename T >`  
`void take_as_init (const T &t)`  
*Take as initialization the value t.*
- `template<typename T >`  
`void take_n_times (unsigned n, const T &t)`  
*Take n times the value t.*
- `void init ()`  
*Manipulators.*

- `std::pair< typename min< V >::result, typename max< V >::result > to_result () const`  
*Get the value of the accumulator.*

## 10.68.1 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.

## 10.68.2 Member Function Documentation

**10.68.2.1** `min< V >::result mln::accu::pair< min< V >, max< V >, mln_argument(min< V > )>::first ( ) const [inherited]`

Return the result of the first accumulator.

**10.68.2.2** `min< V > mln::accu::pair< min< V >, max< V >, mln_argument(min< V > )>::first_accu ( ) const [inherited]`

Return the first accumulator.

**10.68.2.3** `void mln::accu::pair< min< V >, max< V >, mln_argument(min< V > )>::init ( ) [inherited]`

Manipulators.

**10.68.2.4** `bool mln::accu::pair< min< V >, max< V >, mln_argument(min< V > )>::is_valid ( ) const [inherited]`

Check whether this accu is able to return a result.

Always true here.

**10.68.2.5** `max< V >::result mln::accu::pair< min< V >, max< V >, mln_argument(min< V > )>::second ( ) const [inherited]`

Return the result of the second accumulator.

**10.68.2.6** `max< V > mln::accu::pair< min< V >, max< V >, mln_argument(min< V > )>::second_accu ( ) const [inherited]`

Return the second accumulator.

**10.68.2.7** `template<typename E > template<typename T > void mln::Accumulator< E >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

**10.68.2.8** `template<typename E > template<typename T > void mln::Accumulator< E >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

**10.68.2.9** `std::pair<typename min< V > ::result, typename max< V > ::result> mln::accu::pair< min< V >, max< V >, mln_argument(min< V >) >::to_result ( ) const [inherited]`

Get the value of the accumulator.

## 10.69 mln::accu::stat::rank< T > Struct Template Reference

Generic rank accumulator class.

```
#include <rank.hh>
```

Inherits `base< const T &, rank< T > >`.

### Public Member Functions

- `bool is_valid () const`  
*Check whether this `accu` is able to return a result.*
- `unsigned k () const`  
*Give the rank.*
- `void take_as_init (const T &t)`  
*Take as initialization the value  $t$ .*
- `void take_n_times (unsigned n, const T &t)`  
*Take  $n$  times the value  $t$ .*
- `const T & to_result () const`  
*Get the value of the accumulator.*
- `void init ()`  
*Manipulators.*

### 10.69.1 Detailed Description

**template<typename T> struct mln::accu::stat::rank< T >**

Generic rank accumulator class. The parameter T is the type of values.

### 10.69.2 Member Function Documentation

**10.69.2.1 template<typename T > void mln::accu::stat::rank< T >::init ( ) [inline]**

Manipulators.

**10.69.2.2 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.

**10.69.2.3 template<typename T > unsigned mln::accu::stat::rank< T >::k ( ) const [inline]**

Give the rank.

**10.69.2.4 void mln::Accumulator< rank< T > >::take\_as\_init ( const T & t ) [inherited]**

Take as initialization the value t.

Dev note: this is a final method; override if needed by take\_as\_init\_ (ending with '\_').

**10.69.2.5 void mln::Accumulator< rank< T > >::take\_n\_times ( unsigned n, const T & t ) [inherited]**

Take n times the value t.

Dev note: this is a final method; override if needed by take\_as\_init\_ (ending with '\_').

**10.69.2.6 template<typename T > const T & mln::accu::stat::rank< T >::to\_result ( ) const [inline]**

Get the value of the accumulator.

## 10.70 mln::accu::stat::rank< bool > Struct Template Reference

rank accumulator class for Boolean.

```
#include <rank_bool.hh>
```

Inherits base< bool, rank< bool > >.

## Public Member Functions

- `bool is_valid () const`  
*Check whether this accu is able to return a result.*
- `void take_as_init (const T &t)`  
*Take as initialization the value  $t$ .*
- `void take_n_times (unsigned n, const T &t)`  
*Take  $n$  times the value  $t$ .*
- `bool to_result () const`  
*Get the value of the accumulator.*
  
- `void init ()`  
*Manipulators.*

### 10.70.1 Detailed Description

`template<> struct mln::accu::stat::rank< bool >`

rank accumulator class for Boolean.

### 10.70.2 Member Function Documentation

**10.70.2.1** `void mln::accu::stat::rank< bool >::init ( ) [inline]`

Manipulators.

**10.70.2.2** `bool mln::accu::stat::rank< bool >::is_valid ( ) const [inline]`

Check whether this accu is able to return a result.

Always true here.

**10.70.2.3** `void mln::Accumulator< rank< bool > >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.70.2.4** `void mln::Accumulator< rank< bool > >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

### 10.70.2.5 bool mln::accu::stat::rank< bool >::to\_result ( ) const [inline]

Get the value of the accumulator.

## 10.71 mln::accu::stat::rank\_high\_quant< T > Struct Template Reference

Generic rank accumulator class.

```
#include <rank_high_quant.hh>
```

Inherits base< const T &, rank\_high\_quant< T > >.

### Public Member Functions

- bool [is\\_valid](#) () const  
*Check whether this accu is able to return a result.*
- void [take\\_as\\_init](#) (const T &t)  
*Take as initialization the value  $t$ .*
- void [take\\_n\\_times](#) (unsigned n, const T &t)  
*Take  $n$  times the value  $t$ .*
- const T & [to\\_result](#) () const  
*Get the value of the accumulator.*
- void [init](#) ()  
*Manipulators.*

### 10.71.1 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.

### 10.71.2 Member Function Documentation

#### 10.71.2.1 template<typename T > void mln::accu::stat::rank\_high\_quant< T >::init ( ) [inline]

Manipulators.

**10.71.2.2** `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.

**10.71.2.3** `void mln::Accumulator< rank_high_quant< T > >::take_as_init ( const T & t )`  
`[inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.71.2.4** `void mln::Accumulator< rank_high_quant< T > >::take_n_times ( unsigned n, const T & t )`  
`[inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.71.2.5** `template<typename T > const T & mln::accu::stat::rank_high_quant< T >::to_result ( )`  
`const [inline]`

Get the value of the accumulator.

## 10.72 mln::accu::stat::var< T > Struct Template Reference

Var accumulator class.

```
#include <var.hh>
```

Inherits base< algebra::mat< T::dim, T::dim, float >, var< T > >.

### Public Types

- typedef algebra::vec< dim, float > [mean\\_t](#)  
*Type equipment.*

### Public Member Functions

- bool [is\\_valid](#) () const  
*Check whether this accu returns a valid result.*
- [mean\\_t](#) [mean](#) () const  
*Get the mean vector.*
- unsigned [n\\_items](#) () const  
*Get the number of items.*



- void `take_as_init` (const T &t)  
*Take as initialization the value  $\tau$ .*
- void `take_n_times` (unsigned n, const T &t)  
*Take  $n$  times the value  $\tau$ .*
- result `to_result` () const  
*Get the accumulator result (the var value).*
- result `variance` () const  
*Get the variance matrix.*
- void `init` ()  
*Manipulators.*

### 10.72.1 Detailed Description

`template<typename T> struct mln::accu::stat::var< T >`

Var accumulator class. Parameter T is the type of vectors

### 10.72.2 Member Typedef Documentation

**10.72.2.1** `template<typename T> typedef algebra::vec<dim,float> mln::accu::stat::var< T >::mean_t`

Type equipment.

### 10.72.3 Member Function Documentation

**10.72.3.1** `template<typename T > void mln::accu::stat::var< T >::init ( ) [inline]`

Manipulators.

**10.72.3.2** `template<typename T > bool mln::accu::stat::var< T >::is_valid ( ) const [inline]`

Check whether this accu returns a valid result.

**10.72.3.3** `template<typename T > var< T >::mean_t mln::accu::stat::var< T >::mean ( ) const [inline]`

Get the mean vector.

References `mln::literal::zero`.

**10.72.3.4** `template<typename T> unsigned mln::accu::stat::var< T >::n_items ( ) const`  
`[inline]`

Get the number of items.

**10.72.3.5** `void mln::Accumulator< var< T > >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.72.3.6** `void mln::Accumulator< var< T > >::take_n_times ( unsigned n, const T & t )`  
`[inherited]`

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.72.3.7** `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`.

**10.72.3.8** `template<typename T> var< T >::result mln::accu::stat::var< T >::variance ( )`  
`const [inline]`

Get the variance matrix.

## 10.73 `mln::accu::stat::variance< T, S, R >` Struct Template Reference

Variance accumulator class.

```
#include <variance.hh>
```

Inherits `base< R, variance< T, S, R > >`.

### Public Member Functions

- `bool is_valid () const`  
*Check whether this `accu` is able to return a result.*
- `R mean () const`  
*Get the mean value.*
- `unsigned n_items () const`  
*Get the number of items.*

- R `standard_deviation ()` const  
*Get the standard deviation value.*
- S `sum ()` const  
*Get the sum value.*
- void `take_as_init (const T &t)`  
*Take as initialization the value  $t$ .*
- void `take_n_times (unsigned n, const T &t)`  
*Take  $n$  times the value  $t$ .*
- R `to_result ()` const  
*Get the accumulator result (the variance value).*
- R `var ()` const  
*Get the variance value.*
- void `init ()`  
*Manipulators.*

### 10.73.1 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`.

### 10.73.2 Member Function Documentation

**10.73.2.1** `template<typename T, typename S, typename R > void mln::accu::stat::variance< T, S, R >::init ( ) [inline]`

Manipulators.

References `mln::literal::zero`.

**10.73.2.2** `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.

**10.73.2.3** `template<typename T , typename S , typename R > R mln::accu::stat::variance< T, S, R >::mean ( ) const [inline]`

Get the mean value.

References `mln::literal::zero`.

**10.73.2.4** `template<typename T , typename S , typename R > unsigned mln::accu::stat::variance< T, S, R >::n_items ( ) const [inline]`

Get the number of items.

**10.73.2.5** `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()`.

**10.73.2.6** `template<typename T , typename S , typename R > S mln::accu::stat::variance< T, S, R >::sum ( ) const [inline]`

Get the sum value.

**10.73.2.7** `void mln::Accumulator< variance< T, S, R > >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.73.2.8** `void mln::Accumulator< variance< T, S, R > >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.73.2.9** `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()`.

**10.73.2.10** `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()`.

## 10.74 `mln::accu::tuple< A, n, >` Struct Template Reference

Generic tuple of accumulators.

```
#include <tuple.hh>
```

Inherits `base< boost::tuple< BOOST_PP_REPEAT(10, RESULT_ACCU, Le Ricard ya que ca de vrai!) >, tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T)> >`.

### Public Member Functions

- `bool is_valid () const`  
*Check whether this accu is able to return a result.*
- `void take_as_init (const T &t)`  
*Take as initialization the value  $t$ .*
- `void take_n_times (unsigned n, const T &t)`  
*Take  $n$  times the value  $t$ .*
- `res to_result () const`  
*Get the value of the accumulator.*
  
- `void init ()`  
*Manipulators.*

### 10.74.1 Detailed Description

```
template<typename A, unsigned n, BOOST_PP_ENUM_PARAMS_WITH_A_DEFAULT(10, type-  
name T, boost::tuples::null_type)> struct mln::accu::tuple< A, n, >
```

Generic tuple of accumulators. The parameter `T` is the type of values.

### 10.74.2 Member Function Documentation

**10.74.2.1** `template<typename A , unsigned n, BOOST_PP_ENUM_PARAMS(10, typename T) >  
void mln::accu::tuple< A, n, >::init ( ) [inline]`

Manipulators.

**10.74.2.2** `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.

**10.74.2.3** `void mln::Accumulator< tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T)> >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.74.2.4** `void mln::Accumulator< tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T)> >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.74.2.5** `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.

## 10.75 mln::accu::val< A > Struct Template Reference

Generic val of accumulators.

```
#include <v.hh>
```

Inherits base< const A::result &, val< A > >.

### Public Member Functions

- bool `is_valid` () const  
*Check whether this accu is able to return a result.*
- void `take_as_init` (const T &t)  
*Take as initialization the value  $t$ .*
- void `take_n_times` (unsigned n, const T &t)  
*Take  $n$  times the value  $t$ .*
- const A::result & `to_result` () const  
*Get the value of the accumulator.*
- void `init` ()  
*Manipulators.*

### 10.75.1 Detailed Description

```
template<typename A> struct mln::accu::val< A >
```

Generic val of accumulators.

### 10.75.2 Member Function Documentation

**10.75.2.1** `template<typename A > void mln::accu::val< A >::init ( ) [inline]`

Manipulators.

**10.75.2.2** `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.

**10.75.2.3** `void mln::Accumulator< val< A > >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.75.2.4** `void mln::Accumulator< val< A > >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.75.2.5** `template<typename A > const A::result & mln::accu::val< A >::to_result ( ) const [inline]`

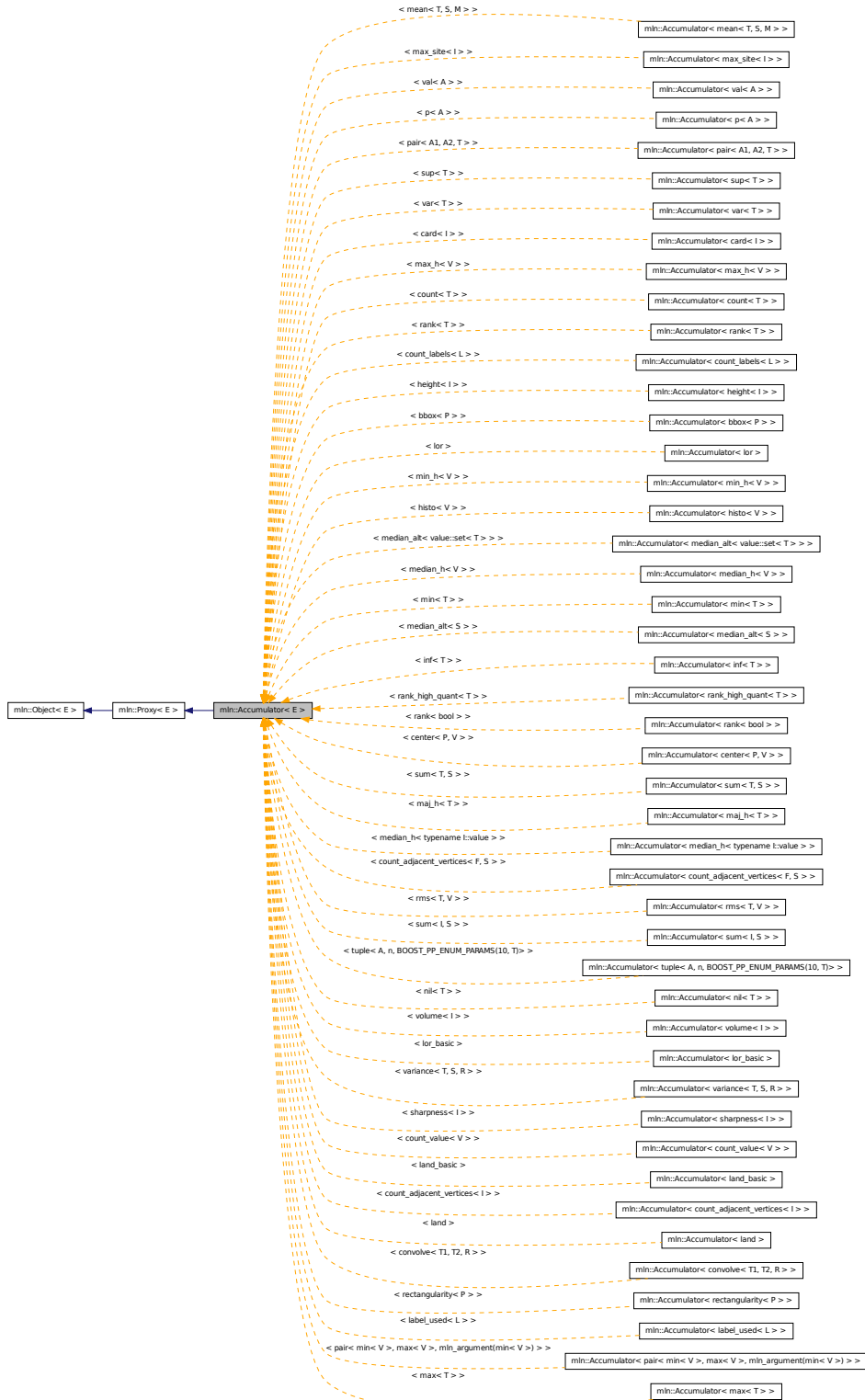
Get the value of the accumulator.

## 10.76 mln::Accumulator< E > Struct Template Reference

Base class for implementation of accumulators.

```
#include <accumulator.hh>
```

Inheritance diagram for mln::Accumulator< E >:





## Public Member Functions

- `template<typename T >`  
`void take_as_init (const T &t)`  
*Take as initialization the value  $t$ .*
- `template<typename T >`  
`void take_n_times (unsigned n, const T &t)`  
*Take  $n$  times the value  $t$ .*

### 10.76.1 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.

### 10.76.2 Member Function Documentation

**10.76.2.1** `template<typename E > template<typename T > void mln::Accumulator< E >::take_as_init ( const T & t )`

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

**10.76.2.2** `template<typename E > template<typename T > void mln::Accumulator< E >::take_n_times ( unsigned n, const T & t )`

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

## 10.77 `mln::algebra::h_mat< d, T >` Struct Template Reference

N-Dimensional matrix with homogeneous coordinates.

```
#include <h_mat.hh>
```

Inherits `mln::algebra::mat< d+1, d+1, T >`.

### Public Types

- `enum`

*Dimension is the 'natural' one (3 for 3D), not the one of the vector (dim + 1).*

## Public Member Functions

- `mat< n, m, T > _1 () const`  
*Return the inverse of the matrix.*
- `h_mat ()`  
*Constructor without argument.*
- `h_mat (const mat< d+1, d+1, T > &x)`  
*Constructor with the underlying matrix.*
- `mat< m, n, T > t () const`  
*Return the transpose of the matrix.*

### 10.77.1 Detailed Description

`template<unsigned d, typename T> struct mln::algebra::h_mat< d, T >`

N-Dimensional matrix with homogeneous coordinates.

### 10.77.2 Member Enumeration Documentation

#### 10.77.2.1 `template<unsigned d, typename T> anonymous enum`

Dimension is the 'natural' one (3 for 3D), not the one of the vector (dim + 1).

### 10.77.3 Constructor & Destructor Documentation

#### 10.77.3.1 `template<unsigned d, typename T > mln::algebra::h_mat< d, T >::h_mat ( )` `[inline]`

Constructor without argument.

#### 10.77.3.2 `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.

### 10.77.4 Member Function Documentation

#### 10.77.4.1 `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.

**10.77.4.2** `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.

## 10.78 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 ()`  
*Constructor without argument.*
- `h_vec (const vec< d+1, C > &other)`  
*Constructor with the underlying vector.*
- `template<typename U > operator mat< n, 1, U > () const`  
*Conversion to a matrix.*
- `mat< 1, n, T > t () const`  
*Transposition.*
- `vec< d, C > to_vec () const`  
*Back to the natural (non-homogeneous) space.*

### Static Public Attributes

- `static const vec< n, T > origin = all_to(0)`  
*Origin value.*
- `static const vec< n, T > zero = all_to(0)`  
*Zero value.*

### 10.78.1 Detailed Description

```
template<unsigned d, typename C> struct mln::algebra::h_vec< d, C >
```

N-Dimensional vector with homogeneous coordinates.

### 10.78.2 Member Enumeration Documentation

**10.78.2.1** `template<unsigned d, typename C> anonymous enum`

Dimension is the 'natural' one (3 for 3D), not the one of the vector (dim + 1).

### 10.78.3 Constructor & Destructor Documentation

**10.78.3.1** `template<unsigned d, typename C > mln::algebra::h_vec< d, C >::h_vec ( )`  
`[inline]`

Constructor without argument.

References `mln::literal::one`.

**10.78.3.2** `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.

### 10.78.4 Member Function Documentation

**10.78.4.1** `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.

**10.78.4.2** `template<unsigned n, typename T > mat< 1, n, T > mln::algebra::vec< n, T >::t ( )`  
`const [inline, inherited]`

Transposition.

**10.78.4.3** `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.

### 10.78.5 Member Data Documentation

**10.78.5.1** `template<unsigned n, typename T> const vec< n, T > mln::algebra::vec< n, T`  
`>::origin = all_to(0) [static, inherited]`

Origin value.

**10.78.5.2** `template<unsigned n, typename T> const vec< n, T > mln::algebra::vec< n, T >::zero = all_to(0) [static, inherited]`

Zero value.

## 10.79 mln::bkd\_pixter1d< I > Class Template Reference

Backward pixel iterator on a 1-D image with border.

```
#include <pixter1d.hh>
```

Inherits `backward_pixel_iterator_base_< I, bkd_pixter1d< I > >`.

### Public Types

- typedef I `image`  
*Image type.*

### Public Member Functions

- `bkd_pixter1d (I &image)`  
*Constructor.*
- void `next ()`  
*Go to the next element.*

#### 10.79.1 Detailed Description

```
template<typename I> class mln::bkd_pixter1d< I >
```

Backward pixel iterator on a 1-D image with border.

#### 10.79.2 Member Typedef Documentation

**10.79.2.1** `template<typename I > typedef I mln::bkd_pixter1d< I >::image`

`Image` type.

#### 10.79.3 Constructor & Destructor Documentation

**10.79.3.1** `template<typename I > mln::bkd_pixter1d< I >::bkd_pixter1d ( I & image ) [inline]`

Constructor.

#### Parameters

[in] *image* The image this pixel iterator is bound to.

## 10.79.4 Member Function Documentation

### 10.79.4.1 void mln::Iterator< bkd\_pixter1d< I > >::next ( ) [inherited]

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

#### Precondition

The iterator is valid.

## 10.80 mln::bkd\_pixter2d< I > Class Template Reference

Backward pixel iterator on a 2-D image with border.

```
#include <pixter2d.hh>
```

Inherits backward\_pixel\_iterator\_base\_< I, bkd\_pixter2d< I > >.

### Public Types

- typedef I [image](#)  
*Image type.*

### Public Member Functions

- [bkd\\_pixter2d](#) (I &[image](#))  
*Constructor.*
- void [next](#) ()  
*Go to the next element.*

### 10.80.1 Detailed Description

```
template<typename I> class mln::bkd_pixter2d< I >
```

Backward pixel iterator on a 2-D image with border.

### 10.80.2 Member Typedef Documentation

#### 10.80.2.1 template<typename I > typedef I mln::bkd\_pixter2d< I >::image

[Image](#) type.

### 10.80.3 Constructor & Destructor Documentation

**10.80.3.1** `template<typename I > mln::bkd_pixter2d< I >::bkd_pixter2d ( I & image )`  
`[inline]`

Constructor.

#### Parameters

`[in]` *image* The image this pixel iterator is bound to.

### 10.80.4 Member Function Documentation

**10.80.4.1** `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.

## 10.81 mln::bkd\_pixter3d< I > Class Template Reference

Backward pixel iterator on a 3-D image with border.

```
#include <pixter3d.hh>
```

Inherits `backward_pixel_iterator_base_< I, bkd_pixter3d< I > >`.

### Public Types

- typedef I [image](#)  
*Image* type.

### Public Member Functions

- [bkd\\_pixter3d](#) (I &[image](#))  
*Constructor.*
- void [next](#) ()  
*Go to the next element.*

### 10.81.1 Detailed Description

```
template<typename I> class mln::bkd_pixter3d< I >
```

Backward pixel iterator on a 3-D image with border.

### 10.81.2 Member Typedef Documentation

10.81.2.1 `template<typename I > typedef I mln::bkd_pixter3d< I >::image`

[Image](#) type.

### 10.81.3 Constructor & Destructor Documentation

10.81.3.1 `template<typename I > mln::bkd_pixter3d< I >::bkd_pixter3d ( I & image )`  
`[inline]`

Constructor.

#### Parameters

[in] *image* The image this pixel iterator is bound to.

### 10.81.4 Member Function Documentation

10.81.4.1 `void mln::Iterator< bkd_pixter3d< I > >::next ( )` `[inherited]`

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

#### Precondition

The iterator is valid.

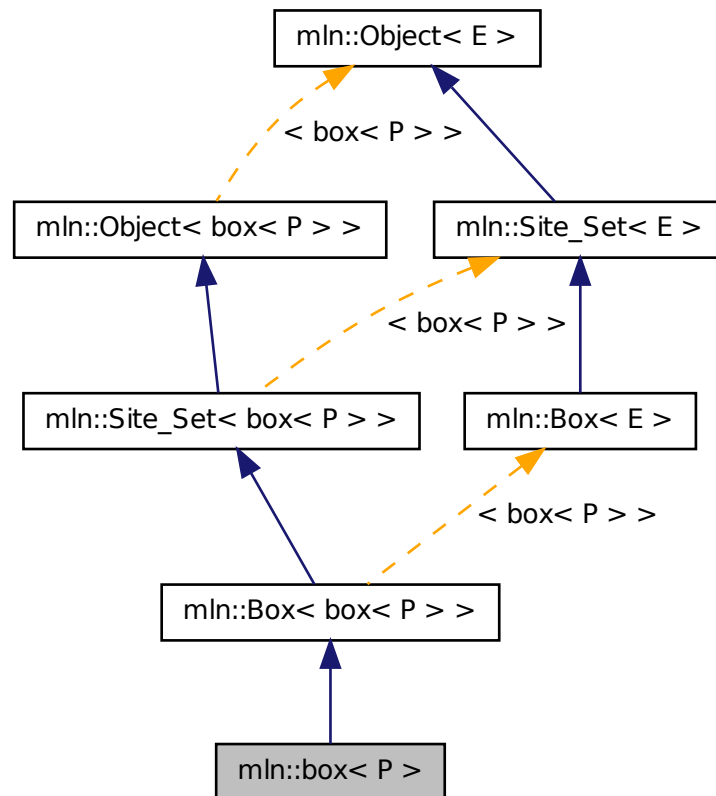
## 10.82 mln::box< P > Struct Template Reference

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](#) ()  
*Constructor without argument.*
- [box](#) (const [site](#) &pmin, const [site](#) &pmax)  
*Constructor of a box going from pmin to pmax.*
- void [crop\\_wrt](#) (const [box](#)< P > &b)  
*Crop this bbox in order to fit in the reference box b.*
- void [enlarge](#) (unsigned b)  
*Enlarge the box with a border b.*
- void [enlarge](#) (unsigned dim, unsigned b)  
*Enlarge the box with a border b for dimension dim.*
- bool [has](#) (const P &p) const  
*Test if p belongs to the box.*
- bool [is\\_empty](#) () const  
*Test if this box is empty.*
- bool [is\\_valid](#) () const  
*Test that the box owns valid data, i.e., is initialized and with pmin being 'less-than' pmax.*
- unsigned [len](#) (unsigned i) const  
*Give the length of the i-th side of the box.*
- std::size\_t [memory\\_size](#) () const  
*Return the size of this site set in memory.*
- void [merge](#) (const [box](#)< P > &b)  
*Merge inplace with another box.*
- unsigned [nsites](#) () const  
*Give the number of sites of this box.*

- P [pcenter](#) () const  
*Return the approximated central site of this box.*
- P & [pmax](#) ()  
*Reference to the maximum point.*
- P [pmax](#) () const  
*Maximum point.*
- P [pmin](#) () const  
*Minimum point.*
- P & [pmin](#) ()  
*Reference to the minimum point.*
- [box](#)< P > [to\\_larger](#) (unsigned b) const  
*Give a larger box.*
- [box](#) (typename P::coord ninds)

## Related Functions

(Note that these are not member functions.)

- `template<typename P > std::ostream & operator<< (std::ostream &ostr, const box< P > &b)`  
*Print a generic box *b* into the output stream *ostr*.*

### 10.82.1 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.

### 10.82.2 Member Typedef Documentation

**10.82.2.1** `template<typename P> typedef box\_bkd\_piter<P> mln::box< P >::bkd_piter`

Backward [Site\\_Iterator](#) associated type.

**10.82.2.2** `template<typename P> typedef P mln::box< P >::element`

Element associated type.

**10.82.2.3** `template<typename P> typedef box_fwd_piter_<P> mln::box< P >::fwd_piter`

Forward [Site\\_Iterator](#) associated type.

**10.82.2.4** `template<typename P> typedef fwd_piter mln::box< P >::piter`

[Site\\_Iterator](#) associated type.

**10.82.2.5** `template<typename P> typedef P mln::box< P >::psite`

Psite associated type.

**10.82.2.6** `template<typename P> typedef P mln::box< P >::site`

[Site](#) associated type.

**10.82.3 Member Enumeration Documentation****10.82.3.1** `template<typename P> anonymous enum`

Dimension.

**10.82.4 Constructor & Destructor Documentation****10.82.4.1** `template<typename P> mln::box< P >::box ( ) [inline]`

Constructor without argument.

**10.82.4.2** `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().

**10.82.4.3** `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.

**10.82.5 Member Function Documentation****10.82.5.1** `const box< P > & mln::Box< box< P > >::bbox ( ) const [inherited]`

Give the bounding box of this site set.

Return the bounding box of this site set, so that is itself. This method is declared by the [mln::Site\\_Set](#) concept.

### Warning

This method is final for all box classes.

#### 10.82.5.2 `template<typename P> void mln::box< P >::crop_wrt ( const box< P > & b ) [inline]`

Crop this bbox in order to fit in the reference box *b*.

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

Referenced by `mln::make_debug_graph_image()`.

#### 10.82.5.3 `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()`.

#### 10.82.5.4 `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()`.

#### 10.82.5.5 `template<typename P> bool mln::box< P >::has ( const P & p ) const [inline]`

Test if *p* belongs to the box.

### Parameters

[in] *p* A point site.

References `mln::box< P >::is_valid()`.

Referenced by `mln::morpho::line_gradient()`.

#### 10.82.5.6 `bool mln::Box< box< P > >::is_empty ( ) const [inherited]`

Test if this box is empty.

#### 10.82.5.7 `template<typename P> bool mln::box< P >::is_valid ( ) const [inline]`

Test that the box owns valid data, i.e., is initialized and with *pmin* being 'less-than' *pmax*.

References `mln::util::ord_weak()`.

Referenced by `mln::box< P >::box()`, `mln::transform::distance_and_closest_point_geodesic()`, `mln::box< P >::enlarge()`, `mln::box< P >::has()`, `mln::box< P >::merge()`, `mln::box< P >::pcenter()`, `mln::box< P >::pmax()`, `mln::box< P >::pmin()`, and `mln::box< P >::to_larger()`.

#### 10.82.5.8 `unsigned mln::Box< box< P > >::len ( unsigned i ) const [inherited]`

Give the length of the `i`-th side of the box.

##### Precondition

`i < site::dim`

##### Warning

This method is final for all box classes.

#### 10.82.5.9 `template<typename P > std::size_t mln::box< P >::memory_size ( ) const [inline]`

Return the size of this site set in memory.

#### 10.82.5.10 `template<typename P > void mln::box< P >::merge ( const box< P > & b ) [inline]`

Merge inplace with another box.

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

#### 10.82.5.11 `unsigned mln::Box< box< P > >::nsites ( ) const [inherited]`

Give the number of sites of this box.

Return the number of sites of this box. This method is declared by the [mln::Site\\_Set](#) concept.

##### Warning

This method is final for all box classes.

#### 10.82.5.12 `template<typename P > P mln::box< P >::pcenter ( ) const [inline]`

Return the approximated central site of this box.

References `mln::box< P >::is_valid()`.

#### 10.82.5.13 `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()`, `mln::larger_than()`, `mln::io::fld::load()`, and `mln::box< P >::merge()`.

**10.82.5.14** `template<typename P> P & mln::box< P >::pmax ( ) [inline]`

Reference to the maximum point.

**10.82.5.15** `template<typename P> P & mln::box< P >::pmin ( ) [inline]`

Reference to the minimum point.

**10.82.5.16** `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(), mln::larger\_than(), mln::io::fld::load(), and mln::box< P >::merge().

**10.82.5.17** `template<typename P> box< P > mln::box< P >::to_larger ( unsigned b ) const [inline]`

Give a larger box.

References mln::box< P >::is\_valid().

**10.82.6 Friends And Related Function Documentation****10.82.6.1** `template<typename P> std::ostream & operator<< ( std::ostream & ostr, const box< P > & b ) [related]`

Print a generic box *b* into the output stream *ostr*.

**Parameters**

[in, out] *ostr* An output stream.

[in] *b* A generic box.

**Returns**

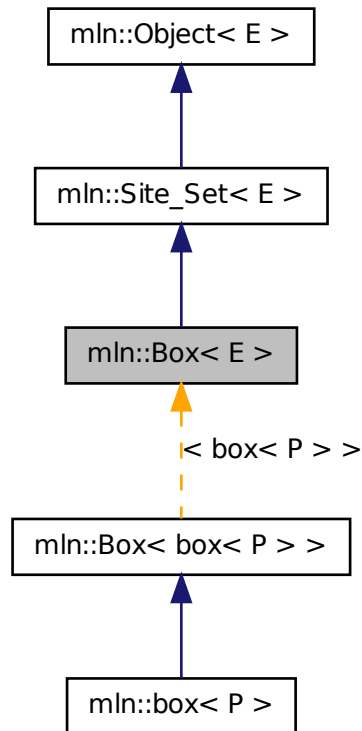
The modified output stream *ostr*.

**10.83 mln::Box< E > Struct Template Reference**

Base class for implementation classes of boxes.

```
#include <box.hh>
```

Inheritance diagram for `mln::Box< E >`:



## Public Member Functions

- `const E & bbox () const`  
*Give the bounding box of this site set.*
- `bool is_empty () const`  
*Test if this box is empty.*
- `unsigned len (unsigned i) const`  
*Give the length of the *i*-th side of the box.*
- `unsigned nsites () const`  
*Give the number of sites of this box.*

## Related Functions

(Note that these are not member functions.)



- `template<typename SI, typename Sr >`  
`p_set< typename SI::site > diff (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`  
*Set theoretic difference of lhs and rhs.*
- `template<typename SI, typename Sr >`  
`p_set< typename SI::site > inter (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`  
*Intersection between a couple of point sets.*
- `template<typename SI, typename Sr >`  
`bool operator< (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`  
*Strict inclusion test between site sets lhs and rhs.*
- `template<typename BI, typename Br >`  
`bool operator< (const Box< BI > &lhs, const Box< Br > &rhs)`  
*Strict inclusion test between boxes lhs and rhs.*
- `template<typename S >`  
`std::ostream & operator<< (std::ostream &ostr, const Site_Set< S > &set)`  
*Print a site set set into the output stream ostr.*
- `template<typename SI, typename Sr >`  
`bool operator<= (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`  
*Inclusion test between site sets lhs and rhs.*
- `template<typename BI, typename Br >`  
`bool operator<= (const Box< BI > &lhs, const Box< Br > &rhs)`  
*Inclusion test between boxes lhs and rhs.*
- `template<typename SI, typename Sr >`  
`bool operator== (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`  
*Equality test between site sets lhs and rhs.*
- `template<typename SI, typename Sr >`  
`p_set< typename SI::site > sym_diff (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`  
*Set theoretic symmetrical difference of lhs and rhs.*
- `template<typename SI, typename Sr >`  
`p_set< typename SI::site > uni (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`  
*Union of a couple of point sets.*
- `template<typename S >`  
`p_set< typename S::site > unique (const Site_Set< S > &s)`  
*Give the unique set of s.*

### 10.83.1 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.

**10.83.2 Member Function Documentation****10.83.2.1 `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.

**10.83.2.2 `template<typename E> bool mln::Box< E >::is_empty ( ) const [inline]`**

Test if this box is empty.

**10.83.2.3 `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.

**10.83.2.4 `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()`.

**10.83.3 Friends And Related Function Documentation****10.83.3.1 `template<typename SI, typename Sr> p_set< typename SI::site > diff ( const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs ) [related, inherited]`**

Set theoretic difference of `lhs` and `rhs`.

**10.83.3.2** `template<typename SI, typename Sr > p_set< typename SI::site > inter ( const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs )` [related, inherited]

Intersection between a couple of point sets.

**10.83.3.3** `template<typename SI, typename Sr > bool operator< ( const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs )` [related, inherited]

Strict inclusion test between site sets `lhs` and `rhs`.

#### Parameters

[in] *lhs* A site set (strictly included?).

[in] *rhs* Another site set (includer?).

**10.83.3.4** `template<typename BI, typename Br > bool operator< ( const Box< BI > & lhs, const Box< Br > & rhs )` [related]

Strict inclusion test between boxes `lhs` and `rhs`.

#### Parameters

[in] *lhs* A box (strictly included?).

[in] *rhs* Another box (includer?).

**10.83.3.5** `template<typename S > std::ostream & operator<< ( std::ostream & ostr, const Site_Set< S > & set )` [related, inherited]

Print a site set `set` into the output stream `ostr`.

#### Parameters

[in, out] *ostr* An output stream.

[in] *set* A site set.

#### Returns

The modified output stream `ostr`.

**10.83.3.6** `template<typename SI, typename Sr > bool operator<= ( const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs )` [related, inherited]

Inclusion test between site sets `lhs` and `rhs`.

#### Parameters

[in] *lhs* A site set (included?).

[in] *rhs* Another site set (includer?).

**10.83.3.7** `template<typename Bl, typename Br > bool operator<= ( const Box< Bl > & lhs, const Box< Br > & rhs ) [related]`

Inclusion test between boxes `lhs` and `rhs`.

#### Parameters

- [in] *lhs* A box (included?).
- [in] *rhs* Another box (includor?).

**10.83.3.8** `template<typename Sl, typename Sr > bool operator== ( const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs ) [related, inherited]`

Equality test between site sets `lhs` and `rhs`.

#### Parameters

- [in] *lhs* A site set.
- [in] *rhs* Another site set.

**10.83.3.9** `template<typename Sl, typename Sr > p_set< typename Sl::site > sym_diff ( const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs ) [related, inherited]`

Set theoretic symmetrical difference of `lhs` and `rhs`.

**10.83.3.10** `template<typename Sl, typename Sr > p_set< typename Sl::site > uni ( const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs ) [related, inherited]`

Union of a couple of point sets.

**10.83.3.11** `template<typename S > p_set< typename S::site > unique ( const Site_Set< S > & s ) [related, inherited]`

Give the unique set of `s`.

## 10.84 mln::box\_runend\_piter< P > Class Template Reference

A generic backward iterator on points by lines.

```
#include <box_runend_piter.hh>
```

Inherits `site_set_iterator_base< box< P >, box_runend_piter< P > >`.

### Public Member Functions

- `box_runend_piter` (const `box< P >` &b)

*Constructor:*

- void `next()`  
Go to the next element.
- unsigned `run_length()` const  
Give the length of the run.

### 10.84.1 Detailed Description

`template<typename P> class mln::box_runend_piter< P >`

A generic backward iterator on points by lines. The parameter `P` is the type of points.

### 10.84.2 Constructor & Destructor Documentation

**10.84.2.1** `template<typename P > mln::box_runend_piter< P >::box_runend_piter ( const box< P > & b ) [inline]`

Constructor.

#### Parameters

[in] `b` A box.

### 10.84.3 Member Function Documentation

**10.84.3.1** `void mln::Site_Iterator< box_runend_piter< P > >::next ( ) [inherited]`

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

#### Precondition

The iterator is valid.

**10.84.3.2** `template<typename P > unsigned mln::box_runend_piter< P >::run_length ( ) const [inline]`

Give the length of the run.

## 10.85 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.*

### 10.85.1 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.

### 10.85.2 Constructor & Destructor Documentation

```
10.85.2.1 template<typename P > mln::box_runstart_piter< P >::box_runstart_piter ( const
box< P > & b ) [inline]
```

Constructor.

#### Parameters

[in] *b* A box.

### 10.85.3 Member Function Documentation

```
10.85.3.1 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.

```
10.85.3.2 template<typename P > unsigned mln::box_runstart_piter< P >::run_length ( ) const
[inline]
```

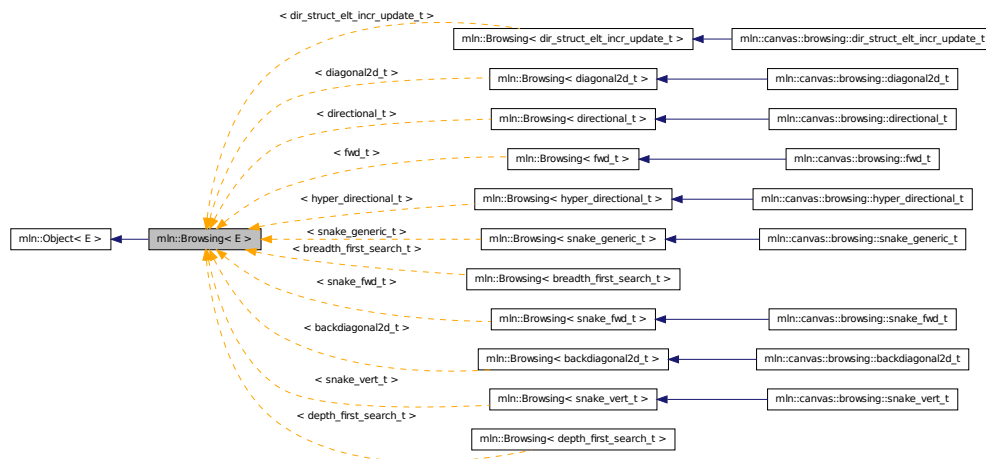
Give the lenght of the run.

## 10.86 mln::Browsing< E > Struct Template Reference

Base class for implementation classes that are browsings.

```
#include <browsing.hh>
```

Inheritance diagram for mln::Browsing< E >:



### 10.86.1 Detailed Description

```
template<typename E> struct mln::Browsing< E >
```

Base class for implementation classes that are browsings.

#### See also

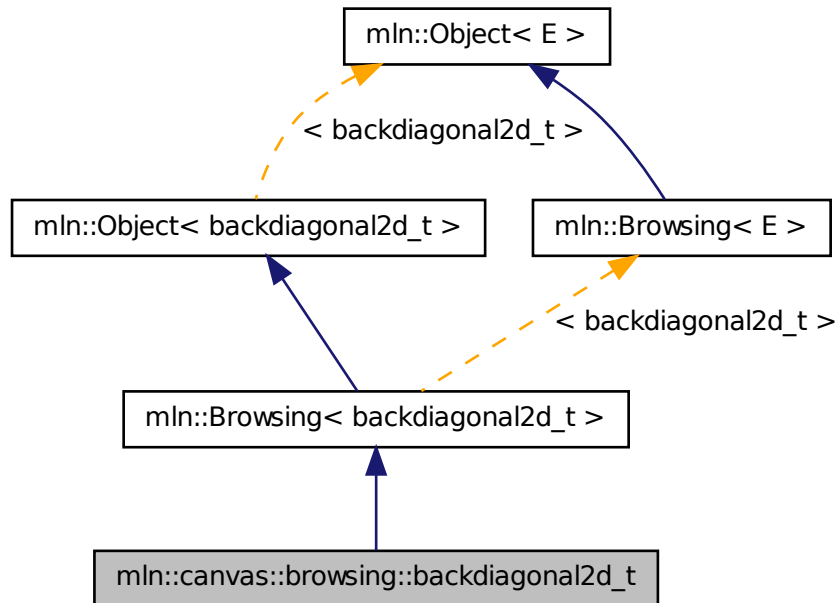
`mln::doc::Browsing` for a complete documentation of this class contents.

## 10.87 mln::canvas::browsing::backdiagonal2d\_t Struct Reference

[Browsing](#) in a certain direction.

```
#include <backdiagonal2d.hh>
```

Inheritance diagram for `mln::canvas::browsing::backdiagonal2d_t`:



### 10.87.1 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 functor).
- `final()`: Will be called at the end.

F shall features :

```

{
--- as types:
I;
--- as attributes:
dim;
dir; // and test dir < dim
input;

```



```
p;  
--- as methods:  
void init();  
void next();  
void final();  
}
```

Example :

```
-----> | 4 7 9 | 2 5 8 | 1 3 6
```

## 10.88 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 >.
```

### 10.88.1 Detailed Description

Breadth-first search algorithm for graph, on vertices.

## 10.89 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 >.
```

### 10.89.1 Detailed Description

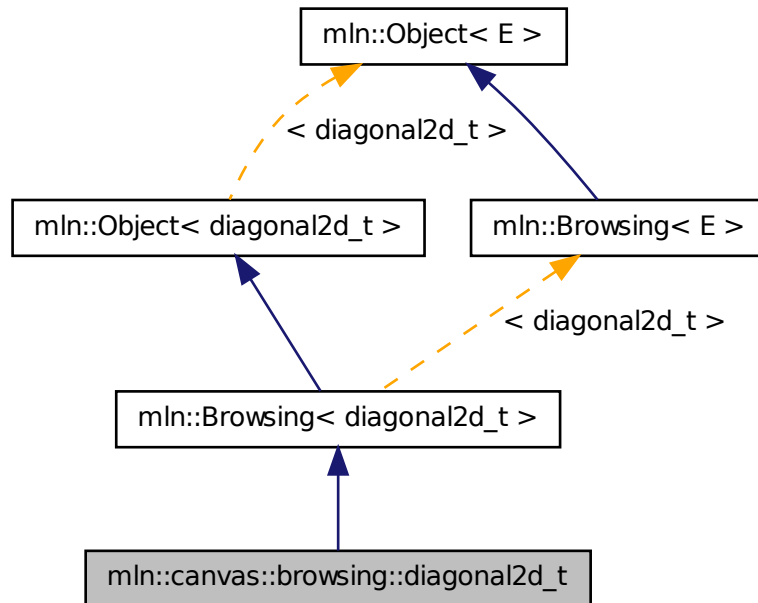
Breadth-first search algorithm for graph, on vertices.

## 10.90 mln::canvas::browsing::diagonal2d\_t Struct Reference

[Browsing](#) in a certain direction.

```
#include <diagonal2d.hh>
```

Inheritance diagram for mln::canvas::browsing::diagonal2d\_t:



### 10.90.1 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 functor).
- `final()` : Will be called at the end.

F shall features :

```

{
--- as types:
I;
--- as attributes:
dim;
dir; // and test dir < dim
input;

```

p;

--- as methods:

```
void init();
```

```
void next();
```

```
void final();
```

```
}
```

Example :

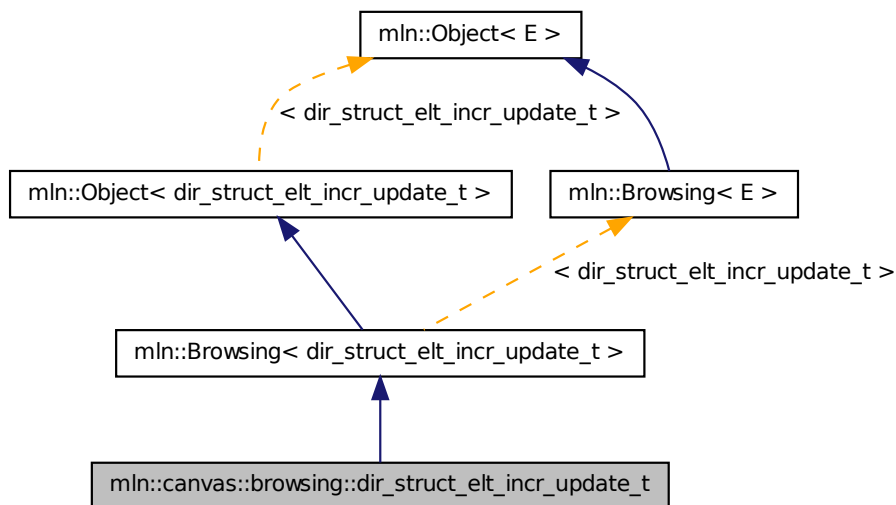
```
| 1 3 6 | 2 5 8 | 4 7 9 L----->
```

## 10.91 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:



### 10.91.1 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 :

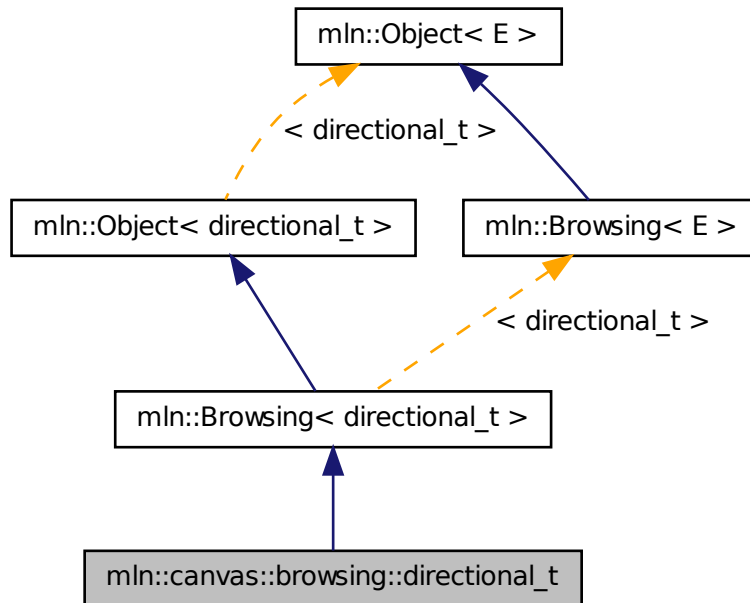
```
{  
--- as types:  
I;  
--- as attributes:  
dim;  
dir; // and test dir < dim  
input;  
p;  
length;  
--- as methods:  
void init();  
void init_line();  
void add_point(q)  
void remove_point(q)  
void next();  
void final();  
}
```

## 10.92 `mln::canvas::browsing::directional_t` Struct Reference

[Browsing](#) in a certain direction.

```
#include <directional.hh>
```

Inheritance diagram for mln::canvas::browsing::directional\_t:



### 10.92.1 Detailed Description

**Browsing** in a certain direction. This canvas browse all the point of an image 'input' of type 'I' and of dimension 'dim' in the direction 'dir'.

The functor should provide (In addition to 'input', 'I', 'dim' and 'dir') three methods :

- `init()` : Will be called at the beginning.
- `next()` : Will be called at each point 'p' (also provided by the functor).
- `final()` : Will be called at the end.

F shall features :

```

{
--- as types:
I;
--- as attributes:
dim;
dir; // and test dir < dim
input;

```

```

P;
--- as methods:
void init();
void next();
void final();
}

```

Example :

```

1 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

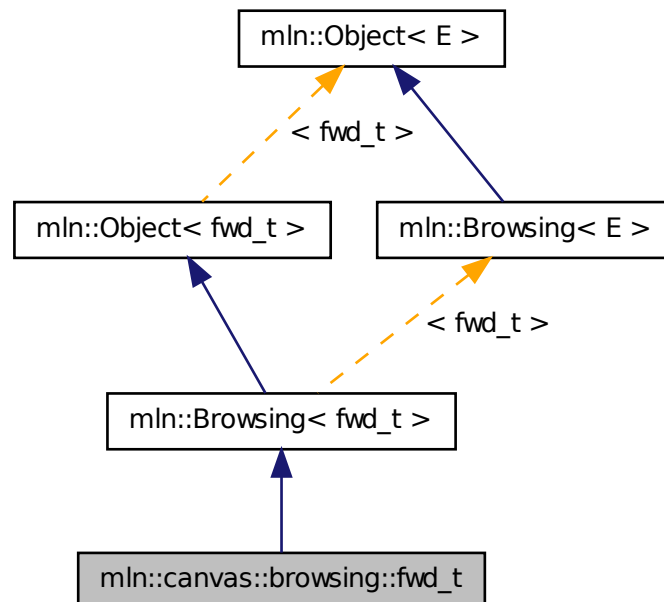
```

### 10.93 mln::canvas::browsing::fwd\_t Struct Reference

Canvas for forward browsing.

```
#include <fwd.hh>
```

Inheritance diagram for mln::canvas::browsing::fwd\_t:



### 10.93.1 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 functor should provide (In addition of 'I' and 'input') three methods :

- `init()` : Will be called at the beginning.
- `next()` : Will be called at each point 'p' (also provided by the functor).
- `final()`: Will be called at the end.

F shall feature:

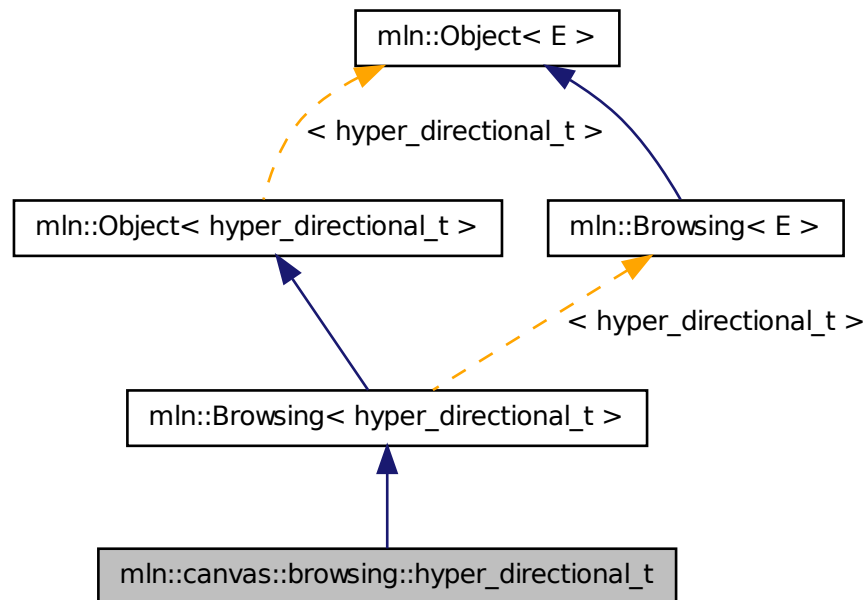
```
{  
--- as typedef:  
I;  
--as attributes:  
input;  
p;  
--- as method:  
void init();  
void next();  
void final();  
}
```

## 10.94 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:



### 10.94.1 Detailed Description

**Browsing** in a certain direction. This canvas browse all the point of an image 'input' of type 'I' and of dimension 'dim' in the direction 'dir'.

The functor should provide (In addition to 'input', 'I', 'dim' and 'dir') three methods :

- `init()` : Will be called at the beginning.
- `next()` : Will be called at each point 'p' (also provided by the functor).
- `final()` : Will be called at the end.

F shall features :

```

{
--- as types:
I;
--- as attributes:
dim;
dir; // and test dir < dim
input;

```



```

p;
--- as methods:
void init();
void next();
void final();
}

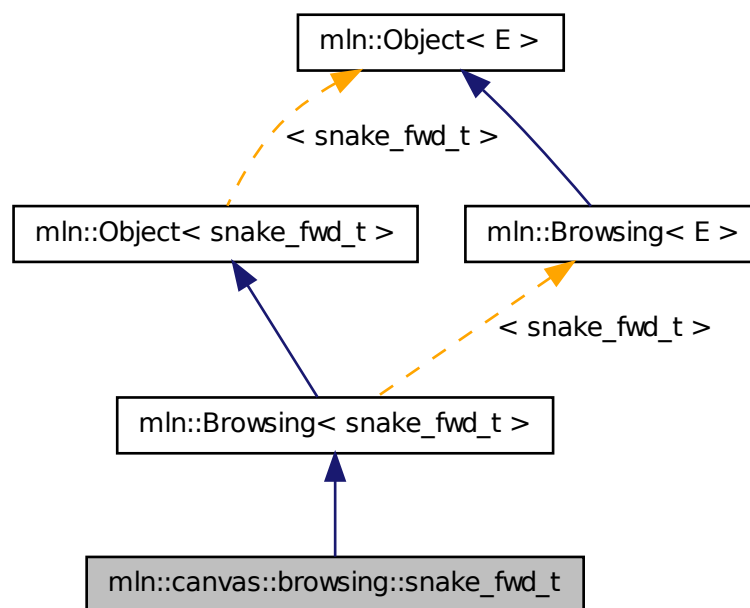
```

## 10.95 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:



### 10.95.1 Detailed Description

[Browsing](#) in a snake-way, forward. This canvas browse all the point of an image 'input' like this :

```
-----> <-----' '----->
```

The functor should provide (In addition to 'input') four methods :

- `init()` : Will be called at the beginning.

- `down()` : Will be called after each moving down. (will also be called once at the first point).
  
- `fwd()` : Will be called after each moving right.
  
- `bwd()` : Will be called after each moving left.

This methods should access to the current working point 'p' also provided by the functor.

Warning: This canvas works only on 2D.

F shall feature:

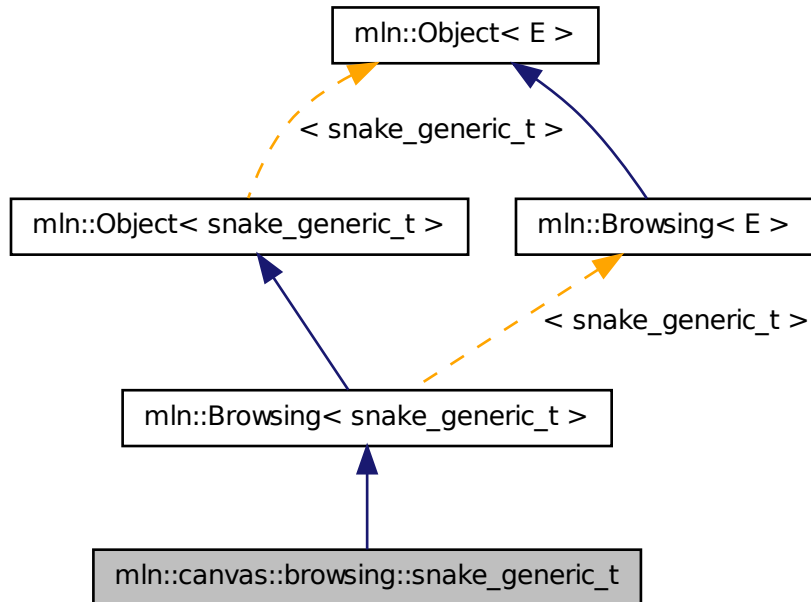
```
{  
--- as attributes:  
input;  
p;  
--- as methods:  
void init();  
void down();  
void fwd();  
void bkd();  
}
```

## 10.96 `mln::canvas::browsing::snake_generic_t` Struct Reference

Multidimensional [Browsing](#) in a given-way.

```
#include <snake_generic.hh>
```

Inheritance diagram for mln::canvas::browsing::snake\_generic\_t:



### 10.96.1 Detailed Description

Multidimensional [Browsing](#) in a given-way. F shall feature:

```

{
--- as attributes:
input;
p;
--- as methods:
void init();
void *() moves[];
dpsite dps[];
}

```

init is called before browsing

The snake follow dimension using the delta point site of dps. dps[0] = delta psite following the global dimension (forward) dps[1] = delta psite following the 2nd dimension to follow (forward). dps[2] = delta psite following the 2nd dimension to follow (backward). dps[3] = delta psite following the 3rd dimension to follow (forward). dps[3] = delta psite following the 3rd dimension to follow (backward).

moves contains pointer to f's members. These members will be call in each time the snake progress in the

correct dimension :

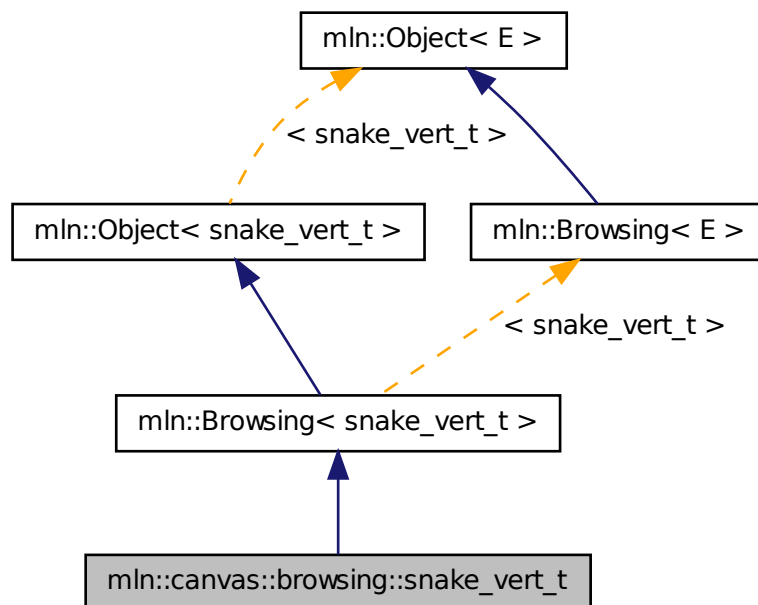
moves[i] is called at each move following the delta psite dps[i]

## 10.97 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:



### 10.97.1 Detailed Description

[Browsing](#) in a snake-way, forward. This canvas browse all the point of an image 'input' like this :

```
| \ | | | | \ | \
```

The functor should provide (In addition to 'input') four methods :

- `init()` : Will be called at the beginning.
- `down()` : Will be called after each moving down.
- `up()` : Will be called after each moving up.
- `fwd()` : Will be called after each moving right. (will also be called once at the first point).

This methods should acces to the current working point 'p' also provided by the functor.

Warning: This canvas works only on 2D.

F shall feature:

```
{
--- as attributes:
input;
p;
--- as methods:
void init();
void down();
void up();
void fwd();
}
```

## 10.98 mln::canvas::chamfer< F > Struct Template Reference

Compute chamfer distance.

```
#include <chamfer.hh>
```

### 10.98.1 Detailed Description

```
template<typename F> struct mln::canvas::chamfer< F >
```

Compute chamfer distance.

## 10.99 mln::category< R(\*) (A) > Struct Template Reference

Category declaration for a unary C function.

```
#include <c.hh>
```

### 10.99.1 Detailed Description

```
template<typename R, typename A> struct mln::category< R(*) (A) >
```

Category declaration for a unary C function.

## 10.100 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 `V` & [lvalue](#)  
*Return type of read-write access.*
- typedef `const V` & [rvalue](#)  
*Return type of read-only access.*
- typedef `complex_image< D, tag::psite_< G >, tag::value_< V > >` [skeleton](#)  
*Skeleton.*
- typedef `V` [value](#)  
*Value associated type.*

## Public Member Functions

- `rvalue operator()` (`const complex_psite< D, G > &p`) `const`  
*Read-only access of face value at point site `p`.*
- `lvalue operator()` (`const complex_psite< D, G > &p`)  
*Read-write access of face value at point site `p`.*
- `complex_image ()`  
*Constructors.*
- `const p_complex< D, G > & domain () const`  
*Accessors.*
- `const metal::vec< D+1, std::vector< mlc_unbool(V) > > & values () const`  
*Return the array of values associated to the faces.*

## Static Public Attributes

- `static const unsigned dim = D`  
*The dimension of the complex.*

### 10.100.1 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.

### 10.100.2 Member Typedef Documentation

**10.100.2.1** `template<unsigned D, typename G, typename V> typedef G mln::complex_image< D, G, V >::geom`

The geometry type of the complex.

**10.100.2.2** `template<unsigned D, typename G, typename V> typedef V& mln::complex_image< D, G, V >::lvalue`

Return type of read-write access.

**10.100.2.3** `template<unsigned D, typename G, typename V> typedef const V& mln::complex_image< D, G, V >::rvalue`

Return type of read-only access.

**10.100.2.4** `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.

**10.100.2.5** `template<unsigned D, typename G, typename V> typedef V mln::complex_image< D, G, V >::value`

[Value](#) associated type.

### 10.100.3 Constructor & Destructor Documentation

**10.100.3.1** `template<unsigned D, typename G, typename V> mln::complex_image< D, G, V >::complex_image ( ) [inline]`

Constructors.

### 10.100.4 Member Function Documentation

**10.100.4.1** `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.

**10.100.4.2** `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().

**10.100.4.3** `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().

**10.100.4.4** `template<unsigned D, typename G , typename V > const metal::vec< D+1, std::vector< mlc_unbool(V) > > & mln::complex_image< D, G, V >::values ( ) const [inline]`

Return the array of values associated to the faces.

### 10.100.5 Member Data Documentation

**10.100.5.1** `template<unsigned D, typename G, typename V> const unsigned mln::complex_image< D, G, V >::dim = D [static]`

The dimension of the complex.

## 10.101 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.

### 10.101.1 Detailed Description

`template<typename I, typename G, typename N> class mln::complex_neighborhood_bkd_piter< I, G, N >`

Backward iterator on complex neighborhood.

### 10.101.2 Member Typedef Documentation

**10.101.2.1** `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.

**10.101.2.2** `template<typename I, typename G, typename N> typedef N::psite mln::complex_neighborhood_bkd_piter< I, G, N >::psite`

The [Pseudo\\_Site](#) type.

### 10.101.3 Constructor & Destructor Documentation

**10.101.3.1** `template<typename I , typename G , typename N > mln::complex_neighborhood_bkd_piter< I, G, N >::complex_neighborhood_bkd_piter ( ) [inline]`

Construction.

### 10.101.4 Member Function Documentation

**10.101.4.1** `template<typename I, typename G, typename N> const N::complex_bkd_iter & mln::complex_neighborhood_bkd_piter< I, G, N >::iter ( ) const [inline]`

Accessors.

**10.101.4.2** `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.

## 10.102 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.*

### 10.102.1 Detailed Description

`template<typename I, typename G, typename N> class mln::complex_neighborhood_fwd_piter< I, G, N >`

Forward iterator on complex neighborhood.

### 10.102.2 Member Typedef Documentation

**10.102.2.1** `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.

**10.102.2.2** `template<typename I, typename G, typename N> typedef N::psite mln::complex_neighborhood_fwd_piter< I, G, N >::psite`

The [Pseudo\\_Site](#) type.

### 10.102.3 Constructor & Destructor Documentation

**10.102.3.1** `template<typename I , typename G , typename N > mln::complex_neighborhood_fwd_piter< I, G, N >::complex_neighborhood_fwd_piter ( ) [inline]`

Construction.

### 10.102.4 Member Function Documentation

**10.102.4.1** `template<typename I, typename G , typename N > const N::complex_fwd_iter & mln::complex_neighborhood_fwd_piter< I, G, N >::iter ( ) const [inline]`

Accessors.

**10.102.4.2** `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.

## 10.103 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

- [complex\\_psite](#) ()  
*Construction and assignment.*
- [complex\\_psite](#) (const [p\\_complex](#)< D, G > &pc, const [topo::face](#)< D > &face)
- bool [is\\_valid](#) () const  
*Psite manipulators.*
- void [invalidate](#) ()  
*Invalidate this psite.*
- const [target](#) & [site\\_set](#) () const  
*Site set manipulators.*
- void [change\\_target](#) (const [target](#) &new\_target)  
*Set the target site\_set.*
- const [topo::face](#)< D > & [face](#) () const  
*Face handle manipulators.*
- unsigned [n](#) () const  
*Return the dimension of the face of this psite.*
- unsigned [face\\_id](#) () const  
*Return the id of the face of this psite.*

### 10.103.1 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.

## 10.103.2 Constructor & Destructor Documentation

**10.103.2.1** `template<unsigned D, typename G > mln::complex_psite< D, G >::complex_psite ( ) [inline]`

Construction and assignment.

References mln::complex\_psite< D, G >::invalidate().

**10.103.2.2** `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().

## 10.103.3 Member Function Documentation

**10.103.3.1** `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().

**10.103.3.2** `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==(()).

**10.103.3.3** `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()().

**10.103.3.4** `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().

**10.103.3.5** `template<unsigned D, typename G > bool mln::complex_site< D, G >::is_valid ( )`  
`const [inline]`

Psite manipulators.

Is this psite valid?

Referenced by `mln::complex_site< D, G >::complex_site()`, and `mln::p_complex< D, G >::has()`.

**10.103.3.6** `template<unsigned D, typename G > unsigned mln::complex_site< 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()`.

**10.103.3.7** `template<unsigned D, typename G > const p_complex< D, G > &`  
`mln::complex_site< 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==()`.

## 10.104 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.*

### 10.104.1 Detailed Description

`template<typename I, typename G, typename W> class mln::complex_window_bkd_piter< I, G, W >`

Backward iterator on complex window.

### 10.104.2 Member Typedef Documentation

**10.104.2.1** `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.

**10.104.2.2** `template<typename I, typename G, typename W> typedef W::psite mln::complex_window_bkd_piter< I, G, W >::psite`

The [Pseudo\\_Site](#) type.

### 10.104.3 Constructor & Destructor Documentation

**10.104.3.1** `template<typename I , typename G , typename W > mln::complex_window_bkd_piter< I, G, W >::complex_window_bkd_piter ( )`  
`[inline]`

Construction.

### 10.104.4 Member Function Documentation

**10.104.4.1** `template<typename I , typename G , typename W > const W::complex_bkd_iter & mln::complex_window_bkd_piter< I, G, W >::iter ( ) const` `[inline]`

Accessors.

**10.104.4.2** `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-define this method. The actual "next" operation has to be defined through the *next\_* method.

### Precondition

The iterator is valid.

## 10.105 `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.*

#### 10.105.1 Detailed Description

```
template<typename I, typename G, typename W> class mln::complex_window_fwd_piter< I, G, W >
```

Forward iterator on complex window.



## 10.105.2 Member Typedef Documentation

**10.105.2.1** `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.

**10.105.2.2** `template<typename I, typename G, typename W> typedef W ::psite mln::complex_window_fwd_piter< I, G, W >::psite`

The [Pseudo\\_Site](#) type.

## 10.105.3 Constructor & Destructor Documentation

**10.105.3.1** `template<typename I , typename G , typename W > mln::complex_window_fwd_piter< I, G, W >::complex_window_fwd_piter ( ) [inline]`

Construction.

## 10.105.4 Member Function Documentation

**10.105.4.1** `template<typename I , typename G , typename W > const W::complex_fwd_iter & mln::complex_window_fwd_piter< I, G, W >::iter ( ) const [inline]`

Accessors.

**10.105.4.2** `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.

## 10.106 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 > >`.

## Public Types

- typedef [impl\\_::lvalue](#) **lvalue**  
*Return type of read-write access.*
- typedef [I::psite](#) **psite**  
*Type of the psite.*
- typedef [I::rvalue](#) **rvalue**  
*Return type of read-only access.*
- typedef [decorated\\_image< tag::image\\_< I >, tag::data\\_< D > >](#) **skeleton**  
*Skeleton.*

## Public Member Functions

- [decorated\\_image](#) ()  
*Ctors.*
- [D & decoration](#) ()  
*Give the decoration.*
- [const D & decoration](#) () const  
*Give the decoration.*
- [operator decorated\\_image< const I, D >](#) () const  
*Const promotion via conversion.*
- [rvalue operator\(\)](#) (const [psite](#) &p) const  
*Read-only access of pixel value at point site p.*
- [lvalue operator\(\)](#) (const [psite](#) &p)  
*Read-write access of pixel value at point site p.*
- [~decorated\\_image](#) ()  
*Dtor.*

### 10.106.1 Detailed Description

**template<typename I, typename D> struct mln::decorated\_image< I, D >**

[Image](#) that can have additional features.

## 10.106.2 Member Typedef Documentation

**10.106.2.1** `template<typename I, typename D> typedef impl_::lvalue mln::decorated_image< I, D >::lvalue`

Return type of read-write access.

**10.106.2.2** `template<typename I, typename D> typedef I ::psite mln::decorated_image< I, D >::psite`

Type of the psite.

**10.106.2.3** `template<typename I, typename D> typedef I ::rvalue mln::decorated_image< I, D >::rvalue`

Return type of read-only access.

**10.106.2.4** `template<typename I, typename D> typedef decorated_image< tag::image_<I>, tag::data_<D> > mln::decorated_image< I, D >::skeleton`

Skeleton.

## 10.106.3 Constructor & Destructor Documentation

**10.106.3.1** `template<typename I , typename D > mln::decorated_image< I, D >::decorated_image ( ) [inline]`

Ctors.

**10.106.3.2** `template<typename I , typename D > mln::decorated_image< I, D >::~~decorated_image ( ) [inline]`

Dtor.

## 10.106.4 Member Function Documentation

**10.106.4.1** `template<typename I , typename D > const D & mln::decorated_image< I, D >::decoration ( ) const [inline]`

Give the decoration.

**10.106.4.2** `template<typename I , typename D > D & mln::decorated_image< I, D >::decoration ( ) [inline]`

Give the decoration.

**10.106.4.3** `template<typename I , typename D > mln::decorated_image< I, D >::operator decorated_image< const I, D > ( ) const [inline]`

Const promotion via conversion.

**10.106.4.4** `template<typename I , typename D > decorated_image< I, D >::rvalue mln::decorated_image< I, D >::operator() ( const psite & p ) const [inline]`

Read-only access of pixel value at point site p.

**10.106.4.5** `template<typename I , typename D > decorated_image< I, D >::lvalue mln::decorated_image< I, D >::operator() ( const psite & p ) [inline]`

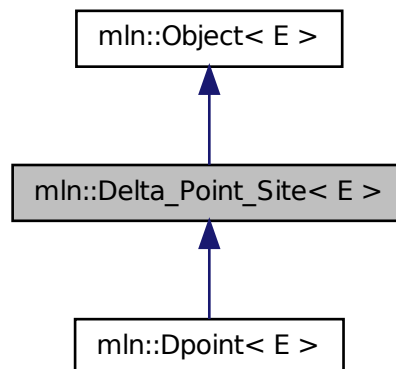
Read-write access of pixel value at point site p.

## 10.107 mln::Delta\_Point\_Site< E > Struct Template Reference

FIXME: Doc!

```
#include <delta_point_site.hh>
```

Inheritance diagram for mln::Delta\_Point\_Site< E >:



### 10.107.1 Detailed Description

```
template<typename E> struct mln::Delta_Point_Site< E >
```

FIXME: Doc!

## 10.108 mln::Delta\_Point\_Site< void > Struct Template Reference

Delta point site category flag type.

```
#include <delta_point_site.hh>
```

### 10.108.1 Detailed Description

```
template<> struct mln::Delta_Point_Site< void >
```

Delta point site category flag type.

## 10.109 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).*

### 10.109.1 Detailed Description

```
template<typename E> struct mln::doc::Accumulator< E >
```

Documentation class for [mln::Accumulator](#).

See also

[mln::Accumulator](#)

## 10.109.2 Member Typedef Documentation

### 10.109.2.1 `template<typename E> typedef void mln::doc::Accumulator< E >::argument`

The argument type of elements to accumulate.

## 10.109.3 Member Function Documentation

### 10.109.3.1 `template<typename E> void mln::doc::Accumulator< E >::init ( )`

Initialize the accumulator.

### 10.109.3.2 `template<typename E> void mln::doc::Accumulator< E >::take ( const E & other )`

Take into account another accumulator *other*.

### 10.109.3.3 `template<typename E> void mln::doc::Accumulator< E >::take ( const argument & t )`

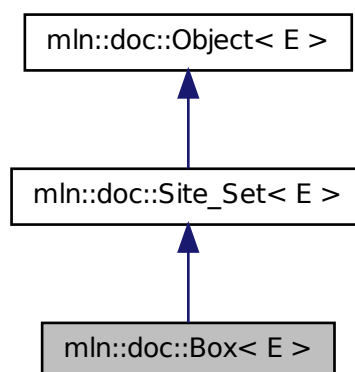
Take into account a argument *t* (an element).

## 10.110 `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.*

### 10.110.1 Detailed Description

`template<typename E> struct mln::doc::Box< E >`

Documentation class for [mln::Box](#).

See also

[mln::Box](#)

### 10.110.2 Member Typedef Documentation

**10.110.2.1** `template<typename E > typedef void mln::doc::Site_Set< E >::bkd_piter`  
[inherited]

Backward [Site\\_Iterator](#) associated type.

**10.110.2.2** `template<typename E > typedef void mln::doc::Site_Set< E >::fwd_piter`  
**[inherited]**

Forward [Site\\_Iterator](#) associated type.

**10.110.2.3** `template<typename E > typedef void mln::doc::Site_Set< E >::psite` **[inherited]**

PSite associated type.

**10.110.2.4** `template<typename E > typedef void mln::doc::Site_Set< E >::site` **[inherited]**

[Site](#) associated type.

### 10.110.3 Member Function Documentation

**10.110.3.1** `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.

**10.110.3.2** `template<typename E > bool mln::doc::Site_Set< E >::has ( const psite & p ) const`  
**[inherited]**

Test if *p* belongs to this site set.

#### Parameters

[in] *p* A psite.

#### Returns

True if *p* is an element of the site set.

**10.110.3.3** `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.



**10.110.3.4** `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.

**10.110.3.5** `template<typename E> const site& mln::doc::Box< E >::pmin ( ) const`

Give the box "minimum" point.

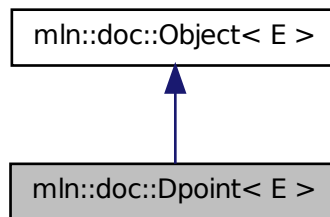
Return the "minimum" point w.r.t. the ordering between points. For instance, with `mln::box2d`, this minimum is the top left point of the box.

**10.111 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.*

### 10.111.1 Detailed Description

**template<typename E> struct mln::doc::Dpoint< E >**

Documentation class for [mln::Dpoint](#).

See also

[mln::Dpoint](#)

### 10.111.2 Member Typedef Documentation

**10.111.2.1 template<typename E > typedef void mln::doc::Dpoint< E >::coord**

Coordinate associated type.

**10.111.2.2 template<typename E > typedef void mln::doc::Dpoint< E >::dpoint**

Dpsite associated type.

**Invariant**

This type has to derive from [mln::Dpoint](#).

**10.111.2.3 template<typename E > typedef void mln::doc::Dpoint< E >::point**

[Site](#) associated type.

**Invariant**

This type has to derive from [mln::Point](#).

### 10.111.3 Member Enumeration Documentation

**10.111.3.1 template<typename E > anonymous enum**

**Enumerator:**

*dim* Dimension of the space.

**Invariant**

$dim > 0$

### 10.111.4 Member Function Documentation

**10.111.4.1 template<typename E > coord mln::doc::Dpoint< E >::operator[] ( unsigned  $i$  ) const**

Read-only access to the  $i$ -th coordinate value.

**Parameters**

[in] *i* The coordinate index.

**Precondition**

$i < \text{dim}$

**Returns**

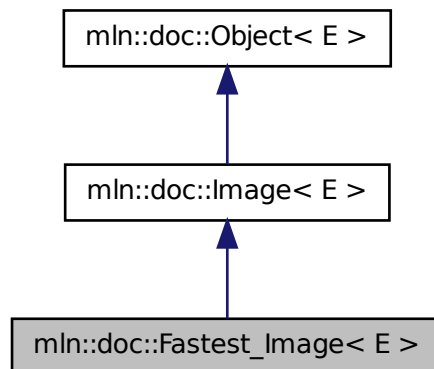
The value of the *i*-th coordinate.

## 10.112 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 the image owns the point site p.*
- bool [has](#) (const [psite](#) &p) const  
*Test if p belongs to the image domain.*

- bool `is_valid ()` const  
*Test if the image have been initialized.*
- unsigned `nelements ()` const  
*Give the number of pixels of the image including those of the virtual border.*
- unsigned `nsites ()` const  
*Give the number of points of the image domain.*
- `lvalue operator()` (const `psite &p`)  
*Read-write access to the image value located at `p`.*
- `rvalue operator()` (const `psite &p`) const  
*Read-only access to the image value located at `p`.*
- `rvalue operator[]` (unsigned `o`) const  
*Read-only access to the image value at offset `o`.*
- `lvalue operator[]` (unsigned `o`)  
*Read-write access to the image value at offset `o`.*
- `point point_at_index` (unsigned `o`) const  
*Give the point at offset `o`.*
- const `vset & values ()` const  
*Give the set of values of the image.*

### 10.112.1 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".

### 10.112.2 Member Typedef Documentation

**10.112.2.1** `template<typename E > typedef void mln::doc::Image< E >::bkd_piter`  
`[inherited]`

Backward point iterator associated type.

#### Invariant

This type has to derive from `mln::Site_Iterator`.

**10.112.2.2** `template<typename E > typedef void mln::doc::Image< E >::coord` `[inherited]`

Coordinate associated type.

**10.112.2.3** `template<typename E > typedef void mln::doc::Image< E >::dpoint` `[inherited]`

Dpsite associated type.

**Invariant**

This type has to derive from [mln::Dpoint](#).

**10.112.2.4** `template<typename E > typedef void mln::doc::Image< E >::fwd_piter`  
`[inherited]`

Forward point iterator associated type.

**Invariant**

This type has to derive from [mln::Site\\_Iterator](#).

**10.112.2.5** `template<typename E > typedef void mln::doc::Image< E >::lvalue` `[inherited]`

Type returned by the read-write pixel value operator.

**10.112.2.6** `template<typename E > typedef void mln::doc::Image< E >::point` `[inherited]`

[Site](#) associated type.

**Invariant**

This type has to derive from [mln::Point](#).

**10.112.2.7** `template<typename E > typedef void mln::doc::Image< E >::pset` `[inherited]`

[Point](#) set associated type.

**Invariant**

This type has to derive from [mln::Site\\_Set](#).

**10.112.2.8** `template<typename E > typedef void mln::doc::Image< E >::psite` `[inherited]`

[Point\\_Site](#) associated type.

**Invariant**

This type has to derive from [mln::Point\\_Site](#).

**10.112.2.9** `template<typename E > typedef void mln::doc::Image< E >::rvalue` `[inherited]`

Type returned by the read pixel value operator.

**10.112.2.10** `template<typename E > typedef void mln::doc::Image< E >::skeleton [inherited]`

Associate type that describes how this type of image is constructed.

**10.112.2.11** `template<typename E > typedef void mln::doc::Image< E >::value [inherited]`

[Value](#) associated type.

**Invariant**

This type is neither qualified by const, nor by reference.

**10.112.2.12** `template<typename E > typedef void mln::doc::Image< E >::vset [inherited]`

[Value](#) set associated type.

**Invariant**

This type has to derive from [mln::Value\\_Set](#).

**10.112.3 Member Function Documentation****10.112.3.1** `template<typename E > const box<point>& mln::doc::Image< E >::bbox ( ) const [inherited]`

Give a bounding box of the image domain.

This bounding box may be larger than the smallest bounding box (the optimal one). Practically an image type is not obliged to update its bounding box so that it is always optimal.

**Returns**

A bounding box of the image domain.

**10.112.3.2** `template<typename E > unsigned mln::doc::Fastest_Image< E >::border ( )`

Give the border thickness.

**Precondition**

The image has to be initialized.

**10.112.3.3** `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.

### 10.112.3.4 `template<typename E> int mln::doc::Fastest_Image< E >::delta_index ( const dpoint & dp )`

Give the offset corresponding to the delta-point `dp`.

#### Parameters

[in] `dp` A delta-point.

#### Precondition

The image has to be initialized.

### 10.112.3.5 `template<typename E> const pset& mln::doc::Image< E >::domain ( ) const` [inherited]

Give the definition domain of the image.

#### Returns

A reference to the domain point set.

### 10.112.3.6 `template<typename E> bool mln::doc::Image< E >::has ( const psite & p ) const` [inherited]

Test if the image owns the point site `p`.

#### Returns

True if accessing the image value at `p` is possible, that is, does not abort the execution.

### 10.112.3.7 `template<typename E> bool mln::doc::Image< E >::has ( const psite & p ) const` [inherited]

Test if `p` belongs to the image domain.

#### Parameters

[in] `p` A point site.

#### Returns

True if `p` belongs to the image domain.

#### Invariant

`has(p)` is true => `has(p)` is also true.

### 10.112.3.8 `template<typename E> bool mln::doc::Image< E >::is_valid ( ) const` [inherited]

Test if the image have been initialized.



**10.112.3.9** `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.

**10.112.3.10** `template<typename E> unsigned mln::doc::Image< E >::nsites ( ) const [inherited]`

Give the number of points of the image domain.

**10.112.3.11** `template<typename E> lvalue mln::doc::Image< E >::operator() ( const psite & p ) [inherited]`

Read-write access to the image value located at *p*.

**Parameters**

[in] *p* A point site.

**Precondition**

The image has to own the site *p*.

**Returns**

The value at *p* (assignable).

**10.112.3.12** `template<typename E> rvalue mln::doc::Image< E >::operator() ( const psite & p ) const [inherited]`

Read-only access to the image value located at *p*.

**Parameters**

[in] *p* A point site.

**Precondition**

The image has to own the site *p*.

**Returns**

The value at *p* (not assignable).

**10.112.3.13** `template<typename E > rvalue mln::doc::Fastest_Image< E >::operator[] ( unsigned o ) const`

Read-only access to the image value at offset *o*.

**Parameters**

[in] *o* An offset.

**Precondition**

*o* < [nelements\(\)](#)

**Returns**

The value at *o* (not assignable).

**10.112.3.14** `template<typename E > lvalue mln::doc::Fastest_Image< E >::operator[] ( unsigned o )`

Read-write access to the image value at offset *o*.

**Parameters**

[in] *o* An offset.

**Precondition**

*o* < [nelements\(\)](#)

**Returns**

The value at *o* (assignable).

**10.112.3.15** `template<typename E > point mln::doc::Fastest_Image< E >::point_at_index ( unsigned o ) const`

Give the point at offset *o*.

**Parameters**

[in] *o* An offset.

**Precondition**

The image has to be initialized.

*o* < [nelements\(\)](#)

**10.112.3.16** `template<typename E > const vset& mln::doc::Image< E >::values ( ) const [inherited]`

Give the set of values of the image.

**Returns**

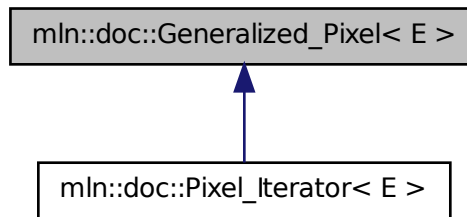
A reference to the value set.

## 10.113 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.*

#### 10.113.1 Detailed Description

```
template<typename E> struct mln::doc::Generalized_Pixel< E >
```

Documentation class for [mln::Generalized\\_Pixel](#).

See also

[mln::Generalized\\_Pixel](#)

## 10.113.2 Member Typedef Documentation

**10.113.2.1** `template<typename E > typedef void mln::doc::Generalized_Pixel< E >::image`

[Image](#) associated type (with possible const qualification).

**10.113.2.2** `template<typename E > typedef void mln::doc::Generalized_Pixel< E >::rvalue`

Read-only value associated type.

**10.113.2.3** `template<typename E > typedef void mln::doc::Generalized_Pixel< E >::value`

[Value](#) associated type.

## 10.113.3 Member Function Documentation

**10.113.3.1** `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.

**10.113.3.2** `template<typename E > rvalue mln::doc::Generalized_Pixel< E >::val ( ) const`

Give the value of this generalized pixel.

### Returns

A read-only value.

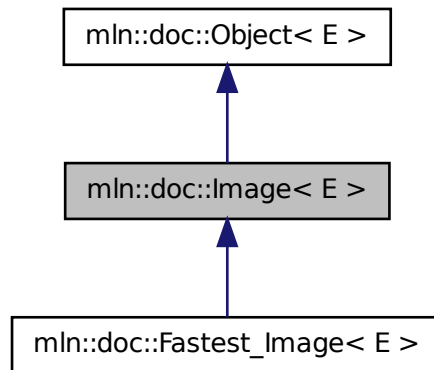
Reimplemented in [mln::doc::Pixel\\_Iterator< E >](#).

## 10.114 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 the image owns the point site p.*
- bool [has](#) (const [psite](#) &p) const  
*Test if p belongs to the image domain.*
- bool [is\\_valid](#) () const  
*Test if the image have been initialized.*
- unsigned [nsites](#) () const  
*Give the number of points of the image domain.*
- [rvalue operator](#)() (const [psite](#) &p) const  
*Read-only access to the image value located at p.*
- [lvalue operator](#)() (const [psite](#) &p)  
*Read-write access to the image value located at p.*
- const [vset](#) & [values](#) () const  
*Give the set of values of the image.*

### 10.114.1 Detailed Description

**template**<typename [E](#)> **struct** [mln::doc::Image](#)< [E](#) >

Documentation class for [mln::Image](#).

See also

[mln::Image](#)

## 10.114.2 Member Typedef Documentation

### 10.114.2.1 `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](#).

### 10.114.2.2 `template<typename E > typedef void mln::doc::Image< E >::coord`

Coordinate associated type.

### 10.114.2.3 `template<typename E > typedef void mln::doc::Image< E >::dpoint`

Dpsite associated type.

#### Invariant

This type has to derive from [mln::Dpoint](#).

### 10.114.2.4 `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](#).

### 10.114.2.5 `template<typename E > typedef void mln::doc::Image< E >::lvalue`

Type returned by the read-write pixel value operator.

### 10.114.2.6 `template<typename E > typedef void mln::doc::Image< E >::point`

[Site](#) associated type.

#### Invariant

This type has to derive from [mln::Point](#).

### 10.114.2.7 `template<typename E > typedef void mln::doc::Image< E >::pset`

[Point](#) set associated type.

#### Invariant

This type has to derive from [mln::Site\\_Set](#).

**10.114.2.8** `template<typename E > typedef void mln::doc::Image< E >::psite`

[Point\\_Site](#) associated type.

**Invariant**

This type has to derive from [mln::Point\\_Site](#).

**10.114.2.9** `template<typename E > typedef void mln::doc::Image< E >::rvalue`

Type returned by the read pixel value operator.

**10.114.2.10** `template<typename E > typedef void mln::doc::Image< E >::skeleton`

Associate type that describes how this type of image is constructed.

**10.114.2.11** `template<typename E > typedef void mln::doc::Image< E >::value`

[Value](#) associated type.

**Invariant**

This type is neither qualified by const, nor by reference.

**10.114.2.12** `template<typename E > typedef void mln::doc::Image< E >::vset`

[Value](#) set associated type.

**Invariant**

This type has to derive from [mln::Value\\_Set](#).

**10.114.3 Member Function Documentation****10.114.3.1** `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.

**10.114.3.2** `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.



**10.114.3.3** `template<typename E> bool mln::doc::Image< E >::has ( const psite & p ) const`

Test if the image owns the point site *p*.

**Returns**

True if accessing the image value at *p* is possible, that is, does not abort the execution.

**10.114.3.4** `template<typename E> bool mln::doc::Image< E >::has ( const psite & p ) const`

Test if *p* belongs to the image domain.

**Parameters**

[in] *p* A point site.

**Returns**

True if *p* belongs to the image domain.

**Invariant**

has(*p*) is true => has(*p*) is also true.

**10.114.3.5** `template<typename E> bool mln::doc::Image< E >::is_valid ( ) const`

Test if the image have been initialized.

**10.114.3.6** `template<typename E> unsigned mln::doc::Image< E >::nsites ( ) const`

Give the number of points of the image domain.

**10.114.3.7** `template<typename E> rvalue mln::doc::Image< E >::operator() ( const psite & p ) const`

Read-only access to the image value located at *p*.

**Parameters**

[in] *p* A point site.

**Precondition**

The image has to own the site *p*.

**Returns**

The value at *p* (not assignable).

### 10.114.3.8 `template<typename E> lvalue mln::doc::Image< E >::operator() ( const psite & p )`

Read-write access to the image value located at `p`.

#### Parameters

[in] `p` A point site.

#### Precondition

The image has to own the site `p`.

#### Returns

The value at `p` (assignable).

### 10.114.3.9 `template<typename E> const vset& mln::doc::Image< E >::values ( ) const`

Give the set of values of the image.

#### Returns

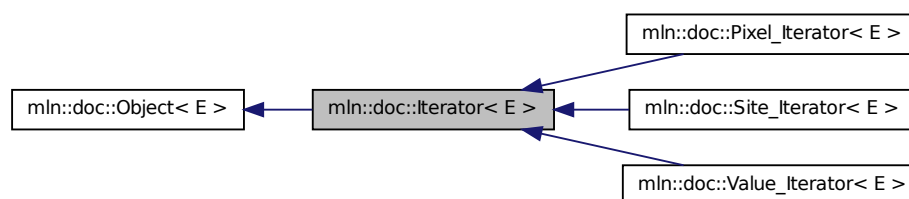
A reference to the value set.

## 10.115 `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.*

### 10.115.1 Detailed Description

`template<typename E> struct mln::doc::Iterator< E >`

Documentation class for [mln::Iterator](#).

See also

[mln::Iterator](#)

### 10.115.2 Member Function Documentation

**10.115.2.1** `template<typename E > void mln::doc::Iterator< E >::invalidate ( )`

Invalidate the iterator.

**10.115.2.2** `template<typename E > bool mln::doc::Iterator< E >::is_valid ( ) const`

Returns true if the iterator is valid, that is, designates an element.

**10.115.2.3** `template<typename E > void mln::doc::Iterator< E >::start ( )`

Start an iteration.

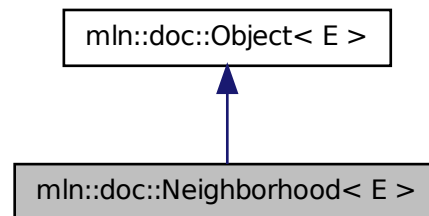
Make the iterator designate the first element if it exists. If this first element does not exist, the iterator is not valid.

## 10.116 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.

### 10.116.1 Detailed Description

`template<typename E> struct mln::doc::Neighborhood< E >`

Documentation class for `mln::Neighborhood`.

See also

[mln::Neighborhood](#)

### 10.116.2 Member Typedef Documentation

**10.116.2.1** `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.

**10.116.2.2** `template<typename E > typedef void mln::doc::Neighborhood< E >::dpoint`

Dpsite associated type.

**10.116.2.3** `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.

**10.116.2.4** `template<typename E > typedef void mln::doc::Neighborhood< E >::niter`

[Site\\_Iterator](#) type associated to this neighborhood to browse neighbors.

**10.116.2.5** `template<typename E > typedef void mln::doc::Neighborhood< E >::point`

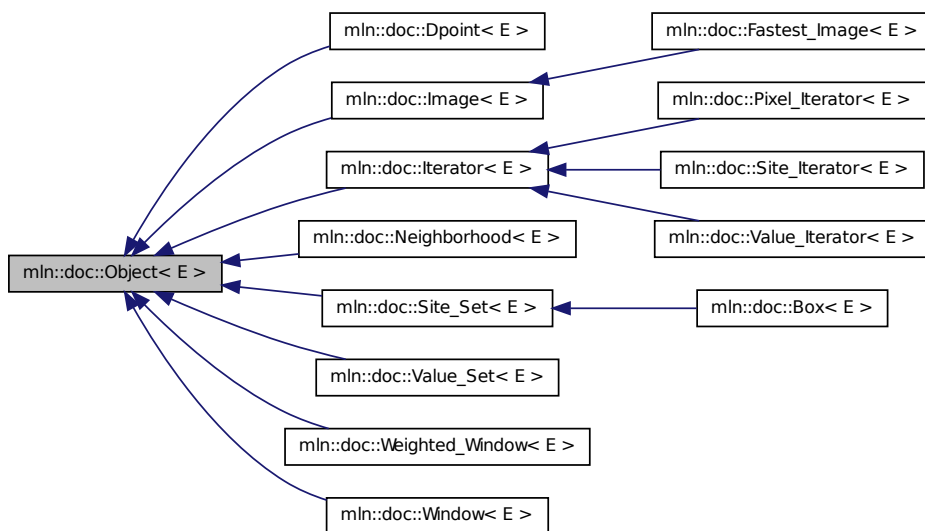
[Site](#) associated type.

## 10.117 mln::doc::Object< E > Struct Template Reference

Documentation class for [mln::Object](#).

```
#include <object.hh>
```

Inheritance diagram for mln::doc::Object< E >:



### 10.117.1 Detailed Description

`template<typename E> struct mln::doc::Object< E >`

Documentation class for [mln::Object](#).

See also

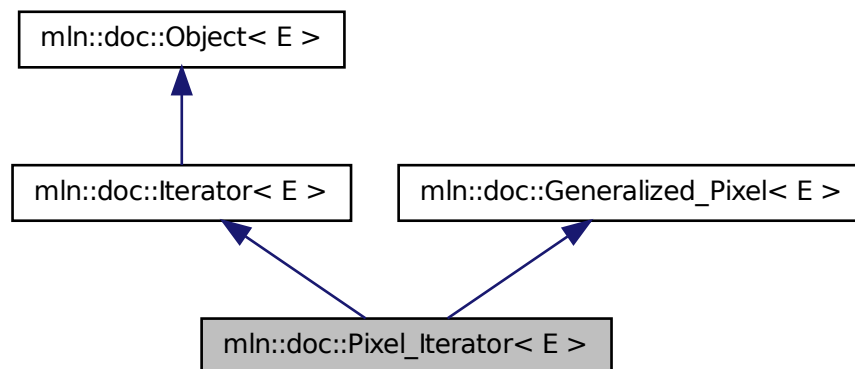
[mln::Object](#)

### 10.118 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.*

### 10.118.1 Detailed Description

`template<typename E> struct mln::doc::Pixel_Iterator< E >`

Documentation class for [mln::Iterator](#).

See also

[mln::Pixel\\_Iterator](#)

### 10.118.2 Member Typedef Documentation

**10.118.2.1** `template<typename E > typedef void mln::doc::Generalized_Pixel< E >::image`  
`[inherited]`

[Image](#) associated type (with possible const qualification).

**10.118.2.2** `template<typename E > typedef void mln::doc::Pixel_Iterator< E >::lvalue`

Type returned by the read-write dereference operator.

**10.118.2.3** `template<typename E > typedef void mln::doc::Generalized_Pixel< E >::rvalue`  
`[inherited]`

Read-only value associated type.

**10.118.2.4** `template<typename E > typedef void mln::doc::Generalized_Pixel< E >::value`  
`[inherited]`

[Value](#) associated type.

### 10.118.3 Member Function Documentation

**10.118.3.1** `template<typename E> image& mln::doc::Generalized_Pixel< E >::ima ( ) const`  
**[inherited]**

Give the image of this generalized pixel.

The constness of a pixel object is not transmitted to the underlying image.

**10.118.3.2** `template<typename E> void mln::doc::Iterator< E >::invalidate ( )`  
**[inherited]**

Invalidate the iterator.

**10.118.3.3** `template<typename E> bool mln::doc::Iterator< E >::is_valid ( ) const`  
**[inherited]**

Returns true if the iterator is valid, that is, designates an element.

**10.118.3.4** `template<typename E> void mln::doc::Iterator< E >::start ( )` **[inherited]**

Start an iteration.

Make the iterator designate the first element if it exists. If this first element does not exist, the iterator is not valid.

**10.118.3.5** `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 >](#).

## 10.119 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.*

### 10.119.1 Detailed Description

`template<typename E> struct mln::doc::Point_Site< E >`

Documentation class for [mln::Point\\_Site](#).

See also

[mln::Point\\_Site](#)

### 10.119.2 Member Typedef Documentation

**10.119.2.1 `template<typename E > typedef void mln::doc::Point_Site< E >::coord`**

Coordinate associated type.

**10.119.2.2 `template<typename E > typedef void mln::doc::Point_Site< E >::dpoint`**

Dpsite associated type.

#### Invariant

This type has to derive from [mln::Dpoint](#).

**10.119.2.3 `template<typename E > typedef void mln::doc::Point_Site< E >::mesh`**

[Mesh](#) associated type.

#### Invariant

This type has to derive from [mln::Mesh](#).

**10.119.2.4** `template<typename E > typedef void mln::doc::Point_Site< E >::point`

[Site](#) associated type.

**Invariant**

This type has to derive from [mln::Point](#).

**10.119.3 Member Enumeration Documentation****10.119.3.1** `template<typename E > anonymous enum`**Enumerator:**

*dim* Dimension of the space.

**Invariant**

`dim > 0`

**10.119.4 Member Function Documentation****10.119.4.1** `template<typename E > coord mln::doc::Point_Site< E >::operator[] ( unsigned i ) const`

Read-only access to the `i`-th coordinate value.

**Parameters**

[in] *i* The coordinate index.

**Precondition**

`i < dim`

**Returns**

The value of the `i`-th coordinate.

**10.119.4.2** `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**

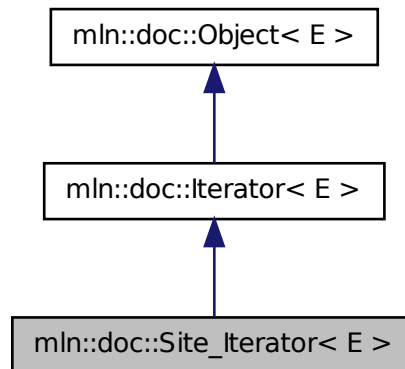
A point constant reference.

**10.120 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.*

### 10.120.1 Detailed Description

`template<typename E> struct mln::doc::Site_Iterator< E >`

Documentation class for `mln::Site_Iterator`.

See also

[mln::Site\\_Iterator](#)

## 10.120.2 Member Typedef Documentation

### 10.120.2.1 `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`.

## 10.120.3 Member Function Documentation

### 10.120.3.1 `template<typename E > void mln::doc::Iterator< E >::invalidate ( )` [`inherited`]

Invalidate the iterator.

### 10.120.3.2 `template<typename E > bool mln::doc::Iterator< E >::is_valid ( ) const` [`inherited`]

Returns true if the iterator is valid, that is, designates an element.

### 10.120.3.3 `template<typename E > mln::doc::Site_Iterator< E >::operator psite ( ) const`

Conversion into a point-site.

#### Returns

A point site.

### 10.120.3.4 `template<typename E > void mln::doc::Iterator< E >::start ( )` [`inherited`]

Start an iteration.

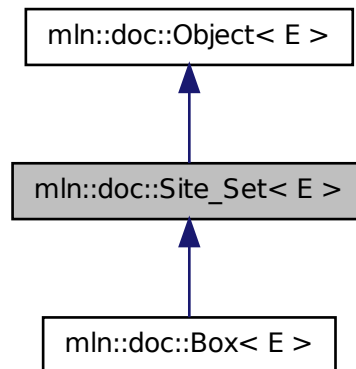
Make the iterator designate the first element if it exists. If this first element does not exist, the iterator is not valid.

## 10.121 `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.*

### 10.121.1 Detailed Description

`template<typename E> struct mln::doc::Site_Set< E >`

Documentation class for [mln::Site\\_Set](#).

See also

[mln::Site\\_Set](#)

### 10.121.2 Member Typedef Documentation

#### 10.121.2.1 `template<typename E > typedef void mln::doc::Site_Set< E >::bkd_piter`

Backward [Site\\_Iterator](#) associated type.

#### 10.121.2.2 `template<typename E > typedef void mln::doc::Site_Set< E >::fwd_piter`

Forward [Site\\_Iterator](#) associated type.

#### 10.121.2.3 `template<typename E > typedef void mln::doc::Site_Set< E >::psite`

PSite associated type.

#### 10.121.2.4 `template<typename E > typedef void mln::doc::Site_Set< E >::site`

[Site](#) associated type.

### 10.121.3 Member Function Documentation

#### 10.121.3.1 `template<typename E > bool mln::doc::Site_Set< E >::has ( const psite & p ) const`

Test if `p` belongs to this site set.

#### Parameters

[in] `p` A psite.

#### Returns

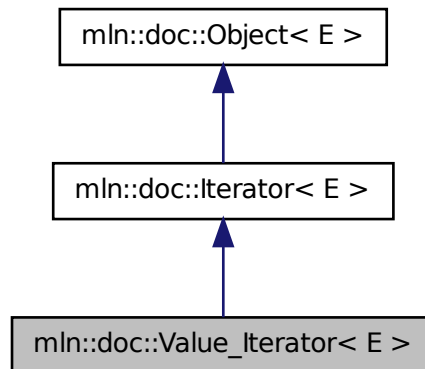
True if `p` is an element of the site set.

## 10.122 `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.*

### 10.122.1 Detailed Description

`template<typename E> struct mln::doc::Value_Iterator< E >`

Documentation class for [mln::Value\\_Iterator](#).

See also

[mln::Value\\_Iterator](#)

## 10.122.2 Member Typedef Documentation

**10.122.2.1** `template<typename E > typedef void mln::doc::Value_Iterator< E >::value`

[Value](#) associated type.

## 10.122.3 Member Function Documentation

**10.122.3.1** `template<typename E > void mln::doc::Iterator< E >::invalidate ( )`  
[[inherited](#)]

Invalidate the iterator.

**10.122.3.2** `template<typename E > bool mln::doc::Iterator< E >::is_valid ( ) const`  
[[inherited](#)]

Returns true if the iterator is valid, that is, designates an element.

**10.122.3.3** `template<typename E > mln::doc::Value_Iterator< E >::operator value ( ) const`

Conversion into a value.

### Returns

A value.

**10.122.3.4** `template<typename E > void mln::doc::Iterator< E >::start ( )` [[inherited](#)]

Start an iteration.

Make the iterator designate the first element if it exists. If this first element does not exist, the iterator is not valid.

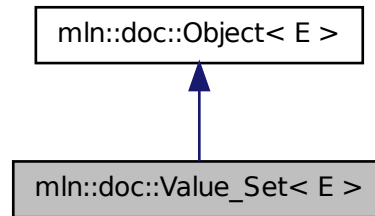
## 10.123 mln::doc::Value\_Set< E > Struct Template Reference

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.*

### 10.123.1 Detailed Description

template<typename E> struct mln::doc::Value\_Set< E >

Documentation class for [mln::Value\\_Set](#).

See also

[mln::Value\\_Set](#)

## 10.123.2 Member Typedef Documentation

**10.123.2.1** `template<typename E> typedef void mln::doc::Value_Set< E >::bkd_viter`

Backward [Value\\_Iterator](#) associated type.

**10.123.2.2** `template<typename E> typedef void mln::doc::Value_Set< E >::fwd_viter`

Forward [Value\\_Iterator](#) associated type.

**10.123.2.3** `template<typename E> typedef void mln::doc::Value_Set< E >::value`

[Value](#) associated type.

## 10.123.3 Member Function Documentation

**10.123.3.1** `template<typename E> bool mln::doc::Value_Set< E >::has ( const value & v ) const`

Test if  $v$  belongs to this set of values.

### Parameters

[in]  $v$  A value.

### Returns

True if  $v$  is an element of the set of values.

**10.123.3.2** `template<typename E> unsigned mln::doc::Value_Set< E >::index_of ( const value & v ) const`

Give the index of value  $v$  in this set.

**10.123.3.3** `template<typename E> unsigned mln::doc::Value_Set< E >::nvalues ( ) const`

Give the number of values in this set.

**10.123.3.4** `template<typename E> value mln::doc::Value_Set< E >::operator[] ( unsigned i ) const`

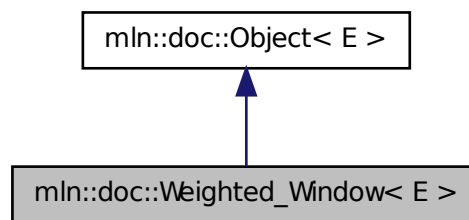
Give the  $i$ -th value of this set.

## 10.124 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`.*

### 10.124.1 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](#)

### 10.124.2 Member Typedef Documentation

**10.124.2.1** `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.

**10.124.2.2** `template<typename E > typedef void mln::doc::Weighted_Window< E >::dpoint`

Dpsite associated type.

**10.124.2.3** `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.

**10.124.2.4** `template<typename E > typedef void mln::doc::Weighted_Window< E >::point`

[Site](#) associated type.

**10.124.2.5** `template<typename E > typedef void mln::doc::Weighted_Window< E >::weight`

Weight associated type.

**10.124.2.6** `template<typename E > typedef void mln::doc::Weighted_Window< E >::window`

[Window](#) associated type.

### 10.124.3 Member Function Documentation

#### 10.124.3.1 `template<typename E> unsigned mln::doc::Weighted_Window< E >::delta ( ) const`

Give the maximum coordinate gap between the window center and a window point.

#### 10.124.3.2 `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.

#### 10.124.3.3 `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.

#### 10.124.3.4 `template<typename E> E& mln::doc::Weighted_Window< E >::sym ( )`

Apply a central symmetry to the target weighted window.

#### 10.124.3.5 `template<typename E> const window& mln::doc::Weighted_Window< E >::win ( ) const`

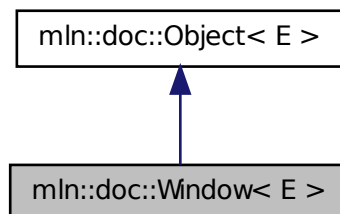
Give the corresponding window.

## 10.125 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.

### 10.125.1 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](#)

### 10.125.2 Member Typedef Documentation

**10.125.2.1** `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.

**10.125.2.2** `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.

**10.125.2.3** `template<typename E > typedef void mln::doc::Window< E >::qiter`

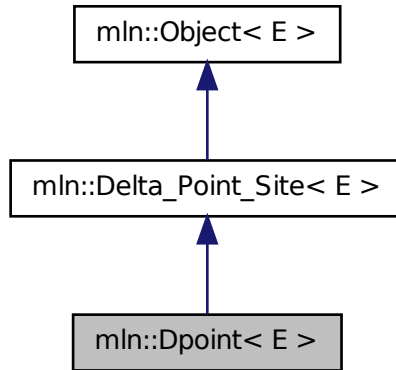
[Site\\_Iterator](#) type associated to this window to browse its points.

## 10.126 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.*

### 10.126.1 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.

### 10.126.2 Member Function Documentation

**10.126.2.1** `template<typename E > const E & mln::Dpoint< E >::to_dpoint ( ) const`  
`[inline]`

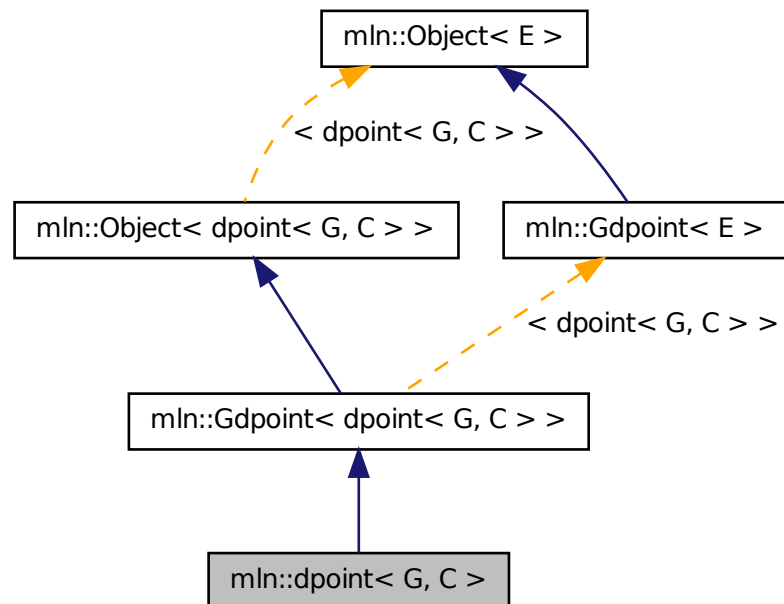
It is a [Dpoint](#) so it returns itself.

## 10.127 mln::dpoint< G, C > Struct Template Reference

Generic delta-point class.

```
#include <dpoint.hh>
```

Inheritance diagram for `mln::dpoint< G, C >`:



## Public Types

- enum { `dim = G::dim` }
- typedef C `coord`  
*Coordinate associated type.*
- typedef G `grid`  
*Grid associated type.*
- typedef `point< G, C >` `psite`  
*Psite associated type.*
- typedef `point< G, C >` `site`  
*Site associated type.*
- typedef `algebra::vec< G::dim, C >` `vec`  
*Algebra vector (vec) associated type.*



## Public Member Functions

- [dpoint](#) ()  
*Constructor without argument.*
- `template<typename C2 >`  
[dpoint](#) (const algebra::vec< dim, C2 > &v)  
*Constructor from an algebra vector.*
- `template<typename F >`  
[dpoint](#) (const [Function\\_v2v](#)< F > &f)  
*Constructor; coordinates are set by function  $f$ .*
- `template<typename Q >`  
[operator mln::algebra::vec< dpoint< G, C >::dim, Q > \(\)](#) const  
*Conversion towards a algebra::vec.*
- `C` [operator\[\]](#) (unsigned i) const  
*Read-only access to the  $i$ -th coordinate value.*
- `C &` [operator\[\]](#) (unsigned i)  
*Read-write access to the  $i$ -th coordinate value.*
- void [set\\_all](#) (C c)  
*Set all coordinates to the value  $c$ .*
- [vec to\\_vec](#) () const  
*Explicit conversion.*
- [dpoint](#) (C ind)
- [dpoint](#) (const [literal::zero\\_t](#) &)  
*Constructors/assignments with literals.*

### 10.127.1 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.

### 10.127.2 Member Typedef Documentation

**10.127.2.1** `template<typename G, typename C> typedef C mln::dpoint< G, C >::coord`

Coordinate associated type.

**10.127.2.2** `template<typename G, typename C> typedef G mln::dpoint< G, C >::grid`

Grid associated type.

**10.127.2.3** `template<typename G, typename C> typedef point<G,C> mln::dpoint< G, C >::psite`

Psite associated type.

**10.127.2.4** `template<typename G, typename C> typedef point<G,C> mln::dpoint< G, C >::site`

[Site](#) associated type.

**10.127.2.5** `template<typename G, typename C> typedef algebra::vec<G::dim, C> mln::dpoint< G, C >::vec`

Algebra vector (vec) associated type.

**10.127.3 Member Enumeration Documentation****10.127.3.1** `template<typename G, typename C> anonymous enum`

**Enumerator:**

*dim* Dimension of the space.

**Invariant**

`dim > 0`

**10.127.4 Constructor & Destructor Documentation****10.127.4.1** `template<typename G , typename C > mln::dpoint< G, C >::dpoint ( ) [inline]`

Constructor without argument.

**10.127.4.2** `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`.

**10.127.4.3** `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.

**10.127.4.4** `template<typename G , typename C> mln::dpoint< G, C >::dpoint ( const literal::zero_t & ) [inline]`

Constructors/assignments with literals.

**10.127.4.5** `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.

## 10.127.5 Member Function Documentation

**10.127.5.1** `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().

**10.127.5.2** `template<typename G , typename C > C & mln::dpoint< G, C >::operator[] ( unsigned i ) [inline]`

Read-write access to the i-th coordinate value.

### Parameters

[in] *i* The coordinate index.

### Precondition

$i < \text{dim}$

References mln::dpoint< G, C >::dim.

**10.127.5.3** `template<typename G , typename C > C mln::dpoint< G, C >::operator[] ( unsigned i ) const [inline]`

Read-only access to the i-th coordinate value.

### Parameters

[in] *i* The coordinate index.

### Precondition

$i < \text{dim}$

References mln::dpoint< G, C >::dim.

**10.127.5.4** `template<typename G , typename C> void mln::dpoint< G, C >::set_all ( C c ) [inline]`

Set all coordinates to the value c.

Referenced by mln::win::line< M, i, C >::line().

### 10.127.5.5 `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`.

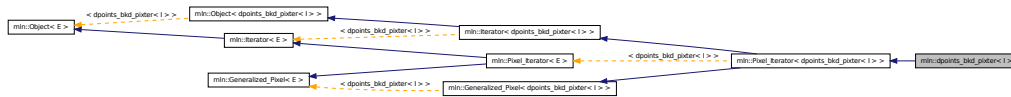
Referenced by `mln::dpoint< G, C >::operator mln::algebra::vec< dpoint< G, C >::dim, Q >()`.

## 10.128 `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>
```

Inheritance diagram for `mln::dpoints_bkd_pixter< I >`:



### Public Member Functions

- `const I::value & center_val () const`  
*The value around which this iterator moves.*
- `template<typename Dps , typename Pref > dpoints_bkd_pixter (const Generalized_Pixel< Pref > &pxl_ref, const Dps &dps)`  
*Constructor (using a generalized pixel).*
- `template<typename Dps , typename Pref > dpoints_bkd_pixter (I &image, const Dps &dps, const Pref &p_ref)`  
*Constructor (using an image).*
- `void next ()`  
*Go to the next element.*
- `void start ()`  
*Manipulation.*
- `void invalidate ()`  
*Invalidate the iterator.*
- `bool is_valid () const`  
*Test the iterator validity.*
- `void update ()`  
*Force this iterator to update its location to take into account that its center point may have moved.*

## 10.128.1 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.

## 10.128.2 Constructor & Destructor Documentation

**10.128.2.1 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

- [in] *image* The image to iterate over.
- [in] *dps* An object (neighborhood or window) that can provide a set of delta-points.
- [in] *p\_ref* Center (resp. reference) point of the neighborhood (resp. window).

**10.128.2.2 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

- [in] *pxl\_ref* Center (generalized) pixel to iterate around.
- [in] *dps* An object (neighborhood or window) that can provide a set of delta-points.

## 10.128.3 Member Function Documentation

**10.128.3.1 template<typename I > const I::value & mln::dpoints\_bkd\_pixter< I >::center\_val ( ) const [inline]**

The value around which this iterator moves.

**10.128.3.2 template<typename I > void mln::dpoints\_bkd\_pixter< I >::invalidate ( ) [inline]**

Invalidate the iterator.

**10.128.3.3 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().

### 10.128.3.4 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.

### 10.128.3.5 template<typename I > void mln::dpoints\_bkd\_pixter< I >::start ( ) [inline]

Manipulation.

Start an iteration.

References mln::dpoints\_bkd\_pixter< I >::update().

### 10.128.3.6 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().

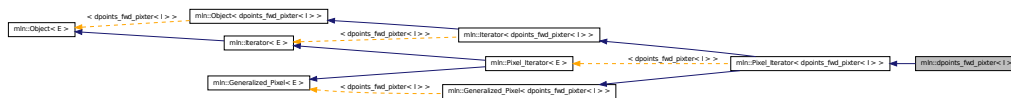
Referenced by mln::dpoints\_bkd\_pixter< I >::start().

## 10.129 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>
```

Inheritance diagram for mln::dpoints\_fwd\_pixter< I >:



### Public Member Functions

- const I::value & [center\\_val](#) () const  
*The value around which this iterator moves.*
- template<typename Dps , typename Pref >  
[dpoints\\_fwd\\_pixter](#) (const [Generalized\\_Pixel](#)< Pref > &pxl\_ref, const Dps &dps)  
*Constructor (using a generalized pixel).*

- `template<typename Dps , typename Pref >`  
`dpoints_fwd_pixter` (I &image, const Dps &dps, const Pref &p\_ref)  
*Constructor (using an image).*
- `void next ()`  
*Go to the next element.*
- `void start ()`  
*Manipulation.*
- `void invalidate ()`  
*Invalidate the iterator.*
- `bool is_valid () const`  
*Test the iterator validity.*
- `void update ()`  
*Force this iterator to update its location to take into account that its center point may have moved.*

### 10.129.1 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.

### 10.129.2 Constructor & Destructor Documentation

**10.129.2.1** `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

- [in] *image* The image to iterate over.
- [in] *dps* An object (neighborhood or window) that can provide a set of delta-points.
- [in] *p\_ref* Center (resp. reference) point of the neighborhood (resp. window).

**10.129.2.2** `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

- [in] *pxl\_ref* Center (generalized) pixel to iterate around.
- [in] *dps* An object (neighborhood or window) that can provide a set of delta-points.

### 10.129.3 Member Function Documentation

**10.129.3.1** `template<typename I> const I::value & mln::dpoints_fwd_pixter< I >::center_val ( ) const [inline]`

The value around which this iterator moves.

**10.129.3.2** `template<typename I> void mln::dpoints_fwd_pixter< I >::invalidate ( ) [inline]`

Invalidate the iterator.

**10.129.3.3** `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()`.

**10.129.3.4** `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.

**10.129.3.5** `template<typename I> void mln::dpoints_fwd_pixter< I >::start ( ) [inline]`

Manipulation.

Start an iteration.

References `mln::dpoints_fwd_pixter< I >::update()`.

**10.129.3.6** `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()`.

## 10.130 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

- `template<typename P > dpsites_bkd_piter (const V &v, const P &c)`

*Constructor.*

- `dpsites_bkd_piter ()`

*Constructor without argument.*

- `void next ()`

*Go to the next element.*

### 10.130.1 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.

### 10.130.2 Constructor & Destructor Documentation

**10.130.2.1** `template<typename V > template<typename P > mln::dpsites_bkd_piter< V >::dpsites_bkd_piter ( const V & v, const P & c ) [inline]`

Constructor.

#### Parameters

[in] `v` **Object** that can provide an array of delta-points.

[in] `c` Center point to iterate around.

**10.130.2.2** `template<typename V > mln::dpsites_bkd_piter< V >::dpsites_bkd_piter ( ) [inline]`

Constructor without argument.

### 10.130.3 Member Function Documentation

**10.130.3.1** `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-define this method. The actual "next" operation has to be defined through the *next\_* method.

**Precondition**

The iterator is valid.

**10.131 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**

- `template<typename P > dpsites_fwd_piter (const V &v, const P &c)`  
*Constructor.*
- `dpsites_fwd_piter ()`  
*Constructor without argument.*
- `void next ()`  
*Go to the next element.*

**10.131.1 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.

**10.131.2 Constructor & Destructor Documentation**

```
10.131.2.1 template<typename V > template<typename P > mln::dpsites_fwd_piter< V >::dpsites_fwd_piter ( const V & v, const P & c ) [inline]
```

Constructor.

**Parameters**

- [in] `v` [Object](#) that can provide an array of delta-points.
- [in] `c` Center point to iterate around.

**10.131.2.2** `template<typename V > mln::dpsites_fwd_piter< V >::dpsites_fwd_piter ( )`  
`[inline]`

Constructor without argument.

### 10.131.3 Member Function Documentation

**10.131.3.1** `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.

## 10.132 mln::Edge< E > Struct Template Reference

edge category flag type.

```
#include <edge.hh>
```

### 10.132.1 Detailed Description

`template<typename E> struct mln::Edge< E >`

edge category flag type.

## 10.133 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 `graph_elt_neighborhood< G, p_edges< G, site_function_t >> edge_nbh_t`  
*Neighborhood* type.
- typedef `graph_elt_window< G, p_edges< G, site_function_t >> edge_win_t`  
*Edge Window* type.
- typedef G `graph_t`

The type of the underlying graph.

- typedef [edge\\_nbh\\_t](#) `nbh_t`  
Default *Neighborhood* type.
- typedef `internal::efsite_selector< P, G >::site_function_t` `site_function_t`  
*Function* mapping graph elements to sites.
- typedef [edge\\_image](#)< tag::psite\_< P >, tag::value\_< V >, tag::graph\_< G > > `skeleton`  
*Skeleton* type.
- typedef [edge\\_win\\_t](#) `win_t`  
Default *Window* type.

## Public Member Functions

- [edge\\_image](#) ()  
*Constructors.*
- rvalue [operator\(\)](#) (unsigned e\_id) const  
*Value accessors/operators overloads.*

### 10.133.1 Detailed Description

`template<typename P, typename V, typename G = util::graph> class mln::edge_image< P, V, G >`

[Image](#) based on graph edges.

### 10.133.2 Member Typedef Documentation

**10.133.2.1** `template<typename P, typename V, typename G = util::graph> typedef graph_elt_neighborhood<G,p_edges<G,site_function_t> > mln::edge_image< P, V, G >::edge_nbh_t`

[Neighborhood](#) type.

**10.133.2.2** `template<typename P, typename V, typename G = util::graph> typedef graph_elt_window<G,p_edges<G,site_function_t> > mln::edge_image< P, V, G >::edge_win_t`

[Edge Window](#) type.

**10.133.2.3** `template<typename P, typename V, typename G = util::graph> typedef G mln::edge_image< P, V, G >::graph_t`

The type of the underlying graph.

**10.133.2.4** `template<typename P, typename V, typename G = util::graph> typedef edge_nbh_t mln::edge_image< P, V, G >::nbh_t`

Default [Neighborhood](#) type.

**10.133.2.5** `template<typename P, typename V, typename G = util::graph> typedef internal::efsite_selector<P,G>::site_function_t mln::edge_image< P, V, G >::site_function_t`

[Function](#) mapping graph elements to sites.

**10.133.2.6** `template<typename P, typename V, typename G = util::graph> typedef edge_image< tag::psite_<P>, tag::value_<V>, tag::graph_<G> > mln::edge_image< P, V, G >::skeleton`

Skeleton type.

**10.133.2.7** `template<typename P, typename V, typename G = util::graph> typedef edge_win_t mln::edge_image< P, V, G >::win_t`

Default [Window](#) type.

### 10.133.3 Constructor & Destructor Documentation

**10.133.3.1** `template<typename P, typename V, typename G > mln::edge_image< P, V, G >::edge_image( ) [inline]`

Constructors.

### 10.133.4 Member Function Documentation

**10.133.4.1** `template<typename P, typename V, typename G > edge_image< P, V, G >::rvalue mln::edge_image< P, V, G >::operator() ( unsigned e_id ) const`

[Value](#) accessors/operators overloads.

## 10.134 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](#) ()  
*Constructor without argument.*
- [extended](#) ( I &ima, const [box](#)< typename I::site > &b)  
*Constructor.*

### 10.134.1 Detailed Description

`template<typename I> struct mln::extended< I >`

Makes an image become restricted by a point set.

### 10.134.2 Member Typedef Documentation

**10.134.2.1** `template<typename I> typedef tag::image_<I> mln::extended< I >::skeleton`

Skeleton.

**10.134.2.2** `template<typename I> typedef I ::value mln::extended< I >::value`

[Value](#) type.

### 10.134.3 Constructor & Destructor Documentation

**10.134.3.1** `template<typename I > mln::extended< I >::extended ( ) [inline]`

Constructor without argument.

**10.134.3.2** `template<typename I > mln::extended< I >::extended ( I & ima, const box< typename I::site > & b ) [inline]`

Constructor.

### 10.134.4 Member Function Documentation

**10.134.4.1** `template<typename I > const box< typename I::site > & mln::extended< I >::domain ( ) const [inline]`

Give the definition domain.

## 10.135 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`.*

### 10.135.1 Detailed Description

**template<typename I, typename F> class mln::extension\_fun< I, F >**

Extends the domain of an image with a function.

### 10.135.2 Member Typedef Documentation

**10.135.2.1 template<typename I, typename F> typedef I ::value mln::extension\_fun< I, F >::rvalue**

Return type of read-only access.

**10.135.2.2 template<typename I, typename F> typedef extension\_fun< tag::image\_<I>, tag::function\_<F> > mln::extension\_fun< I, F >::skeleton**

Skeleton.

**10.135.2.3 template<typename I, typename F> typedef I ::value mln::extension\_fun< I, F >::value**

[Image](#) value type.

### 10.135.3 Constructor & Destructor Documentation

**10.135.3.1 template<typename I, typename F > mln::extension\_fun< I, F >::extension\_fun ( ) [inline]**

Constructor without argument.

**10.135.3.2 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*.

### 10.135.4 Member Function Documentation

**10.135.4.1 template<typename I, typename F > const F & mln::extension\_fun< I, F >::extension ( ) const [inline]**

Give the extension function.

**10.135.4.2 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*.



**10.135.4.3** `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*.

**10.135.4.4** `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*;

## 10.136 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;*

### 10.136.1 Detailed Description

`template<typename I, typename J> class mln::extension_ima< I, J >`

Extends the domain of an image with an image.

### 10.136.2 Member Typedef Documentation

**10.136.2.1** `template<typename I, typename J> typedef I ::value mln::extension_ima< I, J >::rvalue`

Return type of read-only access.

**10.136.2.2** `template<typename I, typename J> typedef extension_ima< tag::image_<I>, tag::ext_<J> > mln::extension_ima< I, J >::skeleton`

Skeleton.

**10.136.2.3** `template<typename I, typename J> typedef I ::value mln::extension_ima< I, J >::value`

[Image](#) value type.

### 10.136.3 Constructor & Destructor Documentation

**10.136.3.1** `template<typename I, typename J> mln::extension_ima< I, J >::extension_ima ( ) [inline]`

Constructor without argument.

**10.136.3.2** `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`.

### 10.136.4 Member Function Documentation

**10.136.4.1** `template<typename I, typename J> const J & mln::extension_ima< I, J >::extension ( ) const [inline]`

Read-only access to the extension domain (image).

**10.136.4.2** `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.

**10.136.4.3** `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`.

**10.136.4.4** `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`;

## 10.137 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  $p$ .*

### 10.137.1 Detailed Description

`template<typename I> class mln::extension_val< I >`

Extends the domain of an image with a value.

### 10.137.2 Member Typedef Documentation

**10.137.2.1 `template<typename I> typedef I::value mln::extension_val< I >::rvalue`**

Return type of read-only access.

**10.137.2.2 `template<typename I> typedef extension_val< tag::image_<I> > mln::extension_val< I >::skeleton`**

Skeleton.

**10.137.2.3 `template<typename I> typedef I::value mln::extension_val< I >::value`**

[Image](#) value type.

### 10.137.3 Constructor & Destructor Documentation

**10.137.3.1 `template<typename I > mln::extension_val< I >::extension_val ( ) [inline]`**

Constructor without argument.

**10.137.3.2 `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`.

### 10.137.4 Member Function Documentation

**10.137.4.1** `template<typename I> void mln::extension_val< I >::change_extension ( const typename I::value & val ) [inline]`

Change the value of the extension domain.

**10.137.4.2** `template<typename I> const I::value & mln::extension_val< I >::extension ( ) const [inline]`

Read-only access to the value of the extension domain.

**10.137.4.3** `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.

**10.137.4.4** `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.

**10.137.4.5** `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;.

## 10.138 mln::faces\_psite< N, D, P > Class Template Reference

[Point](#) site associated to a [mln::p\\_faces](#).

```
#include <faces_psite.hh>
```

Inherits `pseudo_site_base_< const P &, faces_psite< N, D, P > >`.

### Public Member Functions

- [faces\\_psite](#) ()  
*Construction and assignment.*
- [faces\\_psite](#) (const [p\\_faces](#)< N, D, P > &pf, const [topo::n\\_face](#)< N, D > &face)
- bool [is\\_valid](#) () const  
*Psite manipulators.*
- void [invalidate](#) ()  
*Invalidate this psite.*

- const `target & site_set ()` const  
*Site set manipulators.*
- void `change_target (const target &new_target)`  
*Set the target site\_set.*
- `topo::n_face< N, D > face ()` const  
*Face handle manipulators.*
- unsigned `n ()` const  
*Return the dimension of the face of this psite.*
- unsigned `face_id ()` const  
*Return the id of the face of this psite.*

### 10.138.1 Detailed Description

`template<unsigned N, unsigned D, typename P> class mln::faces_psite< N, D, P >`

[Point](#) site associated to a [mln::p\\_faces](#).

#### Template Parameters

- N* The dimension of the face associated to this psite.
- D* The dimension of the complex this psite belongs to.
- P* The type of point associated to this psite.

### 10.138.2 Constructor & Destructor Documentation

**10.138.2.1** `template<unsigned N, unsigned D, typename P > mln::faces_psite< N, D, P >::faces_psite ( ) [inline]`

Construction and assignment.

References `mln::faces_psite< N, D, P >::invalidate()`.

**10.138.2.2** `template<unsigned N, unsigned D, typename P > mln::faces_psite< N, D, P >::faces_psite ( const p_faces< N, D, P > & pf, const topo::n_face< N, D > & face ) [inline]`

#### Precondition

`pf.cplx() == face.cplx()`.

### 10.138.3 Member Function Documentation

**10.138.3.1** `template<unsigned N, unsigned D, typename P > void mln::faces_psite< N, D, P >::change_target ( const target & new_target ) [inline]`

Set the target site\_set.

References mln::p\_faces< N, D, P >::cplx(), and mln::faces\_psite< N, D, P >::invalidate().

**10.138.3.2** `template<unsigned N, unsigned D, typename P > topo::n_face< N, D >  
mln::faces_psite< N, D, P >::face ( ) const [inline]`

Face handle manipulators.

Return the face handle of this point site.

Referenced by mln::operator!==( ), and mln::operator==( ).

**10.138.3.3** `template<unsigned N, unsigned D, typename P > unsigned mln::faces_psite< N, D, P  
>::face_id ( ) const [inline]`

Return the id of the face of this psite.

**10.138.3.4** `template<unsigned N, unsigned D, typename P > void mln::faces_psite< N, D, P  
>::invalidate ( ) [inline]`

Invalidate this psite.

Referenced by mln::faces\_psite< N, D, P >::change\_target(), and mln::faces\_psite< N, D, P >::faces\_psite().

**10.138.3.5** `template<unsigned N, unsigned D, typename P > bool mln::faces_psite< N, D, P  
>::is_valid ( ) const [inline]`

Psite manipulators.

Is this psite valid?

**10.138.3.6** `template<unsigned N, unsigned D, typename P > unsigned mln::faces_psite< N, D, P  
>::n ( ) const [inline]`

Return the dimension of the face of this psite.

**10.138.3.7** `template<unsigned N, unsigned D, typename P > const p_faces< N, D, P > &  
mln::faces_psite< N, D, P >::site_set ( ) const [inline]`

[Site](#) set manipulators.

Return the [p\\_faces](#) this site is built on. (shortcut for \*target()).

### Precondition

Member face\_ is valid.

Referenced by mln::operator!==( ), and mln::operator==( ).

## 10.139 mln::flat\_image< T, S > Struct Template Reference

[Image](#) with a single value.

```
#include <flat_image.hh>
```

Inherits [image\\_primary< T, S, flat\\_image< T, S > >](#).

### Public Types

- typedef T & [lvalue](#)  
*Return type of read-write access.*
- typedef const T & [rvalue](#)  
*Return type of read-only access.*
- typedef [flat\\_image](#)< tag::value\_< T >, tag::domain\_< S > > [skeleton](#)  
*Skeleton.*
- typedef T [value](#)  
*Value associated type.*

### Public Member Functions

- const S & [domain](#) () const  
*Give the definition domain.*
- [flat\\_image](#) (const T &val, const S &pset)  
*Constructor.*
- [flat\\_image](#) ()  
*Constructor without argument.*
- bool [has](#) (const typename S::psite &p) const  
*Test if p is valid: always return true.*
- const T & [operator\(\)](#) (const typename S::psite &p) const  
*Read-only access to the image value located at point p.*
- T & [operator\(\)](#) (const typename S::psite &p)  
*Read-write access to the image value located at point p.*

### 10.139.1 Detailed Description

```
template<typename T, typename S> struct mln::flat_image< T, S >
```

[Image](#) with a single value.



## 10.139.2 Member Typedef Documentation

### 10.139.2.1 `template<typename T, typename S> typedef T& mln::flat_image< T, S >::lvalue`

Return type of read-write access.

### 10.139.2.2 `template<typename T, typename S> typedef const T& mln::flat_image< T, S >::rvalue`

Return type of read-only access.

### 10.139.2.3 `template<typename T, typename S> typedef flat_image< tag::value_<T>, tag::domain_<S> > mln::flat_image< T, S >::skeleton`

Skeleton.

### 10.139.2.4 `template<typename T, typename S> typedef T mln::flat_image< T, S >::value`

[Value](#) associated type.

## 10.139.3 Constructor & Destructor Documentation

### 10.139.3.1 `template<typename T , typename S > mln::flat_image< T, S >::flat_image ( ) [inline]`

Constructor without argument.

### 10.139.3.2 `template<typename T , typename S > mln::flat_image< T, S >::flat_image ( const T & val, const S & pset ) [inline]`

Constructor.

## 10.139.4 Member Function Documentation

### 10.139.4.1 `template<typename T , typename S > const S & mln::flat_image< T, S >::domain ( ) const [inline]`

Give the definition domain.

### 10.139.4.2 `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.

### 10.139.4.3 `template<typename T , typename S > const T & mln::flat_image< T, S >::operator() ( const typename S::psite & p ) const [inline]`

Read-only access to the image value located at point *p*.

**10.139.4.4** `template<typename T , typename S > T & mln::flat_image< T, S >::operator() ( const typename S::psite & p ) [inline]`

Read-write access to the image value located at point p.

## 10.140 `mln::fun::from_accu< A >` Struct Template Reference

Wrap an accumulator into a function.

```
#include <from_accu.hh>
```

Inherits `mln::fun::unary_param< from_accu< A >, A * >`.

### 10.140.1 Detailed Description

```
template<typename A> struct mln::fun::from_accu< A >
```

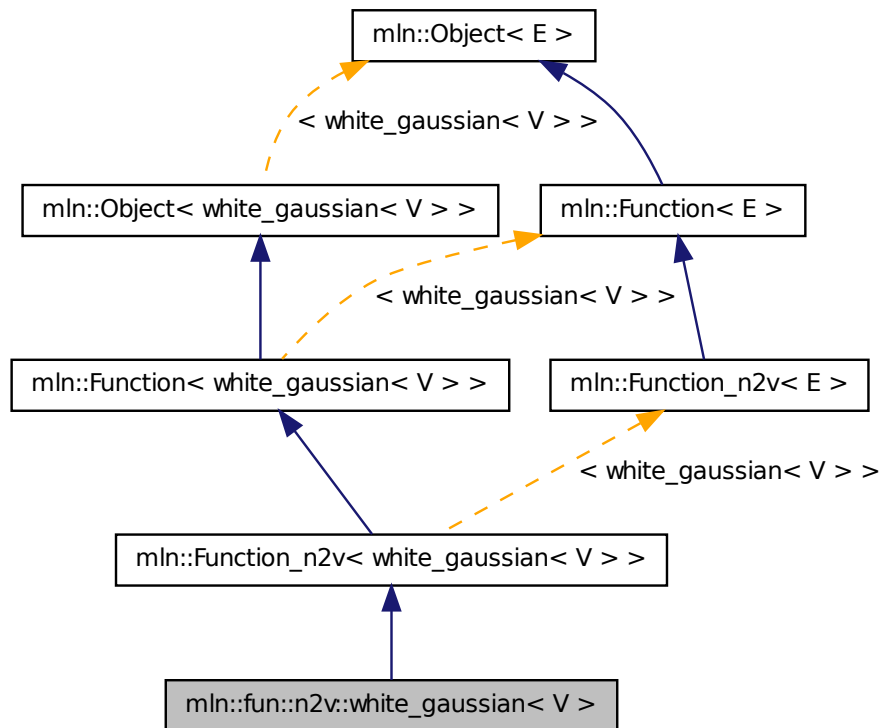
Wrap an accumulator into a function.

## 10.141 `mln::fun::n2v::white_gaussian< V >` Struct Template Reference

Generate a White Gaussian Noise.

```
#include <white_gaussian.hh>
```

Inheritance diagram for mln::fun::n2v::white\_gaussian< V >:



### 10.141.1 Detailed Description

```
template<typename V> struct mln::fun::n2v::white_gaussian< V >
```

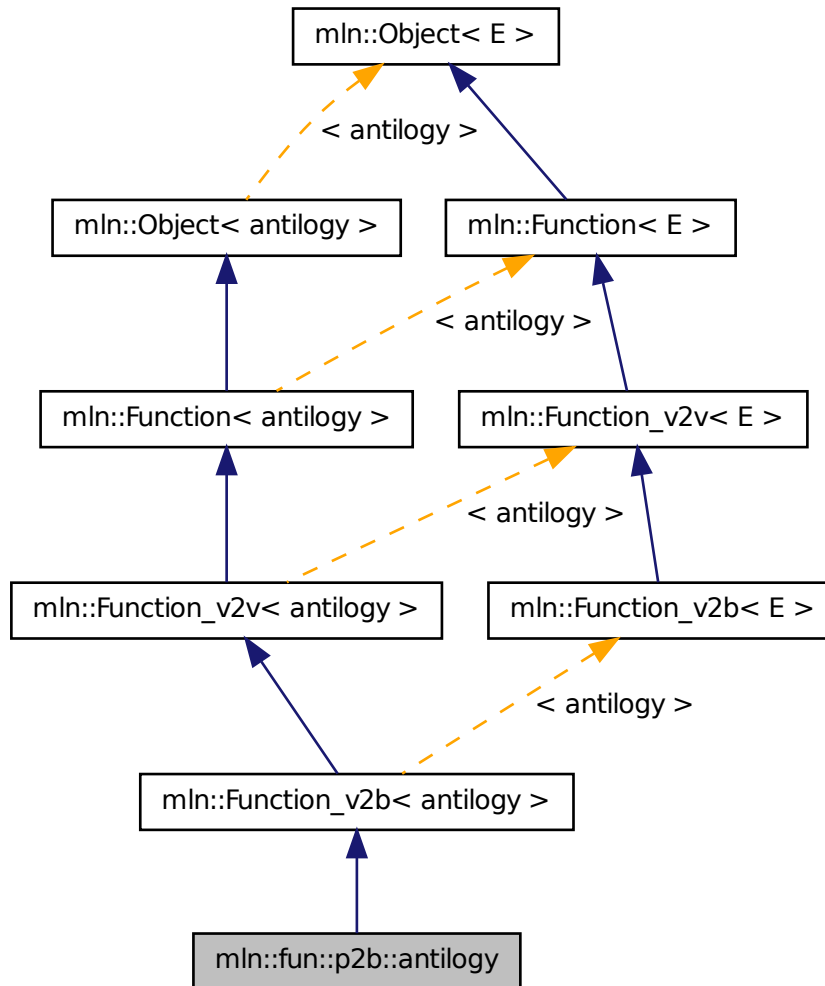
Generate a White Gaussian Noise. Reference: <http://www.dspguru.com/dsp/howtos/how-to-generate-white-gaussian-noise>

## 10.142 mln::fun::p2b::antilogy Struct Reference

A `p2b` function always returning `false`.

```
#include <antilogy.hh>
```

Inheritance diagram for `mln::fun::p2b::antilogy`:



### 10.142.1 Detailed Description

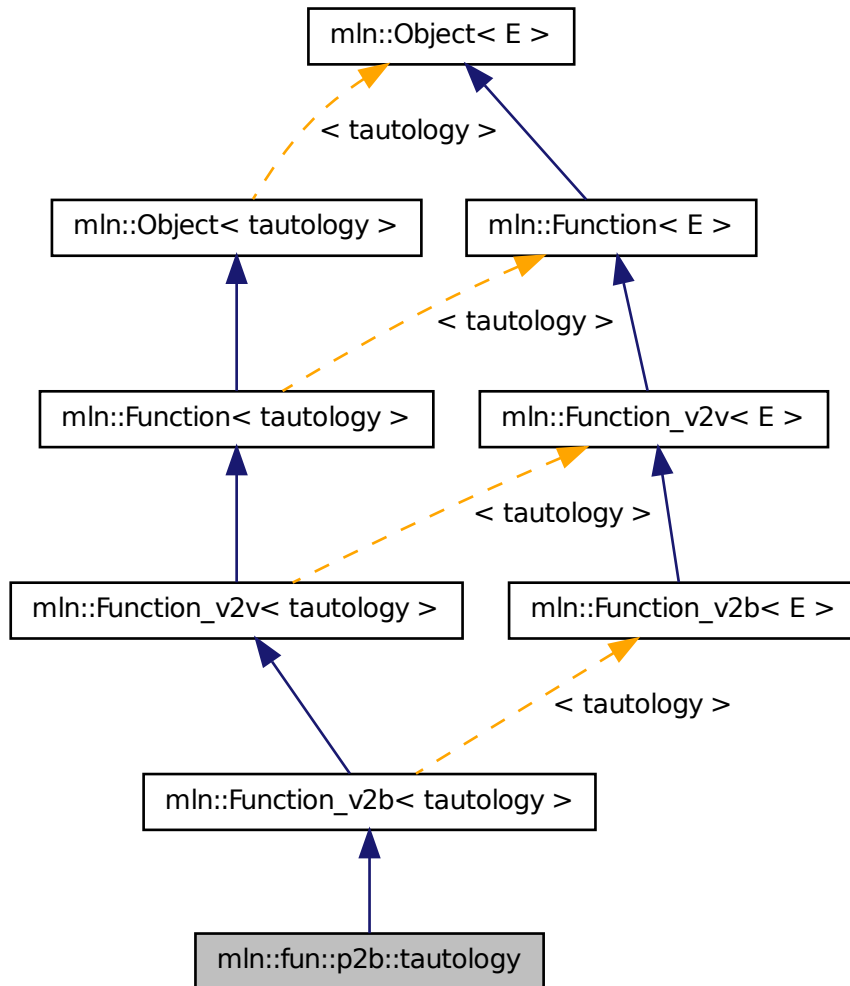
A `p2b` function always returning `false`. A simpler name would be `false`, but this is not a valid C++ identifier, as `false` is a keyword of the language.

## 10.143 `mln::fun::p2b::tautology` Struct Reference

A `p2b` function always returning `true`.

```
#include <tautology.hh>
```

Inheritance diagram for mln::fun::p2b::tautology:



### 10.143.1 Detailed Description

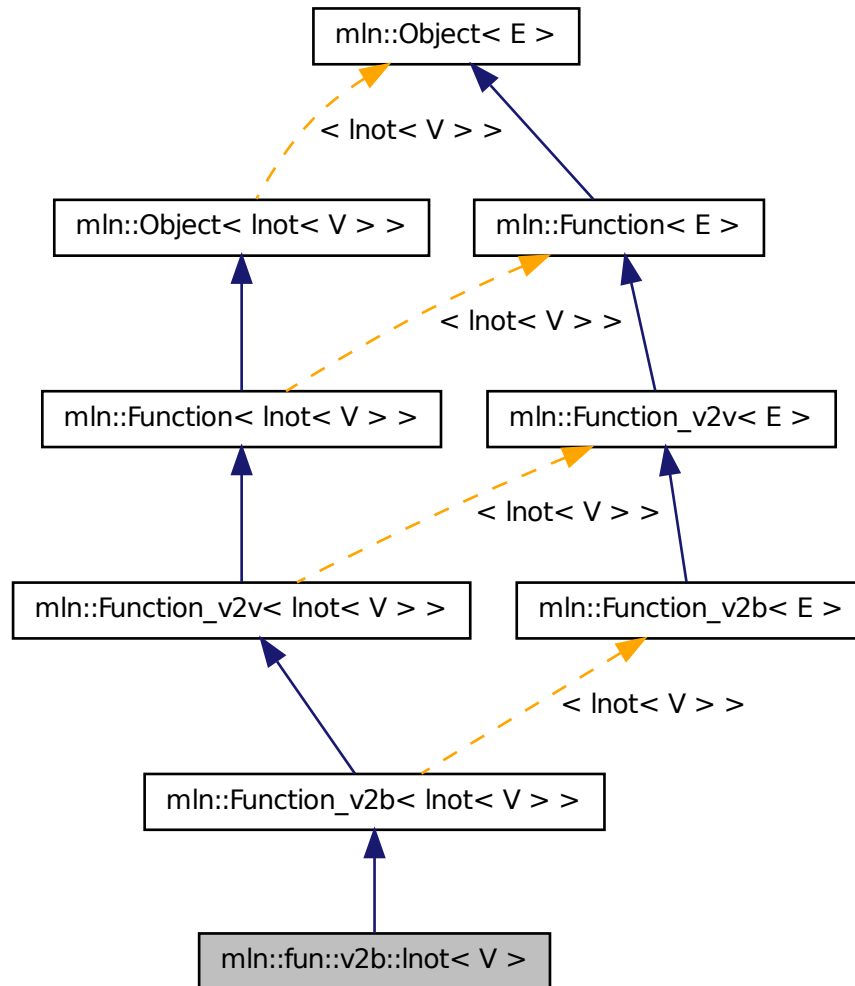
A `p2b` function always returning `true`. A simpler name would be `true`, but this is not a valid C++ identifier, as `true` is a keyword of the language.

## 10.144 mln::fun::v2b::lnot< V > Struct Template Reference

Functor computing logical-not on a value.

```
#include <lnot.hh>
```

Inheritance diagram for `mln::fun::v2b::lnot< V >`:



### 10.144.1 Detailed Description

```
template<typename V> struct mln::fun::v2b::lnot< V >
```

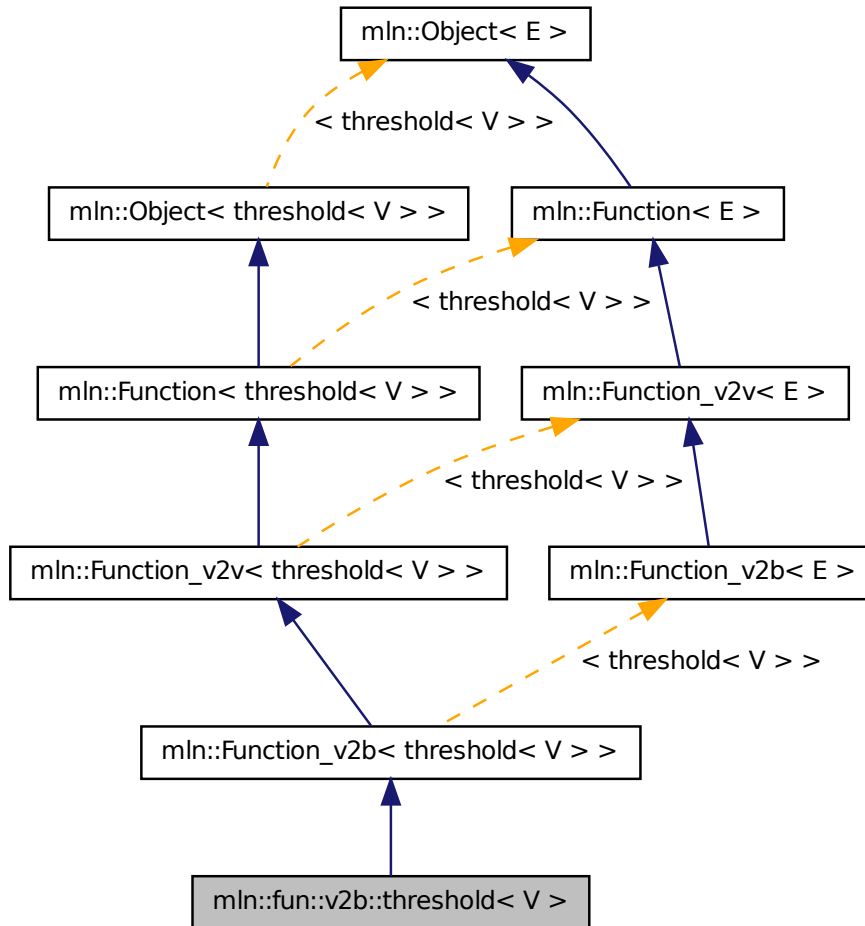
Functor computing logical-not on a value.

### 10.145 `mln::fun::v2b::threshold< V >` Struct Template Reference

Threshold function.

```
#include <threshold.hh>
```

Inheritance diagram for mln::fun::v2b::threshold< V >:



### 10.145.1 Detailed Description

```
template<typename V> struct mln::fun::v2b::threshold< V >
```

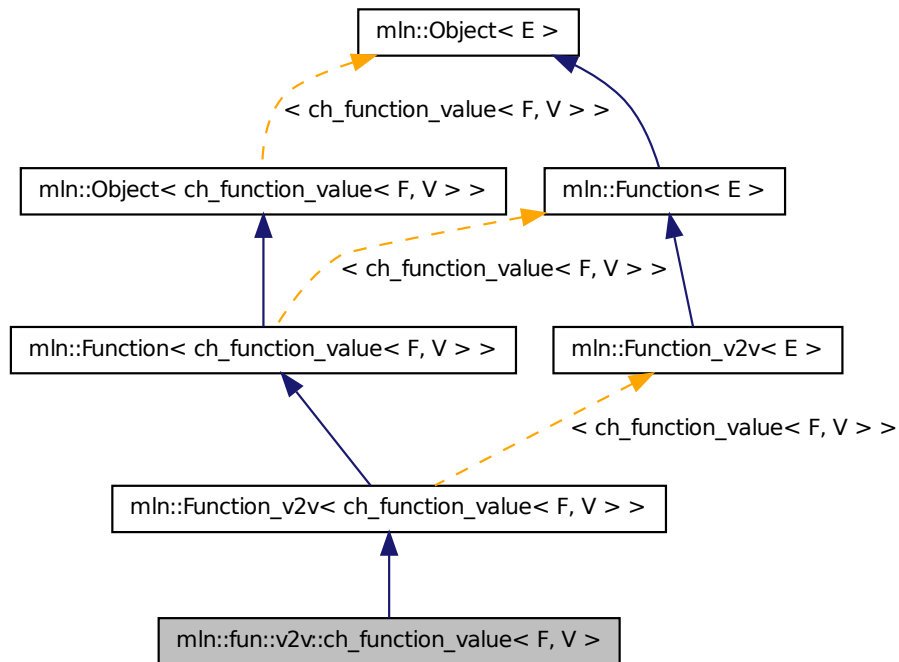
Threshold function.  $f(v) = (v \geq \text{threshold})$ .

## 10.146 mln::fun::v2v::ch\_function\_value< F, V > Class Template Reference

Wrap a function `v2v` and convert its result to another type.

```
#include <ch_function_value.hh>
```

Inheritance diagram for `mln::fun::v2v::ch_function_value< F, V >`:



### 10.146.1 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.

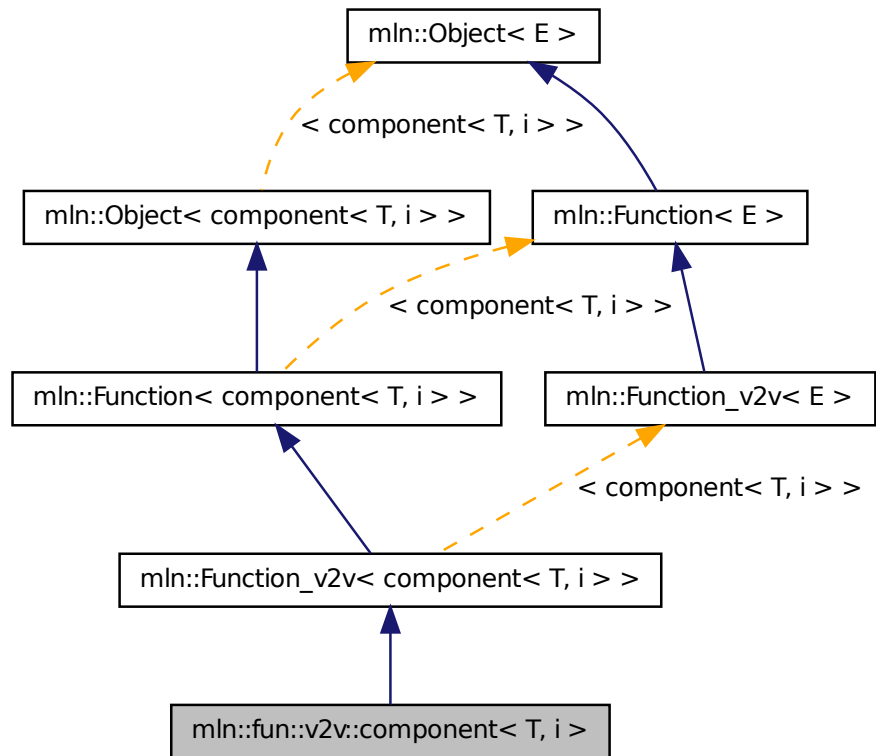
### 10.147 `mln::fun::v2v::component< T, i >` Struct Template Reference

Functor that accesses the `i`-th component of a value.

```
#include <component.hh>
```



Inheritance diagram for mln::fun::v2v::component< T, i >:



### 10.147.1 Detailed Description

```
template<typename T, unsigned i> struct mln::fun::v2v::component< T, i >
```

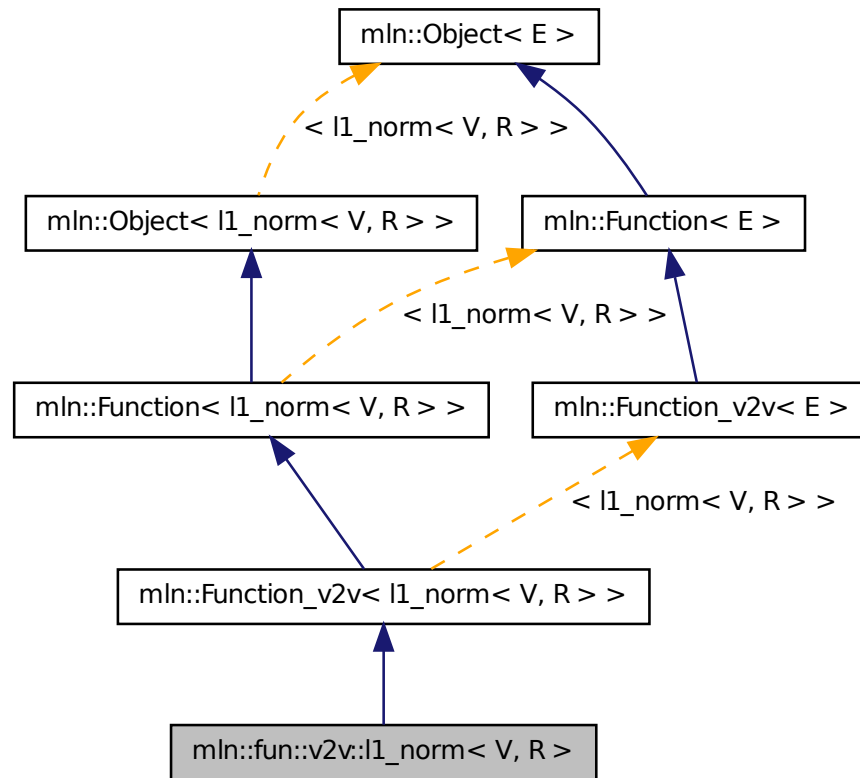
Functor that accesses the *i*-th component of a value.

## 10.148 mln::fun::v2v::l1\_norm< V, R > Struct Template Reference

L1-norm.

```
#include <norm.hh>
```

Inheritance diagram for `mln::fun::v2v::l1_norm< V, R >`:



### 10.148.1 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

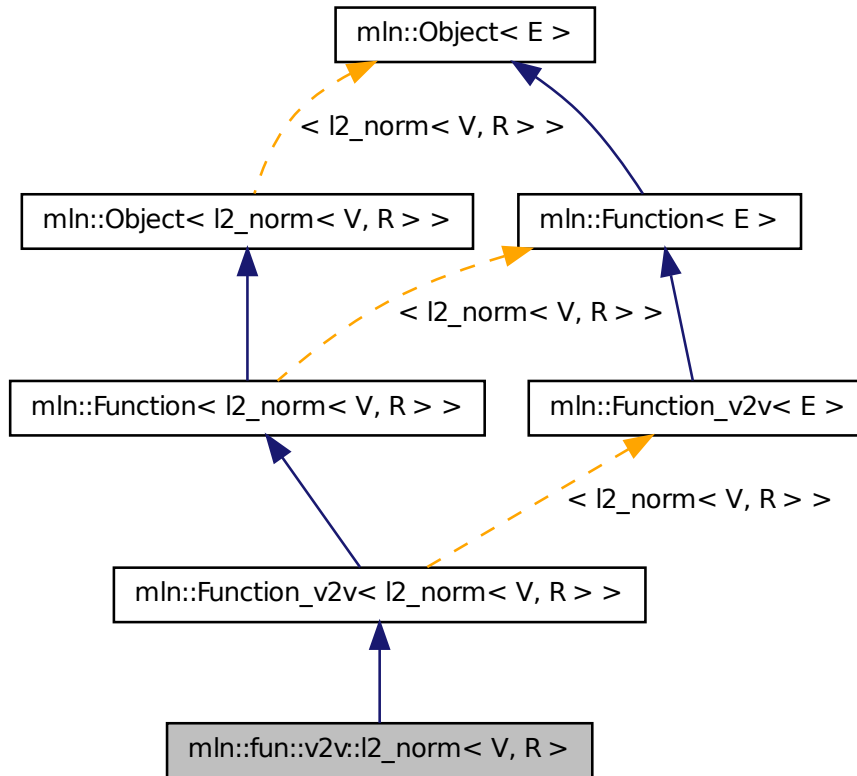
[mln::norm::l1](#).

## 10.149 mln::fun::v2v::l2\_norm< V, R > Struct Template Reference

L2-norm.

```
#include <norm.hh>
```

Inheritance diagram for mln::fun::v2v::l2\_norm< V, R >:



### 10.149.1 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

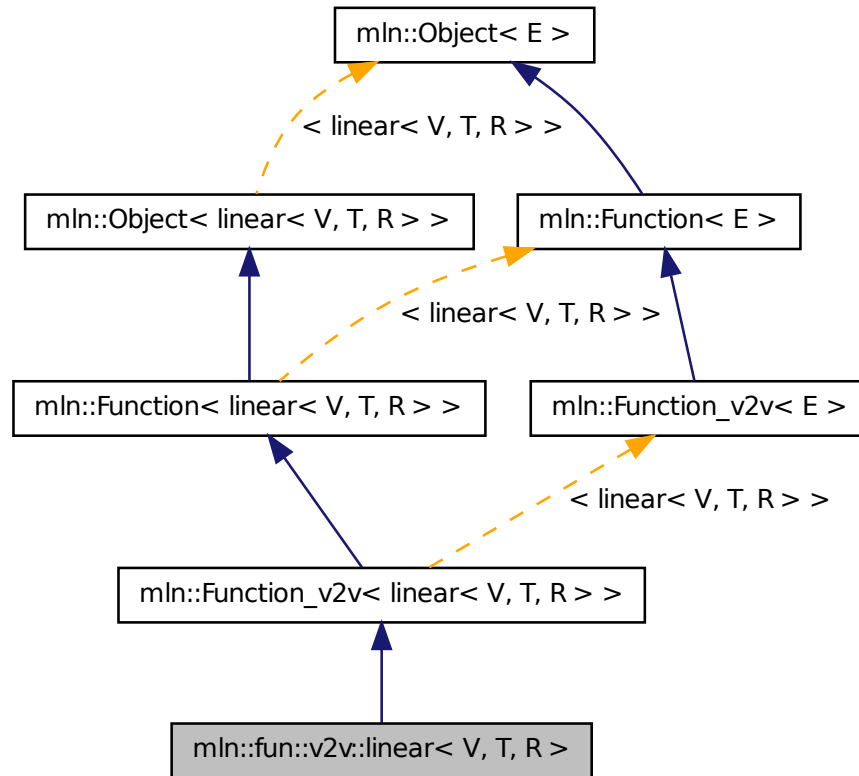
mln::norm::l2.

## 10.150 mln::fun::v2v::linear< V, T, R > Struct Template Reference

Linear function.  $f(v) = a * v + b$ . V is the type of input values; T is the type used to compute the result; R is the result type.

```
#include <linear.hh>
```

Inheritance diagram for `mln::fun::v2v::linear< V, T, R >`:



### 10.150.1 Detailed Description

```
template<typename V, typename T = V, typename R = T> struct mln::fun::v2v::linear< V, T, R >
```

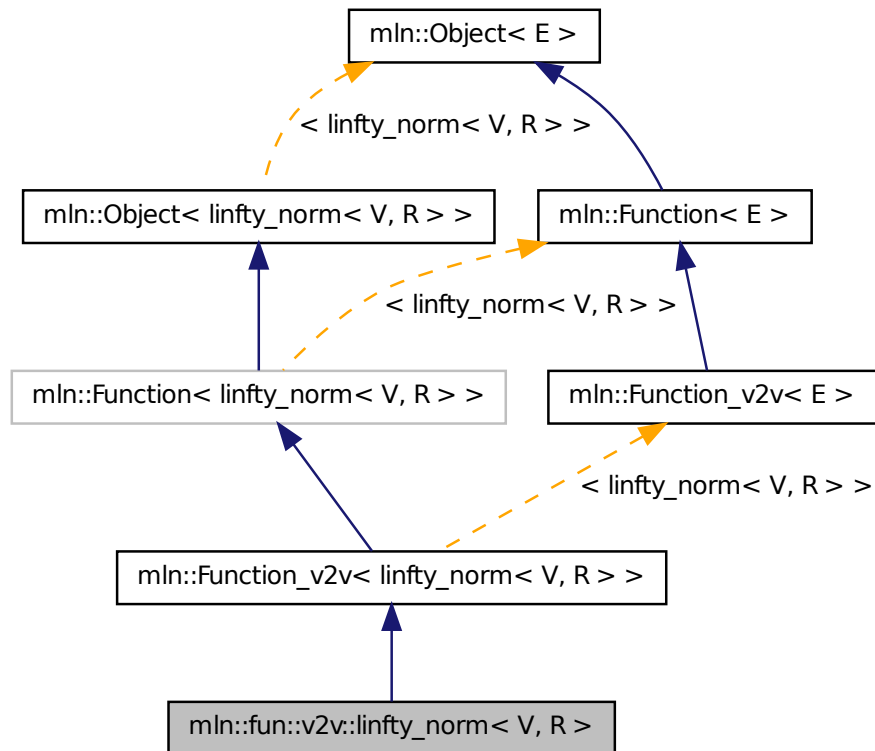
Linear function.  $f(v) = a * v + b$ . `V` is the type of input values; `T` is the type used to compute the result; `R` is the result type. By default, `T` is `V` and `R` is `T`.

## 10.151 `mln::fun::v2v::lifty_norm< V, R >` Struct Template Reference

L-infty norm.

```
#include <norm.hh>
```

Inheritance diagram for mln::fun::v2v::linfty\_norm< V, R >:



### 10.151.1 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

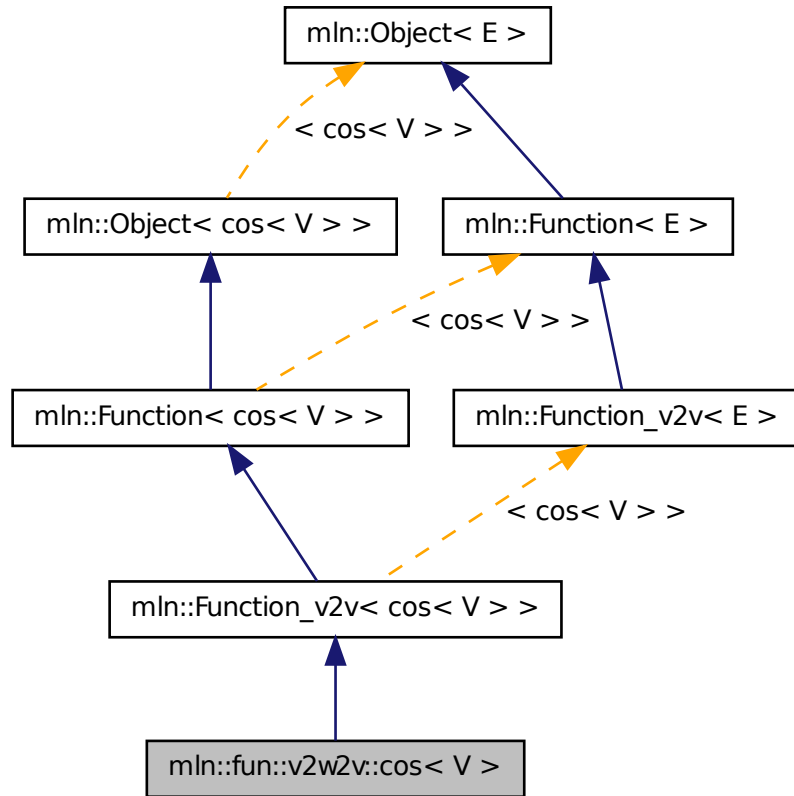
[mln::norm::linfty](#).

## 10.152 mln::fun::v2w2v::cos< V > Struct Template Reference

Cosinus bijective functor.

```
#include <cos.hh>
```

Inheritance diagram for `mln::fun::v2w2v::cos< V >`:



### 10.152.1 Detailed Description

```
template<typename V> struct mln::fun::v2w2v::cos< V >
```

Cosinus bijective functor. `V` is the type of input values and the result type.

See also

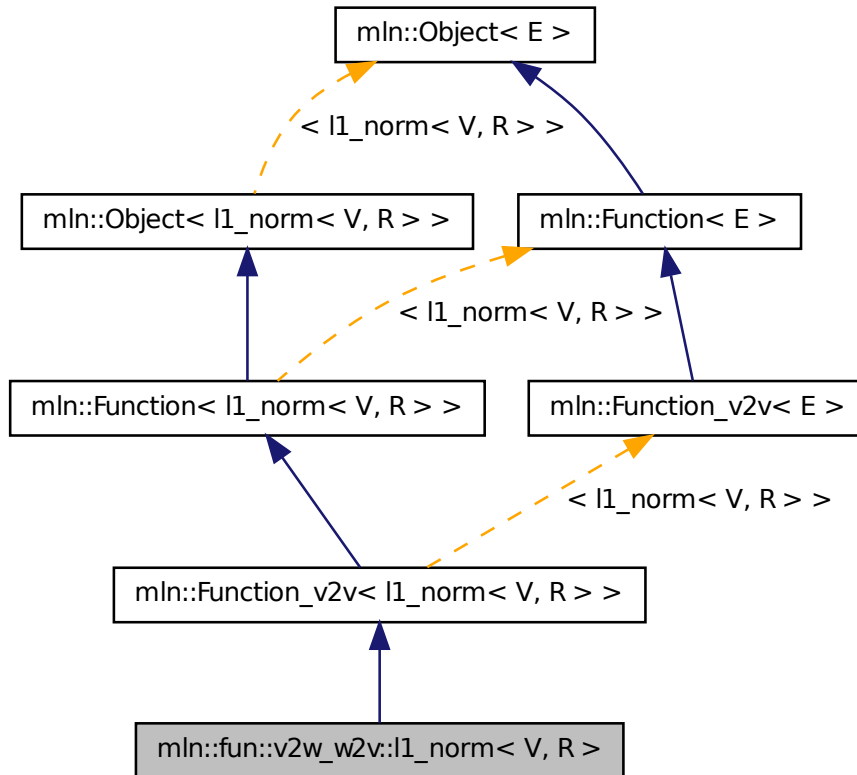
`mln::math::cos`.

### 10.153 `mln::fun::v2w_w2v::l1_norm< V, R >` Struct Template Reference

L1-norm.

```
#include <norm.hh>
```

Inheritance diagram for `mln::fun::v2w_w2v::l1_norm< V, R >`:



### 10.153.1 Detailed Description

```
template<typename V, typename R> struct mln::fun::v2w_w2v::l1_norm< V, R >
```

L1-norm. `V` is the type of input values; `R` is the result type.

See also

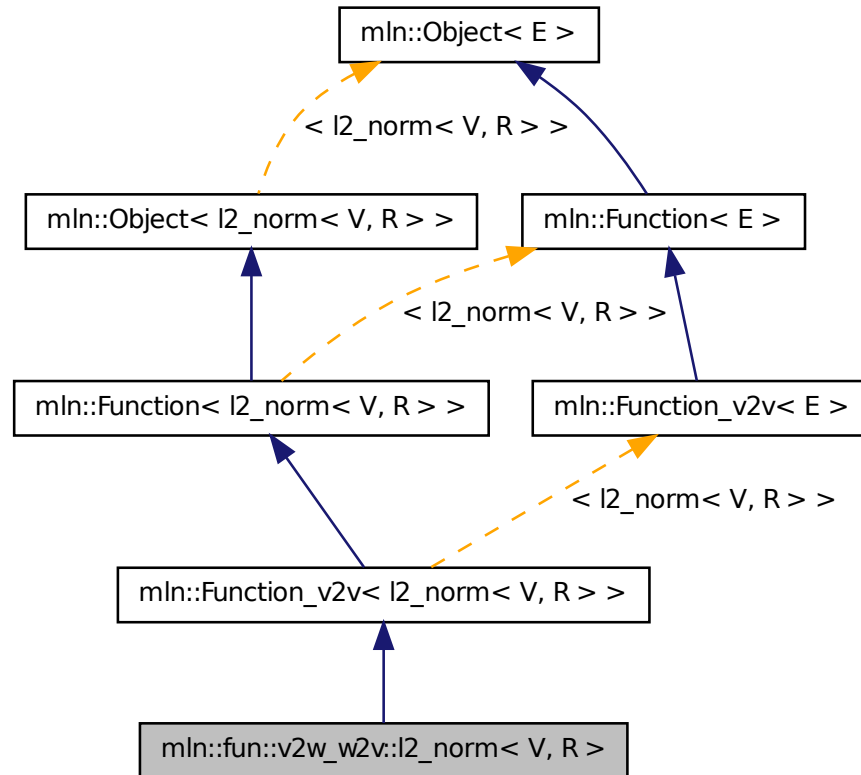
[mln::norm::l1](#).

## 10.154 `mln::fun::v2w_w2v::l2_norm< V, R >` Struct Template Reference

L2-norm.

```
#include <norm.hh>
```

Inheritance diagram for `mln::fun::v2w_w2v::l2_norm< V, R >`:



### 10.154.1 Detailed Description

```
template<typename V, typename R> struct mln::fun::v2w_w2v::l2_norm< V, R >
```

L2-norm. `V` is the type of input values; `R` is the result type.

See also

`mln::norm::l2`.

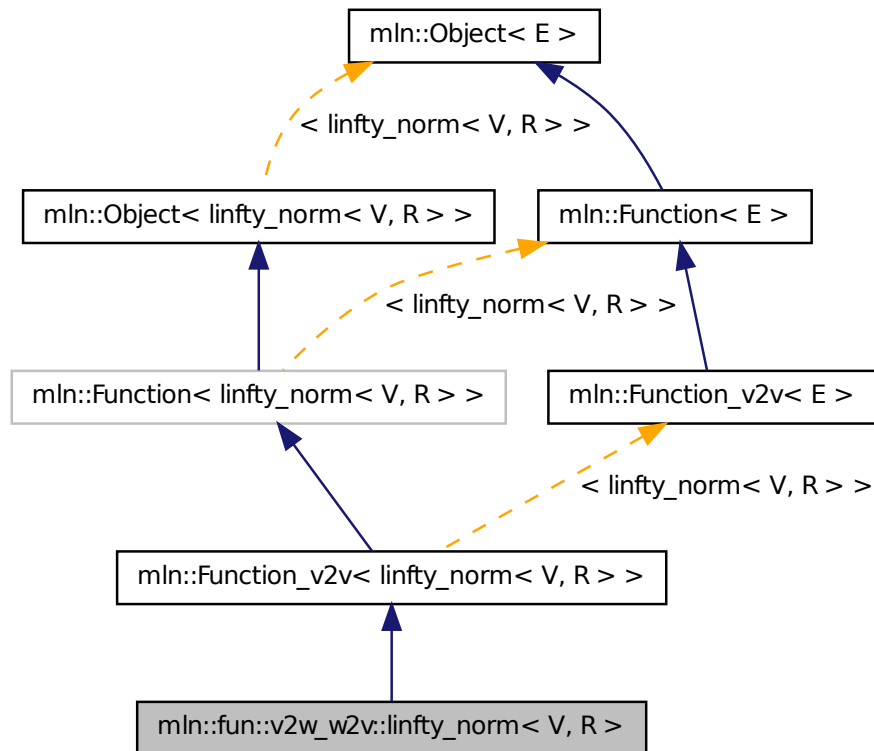
## 10.155 `mln::fun::v2w_w2v::lifty_norm< V, R >` Struct Template Reference

L-infty norm.

```
#include <norm.hh>
```



Inheritance diagram for mln::fun::v2w\_w2v::linfty\_norm< V, R >:



### 10.155.1 Detailed Description

```
template<typename V, typename R> struct mln::fun::v2w_w2v::linfty_norm< V, R >
```

L-infty norm. V is the type of input values; R is the result type.

See also

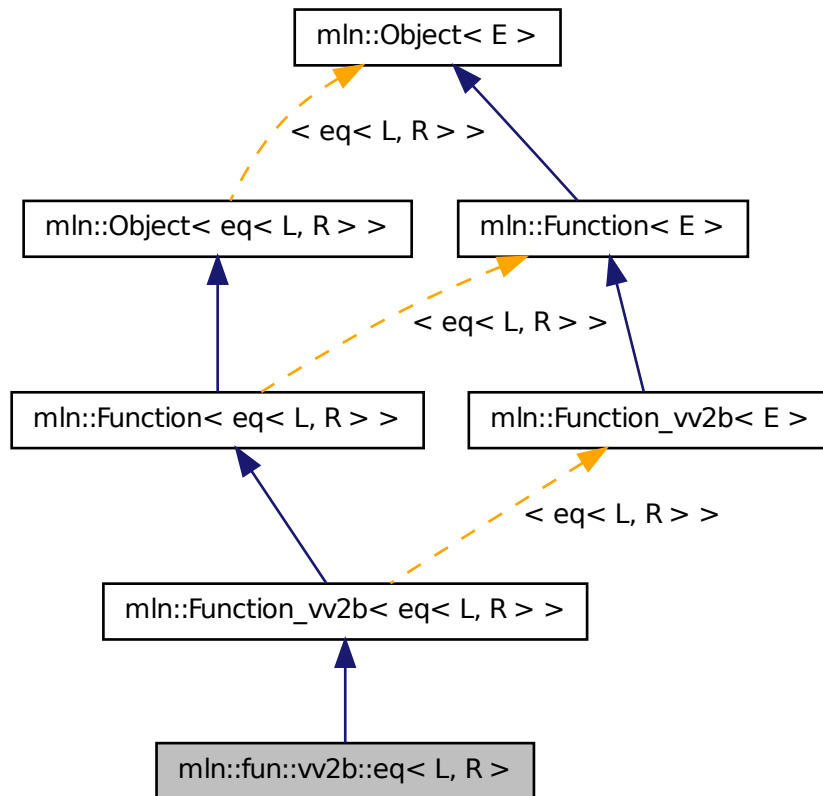
[mln::norm::linfty](#).

## 10.156 mln::fun::vv2b::eq< L, R > Struct Template Reference

Functor computing equal between two values.

```
#include <eq.hh>
```

Inheritance diagram for `mln::fun::vv2b::eq< L, R >`:



### 10.156.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2b::eq< L, R >
```

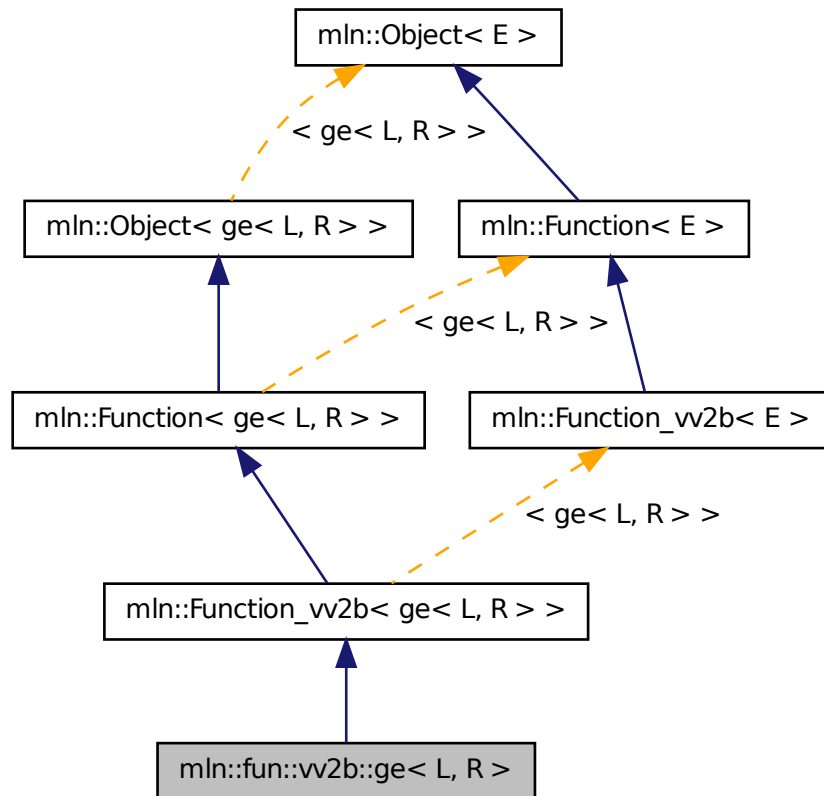
Functor computing equal between two values.

## 10.157 `mln::fun::vv2b::ge< L, R >` Struct Template Reference

Functor computing "greater or equal than" between two values.

```
#include <ge.hh>
```

Inheritance diagram for mln::fun::vv2b::ge< L, R >:



### 10.157.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2b::ge< L, R >
```

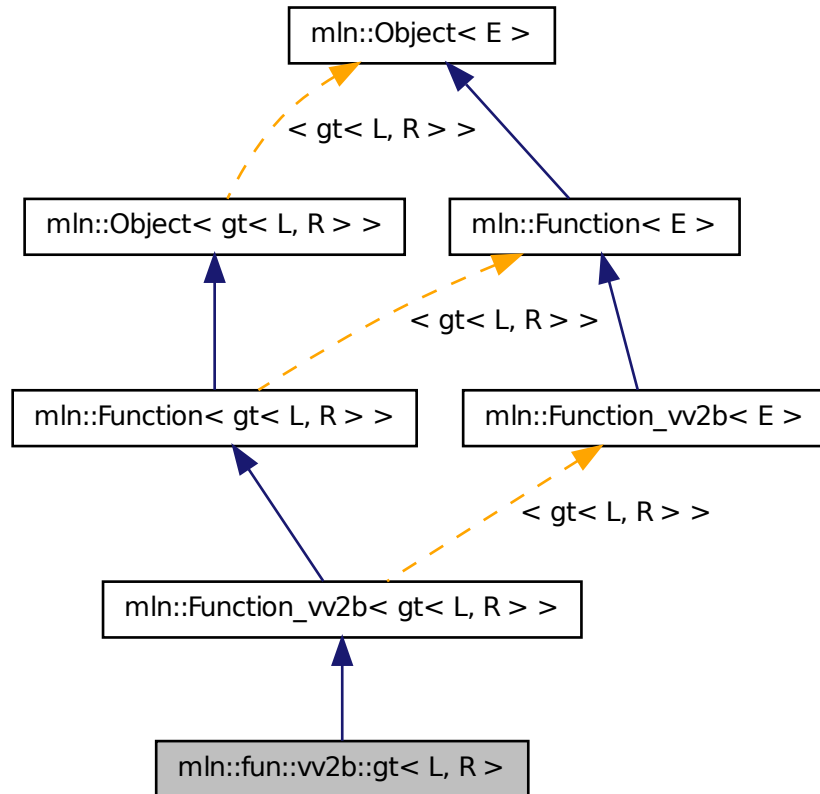
Functor computing "greater or equal than" between two values.

## 10.158 mln::fun::vv2b::gt< L, R > Struct Template Reference

Functor computing "greater than" between two values.

```
#include <gt.hh>
```

Inheritance diagram for `mln::fun::vv2b::gt< L, R >`:



### 10.158.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2b::gt< L, R >
```

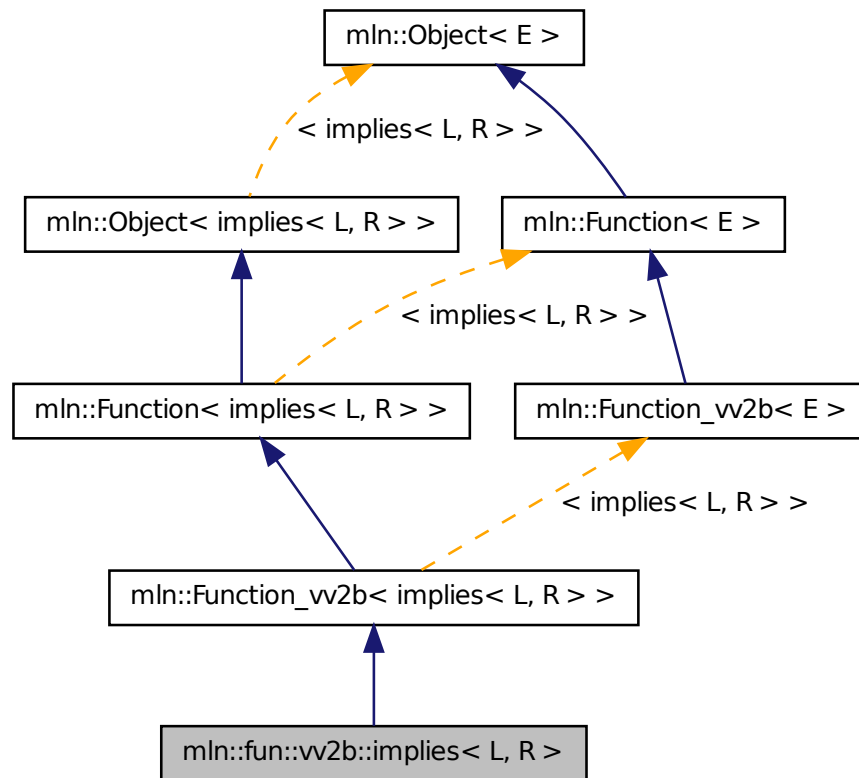
Functor computing "greater than" between two values.

## 10.159 mln::fun::vv2b::implies< L, R > Struct Template Reference

Functor computing logical-implies between two values.

```
#include <implies.hh>
```

Inheritance diagram for mln::fun::vv2b::implies< L, R >:



### 10.159.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2b::implies< L, R >
```

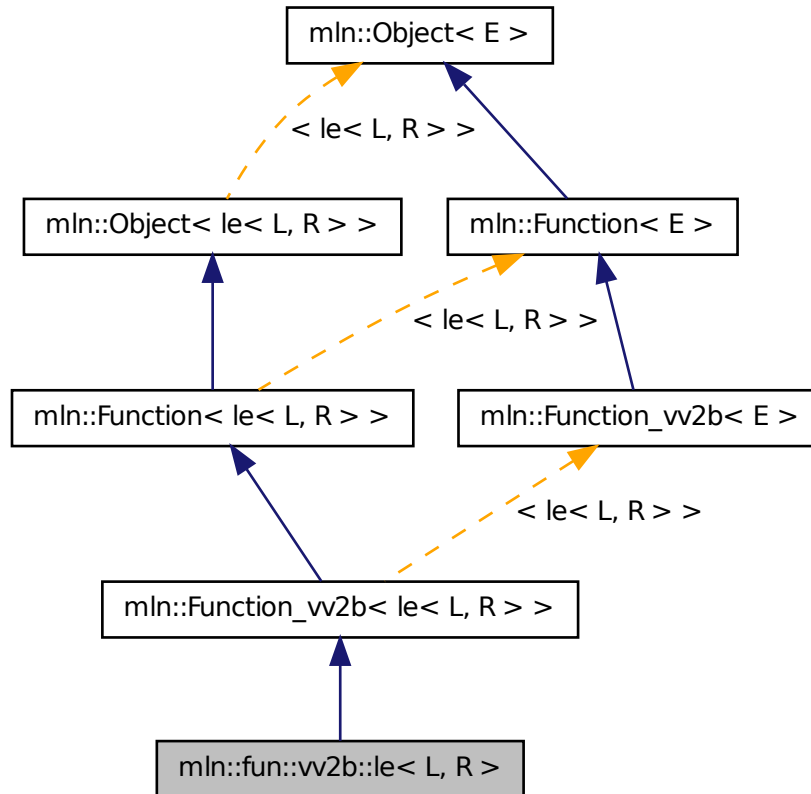
Functor computing logical-implies between two values.

## 10.160 mln::fun::vv2b::le< L, R > Struct Template Reference

Functor computing "lower or equal than" between two values.

```
#include <le.hh>
```

Inheritance diagram for `mln::fun::vv2b::le< L, R >`:



### 10.160.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2b::le< L, R >
```

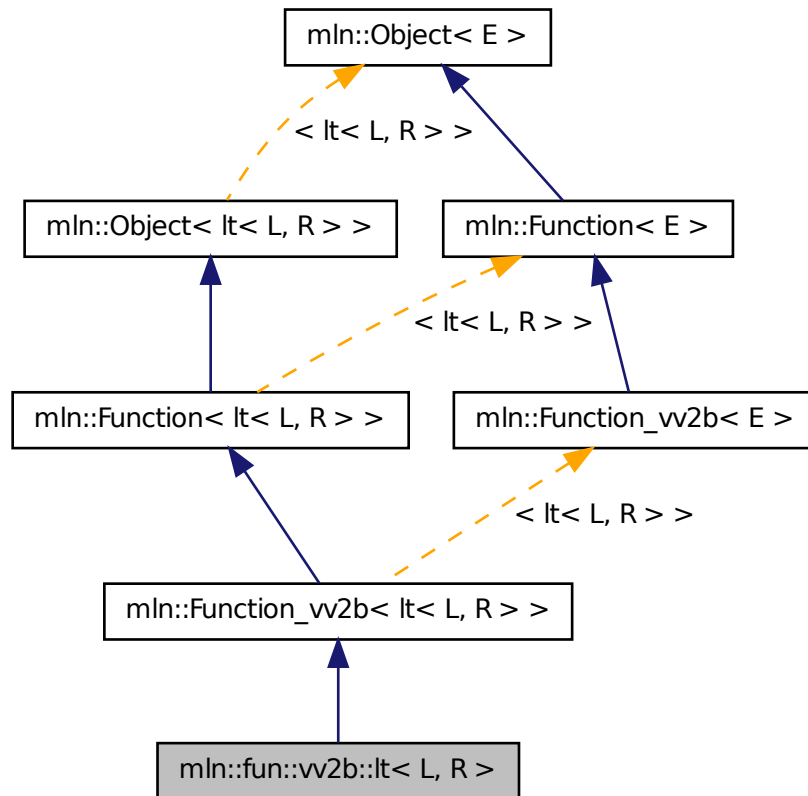
Functor computing "lower or equal than" between two values.

### 10.161 `mln::fun::vv2b::lt< L, R >` Struct Template Reference

Functor computing "lower than" between two values.

```
#include <lt.hh>
```

Inheritance diagram for mln::fun::vv2b::lt< L, R >:



### 10.161.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2b::lt< L, R >
```

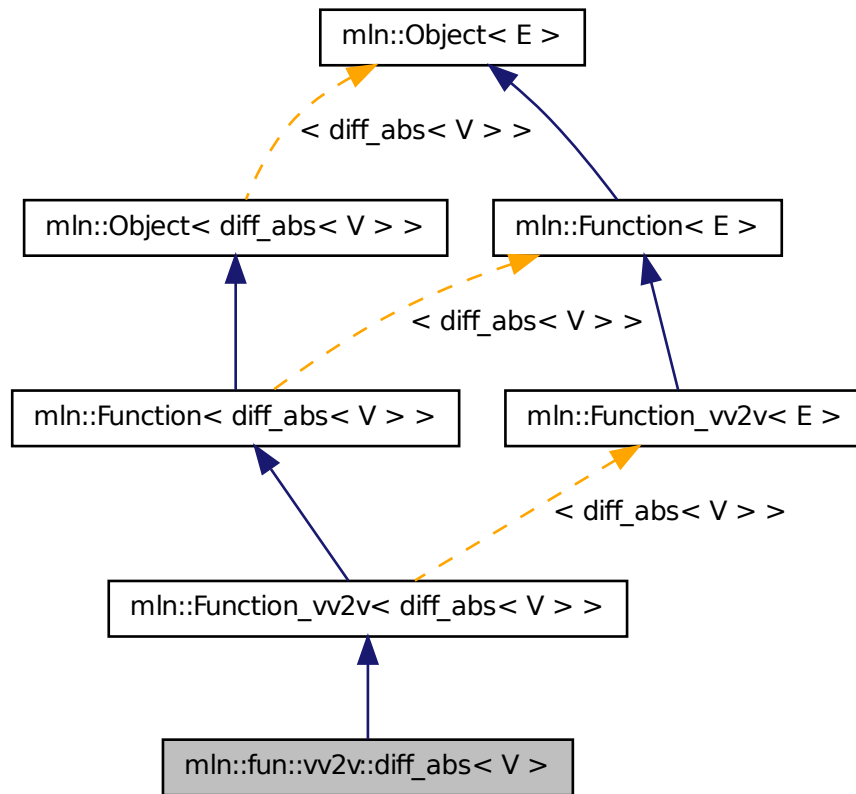
Functor computing "lower than" between two values.

## 10.162 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 >`:



### 10.162.1 Detailed Description

```
template<typename V> struct mln::fun::vv2v::diff_abs< V >
```

A functor computing the `diff_abs`imum of two values.

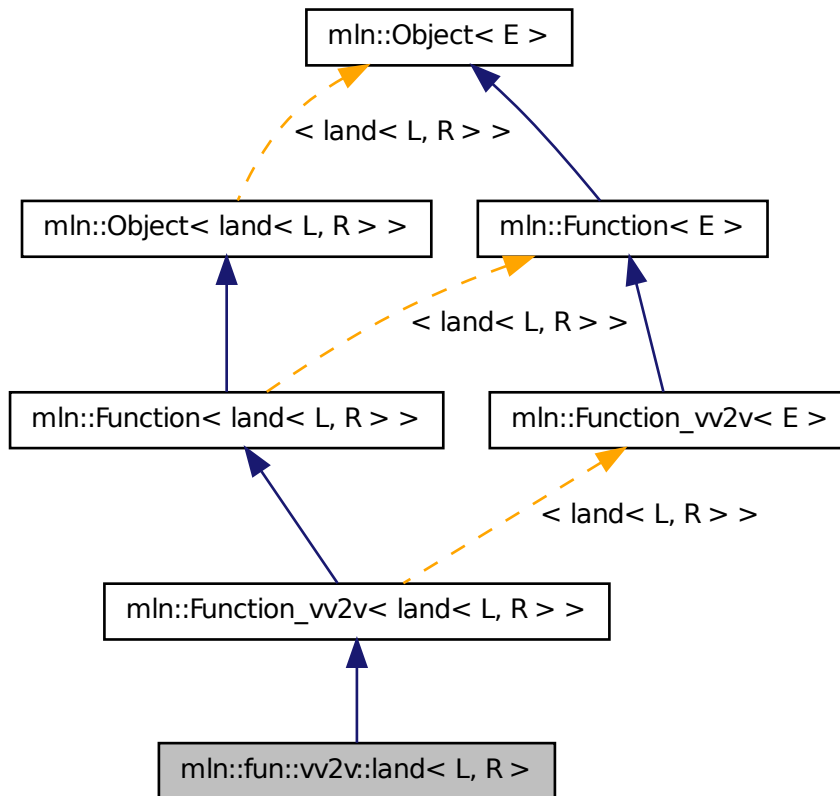
## 10.163 `mln::fun::vv2v::land< L, R >` Struct Template Reference

Functor computing logical-and between two values.

```
#include <land.hh>
```



Inheritance diagram for mln::fun::vv2v::land< L, R >:



### 10.163.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2v::land< L, R >
```

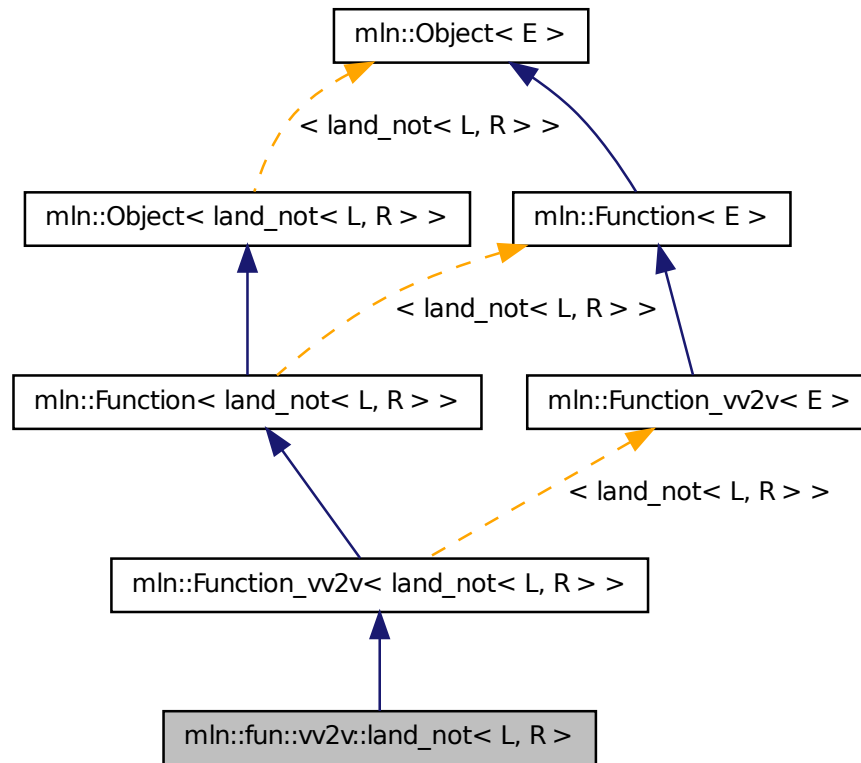
Functor computing logical-and between two values.

## 10.164 mln::fun::vv2v::land\_not< L, R > Struct Template Reference

Functor computing logical and-not between two values.

```
#include <land_not.hh>
```

Inheritance diagram for `mln::fun::vv2v::land_not< L, R >`:



### 10.164.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2v::land_not< L, R >
```

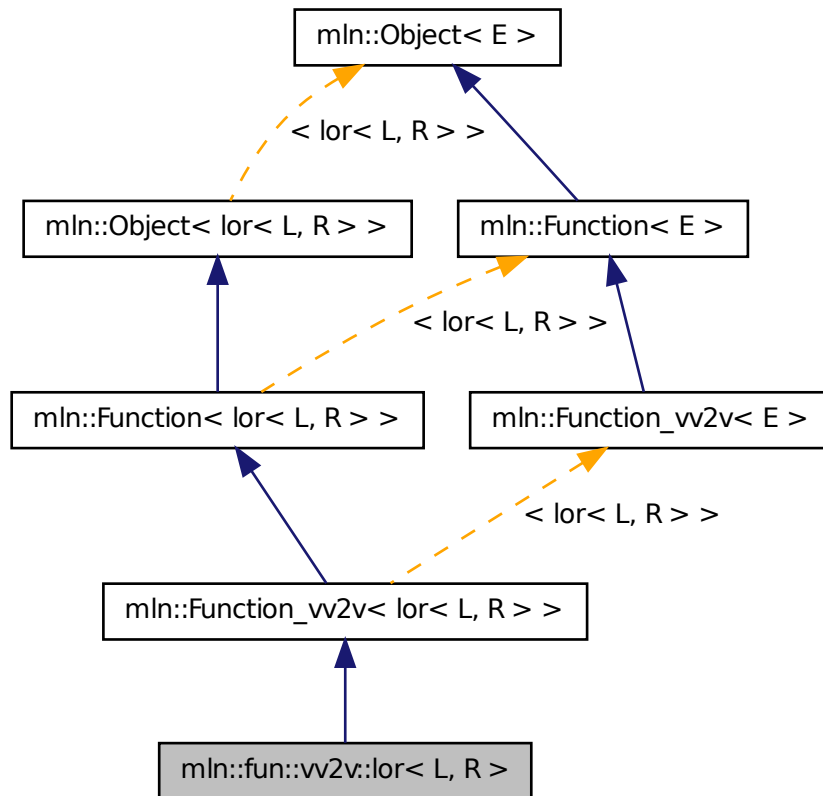
Functor computing logical and-not between two values.

## 10.165 `mln::fun::vv2v::lor< L, R >` Struct Template Reference

Functor computing logical-or between two values.

```
#include <lor.hh>
```

Inheritance diagram for mln::fun::vv2v::lor< L, R >:



### 10.165.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2v::lor< L, R >
```

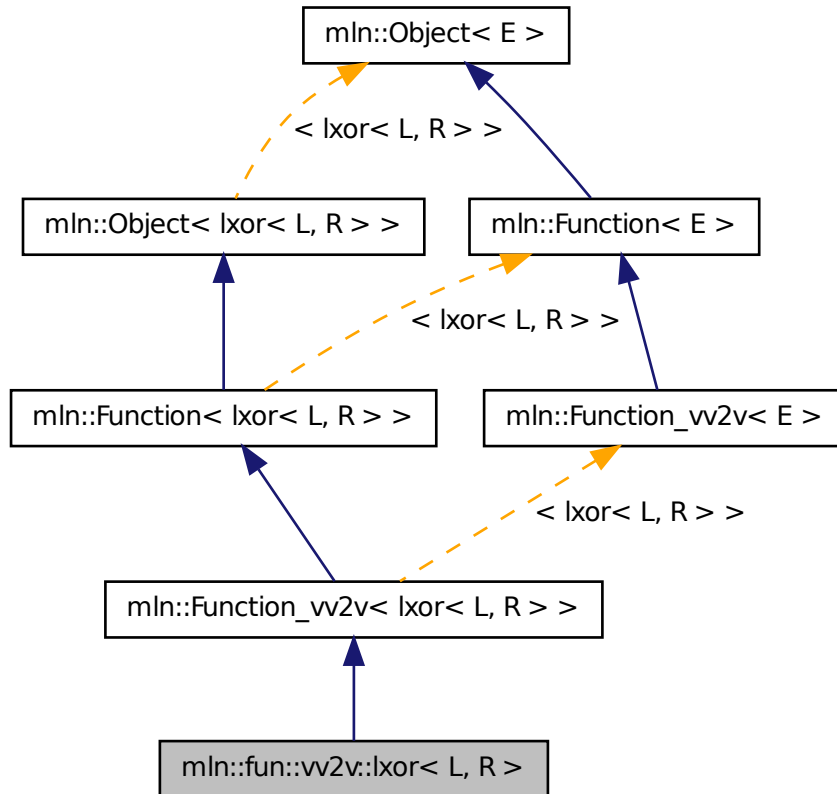
Functor computing logical-or between two values.

## 10.166 mln::fun::vv2v::lxor< L, R > Struct Template Reference

Functor computing logical-xor between two values.

```
#include <lxor.hh>
```

Inheritance diagram for `mln::fun::vv2v::lxor< L, R >`:



### 10.166.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2v::lxor< L, R >
```

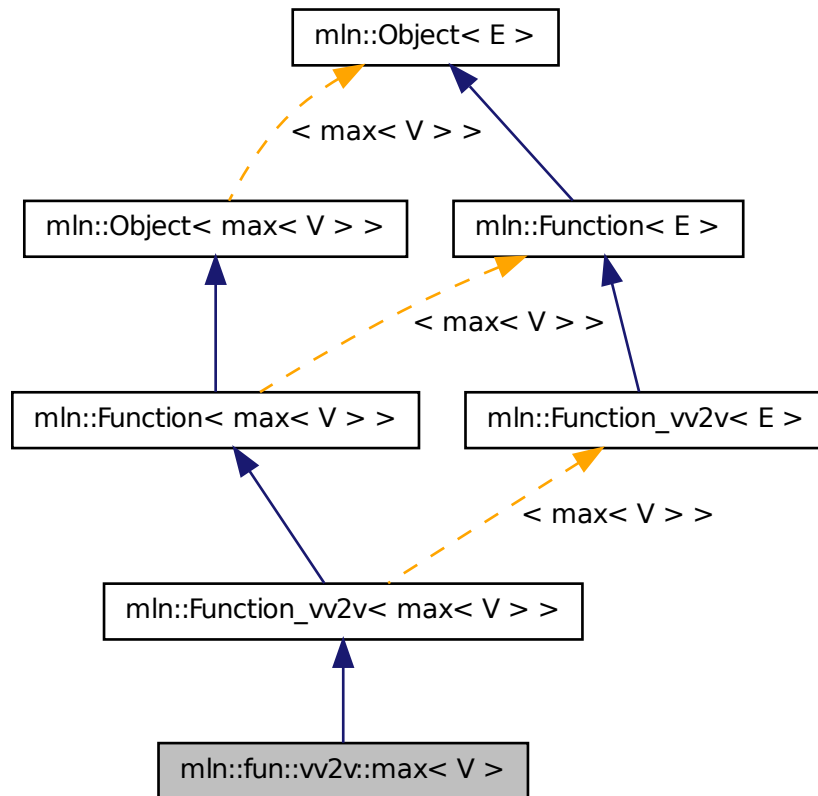
Functor computing logical-xor between two values.

## 10.167 `mln::fun::vv2v::max< V >` Struct Template Reference

A functor computing the maximum of two values.

```
#include <max.hh>
```

Inheritance diagram for mln::fun::vv2v::max< V >:



### 10.167.1 Detailed Description

```
template<typename V> struct mln::fun::vv2v::max< V >
```

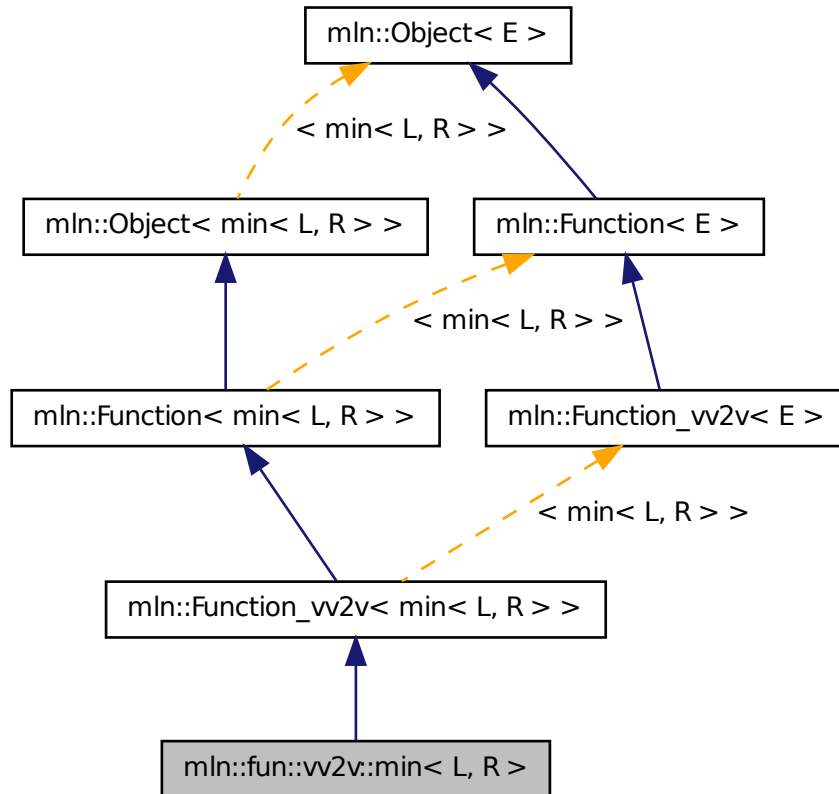
A functor computing the maximum of two values.

## 10.168 mln::fun::vv2v::min< L, R > Struct Template Reference

A functor computing the minimum of two values.

```
#include <min.hh>
```

Inheritance diagram for `mln::fun::vv2v::min< L, R >`:



### 10.168.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2v::min< L, R >
```

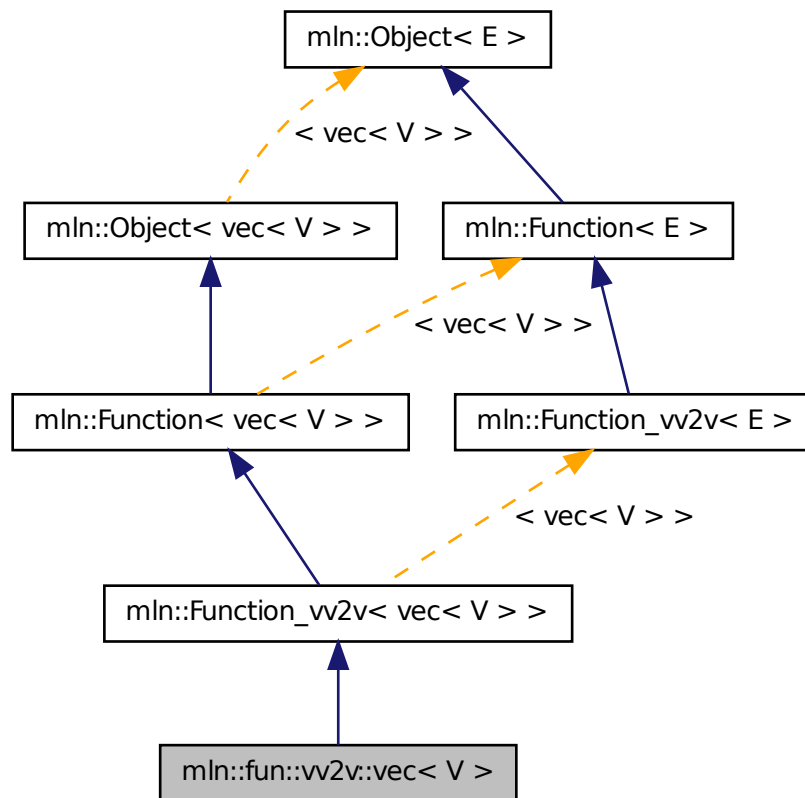
A functor computing the minimum of two values.

## 10.169 mln::fun::vv2v::vec< V > Struct Template Reference

A functor computing the vecimum of two values.

```
#include <vec.hh>
```

Inheritance diagram for mln::fun::vv2v::vec< V >:



### 10.169.1 Detailed Description

```
template<typename V> struct mln::fun::vv2v::vec< V >
```

A functor computing the vecimum of two values.

## 10.170 mln::fun::x2p::closest\_point< P > Struct Template Reference

FIXME: doxygen + concept checking.

```
#include <closest_point.hh>
```

### 10.170.1 Detailed Description

```
template<typename P> struct mln::fun::x2p::closest_point< P >
```

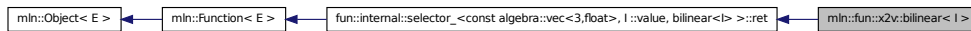
FIXME: doxygen + concept checking.

## 10.171 mln::fun::x2v::bilinear< I > Struct Template Reference

Represent a bilinear interolation of values from an underlying image.

```
#include <bilinear.hh>
```

Inheritance diagram for mln::fun::x2v::bilinear< I >:



### Public Member Functions

- template<typename T >  
I::value [operator\(\)](#) (const algebra::vec< 3, T > &v) const  
*Bilinear filtering on 3d images. Work on slices.*
- template<typename T >  
I::value [operator\(\)](#) (const algebra::vec< 2, T > &v) const  
*Bilinear filtering on 2d images.*

### 10.171.1 Detailed Description

```
template<typename I> struct mln::fun::x2v::bilinear< I >
```

Represent a bilinear interolation of values from an underlying image.

### 10.171.2 Member Function Documentation

**10.171.2.1** template<typename I > template<typename T > I::value mln::fun::x2v::bilinear< I >::operator() ( const algebra::vec< 2, T > & v ) const

Bilinear filtering on 2d images.

**10.171.2.2** template<typename I > template<typename T > I::value mln::fun::x2v::bilinear< I >::operator() ( const algebra::vec< 3, T > & v ) const

Bilinear filtering on 3d images. Work on slices.

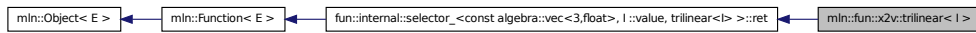


## 10.172 mln::fun::x2v::trilinear< I > Struct Template Reference

Represent a trilinear interolation of values from an underlying image.

```
#include <trilinear.hh>
```

Inheritance diagram for mln::fun::x2v::trilinear< I >:



### 10.172.1 Detailed Description

```
template<typename I> struct mln::fun::x2v::trilinear< I >
```

Represent a trilinear interolation of values from an underlying image.

## 10.173 mln::fun::x2x::composed< T2, T1 > Struct Template Reference

Represent a composition of two transformations.

```
#include <composed.hh>
```

### Public Member Functions

- [composed](#) ()  
*Constructor without argument.*
- [composed](#) (const T2 &f, const T1 &g)  
*Constructor with the two transformation to be composed.*

### 10.173.1 Detailed Description

```
template<typename T2, typename T1> struct mln::fun::x2x::composed< T2, T1 >
```

Represent a composition of two transformations.

### 10.173.2 Constructor & Destructor Documentation

**10.173.2.1** `template<typename T2, typename T1> mln::fun::x2x::composed< T2, T1 >::composed ( ) [inline]`

Constructor without argument.

### 10.173.2.2 `template<typename T2, typename T1> mln::fun::x2x::composed< T2, T1 >::composed ( const T2 & f, const T1 & g ) [inline]`

Constructor with the two transformation to be composed.

## 10.174 `mln::fun::x2x::linear< I >` Struct Template Reference

Represent a linear interpolation of values from an underlying image.

```
#include <linear.hh>
```

Inheritance diagram for `mln::fun::x2x::linear< I >`:



### Public Member Functions

- `linear` (const I & *ima*)

*Constructor with the underlying image.*

- `template<typename C > I::value operator() (const algebra::vec< 1, C > &v) const`

*Return the interpolated value in the underlying image at the given 'point' v.*

### Public Attributes

- const I & *ima*

*Underlying image.*

### 10.174.1 Detailed Description

```
template<typename I> struct mln::fun::x2x::linear< I >
```

Represent a linear interpolation of values from an underlying image.

### 10.174.2 Constructor & Destructor Documentation

#### 10.174.2.1 `template<typename I > mln::fun::x2x::linear< I >::linear ( const I & ima )`

Constructor with the underlying image.

### 10.174.3 Member Function Documentation

**10.174.3.1** `template<typename I > template<typename C > I::value mln::fun::x2x::linear< I >::operator() ( const algebra::vec< 1, C > & v ) const`

Return the interpolated value in the underlying image at the given 'point' v.

### 10.174.4 Member Data Documentation

**10.174.4.1** `template<typename I > const I& mln::fun::x2x::linear< I >::ima`

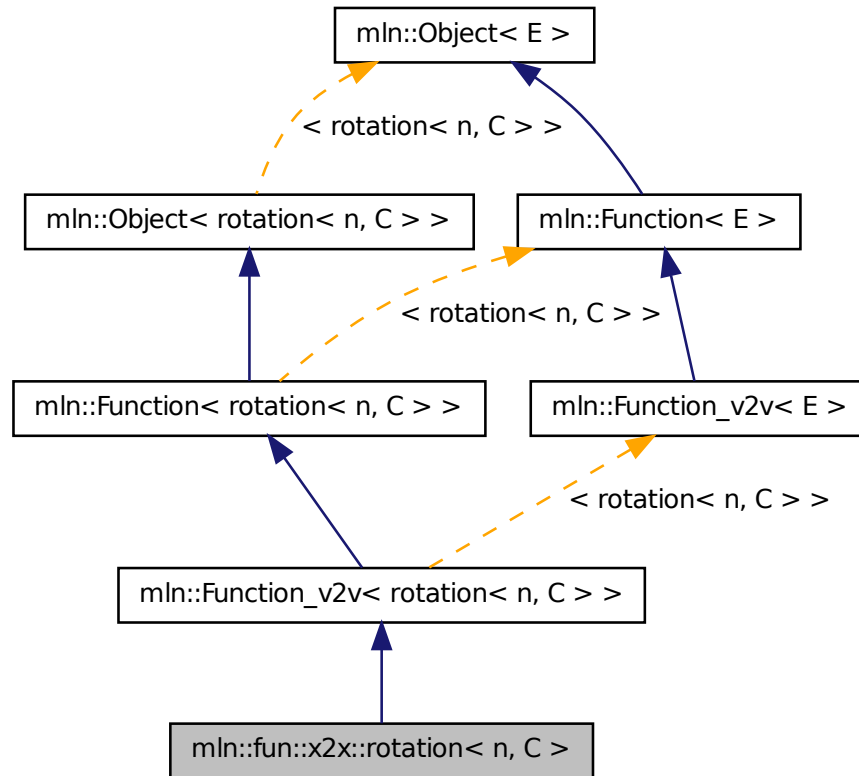
Underlying image.

## 10.175 mln::fun::x2x::rotation< n, C > Struct Template Reference

Represent a rotation function.

```
#include <rotation.hh>
```

Inheritance diagram for `mln::fun::x2x::rotation< n, C >`:



## Public Types

- typedef `C` `data_t`  
*Type of the underlying data stored in vectors and matrices.*
- typedef `rotation< n, C >` `invert`  
*Type of the inverse function.*

## Public Member Functions

- `invert inv () const`  
*Return the inverse function.*
- `algebra::vec< n, C > operator() (const algebra::vec< n, C > &v) const`  
*Perform the rotation of the given vector.*

- `rotation ()`  
*Constructor without argument.*
- `rotation (const algebra::quat &q)`  
*Constructor with quaternion.*
- `rotation (const algebra::h_mat< n, C > &m)`  
*Constructor with h\_mat.*
- `rotation (C alpha, const algebra::vec< n, C > &axis)`  
*Constructor with radian alpha and a facultative direction (rotation axis).*
- `void set_alpha (C alpha)`  
*Set a new grade alpha.*
- `void set_axis (const algebra::vec< n, C > &axis)`  
*Set a new rotation axis.*

### 10.175.1 Detailed Description

`template<unsigned n, typename C> struct mln::fun::x2x::rotation< n, C >`

Represent a rotation function.

### 10.175.2 Member Typedef Documentation

**10.175.2.1** `template<unsigned n, typename C > typedef C mln::fun::x2x::rotation< n, C >::data_t`

Type of the underlying data stored in vectors and matrices.

**10.175.2.2** `template<unsigned n, typename C > typedef rotation<n,C> mln::fun::x2x::rotation< n, C >::invert`

Type of the inverse function.

### 10.175.3 Constructor & Destructor Documentation

**10.175.3.1** `template<unsigned n, typename C > mln::fun::x2x::rotation< n, C >::rotation ( )`  
`[inline]`

Constructor without argument.

**10.175.3.2** `template<unsigned n, typename C > mln::fun::x2x::rotation< n, C >::rotation ( C`  
`alpha, const algebra::vec< n, C > & axis ) [inline]`

Constructor with radian alpha and a facultative direction (rotation axis).

**10.175.3.3** `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()`.

**10.175.3.4** `template<unsigned n, typename C > mln::fun::x2x::rotation< n, C >::rotation ( const algebra::h_mat< n, C > & m ) [inline]`

Constructor with `h_mat`.

## 10.175.4 Member Function Documentation

**10.175.4.1** `template<unsigned n, typename C > rotation< n, C > mln::fun::x2x::rotation< n, C >::inv ( ) const [inline]`

Return the inverse function.

**10.175.4.2** `template<unsigned n, typename C > algebra::vec< n, C > mln::fun::x2x::rotation< n, C >::operator() ( const algebra::vec< n, C > & v ) const [inline]`

Perform the rotation of the given vector.

**10.175.4.3** `template<unsigned n, typename C > void mln::fun::x2x::rotation< n, C >::set_alpha ( C alpha ) [inline]`

Set a new grade alpha.

**10.175.4.4** `template<unsigned n, typename C > void mln::fun::x2x::rotation< n, C >::set_axis ( const algebra::vec< n, C > & axis ) [inline]`

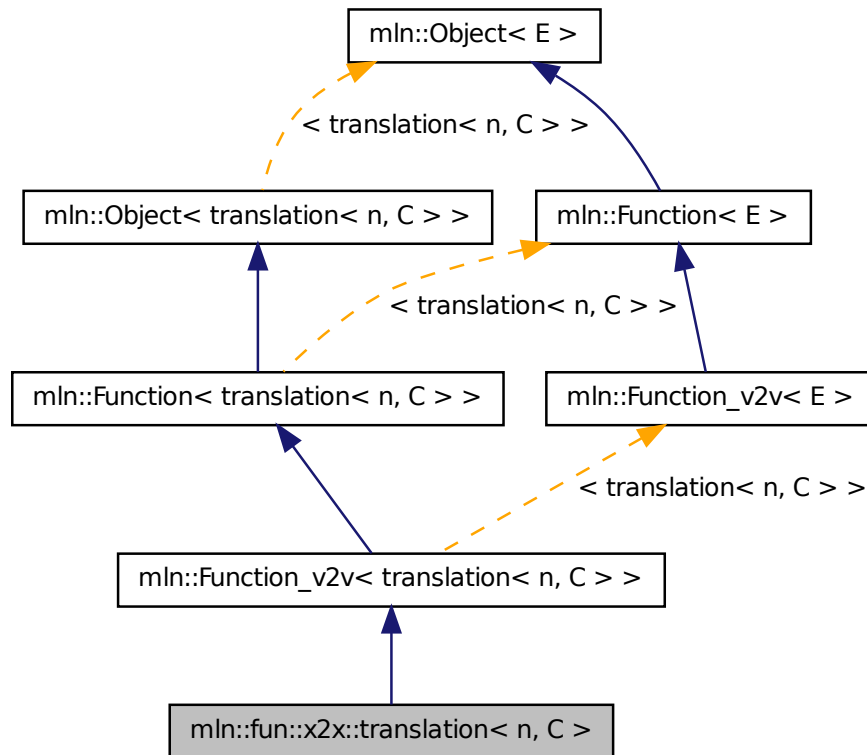
Set a new rotation axis.

## 10.176 mln::fun::x2x::translation< n, C > Struct Template Reference

Translation function-object.

```
#include <translation.hh>
```

Inheritance diagram for mln::fun::x2x::translation< n, C >:



## Public Types

- typedef `C` [data\\_t](#)  
*Type of the underlying data stored in vectors and matrices.*
- typedef `translation< n, C >` [invert](#)  
*Type of the inverse function.*

## Public Member Functions

- `invert inv () const`  
*Return the inverse function.*
- `algebra::vec< n, C > operator() (const algebra::vec< n, C > &v) const`  
*Perform the translation of the given vector.*

- void `set_t` (const algebra::vec< n, C > &t)  
*Set a net translation vector.*
- const algebra::vec< n, C > &t () const  
*Return the translation vector.*
- `translation` (const algebra::vec< n, C > &t)  
*Constructor with the translation vector.*
- `translation` ()  
*Constructor without argument.*

### 10.176.1 Detailed Description

`template<unsigned n, typename C> struct mln::fun::x2x::translation< n, C >`

Translation function-object.

### 10.176.2 Member Typedef Documentation

**10.176.2.1** `template<unsigned n, typename C > typedef C mln::fun::x2x::translation< n, C >::data_t`

Type of the underlying data stored in vectors and matrices.

**10.176.2.2** `template<unsigned n, typename C > typedef translation<n,C> mln::fun::x2x::translation< n, C >::invert`

Type of the inverse function.

### 10.176.3 Constructor & Destructor Documentation

**10.176.3.1** `template<unsigned n, typename C > mln::fun::x2x::translation< n, C >::translation ( ) [inline]`

Constructor without argument.

**10.176.3.2** `template<unsigned n, typename C > mln::fun::x2x::translation< n, C >::translation ( const algebra::vec< n, C > & t ) [inline]`

Constructor with the translation vector.



### 10.176.4 Member Function Documentation

**10.176.4.1** `template<unsigned n, typename C > translation< n, C > mln::fun::x2x::translation< n, C >::inv ( ) const [inline]`

Return the inverse function.

**10.176.4.2** `template<unsigned n, typename C > algebra::vec< n, C > mln::fun::x2x::translation< n, C >::operator() ( const algebra::vec< n, C > & v ) const [inline]`

Perform the translation of the given vector.

**10.176.4.3** `template<unsigned n, typename C > void mln::fun::x2x::translation< n, C >::set_t ( const algebra::vec< n, C > & t ) [inline]`

Set a net translation vector.

**10.176.4.4** `template<unsigned n, typename C > const algebra::vec< n, C > & mln::fun::x2x::translation< n, C >::t ( ) const [inline]`

Return the translation vector.

## 10.177 mln::fun\_image< F, I > Struct Template Reference

[Image](#) read through a function.

```
#include <fun_image.hh>
```

Inherits `image_value_morpher< I, F::result, fun_image< F, I > >`.

### Public Types

- typedef `F::result lvalue`  
*Return type of read-write access.*
- typedef `F::result rvalue`  
*Return type of read-only access.*
- typedef `fun_image< tag::value_< typename F::result >, tag::image_< I > > skeleton`  
*Skeleton.*
- typedef `F::result value`  
*Value associated type.*

### Public Member Functions

- `fun_image ( )`

*Constructor:*

- `fun_image` (const `Function_v2v`< F > &f, const `Image`< I > &ima)

*Constructor:*

- `fun_image` (const `Image`< I > &ima)

*Constructor:*

- `F::result operator()` (const typename I::psite &p) const

*Read-only access of pixel value at point site p.*

- `F::result operator()` (const typename I::psite &p)

*Mutable access is for reading only.*

### 10.177.1 Detailed Description

`template<typename F, typename I> struct mln::fun_image< F, I >`

`Image` read through a function.

### 10.177.2 Member Typedef Documentation

**10.177.2.1** `template<typename F, typename I> typedef F ::result mln::fun_image< F, I >::lvalue`

Return type of read-write access.

**10.177.2.2** `template<typename F, typename I> typedef F ::result mln::fun_image< F, I >::rvalue`

Return type of read-only access.

**10.177.2.3** `template<typename F, typename I> typedef fun_image< tag::value_<typename F ::result>, tag::image_<I> > mln::fun_image< F, I >::skeleton`

Skeleton.

**10.177.2.4** `template<typename F, typename I> typedef F ::result mln::fun_image< F, I >::value`

`Value` associated type.

### 10.177.3 Constructor & Destructor Documentation

**10.177.3.1** `template<typename F, typename I> mln::fun_image< F, I >::fun_image ( )`  
`[inline]`

Constructor.

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

Constructor.

**10.177.3.3** `template<typename F , typename I > mln::fun_image< F, I >::fun_image ( const Image< I > & ima ) [inline]`

Constructor.

## 10.177.4 Member Function Documentation

**10.177.4.1** `template<typename F , typename I > F::result mln::fun_image< F, I >::operator() ( const typename I::psite & p ) const [inline]`

Read-only access of pixel value at point site p.

**10.177.4.2** `template<typename F , typename I > F::result mln::fun_image< F, I >::operator() ( const typename I::psite & p ) [inline]`

Mutable access is for reading only.

## 10.178 mln::Function< E > Struct Template Reference

Base class for implementation of function-objects.

```
#include <function.hh>
```

Inherits [mln::Object< E >](#).

Inherited by [mln::Function\\_n2v< E >](#), [mln::Function\\_v2v< E >](#), [mln::Function\\_vv2b< E >](#), and [mln::Function\\_vv2v< E >](#).

### Protected Member Functions

- [Function \(\)](#)

*An operator() has to be provided.*

### 10.178.1 Detailed Description

```
template<typename E> struct mln::Function< E >
```

Base class for implementation of function-objects. The parameter *E* is the exact type.

## 10.178.2 Constructor & Destructor Documentation

**10.178.2.1** `template<typename E > mln::Function< E >::Function ( ) [inline, protected]`

An operator() has to be provided.

Its signature depends on the particular function-object one considers.

## 10.179 mln::Function< void > Struct Template Reference

[Function](#) category flag type.

```
#include <function.hh>
```

### 10.179.1 Detailed Description

```
template<> struct mln::Function< void >
```

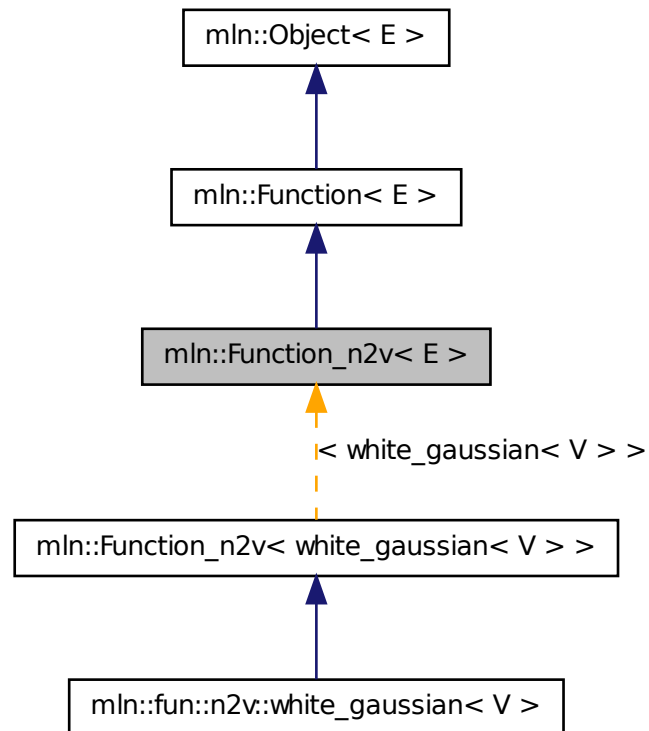
[Function](#) category flag type.

## 10.180 mln::Function\_n2v< E > Struct Template Reference

Base class for implementation of function-objects from Nil to value.

```
#include <function.hh>
```

Inheritance diagram for mln::Function\_n2v< E >:



### 10.180.1 Detailed Description

```
template<typename E> struct mln::Function_n2v< E >
```

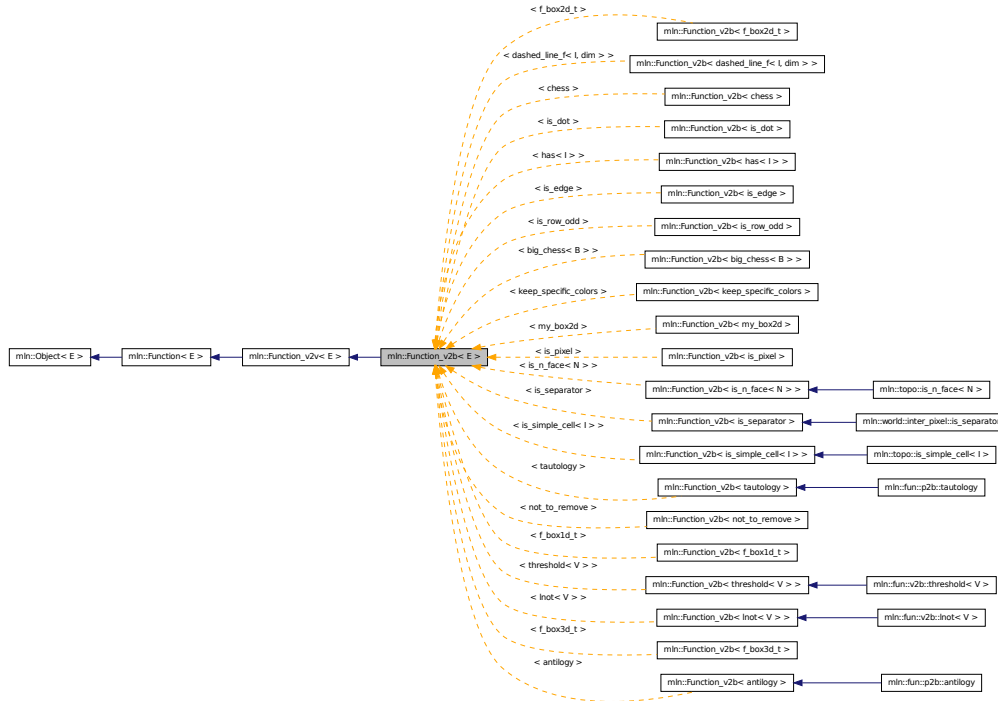
Base class for implementation of function-objects from Nil to value. The parameter *E* is the exact type.

## 10.181 mln::Function\_v2b< E > Struct Template Reference

Base class for implementation of function-objects from a value to a Boolean.

```
#include <function.hh>
```

Inheritance diagram for `mln::Function_v2b< E >`:



### 10.181.1 Detailed Description

```
template<typename E> struct mln::Function_v2b< E >
```

Base class for implementation of function-objects from a value to a Boolean. The parameter *E* is the exact type.

## 10.182 mln::Function\_v2v< E > Struct Template Reference

Base class for implementation of function-objects from value to value.

```
#include <function.hh>
```

Inherits [mln::Function< E >](#).

Inherited by [mln::fun::C< R\(\\*\) \(A\) >](#), [mln::fun::v2v::dec< T >](#), [mln::fun::v2v::id< T >](#), [mln::fun::v2v::inc< T >](#), [mln::fun::x2v::bilinear< I >](#), [mln::fun::x2v::trilinear< I >](#), [mln::fun::x2x::internal::helper\\_composed< T2, T1, E, false >](#), [mln::fun::x2x::internal::helper\\_composed< T2, T1, E, true >](#), [mln::fun::x2x::linear< I >](#), [mln::fun::x2x::neighbor< I >](#), and [mln::Function\\_v2b< E >](#) [virtual].

### 10.182.1 Detailed Description

```
template<typename E> struct mln::Function_v2v< E >
```

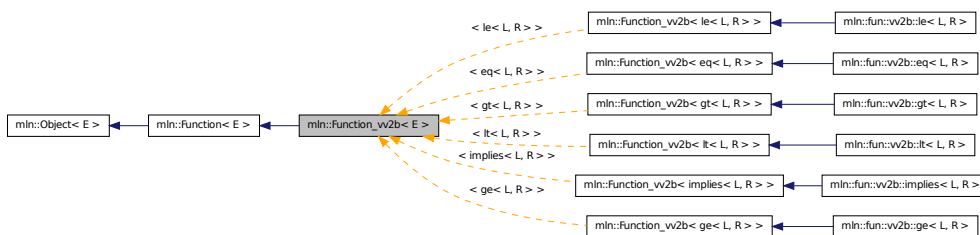
Base class for implementation of function-objects from value to value. The parameter *E* is the exact type.

## 10.183 mln::Function\_vv2b< E > Struct Template Reference

Base class for implementation of function-objects from a couple of values to a Boolean.

```
#include <function.hh>
```

Inheritance diagram for mln::Function\_vv2b< E >:



### 10.183.1 Detailed Description

```
template<typename E> struct mln::Function_vv2b< E >
```

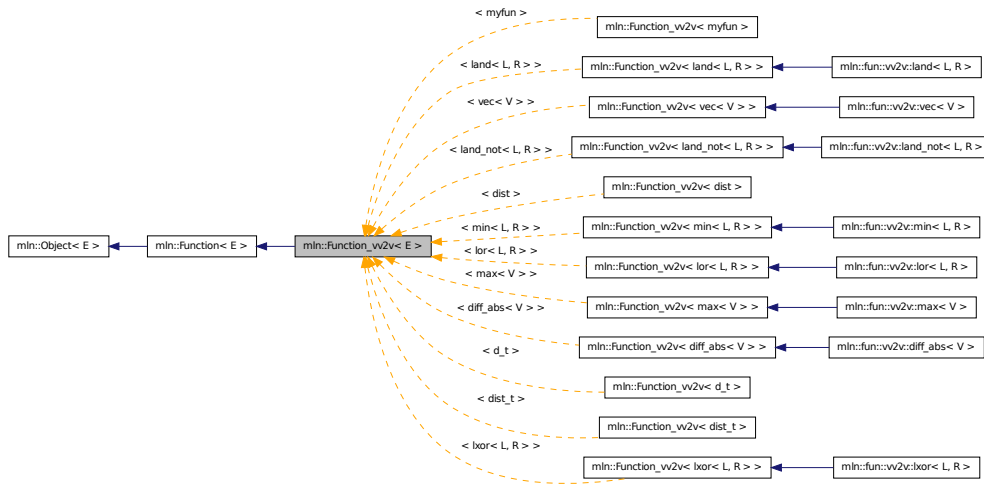
Base class for implementation of function-objects from a couple of values to a Boolean. The parameter *E* is the exact type.

## 10.184 mln::Function\_vv2v< E > Struct Template Reference

Base class for implementation of function-objects from a couple of values to a value.

```
#include <function.hh>
```

Inheritance diagram for `mln::Function_vv2v< E >`:



### 10.184.1 Detailed Description

`template<typename E> struct mln::Function_vv2v< E >`

Base class for implementation of function-objects from a couple of values to a value. The parameter *E* is the exact type.

## 10.185 mln::fwd\_pixter1d< I > Class Template Reference

Forward pixel iterator on a 1-D image with border.

```
#include <pixter1d.hh>
```

Inherits `forward_pixel_iterator_base< I, fwd_pixter1d< I > >`.

### Public Types

- typedef `I` `image`  
*Image type.*

### Public Member Functions

- `fwd_pixter1d` (`I &image`)  
*Constructor.*
- `void next` ()  
*Go to the next element.*



### 10.185.1 Detailed Description

`template<typename I> class mln::fwd_pixter1d< I >`

Forward pixel iterator on a 1-D image with border.

### 10.185.2 Member Typedef Documentation

**10.185.2.1** `template<typename I > typedef I mln::fwd_pixter1d< I >::image`

[Image](#) type.

### 10.185.3 Constructor & Destructor Documentation

**10.185.3.1** `template<typename I > mln::fwd_pixter1d< I >::fwd_pixter1d ( I & image )`  
**[inline]**

Constructor.

#### Parameters

**[in]** *image* The image this pixel iterator is bound to.

### 10.185.4 Member Function Documentation

**10.185.4.1** `void mln::Iterator< fwd_pixter1d< I > >::next ( )` **[inherited]**

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

#### Precondition

The iterator is valid.

## 10.186 mln::fwd\_pixter2d< I > Class Template Reference

Forward pixel iterator on a 2-D image with border.

```
#include <pixter2d.hh>
```

Inherits `forward_pixel_iterator_base_< I, fwd_pixter2d< I > >`.

### Public Types

- typedef I [image](#)  
*Image* type.

## Public Member Functions

- [fwd\\_pixter2d](#) (I &image)

*Constructor.*

- void [next](#) ()

*Go to the next element.*

### 10.186.1 Detailed Description

```
template<typename I> class mln::fwd_pixter2d< I >
```

Forward pixel iterator on a 2-D image with border.

### 10.186.2 Member Typedef Documentation

**10.186.2.1** `template<typename I > typedef I mln::fwd_pixter2d< I >::image`

[Image](#) type.

### 10.186.3 Constructor & Destructor Documentation

**10.186.3.1** `template<typename I > mln::fwd_pixter2d< I >::fwd_pixter2d ( I & image )`  
`[inline]`

Constructor.

#### Parameters

`[in]` *image* The image this pixel iterator is bound to.

### 10.186.4 Member Function Documentation

**10.186.4.1** `void mln::Iterator< fwd_pixter2d< I > >::next ( )` `[inherited]`

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

#### Precondition

The iterator is valid.

## 10.187 mln::fwd\_pixter3d< I > Class Template Reference

Forward pixel iterator on a 3-D image with border.

```
#include <pixter3d.hh>
```

Inherits forward\_pixel\_iterator\_base\_< I, fwd\_pixter3d< I > >.

### Public Types

- typedef I [image](#)

*Image type.*

### Public Member Functions

- [fwd\\_pixter3d](#) (I &image)

*Constructor.*

- void [next](#) ()

*Go to the next element.*

### 10.187.1 Detailed Description

```
template<typename I> class mln::fwd_pixter3d< I >
```

Forward pixel iterator on a 3-D image with border.

### 10.187.2 Member Typedef Documentation

10.187.2.1 `template<typename I > typedef I mln::fwd_pixter3d< I >::image`

[Image](#) type.

### 10.187.3 Constructor & Destructor Documentation

10.187.3.1 `template<typename I > mln::fwd_pixter3d< I >::fwd_pixter3d ( I & image )`  
`[inline]`

Constructor.

#### Parameters

`[in]` *image* The image this pixel iterator is bound to.

## 10.187.4 Member Function Documentation

### 10.187.4.1 void mln::Iterator< fwd\_pixter3d< I > >::next ( ) [inherited]

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

#### Precondition

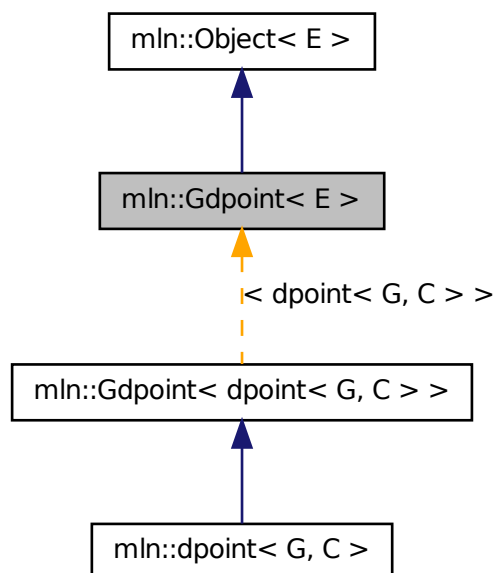
The iterator is valid.

## 10.188 mln::Gdpoint< E > Struct Template Reference

FIXME: Doc!

```
#include <gdpoint.hh>
```

Inheritance diagram for mln::Gdpoint< E >:



### 10.188.1 Detailed Description

`template<typename E> struct mln::Gdpoint< E >`

FIXME: Doc!

## 10.189 mln::Gdpoint< void > Struct Template Reference

Delta point site category flag type.

`#include <gdpoint.hh>`

### 10.189.1 Detailed Description

`template<> struct mln::Gdpoint< void >`

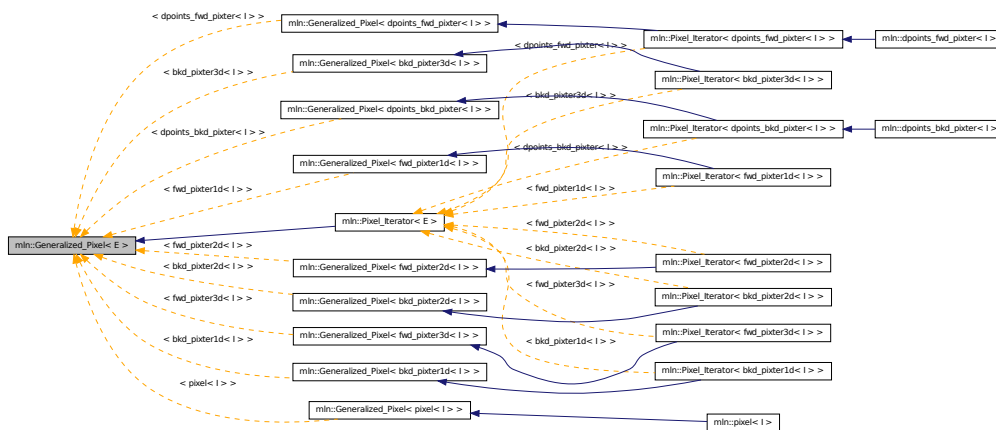
Delta point site category flag type.

## 10.190 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 >`:



### 10.190.1 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.

## 10.191 mln::geom::complex\_geometry< D, P > Class Template Reference

A functor returning the sites of the faces of a complex where the locations of each 0-face is stored.

```
#include <complex_geometry.hh>
```

**Public Member Functions**

- unsigned [add\\_location](#) (const P &p)  
*Populate the set of locations.*
- [complex\\_geometry](#) ()  
*Build a complex geometry object.*
- site [operator\(\)](#) (const [mln::topo::face](#)< D > &f) const  
*Retrieve the site associated to f.*

**10.191.1 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.

## 10.191.2 Constructor & Destructor Documentation

**10.191.2.1** `template<unsigned D, typename P > mln::geom::complex_geometry< D, P >::complex_geometry( ) [inline]`

Build a complex geometry object.

## 10.191.3 Member Function Documentation

**10.191.3.1** `template<unsigned D, typename P > unsigned mln::geom::complex_geometry< D, P >::add_location( const P & p ) [inline]`

Populate the set of locations.

Append a new location *p*. Return the index of the newly created location (which should semantically match the id of the corresponding 0-face in the complex).

**10.191.3.2** `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*.

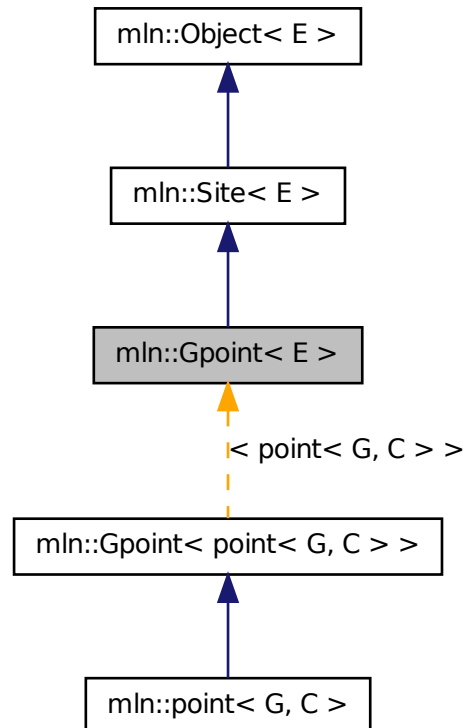
References `mln::topo::face< D >::face_id()`, and `mln::topo::face< D >::n()`.

## 10.192 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)`  
*Divide a point by a scalar  $s$ .*
- `template<typename P >`  
`std::ostream & operator<< (std::ostream &ostr, const Gpoint< P > &p)`  
*Print a grid point  $p$  into the output stream  $ostr$ .*
- `template<typename L, typename R >`  
`bool operator== (const Gpoint< L > &lhs, const Gpoint< R > &rhs)`  
*Equality comparison between a couple of grid point  $lhs$  and  $rhs$ .*

### 10.192.1 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.

### 10.192.2 Friends And Related Function Documentation

**10.192.2.1** `template<typename P, typename D > P operator+ ( const Gpoint< P > & p, const Gdpoint< D > & dp )` [**related**]

Add a delta-point `rhs` to a grid point `lhs`.

#### Parameters

- [in] `p` A grid point.
- [in] `dp` A delta-point.

The type of `dp` has to compatible with the type of `p`.

#### Returns

A point (temporary object).

#### See also

[mln::Gdpoint](#)

**10.192.2.2** `template<typename P, typename D > P & operator+= ( Gpoint< P > & p, const Gdpoint< D > & dp )` [**related**]

Shift a point by a delta-point `dp`.

**Parameters**

[in, out] *p* The targeted point.  
 [in] *dp* A delta-point.

**Returns**

A reference to the point *p* once translated by *dp*.

**Precondition**

The type of *dp* has to be compatible with the type of *p*.

### 10.192.2.3 `template<typename L , typename R > L::delta operator- ( const Gpoint< L > & lhs, const Gpoint< R > & rhs ) [related]`

Difference between a couple of grid point *lhs* and *rhs*.

**Parameters**

[in] *lhs* A first grid point.  
 [in] *rhs* A second grid point.

**Warning**

There is no type promotion in Milena so the client has to make sure that both points are defined with the same type of coordinates.

**Precondition**

Both *lhs* and *rhs* have to be defined on the same topology and with the same type of coordinates; otherwise this test does not compile.

**Postcondition**

The result, *dp*, is such as  $lhs == rhs + dp$ .

**Returns**

A delta point (temporary object).

**See also**

[mln::Gdpoint](#)

### 10.192.2.4 `template<typename P , typename D > P & operator-= ( Gpoint< P > & p, const Gdpoint< D > & dp ) [related]`

Shift a point *by* the negate of a delta-point *dp*.

**Parameters**

[in, out] *p* The targeted point.  
 [in] *dp* A delta-point.

**Returns**

A reference to the point *p* once translated by  $- dp$ .

**Precondition**

The type of *dp* has to be compatible with the type of *p*.

**10.192.2.5** `template<typename P, typename D > P operator/( const Gpoint< P > & p, const value::scalar_< D > & dp ) [related]`

Divide a point by a scalar *s*.

**Parameters**

[in, out] *p* The targeted point.

[in] *dp* A scalar.

**Returns**

A reference to the point *p* once divided by *s*.

**10.192.2.6** `template<typename P > std::ostream & operator<< ( std::ostream & ostr, const Gpoint< P > & p ) [related]`

Print a grid point *p* into the output stream *ostr*.

**Parameters**

[in, out] *ostr* An output stream.

[in] *p* A grid point.

**Returns**

The modified output stream *ostr*.

References mln::debug::format().

**10.192.2.7** `template<typename L, typename R > bool operator==( const Gpoint< L > & lhs, const Gpoint< R > & rhs ) [related]`

Equality comparison between a couple of grid point *lhs* and *rhs*.

**Parameters**

[in] *lhs* A first grid point.

[in] *rhs* A second grid point.

**Precondition**

Both *lhs* and *rhs* have to be defined on the same topology; otherwise this test does not compile.

**Returns**

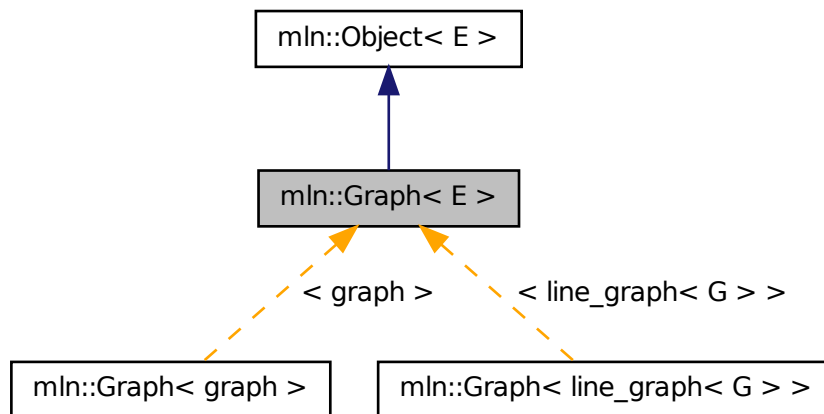
True if both grid points have the same coordinates, otherwise false.

## 10.193 mln::Graph< E > Struct Template Reference

Base class for implementation of graph classes.

```
#include <graph.hh>
```

Inheritance diagram for mln::Graph< E >:



### 10.193.1 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.

## 10.194 mln::graph::attribute::card\_t Struct Reference

Compute the cardinality of every component in a graph.

```
#include <card.hh>
```

### Public Types

- typedef `util::array< unsigned >` `result`

*Type of the computed value.*

### 10.194.1 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.

### 10.194.2 Member Typedef Documentation

#### 10.194.2.1 typedef util::array<unsigned> mln::graph::attribute::card\_t::result

Type of the computed value.

## 10.195 mln::graph::attribute::representative\_t Struct Reference

Compute the representative vertex of every component in a graph.

```
#include <representative.hh>
```

### Public Types

- typedef [util::array< unsigned > result](#)

*Type of the computed value.*

### 10.195.1 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.

### 10.195.2 Member Typedef Documentation

#### 10.195.2.1 typedef util::array<unsigned> mln::graph::attribute::representative\_t::result

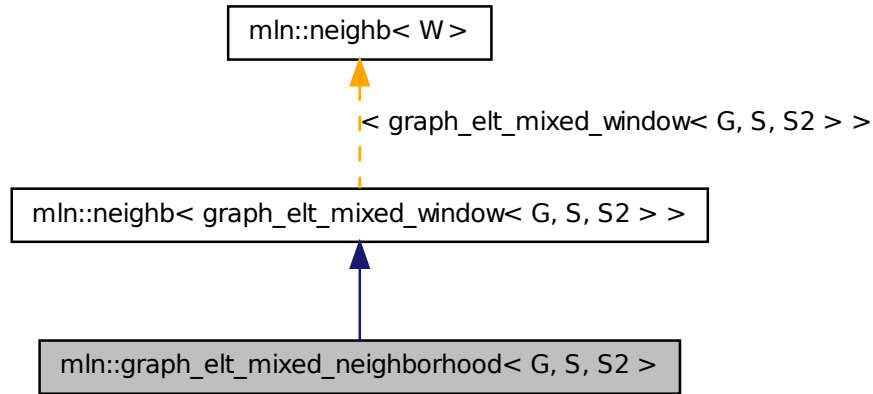
Type of the computed value.

## 10.196 mln::graph\_elt\_mixed\_neighborhood< G, S, S2 > Struct Template Reference

Elementary neighborhood on graph class.

```
#include <graph_elt_mixed_neighborhood.hh>
```

Inheritance diagram for `mln::graph_elt_mixed_neighborhood< G, S, S2 >`:



## Public Types

- typedef `neighb_bkd_niter< graph_elt_mixed_window< G, S, S2 > > bkd_niter`  
*Backward site iterator associated type.*
- typedef `neighb_fwd_niter< graph_elt_mixed_window< G, S, S2 > > fwd_niter`  
*Forward site iterator associated type.*
- typedef `fwd_niter niter`  
*Site iterator associated type.*

### 10.196.1 Detailed Description

```
template<typename G, typename S, typename S2> struct mln::graph_elt_mixed_neighborhood<
G, S, S2 >
```

Elementary neighborhood on graph class.

#### Template Parameters

*G* is a graph type.

*S* is a site set type.

*S2* is the site set type of the neighbors.

## 10.196.2 Member Typedef Documentation

**10.196.2.1** `typedef neighb_bkd_niter<graph_elt_mixed_window< G, S, S2 > > mln::neighb<graph_elt_mixed_window< G, S, S2 > >::bkd_niter` [inherited]

Backward site iterator associated type.

**10.196.2.2** `typedef neighb_fwd_niter<graph_elt_mixed_window< G, S, S2 > > mln::neighb<graph_elt_mixed_window< G, S, S2 > >::fwd_niter` [inherited]

Forward site iterator associated type.

**10.196.2.3** `typedef fwd_niter mln::neighb< graph_elt_mixed_window< G, S, S2 > >::niter` [inherited]

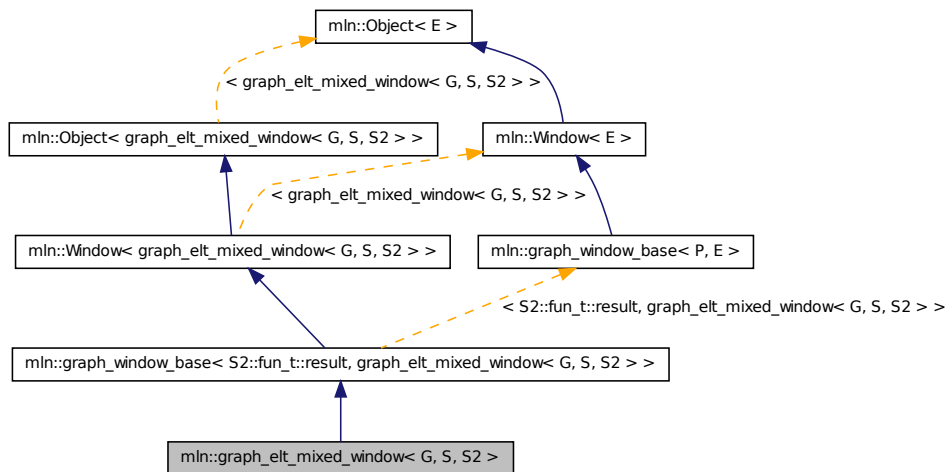
Site iterator associated type.

## 10.197 mln::graph\_elt\_mixed\_window< G, S, S2 > Class Template Reference

Elementary window on graph class.

```
#include <graph_elt_mixed_window.hh>
```

Inheritance diagram for mln::graph\_elt\_mixed\_window< G, S, S2 >:



## Public Types

- typedef `super_::target` [target](#)  
Associated types.

- typedef target::psite [psite](#)  
*The type of psite corresponding to the window.*
- typedef S::psite [center\\_t](#)  
*Type of the window center element.*
- typedef target::graph\_element [graph\\_element](#)  
*Type of the graph element pointed by this iterator.*
- typedef [graph\\_window\\_piter](#)< target, self\_, nbh\_fwd\_iter\_ > [fwd\\_qiter](#)  
*Site\_Iterator type to browse the psites of the window w.r.t.*
- typedef [graph\\_window\\_piter](#)< target, self\_, nbh\_bkd\_iter\_ > [bkd\\_qiter](#)  
*Site\_Iterator type to browse the psites of the window w.r.t.*
- typedef [fwd\\_qiter](#) [qiter](#)  
*The default qiter type.*
- typedef S2::fun\_t::result [site](#)  
*Associated types.*

## Public Member Functions

- bool [is\\_valid](#) () const  
*Return true by default.*
- bool [is\\_empty](#) () const  
*Interface of the concept Window.*
- bool [is\\_centered](#) () const  
*Is the window centered?*
- bool [is\\_symmetric](#) () const  
*Is the window symmetric?*
- unsigned [delta](#) () const  
*Return the maximum coordinate gap between the window center and a window point.*
- [self\\_ & sym](#) ()  
*Apply a central symmetry to the target window.*

### 10.197.1 Detailed Description

```
template<typename G, typename S, typename S2> class mln::graph_elt_mixed_window< G, S, S2 >
```

Elementary window on graph class. `G` is the graph type. `S` is an image site set from where the center is extracted. `S2` is an image site set from where the neighbors are extracted.



## 10.197.2 Member Typedef Documentation

**10.197.2.1** `template<typename G , typename S , typename S2 > typedef graph_window_piter<target,self_,nbh_bkd_iter_> mln::graph_elt_mixed_window< G, S, S2 >::bkd_qiter`

[Site\\_Iterator](#) type to browse the psites of the window w.r.t.

the reverse ordering of vertices.

**10.197.2.2** `template<typename G , typename S , typename S2 > typedef S ::psite mln::graph_elt_mixed_window< G, S, S2 >::center_t`

Type of the window center element.

**10.197.2.3** `template<typename G , typename S , typename S2 > typedef graph_window_piter<target,self_,nbh_fwd_iter_> mln::graph_elt_mixed_window< G, S, S2 >::fwd_qiter`

[Site\\_Iterator](#) type to browse the psites of the window w.r.t.

the ordering of vertices.

**10.197.2.4** `template<typename G , typename S , typename S2 > typedef target ::graph_element mln::graph_elt_mixed_window< G, S, S2 >::graph_element`

Type of the graph element pointed by this iterator.

**10.197.2.5** `template<typename G , typename S , typename S2 > typedef target ::psite mln::graph_elt_mixed_window< G, S, S2 >::psite`

The type of psite corresponding to the window.

**10.197.2.6** `template<typename G , typename S , typename S2 > typedef fwd_qiter mln::graph_elt_mixed_window< G, S, S2 >::qiter`

The default qiter type.

**10.197.2.7** `typedef S2::fun_t::result mln::graph_window_base< S2::fun_t::result , graph_elt_mixed_window< G, S, S2 > >::site [inherited]`

Associated types.

The type of site corresponding to the window.

**10.197.2.8** `template<typename G , typename S , typename S2 > typedef super_::target mln::graph_elt_mixed_window< G, S, S2 >::target`

Associated types.

### 10.197.3 Member Function Documentation

**10.197.3.1** `unsigned mln::graph_window_base< S2::fun_t::result , graph_elt_mixed_window< G, S, S2 > >::delta ( ) const` `[inherited]`

Return the maximum coordinate gap between the window center and a window point.

**10.197.3.2** `bool mln::graph_window_base< S2::fun_t::result , graph_elt_mixed_window< G, S, S2 > >::is_centered ( ) const` `[inherited]`

Is the window centered?

**10.197.3.3** `bool mln::graph_window_base< S2::fun_t::result , graph_elt_mixed_window< G, S, S2 > >::is_empty ( ) const` `[inherited]`

Interface of the concept Window.

Is the window is empty?

**10.197.3.4** `bool mln::graph_window_base< S2::fun_t::result , graph_elt_mixed_window< G, S, S2 > >::is_symmetric ( ) const` `[inherited]`

Is the window symmetric?

**10.197.3.5** `bool mln::graph_window_base< S2::fun_t::result , graph_elt_mixed_window< G, S, S2 > >::is_valid ( ) const` `[inherited]`

Return true by default.

**10.197.3.6** `self_ & mln::graph_window_base< S2::fun_t::result , graph_elt_mixed_window< G, S, S2 > >::sym ( )` `[inherited]`

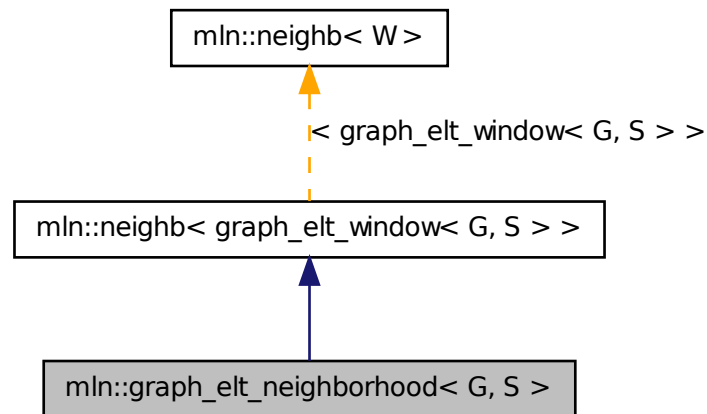
Apply a central symmetry to the target window.

## 10.198 mln::graph\_elt\_neighborhood< G, S > Struct Template Reference

Elementary neighborhood on graph class.

```
#include <graph_elt_neighborhood.hh>
```

Inheritance diagram for mln::graph\_elt\_neighborhood< G, S >:



## Public Types

- typedef `neighb_bkd_niter< graph\_elt\_window< G, S > > bkd_niter`  
*Backward site iterator associated type.*
- typedef `neighb_fwd_niter< graph\_elt\_window< G, S > > fwd_niter`  
*Forward site iterator associated type.*
- typedef `fwd_niter niter`  
*Site iterator associated type.*

### 10.198.1 Detailed Description

```
template<typename G, typename S> struct mln::graph_elt_neighborhood< G, S >
```

Elementary neighborhood on graph class.

#### Template Parameters

*G* is a graph type.

*S* is a site set type.

## 10.198.2 Member Typedef Documentation

**10.198.2.1** `typedef neighb_bkd_niter<graph_elt_window< G, S >> mln::neighb<graph_elt_window< G, S >>::bkd_niter` `[inherited]`

Backward site iterator associated type.

**10.198.2.2** `typedef neighb_fwd_niter<graph_elt_window< G, S >> mln::neighb<graph_elt_window< G, S >>::fwd_niter` `[inherited]`

Forward site iterator associated type.

**10.198.2.3** `typedef fwd_niter mln::neighb< graph_elt_window< G, S >>::niter` `[inherited]`

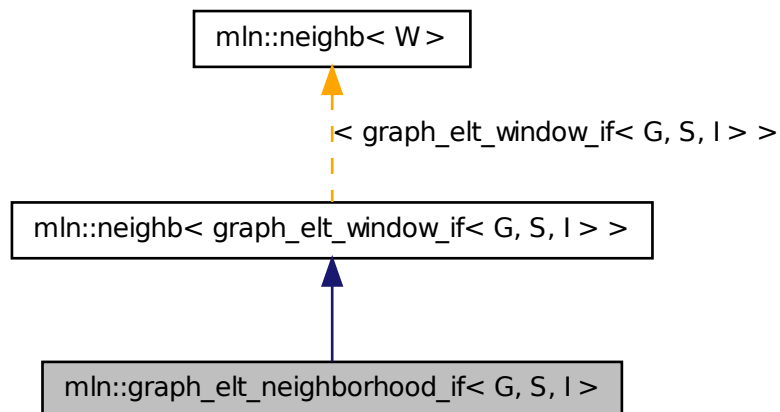
Site iterator associated type.

## 10.199 mln::graph\_elt\_neighborhood\_if< G, S, I > Struct Template Reference

Elementary neighborhood\_if on graph class.

```
#include <graph_elt_neighborhood_if.hh>
```

Inheritance diagram for mln::graph\_elt\_neighborhood\_if< G, S, I >:



### Public Types

- `typedef neighb_bkd_niter< graph\_elt\_window\_if< G, S, I >> bkd_niter`  
*Backward site iterator associated type.*

- typedef `neighb_fwd_niter< graph_elt_window_if< G, S, I > > fwd_niter`  
*Forward site iterator associated type.*
- typedef `fwd_niter niter`  
*Site iterator associated type.*

## Public Member Functions

- `graph_elt_neighborhood_if()`  
*Constructors @ { Construct an invalid neighborhood.*
- `graph_elt_neighborhood_if(const Image< I > &mask)`
- `const I & mask() const`  
*@ }*

### 10.199.1 Detailed Description

```
template<typename G, typename S, typename I> struct mln::graph_elt_neighborhood_if< G, S, I
>
```

Elementary neighborhood\_if on graph class.

### 10.199.2 Member Typedef Documentation

**10.199.2.1** `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.

**10.199.2.2** `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.

**10.199.2.3** `typedef fwd_niter mln::neighb< graph_elt_window_if< G, S, I > >::niter` **[inherited]**

Site iterator associated type.

### 10.199.3 Constructor & Destructor Documentation

**10.199.3.1** `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.

**10.199.3.2** `template<typename G , typename S , typename I > mln::graph_elt_neighborhood_if< G, S, I >::graph_elt_neighborhood_if( const Image< I > & mask ) [inline]`

#### Parameters

[in] *mask* A graph image of Boolean.

### 10.199.4 Member Function Documentation

**10.199.4.1** `template<typename G , typename S , typename I > const I & mln::graph_elt_neighborhood_if< G, S, I >::mask ( ) const [inline]`

@}

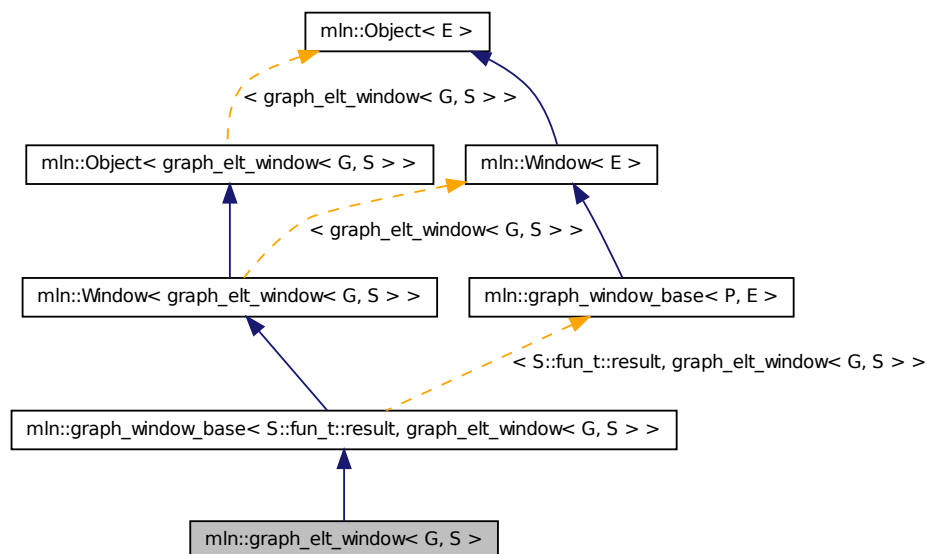
Return the graph image used as mask.

## 10.200 mln::graph\_elt\_window< G, S > Class Template Reference

Elementary window on graph class.

```
#include <graph_elt_window.hh>
```

Inheritance diagram for `mln::graph_elt_window< G, S >`:



### Public Types

- typedef `S target`  
*Associated types.*

- typedef S::psite [psite](#)  
*The type of psite corresponding to the window.*
- typedef S::psite [center\\_t](#)  
*Type of the window center element.*
- typedef S::graph\_element [graph\\_element](#)  
*Type of the graph element pointed by this iterator.*
- typedef [graph\\_window\\_piter](#)< S, [self\\_](#), [nbh\\_fwd\\_iter\\_](#) > [fwd\\_qiter](#)  
*Site\_Iterator type to browse the psites of the window w.r.t.*
- typedef [graph\\_window\\_piter](#)< S, [self\\_](#), [nbh\\_bkd\\_iter\\_](#) > [bkd\\_qiter](#)  
*Site\_Iterator type to browse the psites of the window w.r.t.*
- typedef [fwd\\_qiter](#) [qiter](#)  
*The default qiter type.*
  
- typedef S::fun\_t::result [site](#)  
*Associated types.*

## Public Member Functions

- bool [is\\_valid](#) () const  
*Return true by default.*
  
- bool [is\\_empty](#) () const  
*Interface of the concept Window.*
  
- bool [is\\_centered](#) () const  
*Is the window centered?*
  
- bool [is\\_symmetric](#) () const  
*Is the window symmetric?*
  
- unsigned [delta](#) () const  
*Return the maximum coordinate gap between the window center and a window point.*
  
- [self\\_ & sym](#) ()  
*Apply a central symmetry to the target window.*

### 10.200.1 Detailed Description

**template**<typename G, typename S> **class** mln::graph\_elt\_window< G, S >

Elementary window on graph class. G is the graph type. S is an image site set from where the center is extracted. S2 is an image site set from where the neighbors are extracted.

## 10.200.2 Member Typedef Documentation

**10.200.2.1** `template<typename G , typename S > typedef graph_window_ -  
piter<S,self,_nbh_bkd_iter_> mln::graph_elt_window< G, S  
>::bkd_qiter`

[Site\\_Iterator](#) type to browse the psites of the window w.r.t.  
the reverse ordering of vertices.

**10.200.2.2** `template<typename G , typename S > typedef S ::psite mln::graph_elt_window< G, S  
>::center_t`

Type of the window center element.

**10.200.2.3** `template<typename G , typename S > typedef graph_window_ -  
piter<S,self,_nbh_fwd_iter_> mln::graph_elt_window< G, S  
>::fwd_qiter`

[Site\\_Iterator](#) type to browse the psites of the window w.r.t.  
the ordering of vertices.

**10.200.2.4** `template<typename G , typename S > typedef S ::graph_element  
mln::graph_elt_window< G, S >::graph_element`

Type of the graph element pointed by this iterator.

**10.200.2.5** `template<typename G , typename S > typedef S ::psite mln::graph_elt_window< G, S  
>::psite`

The type of psite corresponding to the window.

**10.200.2.6** `template<typename G , typename S > typedef fwd_qiter mln::graph_elt_window< G,  
S >::qiter`

The default qiter type.

**10.200.2.7** `typedef S::fun_t::result mln::graph_window_base< S::fun_t::result ,  
graph_elt_window< G, S > >::site [inherited]`

Associated types.

The type of site corresponding to the window.

**10.200.2.8** `template<typename G , typename S > typedef S mln::graph_elt_window< G, S  
>::target`

Associated types.



### 10.200.3 Member Function Documentation

**10.200.3.1** `unsigned mln::graph_window_base< S::fun_t::result , graph_elt_window< G, S > >::delta ( ) const [inherited]`

Return the maximum coordinate gap between the window center and a window point.

**10.200.3.2** `bool mln::graph_window_base< S::fun_t::result , graph_elt_window< G, S > >::is_centered ( ) const [inherited]`

Is the window centered?

**10.200.3.3** `bool mln::graph_window_base< S::fun_t::result , graph_elt_window< G, S > >::is_empty ( ) const [inherited]`

Interface of the concept Window.

Is the window is empty?

**10.200.3.4** `bool mln::graph_window_base< S::fun_t::result , graph_elt_window< G, S > >::is_symmetric ( ) const [inherited]`

Is the window symmetric?

**10.200.3.5** `bool mln::graph_window_base< S::fun_t::result , graph_elt_window< G, S > >::is_valid ( ) const [inherited]`

Return true by default.

**10.200.3.6** `self_& mln::graph_window_base< S::fun_t::result , graph_elt_window< G, S > >::sym ( ) [inherited]`

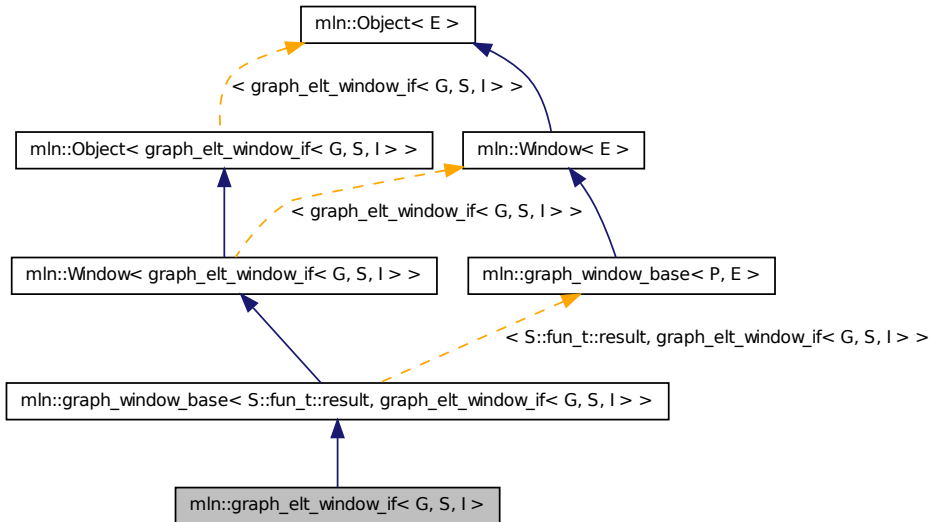
Apply a central symmetry to the target window.

## 10.201 mln::graph\_elt\_window\_if< G, S, I > Class Template Reference

Custom window on graph class.

```
#include <graph_elt_window_if.hh>
```

Inheritance diagram for `mln::graph_elt_window_if< G, S, I >`:



## Public Types

- typedef I [mask\\_t](#)

*The type of the image used as mask.*

- typedef S [target](#)

@}

- typedef `target::psite` [psite](#)

*The type of psite corresponding to the window.*

- typedef `graph_window_if_piter< target, self_, nbh_fwd_iter_ >` [fwd\\_qiter](#)

*Site\_Iterator type to browse the psites of the window w.r.t.*

- typedef `graph_window_if_piter< target, self_, nbh_bkd_iter_ >` [bkd\\_qiter](#)

*Site\_Iterator type to browse the psites of the window w.r.t.*

- typedef `fwd_qiter` [qiter](#)

*The default qiter type.*

- typedef `S::fun_t::result` [site](#)

*Associated types.*

## Public Member Functions

- void [change\\_mask](#) (const [Image](#)< I > &mask)  
*Change mask image.*
- [graph\\_elt\\_window\\_if](#) ()  
*Constructor.*
- [graph\\_elt\\_window\\_if](#) (const [Image](#)< I > &mask)
- bool [is\\_valid](#) () const  
*Return true by default.*
- const I & [mask](#) () const  
*Return the graph image used as mask.*
- bool [is\\_empty](#) () const  
*Interface of the concept Window.*
- bool [is\\_centered](#) () const  
*Is the window centered?*
- bool [is\\_symmetric](#) () const  
*Is the window symmetric?*
- unsigned [delta](#) () const  
*Return the maximum coordinate gap between the window center and a window point.*
- [self\\_ & sym](#) ()  
*Apply a central symmetry to the target window.*

### 10.201.1 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.

### 10.201.2 Member Typedef Documentation

**10.201.2.1** `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.

**10.201.2.2** `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.

**10.201.2.3** `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.

**10.201.2.4** `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.

**10.201.2.5** `template<typename G , typename S , typename I > typedef fwd_qiter mln::graph_elt_window_if< G, S, I >::qiter`

The default qiter type.

**10.201.2.6** `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.

**10.201.2.7** `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.

### 10.201.3 Constructor & Destructor Documentation

**10.201.3.1** `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.

**10.201.3.2** `template<typename G , typename S , typename I > mln::graph_elt_window_if< G, S, I >::graph_elt_window_if( const Image< I > & mask ) [inline]`

#### Parameters

[in] *mask* A graph image of bool.

#### See also

[vertex\\_image](#), [edge\\_image](#).

### 10.201.4 Member Function Documentation

**10.201.4.1** `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()`.

**10.201.4.2** `unsigned mln::graph_window_base< S::fun_t::result , graph_elt_window_if< G, S, I > >::delta( ) const [inherited]`

Return the maximum coordinate gap between the window center and a window point.

**10.201.4.3** `bool mln::graph_window_base< S::fun_t::result , graph_elt_window_if< G, S, I > >::is_centered( ) const [inherited]`

Is the window centered?

**10.201.4.4** `bool mln::graph_window_base< S::fun_t::result , graph_elt_window_if< G, S, I > >::is_empty( ) const [inherited]`

Interface of the concept Window.

Is the window is empty?

**10.201.4.5** `bool mln::graph_window_base< S::fun_t::result , graph_elt_window_if< G, S, I > >::is_symmetric( ) const [inherited]`

Is the window symmetric?

**10.201.4.6** `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()`.

#### 10.201.4.7 `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.

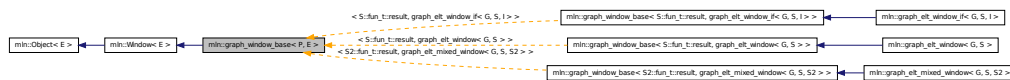
#### 10.201.4.8 `self_ & mln::graph_window_base< S::fun_t::result , graph_elt_window_if< G, S, I >::sym ( ) [inherited]`

Apply a central symmetry to the target window.

## 10.202 `mln::graph_window_base< P, E >` Class Template Reference

```
#include <graph_window_base.hh>
```

Inheritance diagram for `mln::graph_window_base< P, E >`:



### Public Types

- typedef `P` [site](#)  
*Associated types.*

### Public Member Functions

- bool `is_valid ( ) const`  
*Return true by default.*
- bool `is_empty ( ) const`  
*Interface of the concept [Window](#).*
- bool `is_centered ( ) const`  
*Is the window centered?*
- bool `is_symmetric ( ) const`  
*Is the window symmetric?*
- unsigned `delta ( ) const`  
*Return the maximum coordinate gap between the window center and a window point.*
- `self_ & sym ( )`  
*Apply a central symmetry to the target window.*

### 10.202.1 Detailed Description

`template<typename P, typename E> class mln::graph_window_base< P, E >`

#### Template Parameters

*P* Site type.

### 10.202.2 Member Typedef Documentation

**10.202.2.1** `template<typename P, typename E> typedef P mln::graph_window_base< P, E >::site`

Associated types.

The type of site corresponding to the window.

### 10.202.3 Member Function Documentation

**10.202.3.1** `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.

**10.202.3.2** `template<typename P , typename E > bool mln::graph_window_base< P, E >::is_centered ( ) const [inline]`

Is the window centered?

**10.202.3.3** `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?

**10.202.3.4** `template<typename P , typename E > bool mln::graph_window_base< P, E >::is_symmetric ( ) const [inline]`

Is the window symmetric?

**10.202.3.5** `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 >](#).

**10.202.3.6** `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.

## 10.203 mln::graph\_window\_if\_piter< S, W, I > Class Template Reference

Forward iterator on line graph window.

```
#include <graph_window_if_piter.hh>
```

Inherits `site_relative_iterator_base< W, graph_window_if_piter< S, W, I > >`, and `is_masked_impl_selector< S, W::mask_t::domain_t, graph_window_if_piter< S, W, I > >`.

### Public Types

- typedef `S::fun_t::result` `P`  
*Associated types.*

### Public Member Functions

- void `next ()`  
*Go to the next element.*
- `graph_window_if_piter ()`  
*Construction.*
- const `S::graph_element & element () const`  
*Return the graph element pointed by this iterator.*
- unsigned `id () const`  
*Return the graph element id.*

#### 10.203.1 Detailed Description

```
template<typename S, typename W, typename I> class mln::graph_window_if_piter< S, W, I >
```

Forward iterator on line graph window.

#### 10.203.2 Member Typedef Documentation

**10.203.2.1** `template<typename S, typename W, typename I > typedef S::fun_t ::result mln::graph_window_if_piter< S, W, I >::P`

Associated types.



### 10.203.3 Constructor & Destructor Documentation

**10.203.3.1** `template<typename S, typename W, typename I > mln::graph_window_if_piter< S, W, I >::graph_window_if_piter ( ) [inline]`

Construction.

### 10.203.4 Member Function Documentation

**10.203.4.1** `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.

**10.203.4.2** `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.

**10.203.4.3** `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-define this method. The actual "next" operation has to be defined through the *next\_* method.

#### Precondition

The iterator is valid.

## 10.204 mln::graph\_window\_piter< S, W, I > Class Template Reference

Forward iterator on line graph window.

```
#include <graph_window_piter.hh>
```

Inherits `site_relative_iterator_base< W, graph_window_piter< S, W, I >, W::center_t >`, and `impl_selector< W::center_t, W::psite, graph_window_piter< S, W, I > >`.

### Public Types

- `typedef S::fun_t::result P`  
Associated types  
Type of the window elements.

- typedef `W::center_t` `center_t`  
*Type of the window center.*
- typedef `W::graph_element` `graph_element`  
*Type of the graph element pointed by this iterator.*

## Public Member Functions

- void `change_target_site_set` (const `S` &`s`)  
*Change the target site set.*
- void `next` ()  
*Go to the next element.*
- const `S` & `target_site_set` () const  
*Return the target site set.*
- `graph_window_piter` ()  
*Construction.*
- template<typename `Pref` >  
`graph_window_piter` (const `Window`< `W` > &`win`, const `Pref` &`p_ref`)  
*To be used in case the center and neighbor sites have the same type and belong to the same site set.*
- template<typename `Pref` >  
`graph_window_piter` (const `Window`< `W` > &`win`, const `Site_Set`< `S` > &`target_site_set`, const `Pref` &`p_ref`)  
*To be used in case center and neighbors sites do not have the same type and do not belong to the same site set.*
- const `graph_element` & `element` () const  
*Return the graph element pointed by this iterator.*
- unsigned `id` () const  
*Return the graph element id.*

### 10.204.1 Detailed Description

`template<typename S, typename W, typename I> class mln::graph_window_piter< S, W, I >`

Forward iterator on line graph window.

#### Template Parameters

*S* is the site set type.

*W* is the window type.

*I* is the underlying iterator type.

## 10.204.2 Member Typedef Documentation

**10.204.2.1** `template<typename S , typename W , typename I > typedef W::center_t  
mln::graph_window_piter< S, W, I >::center_t`

Type of the window center.

**10.204.2.2** `template<typename S , typename W , typename I > typedef W::graph_element  
mln::graph_window_piter< S, W, I >::graph_element`

Type of the graph element pointed by this iterator.

**10.204.2.3** `template<typename S , typename W , typename I > typedef S::fun_t ::result  
mln::graph_window_piter< S, W, I >::P`

Associated types

Type of the window elements.

## 10.204.3 Constructor & Destructor Documentation

**10.204.3.1** `template<typename S , typename W , typename I > mln::graph_window_piter< S, W,  
I >::graph_window_piter ( ) [inline]`

Construction.

**10.204.3.2** `template<typename S , typename W , typename I > template<typename Pref >  
mln::graph_window_piter< S, W, I >::graph_window_piter ( const Window< W > &  
win, const Pref & p_ref ) [inline]`

To be used in case the center and neighbor sites have the same type and belong to the same site set.

### Parameters

*win* The underlying window.

*p\_ref* [Window](#) center.

**10.204.3.3** `template<typename S , typename W , typename I > template<typename Pref >  
mln::graph_window_piter< S, W, I >::graph_window_piter ( const Window< W > &  
win, const Site_Set< S > & target_site_set, const Pref & p_ref ) [inline]`

To be used in case center and neighbors sites do not have the same type and do not belong to the same site set.

### Parameters

*win* The underlying window.

*target\_site\_set* [Site](#) set in which neighbor sites are extracted.

*p\_ref* [Window](#) center.

### 10.204.4 Member Function Documentation

**10.204.4.1** `template<typename S, typename W, typename I> void mln::graph_window_piter< S, W, I>::change_target_site_set ( const S & s ) [inline]`

Change the target site set.

[Window](#) elements different from the center come from the target site set.

**10.204.4.2** `template<typename S, typename W, typename I> const graph_window_piter< S, W, I>::graph_element & mln::graph_window_piter< S, W, I>::element ( ) const [inline]`

Return the graph element pointed by this iterator.

**10.204.4.3** `template<typename S, typename W, typename I> unsigned mln::graph_window_piter< S, W, I>::id ( ) const [inline]`

Return the graph element id.

FIXME: we do not want to have this member since there is an automatic conversion to the graph element. C++ does not seem to use this conversion operator.

**10.204.4.4** `void mln::Site_Iterator< graph_window_piter< S, W, I> >::next ( ) [inherited]`

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

#### Precondition

The iterator is valid.

**10.204.4.5** `template<typename S, typename W, typename I> const S & mln::graph_window_piter< S, W, I>::target_site_set ( ) const [inline]`

Return the target site set.

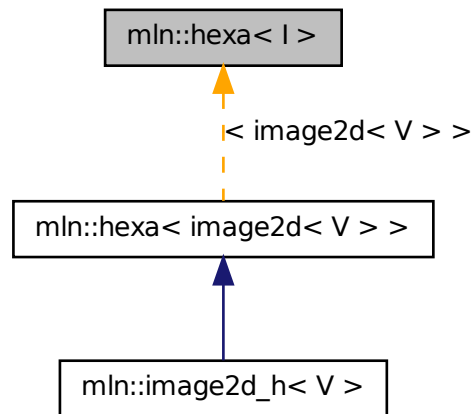
[Window](#) elements different from the center come from the target site set.

## 10.205 mln::hexa< I > Struct Template Reference

hexagonal image class.

```
#include <hexa.hh>
```

Inheritance diagram for mln::hexa< I >:



## Public Types

- typedef hexa\_bkd\_piter\_< box2d > bkd\_piter  
*FIXME : should it be in box2d\_h? Backward Site\_Iterator associated type.*
- typedef hexa\_fwd\_piter\_< box2d > fwd\_piter  
*FIXME : should it be in box2d\_h? Forward Site\_Iterator associated type.*
- typedef I::lvalue lvalue  
*Lvalue associated type.*
- typedef point2d\_h psite  
*Point site type.*
- typedef I::rvalue rvalue  
*Return type of read-only access.*
- typedef hexa< tag::image\_< I > > skeleton  
*Skeleton.*
- typedef I::value value  
*Value associated type.*

## Public Member Functions

- const box2d\_h & domain () const

*Give the definition domain.*

- `bool has (const psite &p) const`  
*Test if  $p$  belongs to the image domain.*
- `hexa (I &ima)`  
*Constructor with an base image.*
- `hexa ()`  
*Constructor without argument.*
- `lvalue operator() (const point2d\_h &p)`  
*Read-write access of pixel value at hexa point site  $p$ .*
- `rvalue operator() (const point2d\_h &p) const`  
*Read-only access of pixel value at hexa point site  $p$ .*

### 10.205.1 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 -----
```

### 10.205.2 Member Typedef Documentation

**10.205.2.1 `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.

**10.205.2.2 `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.

**10.205.2.3 `template<typename I> typedef I ::lvalue mln::hexa< I >::lvalue`**

Lvalue associated type.

**10.205.2.4 `template<typename I> typedef point2d_h mln::hexa< I >::psite`**

[Point](#) site type.

Reimplemented in `mln::image2d_h< V >`.

**10.205.2.5** `template<typename I> typedef I::rvalue mln::hexa< I >::rvalue`

Return type of read-only access.

**10.205.2.6** `template<typename I> typedef hexa< tag::image_<I> > mln::hexa< I >::skeleton`

Skeleton.

**10.205.2.7** `template<typename I> typedef I::value mln::hexa< I >::value`

Value associated type.

**10.205.3** **Constructor & Destructor Documentation****10.205.3.1** `template<typename I> mln::hexa< I >::hexa ( ) [inline]`

Constructor without argument.

**10.205.3.2** `template<typename I> mln::hexa< I >::hexa ( I & ima ) [inline]`

Constructor with an base image.

**10.205.4** **Member Function Documentation****10.205.4.1** `template<typename I> const box2d_h & mln::hexa< I >::domain ( ) const [inline]`

Give the definition domain.

**10.205.4.2** `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().

**10.205.4.3** `template<typename I> hexa< I >::rvalue mln::hexa< I >::operator() ( const point2d_h & p ) const [inline]`

Read-only access of pixel value at hexa point site p.

References mln::hexa< I >::has().

**10.205.4.4** `template<typename I> hexa< I >::lvalue mln::hexa< I >::operator() ( const point2d_h & p ) [inline]`

Read-write access of pixel value at hexa point site p.

References mln::hexa< I >::has().

## 10.206 mln::histo::array< T > Struct Template Reference

Generic histogram class over a value set with type T.

```
#include <array.hh>
```

### 10.206.1 Detailed Description

```
template<typename T> struct mln::histo::array< T >
```

Generic histogram class over a value set with type T.

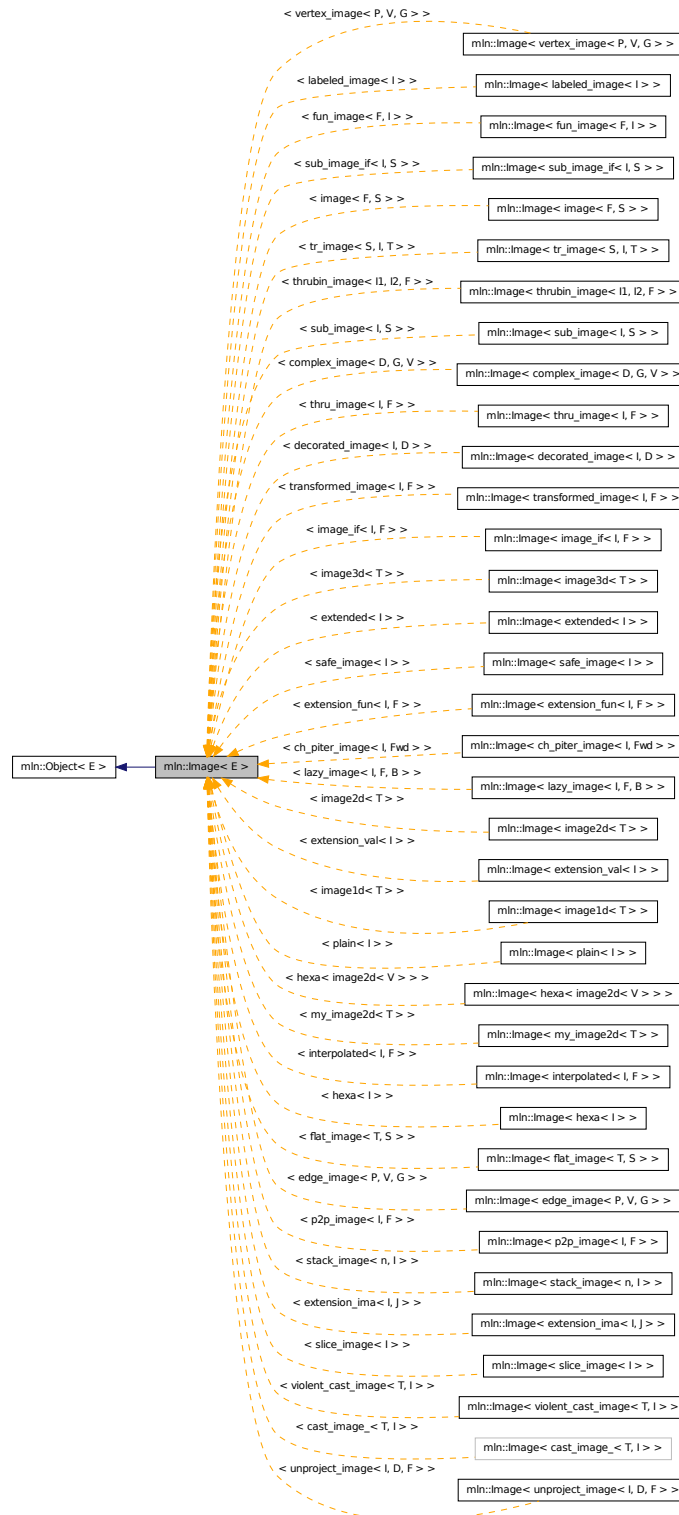
## 10.207 mln::Image< E > Struct Template Reference

Base class for implementation of image classes.

```
#include <image.hh>
```



Inheritance diagram for mln::Image< E >:



### 10.207.1 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.

## 10.208 mln::image1d< T > Struct Template Reference

Basic 1D image class.

```
#include <image1d.hh>
```

Inherits `image_primary< T, box1d, image1d< T > >`.

### Public Types

- typedef T & [lvalue](#)  
*Return type of read-write access.*
- typedef const T & [rvalue](#)  
*Return type of read-only access.*
- typedef [image1d](#)< tag::value\_< T > > [skeleton](#)  
*Skeleton.*
- typedef T [value](#)  
*Value associated type.*

### Public Member Functions

- const [box1d](#) & [bbox](#) () const  
*Give the bounding box domain.*
- unsigned [border](#) () const  
*Give the border thickness.*
- const T \* [buffer](#) () const  
*Give a hook to the value buffer.*
- T \* [buffer](#) ()  
*Give a hook to the value buffer.*
- int [delta\\_index](#) (const [dpoint1d](#) &dp) const  
*Give the offset corresponding to the delta-point 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` (unsigned ninds, unsigned bdr=border::thickness)  
*Constructor with the number of indices and the border thickness.*
- `image1d` ()  
*Constructor without argument.*
- `image1d` (const `box1d` &b, unsigned bdr=border::thickness)  
*Constructor with a box and the border thickness.*
- unsigned `nelements` () const  
*Give the number of cells (points including border ones).*
- unsigned `ninds` () const  
*Give the number of indexes.*
- const T & `operator()` (const `point1d` &p) const  
*Read-only access to the image value located at point  $p$ .*
- T & `operator()` (const `point1d` &p)  
*Read-write access to the image value located at point  $p$ .*
- `point1d` `point_at_index` (unsigned i) const  
*Give the point corresponding to the offset  $o$ .*

### 10.208.1 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.

### 10.208.2 Member Typedef Documentation

**10.208.2.1** `template<typename T> typedef T& mln::image1d< T >::lvalue`

Return type of read-write access.

**10.208.2.2** `template<typename T> typedef const T& mln::image1d< T >::rvalue`

Return type of read-only access.

**10.208.2.3** `template<typename T> typedef image1d< tag::value_<T> > mln::image1d< T >::skeleton`

Skeleton.

**10.208.2.4** `template<typename T> typedef T mln::image1d< T >::value`

[Value](#) associated type.

**10.208.3 Constructor & Destructor Documentation****10.208.3.1** `template<typename T> mln::image1d< T >::image1d ( ) [inline]`

Constructor without argument.

**10.208.3.2** `template<typename T> mln::image1d< T >::image1d ( unsigned ninds, unsigned bdr = border::thickness ) [inline]`

Constructor with the number of indices and the border thickness.

References `mln::make::box1d()`.

**10.208.3.3** `template<typename T> mln::image1d< T >::image1d ( const box1d & b, unsigned bdr = border::thickness ) [inline]`

Constructor with a box and the border thickness.

**10.208.4 Member Function Documentation****10.208.4.1** `template<typename T> const box1d & mln::image1d< T >::bbox ( ) const [inline]`

Give the bounding box domain.

**10.208.4.2** `template<typename T> unsigned mln::image1d< T >::border ( ) const [inline]`

Give the border thickness.

**10.208.4.3** `template<typename T> const T * mln::image1d< T >::buffer ( ) const [inline]`

Give a hook to the value buffer.

**10.208.4.4** `template<typename T> T * mln::image1d< T >::buffer ( ) [inline]`

Give a hook to the value buffer.

**10.208.4.5** `template<typename T> int mln::image1d< T >::delta_index ( const dpoint1d & dp ) const [inline]`

Give the offset corresponding to the delta-point *dp*.

**10.208.4.6** `template<typename T> const box1d & mln::image1d< T >::domain ( ) const [inline]`

Give the definition domain.

**10.208.4.7** `template<typename T> T & mln::image1d< T >::element ( unsigned i ) [inline]`

Read-write access to the *i*-th image value (including the border).

References `mln::image1d< T >::nelements()`.

**10.208.4.8** `template<typename T> const T & mln::image1d< T >::element ( unsigned i ) const [inline]`

Read-only access to the *i*-th image value (including the border).

References `mln::image1d< T >::nelements()`.

**10.208.4.9** `template<typename T> bool mln::image1d< T >::has ( const point1d & p ) const [inline]`

Test if *p* is valid.

Referenced by `mln::image1d< T >::operator()()`.

**10.208.4.10** `template<typename T> unsigned mln::image1d< T >::nelements ( ) const [inline]`

Give the number of cells (points including border ones).

Referenced by `mln::image1d< T >::element()`, and `mln::image1d< T >::point_at_index()`.

**10.208.4.11** `template<typename T> unsigned mln::image1d< T >::ninds ( ) const [inline]`

Give the number of indexes.

**10.208.4.12** `template<typename T> T & mln::image1d< T >::operator() ( const point1d & p ) [inline]`

Read-write access to the image value located at point *p*.

References `mln::image1d< T >::has()`.

**10.208.4.13** `template<typename T > const T & mln::image1d< T >::operator() ( const point1d & p ) const [inline]`

Read-only access to the image value located at point `p`.

References `mln::image1d< T >::has()`.

**10.208.4.14** `template<typename T > point1d mln::image1d< T >::point_at_index ( unsigned i ) const [inline]`

Give the point corresponding to the offset `o`.

References `mln::image1d< T >::nelements()`.

## 10.209 mln::image2d< T > Class Template Reference

Basic 2D image class.

```
#include <image2d.hh>
```

Inherits `image_primary< T, mln::box2d, image2d< T > >`.

### Public Types

- typedef `T & lvalue`  
*Return type of read-write access.*
- typedef `const T & rvalue`  
*Return type of read-only access.*
- typedef `image2d< tag::value_< T > > skeleton`  
*Skeleton.*
- typedef `T value`  
*Value associated type.*

### Public Member Functions

- const `box2d & bbox () const`  
*Give the bounding box domain.*
- unsigned `border () const`  
*Give the border thickness.*
- const `T * buffer () const`  
*Give a hook to the value buffer.*

- `T * buffer ()`  
*Give a hook to the value `buffer`.*
- `int delta_index (const dpoint2d &dp) const`  
*Give the delta-index corresponding to the delta-point `dp`.*
- `const box2d & domain () const`  
*Give the definition domain.*
- `T & element (unsigned i)`  
*Read-write access to the image value located at index `i`.*
- `const T & element (unsigned i) const`  
*Read-only access to the image value located at index `i`.*
- `bool has (const point2d &p) const`  
*Test if `p` is valid.*
- `image2d ()`  
*Constructor without argument.*
- `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.*
- `unsigned ncols () const`  
*Give the number of columns.*
- `unsigned nelements () const`  
*Give the number of elements (points including border ones).*
- `unsigned nrows () const`  
*Give the number of rows.*
- `const T & operator() (const point2d &p) const`  
*Read-only access to the image value located at point `p`.*
- `T & operator() (const point2d &p)`  
*Read-write access to the image value located at point `p`.*
- `point2d point_at_index (unsigned i) const`  
*Give the point corresponding to the index `i`.*

### 10.209.1 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.

### 10.209.2 Member Typedef Documentation

**10.209.2.1** `template<typename T> typedef T& mln::image2d< T >::lvalue`

Return type of read-write access.

**10.209.2.2** `template<typename T> typedef const T& mln::image2d< T >::rvalue`

Return type of read-only access.

**10.209.2.3** `template<typename T> typedef image2d< tag::value_<T> > mln::image2d< T >::skeleton`

Skeleton.

**10.209.2.4** `template<typename T> typedef T mln::image2d< T >::value`

[Value](#) associated type.

### 10.209.3 Constructor & Destructor Documentation

**10.209.3.1** `template<typename T > mln::image2d< T >::image2d ( ) [inline]`

Constructor without argument.

**10.209.3.2** `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()`.

**10.209.3.3** `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).



## 10.209.4 Member Function Documentation

**10.209.4.1** `template<typename T> const box2d & mln::image2d< T >::bbox ( ) const`  
`[inline]`

Give the bounding box domain.

**10.209.4.2** `template<typename T> unsigned mln::image2d< T >::border ( ) const` `[inline]`

Give the border thickness.

**10.209.4.3** `template<typename T> T * mln::image2d< T >::buffer ( )` `[inline]`

Give a hook to the value buffer.

**10.209.4.4** `template<typename T> const T * mln::image2d< T >::buffer ( ) const` `[inline]`

Give a hook to the value buffer.

**10.209.4.5** `template<typename T> int mln::image2d< T >::delta_index ( const dpoint2d & dp )`  
`const [inline]`

Give the delta-index corresponding to the delta-point dp.

**10.209.4.6** `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().

**10.209.4.7** `template<typename T> const T & mln::image2d< T >::element ( unsigned i ) const`  
`[inline]`

Read-only access to the image value located at index i.

References mln::image2d< T >::nelements().

**10.209.4.8** `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().

**10.209.4.9** `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()`.

**10.209.4.10** `template<typename T> unsigned mln::image2d< T >::ncols ( ) const [inline]`

Give the number of columns.

**10.209.4.11** `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()`.

**10.209.4.12** `template<typename T> unsigned mln::image2d< T >::nrows ( ) const [inline]`

Give the number of rows.

**10.209.4.13** `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()`.

**10.209.4.14** `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()`.

**10.209.4.15** `template<typename T> point2d mln::image2d< T >::point_at_index ( unsigned i ) const [inline]`

Give the point corresponding to the index `i`.

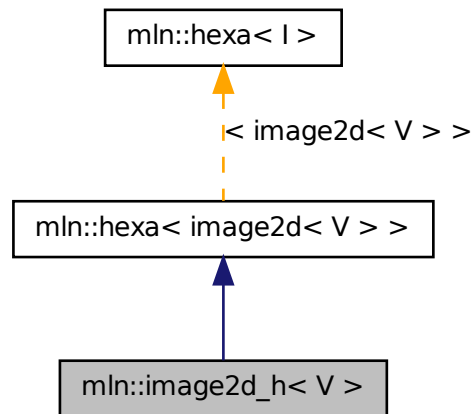
References `mln::image2d< T >::nelements()`.

## 10.210 mln::image2d\_h< V > Struct Template Reference

2d image based on an hexagonal mesh.

```
#include <image2d_h.hh>
```

Inheritance diagram for mln::image2d\_h< V >:



## Public Types

- typedef `hexa_bkd_piter_< box2d > bkd_piter`  
*FIXME : should it be in box2d\_h? Backward Site\_Iterator associated type.*
- typedef `hexa_fwd_piter_< box2d > fwd_piter`  
*FIXME : should it be in box2d\_h? Forward Site\_Iterator associated type.*
- typedef `image2d< V >::lvalue lvalue`  
*Lvalue associated type.*
- typedef `point2d_h psite`  
*Point site type.*
- typedef `image2d< V >::rvalue rvalue`  
*Return type of read-only access.*
- typedef `hexa< tag::image_< image2d< V > > > skeleton`  
*Skeleton.*
- typedef `image2d< V >::value value`  
*Value associated type.*

## Public Member Functions

- const `box2d_h & domain () const`

*Give the definition domain.*

- `bool has (const psite &p) const`  
*Test if  $p$  belongs to the image domain.*
- `image2d_h (int nrows, int ncols, unsigned bdr=border::thickness)`  
*Constructor with the numbers of rows and columns border thickness.*
- `lvalue operator() (const point2d\_h &p)`  
*Read-write access of pixel value at hexa point site  $p$ .*
- `rvalue operator() (const point2d\_h &p) const`  
*Read-only access of pixel value at hexa point site  $p$ .*

### 10.210.1 Detailed Description

```
template<typename V> struct mln::image2d_h< V >
```

2d image based on an hexagonal mesh.

### 10.210.2 Member Typedef Documentation

**10.210.2.1** `typedef hexa_bkd_piter_<box2d> mln::hexa< image2d< V > >::bkd_piter`  
`[inherited]`

FIXME : should it be in box2d\_h? Backward Site\_Iterator associated type.

**10.210.2.2** `typedef hexa_fwd_piter_<box2d> mln::hexa< image2d< V > >::fwd_piter`  
`[inherited]`

FIXME : should it be in box2d\_h? Forward Site\_Iterator associated type.

**10.210.2.3** `typedef image2d< V > ::lvalue mln::hexa< image2d< V > >::lvalue` `[inherited]`

Lvalue associated type.

**10.210.2.4** `template<typename V > typedef point2d_h mln::image2d_h< V >::psite`

[Point](#) site type.

Reimplemented from `mln::hexa< image2d< V > >`.

**10.210.2.5** `typedef image2d< V > ::rvalue mln::hexa< image2d< V > >::rvalue` `[inherited]`

Return type of read-only access.

**10.210.2.6** `typedef hexa< tag::image_<image2d< V > > > mln::hexa< image2d< V > >::skeleton [inherited]`

Skeleton.

**10.210.2.7** `typedef image2d< V > ::value mln::hexa< image2d< V > >::value [inherited]`

Value associated type.

### 10.210.3 Constructor & Destructor Documentation

**10.210.3.1** `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
```

### 10.210.4 Member Function Documentation

**10.210.4.1** `const box2d_h& mln::hexa< image2d< V > >::domain ( ) const [inherited]`

Give the definition domain.

**10.210.4.2** `bool mln::hexa< image2d< V > >::has ( const psite & p ) const [inherited]`

Test if *p* belongs to the image domain.

**10.210.4.3** `rvalue mln::hexa< image2d< V > >::operator() ( const point2d_h & p ) const [inherited]`

Read-only access of pixel value at hexa point site *p*.

**10.210.4.4** `lvalue mln::hexa< image2d< V > >::operator() ( const point2d_h & p ) [inherited]`

Read-write access of pixel value at hexa point site *p*.

## 10.211 mln::image3d< T > Struct Template Reference

Basic 3D image class.

```
#include <image3d.hh>
```

Inherits image\_primary< T, box3d, image3d< T > >.

## Public Types

- typedef T & **lvalue**  
*Return type of read-write access.*
- typedef const T & **rvalue**  
*Return type of read-only access.*
- typedef **image3d**< tag::value\_< T > > **skeleton**  
*Skeleton.*
- typedef T **value**  
*Value associated type.*

## Public Member Functions

- const **box3d** & **bbox** () const  
*Give the bounding box domain.*
- unsigned **border** () const  
*Give the border thickness.*
- const T \* **buffer** () const  
*Give a hook to the value buffer.*
- T \* **buffer** ()  
*Give a hook to the value buffer.*
- int **delta\_index** (const **dpoint3d** &dp) const  
*Fast **Image** method.*
- const **box3d** & **domain** () const  
*Give the definition domain.*
- 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** (const **box3d** &b, unsigned bdr=border::thickness)  
*Constructor with a box and the border thickness (default is 3).*
- **image3d** ()  
*Constructor without argument.*

- `image3d` (int nslices, int nrows, int ncols, unsigned bdr=border::thickness)  
*Constructor with the numbers of indexes and the border thickness.*
- unsigned `ncols` () const  
*Give the number of columns.*
- unsigned `nelements` () const  
*Give the number of cells (points including border ones).*
- unsigned `nrows` () const  
*Give the number of rows.*
- unsigned `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.*

### 10.211.1 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.

### 10.211.2 Member Typedef Documentation

**10.211.2.1 template<typename T> typedef T& mln::image3d< T >::lvalue**

Return type of read-write access.

**10.211.2.2 template<typename T> typedef const T& mln::image3d< T >::rvalue**

Return type of read-only access.

**10.211.2.3 template<typename T> typedef image3d< tag::value\_<T> > mln::image3d< T >::skeleton**

Skeleton.

**10.211.2.4** `template<typename T> typedef T mln::image3d< T >::value`

[Value](#) associated type.

**10.211.3** **Constructor & Destructor Documentation****10.211.3.1** `template<typename T> mln::image3d< T >::image3d ( ) [inline]`

Constructor without argument.

**10.211.3.2** `template<typename T> mln::image3d< T >::image3d ( const box3d & b, unsigned bdr = border::thickness ) [inline]`

Constructor with a box and the border thickness (default is 3).

**10.211.3.3** `template<typename T> mln::image3d< T >::image3d ( int nslices, int nrows, int ncols, unsigned bdr = border::thickness ) [inline]`

Constructor with the numbers of indexes and the border thickness.

References `mln::make::box3d()`.

**10.211.4** **Member Function Documentation****10.211.4.1** `template<typename T> const box3d & mln::image3d< T >::bbox ( ) const [inline]`

Give the bounding box domain.

**10.211.4.2** `template<typename T> unsigned mln::image3d< T >::border ( ) const [inline]`

Give the border thickness.

**10.211.4.3** `template<typename T> T * mln::image3d< T >::buffer ( ) [inline]`

Give a hook to the value buffer.

**10.211.4.4** `template<typename T> const T * mln::image3d< T >::buffer ( ) const [inline]`

Give a hook to the value buffer.

**10.211.4.5** `template<typename T> int mln::image3d< T >::delta_index ( const dpoint3d & dp ) const [inline]`

Fast [Image](#) method.

Give the offset corresponding to the delta-point `dp`.



**10.211.4.6** `template<typename T > const box3d & mln::image3d< T >::domain ( ) const [inline]`

Give the definition domain.

**10.211.4.7** `template<typename T > const T & mln::image3d< T >::element ( unsigned i ) const [inline]`

Read-only access to the image value located at index *i*.

References mln::image3d< T >::nelements().

**10.211.4.8** `template<typename T > T & mln::image3d< T >::element ( unsigned i ) [inline]`

Read-write access to the image value located at index *i*.

References mln::image3d< T >::nelements().

**10.211.4.9** `template<typename T > bool mln::image3d< T >::has ( const point3d & p ) const [inline]`

Test if *p* is valid.

Referenced by mln::image3d< T >::operator()().

**10.211.4.10** `template<typename T > unsigned mln::image3d< T >::ncols ( ) const [inline]`

Give the number of columns.

**10.211.4.11** `template<typename T > unsigned mln::image3d< T >::nelements ( ) const [inline]`

Give the number of cells (points including border ones).

Referenced by mln::image3d< T >::element(), and mln::image3d< T >::point\_at\_index().

**10.211.4.12** `template<typename T > unsigned mln::image3d< T >::nrows ( ) const [inline]`

Give the number of rows.

**10.211.4.13** `template<typename T > unsigned mln::image3d< T >::nslices ( ) const [inline]`

Give the number of slices.

**10.211.4.14** `template<typename T > const T & mln::image3d< T >::operator() ( const point3d & p ) const [inline]`

Read-only access to the image value located at point *p*.

References `mln::image3d< T >::has()`.

**10.211.4.15** `template<typename T > T & mln::image3d< T >::operator() ( const point3d & p )`  
**[inline]**

Read-write access to the image value located at point `p`.

References `mln::image3d< T >::has()`.

**10.211.4.16** `template<typename T > point3d mln::image3d< T >::point_at_index ( unsigned o ) const`  
**[inline]**

Give the point corresponding to the offset `o`.

References `mln::image3d< T >::nelements()`.

## 10.212 mln::image\_if< I, F > Struct Template Reference

[Image](#) which domain is restricted by a function 'site -> Boolean'.

```
#include <image_if.hh>
```

Inherits `image_domain_morpher< I, p_if< I::domain_t, F >, image_if< I, F > >`.

### Public Types

- typedef [image\\_if< tag::image\\_< I >, tag::function\\_< F > >](#) [skeleton](#)  
*Skeleton.*

### Public Member Functions

- const [p\\_if< typename I::domain\\_t, F > & domain](#) () const  
*Give the definition domain.*
- [image\\_if](#) (I &ima, const F &f)  
*Constructor from an image *ima* and a predicate *f*.*
- [image\\_if](#) ()  
*Constructor without argument.*
- [operator image\\_if< const I, F > \(\) const](#)  
*Const promotion via conversion.*

### 10.212.1 Detailed Description

```
template<typename I, typename F> struct mln::image_if< I, F >
```

[Image](#) which domain is restricted by a function 'site -> Boolean'.

## 10.212.2 Member Typedef Documentation

**10.212.2.1** `template<typename I, typename F> typedef image_if< tag::image_<I>, tag::function_<F> > mln::image_if< I, F >::skeleton`

Skeleton.

## 10.212.3 Constructor & Destructor Documentation

**10.212.3.1** `template<typename I, typename F > mln::image_if< I, F >::image_if ( )`  
`[inline]`

Constructor without argument.

**10.212.3.2** `template<typename I, typename F > mln::image_if< I, F >::image_if ( I & ima, const F & f )` `[inline]`

Constructor from an image `ima` and a predicate `f`.

## 10.212.4 Member Function Documentation

**10.212.4.1** `template<typename I, typename F > const p_if< typename I::domain_t, F > & mln::image_if< I, F >::domain ( ) const` `[inline]`

Give the definition domain.

**10.212.4.2** `template<typename I, typename F > mln::image_if< I, F >::operator image_if< const I, F > ( ) const` `[inline]`

Const promotion via conversion.

## 10.213 mln::interpolated< I, F > Struct Template Reference

Makes the underlying image being accessed with floating coordinates.

```
#include <interpolated.hh>
```

Inherits `image_identity< I, I::domain_t, interpolated< I, F > >`.

### Public Types

- typedef `I::lvalue` `lvalue`  
*Return type of read-write access.*
- typedef `I::psite` `psite`  
*Point\_Site associated type.*
- typedef `I::rvalue` `rvalue`

*Return type of read-only access.*

- typedef [interpolated](#)< tag::image\_< I >, F > [skeleton](#)  
*Skeleton.*
- typedef I::value [value](#)  
*Value associated type.*

## Public Member Functions

- template<typename C >  
bool [has](#) (const mln::algebra::vec< I::psite::dim, C > &v) const  
*Test if a pixel value is accessible at v.*
- [interpolated](#) (I &ima)  
*Constructors.*
- bool [is\\_valid](#) () const  
*Test if this image has been initialized.*

### 10.213.1 Detailed Description

**template<typename I, template< class > class F> struct mln::interpolated< I, F >**

Makes the underlying image being accessed with floating coordinates.

### 10.213.2 Member Typedef Documentation

**10.213.2.1 template<typename I, template< class > class F> typedef I ::lvalue  
mln::interpolated< I, F >::lvalue**

Return type of read-write access.

**10.213.2.2 template<typename I, template< class > class F> typedef I ::psite mln::interpolated<  
I, F >::psite**

[Point\\_Site](#) associated type.

**10.213.2.3 template<typename I, template< class > class F> typedef I ::rvalue  
mln::interpolated< I, F >::rvalue**

Return type of read-only access.

**10.213.2.4 template<typename I, template< class > class F> typedef interpolated<  
tag::image\_<I>, F > mln::interpolated< I, F >::skeleton**

Skeleton.

**10.213.2.5** `template<typename I, template< class > class F> typedef I ::value mln::interpolated< I, F >::value`

Value associated type.

### 10.213.3 Constructor & Destructor Documentation

**10.213.3.1** `template<typename I, template< class > class F> mln::interpolated< I, F >::interpolated( I & ima ) [inline]`

Constructors.

FIXME: don't we want a 'const' here?

### 10.213.4 Member Function Documentation

**10.213.4.1** `template<typename I, template< class > class F> template<typename C > bool mln::interpolated< I, F >::has( const mln::algebra::vec< I::psite::dim, C > & v ) const [inline]`

Test if a pixel value is accessible at v.

**10.213.4.2** `template<typename I, template< class > class F> bool mln::interpolated< I, F >::is_valid( ) const [inline]`

Test if this image has been initialized.

## 10.214 mln::io::dicom::dicom\_header Struct Reference

Store dicom file header.

```
#include <get_header.hh>
```

### 10.214.1 Detailed Description

Store dicom file header.

## 10.215 mln::io::dump::dump\_header Struct Reference

Store dump file header.

```
#include <get_header.hh>
```

### 10.215.1 Detailed Description

Store dump file header.

## 10.216 mln::io::fld::fld\_header Struct Reference

Define the header structure of an AVS field data file.

```
#include <header.hh>
```

### 10.216.1 Detailed Description

Define the header structure of an AVS field data file.

## 10.217 mln::io::raw::raw\_header Struct Reference

Store raw file header.

```
#include <get_header.hh>
```

### 10.217.1 Detailed Description

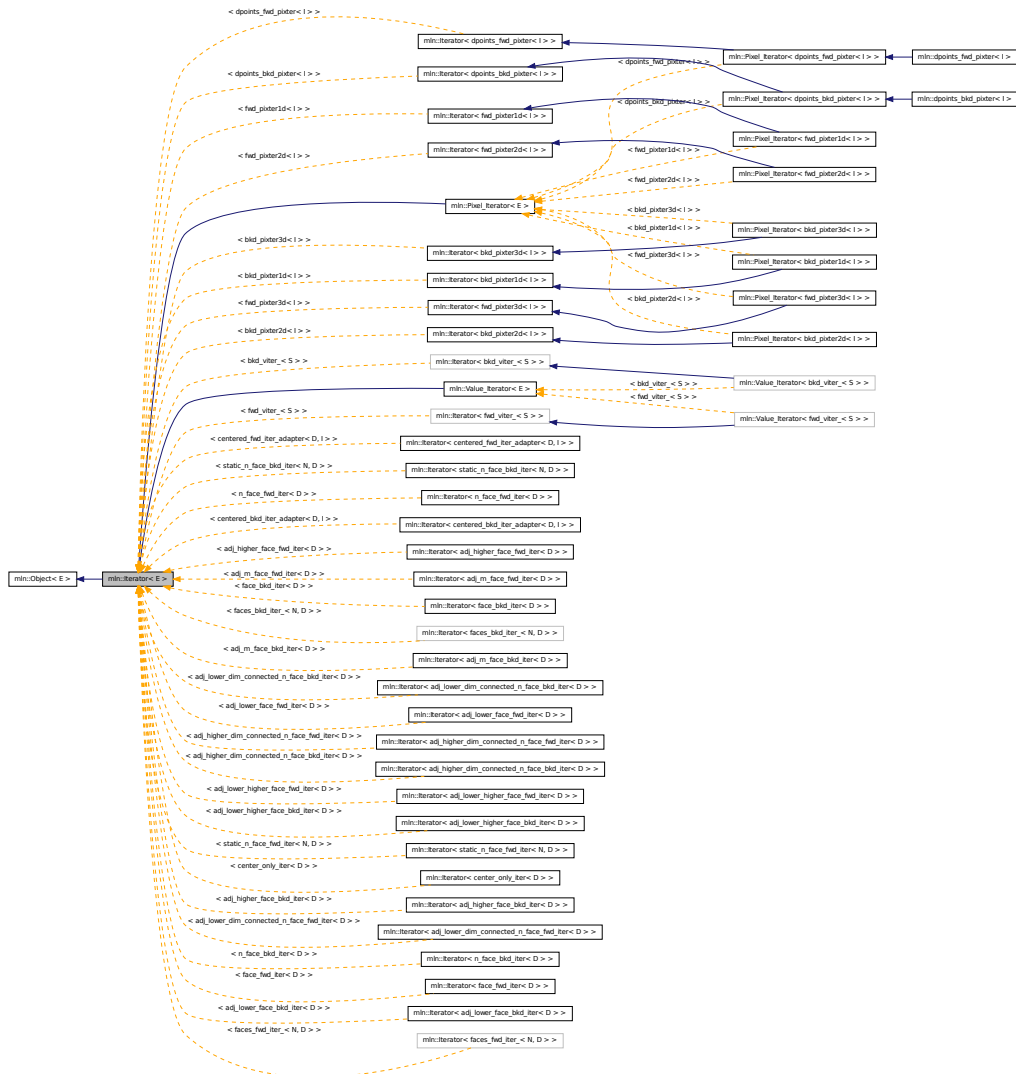
Store raw file header.

## 10.218 mln::Iterator< E > Struct Template Reference

Base class for implementation classes that are iterators.

```
#include <iterator.hh>
```

Inheritance diagram for mln::Iterator< E >:



## Public Member Functions

- void [next](#) ()  
*Go to the next element.*

### 10.218.1 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.

**10.218.2 Member Function Documentation****10.218.2.1** `template<typename E > void mln::Iterator< E >::next ( )`

Go to the next element.

**Warning**

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

**Precondition**

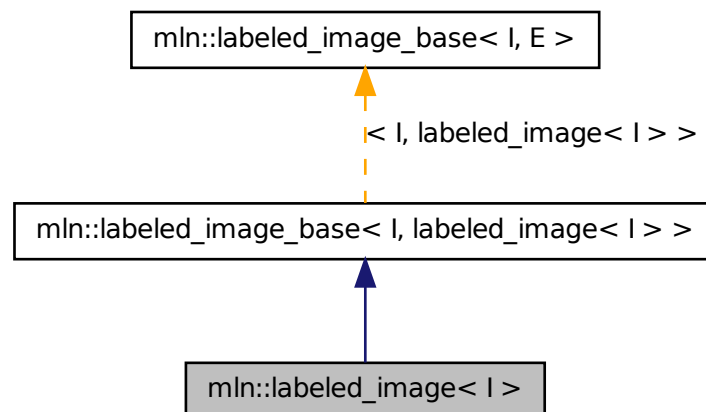
The iterator is valid.

**10.219 mln::labeled\_image< I > Class Template Reference**

Morpher providing an improved interface for labeled image.

```
#include <labeled_image.hh>
```

Inheritance diagram for `mln::labeled_image< I >`:

**Public Types**

- typedef `accu::shape::bbox< typename I::psite >::result` `bbox_t`  
*Type of the bounding component bounding boxes.*



- typedef `labeled_image< tag::image_< I > >` `skeleton`  
*Skeleton.*

## Public Member Functions

- const `bbox_t & bbox` (const typename I::value &label) const  
*Return the bounding box of the component label.*
- const `util::array< bbox_t > & bboxes` () const  
*Return the component bounding boxes.*
- I::value `nlabels` () const  
*Return the number of labels;.*
- `p_if< mln_box(I), fun::eq_v2b_expr< pw::value_< I >, pw::cst_< typename I::value > >` `subdomain` (const typename I::value &label) const  
*Return the domain of the component with label label.*
- `labeled_image` ()  
*Constructors*  
*Constructor without argument.*
- `labeled_image` (const I &ima, const typename I::value &nlabels)  
*Constructor from an image ima and the number of labels nlabels.*
- `labeled_image` (const I &ima, const typename I::value &nlabels, const `util::array< mln_box(I)>` &bboxes)  
*Constructor from an image ima, the number of labels nlabels and the object bounding boxes.*
- void `relabel` (const `Function_v2v< F >` &f)  
*Relabel according to a function.*
- void `relabel` (const `Function_v2b< F >` &f)  
*Labels may be removed.*

## Protected Member Functions

- void `update_data` (const fun::i2v::array< typename I::value > &relabel\_fun)  
*Update bounding boxes information.*

### 10.219.1 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).

## 10.219.2 Member Typedef Documentation

**10.219.2.1** `typedef accu::shape::bbox<typename I ::psite>::result mln::labeled_image_base< I, labeled_image< I > >::bbox_t [inherited]`

Type of the bounding component bounding boxes.

**10.219.2.2** `template<typename I> typedef labeled_image< tag::image_<I> > mln::labeled_image< I >::skeleton`

Skeleton.

## 10.219.3 Constructor & Destructor Documentation

**10.219.3.1** `template<typename I> mln::labeled_image< I >::labeled_image ( ) [inline]`

Constructors

Constructor without argument.

**10.219.3.2** `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`.

**10.219.3.3** `template<typename I> mln::labeled_image< I >::labeled_image ( const I & ima, const typename I::value & nlabels, const util::array< mln_box(I)> & bboxes ) [inline]`

Constructor from an image `ima`, the number of labels `nlabels` and the object bounding boxes.

References `mln::data::compute()`.

## 10.219.4 Member Function Documentation

**10.219.4.1** `const bbox_t& mln::labeled_image_base< I, labeled_image< I > >::bbox ( const typename I::value & label ) const [inherited]`

Return the bounding box of the component `label`.

**10.219.4.2** `const util::array<bbox_t>& mln::labeled_image_base< I, labeled_image< I >>::bboxes ( ) const` `[inherited]`

Return the component bounding boxes.

**10.219.4.3** `I::value mln::labeled_image_base< I, labeled_image< I >>::nlabels ( ) const` `[inherited]`

Return the number of labels;

**10.219.4.4** `void mln::labeled_image_base< I, labeled_image< I >>::relabel ( const Function_v2b< F > & f )` `[inherited]`

Labels may be removed.

This overload make sure the labeling is still contiguous.

**10.219.4.5** `void mln::labeled_image_base< I, labeled_image< I >>::relabel ( const Function_v2v< F > & f )` `[inherited]`

Relabel according to a function.

Merge or delete labels according to the given function. This method ensures that the labeling remains contiguous.

**10.219.4.6** `p_if<mln_box(I), fun::eq_v2b_expr_<pw::value_<I>, pw::cst_<typename I::value>>> mln::labeled_image_base< I, labeled_image< I >>::subdomain ( const typename I::value & label ) const` `[inherited]`

Return the domain of the component with label `label`.

**10.219.4.7** `void mln::labeled_image_base< I, labeled_image< I >>::update_data ( const fun::i2v::array< typename I::value > & relabel_fun )` `[protected, inherited]`

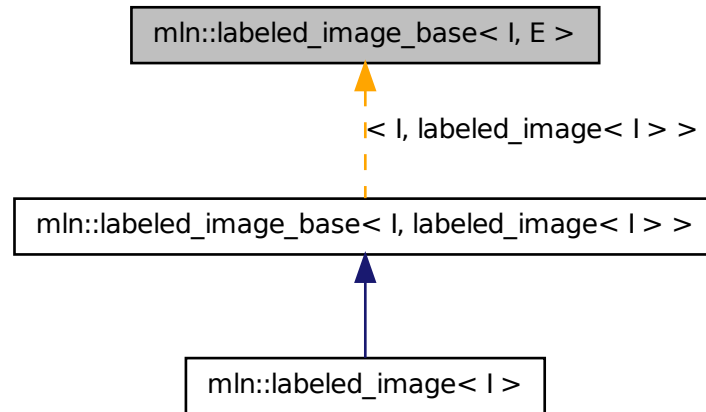
Update bounding boxes information.

## 10.220 mln::labeled\_image\_base< I, E > Class Template Reference

Base class Morpher providing an improved interface for labeled image.

```
#include <labeled_image_base.hh>
```

Inheritance diagram for `mln::labeled_image_base< I, E >`:



## Public Types

- typedef `accu::shape::bbox< typename I::psite >::result` `bbox_t`  
*Type of the bounding component bounding boxes.*

## Public Member Functions

- `const` `bbox_t` & `bbox` (`const` typename `I::value` &`label`) `const`  
*Return the bounding box of the component `label`.*
- `const` `util::array< bbox_t >` & `bboxes` () `const`  
*Return the component bounding boxes.*
- `I::value` `nlabels` () `const`  
*Return the number of labels;.*
- `p_if< mln_box(I), fun::eq_v2b_expr< pw::value_< I >, pw::cst_< typename I::value > >` & `subdomain` (`const` typename `I::value` &`label`) `const`  
*Return the domain of the component with label `label`.*
- `labeled_image_base` ()  
*Constructors*  
*Constructor without argument.*

- `template<typename F >`  
`void relabel (const Function_v2v< F > &f)`  
*Relabel according to a function.*
- `template<typename F >`  
`void relabel (const Function_v2b< F > &f)`  
*Labels may be removed.*

## Protected Member Functions

- `void update_data (const fun::i2v::array< typename I::value > &relabel_fun)`  
*Update bounding boxes information.*

### 10.220.1 Detailed Description

`template<typename I, typename E> class mln::labeled_image_base< I, E >`

Base class Morpher providing an improved interface for labeled image.

#### Template Parameters

*I* The label image type.

This image type allows to access every site set at a given label.

This image type guaranties that labels are contiguous (from 1 to n).

### 10.220.2 Member Typedef Documentation

**10.220.2.1** `template<typename I, typename E> typedef accu::shape::bbox<typename I::site>::result mln::labeled_image_base< I, E >::bbox_t`

Type of the bounding component bounding boxes.

### 10.220.3 Constructor & Destructor Documentation

**10.220.3.1** `template<typename I, typename E > mln::labeled_image_base< I, E >::labeled_image_base ( ) [inline]`

Constructors

Constructor without argument.

### 10.220.4 Member Function Documentation

**10.220.4.1** `template<typename I, typename E > const labeled_image_base< I, E >::bbox_t & mln::labeled_image_base< I, E >::bbox ( const typename I::value & label ) const`

Return the bounding box of the component `label`.

Referenced by `mln::labeled_image_base< I, E >::subdomain()`.

**10.220.4.2** `template<typename I , typename E > const util::array< typename  
labeled_image_base< I, E >::bbox_t > & mln::labeled_image_base< I, E >::bboxes (`  
`) const`

Return the component bounding boxes.

**10.220.4.3** `template<typename I , typename E > I::value mln::labeled_image_base< I, E`  
`>::nlabels ( ) const [inline]`

Return the number of labels;

**10.220.4.4** `template<typename I , typename E > template<typename F > void`  
`mln::labeled_image_base< I, E >::relabel ( const Function_v2b< F > & f )`  
`[inline]`

Labels may be removed.

This overload make sure the labeling is still contiguous.

References `mln::labeling::relabel_inplace()`, `mln::make::relabelfun()`, and `mln::labeled_image_base< I, E >::update_data()`.

**10.220.4.5** `template<typename I , typename E > template<typename F > void`  
`mln::labeled_image_base< I, E >::relabel ( const Function_v2v< F > & f )`  
`[inline]`

Relabel according to a function.

Merge or delete labels according to the given function. This method ensures that the labeling remains contiguous.

References `mln::labeling::relabel_inplace()`, `mln::make::relabelfun()`, and `mln::labeled_image_base< I, E >::update_data()`.

**10.220.4.6** `template<typename I, typename E > p_if< mln_box(I), fun::eq_v2b_expr_<`  
`pw::value_< I >, pw::cst_< typename I::value > > > mln::labeled_image_base< I, E`  
`>::subdomain ( const typename I::value & label ) const`

Return the domain of the component with label `label`.

References `mln::labeled_image_base< I, E >::bbox()`.

**10.220.4.7** `template<typename I, typename E > void mln::labeled_image_base< I, E`  
`>::update_data ( const fun::i2v::array< typename I::value > & relabel_fun )`  
`[protected]`

Update bounding boxes information.

References `mln::util::array< T >::size()`.

Referenced by `mln::labeled_image_base< I, E >::relabel()`.

## 10.221 mln::lazy\_image< I, F, B > Struct Template Reference

Image values are computed on the fly.

```
#include <lazy_image.hh>
```

Inherits image\_identity< mln::trait::ch\_value< I, F::result >::ret, I::domain\_t, lazy\_image< I, F, B > >.

### Public Types

- typedef F::result [lvalue](#)  
*Return type of read-write access.*
- typedef F::result [rvalue](#)  
*Return type of read access.*
- typedef [lazy\\_image](#)< tag::image\_< I >, F, B > [skeleton](#)  
*Skeleton.*

### Public Member Functions

- const [box](#)< typename I::psite > & [domain](#) () const  
*Return domain of lazyd\_image.*
- bool [has](#) (const typename I::psite &) const  
*Test if a pixel value is accessible at p.*
- [lazy\\_image](#) (const F &fun, const B &[box](#))  
*Constructors.*
- [lazy\\_image](#) ()  
*Constructors.*
- F::result [operator\(\)](#) (const typename F::input &x) const  
*Read-only access of pixel value at F::input x.*
- [lvalue operator\(\)](#) (const typename I::psite &p)  
*Read and "write if possible" access of pixel value at point site p.*
- F::result [operator\(\)](#) (const typename F::input &x)  
*Read and "write if possible" access of pixel value at F::input x.*
- [rvalue operator\(\)](#) (const typename I::psite &p) const  
*Read-only access of pixel value at point site p.*

### 10.221.1 Detailed Description

`template<typename I, typename F, typename B> struct mln::lazy_image< I, F, B >`

[Image](#) values are computed on the fly. The parameter `I` is the type of image. The parameter `F` is the type of function. The parameter `B` is the type of box.

This image class take a functor `fun` and a box `box`. Access to `ima(p)` where `p` include `box` return `fun(b)` lazily.

### 10.221.2 Member Typedef Documentation

**10.221.2.1** `template<typename I, typename F, typename B> typedef F ::result mln::lazy_image< I, F, B >::lvalue`

Return type of read-write access.

**10.221.2.2** `template<typename I, typename F, typename B> typedef F ::result mln::lazy_image< I, F, B >::rvalue`

Return type of read access.

**10.221.2.3** `template<typename I, typename F, typename B> typedef lazy_image< tag::image_<I>, F, B > mln::lazy_image< I, F, B >::skeleton`

Skeleton.

### 10.221.3 Constructor & Destructor Documentation

**10.221.3.1** `template<typename I, typename F, typename B> mln::lazy_image< I, F, B >::lazy_image ( )`

Constructors.

**10.221.3.2** `template<typename I , typename F, typename B> mln::lazy_image< I, F, B >::lazy_image ( const F & fun, const B & box ) [inline]`

Constructors.

### 10.221.4 Member Function Documentation

**10.221.4.1** `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.



**10.221.4.2** `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.

**10.221.4.3** `template<typename I, typename F, typename B > lazy_image< I, F, B >::rvalue mln::lazy_image< I, F, B >::operator()( const typename I::psite & p ) const [inline]`

Read-only access of pixel value at point site p.

**10.221.4.4** `template<typename I, typename F, typename B > F::result mln::lazy_image< I, F, B >::operator()( const typename F::input & x ) [inline]`

Read and "write if possible" access of pixel value at F::input x.

**10.221.4.5** `template<typename I, typename F, typename B > F::result mln::lazy_image< I, F, B >::operator()( const typename F::input & x ) const [inline]`

Read-only access of pixel value at F::input x.

**10.221.4.6** `template<typename I, typename F, typename B > lazy_image< I, F, B >::lvalue mln::lazy_image< I, F, B >::operator()( const typename I::psite & p ) [inline]`

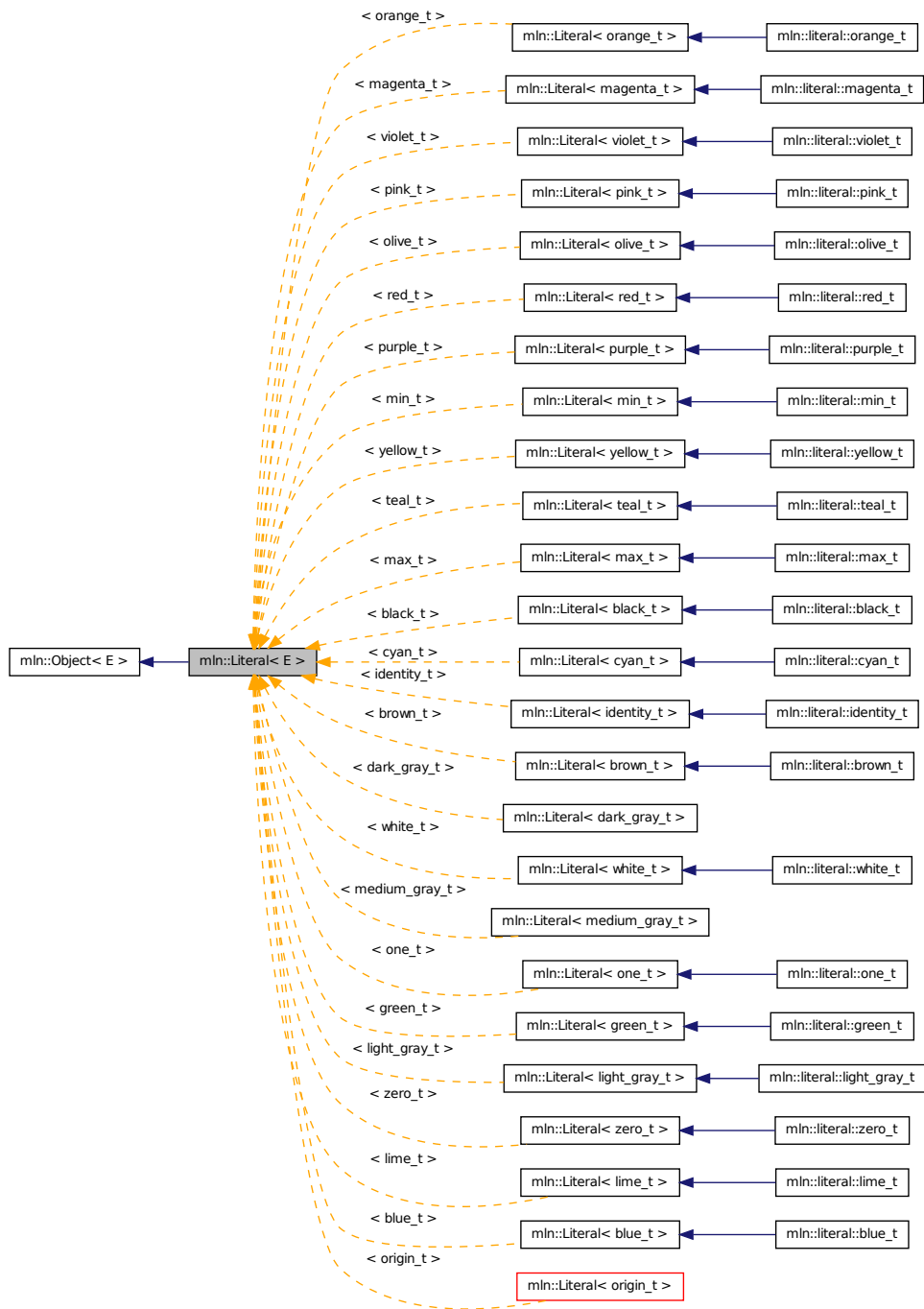
Read and "write if possible" access of pixel value at point site p.

## 10.222 mln::Literal< E > Struct Template Reference

Base class for implementation classes of literals.

```
#include <literal.hh>
```

Inheritance diagram for mln::Literal< E >:



### 10.222.1 Detailed Description

```
template<typename E> struct mln::Literal< E >
```

Base class for implementation classes of literals.

#### See also

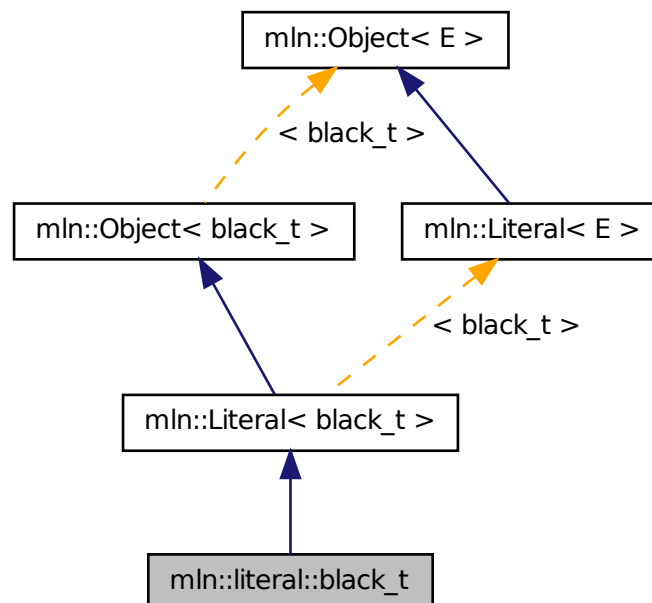
`mln::doc::Literal` for a complete documentation of this class contents.

## 10.223 mln::literal::black\_t Struct Reference

Type of literal black.

```
#include <black.hh>
```

Inheritance diagram for `mln::literal::black_t`:



### 10.223.1 Detailed Description

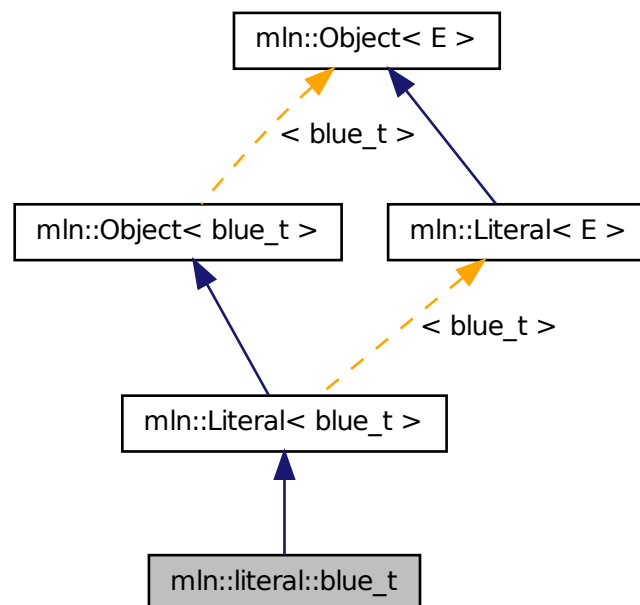
Type of literal black.

## 10.224 mln::literal::blue\_t Struct Reference

Type of literal blue.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::blue\_t:



### 10.224.1 Detailed Description

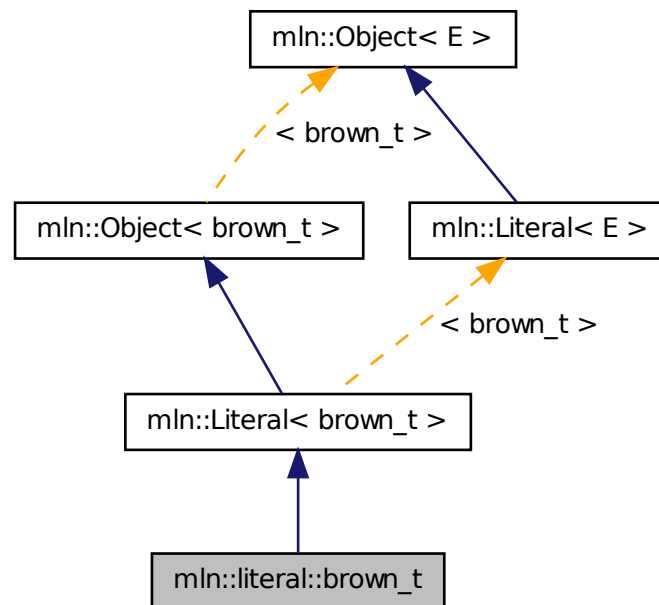
Type of literal blue.

## 10.225 mln::literal::brown\_t Struct Reference

Type of literal brown.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::brown\_t:



### 10.225.1 Detailed Description

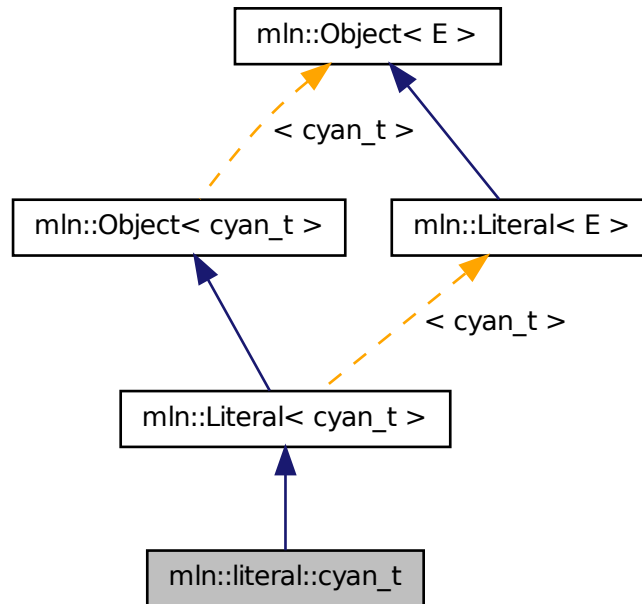
Type of literal brown.

## 10.226 mln::literal::cyan\_t Struct Reference

Type of literal cyan.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::cyan\_t:



### 10.226.1 Detailed Description

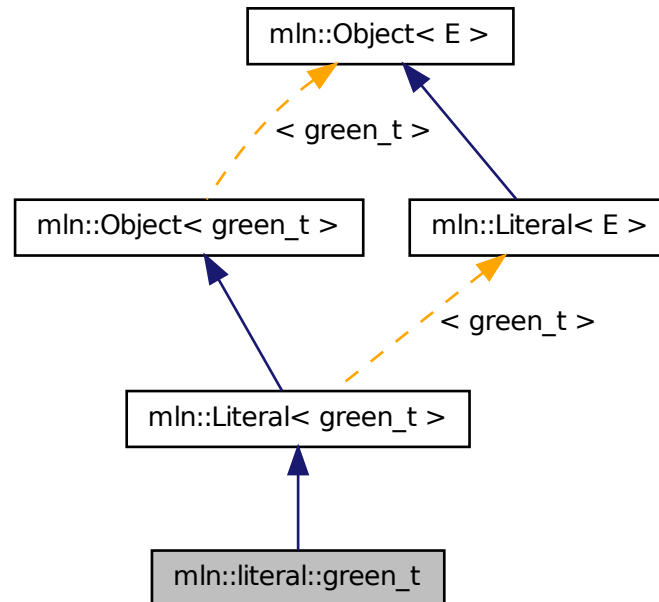
Type of literal cyan.

### 10.227 mln::literal::green\_t Struct Reference

Type of literal green.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::green\_t:



### 10.227.1 Detailed Description

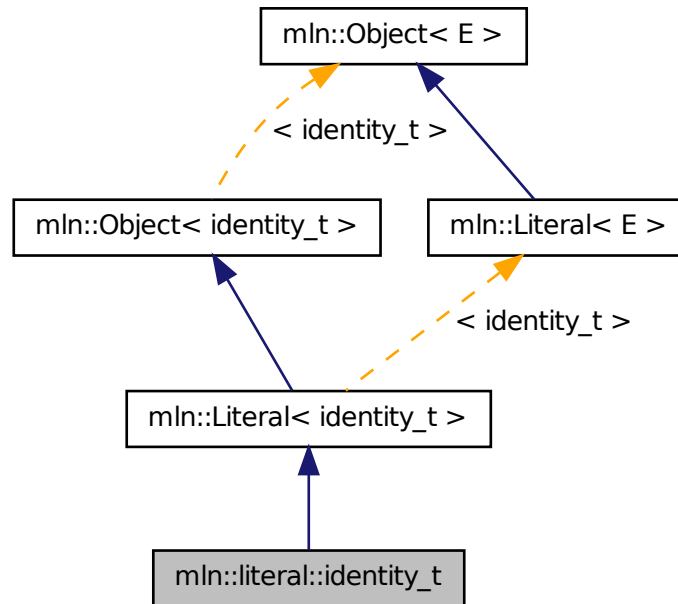
Type of literal green.

## 10.228 mln::literal::identity\_t Struct Reference

Type of literal identity.

```
#include <identity.hh>
```

Inheritance diagram for mln::literal::identity\_t:



### 10.228.1 Detailed Description

Type of literal identity.

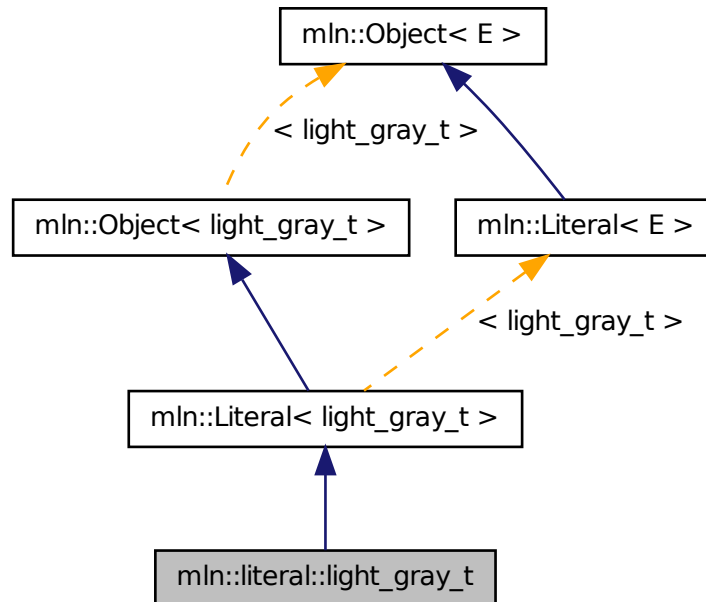
### 10.229 mln::literal::light\_gray\_t Struct Reference

Type of literal grays.

```
#include <grays.hh>
```



Inheritance diagram for mln::literal::light\_gray\_t:



### 10.229.1 Detailed Description

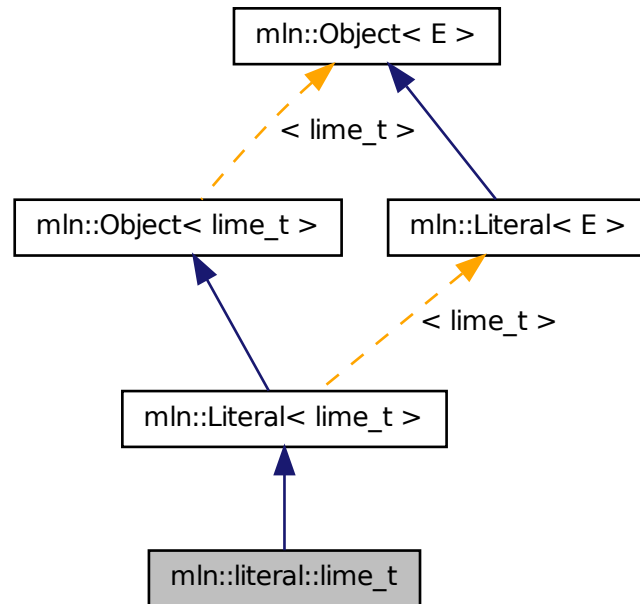
Type of literal grays.

## 10.230 mln::literal::lime\_t Struct Reference

Type of literal lime.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::lime\_t:



### 10.230.1 Detailed Description

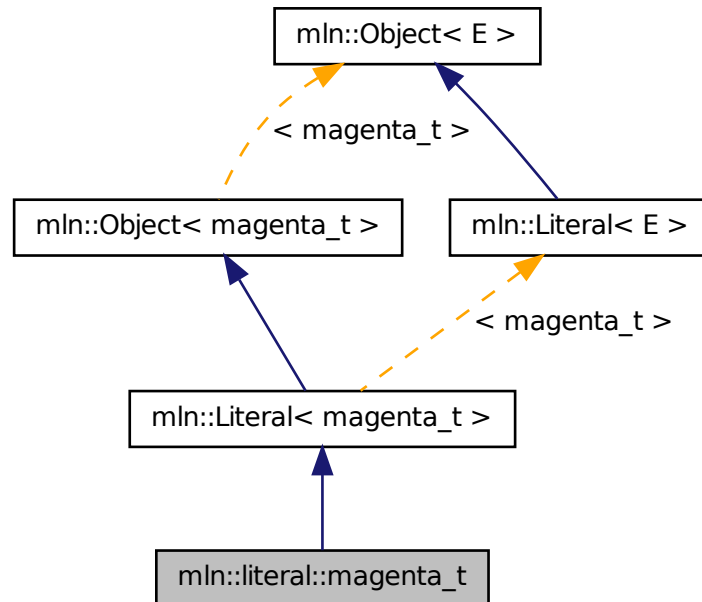
Type of literal lime.

### 10.231 mln::literal::magenta\_t Struct Reference

Type of literal magenta.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::magenta\_t:



### 10.231.1 Detailed Description

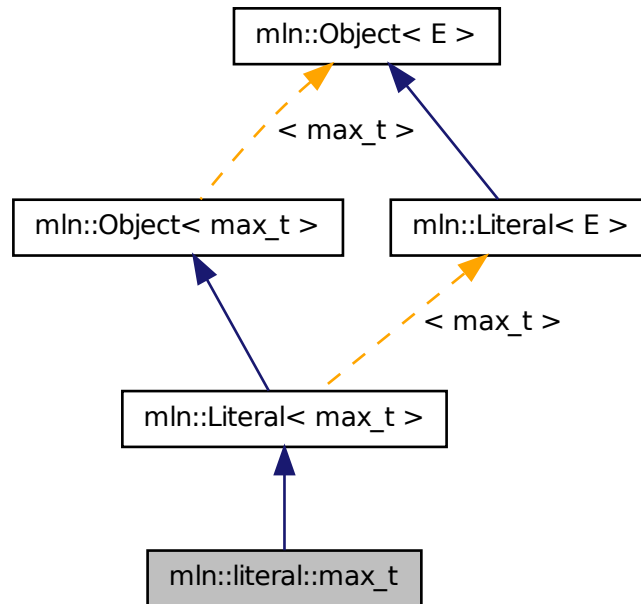
Type of literal magenta.

## 10.232 mln::literal::max\_t Struct Reference

Type of literal max.

```
#include <max.hh>
```

Inheritance diagram for mln::literal::max\_t:



### 10.232.1 Detailed Description

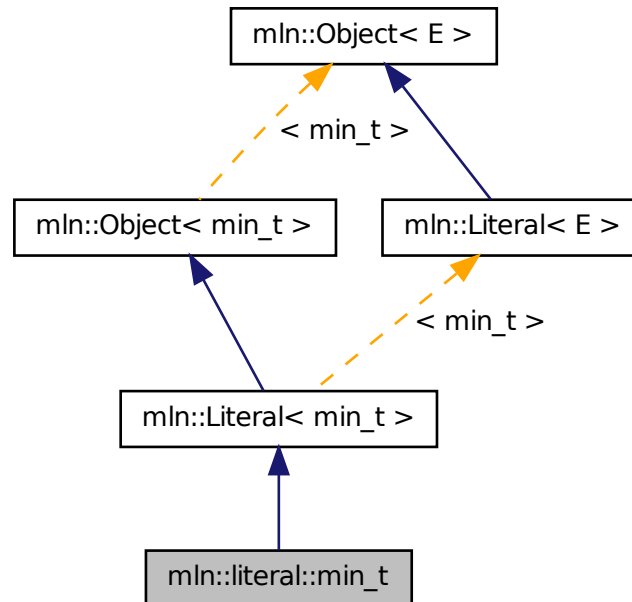
Type of literal max.

### 10.233 mln::literal::min\_t Struct Reference

Type of literal min.

```
#include <min.hh>
```

Inheritance diagram for mln::literal::min\_t:



### 10.233.1 Detailed Description

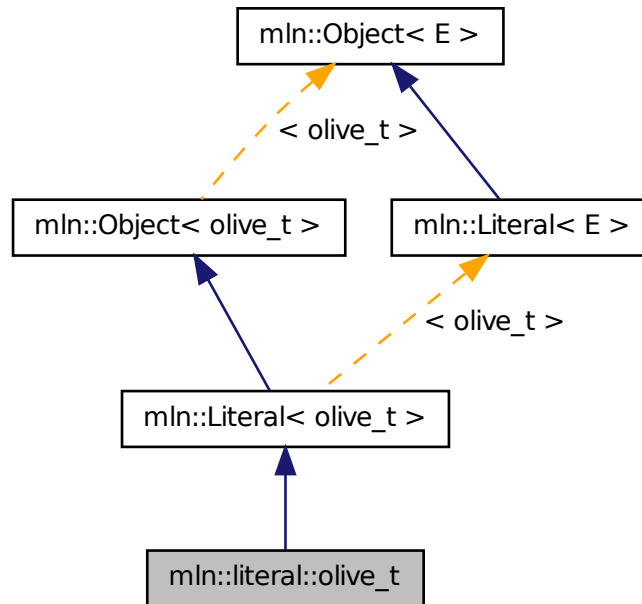
Type of literal min.

## 10.234 mln::literal::olive\_t Struct Reference

Type of literal olive.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::olive\_t:



### 10.234.1 Detailed Description

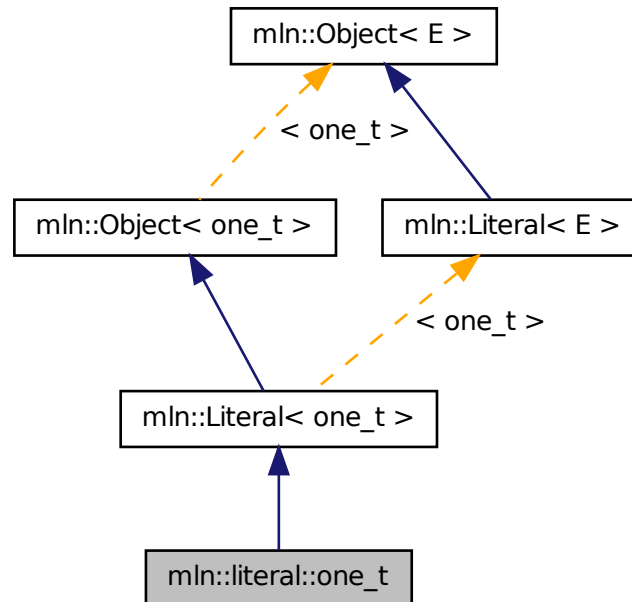
Type of literal olive.

### 10.235 mln::literal::one\_t Struct Reference

Type of literal one.

```
#include <one.hh>
```

Inheritance diagram for mln::literal::one\_t:



### 10.235.1 Detailed Description

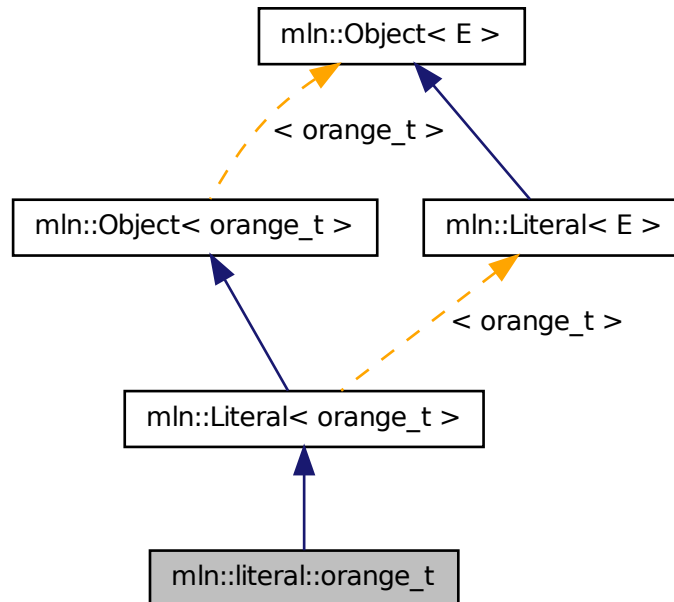
Type of literal one.

## 10.236 mln::literal::orange\_t Struct Reference

Type of literal orange.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::orange\_t:



### 10.236.1 Detailed Description

Type of literal orange.

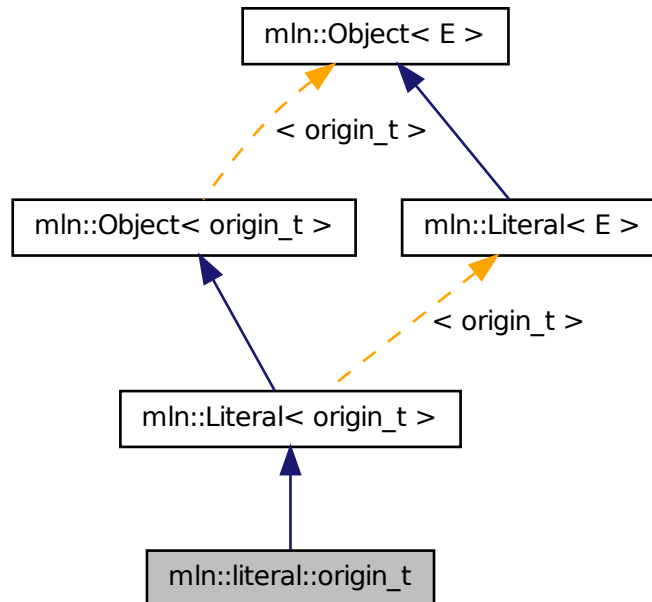
### 10.237 mln::literal::origin\_t Struct Reference

Type of literal origin.

```
#include <origin.hh>
```



Inheritance diagram for mln::literal::origin\_t:



### 10.237.1 Detailed Description

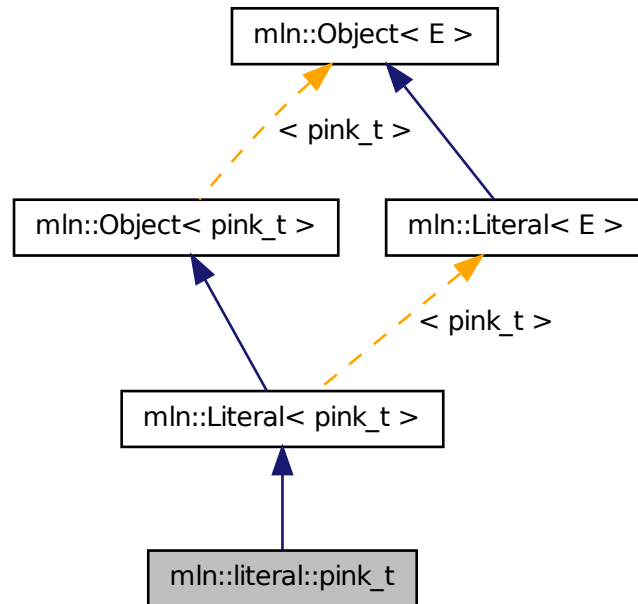
Type of literal origin.

## 10.238 mln::literal::pink\_t Struct Reference

Type of literal pink.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::pink\_t:



### 10.238.1 Detailed Description

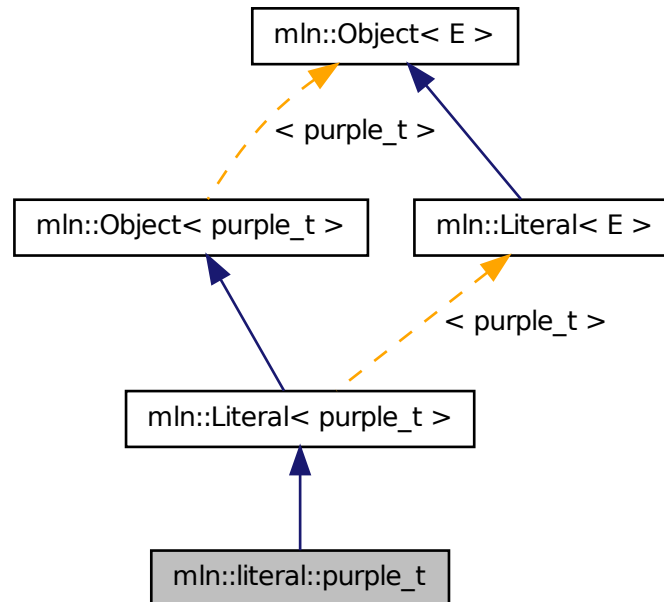
Type of literal pink.

### 10.239 mln::literal::purple\_t Struct Reference

Type of literal purple.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::purple\_t:



### 10.239.1 Detailed Description

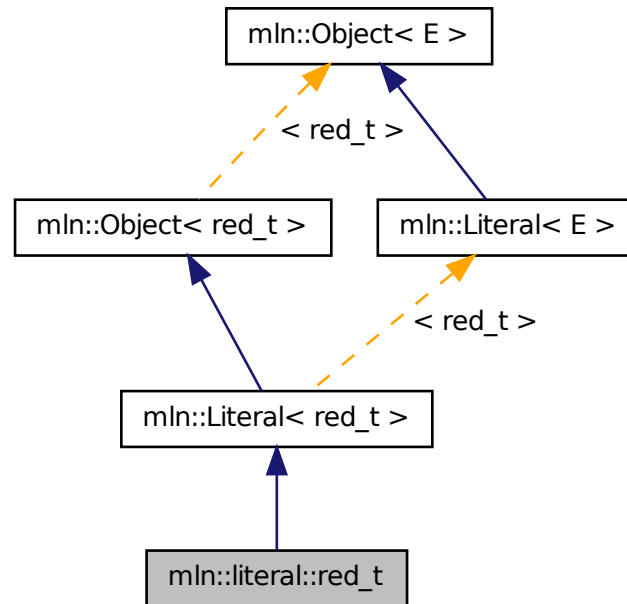
Type of literal purple.

## 10.240 mln::literal::red\_t Struct Reference

Type of literal red.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::red\_t:



### 10.240.1 Detailed Description

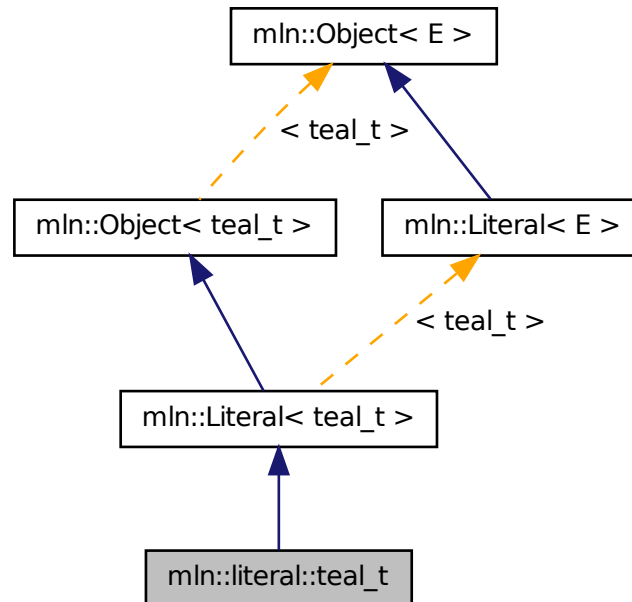
Type of literal red.

### 10.241 mln::literal::teal\_t Struct Reference

Type of literal teal.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::teal\_t:



### 10.241.1 Detailed Description

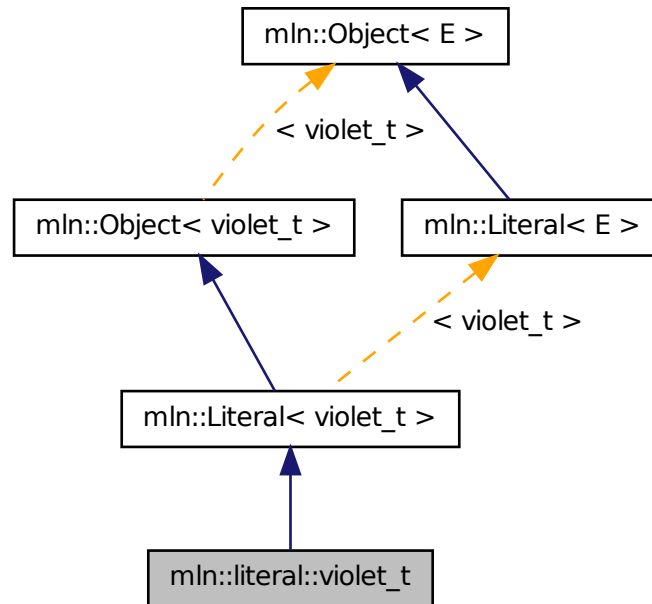
Type of literal teal.

## 10.242 mln::literal::violet\_t Struct Reference

Type of literal violet.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::violet\_t:



### 10.242.1 Detailed Description

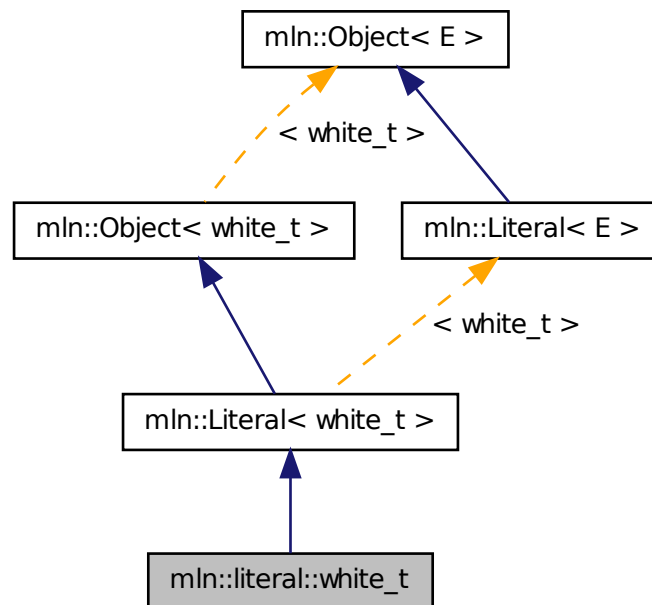
Type of literal violet.

### 10.243 mln::literal::white\_t Struct Reference

Type of literal white.

```
#include <white.hh>
```

Inheritance diagram for mln::literal::white\_t:



### 10.243.1 Detailed Description

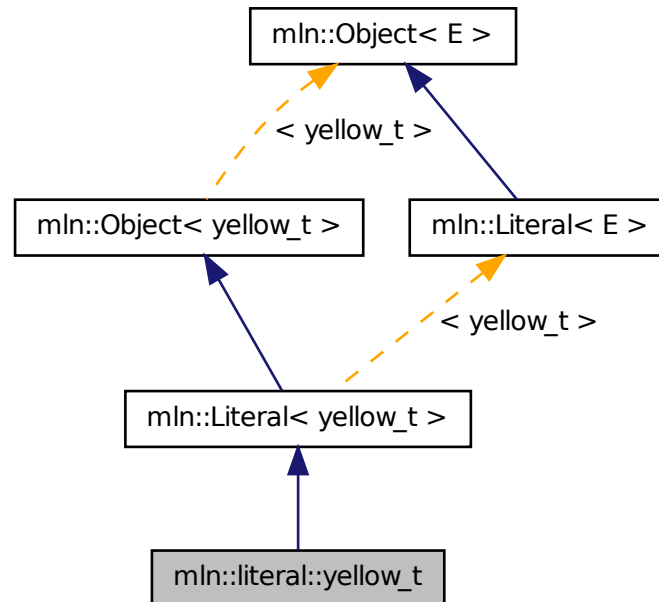
Type of literal white.

## 10.244 mln::literal::yellow\_t Struct Reference

Type of literal yellow.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::yellow\_t:



### 10.244.1 Detailed Description

Type of literal yellow.

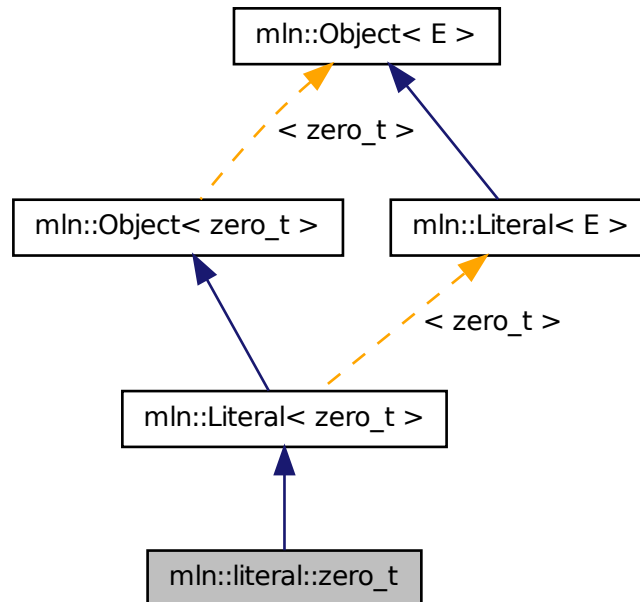
### 10.245 mln::literal::zero\_t Struct Reference

Type of literal zero.

```
#include <zero.hh>
```



Inheritance diagram for mln::literal::zero\_t:



### 10.245.1 Detailed Description

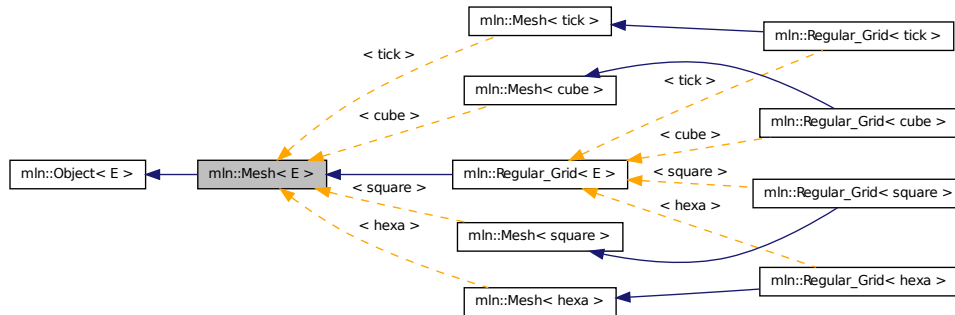
Type of literal zero.

## 10.246 mln::Mesh< E > Struct Template Reference

Base class for implementation classes of meshes.

```
#include <mesh.hh>
```

Inheritance diagram for `mln::Mesh< E >`:



### 10.246.1 Detailed Description

```
template<typename E> struct mln::Mesh< E >
```

Base class for implementation classes of meshes.

#### See also

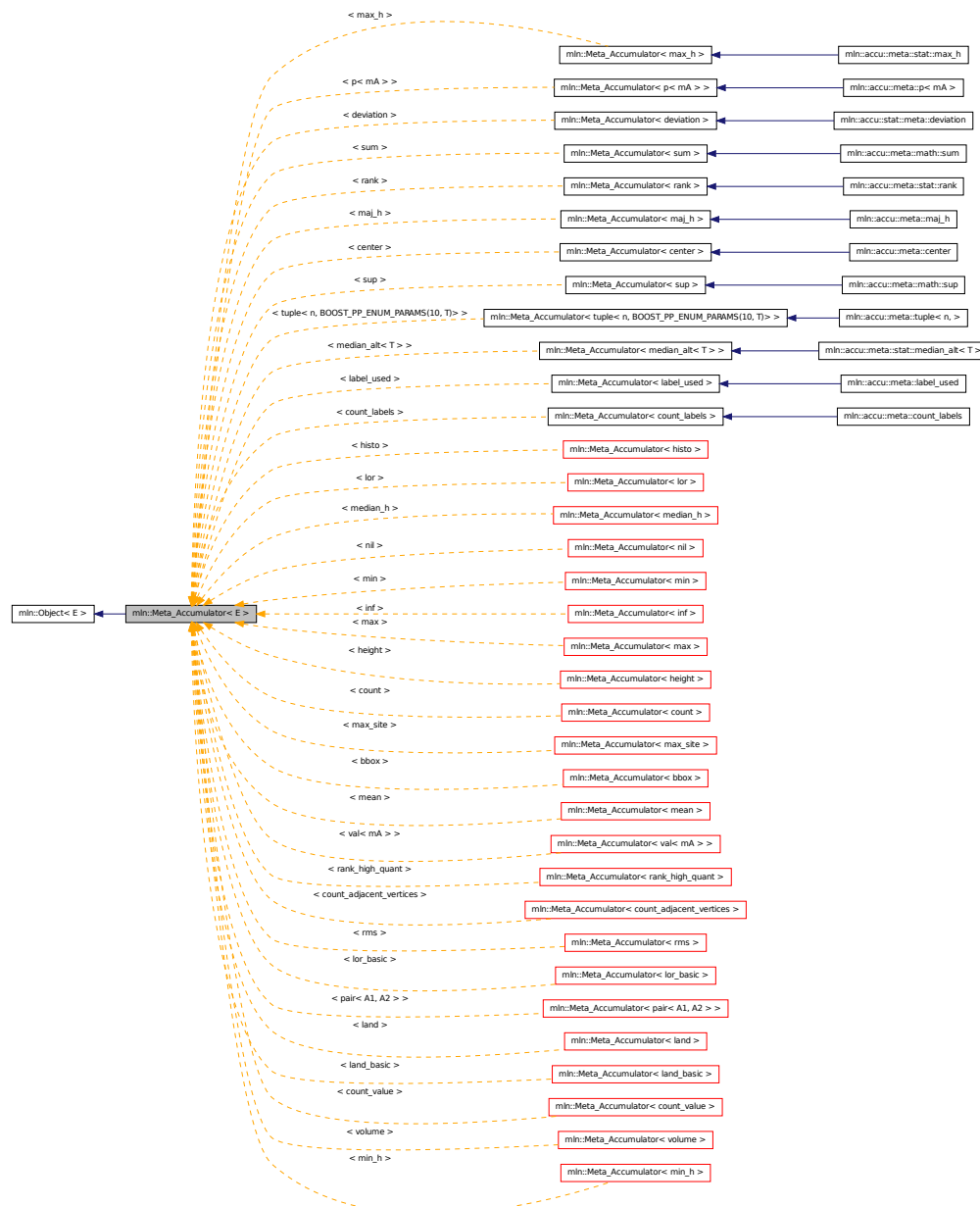
`mln::doc::Mesh` for a complete documentation of this class contents.

## 10.247 mln::Meta\_Accumulator< E > Struct Template Reference

Base class for implementation of meta accumulators.

```
#include <meta_accumulator.hh>
```

Inheritance diagram for mln::Meta\_Accumulator< E >:



## 10.247.1 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

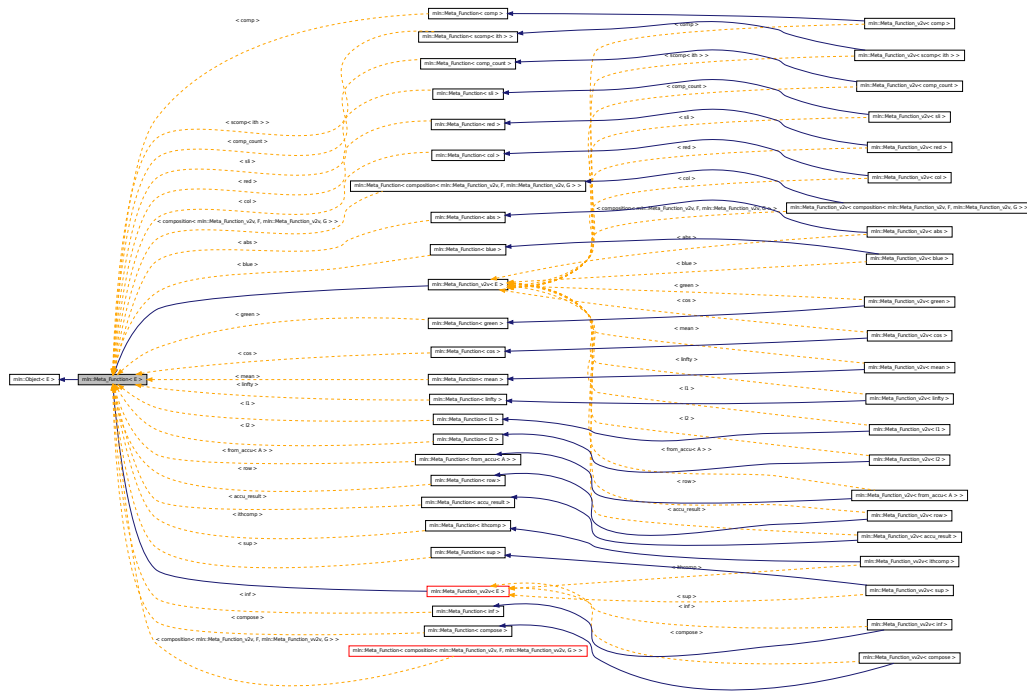
`mln::doc::Meta_Accumulator` for a complete documentation of this class contents.

## 10.248 mln::Meta\_Function< E > Struct Template Reference

Base class for implementation of meta functions.

```
#include <meta_function.hh>
```

Inheritance diagram for mln::Meta\_Function< E >:



### 10.248.1 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

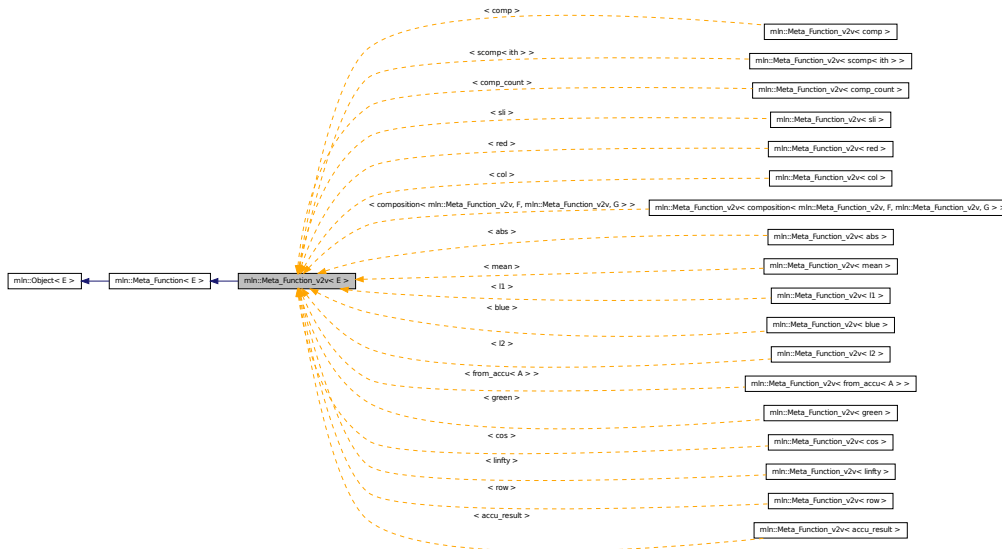
[mln::doc::Meta\\_Function](#) for a complete documentation of this class contents.

## 10.249 mln::Meta\_Function\_v2v< E > Struct Template Reference

Base class for implementation of function-objects from value to value.

```
#include <meta_function.hh>
```

Inheritance diagram for mln::Meta\_Function\_v2v< E >:



### 10.249.1 Detailed Description

`template<typename E> struct mln::Meta_Function_v2v< E >`

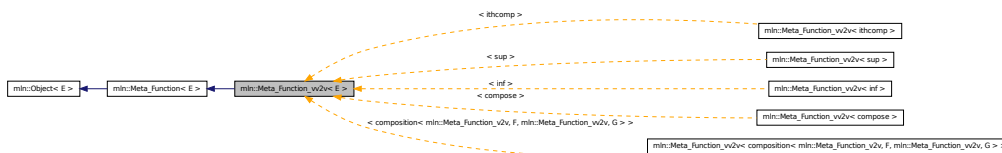
Base class for implementation of function-objects from value to value. The parameter *E* is the exact type.

## 10.250 mln::Meta\_Function\_vv2v< E > Struct Template Reference

Base class for implementation of function-objects from value to value.

```
#include <meta_function.hh>
```

Inheritance diagram for mln::Meta\_Function\_vv2v< E >:



### 10.250.1 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.

## 10.251 mln::metal::ands< E1, E2, E3, E4, E5, E6, E7, E8 > Struct Template Reference

Ands type.

```
#include <ands.hh>
```

### 10.251.1 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.

## 10.252 mln::metal::converts\_to< T, U > Struct Template Reference

"converts-to" check.

```
#include <converts_to.hh>
```

Inherited by mln::metal::converts\_to< T \*, U \* >.

### 10.252.1 Detailed Description

```
template<typename T, typename U> struct mln::metal::converts_to< T, U >
```

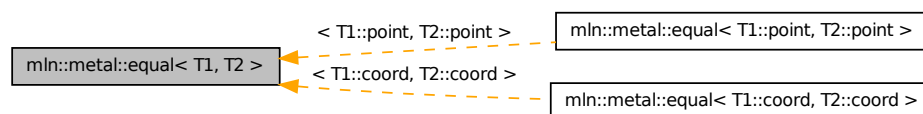
"converts-to" check.

## 10.253 mln::metal::equal< T1, T2 > Struct Template Reference

Definition of a static 'equal' test.

```
#include <equal.hh>
```

Inheritance diagram for mln::metal::equal< T1, T2 >:



### 10.253.1 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.

## 10.254 mln::metal::goes\_to< T, U > Struct Template Reference

"goes-to" check.

```
#include <goes_to.hh>
```

### 10.254.1 Detailed Description

```
template<typename T, typename U> struct mln::metal::goes_to< T, U >
```

"goes-to" check. FIXME: Doc!

## 10.255 mln::metal::is< T, U > Struct Template Reference

"is" check.

```
#include <is.hh>
```

### 10.255.1 Detailed Description

```
template<typename T, typename U> struct mln::metal::is< T, U >
```

"is" check. Check whether T inherits from U.

## 10.256 mln::metal::is\_a< T, M > Struct Template Reference

"is\_a" check.

```
#include <is_a.hh>
```

### 10.256.1 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.

## 10.257 mln::metal::is\_not< T, U > Struct Template Reference

"is\_not" check.

```
#include <is_not.hh>
```

### 10.257.1 Detailed Description

```
template<typename T, typename U> struct mln::metal::is_not< T, U >
```

"is\_not" check. FIXME: Doc!

## 10.258 mln::metal::is\_not\_a< T, M > Struct Template Reference

"is\_not\_a" static Boolean expression.

```
#include <is_not_a.hh>
```

### 10.258.1 Detailed Description

```
template<typename T, template< class > class M> struct mln::metal::is_not_a< T, M >
```

"is\_not\_a" static Boolean expression.

## 10.259 mln::mixed\_neighb< W > Class Template Reference

Adapter class from window to neighborhood.

```
#include <mixed_neighb.hh>
```

Inherits `neighb_base< W, mixed_neighb< W > >`, and `mlc_is_aW`.

### Public Types

- typedef `mixed_neighb_bkd_niter< W >` `bkd_niter`  
*Backward site iterator associated type.*
- typedef `mixed_neighb_fwd_niter< W >` `fwd_niter`  
*Forward site iterator associated type.*
- typedef `fwd_niter niter`  
*Site iterator associated type.*

### Public Member Functions

- `mixed_neighb ()`  
*Constructor without argument.*
- `mixed_neighb (const W &win)`  
*Constructor from a window win.*



### 10.259.1 Detailed Description

`template<typename W> class mln::mixed_neighb< W >`

Adapter class from window to neighborhood.

### 10.259.2 Member Typedef Documentation

**10.259.2.1** `template<typename W> typedef mixed_neighb_bkd_niter<W> mln::mixed_neighb< W >::bkd_niter`

Backward site iterator associated type.

**10.259.2.2** `template<typename W> typedef mixed_neighb_fwd_niter<W> mln::mixed_neighb< W >::fwd_niter`

Forward site iterator associated type.

**10.259.2.3** `template<typename W> typedef fwd_niter mln::mixed_neighb< W >::niter`

[Site](#) iterator associated type.

### 10.259.3 Constructor & Destructor Documentation

**10.259.3.1** `template<typename W > mln::mixed_neighb< W >::mixed_neighb ( ) [inline]`

Constructor without argument.

**10.259.3.2** `template<typename W > mln::mixed_neighb< W >::mixed_neighb ( const W & win ) [inline]`

Constructor from a window `win`.

## 10.260 mln::morpho::attribute::card< I > Class Template Reference

Cardinality accumulator class.

```
#include <card.hh>
```

Inherits `base< unsigned, card< I > >`.

### Public Member Functions

- `bool is_valid () const`

*Check whether this accu is able to return a result.*

- void `take_as_init` (const T &t)  
*Take as initialization the value  $t$ .*
- void `take_n_times` (unsigned n, const T &t)  
*Take  $n$  times the value  $t$ .*
- unsigned `to_result` () const  
*Get the value of the accumulator.*
- void `init` ()  
*Manipulators.*

### 10.260.1 Detailed Description

`template<typename I> class mln::morpho::attribute::card< I >`

Cardinality accumulator class.

### 10.260.2 Member Function Documentation

**10.260.2.1** `template<typename I> void mln::morpho::attribute::card< I >::init ( )`  
`[inline]`

Manipulators.

**10.260.2.2** `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.

**10.260.2.3** `void mln::Accumulator< card< I > >::take_as_init ( const T & t )` `[inherited]`

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.260.2.4** `void mln::Accumulator< card< I > >::take_n_times ( unsigned n, const T & t )`  
`[inherited]`

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.260.2.5** `template<typename I> unsigned mln::morpho::attribute::card< I >::to_result ( )`  
`const [inline]`

Get the value of the accumulator.

## 10.261 mln::morpho::attribute::count\_adjacent\_vertices< I > Struct Template Reference

Count\_Adjacent\_Vertices accumulator class.

```
#include <count_adjacent_vertices.hh>
```

Inherits base< unsigned, count\_adjacent\_vertices< I > >.

### Public Member Functions

- bool [is\\_valid](#) () const  
*Check whether this accu is able to return a result.*
- void [take\\_as\\_init](#) (const T &t)  
*Take as initialization the value t.*
- void [take\\_n\\_times](#) (unsigned n, const T &t)  
*Take n times the value t.*
- unsigned [to\\_result](#) () const  
*Get the value of the accumulator.*
- void [init](#) ()  
*Manipulators.*

### 10.261.1 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.

### 10.261.2 Member Function Documentation

**10.261.2.1** `template<typename I > void mln::morpho::attribute::count_adjacent_vertices< I >::init ( ) [inline]`

Manipulators.

**10.261.2.2** `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.

**10.261.2.3** `void mln::Accumulator< count_adjacent_vertices< I > >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.261.2.4** `void mln::Accumulator< count_adjacent_vertices< I > >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.261.2.5** `template<typename I > unsigned mln::morpho::attribute::count_adjacent_vertices< I >::to_result ( ) const [inline]`

Get the value of the accumulator.

## 10.262 `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.*

### 10.262.1 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.

### 10.262.2 Member Function Documentation

**10.262.2.1** `template<typename I> unsigned mln::morpho::attribute::height< I >::base_level ( ) const [inline]`

Get base & current level of the accumulator.

**10.262.2.2** `template<typename I> void mln::morpho::attribute::height< I >::init ( ) [inline]`

Manipulators.

**10.262.2.3** `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()`.

**10.262.2.4** `void mln::Accumulator< height< I > >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.262.2.5** `void mln::Accumulator< height< I > >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.262.2.6** `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()`.

## 10.263 mln::morpho::attribute::sharpness< I > Struct Template Reference

Sharpness accumulator class.

```
#include <sharpness.hh>
```

Inherits base< double, sharpness< I > >.

### Public Member Functions

- unsigned [area](#) () const  
*Give the area of the component.*
- unsigned [height](#) () const  
*Give the height.*
- bool [is\\_valid](#) () const  
*Check whether this accu is able to return a result.*
- void [take\\_as\\_init](#) (const T &t)  
*Take as initialization the value t.*
- void [take\\_n\\_times](#) (unsigned n, const T &t)  
*Take n times the value t.*
- double [to\\_result](#) () const  
*Get the value of the accumulator.*
- unsigned [volume](#) () const  
*Give the volume of the component.*
  
- void [init](#) ()  
*Manipulators.*

### 10.263.1 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.

### 10.263.2 Member Function Documentation

**10.263.2.1 template<typename I> unsigned mln::morpho::attribute::sharpness< I >::area ( ) const [inline]**

Give the area of the component.

**10.263.2.2** `template<typename I > unsigned mln::morpho::attribute::sharpness< I >::height ( ) const [inline]`

Give the height.

**10.263.2.3** `template<typename I > void mln::morpho::attribute::sharpness< I >::init ( ) [inline]`

Manipulators.

**10.263.2.4** `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.

**10.263.2.5** `void mln::Accumulator< sharpness< I > >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.263.2.6** `void mln::Accumulator< sharpness< I > >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.263.2.7** `template<typename I > double mln::morpho::attribute::sharpness< I >::to_result ( ) const [inline]`

Get the value of the accumulator.

**10.263.2.8** `template<typename I > unsigned mln::morpho::attribute::sharpness< I >::volume ( ) const [inline]`

Give the volume of the component.

## 10.264 mln::morpho::attribute::sum< I, S > Class Template Reference

Suminality accumulator class.

```
#include <sum.hh>
```

Inherits base< S, sum< I, S > >.

## Public Member Functions

- bool `is_valid` () const  
*Check whether this `accu` is able to return a result.*
- void `set_value` (const argument &v)  
*Set the return value of the accumulator.*
- void `take_as_init` (const T &t)  
*Take as initialization the value  $\tau$ .*
- void `take_n_times` (unsigned n, const T &t)  
*Take  $n$  times the value  $\tau$ .*
- S `to_result` () const  
*Get the value of the accumulator.*
- void `untake` (const argument &v)  
*Untake a value from the accumulator.*
- void `init` ()  
*Manipulators.*

### 10.264.1 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.

### 10.264.2 Member Function Documentation

**10.264.2.1** `template<typename I, typename S > void mln::morpho::attribute::sum< I, S >::init ( ) [inline]`

Manipulators.

References `mln::literal::zero`.

**10.264.2.2** `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.



**10.264.2.3** `template<typename I , typename S > void mln::morpho::attribute::sum< I, S >::set_value ( const argument & v ) [inline]`

Set the return value of the accumulator.

**10.264.2.4** `void mln::Accumulator< sum< I, S > >::take_as_init ( const T & t ) [inherited]`

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.264.2.5** `void mln::Accumulator< sum< I, S > >::take_n_times ( unsigned n, const T & t ) [inherited]`

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.264.2.6** `template<typename I , typename S > S mln::morpho::attribute::sum< I, S >::to_result ( ) const [inline]`

Get the value of the accumulator.

**10.264.2.7** `template<typename I , typename S > void mln::morpho::attribute::sum< I, S >::untake ( const argument & v ) [inline]`

Untake a value from the accumulator.

## 10.265 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.

## 10.265.1 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.

## 10.265.2 Member Function Documentation

**10.265.2.1** `template<typename I> unsigned mln::morpho::attribute::volume< I >::area ( )`  
const [inline]

Give the area.

**10.265.2.2** `template<typename I> void mln::morpho::attribute::volume< I >::init ( )`  
[inline]

Manipulators.

**10.265.2.3** `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.

**10.265.2.4** `void mln::Accumulator< volume< I > >::take_as_init ( const T & t )`  
[inherited]

Take as initialization the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

**10.265.2.5** `void mln::Accumulator< volume< I > >::take_n_times ( unsigned n, const T & t )`  
[inherited]

Take  $n$  times the value  $t$ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

### 10.265.2.6 `template<typename I> unsigned mln::morpho::attribute::volume< I >::to_result ( ) const [inline]`

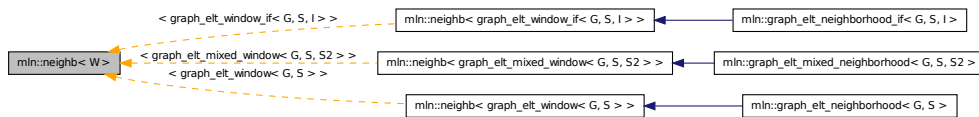
Get the value of the accumulator.

## 10.266 mln::neighb< W > Class Template Reference

Adapter class from window to neighborhood.

```
#include <neighb.hh>
```

Inheritance diagram for mln::neighb< W >:



### Public Types

- `typedef neighb_bkd_niter< W > bkd_niter`  
*Backward site iterator associated type.*
- `typedef neighb_fwd_niter< W > fwd_niter`  
*Forward site iterator associated type.*
- `typedef fwd_niter niter`  
*Site iterator associated type.*

### Public Member Functions

- `neighb ()`  
*Constructor without argument.*
- `neighb (const W &win)`  
*Constructor from a window win.*

### 10.266.1 Detailed Description

```
template<typename W> class mln::neighb< W >
```

Adapter class from window to neighborhood.

## 10.266.2 Member Typedef Documentation

**10.266.2.1** `template<typename W> typedef neighb_bkd_niter<W> mln::neighb< W >::bkd_niter`

Backward site iterator associated type.

**10.266.2.2** `template<typename W> typedef neighb_fwd_niter<W> mln::neighb< W >::fwd_niter`

Forward site iterator associated type.

**10.266.2.3** `template<typename W> typedef fwd_niter mln::neighb< W >::niter`

[Site](#) iterator associated type.

## 10.266.3 Constructor & Destructor Documentation

**10.266.3.1** `template<typename W > mln::neighb< W >::neighb ( ) [inline]`

Constructor without argument.

**10.266.3.2** `template<typename W> mln::neighb< W >::neighb ( const W & win ) [inline]`

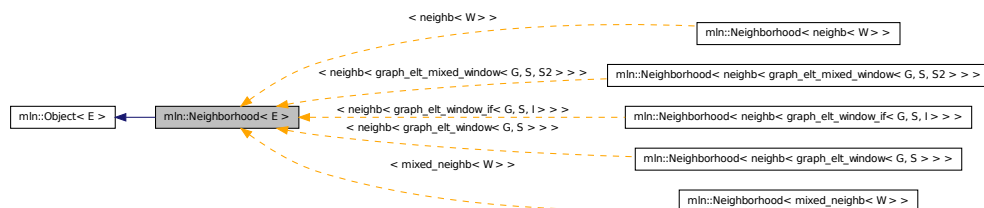
Constructor from a window `win`.

## 10.267 mln::Neighborhood< E > Struct Template Reference

Base class for implementation classes that are neighborhoods.

```
#include <neighborhood.hh>
```

Inheritance diagram for `mln::Neighborhood< E >`:



### 10.267.1 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.

## 10.268 `mln::Neighborhood< void >` Struct Template Reference

[Neighborhood](#) category flag type.

```
#include <neighborhood.hh>
```

### 10.268.1 Detailed Description

```
template<> struct mln::Neighborhood< void >
```

[Neighborhood](#) category flag type.

## 10.269 `mln::Object< E >` Struct Template Reference

Base class for almost every class defined in Milena.

```
#include <object.hh>
```

Inherited by [mln::Base< E >](#), [mln::Browsing< E >](#), [mln::Delta\\_Point\\_Site< E >](#), [mln::Function< E >](#), [mln::Gdpoint< E >](#), [mln::Graph< E >](#), [mln::Image< E >](#), [mln::io::off::internal::off\\_loader< I, E >](#), [mln::io::off::internal::off\\_saver< I, E >](#), [mln::Iterator< E >](#), [mln::Literal< E >](#), [mln::Mesh< E >](#), [mln::Meta\\_Accumulator< E >](#), [mln::Meta\\_Function< E >](#), [mln::Neighborhood< E >](#), [mln::Point\\_Site< E >](#), [mln::Proxy< E >](#), [mln::Site< E >](#), [mln::Site\\_Set< E >](#), [mln::Value< E >](#), [mln::value::HSL< E >](#), [mln::Value\\_Set< E >](#), [mln::Weighted\\_Window< E >](#), and [mln::Window< E >](#).

### 10.269.1 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.

## 10.270 `mln::p2p_image< I, F >` Struct Template Reference

FIXME: Doc!

```
#include <p2p_image.hh>
```

Inherits [image\\_domain\\_morpher< I, I::domain\\_t, p2p\\_image< I, F > >](#).

## Public Types

- typedef `p2p_image< tag::image_< I >, tag::function_< F > >` `skeleton`  
*Skeleton.*

## Public Member Functions

- `const I::domain_t & domain () const`  
*Give the definition domain.*
- `const F & fun () const`  
*Give the p2p function.*
- `I::rvalue operator() (const typename I::psite &p) const`  
*Read-only access to the image value located at point p.*
- `internal::morpher_lvalue_< I >::ret operator() (const typename I::psite &p)`  
*Read-write access to the image value located at point p.*
- `p2p_image (I &ima, const F &f)`  
*Constructor from an image *ima* and a predicate *f*.*
- `p2p_image ()`  
*Constructor without argument.*

### 10.270.1 Detailed Description

```
template<typename I, typename F> struct mln::p2p_image< I, F >
```

FIXME: Doc!

### 10.270.2 Member Typedef Documentation

**10.270.2.1** `template<typename I, typename F> typedef p2p_image< tag::image_<I>, tag::function_<F> > mln::p2p_image< I, F >::skeleton`

Skeleton.

### 10.270.3 Constructor & Destructor Documentation

**10.270.3.1** `template<typename I, typename F > mln::p2p_image< I, F >::p2p_image ( )`  
`[inline]`

Constructor without argument.

**10.270.3.2** `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*.

### 10.270.4 Member Function Documentation

**10.270.4.1** `template<typename I, typename F > const I::domain_t & mln::p2p_image< I, F >::domain ( ) const [inline]`

Give the definition domain.

**10.270.4.2** `template<typename I, typename F > const F & mln::p2p_image< I, F >::fun ( ) const [inline]`

Give the p2p function.

**10.270.4.3** `template<typename I, typename F > I::rvalue mln::p2p_image< I, F >::operator() ( const typename I::psite & p ) const [inline]`

Read-only access to the image value located at point *p*.

**10.270.4.4** `template<typename I, typename F > internal::morpher_lvalue_< I >::ret mln::p2p_image< I, F >::operator() ( const typename I::psite & p ) [inline]`

Read-write access to the image value located at point *p*.

## 10.271 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_site`< self\_ > `psite`  
*Psite associated type.*

## Public Member Functions

- `p_array`< P > & `append` (const P &p)  
*Append a point p.*
- `p_array`< P > & `append` (const `p_array`< P > &other)  
*Append an array other of points.*
- void `change` (const `psite` &p, const P &new\_p)  
*Change site p into new\_p.*
- void `clear` ()  
*Clear this set.*
- bool `has` (const util::index &i) const  
*Test is index i belongs to this site set.*
- bool `has` (const `psite` &p) const  
*Test is p belongs to this site set.*
- void `insert` (const P &p)  
*Insert a point p (equivalent as 'append').*
- bool `is_valid` () const  
*Test this set validity so returns always true.*
- std::size\_t `memory_size` () const  
*Return the size of this site set in memory.*
- unsigned `nsites` () const  
*Give the number of sites.*
- const P & `operator[]` (const util::index &i) const  
*Return the i-th element.*
- P & `operator[]` (unsigned i)  
*Return the i-th site (mutable).*
- const P & `operator[]` (unsigned i) const  
*Return the i-th site (constant).*



- [p\\_array](#) ()  
*Constructor.*
- [p\\_array](#) (const std::vector< P > &vect)  
*Constructor from a vector vect.*
- void [reserve](#) (size\_type n)  
*Reserve n cells.*
- void [resize](#) (size\_t size)  
*Update the size of this array.*
- const std::vector< P > & [std\\_vector](#) () const  
*Return the corresponding std::vector of points.*

### 10.271.1 Detailed Description

**template<typename P> class mln::p\_array< P >**

Multi-set of sites. [Site](#) set class based on std::vector.

### 10.271.2 Member Typedef Documentation

**10.271.2.1 template<typename P> typedef p\_indexed\_bkd\_piter<self\_> mln::p\_array< P >::bkd\_piter**

Backward [Site\\_Iterator](#) associated type.

**10.271.2.2 template<typename P> typedef P mln::p\_array< P >::element**

Element associated type.

**10.271.2.3 template<typename P> typedef p\_indexed\_fwd\_piter<self\_> mln::p\_array< P >::fwd\_piter**

Forward [Site\\_Iterator](#) associated type.

**10.271.2.4 template<typename P> typedef P mln::p\_array< P >::i\_element**

Insertion element associated type.

**10.271.2.5 template<typename P> typedef fwd\_piter mln::p\_array< P >::piter**

[Site\\_Iterator](#) associated type.

**10.271.2.6** `template<typename P> typedef p_indexed_psite<self_> mln::p_array< P >::psite`

Psite associated type.

### 10.271.3 Constructor & Destructor Documentation

**10.271.3.1** `template<typename P > mln::p_array< P >::p_array ( ) [inline]`

Constructor.

**10.271.3.2** `template<typename P > mln::p_array< P >::p_array ( const std::vector< P > & vect ) [inline]`

Constructor from a vector `vect`.

### 10.271.4 Member Function Documentation

**10.271.4.1** `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()`.

**10.271.4.2** `template<typename P > p_array< P > & mln::p_array< P >::append ( const p_array< P > & other ) [inline]`

Append an array `other` of points.

References `mln::p_array< P >::std_vector()`.

**10.271.4.3** `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()`.

**10.271.4.4** `template<typename P > void mln::p_array< P >::clear ( ) [inline]`

Clear this set.

**10.271.4.5** `template<typename P > bool mln::p_array< P >::has ( const psite & p ) const [inline]`

Test if `p` belongs to this site set.

Referenced by `mln::p_array< P >::change()`, and `mln::p_array< P >::operator[]()`.

**10.271.4.6** `template<typename P> bool mln::p_array< P >::has ( const util::index & i ) const [inline]`

Test is index *i* belongs to this site set.

References mln::p\_array< P >::nsites().

**10.271.4.7** `template<typename P> void mln::p_array< P >::insert ( const P & p ) [inline]`

Insert a point *p* (equivalent as 'append').

**10.271.4.8** `template<typename P> bool mln::p_array< P >::is_valid ( ) const [inline]`

Test this set validity so returns always true.

**10.271.4.9** `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().

**10.271.4.10** `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[]().

**10.271.4.11** `template<typename P> P & mln::p_array< P >::operator[] ( unsigned i ) [inline]`

Return the *i*-th site (mutable).

References mln::p\_array< P >::nsites().

**10.271.4.12** `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().

**10.271.4.13** `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().

**10.271.4.14** `template<typename P> void mln::p_array< P >::reserve ( size_type n )`  
**[inline]**

Reserve *n* cells.

Referenced by `mln::convert::to_p_array()`.

**10.271.4.15** `template<typename P> void mln::p_array< P >::resize ( size_t size )` **[inline]**

Update the size of this array.

**10.271.4.16** `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()`.

## 10.272 `mln::p_centered< W >` Class Template Reference

[Site](#) set corresponding to a window centered on a site.

```
#include <p_centered.hh>
```

Inherits `site_set_base< W::psite, p_centered< W >>`, and `mlc_is_aW`.

### Public Types

- typedef `p_centered_piter< W > bkd_piter`  
*Backward [Site\\_Iterator](#) associated type.*
- typedef `psite element`  
*Element associated type.*
- typedef `p_centered_piter< W > fwd_piter`  
*Forward [Site\\_Iterator](#) associated type.*
- typedef `fwd_piter piter`  
*[Site\\_Iterator](#) associated type.*
- typedef `W::psite psite`  
*Psite associated type.*
- typedef `W::site site`  
*[Site](#) associated type.*

## Public Member Functions

- const W::psite & [center](#) () const  
*Give the center of this site set.*
- template<typename P >  
bool [has](#) (const P &p) const  
*Test if p belongs to the box.*
- bool [is\\_valid](#) () const  
*Test if this site set is initialized.*
- std::size\_t [memory\\_size](#) () const  
*Return the size of this site set in memory.*
- [p\\_centered](#) (const W &win, const typename W::psite &c)  
*Constructor from a window win and a center c.*
- [p\\_centered](#) ()  
*Constructor without argument.*
- const W & [window](#) () const  
*Give the window this site set is defined upon.*

### 10.272.1 Detailed Description

template<typename W> class mln::p\_centered< W >

[Site](#) set corresponding to a window centered on a site.

### 10.272.2 Member Typedef Documentation

10.272.2.1 template<typename W> typedef p\_centered\_piter<W> mln::p\_centered< W >::bkd\_piter

Backward [Site\\_Iterator](#) associated type.

10.272.2.2 template<typename W> typedef psite mln::p\_centered< W >::element

Element associated type.

10.272.2.3 template<typename W> typedef p\_centered\_piter<W> mln::p\_centered< W >::fwd\_piter

Forward [Site\\_Iterator](#) associated type.

**10.272.2.4** `template<typename W> typedef fwd_piter mln::p_centered< W >::piter`

[Site\\_Iterator](#) associated type.

**10.272.2.5** `template<typename W> typedef W ::psite mln::p_centered< W >::psite`

Psite associated type.

**10.272.2.6** `template<typename W> typedef W ::site mln::p_centered< W >::site`

[Site](#) associated type.

### 10.272.3 Constructor & Destructor Documentation

**10.272.3.1** `template<typename W > mln::p_centered< W >::p_centered ( ) [inline]`

Constructor without argument.

**10.272.3.2** `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()`.

### 10.272.4 Member Function Documentation

**10.272.4.1** `template<typename W > const W::psite & mln::p_centered< W >::center ( ) const [inline]`

Give the center of this site set.

**10.272.4.2** `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()`.

**10.272.4.3** `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()`.

**10.272.4.4** `template<typename W > std::size_t mln::p_centered< W >::memory_size ( ) const [inline]`

Return the size of this site set in memory.

**10.272.4.5** `template<typename W > const W & mln::p_centered< W >::window ( ) const`  
**[inline]**

Give the window this site set is defined upon.

## 10.273 mln::p\_complex< D, G > Class Template Reference

A complex psite set based on the N-faces of a complex of dimension D (a D-complex).

```
#include <p_complex.hh>
```

Inherits `site_set_base_< complex_psite< D, G >, p_complex< D, G > >`.

### Public Types

- typedef `super_::site element`  
*Associated types.*
- typedef `complex_psite< D, G > psite`  
*Point\_Site associated type.*
- typedef `p_complex_fwd_piter_< D, G > fwd_piter`  
*Forward Site\_Iterator associated type.*
- typedef `p_complex_bkd_piter_< D, G > bkd_piter`  
*Backward Site\_Iterator associated type.*
- typedef `fwd_piter piter`  
*Site\_Iterator associated type.*

### Public Member Functions

- bool `has (const psite &p) const`  
*Does this site set has p?*
- bool `is_valid () const`  
*Is this site set valid?*
- unsigned `nfaces () const`  
*Return the number of faces in the complex.*
- unsigned `nfaces_of_dim (unsigned n) const`  
*Return the number of n-faces in the complex.*
- unsigned `nsites () const`  
*Return The number of sites of the set, i.e., the number of faces.*
- `p_complex (const topo::complex< D > &cplx, const G &geom)`  
*Construct a complex psite set from a complex.*

- `topo::complex< D > & cplx () const`  
*Accessors.*
- `topo::complex< D > & cplx ()`  
*Return the complex associated to the `p_complex` domain (mutable version).*
- `const G & geom () const`  
*Return the geometry of the complex.*

### 10.273.1 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.

### 10.273.2 Member Typedef Documentation

**10.273.2.1** `template<unsigned D, typename G> typedef p_complex_bkd_piter_<D, G>  
mln::p_complex< D, G >::bkd_piter`

Backward `Site_Iterator` associated type.

**10.273.2.2** `template<unsigned D, typename G> typedef super_::site mln::p_complex< D, G  
>::element`

Associated types.

Element associated type.

**10.273.2.3** `template<unsigned D, typename G> typedef p_complex_fwd_piter_<D, G>  
mln::p_complex< D, G >::fwd_piter`

Forward `Site_Iterator` associated type.

**10.273.2.4** `template<unsigned D, typename G> typedef fwd_piter mln::p_complex< D, G  
>::piter`

`Site_Iterator` associated type.



**10.273.2.5** `template<unsigned D, typename G> typedef complex_psite<D, G> mln::p_complex<D, G>::psite`

[Point\\_Site](#) associated type.

### 10.273.3 Constructor & Destructor Documentation

**10.273.3.1** `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

### 10.273.4 Member Function Documentation

**10.273.4.1** `template<unsigned D, typename G> topo::complex< D > & mln::p_complex< D, G >::cplx ( ) const`

Accessors.

Return the complex associated to the [p\\_complex](#) domain (const version)

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

Referenced by `mln::complex_psite< D, G >::change_target()`, `mln::complex_psite< D, G >::complex_psite()`, and `mln::operator==( )`.

**10.273.4.2** `template<unsigned D, typename G> topo::complex< D > & mln::p_complex< D, G >::cplx ( )`

Return the complex associated to the [p\\_complex](#) domain (mutable version).

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

**10.273.4.3** `template<unsigned D, typename G> const G & mln::p_complex< D, G >::geom ( ) const`

Return the geometry of the complex.

**10.273.4.4** `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()`.

**10.273.4.5** `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()`.

**10.273.4.6** `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()`.

**10.273.4.7** `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.

**10.273.4.8** `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()`.

## 10.274 mln::p\_edges< G, F > Class Template Reference

[Site](#) set mapping graph edges and image sites.

```
#include <p_edges.hh>
```

Inherits `site_set_base_< F::result, p_edges< G, F > >`.

### Public Types

- typedef `util::edge< G > edge`  
*Type of graph edge.*
- typedef `F fun_t`  
*Function associated type.*
- typedef `util::edge< G > graph_element`  
*Type of graph element this site set focuses on.*
- typedef `G graph_t`  
*Graph associated type.*

- typedef super\_::site **element**  
*Associated types.*
- typedef p\_edges\_psite< G, F > **psite**  
*Point\_Site associated type.*
- typedef p\_graph\_piter< self\_, mln\_edge\_fwd\_iter(G) > **fwd\_piter**  
*Forward Site\_Iterator associated type.*
- typedef p\_graph\_piter< self\_, mln\_edge\_bkd\_iter(G) > **bkd\_piter**  
*Backward Site\_Iterator associated type.*
- typedef fwd\_piter **piter**  
*Site\_Iterator associated type.*

## Public Member Functions

- bool **has** (const **psite** &p) const  
*Does this site set has site p?*
- template<typename G2 >  
bool **has** (const util::edge< G2 > &e) const  
*Does this site set has edge e?*
- void **invalidate** ()  
*Invalidate this site set.*
- bool **is\_valid** () const  
*Is this site set valid?*
- std::size\_t **memory\_size** () const  
*Does this site set has vertex\_id? FIXME: causes ambiguities while calling has(mln::neighb\_fwd\_niter<>);  
bool has(unsigned vertex\_id) const;.*
- unsigned **nedges** () const  
*Return The number of edges in the graph.*
- unsigned **nsites** () const  
*Return The number of points (sites) of the set, i.e., the number of edges.*
- **p\_edges** ()  
*Constructors  
Default constructor.*
- **p\_edges** (const **Graph**< G > &gr)  
*Construct a graph edge psite set from a graph.*
- **p\_edges** (const **Graph**< G > &gr, const **Function**< F > &f)  
*Construct a graph edge psite set from a graph and a function.*

- `template<typename F2 >`  
`p_edges` (const `Graph`< `G` > &gr, const `Function`< `F2` > &f)  
*Construct a graph edge psite set from a graph and a function.*
- `const G & graph () const`  
*Accessors.*
- `const F & function () const`  
*Return the mapping function.*

### 10.274.1 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.

### 10.274.2 Member Typedef Documentation

**10.274.2.1** `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.

**10.274.2.2** `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.

**10.274.2.3** `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.

**10.274.2.4** `template<typename G, typename F = util::internal::id2element<G,util::edge<G> >>`  
`typedef F mln::p_edges< G, F >::fun_t`

[Function](#) associated type.

**10.274.2.5** `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.

**10.274.2.6** `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.

**10.274.2.7** `template<typename G, typename F = util::internal::id2element<G,util::edge<G>>>`  
`typedef G mln::p_edges< G, F >::graph_t`

[Graph](#) associated type.

**10.274.2.8** `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.

**10.274.2.9** `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.

### 10.274.3 Constructor & Destructor Documentation

**10.274.3.1** `template<typename G , typename F > mln::p_edges< G, F >::p_edges ( )`  
`[inline]`

Constructors

Default constructor.

**10.274.3.2** `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()`.

**10.274.3.3** `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()`.

**10.274.3.4** `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()`.

### 10.274.4 Member Function Documentation

**10.274.4.1** `template<typename G , typename F > const F & mln::p_edges< G, F >::function ( ) const [inline]`

Return the mapping function.

**10.274.4.2** `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==( )`.

**10.274.4.3** `template<typename G , typename F > bool mln::p_edges< G, F >::has ( const psite & p ) const [inline]`

Does this site set has site *p*?

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

**10.274.4.4** `template<typename G , typename F > template<typename G2 > bool mln::p_edges< G, F >::has ( const util::edge< G2 > & e ) const [inline]`

Does this site set has edge *e*?

References `mln::util::edge< G >::graph()`, `mln::util::edge< G >::is_valid()`, and `mln::p_edges< G, F >::is_valid()`.

**10.274.4.5** `template<typename G , typename F > void mln::p_edges< G, F >::invalidate ( ) [inline]`

Invalidate this site set.

**10.274.4.6** `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().

**10.274.4.7** `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;.

**10.274.4.8** `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().

**10.274.4.9** `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().

## 10.275 mln::p\_faces< N, D, P > Struct Template Reference

A complex psite set based on a the N-faces of a complex of dimension D (a D-complex).

```
#include <p_faces.hh>
```

Inherits site\_set\_base\_< faces\_psite< N, D, P >, p\_faces< N, D, P > >.

### Public Types

- typedef super\_::site [element](#)  
*Associated types.*
- typedef [faces\\_psite](#)< N, D, P > [psite](#)  
*Point\_Site associated type.*
- typedef p\_faces\_fwd\_piter\_< N, D, P > [fwd\\_piter](#)  
*Forward Site\_Iterator associated type.*
- typedef p\_faces\_bkd\_piter\_< N, D, P > [bkd\\_piter](#)  
*Backward Site\_Iterator associated type.*
- typedef [fwd\\_piter](#) [piter](#)  
*Site\_Iterator associated type.*

## Public Member Functions

- `bool is_valid () const`  
*Is this site set valid?*
- `unsigned nfaces () const`  
*Return The number of faces in the complex.*
- `unsigned nsites () const`  
*Return The number of sites of the set, i.e., the number of faces.*
- `p_faces (const topo::complex< D > &cplx)`  
*Construct a faces psite set from an mln::complex.*
- `p_faces (const p_complex< D, P > &pc)`  
*Construct a faces psite set from an mln::p\_complex.*
- `topo::complex< D > & cplx () const`  
*Accessors.*
- `topo::complex< D > & cplx ()`  
*Return the complex associated to the p\_faces domain (mutable version).*

### 10.275.1 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).

### 10.275.2 Member Typedef Documentation

**10.275.2.1** `template<unsigned N, unsigned D, typename P> typedef p_faces_bkd_piter_<N, D, P> mln::p_faces< N, D, P >::bkd_piter`

Backward [Site\\_Iterator](#) associated type.

**10.275.2.2** `template<unsigned N, unsigned D, typename P> typedef super_::site mln::p_faces< N, D, P >::element`

Associated types.

Element associated type.

**10.275.2.3** `template<unsigned N, unsigned D, typename P> typedef p_faces_fwd_piter_<N, D, P> mln::p_faces< N, D, P >::fwd_piter`

Forward [Site\\_Iterator](#) associated type.



**10.275.2.4** `template<unsigned N, unsigned D, typename P> typedef fwd_piter mln::p_faces< N, D, P >::piter`

[Site\\_Iterator](#) associated type.

**10.275.2.5** `template<unsigned N, unsigned D, typename P> typedef faces_psite<N, D, P> mln::p_faces< N, D, P >::psite`

[Point\\_Site](#) associated type.

### 10.275.3 Constructor & Destructor Documentation

**10.275.3.1** `template<unsigned N, unsigned D, typename P > mln::p_faces< N, D, P >::p_faces ( const topo::complex< D > & cplx ) [inline]`

Construct a faces psite set from an `mln::complex`.

#### Parameters

*cplx* The complex upon which the complex psite set is built.

**10.275.3.2** `template<unsigned N, unsigned D, typename P > mln::p_faces< N, D, P >::p_faces ( const p_complex< D, P > & pc ) [inline]`

Construct a faces psite set from an [mln::p\\_complex](#).

#### Parameters

*pc* The complex upon which the complex psite set is built.

### 10.275.4 Member Function Documentation

**10.275.4.1** `template<unsigned N, unsigned D, typename P > topo::complex< D > & mln::p_faces< N, D, P >::cplx ( ) const`

Accessors.

Return the complex associated to the [p\\_faces](#) domain (const version).

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

Referenced by `mln::faces_psite< N, D, P >::change_target()`, and `mln::operator==()`.

**10.275.4.2** `template<unsigned N, unsigned D, typename P > topo::complex< D > & mln::p_faces< N, D, P >::cplx ( )`

Return the complex associated to the [p\\_faces](#) domain (mutable version).

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

**10.275.4.3** `template<unsigned N, unsigned D, typename P > bool mln::p_faces< N, D, P >::is_valid ( ) const [inline]`

Is this site set valid?

Referenced by `mln::p_faces< N, D, P >::cplx()`.

**10.275.4.4** `template<unsigned N, unsigned D, typename P > unsigned mln::p_faces< N, D, P >::nfaces ( ) const [inline]`

Return The number of faces in the complex.

Referenced by `mln::p_faces< N, D, P >::nsites()`.

**10.275.4.5** `template<unsigned N, unsigned D, typename P > unsigned mln::p_faces< N, D, P >::nsites ( ) const [inline]`

Return The number of sites of the set, i.e., the number of *faces*.

(Required by the `mln::Site_Set` concept, since the property trait `site_set::nsites::known` of this site set is set to 'known'.)

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

## 10.276 `mln::p_graph_piter< S, I >` Class Template Reference

Generic iterator on point sites of a `mln::S`.

```
#include <p_graph_piter.hh>
```

Inherits `site_set_iterator_base< S, p_graph_piter< S, I > >`.

### Public Member Functions

- `const S::graph_t & graph () const`  
*Return the graph associated to the target S.*
- `unsigned id () const`  
*Return the graph element id.*
- `mln_q_subject (iter) element()`  
*Return the underlying graph element.*
- `void next ()`  
*Go to the next element.*
- `p_graph_piter ()`  
*Constructors.*

### 10.276.1 Detailed Description

`template<typename S, typename I> class mln::p_graph_piter< S, I >`

Generic iterator on point sites of a mln::S.

### 10.276.2 Constructor & Destructor Documentation

**10.276.2.1** `template<typename S, typename I > mln::p_graph_piter< S, I >::p_graph_piter ( )`  
**[inline]**

Constructors.

### 10.276.3 Member Function Documentation

**10.276.3.1** `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.

**10.276.3.2** `template<typename S, typename I > unsigned mln::p_graph_piter< S, I >::id ( ) const` **[inline]**

Return the graph element id.

**10.276.3.3** `template<typename S, typename I > mln::p_graph_piter< S, I >::mln_q_subject ( iter )`

Return the underlying graph element.

**10.276.3.4** `void mln::Site_Iterator< p_graph_piter< S, I > >::next ( )` **[inherited]**

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

#### Precondition

The iterator is valid.

## 10.277 mln::p\_if< S, F > Class Template Reference

[Site](#) set restricted w.r.t.

```
#include <p_if.hh>
```

Inherits `site_set_base_< S::psite, p_if< S, F > >`.

## Public Types

- typedef `p_if_piter_< typename S::bkd_piter, S, F >` `bkd_piter`  
*Backward [Site\\_Iterator](#) associated type.*
- typedef `S::element` `element`  
*Element associated type.*
- typedef `p_if_piter_< typename S::fwd_piter, S, F >` `fwd_piter`  
*Forward [Site\\_Iterator](#) associated type.*
- typedef `fwd_piter` `piter`  
*[Site\\_Iterator](#) associated type.*
- typedef `S::psite` `psite`  
*Psite associated type.*

## Public Member Functions

- `bool has (const psite &p) const`  
*Test if `p` belongs to the subset.*
- `bool is_valid () const`  
*Test if this site set is valid.*
- `std::size_t memory_size () const`  
*Return the size of this site set in memory.*
- `const S & overset () const`  
*Give the primary overset.*
- `p_if ()`  
*Constructor without argument.*
- `p_if (const S &s, const F &f)`  
*Constructor with a site set `s` and a predicate `f`.*
- `bool pred (const psite &p) const`  
*Test predicate on point site `p`.*
- `const F & predicate () const`  
*Give the predicate function.*

### 10.277.1 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.

## 10.277.2 Member Typedef Documentation

**10.277.2.1** `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.

**10.277.2.2** `template<typename S, typename F> typedef S ::element mln::p_if< S, F >::element`

Element associated type.

**10.277.2.3** `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.

**10.277.2.4** `template<typename S, typename F> typedef fwd_piter mln::p_if< S, F >::piter`

[Site\\_Iterator](#) associated type.

**10.277.2.5** `template<typename S, typename F> typedef S ::psite mln::p_if< S, F >::psite`

Psite associated type.

## 10.277.3 Constructor & Destructor Documentation

**10.277.3.1** `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*.

**10.277.3.2** `template<typename S, typename F > mln::p_if< S, F >::p_if ( ) [inline]`

Constructor without argument.

## 10.277.4 Member Function Documentation

**10.277.4.1** `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()`.

**10.277.4.2** `template<typename S, typename F> bool mln::p_if< S, F >::is_valid ( ) const`  
`[inline]`

Test if this site set is valid.

**10.277.4.3** `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.

**10.277.4.4** `template<typename S, typename F> const S & mln::p_if< S, F >::overset ( ) const`  
`[inline]`

Give the primary overset.

**10.277.4.5** `template<typename S, typename F> bool mln::p_if< S, F >::pred ( const psite & p`  
`) const [inline]`

Test predicate on point site p.

**10.277.4.6** `template<typename S, typename F> const F & mln::p_if< S, F >::predicate ( )`  
`const [inline]`

Give the predicate function.

## 10.278 mln::p\_image< I > Class Template Reference

[Site](#) set based on an image of Booleans.

```
#include <p_image.hh>
```

Inherits `site_set_base_< I::psite, p_image< I > >`.

### Public Types

- typedef `S::bkd_piter` `bkd_piter`  
*Backward [Site\\_Iterator](#) associated type.*
- typedef `I::psite` `element`  
*Element associated type.*
- typedef `S::fwd_piter` `fwd_piter`  
*Forward [Site\\_Iterator](#) associated type.*
- typedef `psite` `i_element`  
*Insertion element associated type.*
- typedef `S::piter` `piter`

*Site\_Iterator* associated type.

- typedef I::psite **psite**  
*Psite associated type.*
- typedef psite r\_element  
*Removal element associated type.*
- typedef internal::p\_image\_site\_set< I >::ret **S**  
*Equivalent site\_set type.*

## Public Member Functions

- void **clear** ()  
*Clear this set.*
- bool **has** (const psite &) const  
*Test if the psite p belongs to this site set.*
- void **insert** (const psite &p)  
*Insert a site p.*
- bool **is\_valid** () const  
*Test if this site set is valid, i.e., initialized.*
- std::size\_t **memory\_size** () const  
*Return the size of this site set in memory.*
- unsigned **nsites** () const  
*Give the number of sites.*
- operator typename internal::p\_image\_site\_set< I >::ret () const  
*Conversion towards the equivalent site set.*
- **p\_image** ()  
*Constructor without argument.*
- **p\_image** (const I &ima)  
*Constructor.*
- void **remove** (const psite &p)  
*Remove a site p.*
- void **toggle** (const psite &p)  
*Change the status in/out of a site p.*

## 10.278.1 Detailed Description

`template<typename I> class mln::p_image< I >`

[Site](#) set based on an image of Booleans.

## 10.278.2 Member Typedef Documentation

**10.278.2.1** `template<typename I > typedef S ::bkd_piter mln::p_image< I >::bkd_piter`

Backward [Site\\_Iterator](#) associated type.

**10.278.2.2** `template<typename I > typedef I ::psite mln::p_image< I >::element`

Element associated type.

**10.278.2.3** `template<typename I > typedef S ::fwd_piter mln::p_image< I >::fwd_piter`

Forward [Site\\_Iterator](#) associated type.

**10.278.2.4** `template<typename I > typedef psite mln::p_image< I >::i_element`

Insertion element associated type.

**10.278.2.5** `template<typename I > typedef S ::piter mln::p_image< I >::piter`

[Site\\_Iterator](#) associated type.

**10.278.2.6** `template<typename I > typedef I ::psite mln::p_image< I >::psite`

Psite associated type.

**10.278.2.7** `template<typename I > typedef psite mln::p_image< I >::r_element`

Removal element associated type.

**10.278.2.8** `template<typename I > typedef internal::p_image_site_set<I>::ret mln::p_image< I >::S`

Equivalent `site_set` type.

## 10.278.3 Constructor & Destructor Documentation

**10.278.3.1** `template<typename I > mln::p_image< I >::p_image ( ) [inline]`

Constructor without argument.



**10.278.3.2** `template<typename I> mln::p_image< I >::p_image ( const I & ima ) [inline]`

Constructor.

References mln::p\_image< I >::clear().

**10.278.4 Member Function Documentation****10.278.4.1** `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().

**10.278.4.2** `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().

**10.278.4.3** `template<typename I> void mln::p_image< I >::insert ( const psite & p ) [inline]`

Insert a site p.

References mln::p\_image< I >::is\_valid().

**10.278.4.4** `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().

**10.278.4.5** `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().

**10.278.4.6** `template<typename I> unsigned mln::p_image< I >::nsites ( ) const [inline]`

Give the number of sites.

**10.278.4.7** `template<typename I> mln::p_image< I >::operator typename internal::p_image_site_set< I >::ret ( ) const [inline]`

Conversion towards the equivalent site set.

**10.278.4.8** `template<typename I> void mln::p_image< I >::remove ( const psite & p )`  
**[inline]**

Remove a site p.

References `mln::p_image< I >::is_valid()`.

**10.278.4.9** `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()`.

## 10.279 mln::p\_indexed\_bkd\_piter< S > Class Template Reference

Backward iterator on sites of an indexed site set.

```
#include <p_array.hh>
```

Inherits `site_set_iterator_base< S, p_indexed_bkd_piter< S > >`.

### Public Member Functions

- `int index () const`  
*Return the current index.*
- `void next ()`  
*Go to the next element.*
- `p_indexed_bkd_piter (const S &s)`  
*Constructor.*
- `p_indexed_bkd_piter ()`  
*Constructor with no argument.*

### 10.279.1 Detailed Description

```
template<typename S> class mln::p_indexed_bkd_piter< S >
```

Backward iterator on sites of an indexed site set.

### 10.279.2 Constructor & Destructor Documentation

**10.279.2.1** `template<typename S> mln::p_indexed_bkd_piter< S >::p_indexed_bkd_piter ( )`  
**[inline]**

Constructor with no argument.

**10.279.2.2** `template<typename S > mln::p_indexed_bkd_piter< S >::p_indexed_bkd_piter ( const S & s ) [inline]`

Constructor.

### 10.279.3 Member Function Documentation

**10.279.3.1** `template<typename S > int mln::p_indexed_bkd_piter< S >::index ( ) const [inline]`

Return the current index.

**10.279.3.2** `void mln::Site_Iterator< p_indexed_bkd_piter< S > >::next ( ) [inherited]`

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

#### Precondition

The iterator is valid.

## 10.280 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 ()`  
*Constructor with no argument.*
- `p_indexed_fwd_piter (const S &s)`  
*Constructor.*

### 10.280.1 Detailed Description

```
template<typename S> class mln::p_indexed_fwd_piter< S >
```

Forward iterator on sites of an indexed site set.

## 10.280.2 Constructor & Destructor Documentation

**10.280.2.1** `template<typename S> mln::p_indexed_fwd_piter< S >::p_indexed_fwd_piter ( )`  
`[inline]`

Constructor with no argument.

**10.280.2.2** `template<typename S> mln::p_indexed_fwd_piter< S >::p_indexed_fwd_piter (`  
`const S & s ) [inline]`

Constructor.

## 10.280.3 Member Function Documentation

**10.280.3.1** `template<typename S> int mln::p_indexed_fwd_piter< S >::index ( ) const`  
`[inline]`

Return the current index.

## 10.281 mln::p\_indexed\_psite< S > Class Template Reference

Psite class for indexed site sets such as [p\\_array](#).

```
#include <p_array.hh>
```

Inherits `pseudo_site_base_< const S::element &, p_indexed_psite< S >>`.

### 10.281.1 Detailed Description

```
template<typename S> class mln::p_indexed_psite< S >
```

Psite class for indexed site sets such as [p\\_array](#). .

## 10.282 mln::p\_key< K, P > Class Template Reference

Priority queue class.

```
#include <p_key.hh>
```

Inherits `site_set_base_< P, p_key< K, P >>`.

### Public Types

- typedef `p_double_piter< self_, mln_bkd_eiter(util::set< K >), typename p_set< P >::bkd_piter >`  
`bkd_piter`  
*Backward [Site Iterator](#) associated type.*
- typedef `P element`  
*Element associated type.*

- typedef p\_double\_piter< self\_, mln\_fwd\_eiter(util::set< K >), typename p\_set< P >::fwd\_piter > fwd\_piter  
*Forward Site\_Iterator associated type.*
- typedef std::pair< K, P > i\_element  
*Insertion element associated type.*
- typedef fwd\_piter piter  
*Site\_Iterator associated type.*
- typedef p\_double\_psite< self\_, p\_set< P > > psite  
*Psite associated type.*
- typedef P r\_element  
*Removal element associated type.*

## Public Member Functions

- void change\_key (const K &k, const K &new\_k)  
*Change the key k into a new value new\_k.*
- template<typename F >  
void change\_keys (const Function\_v2v< F > &f)  
*Change the keys by applying the function f.*
- void clear ()  
*Clear this site set.*
- bool exists\_key (const K &key) const  
*Test if the priority exists.*
- bool has (const psite &) const  
*Test is the psite p belongs to this site set.*
- bool has (const P &p) const  
*Test is the psite p belongs to this site set.*
- void insert (const i\_element &k\_p)  
*Insert a pair k\_p (key k, site p).*
- void insert (const K &k, const P &p)  
*Insert a pair (key k, site p).*
- bool is\_valid () const  
*Test this set validity so returns always true.*
- const K & key (const P &p) const

*Give the key associated with site  $p$ .*

- const `util::set< K > & keys ()` const  
*Give the set of keys.*
- `std::size_t memory_size ()` const  
*Return the size of this site set in memory.*
- unsigned `nsites ()` const  
*Give the number of sites.*
- const `p_set< P > & operator()` (const K &key) const  
*Give the queue with the priority  $priority$ .*
- `p_key ()`  
*Constructor.*
- void `remove (const P &p)`  
*Remove a site  $p$ .*
- void `remove_key (const K &k)`  
*Remove all sites with key  $k$ .*

### 10.282.1 Detailed Description

`template<typename K, typename P> class mln::p_key< K, P >`

Priority queue class.

### 10.282.2 Member Typedef Documentation

**10.282.2.1** `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.

**10.282.2.2** `template<typename K , typename P > typedef P mln::p_key< K, P >::element`

Element associated type.

**10.282.2.3** `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.

**10.282.2.4** `template<typename K , typename P > typedef std::pair<K,P> mln::p_key< K, P >::i_element`

Insertion element associated type.

**10.282.2.5** `template<typename K , typename P > typedef fwd_piter mln::p_key< K, P >::piter`

[Site\\_Iterator](#) associated type.

**10.282.2.6** `template<typename K , typename P > typedef p_double_psite< self_, p_set<P> > mln::p_key< K, P >::psite`

Psite associated type.

**10.282.2.7** `template<typename K , typename P > typedef P mln::p_key< K, P >::r_element`

Removal element associated type.

### 10.282.3 Constructor & Destructor Documentation

**10.282.3.1** `template<typename K , typename P > mln::p_key< K, P >::p_key ( ) [inline]`

Constructor.

### 10.282.4 Member Function Documentation

**10.282.4.1** `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()`.

**10.282.4.2** `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()`.

**10.282.4.3** `template<typename K , typename P > void mln::p_key< K, P >::clear ( ) [inline]`

Clear this site set.

**10.282.4.4** `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()()`.

**10.282.4.5** `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.

**10.282.4.6** `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()`.

**10.282.4.7** `template<typename K , typename P > void mln::p_key< K, P >::insert ( const i_element & k_p ) [inline]`

Insert a pair `k_p` (key `k`, site `p`).

**10.282.4.8** `template<typename K , typename P > void mln::p_key< K, P >::insert ( const K & k, const P & p ) [inline]`

Insert a pair (key `k`, site `p`).

References `mln::p_key< K, P >::has()`.

**10.282.4.9** `template<typename K , typename P > bool mln::p_key< K, P >::is_valid ( ) const [inline]`

Test this set validity so returns always true.

**10.282.4.10** `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`.

**10.282.4.11** `template<typename K , typename P > const util::set< K > & mln::p_key< K, P >::keys ( ) const [inline]`

Give the set of keys.

**10.282.4.12** `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.



**10.282.4.13** `template<typename K , typename P > unsigned mln::p_key< K, P >::nsites ( )  
const [inline]`

Give the number of sites.

**10.282.4.14** `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()`.

**10.282.4.15** `template<typename K , typename P > void mln::p_key< K, P >::remove ( const P &  
p ) [inline]`

Remove a site `p`.

**10.282.4.16** `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()`.

## 10.283 mln::p\_line2d Class Reference

2D discrete line of points.

```
#include <p_line2d.hh>
```

Inherits `site_set_base_< point2d, p_line2d >`.

### Public Types

- typedef `p_indexed_bkd_piter< self_ > bkd_piter`  
*Backward [Site\\_Iterator](#) associated type.*
- typedef `point2d element`  
*Element associated type.*
- typedef `p_indexed_fwd_piter< self_ > fwd_piter`  
*Forward [Site\\_Iterator](#) associated type.*
- typedef `p_indexed_fwd_piter< self_ > piter`  
*[Site\\_Iterator](#) associated type.*
- typedef `p_indexed_psite< self_ > psite`  
*Psite associated type.*

- typedef const [box2d](#) & [q\\_box](#)  
*Box (qualified) associated type.*

## Public Member Functions

- const [box2d](#) & [bbox](#) () const  
*Give the exact bounding box.*
- const [point2d](#) & [begin](#) () const  
*Give the point that begins the line.*
- const [point2d](#) & [end](#) () const  
*Give the point that ends the line.*
- bool [has](#) (const util::index &i) const  
*Test if index  $i$  belongs to this point set.*
- bool [has](#) (const [psite](#) &p) const  
*Test if  $p$  belongs to this point set.*
- bool [is\\_valid](#) () const  
*Test if this line is valid, i.e., initialized.*
- std::size\_t [memory\\_size](#) () const  
*Return the size of this site set in memory.*
- unsigned [nsites](#) () const  
*Give the number of points.*
- const [point2d](#) & [operator\[\]](#) (unsigned i) const  
*Return the  $i$ -th point of the line.*
- [p\\_line2d](#) (const [point2d](#) &beg, const [point2d](#) &end, bool is\_end\_excluded=false)  
*Constructor from point  $beg$  to point  $end$ .*
- [p\\_line2d](#) ()  
*Constructor without argument.*
- const std::vector< [point2d](#) > & [std\\_vector](#) () const  
*Return the corresponding std::vector of points.*

### 10.283.1 Detailed Description

2D discrete line of points. It is based on [p\\_array](#).

## 10.283.2 Member Typedef Documentation

### 10.283.2.1 typedef p\_indexed\_bkd\_piter<self\_> mln::p\_line2d::bkd\_piter

Backward [Site\\_Iterator](#) associated type.

### 10.283.2.2 typedef point2d mln::p\_line2d::element

Element associated type.

### 10.283.2.3 typedef p\_indexed\_fwd\_piter<self\_> mln::p\_line2d::fwd\_piter

Forward [Site\\_Iterator](#) associated type.

### 10.283.2.4 typedef p\_indexed\_fwd\_piter<self\_> mln::p\_line2d::piter

[Site\\_Iterator](#) associated type.

### 10.283.2.5 typedef p\_indexed\_psite<self\_> mln::p\_line2d::psite

Psite associated type.

### 10.283.2.6 typedef const box2d& mln::p\_line2d::q\_box

[Box](#) (qualified) associated type.

## 10.283.3 Constructor & Destructor Documentation

### 10.283.3.1 mln::p\_line2d::p\_line2d ( ) [inline]

Constructor without argument.

References [is\\_valid\(\)](#).

### 10.283.3.2 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\(\)](#).

## 10.283.4 Member Function Documentation

### 10.283.4.1 const box2d & mln::p\_line2d::bbox ( ) const [inline]

Give the exact bounding box.

References [is\\_valid\(\)](#).

**10.283.4.2** `const point2d & mln::p_line2d::begin ( ) const [inline]`

Give the point that begins the line.

References `is_valid()`.

Referenced by `mln::debug::draw_graph()`.

**10.283.4.3** `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()`.

**10.283.4.4** `bool mln::p_line2d::has ( const psite & p ) const [inline]`

Test if `p` belongs to this point set.

**10.283.4.5** `bool mln::p_line2d::has ( const util::index & i ) const [inline]`

Test if index `i` belongs to this point set.

References `nsites()`.

**10.283.4.6** `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()`.

**10.283.4.7** `std::size_t mln::p_line2d::memory_size ( ) const [inline]`

Return the size of this site set in memory.

**10.283.4.8** `unsigned mln::p_line2d::nsites ( ) const [inline]`

Give the number of points.

Referenced by `end()`, `has()`, and `operator[]()`.

**10.283.4.9** `const point2d & mln::p_line2d::operator[] ( unsigned i ) const [inline]`

Return the `i`-th point of the line.

References `nsites()`.

**10.283.4.10** `const std::vector< point2d > & mln::p_line2d::std_vector ( ) const [inline]`

Return the corresponding `std::vector` of points.

## 10.284 mln::p\_mutable\_array\_of< S > Class Template Reference

[p\\_mutable\\_array\\_of](#) is a mutable array of site sets.

```
#include <p_mutable_array_of.hh>
```

Inherits [site\\_set\\_base\\_< S::site, p\\_mutable\\_array\\_of< S > >](#).

### Public Types

- typedef [p\\_double\\_piter< self\\_, mln\\_bkd\\_eiter\(array\\_\), typename S::bkd\\_piter > bkd\\_piter](#)  
*Backward [Site\\_Iterator](#) associated type.*
- typedef [S element](#)  
*Element associated type.*
- typedef [p\\_double\\_piter< self\\_, mln\\_fwd\\_eiter\(array\\_\), typename S::fwd\\_piter > fwd\\_piter](#)  
*Forward [Site\\_Iterator](#) associated type.*
- typedef [S i\\_element](#)  
*Insertion element associated type.*
- typedef [fwd\\_piter piter](#)  
*[Site\\_Iterator](#) associated type.*
- typedef [p\\_double\\_psite< self\\_, element > psite](#)  
*Psite associated type.*

### Public Member Functions

- void [clear](#) ()  
*Clear this set.*
- bool [has](#) (const [psite](#) &p) const  
*Test if *p* belongs to this point set.*
- void [insert](#) (const [S](#) &s)  
*Insert a site set *s*.*
- bool [is\\_valid](#) () const  
*Test this set validity so returns always true.*
- [std::size\\_t](#) [memory\\_size](#) () const  
*Return the size of this site set in memory.*
- unsigned [nelements](#) () const  
*Give the number of elements (site sets) of this composite.*
- [S](#) & [operator\[\]](#) (unsigned i)

*Return the  $i$ -th site set (mutable version).*

- `const S & operator[]` (unsigned  $i$ ) `const`  
*Return the  $i$ -th site set (const version).*
- `p_mutable_array_of` ()  
*Constructor without arguments.*
- `void reserve` (unsigned  $n$ )  
*Reserve memory for  $n$  elements.*

### 10.284.1 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.

### 10.284.2 Member Typedef Documentation

**10.284.2.1** `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.

**10.284.2.2** `template<typename S > typedef S mln::p_mutable_array_of< S >::element`

Element associated type.

**10.284.2.3** `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.

**10.284.2.4** `template<typename S > typedef S mln::p_mutable_array_of< S >::i_element`

Insertion element associated type.

**10.284.2.5** `template<typename S > typedef fwd_piter mln::p_mutable_array_of< S >::piter`

[Site\\_Iterator](#) associated type.

**10.284.2.6** `template<typename S > typedef p_double_psite<self_, element>  
mln::p_mutable_array_of< S >::psite`

Psite associated type.

### 10.284.3 Constructor & Destructor Documentation

**10.284.3.1** `template<typename S > mln::p_mutable_array_of< S >::p_mutable_array_of ( )`  
`[inline]`

Constructor without arguments.

### 10.284.4 Member Function Documentation

**10.284.4.1** `template<typename S > void mln::p_mutable_array_of< S >::clear ( )` `[inline]`

Clear this set.

**10.284.4.2** `template<typename S > bool mln::p_mutable_array_of< S >::has ( const psite & p )`  
`const [inline]`

Test if *p* belongs to this point set.

**10.284.4.3** `template<typename S > void mln::p_mutable_array_of< S >::insert ( const S & s )`  
`[inline]`

Insert a site set *s*.

#### Precondition

*s* is valid.

**10.284.4.4** `template<typename S > bool mln::p_mutable_array_of< S >::is_valid ( ) const`  
`[inline]`

Test this set validity so returns always true.

**10.284.4.5** `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.

**10.284.4.6** `template<typename S > unsigned mln::p_mutable_array_of< S >::nelements ( )`  
`const [inline]`

Give the number of elements (site sets) of this composite.

**10.284.4.7** `template<typename S > S & mln::p_mutable_array_of< S >::operator[] ( unsigned i`  
`) [inline]`

Return the *i*-th site set (mutable version).

**10.284.4.8** `template<typename S> const S & mln::p_mutable_array_of< S >::operator[] ( unsigned i ) const [inline]`

Return the *i*-th site set (const version).

**10.284.4.9** `template<typename S> void mln::p_mutable_array_of< S >::reserve ( unsigned n ) [inline]`

Reserve memory for *n* elements.

## 10.285 mln::p\_n\_faces\_bkd\_piter< D, G > Class Template Reference

Backward iterator on the *n*-faces sites of an `mln::p_complex<D, G>`.

```
#include <p_n_faces_piter.hh>
```

Inherits `p_complex_piter_base_< topo::n_face_bkd_iter< D >, p_complex< D, G >, G::site, p_n_faces_bkd_piter< D, G >>`.

### Public Member Functions

- [p\\_n\\_faces\\_bkd\\_piter\(\)](#)  
*Construction and assignment.*
- `unsigned n() const`  
*Accessors.*

### 10.285.1 Detailed Description

`template<unsigned D, typename G> class mln::p_n_faces_bkd_piter< D, G >`

Backward iterator on the *n*-faces sites of an `mln::p_complex<D, G>`.

### 10.285.2 Constructor & Destructor Documentation

**10.285.2.1** `template<unsigned D, typename G> mln::p_n_faces_bkd_piter< D, G >::p_n_faces_bkd_piter ( ) [inline]`

Construction and assignment.

### 10.285.3 Member Function Documentation

**10.285.3.1** `template<unsigned D, typename G> unsigned mln::p_n_faces_bkd_piter< D, G >::n ( ) const [inline]`

Accessors.



Shortcuts to face\_'s accessors.

## 10.286 mln::p\_n\_faces\_fwd\_piter< D, G > Class Template Reference

Forward iterator on the n-faces sites of an mln::p\_complex<D, G>.

```
#include <p_n_faces_piter.hh>
```

Inherits p\_complex\_piter\_base\_< topo::n\_face\_fwd\_iter< D >, p\_complex< D, G >, G::site, p\_n\_faces\_fwd\_piter< D, G > >.

### Public Member Functions

- void [next](#) ()  
*Go to the next element.*
- [p\\_n\\_faces\\_fwd\\_piter](#) ()  
*Construction and assignment.*
- unsigned [n](#) () const  
*Accessors.*

### 10.286.1 Detailed Description

```
template<unsigned D, typename G> class mln::p_n_faces_fwd_piter< D, G >
```

Forward iterator on the n-faces sites of an mln::p\_complex<D, G>.

### 10.286.2 Constructor & Destructor Documentation

**10.286.2.1** `template<unsigned D, typename G > mln::p_n_faces_fwd_piter< D, G >::p_n_faces_fwd_piter ( ) [inline]`

Construction and assignment.

### 10.286.3 Member Function Documentation

**10.286.3.1** `template<unsigned D, typename G > unsigned mln::p_n_faces_fwd_piter< D, G >::n ( ) const [inline]`

Accessors.

Shortcuts to face\_'s accessors.

**10.286.3.2 void mln::Site\_Iterator< p\_n\_faces\_fwd\_piter< D, G >>::next( ) [inherited]**

Go to the next element.

**Warning**

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

**Precondition**

The iterator is valid.

**10.287 mln::p\_priority< P, Q > Class Template Reference**

Priority queue.

```
#include <p_priority.hh>
```

Inherits site\_set\_base\_< Q::site, p\_priority< P, Q >>.

**Public Types**

- typedef p\_double\_piter< [self\\_](#), mln\_fwd\_eiter(util::set< P >), typename Q::bkd\_piter > [bkd\\_piter](#)  
*Backward Site\_Iterator associated type.*
- typedef Q::element [element](#)  
*Element associated type.*
- typedef p\_double\_piter< [self\\_](#), mln\_bkd\_eiter(util::set< P >), typename Q::fwd\_piter > [fwd\\_piter](#)  
*Forward Site\_Iterator associated type.*
- typedef std::pair< P, [element](#) > [i\\_element](#)  
*Insertion element associated type.*
- typedef fwd\_piter [piter](#)  
*Site\_Iterator associated type.*
- typedef p\_double\_psite< [self\\_](#), Q > [psite](#)  
*Psite associated type.*

**Public Member Functions**

- void [clear](#) ()  
*Clear the queue.*
- bool [exists\\_priority](#) (const P &priority) const  
*Test if the priority exists.*

- const Q::element & **front** () const  
*Give an element with highest priority.*
- bool **has** (const psite &) const  
*Test is the psite p belongs to this site set.*
- const P **highest\_priority** () const  
*Give the highest priority.*
- void **insert** (const i\_element &p\_e)  
*Insert a pair p\_e (priority p, element e).*
- void **insert** (const p\_priority< P, Q > &other)  
*Insert elements from another priority queue.*
- bool **is\_valid** () const  
*Test this set validity so returns always true.*
- const P **lowest\_priority** () const  
*Give the lowest priority.*
- std::size\_t **memory\_size** () const  
*Return the size of this site set in memory.*
- unsigned **nsites** () const  
*Give the number of sites.*
- const Q & **operator()** (const P &priority) const  
*Give the queue with the priority priority.*
- **p\_priority** ()  
*Constructor.*
- void **pop** ()  
*Pop (remove) from the queue an element with highest priority.*
- Q::element **pop\_front** ()  
*Return an element with highest priority and remove it from the set.*
- const util::set< P > & **priorities** () const  
*Give the set of priorities.*
- void **push** (const P &priority, const element &e)  
*Push in the queue with priority the element e.*

## 10.287.1 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>`).

## 10.287.2 Member Typedef Documentation

**10.287.2.1** `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.

**10.287.2.2** `template<typename P, typename Q> typedef Q ::element mln::p_priority< P, Q >::element`

Element associated type.

**10.287.2.3** `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.

**10.287.2.4** `template<typename P, typename Q> typedef std::pair<P, element> mln::p_priority< P, Q >::i_element`

Insertion element associated type.

**10.287.2.5** `template<typename P, typename Q> typedef fwd_piter mln::p_priority< P, Q >::piter`

[Site\\_Iterator](#) associated type.

**10.287.2.6** `template<typename P, typename Q> typedef p_double_psite<self_, Q> mln::p_priority< P, Q >::psite`

Psite associated type.

## 10.287.3 Constructor & Destructor Documentation

**10.287.3.1** `template<typename P, typename Q > mln::p_priority< P, Q >::p_priority ( ) [inline]`

Constructor.

## 10.287.4 Member Function Documentation

**10.287.4.1** `template<typename P , typename Q > void mln::p_priority< P, Q >::clear ( )`  
`[inline]`

Clear the queue.

**10.287.4.2** `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()`.

**10.287.4.3** `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()`, and `mln::morpho::watershed::topological()`.

**10.287.4.4** `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.

**10.287.4.5** `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()`.

**10.287.4.6** `template<typename P , typename Q > void mln::p_priority< P, Q >::insert ( const`  
`i_element & p_e ) [inline]`

Insert a pair `p_e` (priority `p`, element `e`).

References `mln::p_priority< P, Q >::push()`.

**10.287.4.7** `template<typename P , typename Q > void mln::p_priority< P, Q >::insert ( const p_priority< P, Q > & other ) [inline]`

Insert elements from another priority queue.

**10.287.4.8** `template<typename P , typename Q > bool mln::p_priority< P, Q >::is_valid ( ) const [inline]`

Test this set validity so returns always true.

**10.287.4.9** `template<typename P , typename Q > const P mln::p_priority< P, Q >::lowest_priority ( ) const [inline]`

Give the lowest priority.

#### Precondition

! is\_empty()

**10.287.4.10** `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.

**10.287.4.11** `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().

**10.287.4.12** `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().

**10.287.4.13** `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(), and mln::morpho::watershed::topological().

**10.287.4.14** `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()`

**10.287.4.15** `template<typename P , typename Q > const util::set< P > & mln::p_priority< P, Q >::priorities ( ) const [inline]`

Give the set of priorities.

**10.287.4.16** `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::morpho::watershed::topological()`.

## 10.288 mln::p\_queue< P > Class Template Reference

Queue of sites (based on `std::deque`).

`#include <p_queue.hh>`

Inherits `site_set_base_< P, p_queue< P > >`.

### Public Types

- `typedef p_indexed_bkd_piter< self_ > bkd_piter`  
*Backward Site\_Iterator associated type.*
- `typedef P element`  
*Element associated type.*
- `typedef p_indexed_fwd_piter< self_ > fwd_piter`  
*Forward Site\_Iterator associated type.*
- `typedef P i_element`  
*Insertion element associated type.*
- `typedef fwd_piter piter`  
*Site\_Iterator associated type.*
- `typedef p_indexed_psite< self_ > psite`  
*Psite associated type.*

## Public Member Functions

- void `clear ()`  
*Clear the queue.*
- const P & `front () const`  
*Give the front site  $p$  of the queue;  $p$  is the least recently inserted site.*
- bool `has (const psite &p) const`  
*Test if  $p$  belongs to this site set.*
- bool `has (const util::index &i) const`  
*Test if index  $i$  belongs to this site set.*
- void `insert (const P &p)`  
*Insert a site  $p$  (equivalent as 'push').*
- bool `is_valid () const`  
*This set is always valid so it returns true.*
- std::size\_t `memory_size () const`  
*Return the size of this site set in memory.*
- unsigned `nsites () const`  
*Give the number of sites.*
- const P & `operator[] (unsigned i) const`  
*Return the  $i$ -th site.*
- `p_queue ()`  
*Constructor without argument.*
- void `pop ()`  
*Pop (remove) the front site  $p$  from the queue;  $p$  is the least recently inserted site.*
- P `pop_front ()`  
*Pop (remove) the front site  $p$  from the queue;  $p$  is the least recently inserted site and give the front site  $p$  of the queue;  $p$  is the least recently inserted site.*
- void `push (const P &p)`  
*Push a site  $p$  in the queue.*
- const std::deque< P > & `std_deque () const`  
*Return the corresponding `std::deque` of sites.*

### 10.288.1 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.



## 10.288.2 Member Typedef Documentation

**10.288.2.1** `template<typename P> typedef p_indexed_bkd_piter<self_> mln::p_queue< P >::bkd_piter`

Backward [Site\\_Iterator](#) associated type.

**10.288.2.2** `template<typename P> typedef P mln::p_queue< P >::element`

Element associated type.

**10.288.2.3** `template<typename P> typedef p_indexed_fwd_piter<self_> mln::p_queue< P >::fwd_piter`

Forward [Site\\_Iterator](#) associated type.

**10.288.2.4** `template<typename P> typedef P mln::p_queue< P >::i_element`

Insertion element associated type.

**10.288.2.5** `template<typename P> typedef fwd_piter mln::p_queue< P >::piter`

[Site\\_Iterator](#) associated type.

**10.288.2.6** `template<typename P> typedef p_indexed_psite<self_> mln::p_queue< P >::psite`

Psite associated type.

## 10.288.3 Constructor & Destructor Documentation

**10.288.3.1** `template<typename P > mln::p_queue< P >::p_queue ( ) [inline]`

Constructor without argument.

## 10.288.4 Member Function Documentation

**10.288.4.1** `template<typename P > void mln::p_queue< P >::clear ( ) [inline]`

Clear the queue.

**10.288.4.2** `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()`, and `mln::geom::impl::seeds2tiling()`.

**10.288.4.3** `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()`.

**10.288.4.4** `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()`.

**10.288.4.5** `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()`.

**10.288.4.6** `template<typename P> bool mln::p_queue< P >::is_valid ( ) const [inline]`

This set is always valid so it returns true.

**10.288.4.7** `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()`.

**10.288.4.8** `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[]()`.

**10.288.4.9** `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()`.

**10.288.4.10** `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()`, and `mln::geom::impl::seeds2tiling()`.

**10.288.4.11** `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()`.

**10.288.4.12** `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()`, and `mln::geom::impl::seeds2tiling()`.

**10.288.4.13** `template<typename P> const std::deque< P > & mln::p_queue< P >::std_deque ( ) const [inline]`

Return the corresponding `std::deque` of sites.

**10.289** `mln::p_queue_fast< P >` Class Template Reference

Queue of sites class (based on [p\\_array](#)).

```
#include <p_queue_fast.hh>
```

Inherits `site_set_base_< P, p_queue_fast< P >>`.

**Public Types**

- typedef `p_indexed_bkd_piter< self_ > bkd_piter`  
*Backward Site\_Iterator associated type.*
- typedef `P element`  
*Element associated type.*
- typedef `p_indexed_fwd_piter< self_ > fwd_piter`  
*Forward Site\_Iterator associated type.*
- typedef `P i_element`  
*Insertion element associated type.*
- typedef `fwd_piter piter`  
*Site\_Iterator associated type.*
- typedef `p_indexed_psite< self_ > psite`  
*Psite associated type.*

## Public Member Functions

- void `clear ()`  
*Clear the queue.*
- bool `compute_has (const P &p) const`  
*Test if  $p$  belongs to this site set.*
- bool `empty () const`  
*Test if the queue is empty.*
- const P & `front () const`  
*Give the front site  $p$  of the queue;  $p$  is the least recently inserted site.*
- bool `has (const util::index &i) const`  
*Test if index  $i$  belongs to this site set.*
- bool `has (const psite &p) const`  
*Test if  $p$  belongs to this site set.*
- void `insert (const P &p)`  
*Insert a site  $p$  (equivalent as 'push').*
- bool `is_valid () const`  
*This set is always valid so it returns true.*
- std::size\_t `memory_size () const`  
*Return the size of this site set in memory.*
- unsigned `nsites () const`  
*Give the number of sites.*
- const P & `operator[] (unsigned i) const`  
*Return the  $i$ -th site.*
- `p_queue_fast ()`  
*Constructor without argument.*
- void `pop ()`  
*Pop (remove) the front site  $p$  from the queue;  $p$  is the least recently inserted site.*
- const P & `pop_front ()`  
*Pop (remove) the front site  $p$  from the queue;  $p$  is the least recently inserted site and give the front site  $p$  of the queue;  $p$  is the least recently inserted site.*
- void `purge ()`  
*Purge the queue to save (free) some memory.*
- void `push (const P &p)`  
*Push a site  $p$  in the queue.*

- void [reserve](#) (typename [p\\_array](#)< P >::size\_type n)  
*Reserve n cells.*
- const std::vector< P > & [std\\_vector](#) () const  
*Return the corresponding std::vector of sites.*

## 10.289.1 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.

## 10.289.2 Member Typedef Documentation

**10.289.2.1** `template<typename P > typedef p_indexed_bkd_piter<self_> mln::p_queue_fast< P >::bkd_piter`

Backward [Site\\_Iterator](#) associated type.

**10.289.2.2** `template<typename P > typedef P mln::p_queue_fast< P >::element`

Element associated type.

**10.289.2.3** `template<typename P > typedef p_indexed_fwd_piter<self_> mln::p_queue_fast< P >::fwd_piter`

Forward [Site\\_Iterator](#) associated type.

**10.289.2.4** `template<typename P > typedef P mln::p_queue_fast< P >::i_element`

Insertion element associated type.

**10.289.2.5** `template<typename P > typedef fwd_piter mln::p_queue_fast< P >::piter`

[Site\\_Iterator](#) associated type.

**10.289.2.6** `template<typename P > typedef p_indexed_psite<self_> mln::p_queue_fast< P >::psite`

Psite associated type.

### 10.289.3 Constructor & Destructor Documentation

**10.289.3.1** `template<typename P> mln::p_queue_fast<P>::p_queue_fast ( ) [inline]`

Constructor without argument.

### 10.289.4 Member Function Documentation

**10.289.4.1** `template<typename P> void mln::p_queue_fast<P>::clear ( ) [inline]`

Clear the queue.

**10.289.4.2** `template<typename P> bool mln::p_queue_fast<P>::compute_has ( const P & p ) const [inline]`

Test if *p* belongs to this site set.

**10.289.4.3** `template<typename P> bool mln::p_queue_fast<P>::empty ( ) const [inline]`

Test if the queue is empty.

**10.289.4.4** `template<typename P> const P & mln::p_queue_fast<P>::front ( ) const [inline]`

Give the front site *p* of the queue; *p* is the least recently inserted site.

Referenced by `mln::p_queue_fast<P>::pop_front()`.

**10.289.4.5** `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()`.

**10.289.4.6** `template<typename P> bool mln::p_queue_fast<P>::has ( const util::index & i ) const [inline]`

Test if index *i* belongs to this site set.

References `mln::p_queue_fast<P>::nsites()`.

**10.289.4.7** `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()`.

**10.289.4.8** `template<typename P > bool mln::p_queue_fast< P >::is_valid ( ) const`  
`[inline]`

This set is always valid so it returns true.

**10.289.4.9** `template<typename P > std::size_t mln::p_queue_fast< P >::memory_size ( ) const`  
`[inline]`

Return the size of this site set in memory.

**10.289.4.10** `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[]().

**10.289.4.11** `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().

**10.289.4.12** `template<typename P > void mln::p_queue_fast< P >::pop ( ) [inline]`

Pop (remove) the front site *p* from the queue; *p* is the least recently inserted site.

Referenced by mln::p\_queue\_fast< P >::pop\_front().

**10.289.4.13** `template<typename P > const P & mln::p_queue_fast< P >::pop_front ( )`  
`[inline]`

Pop (remove) the front site *p* from the queue; *p* is the least recently inserted site and give the front site *p* of the queue; *p* is the least recently inserted site.

References mln::p\_queue\_fast< P >::front(), and mln::p\_queue\_fast< P >::pop().

**10.289.4.14** `template<typename P > void mln::p_queue_fast< P >::purge ( ) [inline]`

Purge the queue to save (free) some memory.

**10.289.4.15** `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().

**10.289.4.16** `template<typename P> void mln::p_queue_fast< P >::reserve ( typename p_array< P >::size_type n ) [inline]`

Reserve *n* cells.

**10.289.4.17** `template<typename P> const std::vector< P > & mln::p_queue_fast< P >::std_vector ( ) const [inline]`

Return the corresponding `std::vector` of sites.

## 10.290 mln::p\_run< P > Class Template Reference

[Point](#) set class in run.

```
#include <p_run.hh>
```

Inherits `site_set_base_< P, p_run< P >>`.

### Public Types

- `typedef p_run_bkd_piter_< P > bkd_piter`  
*Backward [Site\\_Iterator](#) associated type.*
- `typedef P element`  
*Element associated type.*
- `typedef p_run_fwd_piter_< P > fwd_piter`  
*Forward [Site\\_Iterator](#) associated type.*
- `typedef fwd_piter piter`  
*[Site\\_Iterator](#) associated type.*
- `typedef p_run_psite< P > psite`  
*[Psite](#) associated type.*
- `typedef mln::box< P > q_box`  
*[Box](#) associated type.*

### Public Member Functions

- `mln::box< P > bbox () const`  
*Give the exact bounding box.*
- `P end () const`  
*Return (compute) the ending point.*
- `bool has (const P &p) const`  
*Test if *p* belongs to this point set.*



- bool `has` (const `psite &p`) const  
*Test if  $p$  belongs to this point set.*
- bool `has_index` (unsigned short `i`) const  
*Test if index  $i$  belongs to this point set.*
- void `init` (const `P &start`, unsigned short `len`)  
*Set the starting point.*
- bool `is_valid` () const  
*Test if this run is valid, i.e., with length  $> 0$ .*
- unsigned short `length` () const  
*Give the length of the run.*
- `std::size_t` `memory_size` () const  
*Return the size of this site set in memory.*
- unsigned `nsites` () const  
*Give the number of sites.*
- `P` `operator[]` (unsigned short `i`) const  
*Return the  $i$ -th point.*
- `p_run` ()  
*Constructor without argument.*
- `p_run` (const `P &start`, unsigned short `len`)  
*Constructor.*
- `p_run` (const `P &start`, const `P &end`)  
*Constructor.*
- const `P & start` () const  
*Return the starting point.*

### 10.290.1 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.

### 10.290.2 Member Typedef Documentation

**10.290.2.1** `template<typename P> typedef p_run_bkd_piter_<P> mln::p_run< P >::bkd_piter`

Backward `Site_Iterator` associated type.

**10.290.2.2** `template<typename P> typedef P mln::p_run< P >::element`

Element associated type.

**10.290.2.3** `template<typename P> typedef p_run_fwd_piter_<P> mln::p_run< P >::fwd_piter`

Forward [Site\\_Iterator](#) associated type.

**10.290.2.4** `template<typename P> typedef fwd_piter mln::p_run< P >::piter`

[Site\\_Iterator](#) associated type.

**10.290.2.5** `template<typename P> typedef p_run_psite<P> mln::p_run< P >::psite`

Psite associated type.

**10.290.2.6** `template<typename P> typedef mln::box<P> mln::p_run< P >::q_box`

[Box](#) associated type.

**10.290.3** **Constructor & Destructor Documentation****10.290.3.1** `template<typename P> mln::p_run< P >::p_run ( ) [inline]`

Constructor without argument.

**10.290.3.2** `template<typename P> mln::p_run< P >::p_run ( const P & start, unsigned short len ) [inline]`

Constructor.

References `mln::p_run< P >::init()`.

**10.290.3.3** `template<typename P> mln::p_run< P >::p_run ( const P & start, const P & end ) [inline]`

Constructor.

**10.290.4** **Member Function Documentation****10.290.4.1** `template<typename P> mln::box< P > mln::p_run< P >::bbox ( ) const [inline]`

Give the exact bounding box.

References `mln::p_run< P >::end()`.

**10.290.4.2** `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()`.

**10.290.4.3** `template<typename P> bool mln::p_run< P >::has ( const psite & p ) const [inline]`

Test if `p` belongs to this point set.

**10.290.4.4** `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()`.

**10.290.4.5** `template<typename P> bool mln::p_run< P >::has_index ( unsigned short i ) const [inline]`

Test if index `i` belongs to this point set.

**10.290.4.6** `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()`.

**10.290.4.7** `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[]()`.

**10.290.4.8** `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()`.

**10.290.4.9** `template<typename P> std::size_t mln::p_run< P >::memory_size ( ) const [inline]`

Return the size of this site set in memory.

**10.290.4.10** `template<typename P> unsigned mln::p_run<P>::nsites ( ) const [inline]`

Give the number of sites.

References `mln::p_run<P>::is_valid()`.

**10.290.4.11** `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()`.

**10.290.4.12** `template<typename P> const P & mln::p_run<P>::start ( ) const [inline]`

Return the starting point.

## 10.291 `mln::p_set<P>` Class Template Reference

Mathematical set of sites (based on [util::set](#)).

```
#include <p_set.hh>
```

Inherits `site_set_base_<P, p_set<P>>`.

### Public Types

- typedef `p_indexed_bkd_piter<self_> bkd_piter`  
*Backward Site\_Iterator associated type.*
- typedef `P element`  
*Element associated type.*
- typedef `p_indexed_fwd_piter<self_> fwd_piter`  
*Forward Site\_Iterator associated type.*
- typedef `P i_element`  
*Insertion element associated type.*
- typedef `fwd_piter piter`  
*Site\_Iterator associated type.*
- typedef `p_indexed_psite<self_> psite`  
*Psite associated type.*
- typedef `P r_element`  
*Removal element associated type.*

## Public Member Functions

- void `clear` ()  
*Clear this set.*
- bool `has` (const `psite` &p) const  
*Test if psite  $p$  belongs to this point set.*
- bool `has` (const util::index &i) const  
*Test if index  $i$  belongs to this point set.*
- bool `has` (const P &p) const  
*Test if  $p$  belongs to this point set.*
- void `insert` (const P &p)  
*Insert a site  $p$ .*
- bool `is_valid` () const  
*Test this set validity so returns always true.*
- std::size\_t `memory_size` () const  
*Return the size of this site set in memory.*
- unsigned `nsites` () const  
*Give the number of sites.*
- const P & `operator[]` (unsigned i) const  
*Return the  $i$ -th site.*
- `p_set` ()  
*Constructor.*
- void `remove` (const P &p)  
*Remove a site  $p$ .*
- const std::vector< P > & `std_vector` () const  
*Return the corresponding std::vector of sites.*
- const util::set< P > & `util_set` () const  
*Return the corresponding util::set of sites.*

### 10.291.1 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.

## 10.291.2 Member Typedef Documentation

**10.291.2.1** `template<typename P> typedef p_indexed_bkd_piter<self_> mln::p_set< P >::bkd_piter`

Backward [Site\\_Iterator](#) associated type.

**10.291.2.2** `template<typename P> typedef P mln::p_set< P >::element`

Element associated type.

**10.291.2.3** `template<typename P> typedef p_indexed_fwd_piter<self_> mln::p_set< P >::fwd_piter`

Forward [Site\\_Iterator](#) associated type.

**10.291.2.4** `template<typename P> typedef P mln::p_set< P >::i_element`

Insertion element associated type.

**10.291.2.5** `template<typename P> typedef fwd_piter mln::p_set< P >::piter`

[Site\\_Iterator](#) associated type.

**10.291.2.6** `template<typename P> typedef p_indexed_psite<self_> mln::p_set< P >::psite`

Psite associated type.

**10.291.2.7** `template<typename P> typedef P mln::p_set< P >::r_element`

Removal element associated type.

## 10.291.3 Constructor & Destructor Documentation

**10.291.3.1** `template<typename P > mln::p_set< P >::p_set ( ) [inline]`

Constructor.

## 10.291.4 Member Function Documentation

**10.291.4.1** `template<typename P > void mln::p_set< P >::clear ( ) [inline]`

Clear this set.

**10.291.4.2** `template<typename P> bool mln::p_set< P >::has ( const psite & p ) const [inline]`

Test if psite *p* belongs to this point set.

**10.291.4.3** `template<typename P> bool mln::p_set< P >::has ( const P & p ) const [inline]`

Test if *p* belongs to this point set.

**10.291.4.4** `template<typename P> bool mln::p_set< P >::has ( const util::index & i ) const [inline]`

Test if index *i* belongs to this point set.

References `mln::p_set< P >::nsites()`.

**10.291.4.5** `template<typename P> void mln::p_set< P >::insert ( const P & p ) [inline]`

Insert a site *p*.

Referenced by `mln::convert::to_p_set()`.

**10.291.4.6** `template<typename P> bool mln::p_set< P >::is_valid ( ) const [inline]`

Test this set validity so returns always true.

**10.291.4.7** `template<typename P> std::size_t mln::p_set< P >::memory_size ( ) const [inline]`

Return the size of this site set in memory.

**10.291.4.8** `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()`.

**10.291.4.9** `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()`.

**10.291.4.10** `template<typename P> void mln::p_set< P >::remove ( const P & p ) [inline]`

Remove a site *p*.

**10.291.4.11** `template<typename P> const std::vector< P> & mln::p_set< P>::std_vector ( ) const [inline]`

Return the corresponding `std::vector` of sites.

**10.291.4.12** `template<typename P> const util::set< P> & mln::p_set< P>::util_set ( ) const [inline]`

Return the corresponding `util::set` of sites.

## 10.292 `mln::p_set_of< S>` Class Template Reference

`p_set_of` is a set of site sets.

```
#include <p_set_of.hh>
```

Inherits `site_set_base_< S::site, p_set_of< S>>`, and `site_set_impl< S>`.

### Public Types

- typedef `p_double_piter< self_, mln_bkd_eiter(set_), typename S::bkd_piter>` `bkd_piter`  
*Backward [Site\\_Iterator](#) associated type.*
- typedef `S element`  
*Element associated type.*
- typedef `p_double_piter< self_, mln_fwd_eiter(set_), typename S::fwd_piter>` `fwd_piter`  
*Forward [Site\\_Iterator](#) associated type.*
- typedef `S i_element`  
*Insertion element associated type.*
- typedef `fwd_piter piter`  
*[Site\\_Iterator](#) associated type.*
- typedef `p_double_psite< self_, element>` `psite`  
*Psite associated type.*

### Public Member Functions

- void `clear ()`  
*Clear this set.*
- bool `has (const psite &p) const`  
*Test if `p` belongs to this point set.*
- void `insert (const S &s)`  
*Insert a site set `s`.*



- bool [is\\_valid](#) () const  
*Test if this set of runs is valid.*
- std::size\_t [memory\\_size](#) () const  
*Return the size of this site set in memory.*
- unsigned [nelements](#) () const  
*Give the number of elements (site sets) of this composite.*
- const S & [operator\[\]](#) (unsigned i) const  
*Return the *i*-th site set.*
- [p\\_set\\_of](#) ()  
*Constructor without arguments.*

### 10.292.1 Detailed Description

`template<typename S> class mln::p_set_of< S >`

[p\\_set\\_of](#) is a set of site sets. Parameter *S* is the type of the contained site sets.

### 10.292.2 Member Typedef Documentation

**10.292.2.1** `template<typename S > typedef p_double_piter<self_, mln_bkd_eiter(set_), typename S ::bkd_piter> mln::p_set_of< S >::bkd_piter`

Backward [Site\\_Iterator](#) associated type.

**10.292.2.2** `template<typename S > typedef S mln::p_set_of< S >::element`

Element associated type.

**10.292.2.3** `template<typename S > typedef p_double_piter<self_, mln_fwd_eiter(set_), typename S ::fwd_piter> mln::p_set_of< S >::fwd_piter`

Forward [Site\\_Iterator](#) associated type.

**10.292.2.4** `template<typename S > typedef S mln::p_set_of< S >::i_element`

Insertion element associated type.

**10.292.2.5** `template<typename S > typedef fwd_piter mln::p_set_of< S >::piter`

[Site\\_Iterator](#) associated type.

**10.292.2.6** `template<typename S > typedef p_double_psite<self_, element> mln::p_set_of< S >::psite`

Psite associated type.

### 10.292.3 Constructor & Destructor Documentation

**10.292.3.1** `template<typename S > mln::p_set_of< S >::p_set_of ( ) [inline]`

Constructor without arguments.

### 10.292.4 Member Function Documentation

**10.292.4.1** `template<typename S > void mln::p_set_of< S >::clear ( ) [inline]`

Clear this set.

**10.292.4.2** `template<typename S > bool mln::p_set_of< S >::has ( const psite & p ) const [inline]`

Test if *p* belongs to this point set.

**10.292.4.3** `template<typename S > void mln::p_set_of< S >::insert ( const S & s ) [inline]`

Insert a site set *s*.

**10.292.4.4** `template<typename S > bool mln::p_set_of< S >::is_valid ( ) const [inline]`

Test if this set of runs is valid.

**10.292.4.5** `template<typename S > std::size_t mln::p_set_of< S >::memory_size ( ) const [inline]`

Return the size of this site set in memory.

**10.292.4.6** `template<typename S > unsigned mln::p_set_of< S >::nelements ( ) const [inline]`

Give the number of elements (site sets) of this composite.

**10.292.4.7** `template<typename S > const S & mln::p_set_of< S >::operator[] ( unsigned i ) const [inline]`

Return the *i*-th site set.

## 10.293 mln::p\_transformed< S, F > Class Template Reference

[Site](#) set transformed through a function.

```
#include <p_transformed.hh>
```

Inherits [site\\_set\\_base\\_< S::psite, p\\_transformed< S, F > >](#).

### Public Types

- typedef [p\\_transformed\\_piter](#)< typename S::bkd\_piter, S, F > [bkd\\_piter](#)  
*Backward [Site\\_Iterator](#) associated type.*
- typedef S::element [element](#)  
*Element associated type.*
- typedef [p\\_transformed\\_piter](#)< typename S::fwd\_piter, S, F > [fwd\\_piter](#)  
*Forward [Site\\_Iterator](#) associated type.*
- typedef [fwd\\_piter](#) [piter](#)  
*[Site\\_Iterator](#) associated type.*
- typedef S::psite [psite](#)  
*Psite associated type.*

### Public Member Functions

- const F & [function](#) () const  
*Return the transformation function.*
- bool [has](#) (const [psite](#) &p) const  
*Test if *p* belongs to the subset.*
- bool [is\\_valid](#) () const  
*Test if this site set is valid.*
- std::size\_t [memory\\_size](#) () const  
*Return the size of this site set in memory.*
- [p\\_transformed](#) ()  
*Constructor without argument.*
- [p\\_transformed](#) (const S &s, const F &f)  
*Constructor with a site set *s* and a predicate *f*.*
- const S & [primary\\_set](#) () const  
*Return the primary set.*

### 10.293.1 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.

### 10.293.2 Member Typedef Documentation

**10.293.2.1** `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.

**10.293.2.2** `template<typename S, typename F> typedef S ::element mln::p_transformed< S, F  
>::element`

Element associated type.

**10.293.2.3** `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.

**10.293.2.4** `template<typename S, typename F> typedef fwd_piter mln::p_transformed< S, F  
>::piter`

[Site\\_Iterator](#) associated type.

**10.293.2.5** `template<typename S, typename F> typedef S ::psite mln::p_transformed< S, F  
>::psite`

Psite associated type.

### 10.293.3 Constructor & Destructor Documentation

**10.293.3.1** `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*.

**10.293.3.2** `template<typename S, typename F > mln::p_transformed< S, F >::p_transformed (  
) [inline]`

Constructor without argument.

### 10.293.4 Member Function Documentation

**10.293.4.1** `template<typename S , typename F > const F & mln::p_transformed< S, F >::function ( ) const [inline]`

Return the transformation function.

**10.293.4.2** `template<typename S , typename F > bool mln::p_transformed< S, F >::has ( const psite & p ) const [inline]`

Test if p belongs to the subset.

**10.293.4.3** `template<typename S , typename F > bool mln::p_transformed< S, F >::is_valid ( ) const [inline]`

Test if this site set is valid.

**10.293.4.4** `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.

**10.293.4.5** `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().

## 10.294 mln::p\_transformed\_piter< Pi, S, F > Struct Template Reference

[Iterator](#) on p\_transformed<S,F>.

```
#include <p_transformed_piter.hh>
```

Inherits mln::internal::site\_set\_iterator\_base< p\_transformed< S, F >,p\_transformed\_piter< Pi, S, F > >.

### Public Member Functions

- void [change\\_target](#) (const [p\\_transformed](#)< S, F > &s)  
*Change the set site targeted by this iterator.*
- void [next](#) ()  
*Go to the next element.*
- [p\\_transformed\\_piter](#) (const [p\\_transformed](#)< S, F > &s)  
*Constructor from a site set.*

- [p\\_transformed\\_piter\(\)](#)  
*Constructor without argument.*

### 10.294.1 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](#)

### 10.294.2 Constructor & Destructor Documentation

**10.294.2.1** `template<typename Pi, typename S, typename F > mln::p_transformed_piter< Pi, S, F >::p_transformed_piter( ) [inline]`

Constructor without argument.

**10.294.2.2** `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()`.

### 10.294.3 Member Function Documentation

**10.294.3.1** `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()`.

**10.294.3.2** `template<typename E > void mln::Site_Iterator< E >::next( ) [inline, inherited]`

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

#### Precondition

The iterator is valid.

## 10.295 mln::p\_vaccess< V, S > Class Template Reference

[Site](#) set in which sites are grouped by their associated value.

```
#include <p_vaccess.hh>
```

Inherits [site\\_set\\_base\\_< S::site, p\\_vaccess< V, S > >](#), and [site\\_set\\_impl< S >](#).

### Public Types

- typedef [p\\_double\\_piter< self\\_, typename vset::bkd\\_viter, typename S::bkd\\_piter >](#) [bkd\\_piter](#)  
*Backward [Site\\_Iterator](#) associated type.*
- typedef [S::element](#) [element](#)  
*Element associated type.*
- typedef [p\\_double\\_piter< self\\_, typename vset::fwd\\_viter, typename S::fwd\\_piter >](#) [fwd\\_piter](#)  
*Forward [Site\\_Iterator](#) associated type.*
- typedef [std::pair< V, element >](#) [i\\_element](#)  
*Insertion element associated type.*
- typedef [fwd\\_piter](#) [piter](#)  
*[Site\\_Iterator](#) associated type.*
- typedef [S](#) [pset](#)  
*Inner site set associated type.*
- typedef [p\\_double\\_psite< self\\_, S >](#) [psite](#)  
*Psite associated type.*
- typedef [V](#) [value](#)  
*Value associated type.*
- typedef [mln::value::set< V >](#) [vset](#)  
*[Value\\_Set](#) associated type.*

### Public Member Functions

- bool [has](#) (const [psite](#) &p) const  
*Test if *p* belongs to this site set.*
- bool [has](#) (const [V](#) &v, const typename [S::psite](#) &p) const  
*Test if the couple (value *v*, psite *p*) belongs to this site set.*
- void [insert](#) (const [i\\_element](#) &v\_e)  
*Insert a pair *v\_e* (value *v*, element *e*).*
- void [insert](#) (const [V](#) &v, const [element](#) &e)

*Insert  $e$  at value  $v$ .*

- `bool is_valid () const`  
*Test if this site set is valid.*
- `std::size_t memory_size () const`  
*Return the size of this site set in memory.*
- `const S & operator() (const V &v) const`  
*Return the site set at value  $v$ .*
- `p_vaccess ()`  
*Constructor.*
- `const mln::value::set< V > & values () const`  
*Give the set of values.*

### 10.295.1 Detailed Description

`template<typename V, typename S> class mln::p_vaccess< V, S >`

[Site](#) set in which sites are grouped by their associated value.

### 10.295.2 Member Typedef Documentation

**10.295.2.1** `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.

**10.295.2.2** `template<typename V , typename S > typedef S ::element mln::p_vaccess< V, S  
>::element`

Element associated type.

**10.295.2.3** `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.

**10.295.2.4** `template<typename V , typename S > typedef std::pair<V, element> mln::p_vaccess<  
V, S >::i_element`

Insertion element associated type.

**10.295.2.5** `template<typename V , typename S > typedef fwd_piter mln::p_vaccess< V, S >::piter`

[Site\\_Iterator](#) associated type.



**10.295.2.6** `template<typename V , typename S > typedef S mln::p_vaccess< V, S >::pset`

Inner site set associated type.

**10.295.2.7** `template<typename V , typename S > typedef p_double_psite<self_, S>  
mln::p_vaccess< V, S >::psite`

Psite associated type.

**10.295.2.8** `template<typename V , typename S > typedef V mln::p_vaccess< V, S >::value`

[Value](#) associated type.

**10.295.2.9** `template<typename V , typename S > typedef mln::value::set<V> mln::p_vaccess<  
V, S >::vset`

[Value\\_Set](#) associated type.

**10.295.3** **Constructor & Destructor Documentation****10.295.3.1** `template<typename V , typename S > mln::p_vaccess< V, S >::p_vaccess ( )  
[inline]`

Constructor.

**10.295.4** **Member Function Documentation****10.295.4.1** `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.

**10.295.4.2** `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.

**10.295.4.3** `template<typename V , typename S > void mln::p_vaccess< V, S >::insert ( const  
i_element & v_e ) [inline]`

Insert a pair *v\_e* (value *v*, element *e*).

**10.295.4.4** `template<typename V , typename S > void mln::p_vaccess< V, S >::insert ( const V  
& v, const element & e ) [inline]`

Insert *e* at value *v*.

**10.295.4.5** `template<typename V , typename S > bool mln::p_vaccess< V, S >::is_valid ( ) const [inline]`

Test if this site set is valid.

**10.295.4.6** `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.

**10.295.4.7** `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*.

**10.295.4.8** `template<typename V , typename S > const mln::value::set< V > & mln::p_vaccess< V, S >::values ( ) const [inline]`

Give the set of values.

## 10.296 mln::p\_vertices< G, F > Class Template Reference

[Site](#) set based mapping graph vertices to sites.

```
#include <p_vertices.hh>
```

Inherits `site_set_base_< F::result, p_vertices< G, F > >`.

### Public Types

- typedef F [fun\\_t](#)  
*Function associated type.*
- typedef [util::vertex](#)< G > [graph\\_element](#)  
*Type of graph element this site set focuses on.*
- typedef G [graph\\_t](#)  
*Graph associated type.*
- typedef [util::vertex](#)< G > [vertex](#)  
*Type of graph vertex.*
- typedef [super\\_::site](#) [element](#)  
*Associated types.*
- typedef [p\\_vertices\\_psite](#)< G, F > [psite](#)  
*Point\_Site associated type.*

- typedef `p_graph_piter< self_, mln_vertex_fwd_iter(G) > fwd_piter`  
*Forward `Site_Iterator` associated type.*
- typedef `p_graph_piter< self_, mln_vertex_bkd_iter(G) > bkd_piter`  
*Backward `Site_Iterator` associated type.*
- typedef `fwd_piter piter`  
*`Site_Iterator` associated type.*

## Public Member Functions

- `bool has (const psite &p) const`  
*Does this site set has p?*
- `template<typename G2 > bool has (const util::vertex< G2 > &v) const`  
*Does this site set has v?*
- `void invalidate ()`  
*Invalidate this site set.*
- `bool is_valid () const`  
*Test this site set validity.*
- `std::size_t memory_size () const`  
*Does this site set has vertex\_id? FIXME: causes ambiguities while calling has(mln::neighb\_fwd\_niter<>); bool has(unsigned vertex\_id) const;*
- `unsigned nsites () const`  
*Return The number of points (sites) of the set, i.e., the number of vertices.*
- `unsigned nvertices () const`  
*Return The number of vertices in the graph.*
- `p_vertices (const Graph< G > &gr)`  
*Construct a graph psite set from a graph of points.*
- `p_vertices (const Graph< G > &gr, const Function< F > &f)`  
*Construct a graph psite set from a graph of points.*
- `template<typename F2 > p_vertices (const p_vertices< G, F2 > &other)`  
*Copy constructor.*
- `p_vertices ()`  
*Constructor without argument.*
- `template<typename F2 > p_vertices (const Graph< G > &gr, const Function< F2 > &f)`

*Construct a graph psite set from a graph of points.*

- `F::result operator() (const psite &p) const`  
*Return the value associated to an element of this site set.*
- `const G & graph () const`  
*Accessors.*
- `const F & function () const`  
*Return the association function.*

### 10.296.1 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.

### 10.296.2 Member Typedef Documentation

**10.296.2.1** `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.

**10.296.2.2** `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.

**10.296.2.3** `template<typename G, typename F = util::internal::id2element<G,util::vertex<G> >> typedef F mln::p_vertices< G, F >::fun_t`

[Function](#) associated type.

**10.296.2.4** `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.

**10.296.2.5** `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.

**10.296.2.6** `template<typename G, typename F = util::internal::id2element<G,util::vertex<G>>> typedef G mln::p_vertices< G, F >::graph_t`

[Graph](#) associated type.

**10.296.2.7** `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.

**10.296.2.8** `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.

**10.296.2.9** `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.

### 10.296.3 Constructor & Destructor Documentation

**10.296.3.1** `template<typename G , typename F > mln::p_vertices< G, F >::p_vertices ( ) [inline]`

Constructor without argument.

**10.296.3.2** `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()`.

**10.296.3.3** `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()`.

**10.296.3.4** `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()`.

**10.296.3.5** `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()`.

**10.296.4 Member Function Documentation**

**10.296.4.1** `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()`.

**10.296.4.2** `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()`.

**10.296.4.3** `template<typename G , typename F > bool mln::p_vertices< G, F >::has ( const psite & p ) const [inline]`

Does this site set has *p*?

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

**10.296.4.4** `template<typename G , typename F > template<typename G2 > bool  
mln::p_vertices< G, F >::has ( const util::vertex< G2 > & v ) const [inline]`

Does this site set has *v*?

References mln::util::vertex< G >::graph(), mln::util::vertex< G >::is\_valid(), and mln::p\_vertices< G, F >::is\_valid().

**10.296.4.5** `template<typename G , typename F > void mln::p_vertices< G, F >::invalidate ( )  
[inline]`

Invalidate this site set.

**10.296.4.6** `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().

**10.296.4.7** `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;.

**10.296.4.8** `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().

**10.296.4.9** `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().

**10.296.4.10** `template<typename G , typename F > F::result mln::p_vertices< G, F >::operator()  
( const psite & p ) const [inline]`

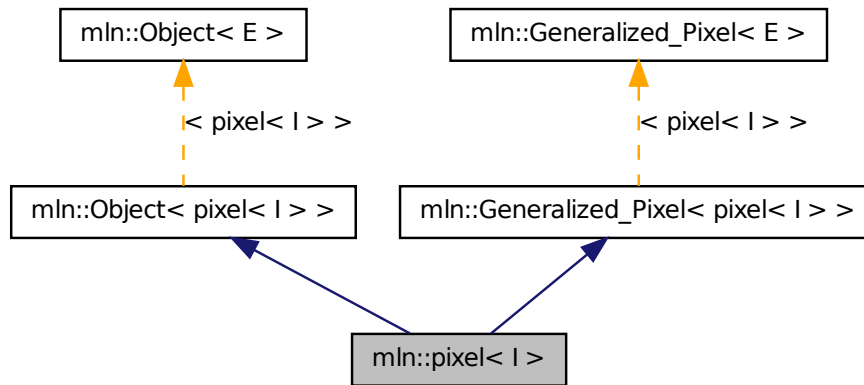
Return the value associated to an element of this site set.

## 10.297 mln::pixel< I > Struct Template Reference

Generic pixel class.

```
#include <pixel.hh>
```

Inheritance diagram for `mln::pixel< I >`:



## Public Member Functions

- void [change\\_to](#) (const typename I::psite &p)  
*Change the pixel to the one at point p.*
- bool [is\\_valid](#) () const  
*Test if this pixel is valid.*
- [pixel](#) (I &image)  
*Constructor.*
- [pixel](#) (I &image, const typename I::psite &p)  
*Constructor.*

### 10.297.1 Detailed Description

```
template<typename I> struct mln::pixel< I >
```

Generic pixel class. The parameter is  $I$  the type of the image it belongs to.

### 10.297.2 Constructor & Destructor Documentation

**10.297.2.1** `template<typename I> mln::pixel< I >::pixel ( I & image ) [inline]`

Constructor.



**10.297.2.2** `template<typename I > mln::pixel< I >::pixel ( I & image, const typename I::psite & p ) [inline]`

Constructor.

References `mln::pixel< I >::change_to()`.

### 10.297.3 Member Function Documentation

**10.297.3.1** `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()`.

**10.297.3.2** `template<typename I > bool mln::pixel< I >::is_valid ( ) const [inline]`

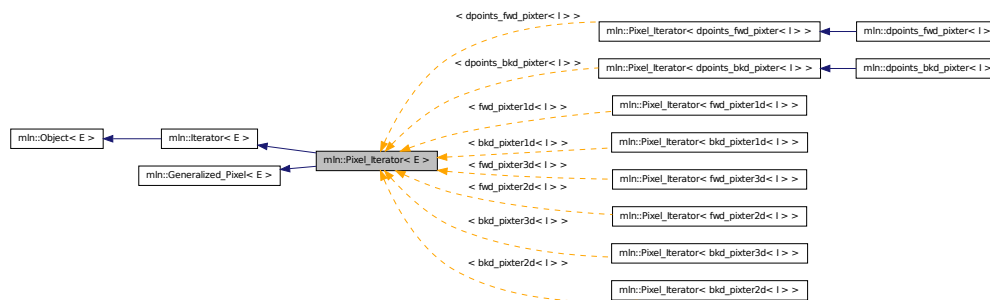
Test if this pixel is valid.

## 10.298 mln::Pixel\_Iterator< E > Struct Template Reference

Base class for the implementation of pixel iterator classes.

```
#include <pixel_iterator.hh>
```

Inheritance diagram for `mln::Pixel_Iterator< E >`:



### Public Member Functions

- void `next ()`

*Go to the next element.*

## 10.298.1 Detailed Description

```
template<typename E> struct mln::Pixel_Iterator< E >
```

Base class for the implementation of pixel iterator classes. An iterator on pixels is an iterator that is bound to a particular image and that browses over a set of image pixels.

### See also

[mln::doc::Pixel\\_Iterator](#) for a complete documentation of this class contents.

## 10.298.2 Member Function Documentation

**10.298.2.1** `template<typename E > void mln::Iterator< E >::next ( ) [inherited]`

Go to the next element.

### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

### Precondition

The iterator is valid.

## 10.299 mln::plain< I > Class Template Reference

Prevents an image from sharing its data.

```
#include <plain.hh>
```

Inherits `image_identity< I, I::domain_t, plain< I > >`.

### Public Types

- typedef `plain< tag::image_< I > >` `skeleton`  
*Skeleton.*

### Public Member Functions

- `operator I () const`  
*Conversion into an image with type I.*
- `plain< I > & operator= (const I &ima)`  
*Assignment operator from an image ima.*
- `plain< I > & operator= (const plain< I > &rhs)`  
*Assignment operator.*

- `plain` (const `plain`< I > &rhs)  
*Copy constructor.*
- `plain` ()  
*Constructor without argument.*
- `plain` (const I &ima)  
*Copy constructor from an image ima.*

### 10.299.1 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.

### 10.299.2 Member Typedef Documentation

**10.299.2.1** `template<typename I> typedef plain< tag::image_<I> > mln::plain< I >::skeleton`

Skeleton.

### 10.299.3 Constructor & Destructor Documentation

**10.299.3.1** `template<typename I> mln::plain< I >::plain ( ) [inline]`

Constructor without argument.

**10.299.3.2** `template<typename I> mln::plain< I >::plain ( const plain< I > & rhs ) [inline]`

Copy constructor.

**10.299.3.3** `template<typename I> mln::plain< I >::plain ( const I & ima ) [inline]`

Copy constructor from an image ima.

### 10.299.4 Member Function Documentation

**10.299.4.1** `template<typename I> mln::plain< I >::operator I ( ) const [inline]`

Conversion into an image with type I.

References `mln::duplicate()`.

**10.299.4.2** `template<typename I> plain< I > & mln::plain< I >::operator= ( const plain< I > & rhs ) [inline]`

Assignment operator.

**10.299.4.3** `template<typename I> plain< I > & mln::plain< I >::operator= ( const I & ima ) [inline]`

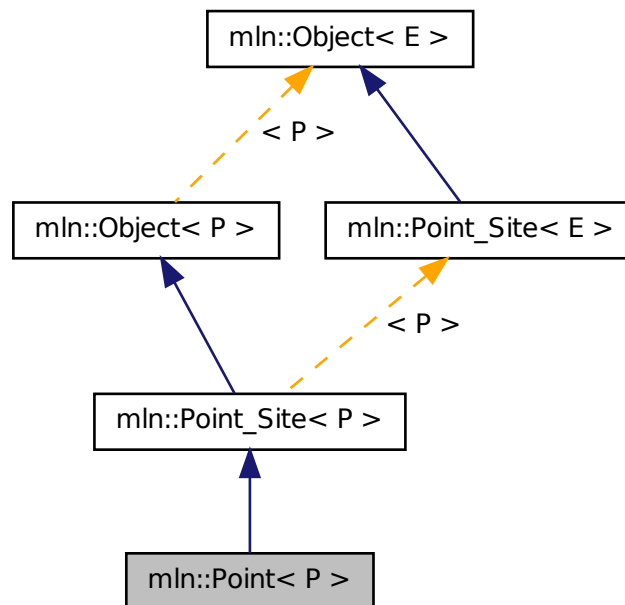
Assignment operator from an image `ima`.

## 10.300 `mln::Point< P >` Struct Template Reference

Base class for implementation of point classes.

```
#include <point.hh>
```

Inheritance diagram for `mln::Point< P >`:



### Public Types

- `typedef P point`

*The associated point type is itself.*

## Public Member Functions

- const P & [to\\_point](#) () const

*It is a [Point](#) so it returns itself.*

## Related Functions

(Note that these are not member functions.)

- `template<typename P , typename D >`  
`P & operator+= ( Point< P > &p, const Dpoint< D > &dp)`  
*Shift a point by a delta-point dp.*
- `template<typename P , typename D >`  
`P & operator-= ( Point< P > &p, const Dpoint< D > &dp)`  
*Shift a point by the negate of a delta-point dp.*
- `template<typename P , typename D >`  
`P & operator/ ( Point< P > &p, const value::Scalar< D > &dp)`  
*Divide a point by a scalar s.*

### 10.300.1 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.

### 10.300.2 Member Typedef Documentation

#### 10.300.2.1 `template<typename P > typedef P mln::Point< P >::point`

The associated point type is itself.

### 10.300.3 Member Function Documentation

#### 10.300.3.1 `template<typename P > const P & mln::Point< P >::to_point ( ) const [inline]`

It is a [Point](#) so it returns itself.

### 10.300.4 Friends And Related Function Documentation

#### 10.300.4.1 `template<typename P , typename D > P & operator+= ( Point< P > & p, const Dpoint< D > & dp ) [related]`

Shift a point by a delta-point dp.

**Parameters**

[in, out] *p* The targeted point.

[in] *dp* A delta-point.

**Returns**

A reference to the point *p* once translated by *dp*.

**Precondition**

The type of *dp* has to be compatible with the type of *p*.

#### 10.300.4.2 `template<typename P, typename D > P & operator-= ( Point< P > & p, const Dpoint< D > & dp ) [related]`

Shift a point by the negate of a delta-point *dp*.

**Parameters**

[in, out] *p* The targeted point.

[in] *dp* A delta-point.

**Returns**

A reference to the point *p* once translated by - *dp*.

**Precondition**

The type of *dp* has to be compatible with the type of *p*.

#### 10.300.4.3 `template<typename P, typename D > P & operator/ ( Point< P > & p, const value::Scalar< D > & dp ) [related]`

Divide a point by a scalar *s*.

**Parameters**

[in, out] *p* The targeted point.

[in] *dp* A scalar.

**Returns**

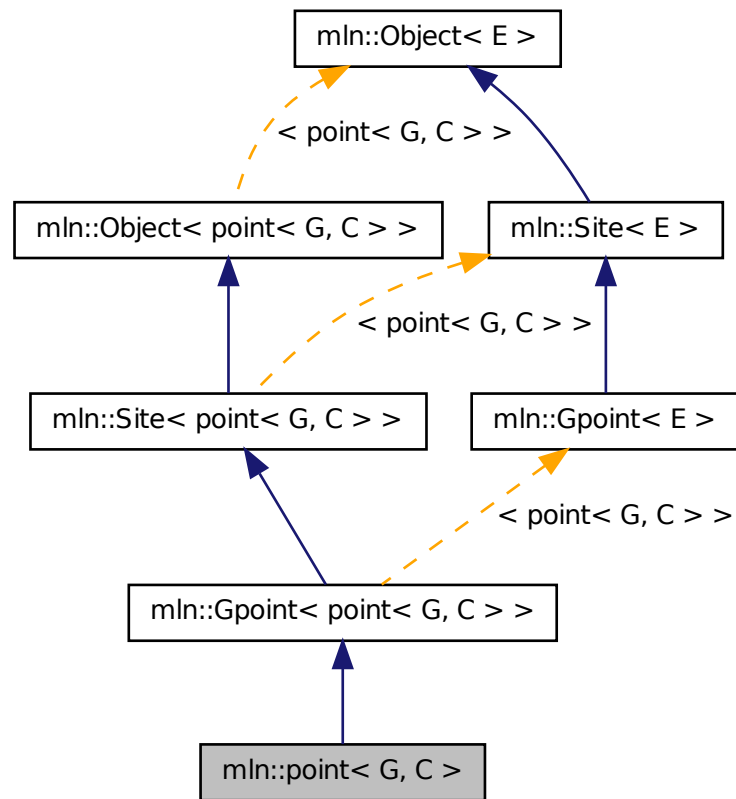
A reference to the point *p* once divided by *s*.

## 10.301 `mln::point< G, C >` Struct Template Reference

Generic point class.

```
#include <point.hh>
```

Inheritance diagram for mln::point< G, C >:



## Public Types

- enum { `dim = G::dim` }
- typedef `C coord`  
*Coordinate associated type.*
- typedef `dpoint< G, C > delta`  
*Delta associated type.*
- typedef `dpoint< G, C > dpsite`  
*DPsite associated type.*
- typedef `G grid`  
*Grid associated type.*
- typedef `mln::algebra::h_vec< G::dim, float > h_vec`

*Algebra hexagonal vector (hvec) associated type.*

- `typedef mln::algebra::vec< G::dim, float > vec`  
*Algebra vector (vec) associated type.*

## Public Member Functions

- `const C & last_coord () const`  
*Read-only access to the last coordinate.*
- `C & last_coord ()`  
*Read-write access to the last coordinate.*
- `point< G, C > & operator+= (const delta &dp)`  
*Shifting by dp.*
- `point< G, C > & operator-= (const delta &dp)`  
*Shifting by the inverse of dp.*
- `C & operator[] (unsigned i)`  
*Read-write access to the i-th coordinate value.*
- `const C & operator[] (unsigned i) const`  
*Read-only access to the i-th coordinate value.*
- `template<typename F >`  
`point (const Function_v2v< F > &f)`  
*Constructor; coordinates are set by function f.*
- `point ()`  
*Constructor without argument.*
- `template<typename C2 >`  
`point (const mln::algebra::vec< dim, C2 > &v)`  
*Constructor from an algebra vector.*
- `void set_all (C c)`  
*Set all coordinates to the value c.*
- `h_vec to_h_vec () const`  
*Transform to point in homogeneous coordinate system.*
- `vec to_vec () const`  
*Explicit conversion towards mln::algebra::vec.*
- `point (C ind)`
- `point (const literal::origin_t &)`  
*Constructors/assignments with literals.*



## Static Public Member Functions

- static const `point< G, C > & minus_infty ()`  
*Point with all coordinates set to the minimum value.*
- static const `point< G, C > & plus_infty ()`  
*Point with all coordinates set to the maximum value.*

## Static Public Attributes

- static const `point< G, C > origin = all_to(0)`  
*Origin point (all coordinates are 0).*

### 10.301.1 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.

### 10.301.2 Member Typedef Documentation

**10.301.2.1** `template<typename G, typename C> typedef C mln::point< G, C >::coord`

Coordinate associated type.

**10.301.2.2** `template<typename G, typename C> typedef dpoint<G,C> mln::point< G, C >::delta`

Delta associated type.

**10.301.2.3** `template<typename G, typename C> typedef dpoint<G,C> mln::point< G, C >::dpsite`

DPsite associated type.

**10.301.2.4** `template<typename G, typename C> typedef G mln::point< G, C >::grid`

Grid associated type.

**10.301.2.5** `template<typename G, typename C> typedef mln::algebra::h_vec<G::dim, float> mln::point< G, C >::h_vec`

Algebra hexagonal vector (hvec) associated type.

**10.301.2.6** `template<typename G, typename C> typedef mln::algebra::vec<G::dim, float>  
mln::point< G, C >::vec`

Algebra vector (vec) associated type.

### 10.301.3 Member Enumeration Documentation

**10.301.3.1** `template<typename G, typename C> anonymous enum`

Enumerator:

*dim* Dimension of the space.

**Invariant**

`dim > 0`

### 10.301.4 Constructor & Destructor Documentation

**10.301.4.1** `template<typename G , typename C > mln::point< G, C >::point ( ) [inline]`

Constructor without argument.

**10.301.4.2** `template<typename G , typename C > template<typename C2 > mln::point< G, C  
>::point ( const mln::algebra::vec< dim, C2 > & v ) [inline]`

Constructor from an algebra vector.

**10.301.4.3** `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.

**10.301.4.4** `template<typename G , typename C> mln::point< G, C >::point ( const  
literal::origin_t & ) [inline]`

Constructors/assignments with literals.

**10.301.4.5** `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.

### 10.301.5 Member Function Documentation

**10.301.5.1** `template<typename G , typename C > const C & mln::point< G, C >::last_coord ( )  
const [inline]`

Read-only access to the last coordinate.

Referenced by `mln::p_run< P >::end()`, `mln::p_run< P >::operator[]()`, and `mln::debug::put_word()`.

**10.301.5.2** `template<typename G , typename C > C & mln::point< G, C >::last_coord ( )`  
`[inline]`

Read-write access to the last coordinate.

**10.301.5.3** `template<typename G , typename C > const point< G, C > & mln::point< G, C >::minus_infty ( )`  
`[inline, static]`

[Point](#) with all coordinates set to the minimum value.

**10.301.5.4** `template<typename G , typename C > point< G, C > & mln::point< G, C >::operator+=( const delta & dp )`  
`[inline]`

Shifting by dp.

**10.301.5.5** `template<typename G , typename C > point< G, C > & mln::point< G, C >::operator-=( const delta & dp )`  
`[inline]`

Shifting by the inverse of dp.

**10.301.5.6** `template<typename G , typename C > C & mln::point< G, C >::operator[] ( unsigned i )`  
`[inline]`

Read-write access to the *i*-th coordinate value.

#### Parameters

`[in]` *i* The coordinate index.

#### Precondition

$i < \text{dim}$

**10.301.5.7** `template<typename G , typename C > const C & mln::point< G, C >::operator[] ( unsigned i ) const`  
`[inline]`

Read-only access to the *i*-th coordinate value.

#### Parameters

`[in]` *i* The coordinate index.

#### Precondition

$i < \text{dim}$

**10.301.5.8** `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.

**10.301.5.9** `template<typename G , typename C> void mln::point< G, C >::set_all ( C c )`  
`[inline]`

Set all coordinates to the value `c`.

**10.301.5.10** `template<typename G , typename C > point< G, C >::h_vec mln::point< G, C >::to_h_vec ( ) const` `[inline]`

Transform to point in homogeneous coordinate system.

**10.301.5.11** `template<typename G , typename C > point< G, C >::vec mln::point< G, C >::to_vec ( ) const` `[inline]`

Explicit conversion towards `mln::algebra::vec`.

Referenced by `mln::io::dicom::load()`.

## 10.301.6 Member Data Documentation

**10.301.6.1** `template<typename G, typename C> const point< G, C > mln::point< G, C >::origin`  
`= all_to(0)` `[static]`

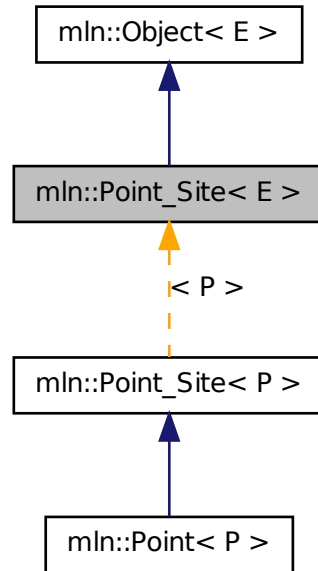
Origin point (all coordinates are 0).

## 10.302 mln::Point\_Site< E > Struct Template Reference

Base class for implementation classes of the notion of "point site".

```
#include <point_site.hh>
```

Inheritance diagram for mln::Point\_Site< E >:



## Related Functions

(Note that these are not member functions.)

- `template<typename L , typename R >`  
`L::dpoint operator-` (const `Point_Site< L >` &lhs, const `Point_Site< R >` &rhs)  
*Difference between a couple of point site lhs and rhs.*
- `template<typename P >`  
`std::ostream & operator<<` (std::ostream &ostr, const `Point_Site< P >` &p)  
*Print a point site p into the output stream ostr.*
- `template<typename L , typename R >`  
`bool operator==` (const `Point_Site< L >` &lhs, const `Point_Site< R >` &rhs)  
*Equality comparison between a couple of point site lhs and rhs.*
- `template<typename P , typename D >`  
`P::point operator+` (const `Point_Site< P >` &p, const `Delta_Point_Site< D >` &dp)  
*Add a delta-point rhs to a point site lhs.*
- `template<typename P , typename D >`  
`P::point operator-` (const `Point_Site< P >` &p, const `Delta_Point_Site< D >` &dp)  
`}`

### 10.302.1 Detailed Description

**template<typename E> struct mln::Point\_Site< E >**

Base class for implementation classes of the notion of "point site". A point site ("psite" for short) is an object that allows an efficient access to data associated with a point. A point site is either a point or designates a point.

When a point site is not really a point, it is automatically convertible to the point it designates.

Let us take the example of a 2D image encoded as an array of runs of values. With a point, a pair (row index, column index), retrieving the corresponding pixel value would mean to browse the array of runs to find the value location. That would not be efficient. Conversely, a point site dedicated to this image structure allows for value access in constant time; precisely the proper point site is a pair (index of run, index within the run).

### 10.302.2 Friends And Related Function Documentation

**10.302.2.1 template<typename P , typename D > P::point operator+ ( const Point\_Site< P > & p, const Delta\_Point\_Site< D > & dp ) [related]**

Add a delta-point `rhs` to a point site `lhs`.

#### Parameters

- [in] *p* A point site.
- [in] *dp* A delta-point.

The type of `dp` has to be compatible with the type of `p`.

#### Returns

A point (temporary object).

#### See also

[mln::Delta\\_Point\\_Site](#)

**10.302.2.2 template<typename P , typename D > P::point operator- ( const Point\_Site< P > & p, const Delta\_Point\_Site< D > & dp ) [related]**

}

Subtract a delta-point `dp` to a point site `p`.

#### Parameters

- [in] *p* A point site.
- [in] *dp* A delta-point.

The type of `dp` has to be compatible with the type of `p`.

#### Returns

A point (temporary object).

**See also**

[mln::Dpoint](#)  
[mln::Delta\\_Point\\_Site](#)

**10.302.2.3** `template<typename L , typename R > L::dpoint operator- ( const Point_Site< L > & lhs, const Point_Site< R > & rhs )` [**related**]

Difference between a couple of point site `lhs` and `rhs`.

**Parameters**

[in] *lhs* A first point site.  
[in] *rhs* A second point site.

**Warning**

There is no type promotion in Milena so the client has to make sure that both points are defined with the same type of coordinates.

**Precondition**

Both `lhs` and `rhs` have to be defined on the same topology and with the same type of coordinates; otherwise this test does not compile.

**Postcondition**

The result, `dp`, is such as `lhs == rhs + dp`.

**Returns**

A delta point (temporary object).

**See also**

[mln::Delta\\_Point\\_Site](#)

**10.302.2.4** `template<typename P > std::ostream & operator<< ( std::ostream & ostr, const Point_Site< P > & p )` [**related**]

Print a point site `p` into the output stream `ostr`.

**Parameters**

[in, out] *ostr* An output stream.  
[in] *p* A point site.

**Returns**

The modified output stream `ostr`.

**10.302.2.5** `template<typename L , typename R > bool operator==( const Point_Site< L > & lhs, const Point_Site< R > & rhs )` [**related**]

Equality comparison between a couple of point site `lhs` and `rhs`.

#### Parameters

[in] *lhs* A first point site.

[in] *rhs* A second point site.

#### Precondition

Both `lhs` and `rhs` have to be defined on the same topology; otherwise this test does not compile.

#### Returns

True if both point sites have the same coordinates, otherwise false.

## 10.303 `mln::Point_Site< void >` Struct Template Reference

[Point](#) site category flag type.

```
#include <point_site.hh>
```

### 10.303.1 Detailed Description

`template<> struct mln::Point_Site< void >`

[Point](#) site category flag type.

## 10.304 `mln::Proxy< E >` Struct Template Reference

Base class for implementation classes of the notion of "proxy".

```
#include <proxy.hh>
```

Inherits [mln::Object< E >](#).

Inherited by [mln::Accumulator< E >](#), [mln::internal::graph\\_iter\\_base< G, Elt, E >](#), [mln::internal::nbh\\_iterator\\_base< G, C, Elt, E >](#), and [mln::Site\\_Proxy< E >](#).

### 10.304.1 Detailed Description

`template<typename E> struct mln::Proxy< E >`

Base class for implementation classes of the notion of "proxy".

## 10.305 `mln::Proxy< void >` Struct Template Reference

[Proxy](#) category flag type.

```
#include <proxy.hh>
```



### 10.305.1 Detailed Description

```
template<> struct mln::Proxy< void >
```

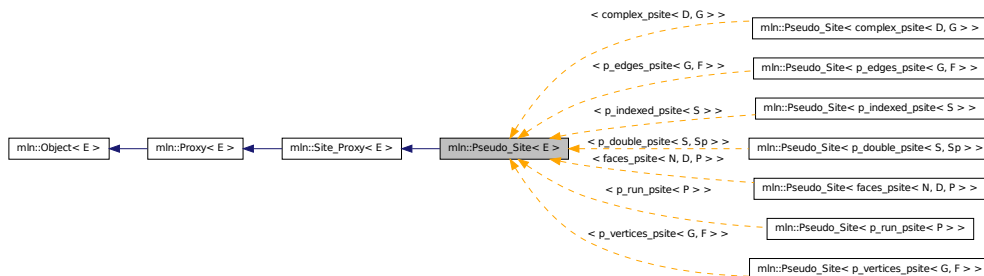
[Proxy](#) category flag type.

## 10.306 mln::Pseudo\_Site< E > Struct Template Reference

Base class for implementation classes of the notion of "pseudo site".

```
#include <pseudo_site.hh>
```

Inheritance diagram for mln::Pseudo\_Site< E >:



### 10.306.1 Detailed Description

```
template<typename E> struct mln::Pseudo_Site< E >
```

Base class for implementation classes of the notion of "pseudo site". FIXME: Explain...

## 10.307 mln::Pseudo\_Site< void > Struct Template Reference

[Pseudo\\_Site](#) category flag type.

```
#include <pseudo_site.hh>
```

### 10.307.1 Detailed Description

```
template<> struct mln::Pseudo_Site< void >
```

[Pseudo\\_Site](#) category flag type.

## 10.308 mln::pw::image< F, S > Class Template Reference

A generic point-wise image implementation.

```
#include <image.hh>
```

Inherits `image_base< F, S, image< F, S >>`.

## Public Types

- typedef `image< tag::function_< F >, tag::domain_< S >>` [skeleton](#)

*Skeleton.*

## Public Member Functions

- `image()`

*Constructor without argument.*

- `image(const Function_v2v< F > &f, const Site_Set< S > &ps)`

*Constructor.*

### 10.308.1 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.

### 10.308.2 Member Typedef Documentation

**10.308.2.1** `template<typename F, typename S> typedef image< tag::function_<F>, tag::domain_<S>> mln::pw::image< F, S >::skeleton`

*Skeleton.*

### 10.308.3 Constructor & Destructor Documentation

**10.308.3.1** `template<typename F, typename S> mln::pw::image< F, S >::image ( )`  
**[inline]**

*Constructor without argument.*

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

*Constructor.*

## 10.309 mln::registration::closest\_point\_basic< P > Class Template Reference

Closest point functor based on map distance.

```
#include <icp.hh>
```

### 10.309.1 Detailed Description

```
template<typename P> class mln::registration::closest_point_basic< P >
```

Closest point functor based on map distance.

## 10.310 mln::registration::closest\_point\_with\_map< P > Class Template Reference

Closest point functor based on map distance.

```
#include <icp.hh>
```

### 10.310.1 Detailed Description

```
template<typename P> class mln::registration::closest_point_with_map< P >
```

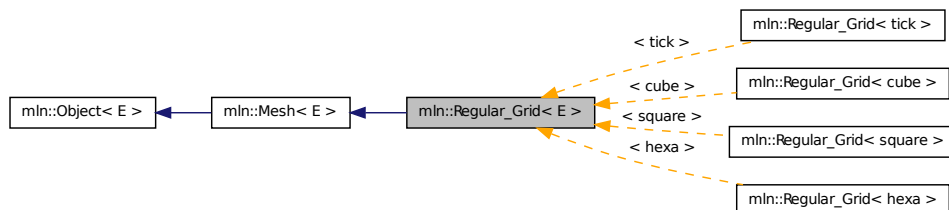
Closest point functor based on map distance.

## 10.311 mln::Regular\_Grid< E > Struct Template Reference

Base class for implementation classes of regular grids.

```
#include <regular_grid.hh>
```

Inheritance diagram for mln::Regular\_Grid< E >:



### 10.311.1 Detailed Description

```
template<typename E> struct mln::Regular_Grid< E >
```

Base class for implementation classes of regular grids.

## 10.312 mln::safe\_image< I > Class Template Reference

Makes an image accessible at undefined location.

```
#include <safe.hh>
```

Inherits image\_identity< I, I::domain\_t, safe\_image< I > >.

### Public Types

- typedef [safe\\_image< tag::image\\_< I > > skeleton](#)

*Skeleton.*

### Public Member Functions

- [operator safe\\_image< const I > \(\) const](#)

*Const promotion via conversion.*

### 10.312.1 Detailed Description

```
template<typename I> class mln::safe_image< I >
```

Makes an image accessible at undefined location.

### 10.312.2 Member Typedef Documentation

**10.312.2.1** `template<typename I> typedef safe_image< tag::image_<I> > mln::safe_image< I >::skeleton`

Skeleton.

### 10.312.3 Member Function Documentation

**10.312.3.1** `template<typename I > mln::safe_image< I >::operator safe_image< const I > ( ) const [inline]`

Const promotion via conversion.

## 10.313 mln::select::p\_of< P > Struct Template Reference

Structure [p\\_of](#).

```
#include <pix.hh>
```

### 10.313.1 Detailed Description

```
template<typename P> struct mln::select::p_of< P >
```

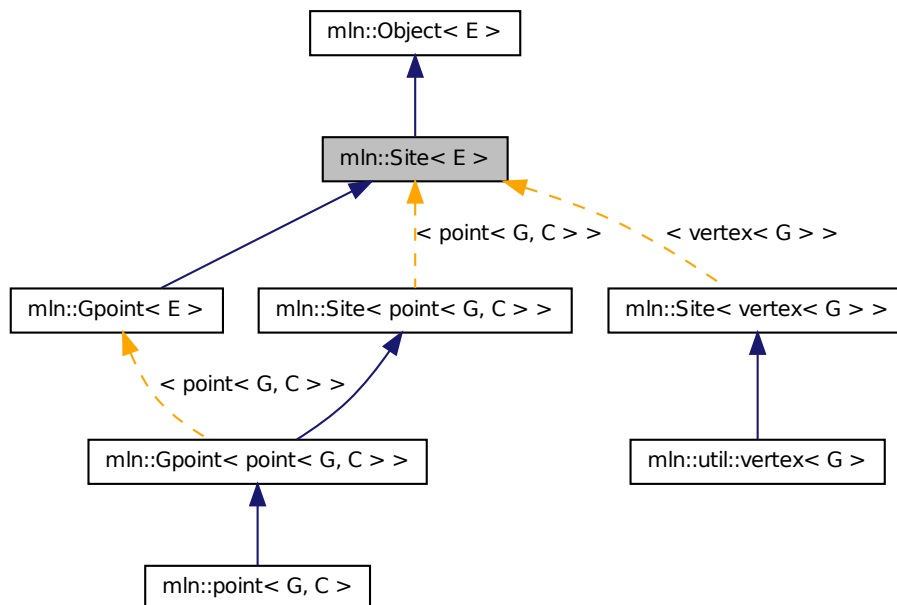
Structure [p\\_of](#).

## 10.314 mln::Site< E > Struct Template Reference

Base class for classes that are explicitly sites.

```
#include <site.hh>
```

Inheritance diagram for mln::Site< E >:



### 10.314.1 Detailed Description

```
template<typename E> struct mln::Site< E >
```

Base class for classes that are explicitly sites.

## 10.315 mln::Site< void > Struct Template Reference

[Site](#) category flag type.

```
#include <site.hh>
```

### 10.315.1 Detailed Description

```
template<> struct mln::Site< void >
```

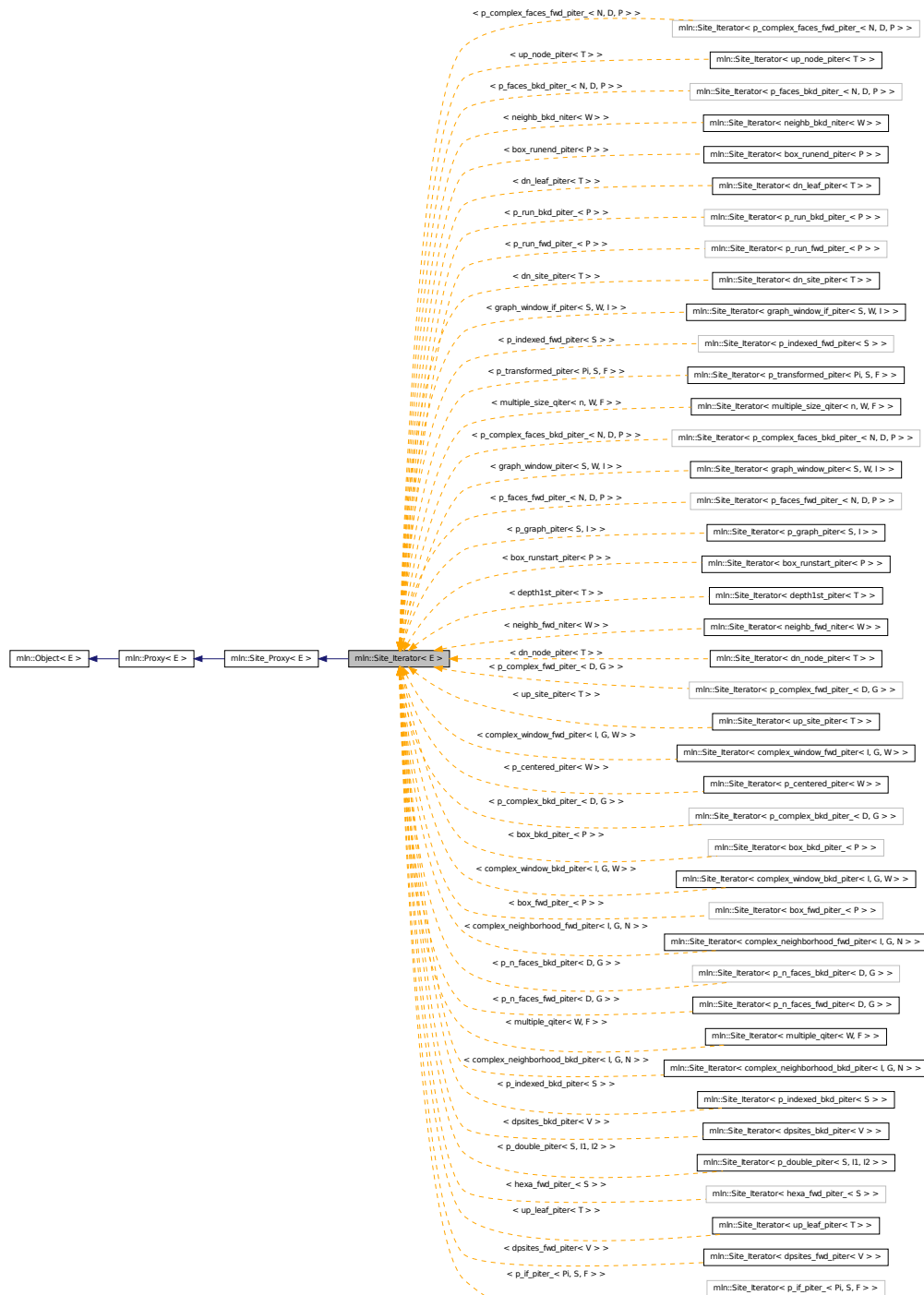
[Site](#) category flag type.

## 10.316 mln::Site\_Iterator< E > Struct Template Reference

Base class for implementation of classes of iterator on points.

```
#include <site_iterator.hh>
```

Inheritance diagram for mln::Site\_Iterator< E >:



## Public Member Functions

- void [next](#) ()  
*Go to the next element.*

### 10.316.1 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.

### 10.316.2 Member Function Documentation

**10.316.2.1** `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.

## 10.317 mln::Site\_Proxy< E > Struct Template Reference

Base class for implementation classes of the notion of "site proxy".

```
#include <site_proxy.hh>
```

Inherits [mln::Proxy< E >](#).

Inherited by [mln::Pseudo\\_Site< E >](#), and [mln::Site\\_Iterator< E >](#).

### 10.317.1 Detailed Description

`template<typename E> struct mln::Site_Proxy< E >`

Base class for implementation classes of the notion of "site proxy". FIXME: Explain...

## 10.318 mln::Site\_Proxy< void > Struct Template Reference

[Site\\_Proxy](#) category flag type.



```
#include <site_proxy.hh>
```

### 10.318.1 Detailed Description

```
template<> struct mln::Site_Proxy< void >
```

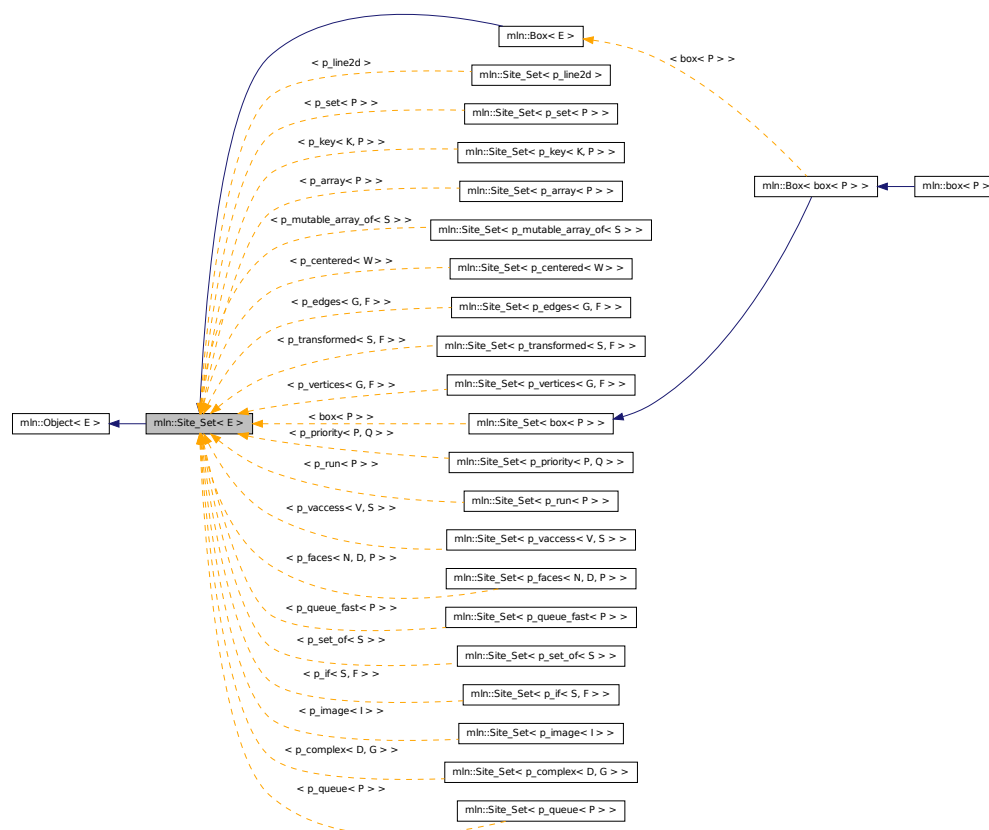
[Site\\_Proxy](#) category flag type.

## 10.319 mln::Site\_Set< E > Struct Template Reference

Base class for implementation classes of site sets.

```
#include <site_set.hh>
```

Inheritance diagram for mln::Site\_Set< E >:



### Related Functions

(Note that these are not member functions.)

- `template<typename SI , typename Sr >`  
`p_set< typename SI::site > diff (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`  
*Set theoretic difference of lhs and rhs.*
- `template<typename SI , typename Sr >`  
`p_set< typename SI::site > inter (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`  
*Intersection between a couple of point sets.*
- `template<typename SI , typename Sr >`  
`bool operator< (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`  
*Strict inclusion test between site sets lhs and rhs.*
- `template<typename S >`  
`std::ostream & operator<< (std::ostream &ostr, const Site_Set< S > &set)`  
*Print a site set set into the output stream ostr.*
- `template<typename SI , typename Sr >`  
`bool operator<= (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`  
*Inclusion test between site sets lhs and rhs.*
- `template<typename SI , typename Sr >`  
`bool operator== (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`  
*Equality test between site sets lhs and rhs.*
- `template<typename SI , typename Sr >`  
`p_set< typename SI::site > sym_diff (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`  
*Set theoretic symmetrical difference of lhs and rhs.*
- `template<typename SI , typename Sr >`  
`p_set< typename SI::site > uni (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`  
*Union of a couple of point sets.*
- `template<typename S >`  
`p_set< typename S::site > unique (const Site_Set< S > &s)`  
*Give the unique set of s.*

### 10.319.1 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.

## 10.319.2 Friends And Related Function Documentation

**10.319.2.1** `template<typename SI , typename Sr > p_set< typename SI::site > diff ( const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs ) [related]`

Set theoretic difference of lhs and rhs.

**10.319.2.2** `template<typename SI , typename Sr > p_set< typename SI::site > inter ( const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs ) [related]`

Intersection between a couple of point sets.

**10.319.2.3** `template<typename SI , typename Sr > bool operator< ( const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs ) [related]`

Strict inclusion test between site sets lhs and rhs.

### Parameters

[in] *lhs* A site set (strictly included?).

[in] *rhs* Another site set (includer?).

**10.319.2.4** `template<typename S > std::ostream & operator<< ( std::ostream & ostr, const Site_Set< S > & set ) [related]`

Print a site set *set* into the output stream *ostr*.

### Parameters

[in, out] *ostr* An output stream.

[in] *set* A site set.

### Returns

The modified output stream *ostr*.

**10.319.2.5** `template<typename SI , typename Sr > bool operator<= ( const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs ) [related]`

Inclusion test between site sets lhs and rhs.

### Parameters

[in] *lhs* A site set (included?).

[in] *rhs* Another site set (includer?).

**10.319.2.6** `template<typename SI, typename Sr > bool operator==( const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs ) [related]`

Equality test between site sets `lhs` and `rhs`.

#### Parameters

[in] `lhs` A site set.

[in] `rhs` Another site set.

**10.319.2.7** `template<typename SI, typename Sr > p_set< typename SI::site > sym_diff ( const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs ) [related]`

Set theoretic symmetrical difference of `lhs` and `rhs`.

**10.319.2.8** `template<typename SI, typename Sr > p_set< typename SI::site > uni ( const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs ) [related]`

Union of a couple of point sets.

**10.319.2.9** `template<typename S > p_set< typename S::site > unique ( const Site_Set< S > & s ) [related]`

Give the unique set of `s`.

## 10.320 mln::Site\_Set< void > Struct Template Reference

[Site\\_Set](#) category flag type.

```
#include <site_set.hh>
```

### 10.320.1 Detailed Description

```
template<> struct mln::Site_Set< void >
```

[Site\\_Set](#) category flag type.

## 10.321 mln::slice\_image< I > Struct Template Reference

2D image extracted from a slice of a 3D image.

```
#include <slice_image.hh>
```

Inherits `image_domain_morpher< I, box2d, slice_image< I > >`.

## Public Types

- typedef `slice_image< tag::image_< I > > skeleton`  
*Skeleton.*

## Public Member Functions

- const `box2d & domain () const`  
*Give the definition domain.*
- `operator slice_image< const I > () const`  
*Const promotion via conversion.*
- `I::rvalue operator() (const point2d &p) const`  
*Read-only access to the image value located at point p.*
- `internal::morpher_lvalue_< I >::ret operator() (const point2d &p)`  
*Read-write access to the image value located at point p.*
- `def::coord sli () const`  
*Give the slice number.*
- `slice_image ()`  
*Constructor without argument.*
- `slice_image (I &ima, def::coord sli)`  
*Constructor from an image ima and a predicate f.*

### 10.321.1 Detailed Description

`template<typename I> struct mln::slice_image< I >`

2D image extracted from a slice of a 3D image.

### 10.321.2 Member Typedef Documentation

**10.321.2.1** `template<typename I> typedef slice_image< tag::image_<I> > mln::slice_image< I >::skeleton`

Skeleton.

### 10.321.3 Constructor & Destructor Documentation

**10.321.3.1** `template<typename I> mln::slice_image< I >::slice_image ( ) [inline]`

Constructor without argument.

**10.321.3.2** `template<typename I> mln::slice_image< I >::slice_image ( I & ima, def::coord sli ) [inline]`

Constructor from an image *ima* and a predicate *f*.

### 10.321.4 Member Function Documentation

**10.321.4.1** `template<typename I> const box2d & mln::slice_image< I >::domain ( ) const [inline]`

Give the definition domain.

**10.321.4.2** `template<typename I> mln::slice_image< I >::operator slice_image< const I > ( ) const [inline]`

Const promotion via conversion.

**10.321.4.3** `template<typename I> internal::morpher_lvalue_< I >::ret mln::slice_image< I >::operator() ( const point2d & p ) [inline]`

Read-write access to the image value located at point *p*.

**10.321.4.4** `template<typename I> I::rvalue mln::slice_image< I >::operator() ( const point2d & p ) const [inline]`

Read-only access to the image value located at point *p*.

**10.321.4.5** `template<typename I> def::coord mln::slice_image< I >::sli ( ) const [inline]`

Give the slice number.

## 10.322 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 ()`  
*Constructor without argument.*
- `sub_image (const I &ima, const S &pset)`  
*Constructor.*

### 10.322.1 Detailed Description

`template<typename I, typename S> struct mln::sub_image< I, S >`

[Image](#) having its domain restricted by a site set.

### 10.322.2 Member Typedef Documentation

**10.322.2.1** `template<typename I, typename S> typedef sub_image< tag::image_<I>, tag::domain_<S> > mln::sub_image< I, S >::skeleton`

Skeleton.

### 10.322.3 Constructor & Destructor Documentation

**10.322.3.1** `template<typename I, typename S > mln::sub_image< I, S >::sub_image ( )`  
`[inline]`

Constructor without argument.

**10.322.3.2** `template<typename I, typename S > mln::sub_image< I, S >::sub_image ( const I & ima, const S & pset )` `[inline]`

Constructor.

### 10.322.4 Member Function Documentation

**10.322.4.1** `template<typename I, typename S > const S & mln::sub_image< I, S >::domain ( )`  
`const` `[inline]`

Give the definition domain.

**10.322.4.2** `template<typename I, typename S > mln::sub_image< I, S >::operator sub_image< const I, S > ( ) const [inline]`

Const promotion via conversion.

## 10.323 mln::sub\_image\_if< I, S > Struct Template Reference

[Image](#) having its domain restricted by a site set and a function.

```
#include <sub_image_if.hh>
```

Inherits `image_domain_morpher< I, p_if< S, fun::p2b::has< I > >, sub_image_if< I, S > >`.

### Public Types

- typedef `sub_image_if< tag::image_< I >, tag::domain_< S > >` [skeleton](#)

*Skeleton.*

### Public Member Functions

- const `p_if< S, fun::p2b::has< I > > & domain () const`

*Give the definition domain.*

- `sub_image_if ()`

*Constructor without argument.*

- `sub_image_if (I &ima, const S &s)`

*Constructor.*

### 10.323.1 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.

### 10.323.2 Member Typedef Documentation

**10.323.2.1** `template<typename I, typename S> typedef sub_image_if< tag::image_<I>, tag::domain_<S> > mln::sub_image_if< I, S >::skeleton`

Skeleton.



### 10.323.3 Constructor & Destructor Documentation

**10.323.3.1** `template<typename I, typename S > mln::sub_image_if< I, S >::sub_image_if ( )`  
`[inline]`

Constructor without argument.

**10.323.3.2** `template<typename I, typename S > mln::sub_image_if< I, S >::sub_image_if ( I & ima, const S & s )`  
`[inline]`

Constructor.

### 10.323.4 Member Function Documentation

**10.323.4.1** `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.

## 10.324 mln::thru\_image< I, F > Class Template Reference

Morph image values through a function.

```
#include <thru_image.hh>
```

### Public Member Functions

- [operator thru\\_image< const I, F > \( \) const](#)  
*Const promotion via conversion.*

### 10.324.1 Detailed Description

```
template<typename I, typename F> class mln::thru_image< I, F >
```

Morph image values through a function.

### 10.324.2 Member Function Documentation

**10.324.2.1** `template<typename I, typename F > mln::thru_image< I, F >::operator thru_image< const I, F > ( ) const`  
`[inline]`

Const promotion via conversion.

## 10.325 mln::thrubin\_image< I1, I2, F > Class Template Reference

Morphes values from two images through a binary function.

```
#include <thrubin_image.hh>
```

Inherits `image_value_morpher< I1, F::result, thrubin_image< I1, I2, F > >`.

## Public Types

- typedef `I1::psite` [psite](#)  
*Point\_Site* associated type.
- typedef `value` [rvalue](#)  
*Return type of read-only access.*
- typedef `thrubin_image< tag::image_< I1 >, tag::image_< I2 >, F >` [skeleton](#)  
*Skeleton.*
- typedef `F::result` [value](#)  
*Value* associated type.

## Public Member Functions

- `operator thrubin_image< const I1, const I2, F > () const`  
*Const promotion via conversion.*

### 10.325.1 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.

### 10.325.2 Member Typedef Documentation

**10.325.2.1** `template<typename I1, typename I2, typename F> typedef I1 ::psite  
mln::thrubin_image< I1, I2, F >::psite`

[Point\\_Site](#) associated type.

**10.325.2.2** `template<typename I1, typename I2, typename F> typedef value  
mln::thrubin_image< I1, I2, F >::rvalue`

Return type of read-only access.

**10.325.2.3** `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.

## 10.326 mln::topo::adj\_higher\_dim\_connected\_n\_face\_bkd\_iter< D > Class Template Reference 889

10.325.2.4 `template<typename I1, typename I2, typename F> typedef F ::result  
mln::thrubin_image< I1, I2, F >::value`

Value associated type.

### 10.325.3 Member Function Documentation

10.325.3.1 `template<typename I1 , typename I2 , typename F > mln::thrubin_image< I1, I2, F  
>::operator thrubin_image< const I1, const I2, F >( ) const [inline]`

Const promotion via conversion.

## 10.326 mln::topo::adj\_higher\_dim\_connected\_n\_face\_bkd\_iter< D > Class Template Reference

Backward iterator on all the n-faces sharing an adjacent (n+1)-face with a (reference) n-face of an mln::complex<D>.

```
#include <adj_higher_dim_connected_n_face_iter.hh>
```

Inherits backward\_complex\_relative\_iterator\_base< topo::face< D >, algebraic\_face< D >, adj\_higher\_dim\_connected\_n\_face\_bkd\_iter< D > >, and mln::topo::internal::adj\_higher\_dim\_connected\_n\_face\_iterator< D >.

### Public Member Functions

- void [next](#) ()

*Go to the next element.*

- [adj\\_higher\\_dim\\_connected\\_n\\_face\\_bkd\\_iter](#) ()

*Construction.*

### 10.326.1 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.

## 10.326.2 Constructor & Destructor Documentation

**10.326.2.1** `template<unsigned D> mln::topo::adj_higher_dim_connected_n_face_bkd_iter< D >::adj_higher_dim_connected_n_face_bkd_iter ( ) [inline]`

Construction.

## 10.326.3 Member Function Documentation

**10.326.3.1** `void mln::Iterator< adj_higher_dim_connected_n_face_bkd_iter< D > >::next ( ) [inherited]`

Go to the next element.

### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

### Precondition

The iterator is valid.

## 10.327 mln::topo::adj\_higher\_dim\_connected\_n\_face\_fwd\_iter< D > > Class Template Reference

Forward iterator on all the n-faces sharing an adjacent (n+1)-face with a (reference) n-face of an mln::complex<D>.

```
#include <adj_higher_dim_connected_n_face_iter.hh>
```

Inherits forward\_complex\_relative\_iterator\_base< topo::face< D >, algebraic\_face< D >, adj\_higher\_dim\_connected\_n\_face\_fwd\_iter< D > >, and mln::topo::internal::adj\_higher\_dim\_connected\_n\_face\_iterator< D >.

## Public Member Functions

- void [next](#) ()  
*Go to the next element.*
- [adj\\_higher\\_dim\\_connected\\_n\\_face\\_fwd\\_iter](#) ()  
*Construction.*

### 10.327.1 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.

**10.327.2 Constructor & Destructor Documentation**

**10.327.2.1** `template<unsigned D> mln::topo::adj_higher_dim_connected_n_face_fwd_iter< D >::adj_higher_dim_connected_n_face_fwd_iter ( ) [inline]`

Construction.

**10.327.3 Member Function Documentation**

**10.327.3.1** `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.

**10.328 mln::topo::adj\_higher\_face\_bkd\_iter< D > Class Template Reference**

Backward iterator on all the adjacent (n+1)-faces of the n-face of an mln::complex<D>.

```
#include <adj_higher_face_iter.hh>
```

Inherits backward\_complex\_relative\_iterator\_base< topo::face< D >, algebraic\_face< D >, adj\_higher\_face\_bkd\_iter< D > >.

**Public Member Functions**

- void [next](#) ()  
*Go to the next element.*

- [adj\\_higher\\_face\\_bkd\\_iter](#) ()  
*Construction.*

**10.328.1 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.

## 10.328.2 Constructor & Destructor Documentation

**10.328.2.1** `template<unsigned D> mln::topo::adj_higher_face_bkd_iter< D >::adj_higher_face_bkd_iter ( ) [inline]`

Construction.

## 10.328.3 Member Function Documentation

**10.328.3.1** `void mln::Iterator< adj_higher_face_bkd_iter< D > >::next ( ) [inherited]`

Go to the next element.

### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

### Precondition

The iterator is valid.

## 10.329 mln::topo::adj\_higher\_face\_fwd\_iter< D > Class Template Reference

Forward iterator on all the adjacent (n+1)-faces of the n-face of an mln::complex<D>.

```
#include <adj_higher_face_iter.hh>
```

Inherits forward\_complex\_relative\_iterator\_base< topo::face< D >, algebraic\_face< D >, adj\_higher\_face\_fwd\_iter< D > >.

### Public Member Functions

- void [next](#) ()  
*Go to the next element.*
- [adj\\_higher\\_face\\_fwd\\_iter](#) ()  
*Construction.*

## 10.329.1 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.

### 10.329.2 Constructor & Destructor Documentation

**10.329.2.1** `template<unsigned D> mln::topo::adj_higher_face_fwd_iter< D >::adj_higher_face_fwd_iter ( ) [inline]`

Construction.

### 10.329.3 Member Function Documentation

**10.329.3.1** `void mln::Iterator< adj_higher_face_fwd_iter< D > >::next ( ) [inherited]`

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

#### Precondition

The iterator is valid.

## 10.330 mln::topo::adj\_lower\_dim\_connected\_n\_face\_bkd\_iter< D > Class Template Reference

Backward iterator on all the n-faces sharing an adjacent (n-1)-face with a (reference) n-face of an mln::complex<D>.

```
#include <adj_lower_dim_connected_n_face_iter.hh>
```

Inherits backward\_complex\_relative\_iterator\_base< topo::face< D >, algebraic\_face< D >, adj\_lower\_dim\_connected\_n\_face\_bkd\_iter< D > >, and mln::topo::internal::adj\_lower\_dim\_connected\_n\_face\_iterator< D >.

### Public Member Functions

- void [next](#) ()

*Go to the next element.*

- [adj\\_lower\\_dim\\_connected\\_n\\_face\\_bkd\\_iter](#) ()

*Construction.*

### 10.330.1 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.

### 10.330.2 Constructor & Destructor Documentation

**10.330.2.1** `template<unsigned D> mln::topo::adj_lower_dim_connected_n_face_bkd_iter< D >::adj_lower_dim_connected_n_face_bkd_iter ( ) [inline]`

Construction.

### 10.330.3 Member Function Documentation

**10.330.3.1** `void mln::Iterator< adj_lower_dim_connected_n_face_bkd_iter< D > >::next ( ) [inherited]`

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

#### Precondition

The iterator is valid.

## 10.331 `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 `mln::topo::internal::adj_lower_dim_connected_n_face_iterator< D >`.

### Public Member Functions

- void `next ()`

*Go to the next element.*



- [adj\\_lower\\_dim\\_connected\\_n\\_face\\_fwd\\_iter \( \)](#)  
*Construction.*

### 10.331.1 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.

### 10.331.2 Constructor & Destructor Documentation

**10.331.2.1** `template<unsigned D> mln::topo::adj_lower_dim_connected_n_face_fwd_iter< D >::adj_lower_dim_connected_n_face_fwd_iter ( ) [inline]`

Construction.

### 10.331.3 Member Function Documentation

**10.331.3.1** `void mln::Iterator< adj_lower_dim_connected_n_face_fwd_iter< D > >::next ( ) [inherited]`

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

#### Precondition

The iterator is valid.

## 10.332 mln::topo::adj\_lower\_face\_bkd\_iter< D > Class Template Reference

Backward iterator on all the adjacent (n-1)-faces of the n-face of an mln::complex<D>.

```
#include <adj_lower_face_iter.hh>
```

Inherits backward\_complex\_relative\_iterator\_base< topo::face< D >, algebraic\_face< D >, adj\_lower\_face\_bkd\_iter< D > >.

### Public Member Functions

- void [next \( \)](#)

*Go to the next element.*

- [adj\\_lower\\_face\\_bkd\\_iter\(\)](#)

*Construction.*

### 10.332.1 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.

### 10.332.2 Constructor & Destructor Documentation

**10.332.2.1 template<unsigned D> mln::topo::adj\_lower\_face\_bkd\_iter< D >::adj\_lower\_face\_bkd\_iter( ) [inline]**

Construction.

### 10.332.3 Member Function Documentation

**10.332.3.1 void mln::Iterator< adj\_lower\_face\_bkd\_iter< D > >::next( ) [inherited]**

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-define this method. The actual "next" operation has to be defined through the *next\_* method.

#### Precondition

The iterator is valid.

## 10.333 mln::topo::adj\_lower\_face\_fwd\_iter< D > Class Template Reference

Forward iterator on all the adjacent (n-1)-faces of the n-face of an mln::complex<D>.

```
#include <adj_lower_face_iter.hh>
```

Inherits forward\_complex\_relative\_iterator\_base< topo::face< D >, algebraic\_face< D >, adj\_lower\_face\_fwd\_iter< D > >.

## Public Member Functions

- void [next](#) ()  
*Go to the next element.*
- [adj\\_lower\\_face\\_fwd\\_iter](#) ()  
*Construction.*

### 10.333.1 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.

### 10.333.2 Constructor & Destructor Documentation

**10.333.2.1** `template<unsigned D> mln::topo::adj_lower_face_fwd_iter< D >::adj_lower_face_fwd_iter ( ) [inline]`

Construction.

### 10.333.3 Member Function Documentation

**10.333.3.1** `void mln::Iterator< adj_lower_face_fwd_iter< D > >::next ( ) [inherited]`

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

#### Precondition

The iterator is valid.

## 10.334 mln::topo::adj\_lower\_higher\_face\_bkd\_iter< D > Class Template Reference

Forward iterator on all the adjacent (n-1)-faces and (n+1)-faces of the n-face of an `mln::complex<D>`.

```
#include <adj_lower_higher_face_iter.hh>
```

Inherits `complex_relative_iterator_sequence< adj_higher_face_bkd_iter< D >, adj_lower_face_bkd_iter< D >, adj_lower_higher_face_bkd_iter< D > >`.

## Public Member Functions

- void `next` ()  
*Go to the next element.*
- `adj_lower_higher_face_bkd_iter` ()  
*Construction.*

### 10.334.1 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.

### 10.334.2 Constructor & Destructor Documentation

**10.334.2.1** `template<unsigned D> mln::topo::adj_lower_higher_face_bkd_iter< D >::adj_lower_higher_face_bkd_iter ( ) [inline]`

Construction.

### 10.334.3 Member Function Documentation

**10.334.3.1** `void mln::Iterator< adj_lower_higher_face_bkd_iter< D > >::next ( ) [inherited]`

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

#### Precondition

The iterator is valid.

## 10.335 `mln::topo::adj_lower_higher_face_fwd_iter< D >` Class Template Reference

Forward iterator on all the adjacent (n-1)-faces and (n+1)-faces of the n-face of an `mln::complex<D>`.

```
#include <adj_lower_higher_face_iter.hh>
```

Inherits `complex_relative_iterator_sequence< adj_lower_face_fwd_iter< D >, adj_higher_face_fwd_iter< D >, adj_lower_higher_face_fwd_iter< D >>`.

## Public Member Functions

- void [next](#) ()  
*Go to the next element.*
- [adj\\_lower\\_higher\\_face\\_fwd\\_iter](#) ()  
*Construction.*

### 10.335.1 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.

### 10.335.2 Constructor & Destructor Documentation

**10.335.2.1** `template<unsigned D> mln::topo::adj_lower_higher_face_fwd_iter< D >::adj_lower_higher_face_fwd_iter ( ) [inline]`

Construction.

### 10.335.3 Member Function Documentation

**10.335.3.1** `void mln::Iterator< adj_lower_higher_face_fwd_iter< D > >::next ( ) [inherited]`

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next*\_method.

#### Precondition

The iterator is valid.

## 10.336 mln::topo::adj\_m\_face\_bkd\_iter< D > Class Template Reference

Backward iterator on all the m-faces transitively adjacent to a (reference) n-face in a complex.

```
#include <adj_m_face_iter.hh>
```

Inherits `backward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_m_face_bkd_iter< D >>`, and `mln::topo::internal::adj_m_face_iterator< D >`.

## Public Member Functions

- void [next](#) ()  
*Go to the next element.*
- [adj\\_m\\_face\\_bkd\\_iter](#) ()  
*Construction.*
- `template<typename Fref >`  
[adj\\_m\\_face\\_bkd\\_iter](#) (const Fref &f\_ref, unsigned m)  
*Constructs an iterator, with *f\_ref* as reference face, and a target dimension equal to *m*.*

### 10.336.1 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.

### 10.336.2 Constructor & Destructor Documentation

**10.336.2.1** `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.

**10.336.2.2** `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*.

### 10.336.3 Member Function Documentation

**10.336.3.1** `void mln::Iterator< adj_m_face_bkd_iter< D > >::next ( ) [inherited]`

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

**Precondition**

The iterator is valid.

## 10.337 mln::topo::adj\_m\_face\_fwd\_iter< D > Class Template Reference

Forward iterator on all the m-faces transitively adjacent to a (reference) n-face in a complex.

```
#include <adj_m_face_iter.hh>
```

Inherits forward\_complex\_relative\_iterator\_base< topo::face< D >, algebraic\_face< D >, adj\_m\_face\_fwd\_iter< D > >, and mln::topo::internal::adj\_m\_face\_iterator< D >.

**Public Member Functions**

- void [next](#) ()  
*Go to the next element.*
- [adj\\_m\\_face\\_fwd\\_iter](#) ()  
*Construction.*
- template<typename Fref >  
[adj\\_m\\_face\\_fwd\\_iter](#) (const Fref &f\_ref, unsigned m)  
*Constructs an iterator, with f\_ref as reference face, and a target dimension equal to m.*

**10.337.1 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*<sub>\_) must be lower or equal to *D*.</sub>

If *m*<sub>\_) is equal to the dimension of the reference face, then the iterated set is empty.</sub>

**10.337.2 Constructor & Destructor Documentation**

**10.337.2.1** 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.

**10.337.2.2** `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*.

### 10.337.3 Member Function Documentation

**10.337.3.1** `void mln::Iterator< adj_m_face_fwd_iter< D > >::next ( ) [inherited]`

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

#### Precondition

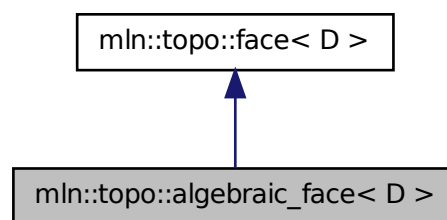
The iterator is valid.

## 10.338 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

- `algebraic_face ()`  
*Build a non-initialized algebraic face handle.*
- `algebraic_face (complex< D > &complex, unsigned n, unsigned face_id, bool sign)`  
*Build an algebraic face handle from complex and face\_id.*



- `template<unsigned N>`  
`algebraic_face` (const `algebraic_n_face`< N, D > &f)  
*Build a face handle from an `mln::topo::algebraic_n_face`.*
- `algebraic_face` (const `face`< D > &f, bool `sign`)  
*Build an algebraic face handle from an `mln::face`.*
- void `invalidate` ()  
*Invalidate this handle.*
- bool `is_valid` () const  
*Is this handle valid?*
  
- bool `sign` () const  
*Accessors.*
- void `set_sign` (bool `sign`)  
*Set the sign of this face.*
  
- `complex`< D > `cplx` () const  
*Accessors.*
- unsigned `n` () const  
*Return the dimension of the face.*
- unsigned `face_id` () const  
*Return the id of the face.*
- void `set_cplx` (const `complex`< D > &cplx)  
*Set the complex the face belongs to.*
- void `set_n` (unsigned n)  
*Set the dimension of the face.*
- void `inc_n` ()  
*Increment the dimension of the face.*
- void `dec_n` ()  
*Decrement the dimension of the face.*
- void `set_face_id` (unsigned face\_id)  
*Set the id of the face.*
- void `inc_face_id` ()  
*Increment the id of the face.*
- void `dec_face_id` ()  
*Decrement the id of the face.*
- `template<unsigned N>`  
`face_data`< N, D > & `data` () const

Return the `mln::topo::face_data` pointed by this handle.

- `std::vector< algebraic_face< D > > lower_dim_adj_faces () const`  
Return an array of face handles pointing to adjacent (n-1)-faces.
- `std::vector< algebraic_face< D > > higher_dim_adj_faces () const`  
Return an array of face handles pointing to adjacent (n+1)-faces.

### 10.338.1 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.

### 10.338.2 Constructor & Destructor Documentation

**10.338.2.1** `template<unsigned D> mln::topo::algebraic_face< D >::algebraic_face ( )`  
[inline]

Build a non-initialized algebraic face handle.

**10.338.2.2** `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`.

**10.338.2.3** `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()`.

**10.338.2.4** `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`.

### 10.338.3 Member Function Documentation

**10.338.3.1** `template<unsigned D> complex< D > mln::topo::face< D >::cplx ( ) const`  
[inline, inherited]

Accessors.

Return the complex the face belongs to.

Referenced by `mln::complex_psite< D, G >::complex_psite()`, `mln::topo::operator!=()`, and `mln::topo::operator==()`.

**10.338.3.2** `template<unsigned D> template<unsigned N> face_data< N, D > & mln::topo::face< D >::data( ) const [inline, inherited]`

Return the mln::topo::face\_data pointed by this handle.

References mln::topo::face< D >::is\_valid().

**10.338.3.3** `template<unsigned D> void mln::topo::face< D >::dec_face_id( ) [inline, inherited]`

Decrement the id of the face.

**10.338.3.4** `template<unsigned D> void mln::topo::face< D >::dec_n( ) [inline, inherited]`

Decrement the dimension of the face.

**10.338.3.5** `template<unsigned D> unsigned mln::topo::face< D >::face_id( ) const [inline, inherited]`

Return the id of the face.

Referenced by mln::geom::complex\_geometry< D, P >::operator(), and mln::topo::operator==( ).

**10.338.3.6** `template<unsigned D> std::vector< algebraic_face< D > > mln::topo::face< D >::higher_dim_adj_faces( ) const [inline, inherited]`

Return an array of face handles pointing to adjacent (n+1)-faces.

**10.338.3.7** `template<unsigned D> void mln::topo::face< D >::inc_face_id( ) [inline, inherited]`

Increment the id of the face.

**10.338.3.8** `template<unsigned D> void mln::topo::face< D >::inc_n( ) [inline, inherited]`

Increment the dimension of the face.

**10.338.3.9** `template<unsigned D> void mln::topo::face< D >::invalidate( ) [inline, inherited]`

Invalidate this handle.

References mln::topo::face< D >::set\_face\_id(), and mln::topo::face< D >::set\_n().

**10.338.3.10** `template<unsigned D> bool mln::topo::face< D >::is_valid( ) const [inline, inherited]`

Is this handle valid?

Referenced by `mln::topo::face< D >::data()`.

**10.338.3.11** `template<unsigned D> std::vector< algebraic_face< D > > mln::topo::face< D >::lower_dim_adj_faces ( ) const [inline, inherited]`

Return an array of face handles pointing to adjacent (n-1)-faces.

**10.338.3.12** `template<unsigned D> unsigned mln::topo::face< D >::n ( ) const [inline, inherited]`

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==()`.

**10.338.3.13** `template<unsigned D> void mln::topo::face< D >::set_cplx ( const complex< D > & cplx ) [inline, inherited]`

Set the complex the face belongs to.

**10.338.3.14** `template<unsigned D> void mln::topo::face< D >::set_face_id ( unsigned face_id ) [inline, inherited]`

Set the id of the face.

Referenced by `mln::topo::face< D >::invalidate()`.

**10.338.3.15** `template<unsigned D> void mln::topo::face< D >::set_n ( unsigned n ) [inline, inherited]`

Set the dimension of the face.

Referenced by `mln::topo::face< D >::invalidate()`.

**10.338.3.16** `template<unsigned D> void mln::topo::algebraic_face< D >::set_sign ( bool sign ) [inline]`

Set the sign of this face.

**10.338.3.17** `template<unsigned D> bool mln::topo::algebraic_face< D >::sign ( ) const [inline]`

Accessors.

Return the sign of this face.

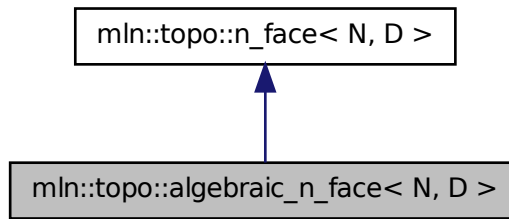
Referenced by `mln::topo::operator==()`.

## 10.339 mln::topo::algebraic\_n\_face< N, D > Class Template Reference

Algebraic N-face handle in a complex.

```
#include <algebraic_n_face.hh>
```

Inheritance diagram for mln::topo::algebraic\_n\_face< N, D >:



### Public Member Functions

- [algebraic\\_n\\_face](#) ()  
*Build a non-initialized algebraic face handle.*
- [algebraic\\_n\\_face](#) ([complex](#)< D > &[complex](#), unsigned face\_id, bool [sign](#))  
*Build an algebraic face handle from complex and face\_id.*
- [algebraic\\_n\\_face](#) (const [n\\_face](#)< N, D > &f, bool [sign](#))  
*Build an algebraic face handle from an mln::n\_face.*
- void [invalidate](#) ()  
*Invalidate this handle.*
- bool [is\\_valid](#) () const  
*Is this handle valid?*
- bool [sign](#) () const  
*Accessors.*
- void [set\\_sign](#) (bool [sign](#))  
*Set the sign of this face.*
- [complex](#)< D > [cplx](#) () const  
*Accessors.*

- unsigned `face_id` () const  
*Return the id of the face.*
- void `set_cplx` (const `complex`< D > &cplx)  
*Set the complex the face belongs to.*
- unsigned `n` () const  
*Return the dimension of the face.*
- void `set_face_id` (unsigned face\_id)  
*Set the id of the face.*
- void `inc_face_id` ()  
*Increment the id of the face.*
- void `dec_face_id` ()  
*Decrement the id of the face.*
- `face_data`< N, D > & `data` () const  
*Return the `mln::topo::face_data` pointed by this handle.*
- `std::vector`< `algebraic_n_face`< N-1, D > > `lower_dim_adj_faces` () const  
*Return an array of face handles pointing to adjacent (n-1)-faces.*
- `std::vector`< `algebraic_n_face`< N+1, D > > `higher_dim_adj_faces` () const  
*Return an array of face handles pointing to adjacent (n+1)-faces.*

### 10.339.1 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.

### 10.339.2 Constructor & Destructor Documentation

**10.339.2.1** `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()`.

**10.339.2.2** `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`.

**10.339.2.3** `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.

### 10.339.3 Member Function Documentation

**10.339.3.1** `template<unsigned N, unsigned D> complex< D > mln::topo::n_face< N, D >::cplx ( ) const [inline, inherited]`

Accessors.

Return the complex the face belongs to.

Referenced by mln::topo::n\_faces\_set< N, D >::add(), mln::topo::operator!=( ), and mln::topo::operator==( ).

**10.339.3.2** `template<unsigned N, unsigned D> face_data< N, D > & mln::topo::n_face< N, D >::data ( ) const [inline, inherited]`

Return the mln::topo::face\_data pointed by this handle.

References mln::topo::n\_face< N, D >::is\_valid().

**10.339.3.3** `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::dec_face_id ( ) [inline, inherited]`

Decrement the id of the face.

**10.339.3.4** `template<unsigned N, unsigned D> unsigned mln::topo::n_face< N, D >::face_id ( ) const [inline, inherited]`

Return the id of the face.

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

**10.339.3.5** `template<unsigned N, unsigned D> std::vector< algebraic_n_face< N+1, D > > mln::topo::n_face< N, D >::higher_dim_adj_faces ( ) const [inline, inherited]`

Return an array of face handles pointing to adjacent (n+1)-faces.

References mln::topo::n\_face< N, D >::is\_valid().

Referenced by mln::topo::edge().

**10.339.3.6** `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::inc_face_id ( ) [inline, inherited]`

Increment the id of the face.

**10.339.3.7** `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::invalidate ( )`  
`[inline, inherited]`

Invalidate this handle.

References `mln::topo::n_face< N, D >::set_face_id()`.

**10.339.3.8** `template<unsigned N, unsigned D> bool mln::topo::n_face< N, D >::is_valid ( )`  
`const [inline, inherited]`

Is this handle valid?

Referenced by `mln::topo::algebraic_n_face< N, D >::algebraic_n_face()`, `mln::topo::n_face< N, D >::data()`, `mln::topo::n_face< N, D >::higher_dim_adj_faces()`, `mln::topo::n_face< N, D >::lower_dim_adj_faces()`, and `mln::topo::n_face< N, D >::n_face()`.

**10.339.3.9** `template<unsigned N, unsigned D> std::vector< algebraic_n_face< N-1, D > > mln::topo::n_face< N, D >::lower_dim_adj_faces ( ) const` `[inline, inherited]`

Return an array of face handles pointing to adjacent (n-1)-faces.

References `mln::topo::n_face< N, D >::is_valid()`.

**10.339.3.10** `template<unsigned N, unsigned D> unsigned mln::topo::n_face< N, D >::n ( )`  
`const [inline, inherited]`

Return the dimension of the face.

**10.339.3.11** `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::set_cplx ( const`  
`complex< D > & cplx ) [inline, inherited]`

Set the complex the face belongs to.

**10.339.3.12** `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::set_face_id (`  
`unsigned face_id ) [inline, inherited]`

Set the id of the face.

Referenced by `mln::topo::n_face< N, D >::invalidate()`.

**10.339.3.13** `template<unsigned N, unsigned D> void mln::topo::algebraic_n_face< N, D >::set_sign ( bool`  
`sign ) [inline]`

Set the sign of this face.

**10.339.3.14** `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==().

## 10.340 mln::topo::center\_only\_iter< D > Class Template Reference

[Iterator](#) on all the adjacent (n-1)-faces of the n-face of an mln::complex<D>.

```
#include <center_only_iter.hh>
```

Inherits forward\_complex\_relative\_iterator\_base< topo::face< D >, algebraic\_face< D >, center\_only\_iter< D > >.

### Public Member Functions

- void [next](#) ()  
*Go to the next element.*
- [center\\_only\\_iter](#) ()  
*Construction.*

### 10.340.1 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
```

### 10.340.2 Constructor & Destructor Documentation

**10.340.2.1** `template<unsigned D> mln::topo::center_only_iter< D >::center_only_iter ( )`  
`[inline]`

Construction.

### 10.340.3 Member Function Documentation

#### 10.340.3.1 void mln::Iterator< center\_only\_iter< D > >::next ( ) [inherited]

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_method*.

#### Precondition

The iterator is valid.

## 10.341 mln::topo::centered\_bkd\_iter\_adapter< D, I > Class Template Reference

Forward complex relative iterator adapters adding the central (reference) point to the set of iterated faces.

```
#include <centered_iter_adapter.hh>
```

Inherits complex\_relative\_iterator\_sequence< I, center\_only\_iter< D >, centered\_bkd\_iter\_adapter< D, I > >.

### Public Member Functions

- void [next](#) ()  
*Go to the next element.*
- [centered\\_bkd\\_iter\\_adapter](#) ()  
*Construction.*

### 10.341.1 Detailed Description

```
template<unsigned D, typename I> class mln::topo::centered_bkd_iter_adapter< D, I >
```

Forward complex relative iterator adapters adding the central (reference) point to the set of iterated faces.

#### Template Parameters

- D* The dimension of the complex this iterator belongs to.
- I* The adapted complex relative iterator.

### 10.341.2 Constructor & Destructor Documentation

#### 10.341.2.1 template<unsigned D, typename I > mln::topo::centered\_bkd\_iter\_adapter< D, I >::centered\_bkd\_iter\_adapter ( ) [inline]

Construction.

### 10.341.3 Member Function Documentation

#### 10.341.3.1 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-define this method. The actual "next" operation has to be defined through the *next\_method*.

#### Precondition

The iterator is valid.

## 10.342 mln::topo::centered\_fwd\_iter\_adapter< D, I > Class Template Reference

Backward complex relative iterator adapters adding the central (reference) point to the set of iterated faces.

```
#include <centered_iter_adapter.hh>
```

Inherits `complex_relative_iterator_sequence< center_only_iter< D >, I, centered_fwd_iter_adapter< D, I >>`.

### Public Member Functions

- void [next](#) ()  
*Go to the next element.*
- [centered\\_fwd\\_iter\\_adapter](#) ()  
*Construction.*

### 10.342.1 Detailed Description

```
template<unsigned D, typename I> class mln::topo::centered_fwd_iter_adapter< D, I >
```

Backward complex relative iterator adapters adding the central (reference) point to the set of iterated faces.

#### Template Parameters

- D* The dimension of the complex this iterator belongs to.
- I* The adapted complex relative iterator.

### 10.342.2 Constructor & Destructor Documentation

#### 10.342.2.1 template<unsigned D, typename I > mln::topo::centered\_fwd\_iter\_adapter< D, I >::centered\_fwd\_iter\_adapter ( ) [inline]

Construction.

### 10.342.3 Member Function Documentation

#### 10.342.3.1 void mln::Iterator< centered\_fwd\_iter\_adapter< D, I > >::next ( ) [inherited]

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_method*.

#### Precondition

The iterator is valid.

## 10.343 mln::topo::complex< D > Class Template Reference

General complex of dimension D.

```
#include <complex.hh>
```

### Public Types

- typedef [face\\_bkd\\_iter< D > bkd\\_citer](#)  
*Backward mln::Iterator type iterating on all faces.*
- typedef [face\\_fwd\\_iter< D > fwd\\_citer](#)  
*Forward mln::Iterator type iterating on all faces.*

### Public Member Functions

- const void \* [addr](#) () const  
*Get the address of the data of this complex.*
- [complex](#) ()  
*Complex construction.*
- [n\\_face< 0, D > add\\_face](#) ()  
*Add a 0-face to the complex.*
- template<unsigned N>  
[n\\_face< N+1, D > add\\_face](#) (const [n\\_faces\\_set< N, D > &adjacent\\_faces](#))  
*Add a (N+1)-face to the complex (with N >= 0).*
- unsigned [nfaces](#) () const  
*Static manipulators.*

- `template<unsigned N>`  
`unsigned nfaces_of_static_dim () const`  
*Return the number of N-faces.*
- `unsigned nfaces_of_dim (unsigned n) const`  
*Dynamic manipulators.*
- `void print (std::ostream &ostr) const`  
*Pretty-printing.*
- `template<unsigned N>`  
`void print_faces (std::ostream &ostr) const`  
*Print the faces of dimension N.*

### 10.343.1 Detailed Description

`template<unsigned D> class mln::topo::complex< D >`

General complex of dimension D.

### 10.343.2 Member Typedef Documentation

**10.343.2.1 `template<unsigned D> typedef face_bkd_iter<D> mln::topo::complex< D >::bkd_citer`**

Backward `mln::Iterator` type iterating on all faces.

**10.343.2.2 `template<unsigned D> typedef face_fwd_iter<D> mln::topo::complex< D >::fwd_citer`**

Forward `mln::Iterator` type iterating on all faces.

### 10.343.3 Constructor & Destructor Documentation

**10.343.3.1 `template<unsigned D> mln::topo::complex< D >::complex ( ) [inline]`**

Complex construction.

Create a new D-complex.

### 10.343.4 Member Function Documentation

**10.343.4.1 `template<unsigned D> n_face< 0, D > mln::topo::complex< D >::add_face ( ) [inline]`**

Add a 0-face to the complex.

**10.343.4.2** `template<unsigned D> template<unsigned N> n_face< N+1, D >  
mln::topo::complex< D >::add_face ( const n_faces_set< N, D > & adjacent_faces )  
[inline]`

Add a (N+1)-face to the complex (with  $N \geq 0$ ).

#### Parameters

*adjacent\_faces* The (N-1)-faces adjacent to the new N-face.

References `mln::topo::n_faces_set< N, D >::faces()`.

**10.343.4.3** `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.

**10.343.4.4** `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.

**10.343.4.5** `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).

**10.343.4.6** `template<unsigned D> template<unsigned N> unsigned mln::topo::complex< D >::nfaces_of_static_dim ( ) const [inline]`

Return the number of N-faces.

**10.343.4.7** `template<unsigned D> void mln::topo::complex< D >::print ( std::ostream & ostr ) const [inline]`

Pretty-printing.

Print the complex.

Referenced by `mln::topo::operator<<()`.

**10.343.4.8** `template<unsigned D> template<unsigned N> void mln::topo::complex< D >::print_faces ( std::ostream & ostr ) const [inline]`

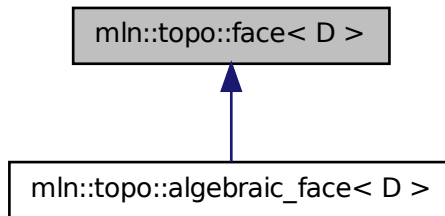
Print the faces of dimension N.

## 10.344 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

- `face ()`  
*Build a non-initialized face handle.*
- `face (complex< D > &complex, unsigned n, unsigned face_id)`  
*Build a face handle from complex and face\_id.*
- `template<unsigned N>`  
`face (const n_face< N, D > &f)`  
*Build a face handle from an mln::topo::n\_face.*
- `void invalidate ()`  
*Invalidate this handle.*
- `bool is_valid () const`  
*Is this handle valid?*
- `complex< D > cplx () const`  
*Accessors.*
- `unsigned n () const`

*Return the dimension of the face.*

- unsigned `face_id` () const  
*Return the id of the face.*
- void `set_cplx` (const `complex`< D > &cplx)  
*Set the complex the face belongs to.*
- void `set_n` (unsigned n)  
*Set the dimension of the face.*
- void `inc_n` ()  
*Increment the dimension of the face.*
- void `dec_n` ()  
*Decrement the dimension of the face.*
- void `set_face_id` (unsigned face\_id)  
*Set the id of the face.*
- void `inc_face_id` ()  
*Increment the id of the face.*
- void `dec_face_id` ()  
*Decrement the id of the face.*
- template<unsigned N>  
  `face_data`< N, D > & `data` () const  
  *Return the `mln::topo::face_data` pointed by this handle.*
- `std::vector`< `algebraic_face`< D > > `lower_dim_adj_faces` () const  
  *Return an array of face handles pointing to adjacent (n-1)-faces.*
- `std::vector`< `algebraic_face`< D > > `higher_dim_adj_faces` () const  
  *Return an array of face handles pointing to adjacent (n+1)-faces.*

### 10.344.1 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.

### 10.344.2 Constructor & Destructor Documentation

**10.344.2.1** `template<unsigned D> mln::topo::face< D >::face ( ) [inline]`

Build a non-initialized face handle.

**10.344.2.2** `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`.



**10.344.2.3** `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](#).

### 10.344.3 Member Function Documentation

**10.344.3.1** `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==()`.

**10.344.3.2** `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.

References `mln::topo::face< D >::is_valid()`.

**10.344.3.3** `template<unsigned D> void mln::topo::face< D >::dec_face_id ( ) [inline]`

Decrement the id of the face.

**10.344.3.4** `template<unsigned D> void mln::topo::face< D >::dec_n ( ) [inline]`

Decrement the dimension of the face.

**10.344.3.5** `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==()`.

**10.344.3.6** `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.

**10.344.3.7** `template<unsigned D> void mln::topo::face< D >::inc_face_id ( ) [inline]`

Increment the id of the face.

**10.344.3.8** `template<unsigned D> void mln::topo::face< D >::inc_n ( ) [inline]`

Increment the dimension of the face.

**10.344.3.9** `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()`.

**10.344.3.10** `template<unsigned D> bool mln::topo::face< D >::is_valid ( ) const [inline]`

Is this handle valid?

Referenced by `mln::topo::face< D >::data()`.

**10.344.3.11** `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.

**10.344.3.12** `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==()`.

**10.344.3.13** `template<unsigned D> void mln::topo::face< D >::set_cplx ( const complex< D > & cplx ) [inline]`

Set the complex the face belongs to.

**10.344.3.14** `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()`.

**10.344.3.15** `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()`.

## 10.345 `mln::topo::face_bkd_iter< D >` Class Template Reference

Backward iterator on all the faces of an `mln::complex<D>`.

```
#include <face_iter.hh>
```

Inherits `complex_set_iterator_base< topo::face< D >, face_bkd_iter< D > >`.

## Public Member Functions

- void `next()`  
*Go to the next element.*
- `face_bkd_iter()`  
*Construction and assignment.*
- void `start()`  
*Manipulation.*

### 10.345.1 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.

### 10.345.2 Constructor & Destructor Documentation

**10.345.2.1** `template<unsigned D> mln::topo::face_bkd_iter< D >::face_bkd_iter ( )`  
[inline]

Construction and assignment.

### 10.345.3 Member Function Documentation

**10.345.3.1** `void mln::Iterator< face_bkd_iter< D > >::next ( )` [inherited]

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

#### Precondition

The iterator is valid.

**10.345.3.2** `template<unsigned D> void mln::topo::face_bkd_iter< D >::start ( ) [inline]`

Manipulation.

Start an iteration.

**10.346** `mln::topo::face_fwd_iter< D >` Class Template Reference

Forward iterator on all the faces of an `mln::complex<D>`.

```
#include <face_iter.hh>
```

Inherits `complex_set_iterator_base< topo::face< D >, face_fwd_iter< D > >`.

**Public Member Functions**

- void [next](#) ()  
*Go to the next element.*
- [face\\_fwd\\_iter](#) ()  
*Construction and assignment.*
- void [start](#) ()  
*Manipulation.*

**10.346.1 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.

**10.346.2 Constructor & Destructor Documentation**

**10.346.2.1** `template<unsigned D> mln::topo::face_fwd_iter< D >::face_fwd_iter ( ) [inline]`

Construction and assignment.

**10.346.3 Member Function Documentation**

**10.346.3.1** `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-define this method. The actual "next" operation has to be defined through the *next\_* method.

**Precondition**

The iterator is valid.

**10.346.3.2** `template<unsigned D> void mln::topo::face_fwd_iter< D >::start ( ) [inline]`

Manipulation.

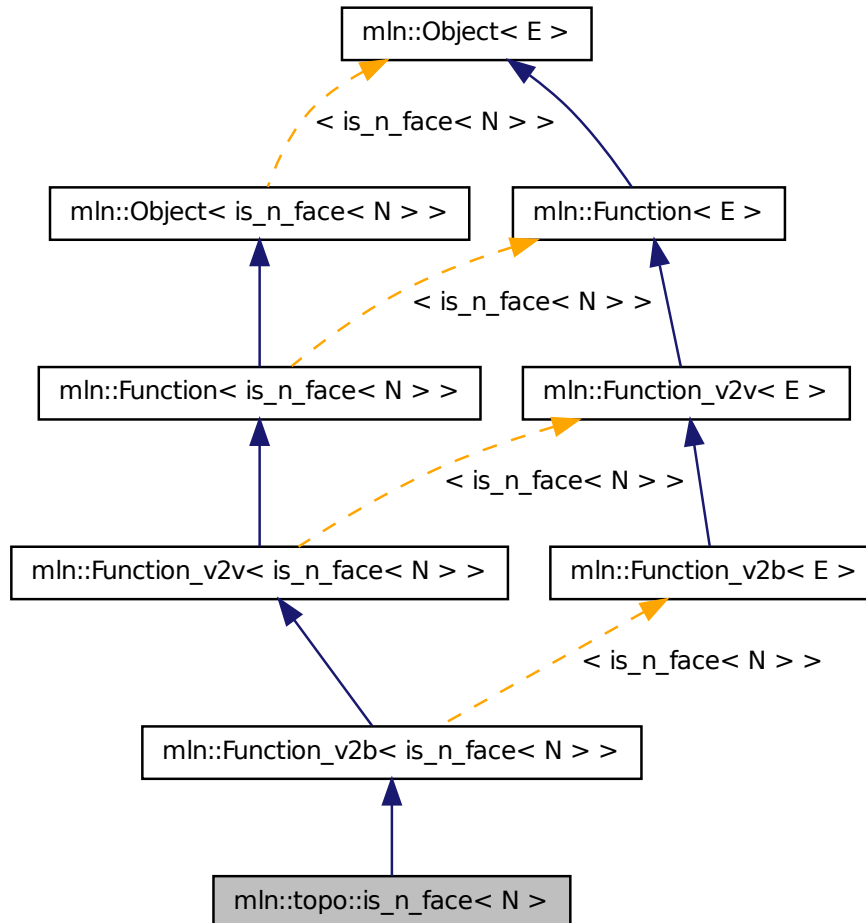
Test if the iterator is valid.

**10.347 mln::topo::is\_n\_face< N > Struct Template Reference**

A functor testing whether a [mln::complex\\_psit](#)e is an N-face.

```
#include <is_n_face.hh>
```

Inheritance diagram for `mln::topo::is_n_face< N >`:



### 10.347.1 Detailed Description

```
template<unsigned N> struct mln::topo::is_n_face< N >
```

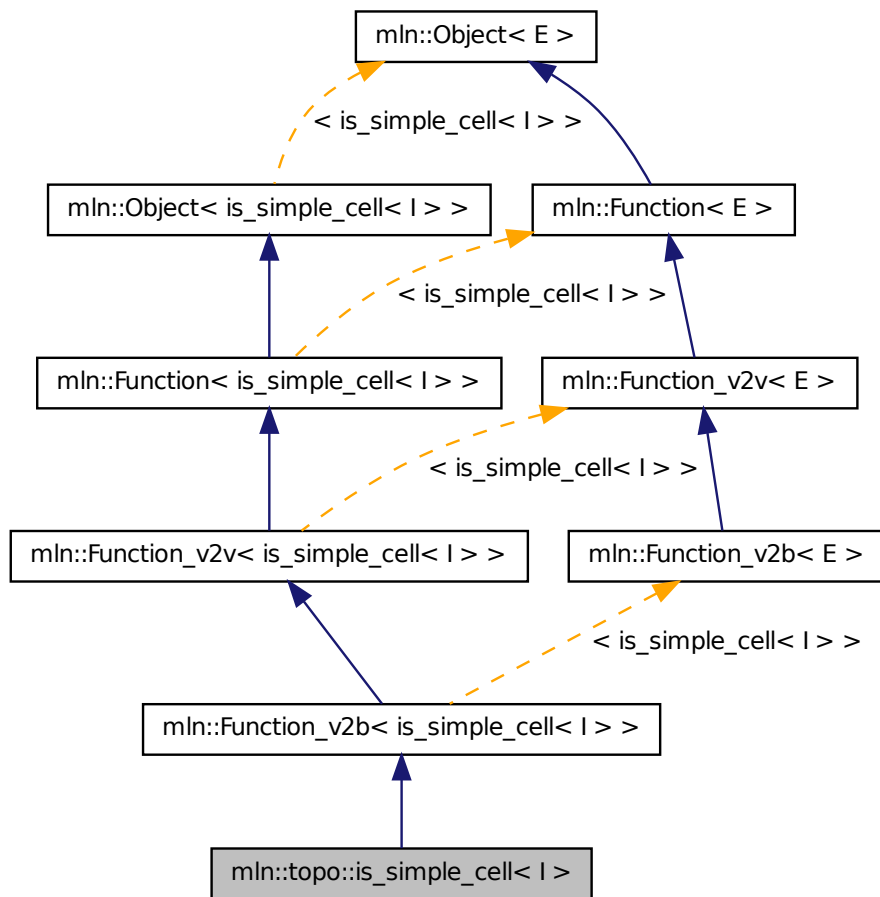
A functor testing whether a [mln::complex\\_site](#) is an N-face.

## 10.348 mln::topo::is\_simple\_cell< I > Class Template Reference

A predicate for the simplicity of a point based on the collapse property of the attachment.

```
#include <is_simple_cell.hh>
```

Inheritance diagram for mln::topo::is\_simple\_cell< I >:



## Public Types

- typedef `mln::complex_psite< D, G > psite`  
*Psite type.*
- typedef `bool result`  
*Result type of the functor.*

## Public Member Functions

- typedef `mln_geom (I) G`  
*Geometry of the image.*

- bool `operator()` (const `mln::complex_psite`< I::dim, `mln_geom`(I)> &p) const  
*Based on the algorithm A2 from couprie.08.pami.*
- void `set_image` (const `mln::Image`< I > &ima)  
*Set the underlying image.*

## Static Public Attributes

- static const unsigned `D` = I::dim  
*Dimension of the image (and therefore of the complex).*

### 10.348.1 Detailed Description

`template<typename I> class mln::topo::is_simple_cell< I >`

A predicate for the simplicity of a point based on the collapse property of the attachment. The functor does not actually take a cell as input, but a face that is expected to be a D-facet.

### 10.348.2 Member Typedef Documentation

**10.348.2.1** `template<typename I > typedef mln::complex_psite<D, G>  
mln::topo::is_simple_cell< I >::psite`

Psite type.

**10.348.2.2** `template<typename I > typedef bool mln::topo::is_simple_cell< I >::result`

Result type of the functor.

Reimplemented from `mln::Function_v2b< is_simple_cell< I > >`.

### 10.348.3 Member Function Documentation

**10.348.3.1** `template<typename I > typedef mln::topo::is_simple_cell< I >::mln_geom ( I )`

Geometry of the image.

**10.348.3.2** `template<typename I > bool mln::topo::is_simple_cell< I >::operator() ( const  
mln::complex_psite< I::dim, mln_geom(I)> & p ) const [inline]`

Based on the algorithm A2 from couprie.08.pami.

References `mln::make::attachment()`.



**10.348.3.3** `template<typename I > void mln::topo::is_simple_cell< I >::set_image ( const mln::Image< I > & ima ) [inline]`

Set the underlying image.

### 10.348.4 Member Data Documentation

**10.348.4.1** `template<typename I > const unsigned mln::topo::is_simple_cell< I >::D = I::dim [static]`

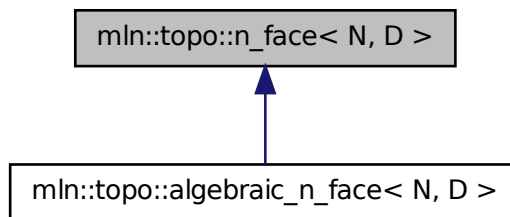
Dimension of the image (and therefore of the complex).

## 10.349 mln::topo::n\_face< N, D > Class Template Reference

N-face handle in a complex.

```
#include <n_face.hh>
```

Inheritance diagram for mln::topo::n\_face< N, D >:



### Public Member Functions

- void `invalidate ()`  
*Invalidate this handle.*
- bool `is_valid () const`  
*Is this handle valid?*
- `n_face ()`  
*Build a non-initialized face handle.*
- `n_face (complex< D > &complex, unsigned face_id)`  
*Build a face handle from complex and face\_id.*

- `complex< D > cplx () const`  
*Accessors.*
- `unsigned face_id () const`  
*Return the id of the face.*
- `void set_cplx (const complex< D > &cplx)`  
*Set the complex the face belongs to.*
- `unsigned n () const`  
*Return the dimension of the face.*
- `void set_face_id (unsigned face_id)`  
*Set the id of the face.*
- `void inc_face_id ()`  
*Increment the id of the face.*
- `void dec_face_id ()`  
*Decrement the id of the face.*
- `face_data< N, D > & data () const`  
*Return the `mln::topo::face_data` pointed by this handle.*
- `std::vector< algebraic_n_face< N-1, D > > lower_dim_adj_faces () const`  
*Return an array of face handles pointing to adjacent (n-1)-faces.*
- `std::vector< algebraic_n_face< N+1, D > > higher_dim_adj_faces () const`  
*Return an array of face handles pointing to adjacent (n+1)-faces.*

### 10.349.1 Detailed Description

`template<unsigned N, unsigned D> class mln::topo::n_face< N, D >`

`N-face` handle in a complex. Contrary to an `mln::topo::face`, the dimension of an `mln::topo::n_face` is fixed.

### 10.349.2 Constructor & Destructor Documentation

**10.349.2.1** `template<unsigned N, unsigned D> mln::topo::n_face< N, D >::n_face ( )`  
`[inline]`

Build a non-initialized face handle.

References `mln::topo::n_face< N, D >::is_valid()`.

**10.349.2.2** `template<unsigned N, unsigned D> mln::topo::n_face< N, D >::n_face ( complex< D > & complex, unsigned face_id )` `[inline]`

Build a face handle from `complex` and `face_id`.

### 10.349.3 Member Function Documentation

**10.349.3.1** `template<unsigned N, unsigned D> complex< D > mln::topo::n_face< N, D >::cplx ( ) const [inline]`

Accessors.

Return the complex the face belongs to.

Referenced by `mln::topo::n_faces_set< N, D >::add()`, `mln::topo::operator!=()`, and `mln::topo::operator==()`.

**10.349.3.2** `template<unsigned N, unsigned D> face_data< N, D > & mln::topo::n_face< N, D >::data ( ) const [inline]`

Return the `mln::topo::face_data` pointed by this handle.

References `mln::topo::n_face< N, D >::is_valid()`.

**10.349.3.3** `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::dec_face_id ( ) [inline]`

Decrement the id of the face.

**10.349.3.4** `template<unsigned N, unsigned D> unsigned mln::topo::n_face< N, D >::face_id ( ) const [inline]`

Return the id of the face.

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

**10.349.3.5** `template<unsigned N, unsigned D> std::vector< algebraic_n_face< N+1, D > > mln::topo::n_face< N, D >::higher_dim_adj_faces ( ) const [inline]`

Return an array of face handles pointing to adjacent (n+1)-faces.

References `mln::topo::n_face< N, D >::is_valid()`.

Referenced by `mln::topo::edge()`.

**10.349.3.6** `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::inc_face_id ( ) [inline]`

Increment the id of the face.

**10.349.3.7** `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()`.

**10.349.3.8** `template<unsigned N, unsigned D> bool mln::topo::n_face< N, D >::is_valid ( ) const [inline]`

Is this handle valid?

Referenced by `mln::topo::algebraic_n_face< N, D >::algebraic_n_face()`, `mln::topo::n_face< N, D >::data()`, `mln::topo::n_face< N, D >::higher_dim_adj_faces()`, `mln::topo::n_face< N, D >::lower_dim_adj_faces()`, and `mln::topo::n_face< N, D >::n_face()`.

**10.349.3.9** `template<unsigned N, unsigned D> std::vector< algebraic_n_face< N-1, D > > mln::topo::n_face< N, D >::lower_dim_adj_faces ( ) const [inline]`

Return an array of face handles pointing to adjacent (n-1)-faces.

References `mln::topo::n_face< N, D >::is_valid()`.

**10.349.3.10** `template<unsigned N, unsigned D> unsigned mln::topo::n_face< N, D >::n ( ) const [inline]`

Return the dimension of the face.

**10.349.3.11** `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::set_cplx ( const complex< D > & cplx ) [inline]`

Set the complex the face belongs to.

**10.349.3.12** `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::set_face_id ( unsigned face_id ) [inline]`

Set the id of the face.

Referenced by `mln::topo::n_face< N, D >::invalidate()`.

## 10.350 mln::topo::n\_face\_bkd\_iter< D > Class Template Reference

Backward iterator on all the faces of an `mln::complex<D>`.

```
#include <n_face_iter.hh>
```

Inherits `complex_set_iterator_base< topo::face< D >, n_face_bkd_iter< D > >`.

### Public Member Functions

- `void next ()`  
*Go to the next element.*
- `n_face_bkd_iter ()`  
*Construction and assignment.*

- void `start()`  
*Manipulation.*
- unsigned `n()` const  
*Accessors.*

### 10.350.1 Detailed Description

`template<unsigned D> class mln::topo::n_face_bkd_iter< D >`

Backward iterator on all the faces of an `mln::complex<D>`.

#### Template Parameters

*D* The dimension of the complex this iterator belongs to.

### 10.350.2 Constructor & Destructor Documentation

**10.350.2.1** `template<unsigned D> mln::topo::n_face_bkd_iter< D >::n_face_bkd_iter ( )`  
`[inline]`

Construction and assignment.

### 10.350.3 Member Function Documentation

**10.350.3.1** `template<unsigned D> unsigned mln::topo::n_face_bkd_iter< D >::n ( ) const`  
`[inline]`

Accessors.

Shortcuts to `face_`'s accessors.

Referenced by `mln::topo::n_face_bkd_iter< D >::start()`.

**10.350.3.2** `void mln::Iterator< n_face_bkd_iter< D > >::next ( ) [inherited]`

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

#### Precondition

The iterator is valid.

**10.350.3.3** `template<unsigned D> void mln::topo::n_face_bkd_iter< D >::start ( ) [inline]`

Manipulation.

Start an iteration.

References `mln::topo::n_face_bkd_iter< D >::n()`.

**10.351** `mln::topo::n_face_fwd_iter< D >` Class Template Reference

Forward iterator on all the faces of an `mln::complex<D>`.

```
#include <n_face_iter.hh>
```

Inherits `complex_set_iterator_base< topo::face< D >, n_face_fwd_iter< D > >`.

**Public Member Functions**

- void `next ()`  
*Go to the next element.*
  
- `n_face_fwd_iter ()`  
*Construction and assignment.*
  
- void `start ()`  
*Manipulation.*
  
- unsigned `n () const`  
*Accessors.*

**10.351.1 Detailed Description**

```
template<unsigned D> class mln::topo::n_face_fwd_iter< D >
```

Forward iterator on all the faces of an `mln::complex<D>`.

**Template Parameters**

*D* The dimension of the complex this iterator belongs to.

**10.351.2 Constructor & Destructor Documentation**

**10.351.2.1** `template<unsigned D> mln::topo::n_face_fwd_iter< D >::n_face_fwd_iter ( ) [inline]`

Construction and assignment.

### 10.351.3 Member Function Documentation

**10.351.3.1** `template<unsigned D> unsigned mln::topo::n_face_fwd_iter< D >::n ( ) const`  
`[inline]`

Accessors.

Shortcuts to face\_'s accessors.

**10.351.3.2** `void mln::Iterator< n_face_fwd_iter< D > >::next ( ) [inherited]`

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

#### Precondition

The iterator is valid.

**10.351.3.3** `template<unsigned D> void mln::topo::n_face_fwd_iter< D >::start ( ) [inline]`

Manipulation.

Test if the iterator is valid.

## 10.352 mln::topo::n\_faces\_set< N, D > Class Template Reference

Set of face handles of dimension N.

```
#include <n_faces_set.hh>
```

### Public Types

- typedef std::vector< [algebraic\\_n\\_face< N, D >](#) > [faces\\_type](#)  
*The type of the set of face handles.*

### Public Member Functions

- void [add](#) (const [algebraic\\_n\\_face< N, D >](#) &f)  
*Append an algebraic face f to the set.*
- void [reserve](#) (size\_t n)  
*Reserve n cells in the set.*
- const [faces\\_type](#) & [faces](#) () const  
*Accessors.*

### 10.352.1 Detailed Description

`template<unsigned N, unsigned D> class mln::topo::n_faces_set< N, D >`

Set of face handles of dimension N.

### 10.352.2 Member Typedef Documentation

**10.352.2.1** `template<unsigned N, unsigned D> typedef std::vector< algebraic_n_face<N, D> > mln::topo::n_faces_set< N, D >::faces_type`

The type of the set of face handles.

### 10.352.3 Member Function Documentation

**10.352.3.1** `template<unsigned N, unsigned D> void mln::topo::n_faces_set< N, D >::add ( const algebraic_n_face< N, D > & f ) [inline]`

Append an algebraic face  $f$  to the set.

References `mln::topo::n_face< N, D >::cplx()`.

Referenced by `mln::topo::operator+()`, and `mln::topo::operator-()`.

**10.352.3.2** `template<unsigned N, unsigned D> const std::vector< algebraic_n_face< N, D > > & mln::topo::n_faces_set< N, D >::faces ( ) const [inline]`

Accessors.

Return the set of handles.

Referenced by `mln::topo::complex< D >::add_face()`.

**10.352.3.3** `template<unsigned N, unsigned D> void mln::topo::n_faces_set< N, D >::reserve ( size_t n ) [inline]`

Reserve  $n$  cells in the set.

This methods does not change the content of `faces_`; it only pre-allocate memory. Method `reserve` is provided for efficiency purpose, and its use is completely optional.

## 10.353 mln::topo::static\_n\_face\_bkd\_iter< N, D > Class Template Reference

Backward iterator on all the N-faces of a `mln::complex<D>`.

```
#include <static_n_face_iter.hh>
```

Inherits `complex_set_iterator_base< topo::face< D >, static_n_face_bkd_iter< N, D > >`.



## Public Member Functions

- void `next` ()  
*Go to the next element.*
- `static_n_face_bkd_iter` ()  
*Construction and assignment.*
- void `start` ()  
*Manipulation.*

### 10.353.1 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.

### 10.353.2 Constructor & Destructor Documentation

**10.353.2.1** `template<unsigned N, unsigned D> mln::topo::static_n_face_bkd_iter< N, D >::static_n_face_bkd_iter ( ) [inline]`

Construction and assignment.

### 10.353.3 Member Function Documentation

**10.353.3.1** `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.

**10.353.3.2** `template<unsigned N, unsigned D> void mln::topo::static_n_face_bkd_iter< N, D >::start ( ) [inline]`

Manipulation.

Start an iteration.

## 10.354 mln::topo::static\_n\_face\_fwd\_iter< N, D > Class Template Reference

Forward iterator on all the `N`-faces of a `mln::complex<D>`.

```
#include <static_n_face_iter.hh>
```

Inherits `complex_set_iterator_base< topo::face< D >, static_n_face_fwd_iter< N, D > >`.

### Public Member Functions

- void `next()`  
*Go to the next element.*
- `static_n_face_fwd_iter()`  
*Construction and assignment.*
- void `start()`  
*Manipulation.*

### 10.354.1 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.

### 10.354.2 Constructor & Destructor Documentation

**10.354.2.1** `template<unsigned N, unsigned D> mln::topo::static_n_face_fwd_iter< N, D >::static_n_face_fwd_iter( ) [inline]`

Construction and assignment.

### 10.354.3 Member Function Documentation

**10.354.3.1** `void mln::Iterator< static_n_face_fwd_iter< N, D > >::next( ) [inherited]`

Go to the next element.

#### Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

**Precondition**

The iterator is valid.

**10.354.3.2** `template<unsigned N, unsigned D> void mln::topo::static_n_face_fwd_iter< N, D >::start ( ) [inline]`

Manipulation.

Test if the iterator is valid.

**10.355 mln::tr\_image< S, I, T > Struct Template Reference**

Transform an image by a given transformation.

```
#include <tr_image.hh>
```

Inherits `image_identity< I, S, tr_image< S, I, T > >`.

**Public Types**

- typedef I::value [lvalue](#)  
*Return type of read-write access.*
- typedef I::psite [psite](#)  
*Point\_Site associated type.*
- typedef I::value [rvalue](#)  
*Return type of read-only access.*
- typedef I::site [site](#)  
*Site associated type.*
- typedef `tr_image< S, tag::image_< I >, T >` [skeleton](#)  
*Skeleton.*
- typedef I::value [value](#)  
*Value associated type.*

**Public Member Functions**

- const S & [domain](#) () const  
*Return the domain morpher.*
- bool [has](#) (const vec\_t &v) const  
*Test if a pixel value is accessible at v.*
- bool [is\\_valid](#) () const

*Test if this image has been initialized.*

- `I::value operator() (const psite &p) const`  
*Read-only access of pixel value at point site  $p$ .*
- `void set\_tr (T &tr)`  
*Set the transformation.*
- `const T & tr () const`  
*Return the underlying transformation.*
- `tr\_image (const S &s, const I &ima, const T &tr)`  
*Constructors.*

### 10.355.1 Detailed Description

`template<typename S, typename I, typename T> struct mln::tr_image< S, I, T >`

Transform an image by a given transformation.

### 10.355.2 Member Typedef Documentation

**10.355.2.1** `template<typename S, typename I, typename T> typedef I ::value mln::tr_image< S, I, T >::lvalue`

Return type of read-write access.

**10.355.2.2** `template<typename S, typename I, typename T> typedef I ::psite mln::tr_image< S, I, T >::psite`

[Point\\_Site](#) associated type.

**10.355.2.3** `template<typename S, typename I, typename T> typedef I ::value mln::tr_image< S, I, T >::rvalue`

Return type of read-only access.

**10.355.2.4** `template<typename S, typename I, typename T> typedef I ::site mln::tr_image< S, I, T >::site`

[Site](#) associated type.

**10.355.2.5** `template<typename S, typename I, typename T> typedef tr_image< S, tag::image_<I>, T> mln::tr_image< S, I, T >::skeleton`

Skeleton.

**10.355.2.6** `template<typename S, typename I, typename T> typedef I ::value mln::tr_image< S, I, T >::value`

Value associated type.

### 10.355.3 Constructor & Destructor Documentation

**10.355.3.1** `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.

### 10.355.4 Member Function Documentation

**10.355.4.1** `template<typename S, typename I, typename T > const S & mln::tr_image< S, I, T >::domain( ) const [inline]`

Return the domain morpher.

**10.355.4.2** `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.

**10.355.4.3** `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.

**10.355.4.4** `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).

**10.355.4.5** `template<typename S, typename I, typename T > void mln::tr_image< S, I, T >::set_tr( T & tr ) [inline]`

Set the transformation.

**10.355.4.6** `template<typename S, typename I, typename T > const T & mln::tr_image< S, I, T >::tr( ) const [inline]`

Return the underlying transformation.

## 10.356 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](#) ()  
*Constructor without argument.*
- [transformed\\_image](#) (I &ima, const F &f)  
*Constructor.*

#### 10.356.1 Detailed Description

```
template<typename I, typename F> struct mln::transformed_image< I, F >
```

[Image](#) having its domain restricted by a site set.

#### 10.356.2 Member Typedef Documentation

- ##### 10.356.2.1 `template<typename I, typename F> typedef transformed_image< tag::image_<I>, tag::function_<F> > mln::transformed_image< I, F >::skeleton`

Skeleton.

### 10.356.3 Constructor & Destructor Documentation

**10.356.3.1** `template<typename I, typename F > mln::transformed_image< I, F >::transformed_image( ) [inline]`

Constructor without argument.

**10.356.3.2** `template<typename I, typename F > mln::transformed_image< I, F >::transformed_image( I & ima, const F & f ) [inline]`

Constructor.

### 10.356.4 Member Function Documentation

**10.356.4.1** `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.

**10.356.4.2** `template<typename I, typename F > mln::transformed_image< I, F >::operator transformed_image< const I, F >( ) const [inline]`

Const promotion via conversion.

**10.356.4.3** `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.

**10.356.4.4** `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.

## 10.357 mln::unproject\_image< I, D, F > Struct Template Reference

Un-projects an image.

```
#include <unproject_image.hh>
```

Inherits image\_domain\_morpher< I, D, unproject\_image< I, D, F > >.

### Public Member Functions

- const D & `domain`( ) const

*Give the definition domain.*

- `internal::morpher_lvalue_< I >::ret operator() (const typename D::psite &p)`  
*Read-write access to the image value located at point p.*
- `I::rvalue operator() (const typename D::psite &p) const`  
*Read-only access to the image value located at point p.*
- `unproject_image ()`  
*Constructor without argument.*
- `unproject_image (I &ima, const D &dom, const F &f)`  
*Constructor from an image ima, a domain dom, and a function f.*

### 10.357.1 Detailed Description

`template<typename I, typename D, typename F> struct mln::unproject_image< I, D, F >`

Un-projects an image.

### 10.357.2 Constructor & Destructor Documentation

**10.357.2.1** `template<typename I, typename D, typename F > mln::unproject_image< I, D, F >::unproject_image ( ) [inline]`

Constructor without argument.

**10.357.2.2** `template<typename I, typename D, typename F > mln::unproject_image< I, D, F >::unproject_image ( I & ima, const D & dom, const F & f ) [inline]`

Constructor from an image ima, a domain dom, and a function f.

### 10.357.3 Member Function Documentation

**10.357.3.1** `template<typename I, typename D, typename F > const D & mln::unproject_image< I, D, F >::domain ( ) const [inline]`

Give the definition domain.

**10.357.3.2** `template<typename I, typename D, typename F > internal::morpher_lvalue_< I >::ret mln::unproject_image< I, D, F >::operator() ( const typename D::psite & p ) [inline]`

Read-write access to the image value located at point p.

**10.357.3.3** `template<typename I, typename D, typename F > I::rvalue mln::unproject_image< I, D, F >::operator() ( const typename D::psite & p ) const [inline]`

Read-only access to the image value located at point p.



## 10.358 mln::util::adjacency\_matrix< V > Class Template Reference

A class of adjacency matrix.

```
#include <adjacency_matrix.hh>
```

Inherits adjacency\_matrix\_impl\_selector< V, mln::metal::equal< mln::trait::value\_< V >::quant, trait::value::quant::low >::eval >.

### Public Member Functions

- [adjacency\\_matrix](#) ()

*Constructors.*

- [adjacency\\_matrix](#) (const V &nelements)

*Construct an adjacency matrix with nelements elements maximum.*

### 10.358.1 Detailed Description

```
template<typename V = def::coord> class mln::util::adjacency_matrix< V >
```

A class of adjacency matrix. Support low and high quantification value types. In case of low quantification value type, it uses an [image2d](#) to store adjacency information. In case of high quantification value type, it uses a [util::set](#) to store the adjacency information.

### 10.358.2 Constructor & Destructor Documentation

**10.358.2.1** `template<typename V > mln::util::adjacency_matrix< V >::adjacency_matrix ( )`

Constructors.

```
@{
```

Default

**10.358.2.2** `template<typename V > mln::util::adjacency_matrix< V >::adjacency_matrix ( const V & nelements )`

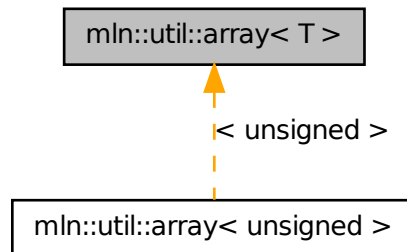
Construct an adjacency matrix with nelements elements maximum.

## 10.359 mln::util::array< T > Class Template Reference

A dynamic array class.

```
#include <array.hh>
```

Inheritance diagram for `mln::util::array< T >`:



## Public Types

- typedef `T` `element`  
*Element associated type.*
- typedef `T` `result`  
*Returned value types.*
- typedef `array_fwd_iter< T >` `fwd_eiter`  
*Iterator types*  
*Forward iterator associated type.*
- typedef `array_bkd_iter< T >` `bkd_eiter`  
*Backward iterator associated type.*
- typedef `fwd_eiter` `eiter`  
*Iterator associated type.*

## Public Member Functions

- `array< T >` & `append` (`const T` &`elt`)  
*Add the element `elt` at the end of this array.*
- `template<typename U >`  
`array< T >` & `append` (`const array< U >` &`other`)  
*Add the elements of `other` at the end of this array.*
- `void` `clear` ()  
*Empty the array.*

- void `fill` (const T &value)  
*Fill the whole array with value value.*
- bool `is_empty` () const  
*Test if the array is empty.*
- std::size\_t `memory_size` () const  
*Return the size of this array in memory.*
- unsigned `nelements` () const  
*Return the number of elements of the array.*
- ro\_result `operator()` (unsigned i) const  
*Return the i-th element of the array.*
- mutable\_result `operator()` (unsigned i)  
*Return the i-th element of the array.*
- ro\_result `operator[]` (unsigned i) const  
*Return the i-th element of the array.*
- mutable\_result `operator[]` (unsigned i)  
*Return the i-th element of the array.*
- void `reserve` (unsigned n)  
*Reserve memory for n elements.*
- void `resize` (unsigned n, 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` ()  
*Constructors*  
*Constructor without arguments.*
- `array` (unsigned n)  
*Construct a new array and resize it to elements.*
- `array` (unsigned n, const T &value)  
*Construct a new array, resize it to elements and fill it with default\_value.*

### 10.359.1 Detailed Description

**template<typename T> class mln::util::array< T >**

A dynamic array class. Elements are stored by copy. Implementation is lazy.

The parameter `T` is the element type, which shall not be const-qualified.

### 10.359.2 Member Typedef Documentation

**10.359.2.1 template<typename T> typedef array\_bkd\_iter<T> mln::util::array< T >::bkd\_eiter**

Backward iterator associated type.

**10.359.2.2 template<typename T> typedef fwd\_eiter mln::util::array< T >::eiter**

[Iterator](#) associated type.

**10.359.2.3 template<typename T> typedef T mln::util::array< T >::element**

Element associated type.

**10.359.2.4 template<typename T> typedef array\_fwd\_iter<T> mln::util::array< T >::fwd\_eiter**

[Iterator](#) types

Forward iterator associated type.

**10.359.2.5 template<typename T> typedef T mln::util::array< T >::result**

Returned value types.

Related to the [Function\\_v2v](#) concept.

### 10.359.3 Constructor & Destructor Documentation

**10.359.3.1 template<typename T> mln::util::array< T >::array ( ) [inline]**

Constructors

Constructor without arguments.

**10.359.3.2 template<typename T> mln::util::array< T >::array ( unsigned n ) [inline]**

Construct a new array and resize it to elements.

**10.359.3.3** `template<typename T> mln::util::array< T >::array ( unsigned n, const T & value ) [inline]`

Construct a new array, resize it to elements and fill it with `default_value`.

## 10.359.4 Member Function Documentation

**10.359.4.1** `template<typename T> array< T > & mln::util::array< T >::append ( const T & elt ) [inline]`

Add the element `elt` at the end of this array.  
Referenced by `mln::io::dicom::get_header()`, and `mln::io::plot::load()`.

**10.359.4.2** `template<typename T > template<typename U > array< T > & mln::util::array< T >::append ( const array< U > & other ) [inline]`

Add the elements of `other` at the end of this array.  
References `mln::util::array< T >::is_empty()`, and `mln::util::array< T >::std_vector()`.

**10.359.4.3** `template<typename T > void mln::util::array< T >::clear ( ) [inline]`

Empty the array.  
All elements contained in the array are destroyed.

### Postcondition

`is_empty() == true`

References `mln::util::array< T >::is_empty()`.  
Referenced by `mln::io::plot::load()`.

**10.359.4.4** `template<typename T> void mln::util::array< T >::fill ( const T & value ) [inline]`

Fill the whole array with value `value`.

**10.359.4.5** `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()`.

**10.359.4.6** `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()`.

**10.359.4.7** `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[]()`, and `mln::util::array< T >::size()`.

**10.359.4.8** `template<typename T> array< T >::ro_result mln::util::array< T >::operator() ( unsigned i ) const [inline]`

Return the `i`-th element of the array.

#### Precondition

`i < nelements()`

**10.359.4.9** `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()`

**10.359.4.10** `template<typename T> array< T >::ro_result mln::util::array< T >::operator[] ( unsigned i ) const [inline]`

Return the `i`-th element of the array.

#### Precondition

`i < nelements()`

References `mln::util::array< T >::nelements()`.

**10.359.4.11** `template<typename T> array< T >::mutable_result mln::util::array< T >::operator[] ( unsigned i ) [inline]`

Return the `i`-th element of the array.

**Precondition**

`i` < `nelements()`

References `mln::util::array< T >::nelements()`.

**10.359.4.12** `template<typename T > void mln::util::array< T >::reserve ( unsigned n )`  
**[inline]**

Reserve memory for `n` elements.

**10.359.4.13** `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.

**10.359.4.14** `template<typename T > void mln::util::array< T >::resize ( unsigned n )`  
**[inline]**

Resize this array to `n` elements.

Referenced by `mln::labeling::impl::generic::compute()`, `mln::labeling::impl::compute_fastest()`, `mln::io::raw::get_header()`, and `mln::io::dump::get_header()`.

**10.359.4.15** `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::labeling::impl::generic::compute()`, `mln::labeling::impl::compute_fastest()`, `mln::value::lut_vec< S, T >::lut_vec()`, and `mln::labeled_image_base< I, E >::update_data()`.

**10.359.4.16** `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==()`.

**10.360 mln::util::branch< T > Class Template Reference**

Class of generic branch.

```
#include <tree.hh>
```

## Public Member Functions

- `tree_node< T > & apex ()`  
*The getter of the apex.*
- `branch (tree< T > &tree, tree_node< T > &apex)`  
*Constructor.*
- `tree< T > & util_tree ()`  
*The getter of the tree.*

### 10.360.1 Detailed Description

`template<typename T> class mln::util::branch< T >`

Class of generic branch.

### 10.360.2 Constructor & Destructor Documentation

**10.360.2.1** `template<typename T > mln::util::branch< T >::branch ( util::tree< T > & tree, util::tree_node< T > & apex ) [inline]`

Constructor.

#### Parameters

- [in] *tree* The tree of the branch.
- [in] *apex* The apex of the branch.

### 10.360.3 Member Function Documentation

**10.360.3.1** `template<typename T > util::tree_node< T > & mln::util::branch< T >::apex ( ) [inline]`

The getter of the apex.

#### Returns

The `tree_node` appex of the current branch.

**10.360.3.2** `template<typename T > mln::util::tree< T > & mln::util::branch< T >::util_tree ( ) [inline]`

The getter of the tree.

#### Returns

The tree of the current branch.



## 10.361 mln::util::branch\_iter< T > Class Template Reference

Basic 2D image class.

```
#include <branch_iter.hh>
```

### Public Member Functions

- unsigned [deepness](#) () const  
*Give how deep is the iterator in the branch.*
- void [invalidate](#) ()  
*Invalidate the iterator.*
- bool [is\\_valid](#) () const  
*Test the iterator validity.*
- void [next](#) ()  
*Go to the next point.*
- [operator mln::util::tree\\_node< T > & \(\) const](#)  
*Conversion to node.*
- void [start](#) ()  
*Start an iteration.*

### 10.361.1 Detailed Description

**template<typename T> class mln::util::branch\_iter< T >**

Basic 2D image class. The parameter T is the type of node's data. [branch\\_iter](#) is used to pre-order walk a branch.

### 10.361.2 Member Function Documentation

**10.361.2.1 template<typename T > unsigned mln::util::branch\_iter< T >::deepness ( ) const**  
**[inline]**

Give how deep is the iterator in the branch.

References [mln::util::branch\\_iter< T >::is\\_valid\(\)](#), and [mln::util::tree\\_node< T >::parent\(\)](#).

**10.361.2.2 template<typename T > void mln::util::branch\_iter< T >::invalidate ( )**  
**[inline]**

Invalidate the iterator.

Referenced by [mln::util::branch\\_iter< T >::next\(\)](#).

**10.361.2.3** `template<typename T> bool mln::util::branch_iter< T >::is_valid ( ) const [inline]`

Test the iterator validity.

Referenced by `mln::util::branch_iter< T >::deepness()`.

**10.361.2.4** `template<typename T> void mln::util::branch_iter< T >::next ( ) [inline]`

Go to the next point.

References `mln::util::branch_iter< T >::invalidate()`.

**10.361.2.5** `template<typename T> mln::util::branch_iter< T >::operator util::tree_node< T > & ( ) const [inline]`

Conversion to node.

**10.361.2.6** `template<typename T> void mln::util::branch_iter< T >::start ( ) [inline]`

Start an iteration.

## 10.362 mln::util::branch\_iter\_ind< T > Class Template Reference

Basic 2D image class.

```
#include <branch_iter_ind.hh>
```

### Public Member Functions

- unsigned `deepness` () const  
*Give how deep is the iterator in the branch.*
- void `invalidate` ()  
*Invalidate the iterator.*
- bool `is_valid` () const  
*Test the iterator validity.*
- void `next` ()  
*Go to the next point.*
- `operator util::tree_node< T > & ()` const  
*Conversion to node.*
- void `start` ()  
*Start an iteration.*

### 10.362.1 Detailed Description

**template<typename T> class mln::util::branch\_iter\_ind< T >**

Basic 2D image class. The parameter T is the type of node's data. [branch\\_iter\\_ind](#) is used to pre-order walk a branch.

### 10.362.2 Member Function Documentation

**10.362.2.1 template<typename T > unsigned mln::util::branch\_iter\_ind< T >::deepness ( ) const [inline]**

Give how deep is the iterator in the branch.

References mln::util::branch\_iter\_ind< T >::is\_valid(), and mln::util::tree\_node< T >::parent().

**10.362.2.2 template<typename T > void mln::util::branch\_iter\_ind< T >::invalidate ( ) [inline]**

Invalidate the iterator.

Referenced by mln::util::branch\_iter\_ind< T >::next().

**10.362.2.3 template<typename T > bool mln::util::branch\_iter\_ind< T >::is\_valid ( ) const [inline]**

Test the iterator validity.

Referenced by mln::util::branch\_iter\_ind< T >::deepness().

**10.362.2.4 template<typename T > void mln::util::branch\_iter\_ind< T >::next ( ) [inline]**

Go to the next point.

References mln::util::branch\_iter\_ind< T >::invalidate().

**10.362.2.5 template<typename T > mln::util::branch\_iter\_ind< T >::operator util::tree\_node< T > & ( ) const [inline]**

Conversion to node.

**10.362.2.6 template<typename T > void mln::util::branch\_iter\_ind< T >::start ( ) [inline]**

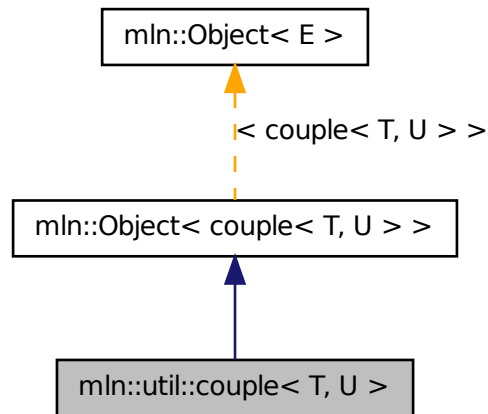
Start an iteration.

## 10.363 mln::util::couple< T, U > Class Template Reference

Definition of a couple.

```
#include <couple.hh>
```

Inheritance diagram for `mln::util::couple< T, U >`:



## Public Member Functions

- void `change_both` (const T &first, const U &second)  
*Replace both members of the couple by val.*
- void `change_first` (const T &val)  
*Replace the first member of the couple by val.*
- void `change_second` (const U &val)  
*Replace the second member of the couple by val.*
- const T & `first` () const  
*Get the first member of the couple.*
- const U & `second` () const  
*Get the second member of the couple.*

### 10.363.1 Detailed Description

```
template<typename T, typename U> class mln::util::couple< T, U >
```

Definition of a couple.

## 10.363.2 Member Function Documentation

**10.363.2.1** `template<typename T , typename U > void mln::util::couple< T, U >::change_both ( const T & first, const U & second ) [inline]`

Replace both members of the couple by *val*.

**10.363.2.2** `template<typename T , typename U > void mln::util::couple< T, U >::change_first ( const T & val ) [inline]`

Replace the first member of the couple by *val*.

**10.363.2.3** `template<typename T , typename U > void mln::util::couple< T, U >::change_second ( const U & val ) [inline]`

Replace the second member of the couple by *val*.

**10.363.2.4** `template<typename T , typename U > const T & mln::util::couple< T, U >::first ( ) const [inline]`

Get the first member of the couple.

**10.363.2.5** `template<typename T , typename U > const U & mln::util::couple< T, U >::second ( ) const [inline]`

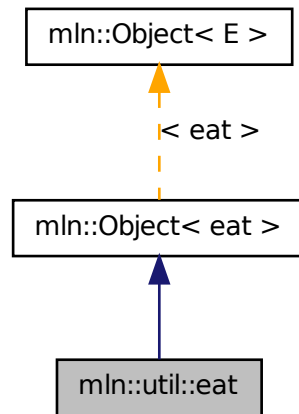
Get the second member of the couple.

## 10.364 mln::util::eat Struct Reference

Eat structure.

```
#include <eat.hh>
```

Inheritance diagram for mln::util::eat:



### 10.364.1 Detailed Description

Eat structure.

## 10.365 mln::util::edge< G > Class Template Reference

Edge of a graph G.

```
#include <edge.hh>
```

Inherits mln::util::internal::edge\_impl\_< G >.

### Public Types

- typedef [Edge](#)< void > [category](#)  
*Object category.*
- typedef G [graph\\_t](#)  
*Graph associated type.*
- typedef [edge\\_id\\_t](#) [id\\_t](#)  
*The edge type id.*
- typedef [edge\\_id\\_t::value\\_t](#) [id\\_value\\_t](#)  
*The underlying type used to store edge ids.*

## Public Member Functions

- [edge](#) ()  
*Constructors.*
- bool [is\\_valid](#) () const  
*Misc.*
- void [invalidate](#) ()  
*Invalidate that vertex.*
- [edge\\_id\\_t id](#) () const  
*Return the edge id.*
- void [update\\_id](#) (const [edge\\_id\\_t](#) &id)  
*Set id\_ with id;*
- operator [edge\\_id\\_t](#) () const  
*Conversion to the edge id.*
- const G & [graph](#) () const  
*Return a reference to the graph holding this edge.*
- void [change\\_graph](#) (const G &g)  
*Set g\_ with g;*
- [vertex\\_id\\_t v\\_other](#) (const [vertex\\_id\\_t](#) &id\_v) const  
*Vertex and edges oriented.*
- [vertex\\_id\\_t v1](#) () const  
*Edge oriented.*
- [vertex\\_id\\_t v2](#) () const  
*Return the highest vertex id adjacent to this edge.*
- [size\\_t nmax\\_nbh\\_edges](#) () const  
*Return the number max of adjacent edges.*
- [edge\\_id\\_t ith\\_nbh\\_edge](#) (unsigned i) const  
*Return the i th adjacent edge.*

### 10.365.1 Detailed Description

`template<typename G> class mln::util::edge< G >`

[Edge](#) of a graph G.

## 10.365.2 Member Typedef Documentation

### 10.365.2.1 `template<typename G> typedef Edge<void> mln::util::edge< G >::category`

[Object](#) category.

### 10.365.2.2 `template<typename G> typedef G mln::util::edge< G >::graph_t`

[Graph](#) associated type.

### 10.365.2.3 `template<typename G> typedef edge_id_t mln::util::edge< G >::id_t`

The edge type id.

### 10.365.2.4 `template<typename G> typedef edge_id_t::value_t mln::util::edge< G >::id_value_t`

The underlying type used to store edge ids.

## 10.365.3 Constructor & Destructor Documentation

### 10.365.3.1 `template<typename G > mln::util::edge< G >::edge ( ) [inline]`

Constructors.

References `mln::util::edge< G >::invalidate()`.

## 10.365.4 Member Function Documentation

### 10.365.4.1 `template<typename G > void mln::util::edge< G >::change_graph ( const G & g ) [inline]`

Set `g_` with `g`;

### 10.365.4.2 `template<typename G > const G & mln::util::edge< G >::graph ( ) const [inline]`

Return a reference to the graph holding this edge.

Referenced by `mln::p_edges< G, F >::has()`, and `mln::util::line_graph< G >::has()`.

### 10.365.4.3 `template<typename G > edge_id_t mln::util::edge< G >::id ( ) const [inline]`

Return the edge id.

Referenced by `mln::util::line_graph< G >::has()`.

### 10.365.4.4 `template<typename G > void mln::util::edge< G >::invalidate ( ) [inline]`

Invalidate that vertex.



Referenced by mln::util::edge< G >::edge().

**10.365.4.5** `template<typename G> bool mln::util::edge< G >::is_valid ( ) const [inline]`

Misc.

Return whether is points to a known edge.

Referenced by mln::p\_edges< G, F >::has().

**10.365.4.6** `template<typename G> edge_id_t mln::util::edge< G >::ith_nbh_edge ( unsigned i ) const [inline]`

Return the *i* th adjacent edge.

**10.365.4.7** `template<typename G> size_t mln::util::edge< G >::nmax_nbh_edges ( ) const [inline]`

Return the number max of adjacent edges.

**10.365.4.8** `template<typename G> mln::util::edge< G >::operator edge_id_t ( ) const [inline]`

Conversion to the edge id.

**10.365.4.9** `template<typename G> void mln::util::edge< G >::update_id ( const edge_id_t & id ) [inline]`

Set `id_` with `id`;

**10.365.4.10** `template<typename G> vertex_id_t mln::util::edge< G >::v1 ( ) const [inline]`

[Edge](#) oriented.

Return the lowest vertex id adjacent to this edge.

Referenced by mln::util::edge< G >::v\_other().

**10.365.4.11** `template<typename G> vertex_id_t mln::util::edge< G >::v2 ( ) const [inline]`

Return the highest vertex id adjacent to this edge.

Referenced by mln::util::edge< G >::v\_other().

**10.365.4.12** `template<typename G> vertex_id_t mln::util::edge< G >::v_other ( const vertex_id_t & id_v ) const [inline]`

[Vertex](#) and edges oriented.

Return the vertex id of this edge which is different from `id_v`.

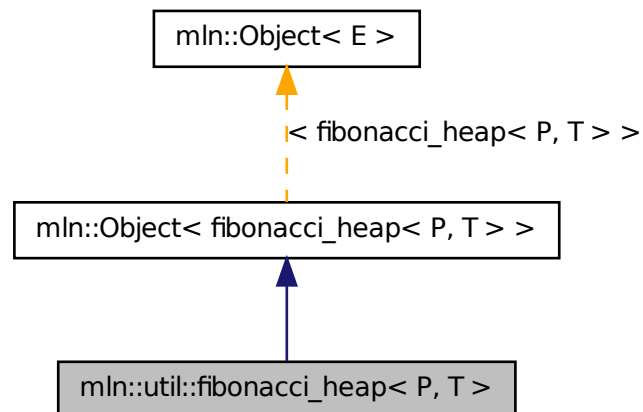
References mln::util::edge< G >::v1(), and mln::util::edge< G >::v2().

## 10.366 mln::util::fibonacci\_heap< P, T > Class Template Reference

Fibonacci heap.

```
#include <fibonacci_heap.hh>
```

Inheritance diagram for mln::util::fibonacci\_heap< P, T >:



### Public Member Functions

- void `clear` ()  
*Clear all elements in the heap and make the heap empty.*
- `fibonacci_heap` ()  
*Default constructor.*
- `fibonacci_heap` (const `fibonacci_heap`< P, T > &node)  
*Copy constructor Be ware that once this heap is constructed, the argument node is cleared and all its elements are part of this new heap.*
- const T & `front` () const  
*Return the minimum value in the heap.*
- bool `is_empty` () const  
*Is it empty?*
- bool `is_valid` () const  
*return false if it is empty.*
- unsigned `nelements` () const  
*Return the number of elements.*

- `fibonacci_heap< P, T > & operator= (fibonacci_heap< P, T > &rhs)`

*Assignment operator.*

- `T pop_front ()`

*Return and remove the minimum value in the heap.*

- `void push (const P &priority, const T &value)`

*Push a new element in the heap.*

- `void push (fibonacci_heap< P, T > &other_heap)`

*Take other\_heap's elements and insert them in this heap.*

### 10.366.1 Detailed Description

`template<typename P, typename T> class mln::util::fibonacci_heap< P, T >`

Fibonacci heap.

### 10.366.2 Constructor & Destructor Documentation

**10.366.2.1** `template<typename P , typename T > mln::util::fibonacci_heap< P, T >::fibonacci_heap ( ) [inline]`

Default constructor.

**10.366.2.2** `template<typename P , typename T > mln::util::fibonacci_heap< P, T >::fibonacci_heap ( const fibonacci_heap< P, T > & node ) [inline]`

Copy constructor Be ware that once this heap is constructed, the argument `node` is cleared and all its elements are part of this new heap.

### 10.366.3 Member Function Documentation

**10.366.3.1** `template<typename P , typename T > void mln::util::fibonacci_heap< P, T >::clear ( ) [inline]`

Clear all elements in the heap and make the heap empty.

References `mln::util::fibonacci_heap< P, T >::pop_front()`.

**10.366.3.2** `template<typename P , typename T > const T & mln::util::fibonacci_heap< P, T >::front ( ) const [inline]`

Return the minimum value in the heap.

**10.366.3.3** `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()`.

**10.366.3.4** `template<typename P , typename T > bool mln::util::fibonacci_heap< P, T >::is_valid ( ) const [inline]`

return false if it is empty.

Referenced by `mln::util::fibonacci_heap< P, T >::pop_front()`.

**10.366.3.5** `template<typename P , typename T > unsigned mln::util::fibonacci_heap< P, T >::nelements ( ) const [inline]`

Return the number of elements.

**10.366.3.6** `template<typename P , typename T > fibonacci_heap< P, T > & mln::util::fibonacci_heap< P, T >::operator= ( fibonacci_heap< P, T > & rhs ) [inline]`

Assignment operator.

Be ware that this operator do *\*not\** copy the data from *rhs* to this heap. It moves all elements which means that afterwards, *rhs* is is cleared and all its elements are part of this new heap.

**10.366.3.7** `template<typename P , typename T > T mln::util::fibonacci_heap< P, T >::pop_front ( ) [inline]`

Return and remove the minimum value in the heap.

References `mln::util::fibonacci_heap< P, T >::is_empty()`, `mln::util::fibonacci_heap< P, T >::is_valid()`, and `mln::util::fibonacci_heap< P, T >::push()`.

Referenced by `mln::util::fibonacci_heap< P, T >::clear()`.

**10.366.3.8** `template<typename P , typename T > void mln::util::fibonacci_heap< P, T >::push ( const P & priority, const T & value ) [inline]`

Push a new element in the heap.

**See also**

`insert`

Referenced by `mln::util::fibonacci_heap< P, T >::pop_front()`.

### 10.366.3.9 `template<typename P, typename T> void mln::util::fibonacci_heap< P, T >::push ( fibonacci_heap< P, T > & other_heap ) [inline]`

Take `other_heap`'s elements and insert them in this heap.

After this call `other_heap` is cleared.

References `mln::util::fibonacci_heap< P, T >::is_empty()`.

## 10.367 mln::util::graph Class Reference

Undirected graph.

```
#include <graph.hh>
```

Inherits `graph_base< graph >`.

### Public Types

- `typedef std::set< edge_data_t > edges_set_t`  
*A set to test the presence of a given edge.*
- `typedef std::vector< edge_data_t > edges_t`  
*The type of the set of edges.*
- `typedef std::vector< vertex_data_t > vertices_t`  
*The type of the set of vertices.*
- `typedef mln::internal::vertex_fwd_iterator< graph > vertex_fwd_iter`  
*Iterator types*  
*Vertex iterators.*
- `typedef mln::internal::vertex_nbh_edge_fwd_iterator< graph > vertex_nbh_edge_fwd_iter`  
*Vertex centered edge iterators.*
- `typedef mln::internal::vertex_nbh_vertex_fwd_iterator< graph > vertex_nbh_vertex_fwd_iter`  
*Vertex centered vertex iterators.*
- `typedef mln::internal::edge_fwd_iterator< graph > edge_fwd_iter`  
*Edge iterators.*
- `typedef mln::internal::edge_nbh_edge_fwd_iterator< graph > edge_nbh_edge_fwd_iter`  
*Edge centered edge iterators.*

## Public Member Functions

- [graph](#) ()
- [graph](#) (unsigned nvertices)  
*Construct a graph with `nvertices` vertices.*
- bool [has\\_v](#) (const [vertex\\_id\\_t](#) &id\_v) const  
*Check whether a vertex id `id_v` exists in the graph.*
- [edge\\_id\\_t v\\_ith\\_nbh\\_edge](#) (const [vertex\\_id\\_t](#) &id\_v, unsigned i) const  
*Returns the `i` th edge adjacent to the vertex `id_v`.*
- [vertex\\_id\\_t v\\_ith\\_nbh\\_vertex](#) (const [vertex\\_id\\_t](#) &id\_v, unsigned i) const  
*Returns the `i` th vertex adjacent to the vertex `id_v`.*
- [size\\_t v\\_nmax](#) () const  
*Return the number of vertices in the graph.*
- [size\\_t v\\_nmax\\_nbh\\_edges](#) (const [vertex\\_id\\_t](#) &id\_v) const  
*Return the number of adjacent edges of vertex `id_v`.*
- [size\\_t v\\_nmax\\_nbh\\_vertices](#) (const [vertex\\_id\\_t](#) &id\_v) const  
*Return the number of adjacent vertices of vertex `id_v`.*
  
- unsigned [add\\_vertex](#) ()  
*Vertex oriented.*
- [std::pair< vertex\\_id\\_t, vertex\\_id\\_t > add\\_vertices](#) (unsigned n)  
*Add `n` vertices to the graph.*
- [vertex\\_t vertex](#) ([vertex\\_id\\_t](#) id\_v) const  
*Return the vertex whose id is `v`.*
  
- [edge\\_id\\_t add\\_edge](#) (const [vertex\\_id\\_t](#) &id\_v1, const [vertex\\_id\\_t](#) &id\_v2)  
*Edge oriented.*
- [edge\\_t edge](#) (const [edge\\_id\\_t](#) &e) const  
*Return the edge whose id is `e`.*
- const [std::vector< util::ord\\_pair< vertex\\_id\\_t > > & edges](#) () const  
*Return the list of all edges.*
- [size\\_t e\\_nmax](#) () const  
*Return the number of edges in the graph.*
- bool [has\\_e](#) (const [edge\\_id\\_t](#) &id\_e) const  
*Return whether `id_e` is in the graph.*
- [edge\\_t edge](#) (const [vertex\\_t](#) &v1, const [vertex\\_t](#) &v2) const  
*Return the corresponding edge id if exists.*

- `vertex_id_t v1` (`const edge_id_t &id_e`) `const`  
Return the first vertex associated to the edge `id_e`.
- `vertex_id_t v2` (`const edge_id_t &id_e`) `const`  
Return the second vertex associated to edge `id_e`.
- `size_t e_nmax_nbh_edges` (`const edge_id_t &id_e`) `const`  
Return the number max of adjacent edge, given an edge `id_e`.
- `edge_id_t e_ith_nbh_edge` (`const edge_id_t &id_e`, unsigned `i`) `const`  
Return the `i` th edge adjacent to the edge `id_e`.
- `template<typename G2 >`  
`bool is_subgraph_of` (`const G2 &g`) `const`  
Return whether this graph is a subgraph Return true if `g` and `*this` have the same `graph_id`.

### 10.367.1 Detailed Description

Undirected graph.

### 10.367.2 Member Typedef Documentation

#### 10.367.2.1 `typedef mln::internal::edge_fwd_iterator<graph> mln::util::graph::edge_fwd_iter`

Edge iterators.

#### 10.367.2.2 `typedef mln::internal::edge_nbh_edge_fwd_iterator<graph>` `mln::util::graph::edge_nbh_edge_fwd_iter`

Edge centered edge iterators.

#### 10.367.2.3 `typedef std::set<edge_data_t> mln::util::graph::edges_set_t`

A set to test the presence of a given edge.

#### 10.367.2.4 `typedef std::vector<edge_data_t> mln::util::graph::edges_t`

The type of the set of edges.

#### 10.367.2.5 `typedef mln::internal::vertex_fwd_iterator<graph> mln::util::graph::vertex_fwd_iter`

Iterator types

Vertex iterators.

**10.367.2.6** `typedef mln::internal::vertex_nbh_edge_fwd_iterator<graph>  
mln::util::graph::vertex_nbh_edge_fwd_iter`

[Vertex](#) centered edge iterators.

**10.367.2.7** `typedef mln::internal::vertex_nbh_vertex_fwd_iterator<graph>  
mln::util::graph::vertex_nbh_vertex_fwd_iter`

[Vertex](#) centered vertex iterators.

**10.367.2.8** `typedef std::vector<vertex_data_t> mln::util::graph::vertices_t`

The type of the set of vertices.

### 10.367.3 Constructor & Destructor Documentation

**10.367.3.1** `mln::util::graph::graph ( ) [inline]`

Constructor.

**10.367.3.2** `mln::util::graph::graph ( unsigned nvertices ) [inline]`

Construct a graph with `nvertices` vertices.

### 10.367.4 Member Function Documentation

**10.367.4.1** `edge_id_t mln::util::graph::add_edge ( const vertex_id_t & id_v1, const vertex_id_t  
& id_v2 ) [inline]`

[Edge](#) oriented.

Add an edge.

#### Returns

The id of the new edge if it does not exist yet; otherwise, return `mln_max (unsigned)`.

References `edge()`, and `has_v()`.

Referenced by `mln::make::voronoi()`.

**10.367.4.2** `unsigned mln::util::graph::add_vertex ( ) [inline]`

[Vertex](#) oriented.

Shortcuts factoring the insertion of vertices and edges. Add a vertex.

#### Returns

The id of the new vertex.



References `v_nmax()`.

Referenced by `mln::make::voronoi()`.

**10.367.4.3** `std::pair< vertex_id_t, vertex_id_t > mln::util::graph::add_vertices ( unsigned n )`  
**[inline]**

Add `n` vertices to the graph.

#### Returns

A range of vertex ids.

References `v_nmax()`.

**10.367.4.4** `edge_id_t mln::util::graph::e_ith_nbh_edge ( const edge_id_t & id_e, unsigned i )`  
**const [inline]**

Return the `i` th edge adjacent to the edge `id_e`.

References `e_nmax()`, `e_nmax_nbh_edges()`, `has_e()`, `v1()`, `v2()`, `v_ith_nbh_edge()`, and `v_nmax_nbh_edges()`.

**10.367.4.5** `size_t mln::util::graph::e_nmax ( ) const [inline]`

Return the number of edges in the graph.

Referenced by `e_ith_nbh_edge()`, and `edge()`.

**10.367.4.6** `size_t mln::util::graph::e_nmax_nbh_edges ( const edge_id_t & id_e ) const`  
**[inline]**

Return the number max of adjacent edge, given an edge `id_e`.

References `has_e()`, `v1()`, `v2()`, and `v_nmax_nbh_edges()`.

Referenced by `e_ith_nbh_edge()`.

**10.367.4.7** `graph::edge_t mln::util::graph::edge ( const edge_id_t & e ) const [inline]`

Return the edge whose id is `e`.

References `e_nmax()`.

Referenced by `add_edge()`.

**10.367.4.8** `graph::edge_t mln::util::graph::edge ( const vertex_t & v1, const vertex_t & v2 )`  
**const [inline]**

Return the corresponding edge id if exists.

If it is not, returns an invalid edge.

References `has_v()`.

**10.367.4.9** `const std::vector< util::ord_pair< vertex_id_t > > & mln::util::graph::edges ( )`  
`const [inline]`

Return the list of all edges.

**10.367.4.10** `bool mln::util::graph::has_e ( const edge_id_t & id_e ) const [inline]`

Return whether `id_e` is in the graph.

Referenced by `e_ith_nbh_edge()`, `e_nmax_nbh_edges()`, `v1()`, and `v2()`.

**10.367.4.11** `bool mln::util::graph::has_v ( const vertex_id_t & id_v ) const [inline]`

Check whether a vertex id `id_v` exists in the graph.

Referenced by `add_edge()`, `edge()`, `v_ith_nbh_edge()`, `v_ith_nbh_vertex()`, `v_nmax_nbh_edges()`, `v_nmax_nbh_vertices()`, and `vertex()`.

**10.367.4.12** `template<typename G2 > bool mln::util::graph::is_subgraph_of ( const G2 & g )`  
`const [inline]`

Return whether this graph is a subgraph Return true if `g` and `*this` have the same `graph_id`.

**10.367.4.13** `vertex_id_t mln::util::graph::v1 ( const edge_id_t & id_e ) const [inline]`

Return the first vertex associated to the edge `id_e`.

References `has_e()`.

Referenced by `e_ith_nbh_edge()`, and `e_nmax_nbh_edges()`.

**10.367.4.14** `vertex_id_t mln::util::graph::v2 ( const edge_id_t & id_e ) const [inline]`

Return the second vertex associated to edge `id_e`.

References `has_e()`.

Referenced by `e_ith_nbh_edge()`, and `e_nmax_nbh_edges()`.

**10.367.4.15** `edge_id_t mln::util::graph::v_ith_nbh_edge ( const vertex_id_t & id_v, unsigned i )`  
`const [inline]`

Returns the `i` th edge adjacent to the vertex `id_v`.

References `has_v()`, and `v_nmax_nbh_edges()`.

Referenced by `e_ith_nbh_edge()`, and `v_ith_nbh_vertex()`.

**10.367.4.16** `vertex_id_t mln::util::graph::v_ith_nbh_vertex ( const vertex_id_t & id_v, unsigned i ) const [inline]`

Returns the `i` th vertex adjacent to the vertex `id_v`.

References `has_v()`, and `v_ith_nbh_edge()`.

**10.367.4.17** `size_t mln::util::graph::v_nmax ( ) const [inline]`

Return the number of vertices in the graph.

Referenced by `add_vertex()`, and `add_vertices()`.

**10.367.4.18** `size_t mln::util::graph::v_nmax_nbh_edges ( const vertex_id_t & id_v ) const [inline]`

Return the number of adjacent edges of vertex `id_v`.

References `has_v()`.

Referenced by `e_ith_nbh_edge()`, `e_nmax_nbh_edges()`, `v_ith_nbh_edge()`, and `v_nmax_nbh_vertices()`.

**10.367.4.19** `size_t mln::util::graph::v_nmax_nbh_vertices ( const vertex_id_t & id_v ) const [inline]`

Return the number of adjacent vertices of vertex `id_v`.

References `has_v()`, and `v_nmax_nbh_edges()`.

**10.367.4.20** `graph::vertex_t mln::util::graph::vertex ( vertex_id_t id_v ) const [inline]`

Return the vertex whose id is `v`.

References `has_v()`.

**10.368 mln::util::greater\_point< I > Class Template Reference**

A “greater than” functor comparing points w.r.t.

```
#include <greater_point.hh>
```

**Public Member Functions**

- `bool operator() (const point &x, const point &y)`

*Is x greater than y?*

**10.368.1 Detailed Description**

```
template<typename I> class mln::util::greater_point< I >
```

A “greater than” functor comparing points w.r.t. the values they refer to in an image.

This functor used in useful to implement ordered queues of points.

## 10.368.2 Member Function Documentation

**10.368.2.1** `template<typename I > bool mln::util::greater_point< I >::operator() ( const point & x, const point & y )`

Is *x* greater than *y*?

## 10.369 mln::util::greater\_psite< I > Class Template Reference

A “greater than” functor comparing psites w.r.t.

```
#include <greater_psite.hh>
```

### Public Member Functions

- `bool operator() (const psite &x, const psite &y)`

*Is x greater than y?*

### 10.369.1 Detailed Description

```
template<typename I> class mln::util::greater_psite< I >
```

A “greater than” functor comparing psites w.r.t. the values they refer to in an image.

This functor used in useful to implement ordered queues of psites.

## 10.369.2 Member Function Documentation

**10.369.2.1** `template<typename I > bool mln::util::greater_psite< I >::operator() ( const psite & x, const psite & y )`

Is *x* greater than *y*?

## 10.370 mln::util::head< T, R > Class Template Reference

Top structure of the soft heap.

```
#include <soft_heap.hh>
```

### 10.370.1 Detailed Description

```
template<typename T, typename R> class mln::util::head< T, R >
```

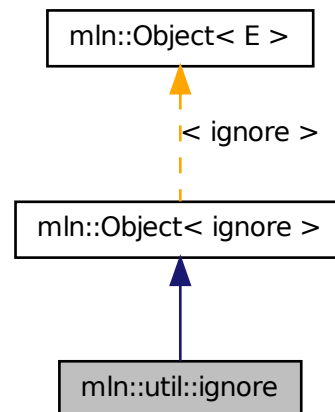
Top structure of the soft heap.

## 10.371 mln::util::ignore Struct Reference

Ignore structure.

```
#include <ignore.hh>
```

Inheritance diagram for mln::util::ignore:



### 10.371.1 Detailed Description

Ignore structure.

## 10.372 mln::util::ilcell< T > Struct Template Reference

Element of an item list. Store the data (key) used in [soft\\_heap](#).

```
#include <soft_heap.hh>
```

### 10.372.1 Detailed Description

```
template<typename T> struct mln::util::ilcell< T >
```

Element of an item list. Store the data (key) used in [soft\\_heap](#).

## 10.373 mln::util::line\_graph< G > Class Template Reference

Undirected line graph of a graph of type G.

```
#include <line_graph.hh>
```

Inherits `graph_base< line_graph< G > >`.

## Public Types

- typedef `std::vector< edge_data_t >` [edges\\_t](#)  
*The type of the set of edges.*
- typedef `std::vector< vertex_data_t >` [vertices\\_t](#)  
*The type of the set of vertices.*
- typedef `mln::internal::vertex_fwd_iterator< line_graph< G > >` [vertex\\_fwd\\_iter](#)  
*Iterator types*  
*Vertex iterators.*
- typedef `mln::internal::edge_fwd_iterator< line_graph< G > >` [edge\\_fwd\\_iter](#)  
*Edge iterators.*
- typedef `mln::internal::edge_nbh_edge_fwd_iterator< line_graph< G > >` [edge\\_nbh\\_edge\\_fwd\\_iter](#)  
*Edge nbh edge iterators.*
- typedef `mln::internal::vertex_nbh_vertex_fwd_iterator< line_graph< G > >` [vertex\\_nbh\\_vertex\\_fwd\\_iter](#)  
*Vertex nbh vertex iterators.*
- typedef `mln::internal::vertex_nbh_edge_fwd_iterator< line_graph< G > >` [vertex\\_nbh\\_edge\\_fwd\\_iter](#)  
*Vertex nbh edge iterators.*

## Public Member Functions

- `template<typename G2 >`  
`bool` [has](#) (`const util::vertex< G2 > &``v`) `const`  
*Check whether a vertex `v` exists in the line graph.*
- `bool` [has\\_v](#) (`const vertex_id_t &``id_v`) `const`  
*Check whether a vertex id `id_v` exists in the line graph.*
- `edge_id_t` [v\\_ith\\_nbh\\_edge](#) (`const vertex_id_t &``id_v`, `unsigned` `i`) `const`  
*Returns the `i` th edge adjacent to the vertex `id_v`.*
- `vertex_id_t` [v\\_ith\\_nbh\\_vertex](#) (`const vertex_id_t &``id_v`, `unsigned` `i`) `const`  
*Returns the `i` th vertex adjacent to the vertex `id_v`.*

- `size_t v_nmax () const`  
*Return the number of vertices in the graph.*
- `size_t v_nmax_nbh_edges (const vertex_id_t &id_v) const`  
*Return the number of adjacent edges of vertex `id_v`.*
- `size_t v_nmax_nbh_vertices (const vertex_id_t &id_v) const`  
*Return the number of adjacent vertices of vertex `id_v`.*
  
- `vertex_t vertex (const vertex_id_t &id_v) const`  
*Vertex oriented.*
  
- `edge_t edge (const edge_id_t &e) const`  
*Edge oriented.*
- `size_t e_nmax () const`  
*Return the number of edges in the graph.*
- `bool has_e (const util::edge_id_t &id_e) const`  
*Return whether `id_e` is in the line graph.*
- `template<typename G2 > bool has (const util::edge< G2 > &e) const`  
*Return whether `e` is in the line graph.*
- `vertex_id_t v1 (const edge_id_t &id_e) const`  
*Return the first vertex associated to the edge `id_e`.*
- `vertex_id_t v2 (const edge_id_t &id_e) const`  
*Return the second vertex associated to edge `id_e`.*
- `size_t e_nmax_nbh_edges (const edge_id_t &id_e) const`  
*Return the number max of adjacent edge, given an edge `id_e`.*
- `edge_id_t e_ith_nbh_edge (const edge_id_t &id_e, unsigned i) const`  
*Return the `i` th edge adjacent to the edge `id_e`.*
- `template<typename G2 > bool is_subgraph_of (const G2 &g) const`  
*Return whether this graph is a subgraph Return true if `g` and `*this` have the same `graph_id`.*
- `const G & graph () const`  
*Return the underlying graph.*

### 10.373.1 Detailed Description

`template<typename G> class mln::util::line_graph< G >`

Undirected line graph of a graph of type `G`.

## 10.373.2 Member Typedef Documentation

**10.373.2.1** `template<typename G> typedef mln::internal::edge_fwd_iterator< line_graph<G>  
> mln::util::line_graph< G >::edge_fwd_iter`

Edge iterators.

**10.373.2.2** `template<typename G> typedef mln::internal::edge_nbh_edge_fwd_iterator<  
line_graph<G> > mln::util::line_graph< G >::edge_nbh_edge_fwd_iter`

Edge nbh edge iterators.

**10.373.2.3** `template<typename G> typedef std::vector<edge_data_t> mln::util::line_graph< G  
>::edges_t`

The type of the set of edges.

**10.373.2.4** `template<typename G> typedef mln::internal::vertex_fwd_iterator< line_graph<G>  
> mln::util::line_graph< G >::vertex_fwd_iter`

Iterator types

Vertex iterators.

**10.373.2.5** `template<typename G> typedef mln::internal::vertex_nbh_edge_fwd_iterator<  
line_graph<G> > mln::util::line_graph< G >::vertex_nbh_edge_fwd_iter`

Vertex nbh edge iterators.

**10.373.2.6** `template<typename G> typedef mln::internal::vertex_nbh_vertex_fwd_iterator<  
line_graph<G> > mln::util::line_graph< G >::vertex_nbh_vertex_fwd_iter`

Vertex nbh vertex iterators.

**10.373.2.7** `template<typename G> typedef std::vector<vertex_data_t> mln::util::line_graph<  
G >::vertices_t`

The type of the set of vertices.

## 10.373.3 Member Function Documentation

**10.373.3.1** `template<typename G > edge_id_t mln::util::line_graph< G >::e_ith_nbh_edge (  
const edge_id_t & id_e, unsigned i ) const [inline]`

Return the *i* th edge adjacent to the edge *id\_e*.

References `mln::util::line_graph< G >::e_nmax()`, `mln::util::line_graph< G >::e_nmax_nbh_edges()`, `mln::util::line_graph< G >::has_e()`, `mln::util::line_graph< G >::v1()`, `mln::util::line_graph< G >::v2()`, `mln::util::line_graph< G >::v_ith_nbh_edge()`, and `mln::util::line_graph< G >::v_nmax_nbh_edges()`.



**10.373.3.2** `template<typename G> size_t mln::util::line_graph< G >::e_nmax ( ) const [inline]`

Return the number of edges in the graph.

Referenced by mln::util::line\_graph< G >::e\_ith\_nbh\_edge(), and mln::util::line\_graph< G >::edge().

**10.373.3.3** `template<typename G> size_t mln::util::line_graph< G >::e_nmax_nbh_edges ( const edge_id_t & id_e ) const [inline]`

Return the number max of adjacent edge, given an edge `id_e`.

References mln::util::line\_graph< G >::has\_e(), mln::util::line\_graph< G >::v1(), mln::util::line\_graph< G >::v2(), and mln::util::line\_graph< G >::v\_nmax\_nbh\_edges().

Referenced by mln::util::line\_graph< G >::e\_ith\_nbh\_edge().

**10.373.3.4** `template<typename G> line_graph< G >::edge_t mln::util::line_graph< G >::edge ( const edge_id_t & e ) const [inline]`

Edge oriented.

Return the edge whose id is `e`.

References mln::util::line\_graph< G >::e\_nmax().

**10.373.3.5** `template<typename G> const G & mln::util::line_graph< G >::graph ( ) const [inline]`

Return the underlying graph.

**10.373.3.6** `template<typename G> template<typename G2> bool mln::util::line_graph< G >::has ( const util::vertex< G2 > & v ) const [inline]`

Check whether a vertex `v` exists in the line graph.

References mln::util::vertex< G >::graph(), mln::util::line\_graph< G >::has\_v(), and mln::util::vertex< G >::id().

**10.373.3.7** `template<typename G> template<typename G2> bool mln::util::line_graph< G >::has ( const util::edge< G2 > & e ) const [inline]`

Return whether `e` is in the line graph.

References mln::util::edge< G >::graph(), mln::util::line\_graph< G >::has\_e(), and mln::util::edge< G >::id().

**10.373.3.8** `template<typename G> bool mln::util::line_graph< G >::has_e ( const util::edge_id_t & id_e ) const [inline]`

Return whether `id_e` is in the line graph.

Referenced by `mln::util::line_graph< G >::e_ith_nbh_edge()`, `mln::util::line_graph< G >::e_nmax_nbh_edges()`, `mln::util::line_graph< G >::has()`, `mln::util::line_graph< G >::v1()`, and `mln::util::line_graph< G >::v2()`.

**10.373.3.9** `template<typename G > bool mln::util::line_graph< G >::has_v ( const vertex_id_t & id_v ) const [inline]`

Check whether a vertex id `id_v` exists in the line graph.

Referenced by `mln::util::line_graph< G >::has()`, `mln::util::line_graph< G >::v_ith_nbh_edge()`, `mln::util::line_graph< G >::v_ith_nbh_vertex()`, `mln::util::line_graph< G >::v_nmax_nbh_edges()`, `mln::util::line_graph< G >::v_nmax_nbh_vertices()`, and `mln::util::line_graph< G >::vertex()`.

**10.373.3.10** `template<typename G > template<typename G2 > bool mln::util::line_graph< G >::is_subgraph_of ( const G2 & g ) const [inline]`

Return whether this graph is a subgraph Return true if `g` and `*this` have the same `graph_id`.

**10.373.3.11** `template<typename G > vertex_id_t mln::util::line_graph< G >::v1 ( const edge_id_t & id_e ) const [inline]`

Return the first vertex associated to the edge `id_e`.

References `mln::util::line_graph< G >::has_e()`.

Referenced by `mln::util::line_graph< G >::e_ith_nbh_edge()`, and `mln::util::line_graph< G >::e_nmax_nbh_edges()`.

**10.373.3.12** `template<typename G > vertex_id_t mln::util::line_graph< G >::v2 ( const edge_id_t & id_e ) const [inline]`

Return the second vertex associated to edge `id_e`.

References `mln::util::line_graph< G >::has_e()`.

Referenced by `mln::util::line_graph< G >::e_ith_nbh_edge()`, and `mln::util::line_graph< G >::e_nmax_nbh_edges()`.

**10.373.3.13** `template<typename G > edge_id_t mln::util::line_graph< G >::v_ith_nbh_edge ( const vertex_id_t & id_v, unsigned i ) const [inline]`

Returns the `i` th edge adjacent to the vertex `id_v`.

References `mln::util::line_graph< G >::has_v()`, `mln::util::line_graph< G >::v_nmax()`, and `mln::util::line_graph< G >::v_nmax_nbh_edges()`.

Referenced by `mln::util::line_graph< G >::e_ith_nbh_edge()`, and `mln::util::line_graph< G >::v_ith_nbh_vertex()`.

**10.373.3.14** `template<typename G > vertex_id_t mln::util::line_graph< G >::v_ith_nbh_vertex ( const vertex_id_t & id_v, unsigned i ) const [inline]`

Returns the `i` th vertex adjacent to the vertex `id_v`.

References mln::util::line\_graph< G >::has\_v(), and mln::util::line\_graph< G >::v\_ith\_nbh\_edge().

**10.373.3.15** `template<typename G > size_t mln::util::line_graph< G >::v_nmax ( ) const [inline]`

Return the number of vertices in the graph.

Referenced by mln::util::line\_graph< G >::v\_ith\_nbh\_edge().

**10.373.3.16** `template<typename G > size_t mln::util::line_graph< G >::v_nmax_nbh_edges ( const vertex_id_t & id_v ) const [inline]`

Return the number of adjacent edges of vertex `id_v`.

References mln::util::line\_graph< G >::has\_v().

Referenced by mln::util::line\_graph< G >::e\_ith\_nbh\_edge(), mln::util::line\_graph< G >::e\_nmax\_nbh\_edges(), mln::util::line\_graph< G >::v\_ith\_nbh\_edge(), and mln::util::line\_graph< G >::v\_nmax\_nbh\_vertices().

**10.373.3.17** `template<typename G > size_t mln::util::line_graph< G >::v_nmax_nbh_vertices ( const vertex_id_t & id_v ) const [inline]`

Return the number of adjacent vertices of vertex `id_v`.

References mln::util::line\_graph< G >::has\_v(), and mln::util::line\_graph< G >::v\_nmax\_nbh\_edges().

**10.373.3.18** `template<typename G > line_graph< G >::vertex_t mln::util::line_graph< G >::vertex ( const vertex_id_t & id_v ) const [inline]`

[Vertex](#) oriented.

Shortcuts factoring the insertion of vertices and edges.

Return the vertex whose id is `v`.

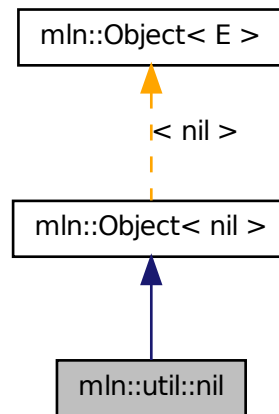
References mln::util::line\_graph< G >::has\_v().

## 10.374 mln::util::nil Struct Reference

Nil structure.

```
#include <nil.hh>
```

Inheritance diagram for mln::util::nil:



### 10.374.1 Detailed Description

Nil structure.

## 10.375 mln::util::node< T, R > Class Template Reference

Meta-data of an element in the heap.

```
#include <soft_heap.hh>
```

### 10.375.1 Detailed Description

```
template<typename T, typename R> class mln::util::node< T, R >
```

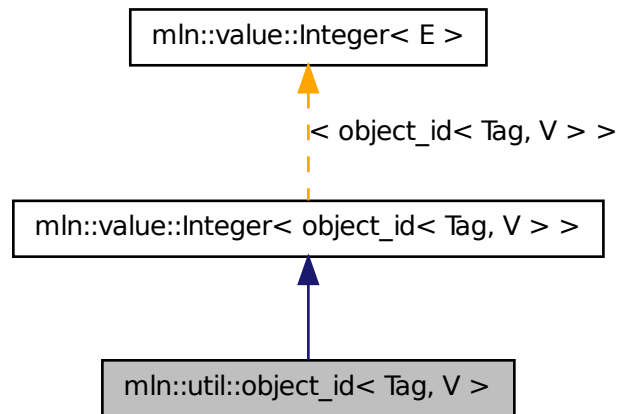
Meta-data of an element in the heap.

## 10.376 mln::util::object\_id< Tag, V > Class Template Reference

Base class of an object id.

```
#include <object_id.hh>
```

Inheritance diagram for mln::util::object\_id< Tag, V >:



## Public Types

- typedef V [value\\_t](#)

*The underlying type id.*

## Public Member Functions

- [object\\_id\(\)](#)

*Constructors.*

### 10.376.1 Detailed Description

```
template<typename Tag, typename V> class mln::util::object_id< Tag, V >
```

Base class of an object id.

#### Template Parameters

*Tag* the tag type

*Equiv* the equivalent value.

## 10.376.2 Member Typedef Documentation

**10.376.2.1** `template<typename Tag, typename V> typedef V mln::util::object_id< Tag, V >::value_t`

The underlying type id.

## 10.376.3 Constructor & Destructor Documentation

**10.376.3.1** `template<typename Tag , typename V > mln::util::object_id< Tag, V >::object_id ( ) [inline]`

Constructors.

## 10.377 mln::util::ord< T > Struct Template Reference

Function-object that defines an ordering between objects with type `T`: *lhs R rhs*.

```
#include <ord.hh>
```

### 10.377.1 Detailed Description

```
template<typename T> struct mln::util::ord< T >
```

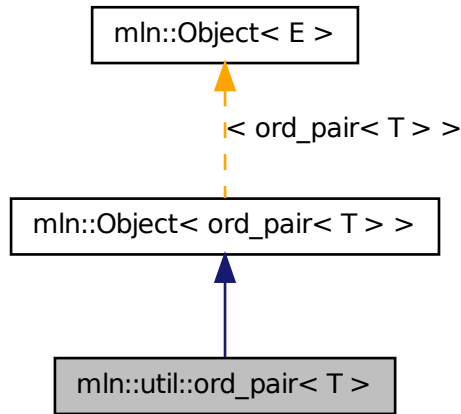
Function-object that defines an ordering between objects with type `T`: *lhs R rhs*. Its meaning is "lhs less-than rhs."

## 10.378 mln::util::ord\_pair< T > Struct Template Reference

Ordered pair structure s.a.

```
#include <ord_pair.hh>
```

Inheritance diagram for mln::util::ord\_pair< T >:



## Public Member Functions

- void [change\\_both](#) (const T &first, const T &second)  
*Replace both members of the pair by val, while keeping the relative order.*
- void [change\\_first](#) (const T &val)  
*Replace the first member of the pair by val, while keeping the relative order.*
- void [change\\_second](#) (const T &val)  
*Replace the second member of the pair by val, while keeping the relative order.*
- const T & [first](#) () const  
*Get the first (lowest) member of the pair.*
- const T & [second](#) () const  
*Get the second (highest) member of the pair.*

### 10.378.1 Detailed Description

```
template<typename T> struct mln::util::ord_pair< T >
```

Ordered pair structure s.a. this->first <= this->second; ordered pairs are partially ordered using lexicographical ordering.

## 10.378.2 Member Function Documentation

**10.378.2.1** `template<typename T> void mln::util::ord_pair< T >::change_both ( const T & first, const T & second ) [inline]`

Replace both members of the pair by *val*, while keeping the relative order.

### Postcondition

*first\_ <= second\_* (with *<=* being the [mln::util::ord\\_weak](#) relationship).

References `mln::util::ord_strict()`, and `mln::util::ord_weak()`.

**10.378.2.2** `template<typename T> void mln::util::ord_pair< T >::change_first ( const T & val ) [inline]`

Replace the first member of the pair by *val*, while keeping the relative order.

### Postcondition

*first\_ <= second\_* (with *<=* being the [mln::util::ord\\_weak](#) relationship).

References `mln::util::ord_strict()`, and `mln::util::ord_weak()`.

**10.378.2.3** `template<typename T> void mln::util::ord_pair< T >::change_second ( const T & val ) [inline]`

Replace the second member of the pair by *val*, while keeping the relative order.

### Postcondition

*first\_ <= second\_* (with *<=* being the [mln::util::ord\\_weak](#) relationship).

References `mln::util::ord_strict()`, and `mln::util::ord_weak()`.

**10.378.2.4** `template<typename T> const T & mln::util::ord_pair< T >::first ( ) const [inline]`

Get the first (lowest) member of the pair.

**10.378.2.5** `template<typename T> const T & mln::util::ord_pair< T >::second ( ) const [inline]`

Get the second (highest) member of the pair.

## 10.379 mln::util::pix< I > Struct Template Reference

Structure `pix`.

```
#include <pix.hh>
```



## Public Types

- typedef I::psite [psite](#)  
*Point\_Site associated type.*
- typedef I::value [value](#)  
*Value associated type.*

## Public Member Functions

- const I & [ima](#) () const  
*The getter of the image associate to pix structure.*
- const I::psite & [p](#) () const  
*The getter of psite associate to pix structure.*
- [pix](#) (const [Image](#)< I > &ima, const typename I::psite &p)  
*Constructor.*
- I::rvalue [v](#) () const  
*The getter of value associate to pix structure.*

### 10.379.1 Detailed Description

`template<typename I> struct mln::util::pix< I >`

Structure pix.

### 10.379.2 Member Typedef Documentation

**10.379.2.1** `template<typename I> typedef I::psite mln::util::pix< I >::psite`

[Point\\_Site](#) associated type.

**10.379.2.2** `template<typename I> typedef I::value mln::util::pix< I >::value`

[Value](#) associated type.

### 10.379.3 Constructor & Destructor Documentation

**10.379.3.1** `template<typename I> mln::util::pix< I >::pix ( const Image< I > & ima, const typename I::psite & p ) [inline]`

Constructor.

**Parameters**

[in] *ima* The image.

[in] *p* The p\_site.

**10.379.4 Member Function Documentation****10.379.4.1 `template<typename I> const I & mln::util::pix< I >::ima ( ) const [inline]`**

The getter of the image associate to pix structure.

**Returns**

The image ima\_.

**10.379.4.2 `template<typename I> const I::psite & mln::util::pix< I >::p ( ) const [inline]`**

The getter of psite associate to pix structure.

**Returns**

The psite p\_.

**10.379.4.3 `template<typename I> I::rvalue mln::util::pix< I >::v ( ) const [inline]`**

The getter of value associate to pix structure.

**Returns**

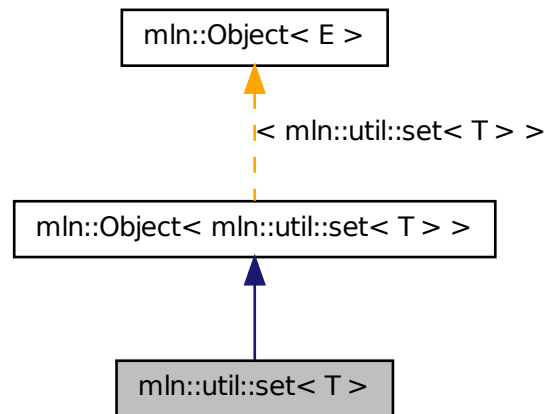
The value of pix.

**10.380 `mln::util::set< T >` Class Template Reference**

An "efficient" mathematical set class.

```
#include <set.hh>
```

Inheritance diagram for mln::util::set< T >:



## Public Types

- typedef set\_bkd\_iter< T > [bkd\\_eiter](#)  
*Backward iterator associated type.*
- typedef [fwd\\_eiter](#) eiter  
*Iterator associated type.*
- typedef T [element](#)  
*Element associated type.*
- typedef set\_fwd\_iter< T > [fwd\\_eiter](#)  
*Forward iterator associated type.*

## Public Member Functions

- void [clear](#) ()  
*Empty the set.*
- const T [first\\_element](#) () const  
*Return the first element of the set.*
- bool [has](#) (const T &elt) const  
*Test if the object elt belongs to the set.*
- set< T > & [insert](#) (const T &elt)

*Insert an element `elt` into the set.*

- `template<typename U >`  
`set< T > & insert (const set< U > &other)`

*Insert the elements of `other` into the set.*

- `bool is_empty () const`

*Test if the set is empty.*

- `const T last_element () const`

*Return the last element of the set.*

- `std::size_t memory_size () const`

*Return the size of this set in memory.*

- `unsigned nelements () const`

*Return the number of elements of the set.*

- `const T & operator[] (unsigned i) const`

*Return the *i*-th element of the set.*

- `set< T > & remove (const T &elt)`

*Remove an element `elt` into the set.*

- `set ()`

*Constructor without arguments.*

- `const std::vector< T > & std_vector () const`

*Give access to the set elements.*

### 10.380.1 Detailed Description

**template<typename T> class mln::util::set< T >**

An "efficient" mathematical set class. This set class is designed to store a mathematical set and to present it to the user as a linear array (`std::vector`).

Elements are stored by copy. Implementation is lazy.

The set has two states: frozen or not. There is an automatic switch of state when the user modifies its contents (insert, remove, or clear) or access to its contents (`op[i]`).

The parameter `T` is the element type, which shall not be const-qualified.

The unicity of set elements is handled by the `mln::util::ord` mechanism.

**See also**

[mln::util::ord](#)

## 10.380.2 Member Typedef Documentation

### 10.380.2.1 `template<typename T> typedef set_bkd_iter<T> mln::util::set< T >::bkd_iter`

Backward iterator associated type.

### 10.380.2.2 `template<typename T> typedef fwd_eiter mln::util::set< T >::eiter`

[Iterator](#) associated type.

### 10.380.2.3 `template<typename T> typedef T mln::util::set< T >::element`

Element associated type.

### 10.380.2.4 `template<typename T> typedef set_fwd_iter<T> mln::util::set< T >::fwd_eiter`

Forward iterator associated type.

## 10.380.3 Constructor & Destructor Documentation

### 10.380.3.1 `template<typename T > mln::util::set< T >::set ( ) [inline]`

Constructor without arguments.

## 10.380.4 Member Function Documentation

### 10.380.4.1 `template<typename T > void mln::util::set< T >::clear ( ) [inline]`

Empty the set.

All elements contained in the set are destroyed so the set is emptied.

#### Postcondition

`is_empty() == true`

References `mln::util::set< T >::is_empty()`.

### 10.380.4.2 `template<typename T > const T mln::util::set< T >::first_element ( ) const [inline]`

Return the first element of the set.

#### Precondition

not `is_empty()`

References `mln::util::set< T >::is_empty()`.

**10.380.4.3** `template<typename T> bool mln::util::set< T >::has ( const T & elt ) const`  
`[inline]`

Test if the object `elt` belongs to the set.

#### Parameters

[in] *elt* A possible element of the set.

#### Returns

True is `elt` is in the set.

**10.380.4.4** `template<typename T> set< T > & mln::util::set< T >::insert ( const T & elt )`  
`[inline]`

Insert an element `elt` into the set.

#### Parameters

[in] *elt* The element to be inserted.

If `elt` is already in the set, this method is a no-op.

#### Returns

The set itself after insertion.

Referenced by `mln::p_key< K, P >::change_keys()`.

**10.380.4.5** `template<typename T> template<typename U> set< T > & mln::util::set< T >::insert ( const set< U > & other ) [inline]`

Insert the elements of `other` into the set.

#### Parameters

[in] *other* The set containing the elements to be inserted.

#### Returns

The set itself after insertion.

References `mln::util::set< T >::is_empty()`, and `mln::util::set< T >::std_vector()`.

**10.380.4.6** `template<typename T> bool mln::util::set< T >::is_empty ( ) const [inline]`

Test if the set is empty.

References `mln::util::set< T >::nelements()`.

Referenced by `mln::util::set< T >::clear()`, `mln::util::set< T >::first_element()`, `mln::util::set< T >::insert()`, and `mln::util::set< T >::last_element()`.

**10.380.4.7** `template<typename T> const T mln::util::set< T >::last_element ( ) const`  
`[inline]`

Return the last element of the set.

**Precondition**

not `is_empty()`

References `mln::util::set< T >::is_empty()`.

**10.380.4.8** `template<typename T> std::size_t mln::util::set< T >::memory_size ( ) const`  
`[inline]`

Return the size of this set in memory.

References `mln::util::set< T >::nelements()`.

**10.380.4.9** `template<typename T> unsigned mln::util::set< T >::nelements ( ) const`  
`[inline]`

Return the number of elements of the set.

Referenced by `mln::util::set< T >::is_empty()`, `mln::util::set< T >::memory_size()`, and `mln::util::set< T >::operator[]()`.

**10.380.4.10** `template<typename T> const T & mln::util::set< T >::operator[] ( unsigned i )`  
`const [inline]`

Return the *i*-th element of the set.

**Parameters**

[in] *i* Index of the element to retrieve.

**Precondition**

*i* < `nelements()`

The element is returned by reference and is constant.

References `mln::util::set< T >::nelements()`.

**10.380.4.11** `template<typename T> set< T > & mln::util::set< T >::remove ( const T & elt )`  
`[inline]`

Remove an element `elt` into the set.

**Parameters**

[in] *elt* The element to be inserted.

If `elt` is already in the set, this method is a no-op.

**Returns**

The set itself after suppression.

**10.380.4.12** `template<typename T> const std::vector< T > & mln::util::set< T >::std_vector ( ) const [inline]`

Give access to the set elements.

The complexity of this method is O(1).

#### Postcondition

The set is frozen.

#### Returns

An array (std::vector) of elements.

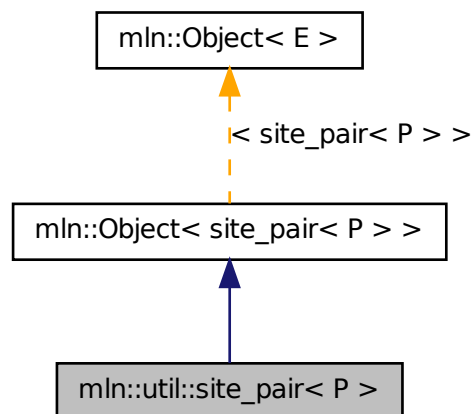
Referenced by `mln::util::set< T >::insert()`.

## 10.381 `mln::util::site_pair< P >` Class Template Reference

A pair of sites.

```
#include <site_pair.hh>
```

Inheritance diagram for `mln::util::site_pair< P >`:



### Public Member Functions

- `const P & first () const`  
*Return the first site.*
- `const util::ord_pair< P > & pair () const`  
*Return the underlying pair.*



- const P & [second](#) () const

*Return the second site.*

### 10.381.1 Detailed Description

```
template<typename P> class mln::util::site_pair< P >
```

A pair of sites. It can be used as site.

### 10.381.2 Member Function Documentation

**10.381.2.1** `template<typename P > const P & mln::util::site_pair< P >::first ( ) const`  
[inline]

Return the first site.

**10.381.2.2** `template<typename P > const util::ord_pair< P > & mln::util::site_pair< P >::pair ( ) const` [inline]

Return the underlying pair.

**10.381.2.3** `template<typename P > const P & mln::util::site_pair< P >::second ( ) const`  
[inline]

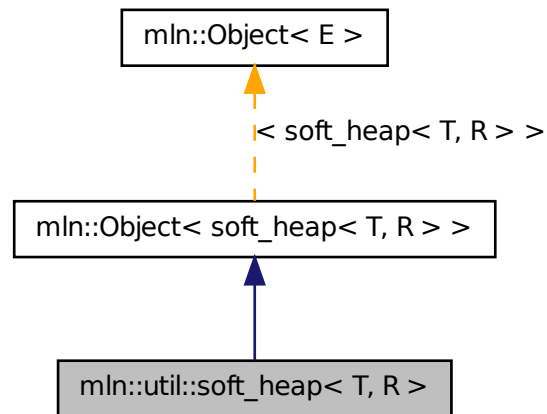
Return the second site.

## 10.382 mln::util::soft\_heap< T, R > Class Template Reference

Soft heap.

```
#include <soft_heap.hh>
```

Inheritance diagram for `mln::util::soft_heap< T, R >`:



## Public Types

- typedef `T element`  
*Element associated type.*

## Public Member Functions

- void `clear ()`  
*Clear the heap.*
- bool `is_empty () const`  
*Return true if there is at least one element.*
- bool `is_valid () const`  
*Return true if there is at least one element.*
- int `nelements () const`  
*Return the number of element in the heap.*
- `T pop_front ()`  
*Returns the element with the lowest priority and remove it from the heap.*
- void `push (soft_heap< T, R > &sh)`  
*Merge sh with this heap.*
- void `push (const T &element)`

*Add a new element element.*

- [soft\\_heap](#) (unsigned r=20)

*Default constructor.*

- [~soft\\_heap](#) ()

*Destructor.*

### 10.382.1 Detailed Description

`template<typename T, typename R> class mln::util::soft_heap< T, R >`

Soft heap. T key, the data to store in the heap. For instance a point 2d. R rank, for instance int\_u8

### 10.382.2 Member Typedef Documentation

**10.382.2.1** `template<typename T, typename R> typedef T mln::util::soft_heap< T, R >::element`

Element associated type.

### 10.382.3 Constructor & Destructor Documentation

**10.382.3.1** `template<typename T , typename R > mln::util::soft_heap< T, R >::soft_heap ( unsigned r = 20 ) [inline]`

Default constructor.

A corruption threshold  $r$  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.

**10.382.3.2** `template<typename T , typename R > mln::util::soft_heap< T, R >::~~soft_heap ( ) [inline]`

Destructor.

### 10.382.4 Member Function Documentation

**10.382.4.1** `template<typename T , typename R > void mln::util::soft_heap< T, R >::clear ( ) [inline]`

Clear the heap.

**10.382.4.2** `template<typename T , typename R > bool mln::util::soft_heap< T, R >::is_empty ( ) const [inline]`

Return true if there is at least one element.

**10.382.4.3** `template<typename T , typename R > bool mln::util::soft_heap< T, R >::is_valid ( )  
const [inline]`

Return true if there is at least one element.

Referenced by `mln::util::soft_heap< T, R >::pop_front()`.

**10.382.4.4** `template<typename T , typename R > int mln::util::soft_heap< T, R >::nelements ( )const [inline]`

Return the number of element in the heap.

Referenced by `mln::util::soft_heap< T, R >::push()`.

**10.382.4.5** `template<typename T , typename R > T mln::util::soft_heap< T, R >::pop_front ( )  
[inline]`

Returns the element with the lowest priority and remove it from the heap.

References `mln::util::soft_heap< T, R >::is_valid()`.

**10.382.4.6** `template<typename T , typename R > void mln::util::soft_heap< T, R >::push ( const  
T & element ) [inline]`

Add a new element `element`.

**10.382.4.7** `template<typename T , typename R > void mln::util::soft_heap< T, R >::push (   
soft_heap< T, R > & sh ) [inline]`

Merge `sh` with this heap.

Be ware that after this call, `sh` will be empty. This heap will hold the elements which were part of `sh`.

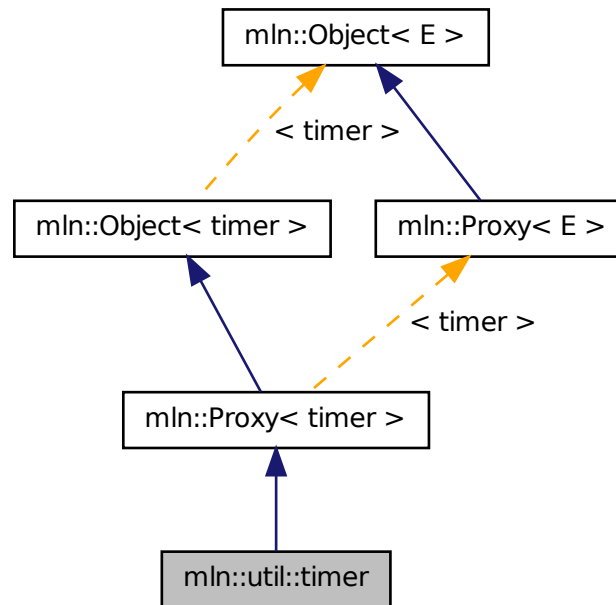
References `mln::util::soft_heap< T, R >::nelements()`.

## 10.383 mln::util::timer Class Reference

Timer structure.

```
#include <timer.hh>
```

Inheritance diagram for mln::util::timer:



### 10.383.1 Detailed Description

Timer structure.

## 10.384 mln::util::tracked\_ptr< T > Struct Template Reference

Smart pointer for shared data with tracking.

```
#include <tracked_ptr.hh>
```

### Public Member Functions

- `operator bool () const`  
*Coercion towards Boolean (for arithmetical tests).*
- `bool operator! () const`  
*Negation (for arithmetical tests).*
- `T * operator-> ()`  
*Mimics the behavior of op-> for a pointer in the mutable case.*

- `const T * operator-> () const`  
*Mimics the behavior of `op->` for a pointer in the `const` case.*
- `tracked_ptr< T > & operator= (T *ptr)`  
*Assignment.*
- `tracked_ptr< T > & operator= (const tracked_ptr< T > &rhs)`  
*Assignment.*
- `~tracked_ptr ()`  
*Destructor.*
- `tracked_ptr ()`  
*Constructors.*
- `tracked_ptr (const tracked_ptr< T > &rhs)`  
*Copy constructor.*

### 10.384.1 Detailed Description

`template<typename T> struct mln::util::tracked_ptr< T >`

Smart pointer for shared data with tracking.

### 10.384.2 Constructor & Destructor Documentation

**10.384.2.1** `template<typename T > mln::util::tracked_ptr< T >::tracked_ptr ( ) [inline]`

Constructors.

**10.384.2.2** `template<typename T > mln::util::tracked_ptr< T >::tracked_ptr ( const tracked_ptr< T > & rhs ) [inline]`

Copy constructor.

**10.384.2.3** `template<typename T > mln::util::tracked_ptr< T >::~~tracked_ptr ( ) [inline]`

Destructor.

### 10.384.3 Member Function Documentation

**10.384.3.1** `template<typename T > mln::util::tracked_ptr< T >::operator bool ( ) const [inline]`

Coercion towards Boolean (for arithmetical tests).

**10.384.3.2** `template<typename T > bool mln::util::tracked_ptr< T >::operator! ( ) const`  
`[inline]`

Negation (for arithmetical tests).

**10.384.3.3** `template<typename T > T * mln::util::tracked_ptr< T >::operator-> ( )`  
`[inline]`

Mimics the behavior of op-> for a pointer in the mutable case.

#### Invariant

Pointer proxy exists.

**10.384.3.4** `template<typename T > const T * mln::util::tracked_ptr< T >::operator-> ( ) const`  
`[inline]`

Mimics the behavior of op-> for a pointer in the const case.

#### Invariant

Pointer proxy exists.

**10.384.3.5** `template<typename T > tracked_ptr< T > & mln::util::tracked_ptr< T >::operator=`  
`( T * ptr ) [inline]`

Assignment.

**10.384.3.6** `template<typename T > tracked_ptr< T > & mln::util::tracked_ptr< T >::operator=`  
`( const tracked_ptr< T > & rhs ) [inline]`

Assignment.

## 10.385 mln::util::tree< T > Class Template Reference

Class of generic tree.

```
#include <tree.hh>
```

### Public Member Functions

- void [add\\_tree\\_down](#) (T &elt)  
*Bind a new tree downer the current.*
- void [add\\_tree\\_up](#) (T &elt)  
*Bind a new tree upper the current.*
- bool [check\\_consistency](#) ()

*Check the consistency of the tree.*

- [branch](#)< T > [main\\_branch](#) ()

*Convert the tree into brach.*

- [tree\\_node](#)< T > \* [root](#) ()

*The getter of the root.*

- [tree](#) ()

*Constructor.*

- [tree](#) ([tree\\_node](#)< T > \*[root](#))

*Constructor.*

### 10.385.1 Detailed Description

`template<typename T> class mln::util::tree< T >`

Class of generic tree.

### 10.385.2 Constructor & Destructor Documentation

**10.385.2.1** `template<typename T > mln::util::tree< T >::tree ( ) [inline]`

Constructor.

**10.385.2.2** `template<typename T > mln::util::tree< T >::tree ( tree\_node< T > * root ) [inline]`

Constructor.

#### Parameters

[in] *root* The root of the tree.

### 10.385.3 Member Function Documentation

**10.385.3.1** `template<typename T > void mln::util::tree< T >::add_tree_down ( T & elt ) [inline]`

Bind a new tree downer the current.

#### Parameters

[in] *elt* The new value of the new [tree\\_node](#) of the new tree add downer the current.



**10.385.3.2** `template<typename T > void mln::util::tree< T >::add_tree_up ( T & elt )`  
**[inline]**

Bind a new tree upper the current.

#### Parameters

[in] *elt* The new value of the new [tree\\_node](#) of the new tree add upper the current.

References `mln::util::tree_node< T >::children()`.

**10.385.3.3** `template<typename T > bool mln::util::tree< T >::check_consistency ( )`  
**[inline]**

Check the consistency of the tree.

#### Returns

true if no error, else false.

References `mln::util::tree< T >::root()`.

**10.385.3.4** `template<typename T > branch< T > mln::util::tree< T >::main_branch ( )`  
**[inline]**

Convert the tree into brach.

#### Returns

The root's [tree\\_node](#) of the the current tree.

References `mln::util::tree< T >::root()`.

**10.385.3.5** `template<typename T > tree_node< T > * mln::util::tree< T >::root ( )`  
**[inline]**

The getter of the root.

#### Returns

The root's [tree\\_node](#) of the the current tree.

Referenced by `mln::util::tree< T >::check_consistency()`, `mln::util::display_tree()`, `mln::util::tree< T >::main_branch()`, and `mln::util::tree_to_fast()`.

## 10.386 mln::util::tree\_node< T > Class Template Reference

Class of generic [tree\\_node](#) for tree.

```
#include <tree.hh>
```

## Public Member Functions

- `tree_node< T > * add_child (T elt)`  
*Create a `tree_node` with `elt` which become the child of the current `tree_node`.*
- `tree_node< T > * add_child (tree_node< T > *tree_node)`  
*Bind `tree_node` to the current `tree_node` and become its child.*
- `bool check_consistency ()`  
*Check the consistency of the `tree_node`.*
- `children_t & children ()`  
*The getter of the children.*
- `const children_t & children () const`  
*The getter of the children.*
- `tree_node< T > * delete_tree_node ()`  
*Delete the current `tree_node`.*
- `T & elt ()`  
*The getter of the element.*
- `const T & elt () const`  
*The const getter of the element.*
- `tree_node< T > * parent ()`  
*The getter of the parent.*
- `void print (std::ostream &ostr, int level=0)`  
*Print on `ostr` the arborescence with the current `tree_node` as root.*
- `tree_node< T > * search (T &elt)`  
*Search the `tree_node` with value `elt` in the arborescence of the current `tree_node`.*
- `int search_rec (tree_node< T > **res, T &elt)`  
*The using method for method search.*
- `void set_parent (tree_node< T > *parent)`  
*Bind `tree_node` to the current `tree_node` and become its parent.*
- `tree_node ()`  
*Constructor.*
- `tree_node (T elt)`  
*Constructor.*

## 10.386.1 Detailed Description

`template<typename T> class mln::util::tree_node< T >`

Class of generic [tree\\_node](#) for tree.

## 10.386.2 Constructor & Destructor Documentation

**10.386.2.1** `template<typename T > mln::util::tree_node< T >::tree_node ( ) [inline]`

Constructor.

**10.386.2.2** `template<typename T > mln::util::tree_node< T >::tree_node ( T elt ) [inline]`

Constructor.

### Parameters

[in] *elt* The element of [tree\\_node](#).

## 10.386.3 Member Function Documentation

**10.386.3.1** `template<typename T > tree_node< T > * mln::util::tree_node< T >::add_child ( T elt ) [inline]`

Create a [tree\\_node](#) with *elt* which become the child of the current [tree\\_node](#).

### Parameters

[in] *elt* The element of the new child to add.

### Returns

The new [tree\\_node](#) created.

**10.386.3.2** `template<typename T > tree_node< T > * mln::util::tree_node< T >::add_child ( tree_node< T > * tree_node ) [inline]`

Bind [tree\\_node](#) to the current [tree\\_node](#) and become its child.

### Parameters

[in] *tree\_node* The new child [tree\\_node](#).

### Returns

The child [tree\\_node](#).

References `mln::util::tree_node< T >::children()`, and `mln::util::tree_node< T >::parent()`.

**10.386.3.3** `template<typename T> bool mln::util::tree_node< T >::check_consistency ( )`  
`[inline]`

Check the consistency of the [tree\\_node](#).

#### Returns

true if no error, else false.

**10.386.3.4** `template<typename T> const std::vector< tree_node< T > * > &`  
`mln::util::tree_node< T >::children ( ) const [inline]`

The getter of the children.

#### Returns

The children of the [tree\\_node](#) in const.

**10.386.3.5** `template<typename T> std::vector< tree_node< T > * > & mln::util::tree_node< T`  
`>::children ( ) [inline]`

The getter of the children.

#### Returns

The children of the [tree\\_node](#).

Referenced by `mln::util::tree_node< T >::add_child()`, and `mln::util::tree< T >::add_tree_up()`.

**10.386.3.6** `template<typename T> tree_node< T > * mln::util::tree_node< T`  
`>::delete_tree_node ( ) [inline]`

Delete the current [tree\\_node](#).

**10.386.3.7** `template<typename T> const T & mln::util::tree_node< T >::elt ( ) const`  
`[inline]`

The const getter of the element.

#### Returns

The element of the [tree\\_node](#) in const.

**10.386.3.8** `template<typename T> T & mln::util::tree_node< T >::elt ( ) [inline]`

The getter of the element.

#### Returns

The element of the [tree\\_node](#).

Referenced by `mln::util::tree_node< T >::print()`.

**10.386.3.9** `template<typename T> tree_node< T > * mln::util::tree_node< T >::parent ( )`  
**[inline]**

The getter of the parent.

#### Returns

The parent of the [tree\\_node](#).

Referenced by `mln::util::tree_node< T >::add_child()`, `mln::util::branch_iter_ind< T >::deepness()`, and `mln::util::branch_iter< T >::deepness()`.

**10.386.3.10** `template<typename T> void mln::util::tree_node< T >::print ( std::ostream & ostr, int level = 0 )`  
**[inline]**

Print on `ostr` the arborescence with the current [tree\\_node](#) as root.

#### Parameters

[in] *ostr* The output stream.

[in] *level* The deep level

References `mln::util::tree_node< T >::elt()`.

**10.386.3.11** `template<typename T> tree_node< T > * mln::util::tree_node< T >::search ( T & elt )`  
**[inline]**

Search the [tree\\_node](#) with value `elt` in the arborescence of the current [tree\\_node](#).

#### Parameters

[in] *elt* The value of the searched [tree\\_node](#).

#### Returns

If not found 0 else the [tree\\_node](#) with `elt` value.

References `mln::util::tree_node< T >::search_rec()`.

**10.386.3.12** `template<typename T> int mln::util::tree_node< T >::search_rec ( tree_node< T > ** res, T & elt )`  
**[inline]**

The using method for method search.

Referenced by `mln::util::tree_node< T >::search()`.

**10.386.3.13** `template<typename T> void mln::util::tree_node< T >::set_parent ( tree_node< T > * parent )`  
**[inline]**

Bind [tree\\_node](#) to the current [tree\\_node](#) and become its parent.

#### Parameters

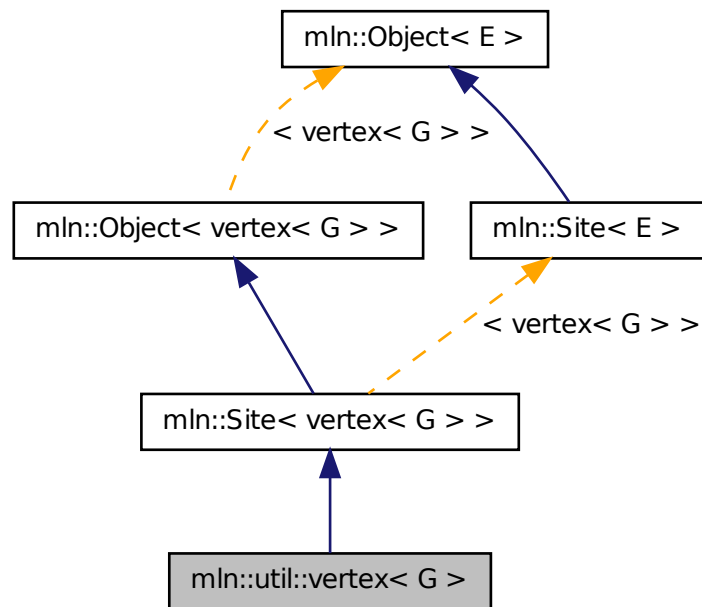
[in] *parent* The new parent [tree\\_node](#).

## 10.387 mln::util::vertex< G > Class Template Reference

Vertex of a graph G.

```
#include <vertex.hh>
```

Inheritance diagram for mln::util::vertex< G >:



### Public Types

- typedef `Vertex< void > Category`  
*Object category.*
- typedef `G graph_t`  
*Graph associated type.*
- typedef `vertex_id_t id_t`  
*The vertex type id.*
- typedef `vertex_id_t::value_t id_value_t`  
*The underlying type used to store vertex ids.*

## Public Member Functions

- void [change\\_graph](#) (const G &g)  
*Change the parent graph of that vertex.*
- [edge](#)< G > [edge\\_with](#) (const [vertex](#)< G > &v\_id) const  
*Returns true if this vertex has an edge with the given vertex.*
- const G & [graph](#) () const  
*Returns the graph pointer this vertex belongs to.*
- const [vertex\\_id\\_t](#) & [id](#) () const  
*Returns the vertex id.*
- void [invalidate](#) ()  
*Invalidate that vertex.*
- bool [is\\_valid](#) () const  
*Check whether the vertex is still part of the graph.*
- [edge\\_id\\_t](#) [ith\\_nbh\\_edge](#) (unsigned i) const  
*Returns the ith edge starting from this vertex.*
- [vertex\\_id\\_t](#) [ith\\_nbh\\_vertex](#) (unsigned i) const  
*Returns the ith vertex adjacent to this vertex.*
- unsigned [nmax\\_nbh\\_edges](#) () const  
*Returns the number max of edges starting from this vertex.*
- unsigned [nmax\\_nbh\\_vertices](#) () const  
*Returns the number max of vertices adjacent to this vertex.*
- operator [vertex\\_id\\_t](#) () const  
*Conversion to the vertex id.*
- [vertex\\_id\\_t](#) [other](#) (const [edge\\_id\\_t](#) &id\_e) const  
*Returns the other vertex located on edge id\_e.*
- void [update\\_id](#) (const [vertex\\_id\\_t](#) &id)  
*Update the vertex id.*
- [vertex](#) ()  
*Constructors.*

### 10.387.1 Detailed Description

`template<typename G> class mln::util::vertex< G >`

[Vertex](#) of a graph G.

## 10.387.2 Member Typedef Documentation

### 10.387.2.1 `template<typename G> typedef Vertex<void> mln::util::vertex< G >::Category`

Object category.

### 10.387.2.2 `template<typename G> typedef G mln::util::vertex< G >::graph_t`

Graph associated type.

### 10.387.2.3 `template<typename G> typedef vertex_id_t mln::util::vertex< G >::id_t`

The vertex type id.

### 10.387.2.4 `template<typename G> typedef vertex_id_t::value_t mln::util::vertex< G >::id_value_t`

The underlying type used to store vertex ids.

## 10.387.3 Constructor & Destructor Documentation

### 10.387.3.1 `template<typename G > mln::util::vertex< G >::vertex ( ) [inline]`

Constructors.

References `mln::util::vertex< G >::invalidate()`.

## 10.387.4 Member Function Documentation

### 10.387.4.1 `template<typename G > void mln::util::vertex< G >::change_graph ( const G & g ) [inline]`

Change the parent graph of that vertex.

### 10.387.4.2 `template<typename G > edge< G > mln::util::vertex< G >::edge_with ( const vertex< G > & v_id ) const [inline]`

Returns true if this vertex has an edge with the given vertex.

### 10.387.4.3 `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==( )`.



**10.387.4.4** `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==( ).

**10.387.4.5** `template<typename G > void mln::util::vertex< G >::invalidate ( ) [inline]`

Invalidate that vertex.

Referenced by mln::util::vertex< G >::vertex().

**10.387.4.6** `template<typename G > bool mln::util::vertex< G >::is_valid ( ) const [inline]`

Check whether the vertex is still part of the graph.

Referenced by mln::p\_vertices< G, F >::has().

**10.387.4.7** `template<typename G > edge_id_t mln::util::vertex< G >::ith_nbh_edge ( unsigned i ) const [inline]`

Returns the ith edge starting from this vertex.

**10.387.4.8** `template<typename G > vertex_id_t mln::util::vertex< G >::ith_nbh_vertex ( unsigned i ) const [inline]`

Returns the ith vertex adjacent to this vertex.

**10.387.4.9** `template<typename G > unsigned mln::util::vertex< G >::nmax_nbh_edges ( ) const [inline]`

Returns the number max of edges starting from this vertex.

If g\_ is a sub graph of another graph, nmax will be retrived from the initial graph.

**10.387.4.10** `template<typename G > unsigned mln::util::vertex< G >::nmax_nbh_vertices ( ) const [inline]`

Returns the number max of vertices adjacent to this vertex.

**10.387.4.11** `template<typename G > mln::util::vertex< G >::operator vertex_id_t ( ) const [inline]`

Conversion to the vertex id.

FIXME: May cause ambiguities... :(

**10.387.4.12** `template<typename G> vertex_id_t mln::util::vertex< G >::other ( const edge_id_t & id_e ) const [inline]`

Returns the other vertex located on edge `id_e`.

**10.387.4.13** `template<typename G> void mln::util::vertex< G >::update_id ( const vertex_id_t & id ) [inline]`

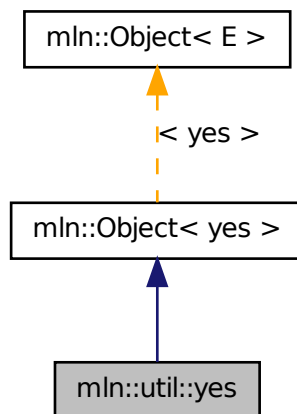
Update the vertex id.

## 10.388 mln::util::yes Struct Reference

[Object](#) that always says "yes".

```
#include <yes.hh>
```

Inheritance diagram for `mln::util::yes`:



### 10.388.1 Detailed Description

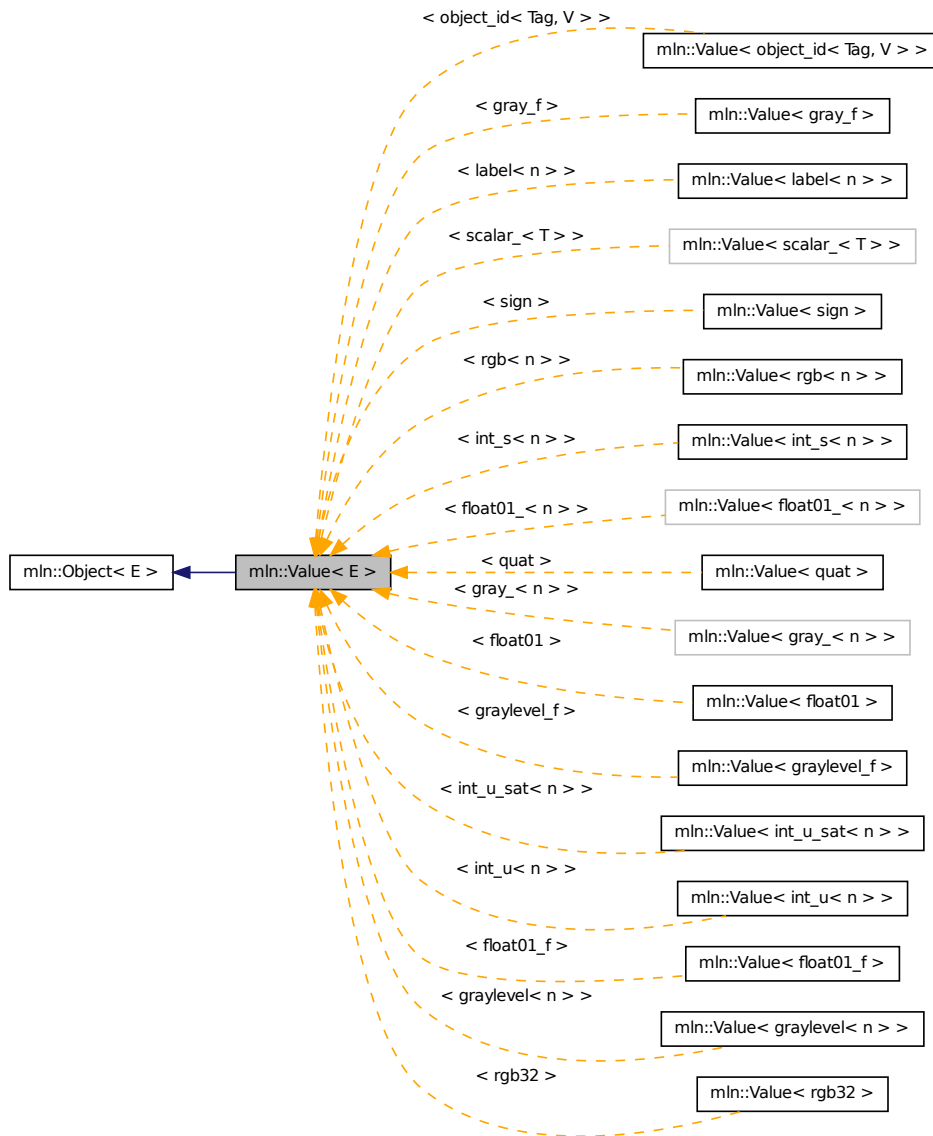
[Object](#) that always says "yes".

## 10.389 mln::Value< E > Struct Template Reference

Base class for implementation classes of values.

```
#include <value.hh>
```

Inheritance diagram for mln::Value< E >:



### 10.389.1 Detailed Description

`template<typename E> struct mln::Value< E >`

Base class for implementation classes of values.

**See also**

`mln::doc::Value` for a complete documentation of this class contents.

## 10.390 mln::value::float01 Class Reference

Class for floating values restricted to the interval [0..1] and discretized with n bits.

```
#include <float01.hh>
```

Inherits mln::value::Floating< float01 >.

### Public Types

- typedef std::pair< unsigned, unsigned long > [enc](#)  
*Encoding associated type.*
- typedef float [equiv](#)  
*Equivalent associated type.*

### Public Member Functions

- [float01](#) ()  
*Ctor.*
- template<unsigned n>  
[float01](#) (const float01\_< n > &val)  
*Ctor.*
- [float01](#) (unsigned nbits, float val)  
*Ctor.*
- unsigned [nbits](#) () const  
*Access to the encoding size.*
- operator float () const  
*Conversion to float.*
- [float01](#) & [set\\_nbits](#) (unsigned nbits)  
*Set the encoding size to nbits.*
- const [float01 to\\_nbits](#) (unsigned nbits) const  
*Return an equivalent gray encoded on nbits bits.*
- float [value](#) () const  
*Access to std type.*
- unsigned long [value\\_ind](#) () const  
*Access to the position in the quantized interval.*

#### 10.390.1 Detailed Description

Class for floating values restricted to the interval [0..1] and discretized with n bits.

## 10.390.2 Member Typedef Documentation

### 10.390.2.1 typedef std::pair<unsigned, unsigned long> mln::value::float01::enc

Encoding associated type.

### 10.390.2.2 typedef float mln::value::float01::equiv

Equivalent associated type.

## 10.390.3 Constructor & Destructor Documentation

### 10.390.3.1 mln::value::float01::float01 ( ) [inline]

Ctor.

### 10.390.3.2 template<unsigned n> mln::value::float01::float01 ( const float01\_<n> & val ) [inline]

Ctor.

### 10.390.3.3 mln::value::float01::float01 ( unsigned nbits, float val ) [inline]

Ctor.

## 10.390.4 Member Function Documentation

### 10.390.4.1 unsigned mln::value::float01::nbits ( ) const [inline]

Access to the encoding size.

### 10.390.4.2 mln::value::float01::operator float ( ) const [inline]

Conversion to float.

### 10.390.4.3 float01 & mln::value::float01::set\_nbits ( unsigned nbits ) [inline]

Set the encoding size to nbits.

Referenced by to\_nbits().

### 10.390.4.4 const float01 mln::value::float01::to\_nbits ( unsigned nbits ) const [inline]

Return an equivalent gray encoded on nbits bits.

References set\_nbits().

**10.390.4.5 float mln::value::float01::value ( ) const [inline]**

Access to std type.

**10.390.4.6 unsigned long mln::value::float01::value\_ind ( ) const [inline]**

Access to the position in the quantized interval.

**10.391 mln::value::float01\_f Struct Reference**

Class for floating values restricted to the interval [0..1].

```
#include <float01_f.hh>
```

Inherits mln::value::Floating< float01\_f >, and mln::value::internal::value\_like\_< float,float,float,float01\_f >.

**Public Member Functions**

- [float01\\_f](#) ()  
*Constructor without argument.*
- [float01\\_f](#) (float val)  
*Constructor from a float.*
- [operator float](#) () const  
*Conversion to a float.*
- [float01\\_f](#) & [operator=](#) (const float val)  
*Assignment from a float.*
- float [value](#) () const  
*Access to float value.*

**10.391.1 Detailed Description**

Class for floating values restricted to the interval [0..1].

**10.391.2 Constructor & Destructor Documentation****10.391.2.1 mln::value::float01\_f::float01\_f ( ) [inline]**

Constructor without argument.

**10.391.2.2 mln::value::float01\_f::float01\_f ( float val ) [inline]**

Constructor from a float.

### 10.391.3 Member Function Documentation

#### 10.391.3.1 mln::value::float01\_f::operator float ( ) const [inline]

Conversion to a float.

#### 10.391.3.2 float01\_f & mln::value::float01\_f::operator= ( const float val ) [inline]

Assignment from a float.

#### 10.391.3.3 float mln::value::float01\_f::value ( ) const [inline]

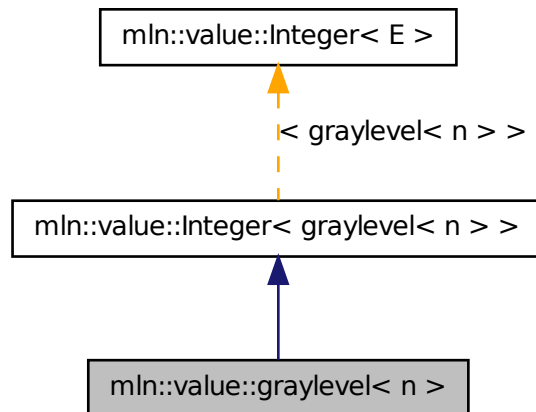
Access to float value.

## 10.392 mln::value::graylevel< n > Struct Template Reference

General gray-level class on n bits.

```
#include <graylevel.hh>
```

Inheritance diagram for mln::value::graylevel< n >:



### Public Member Functions

- [graylevel \(\)](#)  
*Constructor without argument.*
- [graylevel \(const graylevel< n > &rhs\)](#)  
*Copy constructor.*

- `graylevel` (int val)  
*Constructor from int.*
- `template<unsigned m>`  
`graylevel` (const `graylevel`< m > &rhs)  
*Constructor from any graylevel.*
- `graylevel`< n > & `operator=` (const `graylevel`< n > &rhs)  
*Assignment.*
- `graylevel`< n > & `operator=` (int val)  
*Assignment with int.*
- `template<unsigned m>`  
`graylevel`< n > & `operator=` (const `graylevel`< m > &rhs)  
*Assignment with any graylevel.*
- float `to_float` () const  
*Conversion to float between 0 and 1.*
- unsigned `value` () const  
*Access to std type.*
- `graylevel` (const `mln::literal::black_t` &)  
*Ctors with literals.*
- `graylevel`< n > & `operator=` (const `mln::literal::black_t` &)  
*Assignment with literals.*

### 10.392.1 Detailed Description

`template<unsigned n> struct mln::value::graylevel< n >`

General gray-level class on n bits.

### 10.392.2 Constructor & Destructor Documentation

**10.392.2.1** `template<unsigned n> mln::value::graylevel< n >::graylevel ( ) [inline]`

Constructor without argument.

**10.392.2.2** `template<unsigned n> mln::value::graylevel< n >::graylevel ( const graylevel< n > & rhs ) [inline]`

Copy constructor.



**10.392.2.3** `template<unsigned n> mln::value::graylevel< n >::graylevel ( int val ) [inline]`

Constructor from int.

**10.392.2.4** `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().

**10.392.2.5** `template<unsigned n> mln::value::graylevel< n >::graylevel ( const mln::literal::black_t & ) [inline]`

Ctors with literals.

**10.392.3 Member Function Documentation****10.392.3.1** `template<unsigned n> graylevel< n > & mln::value::graylevel< n >::operator= ( const graylevel< n > & rhs ) [inline]`

Assignment.

**10.392.3.2** `template<unsigned n> graylevel< n > & mln::value::graylevel< n >::operator= ( int val ) [inline]`

Assignment with int.

**10.392.3.3** `template<unsigned n> graylevel< n > & mln::value::graylevel< n >::operator= ( const mln::literal::black_t & ) [inline]`

Assignment with literals.

**10.392.3.4** `template<unsigned n> template<unsigned m> graylevel< n > & mln::value::graylevel< n >::operator= ( const graylevel< m > & rhs ) [inline]`

Assignment with any graylevel.

References mln::value::graylevel< n >::value().

**10.392.3.5** `template<unsigned n> float mln::value::graylevel< n >::to_float ( ) const [inline]`

Conversion to float between 0 and 1.

Referenced by mln::value::graylevel\_f::graylevel\_f(), and mln::value::graylevel\_f::operator=().

### 10.392.3.6 `template<unsigned n> unsigned mln::value::graylevel< n >::value ( ) const` `[inline]`

Access to std type.

Referenced by `mln::value::graylevel< n >::graylevel()`, and `mln::value::graylevel< n >::operator=()`.

## 10.393 `mln::value::graylevel_f` Struct Reference

General gray-level class on n bits.

```
#include <graylevel_f.hh>
```

Inherits `mln::value::Floating< graylevel_f >`, and `mln::value::internal::value_like_< float01_f,float01_f::enc, internal::gray_f,graylevel_f >`.

### Public Member Functions

- [graylevel\\_f \( \)](#)  
*Constructor without argument.*
- [graylevel\\_f \(const graylevel\\_f &rhs\)](#)  
*Copy constructor.*
- [graylevel\\_f \(float val\)](#)  
*Constructor from float.*
- `template<unsigned n>`  
[graylevel\\_f \(const graylevel< n > &rhs\)](#)  
*Constructor from graylevel.*
- `template<unsigned n>`  
[operator graylevel< n > \( \) const](#)  
*Conversion to graylevel<n>.*
- [graylevel\\_f & operator= \(float val\)](#)  
*Assignment with float.*
- [graylevel\\_f & operator= \(const graylevel\\_f &rhs\)](#)  
*Assignment.*
- `template<unsigned n>`  
[graylevel\\_f & operator= \(const graylevel< n > &rhs\)](#)  
*Assignment with graylevel.*
- `float value ( ) const`  
*Access to std type.*
- [graylevel\\_f \(const mln::literal::black\\_t &\)](#)

*Ctors with literals.*

- `graylevel_f & operator=` (const `mln::literal::black_t &`)

*Assignment with literals.*

### 10.393.1 Detailed Description

General gray-level class on n bits.

### 10.393.2 Constructor & Destructor Documentation

#### 10.393.2.1 `mln::value::graylevel_f::graylevel_f( )` [`inline`]

Constructor without argument.

#### 10.393.2.2 `mln::value::graylevel_f::graylevel_f( const graylevel_f & rhs )` [`inline`]

Copy constructor.

#### 10.393.2.3 `mln::value::graylevel_f::graylevel_f( float val )` [`inline`]

Constructor from float.

#### 10.393.2.4 `template<unsigned n> mln::value::graylevel_f::graylevel_f( const graylevel< n > & rhs )`

Constructor from graylevel.

References `mln::value::graylevel< n >::to_float()`.

#### 10.393.2.5 `mln::value::graylevel_f::graylevel_f( const mln::literal::black_t & )` [`inline`]

Ctors with literals.

### 10.393.3 Member Function Documentation

#### 10.393.3.1 `template<unsigned n> mln::value::graylevel_f::operator graylevel< n >( ) const` [`inline`]

Conversion to `graylevel<n>`.

#### 10.393.3.2 `graylevel_f & mln::value::graylevel_f::operator=( float val )` [`inline`]

Assignment with float.

**10.393.3.3** `template<unsigned n> graylevel_f & mln::value::graylevel_f::operator= ( const graylevel< n > & rhs )`

Assignment with graylevel.

References `mln::value::graylevel< n >::to_float()`.

**10.393.3.4** `graylevel_f & mln::value::graylevel_f::operator= ( const mln::literal::black_t & )`  
**[inline]**

Assignment with literals.

**10.393.3.5** `graylevel_f & mln::value::graylevel_f::operator= ( const graylevel_f & rhs )`  
**[inline]**

Assignment.

**10.393.3.6** `float mln::value::graylevel_f::value ( ) const` **[inline]**

Access to std type.

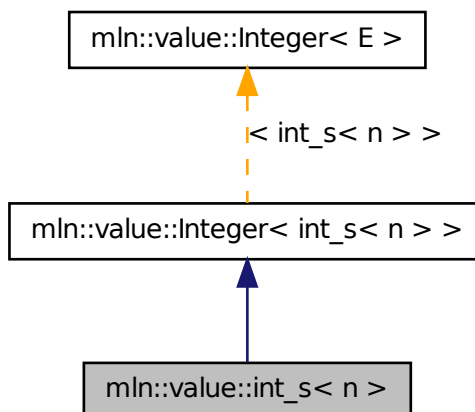
Referenced by `mln::value::operator<<()`.

## 10.394 mln::value::int\_s< n > Struct Template Reference

Signed integer value class.

```
#include <int_s.hh>
```

Inheritance diagram for `mln::value::int_s< n >`:



## Public Member Functions

- [int\\_s](#) ()  
*Constructor without argument.*
- [int\\_s](#) (int i)  
*Constructor from an integer.*
- [operator int](#) () const  
*Conversion to an integer.*
- [int\\_s](#)< n > & [operator=](#) (int i)  
*Assignment from an integer.*
  
- [int\\_s](#) (const [mln::literal::zero\\_t](#) &)  
*Constructors/assignments with literals.*

## Static Public Attributes

- static const [int\\_s](#)< n > [one](#) = 1  
*Unit value.*
- static const [int\\_s](#)< n > [zero](#) = 0  
*Zero value.*

### 10.394.1 Detailed Description

`template<unsigned n> struct mln::value::int_s< n >`

Signed integer value class. The parameter is n the number of encoding bits.

### 10.394.2 Constructor & Destructor Documentation

**10.394.2.1** `template<unsigned n> mln::value::int_s< n >::int_s( ) [inline]`

Constructor without argument.

**10.394.2.2** `template<unsigned n> mln::value::int_s< n >::int_s( int i ) [inline]`

Constructor from an integer.

**10.394.2.3** `template<unsigned n> mln::value::int_s< n >::int_s( const mln::literal::zero_t & ) [inline]`

Constructors/assignments with literals.

### 10.394.3 Member Function Documentation

**10.394.3.1** `template<unsigned n> mln::value::int_s< n >::operator int ( ) const [inline]`

Conversion to an integer.

**10.394.3.2** `template<unsigned n> int_s< n > & mln::value::int_s< n >::operator= ( int i ) [inline]`

Assignment from an integer.

### 10.394.4 Member Data Documentation

**10.394.4.1** `template<unsigned n> const int_s< n > mln::value::int_s< n >::one = 1 [static]`

Unit value.

**10.394.4.2** `template<unsigned n> const int_s< n > mln::value::int_s< n >::zero = 0 [static]`

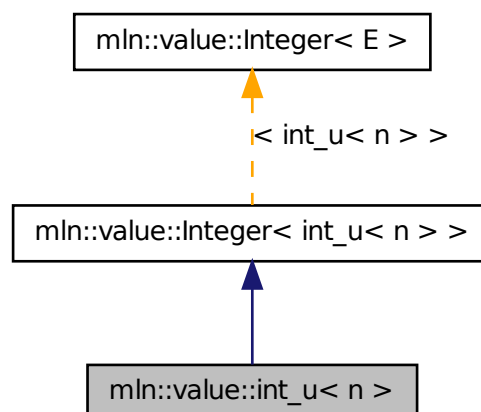
Zero value.

## 10.395 mln::value::int\_u< n > Struct Template Reference

Unsigned integer value class.

```
#include <int_u.hh>
```

Inheritance diagram for mln::value::int\_u< n >:



## Public Member Functions

- `int_u ()`  
*Constructor without argument.*
- `int_u (int i)`  
*Constructor from an integer.*
- `int_u< n > next () const`  
*Give the next value (i.e.,  $i + 1$ ).*
- `operator unsigned () const`  
*Conversion to an unsigned integer.*
- `int operator- () const`  
*Unary operator minus.*
- `int_u< n > & operator= (int i)`  
*Assignment from an integer.*
  
- `int_u (const mln::literal::zero_t &)`  
*Constructors/assignments with literals.*

### 10.395.1 Detailed Description

`template<unsigned n> struct mln::value::int_u< n >`

Unsigned integer value class. The parameter is `n` the number of encoding bits.

### 10.395.2 Constructor & Destructor Documentation

**10.395.2.1** `template<unsigned n> mln::value::int_u< n >::int_u ( ) [inline]`

Constructor without argument.

**10.395.2.2** `template<unsigned n> mln::value::int_u< n >::int_u ( int i ) [inline]`

Constructor from an integer.

**10.395.2.3** `template<unsigned n> mln::value::int_u< n >::int_u ( const mln::literal::zero_t & ) [inline]`

Constructors/assignments with literals.

### 10.395.3 Member Function Documentation

**10.395.3.1** `template<unsigned n> int_u< n > mln::value::int_u< n >::next ( ) const`  
`[inline]`

Give the next value (i.e.,  $i + 1$ ).

**10.395.3.2** `template<unsigned n> mln::value::int_u< n >::operator unsigned ( ) const`  
`[inline]`

Conversion to an unsigned integer.

**10.395.3.3** `template<unsigned n> int mln::value::int_u< n >::operator- ( ) const` `[inline]`

Unary operator minus.

**10.395.3.4** `template<unsigned n> int_u< n > & mln::value::int_u< n >::operator= ( int i )`  
`[inline]`

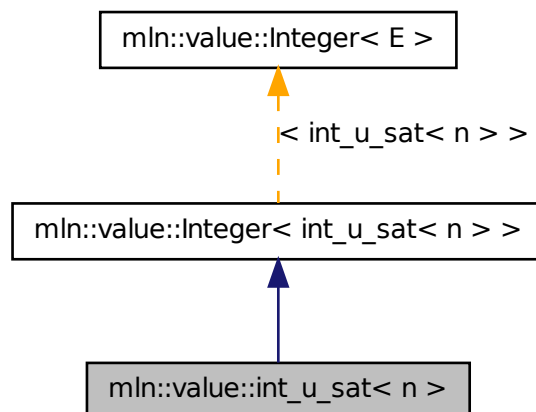
Assignment from an integer.

## 10.396 mln::value::int\_u\_sat< n > Struct Template Reference

Unsigned integer value class with saturation behavior.

```
#include <int_u_sat.hh>
```

Inheritance diagram for `mln::value::int_u_sat< n >`:





## Public Member Functions

- [int\\_u\\_sat \(\)](#)  
*Constructor without argument.*
- [int\\_u\\_sat \(int i\)](#)  
*Constructor from an integer.*
- [operator int \(\) const](#)  
*Conversion to an integer.*
- [int\\_u\\_sat< n > & operator+= \(int i\)](#)  
*Self addition.*
- [int\\_u\\_sat< n > & operator-= \(int i\)](#)  
*Self subtraction.*
- [int\\_u\\_sat< n > & operator= \(int i\)](#)  
*Assignment from an integer.*

## Static Public Attributes

- static const [int\\_u\\_sat< n > one = 1](#)  
*Unit value.*
- static const [int\\_u\\_sat< n > zero = 0](#)  
*Zero value.*

### 10.396.1 Detailed Description

`template<unsigned n> struct mln::value::int_u_sat< n >`

Unsigned integer value class with saturation behavior. The parameter is n the number of encoding bits.

### 10.396.2 Constructor & Destructor Documentation

**10.396.2.1** `template<unsigned n> mln::value::int_u_sat< n >::int_u_sat ( ) [inline]`

Constructor without argument.

**10.396.2.2** `template<unsigned n> mln::value::int_u_sat< n >::int_u_sat ( int i ) [inline]`

Constructor from an integer.

### 10.396.3 Member Function Documentation

**10.396.3.1** `template<unsigned n> mln::value::int_u_sat< n >::operator int ( ) const`  
[inline]

Conversion to an integer.

**10.396.3.2** `template<unsigned n> int_u_sat< n > & mln::value::int_u_sat< n >::operator+= (`  
`int i ) [inline]`

Self addition.

**10.396.3.3** `template<unsigned n> int_u_sat< n > & mln::value::int_u_sat< n >::operator-= (`  
`int i ) [inline]`

Self subtraction.

**10.396.3.4** `template<unsigned n> int_u_sat< n > & mln::value::int_u_sat< n >::operator= ( int`  
`i ) [inline]`

Assignment from an integer.

### 10.396.4 Member Data Documentation

**10.396.4.1** `template<unsigned n> const int_u_sat< n > mln::value::int_u_sat< n >::one = 1`  
[static]

Unit value.

**10.396.4.2** `template<unsigned n> const int_u_sat< n > mln::value::int_u_sat< n >::zero = 0`  
[static]

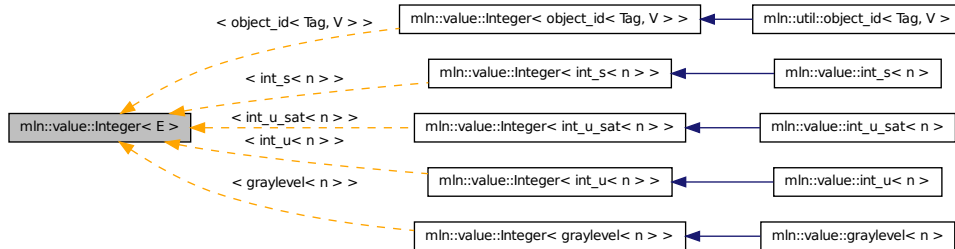
Zero value.

## 10.397 mln::value::Integer< E > Struct Template Reference

Concept of integer.

```
#include <integer.hh>
```

Inheritance diagram for mln::value::Integer< E >:



### 10.397.1 Detailed Description

```
template<typename E> struct mln::value::Integer< E >
```

Concept of integer.

## 10.398 mln::value::Integer< void > Struct Template Reference

Category flag type.

```
#include <integer.hh>
```

### 10.398.1 Detailed Description

```
template<> struct mln::value::Integer< void >
```

Category flag type.

## 10.399 mln::value::label< n > Struct Template Reference

Label value class.

```
#include <label.hh>
```

Inherits mln::value::Symbolic< label< n > >, and mln::value::internal::value\_like\_< unsigned,internal::encoding\_unsigned\_< n >::ret,int,label< n > >.

### Public Types

- typedef internal::encoding\_unsigned\_< n >::ret **enc**

*Encoding associated type.*

## Public Member Functions

- [label](#) ()  
*Constructor without argument.*
- [label](#) (unsigned i)  
*Constructor from an (unsigned) integer.*
- [label](#) (const [literal::zero\\_t](#) &v)  
*Constructor from [literal::zero](#).*
- [label](#)< n > [next](#) () const  
*Return the next value.*
- [operator unsigned](#) () const  
*Conversion to an unsigned integer.*
- [label](#)< n > & [operator++](#) ()  
*Self increment.*
- [label](#)< n > & [operator--](#) ()  
*Self decrement.*
- [label](#)< n > & [operator=](#) (unsigned i)  
*Assignment from an (unsigned) integer.*
- [label](#)< n > & [operator=](#) (const [literal::zero\\_t](#) &v)  
*Assignment from [literal::zero](#).*
- [label](#)< n > [prev](#) () const  
*Return the previous value.*

### 10.399.1 Detailed Description

```
template<unsigned n> struct mln::value::label< n >
```

Label value class. The parameter n is the number of encoding bits.

### 10.399.2 Member Typedef Documentation

**10.399.2.1** `template<unsigned n> typedef internal::encoding_unsigned_<n>::ret mln::value::label< n >::enc`

Encoding associated type.

### 10.399.3 Constructor & Destructor Documentation

**10.399.3.1** `template<unsigned n> mln::value::label< n >::label ( ) [inline]`

Constructor without argument.

**10.399.3.2** `template<unsigned n> mln::value::label< n >::label ( unsigned i ) [inline]`

Constructor from an (unsigned) integer.

**10.399.3.3** `template<unsigned n> mln::value::label< n >::label ( const literal::zero_t & v ) [inline]`

Constructor from [literal::zero](#).

### 10.399.4 Member Function Documentation

**10.399.4.1** `template<unsigned n> label< n > mln::value::label< n >::next ( ) const [inline]`

Return the next value.

**10.399.4.2** `template<unsigned n> mln::value::label< n >::operator unsigned ( ) const [inline]`

Conversion to an unsigned integer.

**10.399.4.3** `template<unsigned n> label< n > & mln::value::label< n >::operator++ ( ) [inline]`

Self increment.

**10.399.4.4** `template<unsigned n> label< n > & mln::value::label< n >::operator-- ( ) [inline]`

Self decrement.

**10.399.4.5** `template<unsigned n> label< n > & mln::value::label< n >::operator= ( unsigned i ) [inline]`

Assignment from an (unsigned) integer.

**10.399.4.6** `template<unsigned n> label< n > & mln::value::label< n >::operator= ( const literal::zero_t & v ) [inline]`

Assignment from [literal::zero](#).

**10.399.4.7** `template<unsigned n> label< n > mln::value::label< n >::prev ( ) const`  
`[inline]`

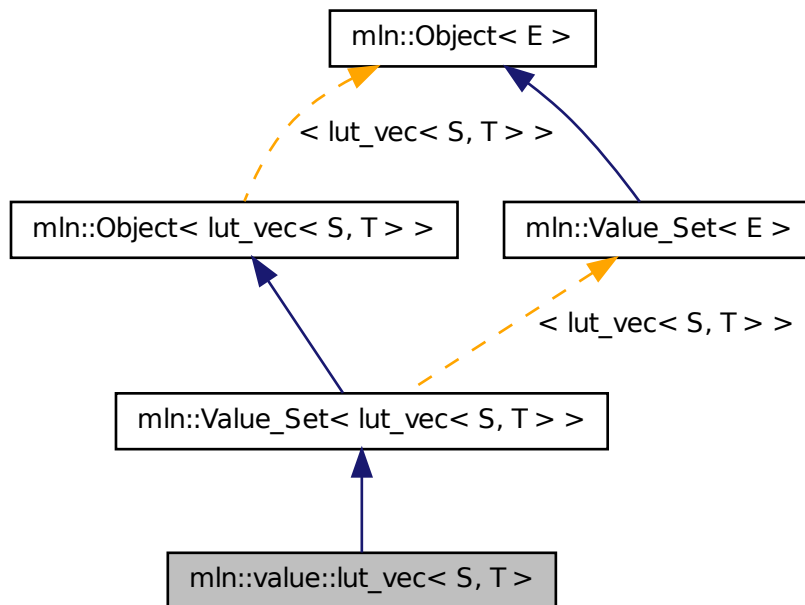
Return the previous value.

## 10.400 `mln::value::lut_vec< S, T >` Struct Template Reference

Class that defines FIXME.

```
#include <lut_vec.hh>
```

Inheritance diagram for `mln::value::lut_vec< S, T >`:



### Public Types

- `typedef bkd_viter_< lut_vec< S, T > > bkd_viter`  
*Backward Value\_Iterator associated type.*
- `typedef fwd_viter_< lut_vec< S, T > > fwd_viter`  
*Forward Value\_Iterator associated type.*
- `typedef T value`  
*Value associated type.*

## Public Member Functions

- bool `has` (const `value` &`v`) const  
*Test if `v` belongs to this set.*
- unsigned `index_of` (const `value` &`v`) const  
*Give the index of value `v` in this set.*
- unsigned `nvalues` () const  
*Give the number of values.*
- T `operator[]` (unsigned `i`) const  
*Give the `i`-th value.*
- template<typename F >  
`lut_vec` (const S &`vset`, const `Function_v2v`< F > &`f`)  
*Constructors*  
*Constructor from a value set and any `Function_v2v`.*
- template<typename V >  
`lut_vec` (const S &`vset`, const `Function_v2v`< fun::i2v::array< V > > &`f`)  
*Constructor from a value set and any `fun::i2v::array`.*
- template<typename V >  
`lut_vec` (const S &`vset`, const `Function_v2v`< util::array< V > > &`f`)  
*Constructor from a value set and any `util::array`.*

### 10.400.1 Detailed Description

`template<typename S, typename T> struct mln::value::lut_vec< S, T >`

Class that defines FIXME.

#### Warning

This is a multi-set!!! FIXME

### 10.400.2 Member Typedef Documentation

**10.400.2.1** `template<typename S , typename T > typedef bkd_viter_< lut_vec<S,T> > mln::value::lut_vec< S, T >::bkd_viter`

Backward `Value_Iterator` associated type.

**10.400.2.2** `template<typename S , typename T > typedef fwd_viter_< lut_vec<S,T> > mln::value::lut_vec< S, T >::fwd_viter`

Forward `Value_Iterator` associated type.

**10.400.2.3** `template<typename S, typename T> typedef T mln::value::lut_vec< S, T >::value`

[Value](#) associated type.

### 10.400.3 Constructor & Destructor Documentation

**10.400.3.1** `template<typename S, typename T> template<typename F> mln::value::lut_vec< S, T >::lut_vec( const S & vset, const Function_v2v< F > & f ) [inline]`

Constructors

Constructor from a value set and any [Function\\_v2v](#).

**10.400.3.2** `template<typename S, typename T> template<typename V> mln::value::lut_vec< S, T >::lut_vec( const S & vset, const Function_v2v< fun::i2v::array< V > > & f ) [inline]`

Constructor from a value set and any [fun::i2v::array](#).

**10.400.3.3** `template<typename S, typename T> template<typename V> mln::value::lut_vec< S, T >::lut_vec( const S & vset, const Function_v2v< util::array< V > > & f ) [inline]`

Constructor from a value set and any [util::array](#).

References `mln::util::array< T >::size()`, and `mln::util::array< T >::std_vector()`.

### 10.400.4 Member Function Documentation

**10.400.4.1** `template<typename S, typename T> bool mln::value::lut_vec< S, T >::has( const value & v ) const`

Test if `v` belongs to this set.

**10.400.4.2** `template<typename S, typename T> unsigned mln::value::lut_vec< S, T >::index_of( const value & v ) const`

Give the index of value `v` in this set.

**10.400.4.3** `template<typename S, typename T> unsigned mln::value::lut_vec< S, T >::nvalues( ) const [inline]`

Give the number of values.

Referenced by `mln::value::lut_vec< S, T >::operator[]()`.

**10.400.4.4** `template<typename S, typename T> T mln::value::lut_vec< S, T >::operator[]( unsigned i ) const [inline]`

Give the `i`-th value.



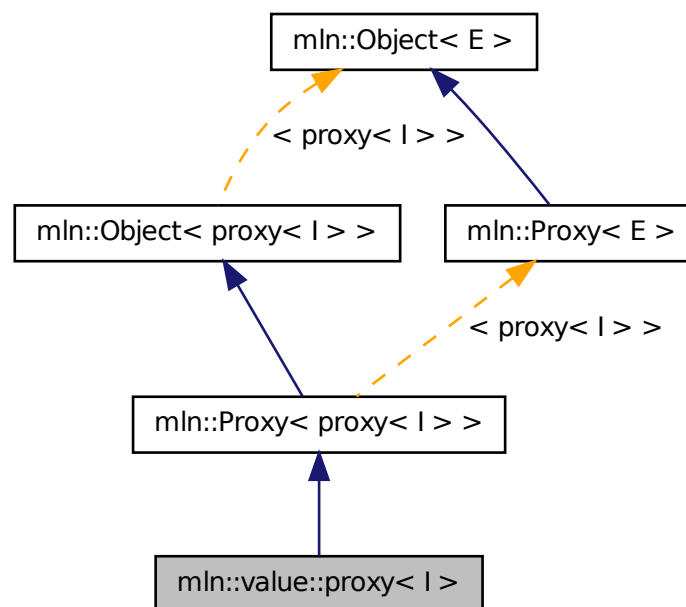
References mln::value::lut\_vec< S, T >::nvalues().

## 10.401 mln::value::proxy< I > Class Template Reference

Generic proxy class for an image pixel value.

```
#include <proxy.hh>
```

Inheritance diagram for mln::value::proxy< I >:



### Public Types

- typedef void [enc](#)  
*Encoding associated type.*
- typedef I::value [equiv](#)  
*Equivalent associated type.*

### Public Member Functions

- [proxy< I > & operator=](#) (const [proxy< I > &rhs](#))  
*Assignment (write access); replacement for default op.*

- `template<typename J >`  
`proxy< I > & operator= (const proxy< J > &rhs)`  
*Assignment (write access); with other proxy.*
- `proxy ()`  
*Constructor.*
- `proxy (I &ima, const typename I::psite &p)`  
*Constructor.*
- `I::value to_value () const`  
*Explicit read access.*
- `~proxy ()`  
*Destructor.*

### 10.401.1 Detailed Description

`template<typename I> class mln::value::proxy< I >`

Generic proxy class for an image pixel value. The parameter `I` is an image type.

### 10.401.2 Member Typedef Documentation

**10.401.2.1** `template<typename I> typedef void mln::value::proxy< I >::enc`

Encoding associated type.

**10.401.2.2** `template<typename I> typedef I::value mln::value::proxy< I >::equiv`

Equivalent associated type.

### 10.401.3 Constructor & Destructor Documentation

**10.401.3.1** `template<typename I > mln::value::proxy< I >::proxy ( ) [inline]`

Constructor.

**10.401.3.2** `template<typename I > mln::value::proxy< I >::proxy ( I & ima, const typename I::psite & p ) [inline]`

Constructor.

**10.401.3.3** `template<typename I > mln::value::proxy< I >::~~proxy ( ) [inline]`

Destructor.

### 10.401.4 Member Function Documentation

**10.401.4.1** `template<typename I> proxy< I> & mln::value::proxy< I>::operator= ( const proxy< I> & rhs ) [inline]`

Assignment (write access); replacement for default op.

References mln::value::proxy< I>::to\_value().

**10.401.4.2** `template<typename I> template<typename J> proxy< I> & mln::value::proxy< I>::operator= ( const proxy< J> & rhs ) [inline]`

Assignment (write access); with other proxy.

References mln::value::proxy< I>::to\_value().

**10.401.4.3** `template<typename I> I::value mln::value::proxy< I>::to_value ( ) const [inline]`

Explicit read access.

Referenced by mln::value::proxy< I>::operator=().

## 10.402 mln::value::qt::rgb32 Struct Reference

Color class for red-green-blue where every component is n-bit encoded.

```
#include <rgb32.hh>
```

Inherits mln::value::Vectorial< rgb32 >, and mln::value::internal::value\_like\_< algebra::vec< 3, int\_u< 8 >>, algebra::vec< 3, int\_u< 8 >>, algebra::vec< 3, int >, rgb32 >.

### Public Member Functions

- `rgb32 & operator= (const rgb32 &rhs)`  
*Assignment.*
- `rgb32 (const algebra::vec< 3, int > &rhs)`  
*Constructor from a algebra::vec.*
- `rgb32 (int r, int g, int b)`  
*Constructor from component values.*
- `rgb32 ()`  
*Constructor without argument.*
- `int_u< 8 > red () const`  
*Acces to red/green/blue component.*

- `rgb32` (const `mln::literal::zero_t` &)  
*Constructors with literals.*

## Static Public Attributes

- static const `rgb32 zero`  
*Zero value.*

### 10.402.1 Detailed Description

Color class for red-green-blue where every component is n-bit encoded.

### 10.402.2 Constructor & Destructor Documentation

#### 10.402.2.1 `mln::value::qt::rgb32::rgb32 ( ) [inline]`

Constructor without argument.

#### 10.402.2.2 `mln::value::qt::rgb32::rgb32 ( int r, int g, int b ) [inline]`

Constructor from component values.

#### 10.402.2.3 `mln::value::qt::rgb32::rgb32 ( const algebra::vec< 3,int > & rhs ) [inline]`

Constructor from a `algebra::vec`.

#### 10.402.2.4 `mln::value::qt::rgb32::rgb32 ( const mln::literal::zero_t & ) [inline]`

Constructors with literals.

### 10.402.3 Member Function Documentation

#### 10.402.3.1 `rgb32 & mln::value::qt::rgb32::operator= ( const rgb32 & rhs ) [inline]`

Assignment.

#### 10.402.3.2 `int_u<8> mln::value::qt::rgb32::red ( ) const [inline]`

Acces to red/green/blue component.

### 10.402.4 Member Data Documentation

#### 10.402.4.1 `const rgb32 mln::value::qt::rgb32::zero [static]`

Zero value.

## 10.403 `mln::value::rgb<n>` Struct Template Reference

Color class for red-green-blue where every component is n-bit encoded.

```
#include <rgb.hh>
```

Inherits `mln::value::Vectorial<rgb<n>>`, and `mln::value::internal::value_like_<algebra::vec<3, int_u<n>>, algebra::vec<3, int_u<n>>, algebra::vec<3, int_u<n>>, rgb<n>>`.

### Public Member Functions

- `rgb<n> & operator= (const rgb<n> &rhs)`  
*Assignment.*
- `rgb (const algebra::vec<3, int> &rhs)`  
*Constructor from a algebra::vec.*
- `rgb (int r, int g, int b)`  
*Constructor from component values.*
- `rgb ()`  
*Constructor without argument.*
- `int_u<n> red () const`  
*Acces to red/green/blue component.*
- `rgb (const mln::literal::white_t &)`  
*Constructors with literals.*

### Static Public Attributes

- `static const rgb<n> zero`  
*Zero value.*

#### 10.403.1 Detailed Description

```
template<unsigned n> struct mln::value::rgb<n>
```

Color class for red-green-blue where every component is n-bit encoded.

#### 10.403.2 Constructor & Destructor Documentation

**10.403.2.1** `template<unsigned n> mln::value::rgb<n>::rgb ( ) [inline]`

Constructor without argument.

**10.403.2.2** `template<unsigned n> mln::value::rgb<n>::rgb ( int r, int g, int b ) [inline]`

Constructor from component values.

**10.403.2.3** `template<unsigned n> mln::value::rgb<n>::rgb ( const algebra::vec<3, int> & rhs ) [inline]`

Constructor from a algebra::vec.

**10.403.2.4** `template<unsigned n> mln::value::rgb<n>::rgb ( const mln::literal::white_t & ) [inline]`

Constructors with literals.

### 10.403.3 Member Function Documentation

**10.403.3.1** `template<unsigned n> rgb<n> & mln::value::rgb<n>::operator= ( const rgb<n> & rhs ) [inline]`

Assignment.

**10.403.3.2** `template<unsigned n> int_u<n> mln::value::rgb<n>::red ( ) const [inline]`

Acces to red/green/blue component.

### 10.403.4 Member Data Documentation

**10.403.4.1** `template<unsigned n> const rgb<n> mln::value::rgb<n>::zero [static]`

Zero value.

## 10.404 mln::value::set< T > Struct Template Reference

Class that defines the set of values of type T.

```
#include <set.hh>
```

Inherits `set_selector_< T, set< T >, mln::metal::equal< mln::trait::value_< T >::quant, mln::trait::value::quant::low >::value >`.

### Static Public Member Functions

- static const `set< T > & the ()`

*Return a singleton.*

### 10.404.1 Detailed Description

**template<typename T> struct mln::value::set< T >**

Class that defines the set of values of type T. This is the exhaustive set of values obtainable from type T.

### 10.404.2 Member Function Documentation

**10.404.2.1 template<typename T > const set< T > & mln::value::set< T >::the ( ) [inline, static]**

Return a singleton.

## 10.405 mln::value::sign Class Reference

The sign class represents the value type composed by the set (-1, 0, 1) sign value type is a subset of the int value type.

```
#include <sign.hh>
```

Inherits Integer< sign >.

### Public Types

- typedef int [enc](#)  
*FIXME Are these typedefs correct?*
- typedef int [equiv](#)  
*Define the equivalent type.*

### Public Member Functions

- [operator int](#) () const  
*Conversion to an integer.*
- [sign](#) & [operator=](#) (int i)  
*Assignment from an integer.*
- [sign](#) ()  
*Constructor without argument.*
- [sign](#) (int i)  
*Constructor from an integer.*
- [sign](#) (const [mln::literal::zero\\_t](#) &)  
*Constructors/assignments with literals.*

## Static Public Attributes

- static const `sign one` = 1  
*Unit value.*
- static const `sign zero` = 0  
*Zero value.*

### 10.405.1 Detailed Description

The sign class represents the value type composed by the set (-1, 0, 1) sign value type is a subset of the int value type.

### 10.405.2 Member Typedef Documentation

#### 10.405.2.1 `typedef int mln::value::sign::enc`

FIXME Are these typedefs correct?

Define the encoding type

#### 10.405.2.2 `typedef int mln::value::sign::equiv`

Define the equivalent type.

### 10.405.3 Constructor & Destructor Documentation

#### 10.405.3.1 `mln::value::sign::sign ( ) [inline]`

Constructor without argument.

#### 10.405.3.2 `mln::value::sign::sign ( int i ) [inline]`

Constructor from an integer.

#### 10.405.3.3 `mln::value::sign::sign ( const mln::literal::zero_t & ) [inline]`

Constructors/assignments with literals.

### 10.405.4 Member Function Documentation

#### 10.405.4.1 `mln::value::sign::operator int ( ) const [inline]`

Conversion to an integer.



**10.405.4.2 sign & mln::value::sign::operator=( int i ) [inline]**

Assignment from an integer.

**10.405.5 Member Data Documentation****10.405.5.1 const sign mln::value::sign::one = 1 [static]**

Unit value.

**10.405.5.2 const sign mln::value::sign::zero = 0 [static]**

Zero value.

**10.406 mln::value::stack\_image< n, I > Struct Template Reference**

Stack image class.

```
#include <stack.hh>
```

Inherits image\_value\_morpher< I, algebra::vec< n, I::value >, stack\_image< n, I > >.

**Public Types**

- typedef I::domain\_t [domain\\_t](#)  
*Site\_Set associated type.*
- typedef internal::helper\_stack\_image\_lvalue\_< n, I >::ret [lvalue](#)  
*Return type of read-write access.*
- typedef I::psite [psite](#)  
*Point\_Site associated type.*
- typedef [value](#) [rvalue](#)  
*Return type of read-only access.*
- typedef [stack\\_image](#)< n, tag::image\_< I > > [skeleton](#)  
*Skeleton.*
- typedef algebra::vec< n, typename I::value > [value](#)  
*Value associated type.*

**Public Member Functions**

- bool [is\\_valid](#) () const  
*Test if this image has been initialized.*

- [lvalue operator\(\)](#) (const [psite](#) &)  
*Read-write access of pixel value at point site p.*
- [rvalue operator\(\)](#) (const [psite](#) &p) const  
*Read-only access of pixel value at point site p.*
- [stack\\_image](#) (const algebra::vec< n, I > &imas)  
*Constructors.*

### 10.406.1 Detailed Description

`template<unsigned n, typename I> struct mln::value::stack_image< n, I >`

Stack image class. [mln::value::stack\\_image](#) stores a vector of n images of the same domain.

The parameter n is the number of images, I is the type of a stack element. Acces a value will compute a vector which contains n coordinates : [stack[0](p), stack[1](p), ... , stack[n](p)]

### 10.406.2 Member Typedef Documentation

**10.406.2.1** `template<unsigned n, typename I> typedef I ::domain_t mln::value::stack_image< n, I >::domain_t`

[Site\\_Set](#) associated type.

**10.406.2.2** `template<unsigned n, typename I> typedef internal::helper_stack_image_lvalue<n,I>::ret mln::value::stack_image< n, I >::lvalue`

Return type of read-write access.

**10.406.2.3** `template<unsigned n, typename I> typedef I ::psite mln::value::stack_image< n, I >::psite`

[Point\\_Site](#) associated type.

**10.406.2.4** `template<unsigned n, typename I> typedef value mln::value::stack_image< n, I >::rvalue`

Return type of read-only access.

The rvalue type is not a const reference, since the value type is built on the fly, and return by value (copy).

**10.406.2.5** `template<unsigned n, typename I> typedef stack_image< n, tag::image_<I> > mln::value::stack_image< n, I >::skeleton`

Skeleton.

**10.406.2.6** `template<unsigned n, typename I> typedef algebra::vec<n, typename I::value>  
mln::value::stack_image< n, I >::value`

Value associated type.

### 10.406.3 Constructor & Destructor Documentation

**10.406.3.1** `template<unsigned n, typename I> mln::value::stack_image< n, I >::stack_image (   
const algebra::vec< n, I > & imas ) [inline]`

Constructors.

### 10.406.4 Member Function Documentation

**10.406.4.1** `template<unsigned n, typename I > bool mln::value::stack_image< n, I >::is_valid (   
) const [inline]`

Test if this image has been initialized.

**10.406.4.2** `template<unsigned n, typename I > stack_image< n, I >::lvalue  
mln::value::stack_image< n, I >::operator() ( const psite & p ) [inline]`

Read-write access of pixel value at point site *p*.

**10.406.4.3** `template<unsigned n, typename I > stack_image< n, I >::rvalue  
mln::value::stack_image< n, I >::operator() ( const psite & p ) const [inline]`

Read-only access of pixel value at point site *p*.

## 10.407 mln::value::super\_value< sign > Struct Template Reference

Specializations:

```
#include <super_value.hh>
```

### 10.407.1 Detailed Description

`template<> struct mln::value::super_value< sign >`

Specializations: Sign type is a subset of the short value type.

## 10.408 mln::value::value\_array< T, V > Struct Template Reference

Generic array class over indexed by a value set with type *T*.

```
#include <value_array.hh>
```

## Public Member Functions

- `const V & operator() (const T &v) const`  
}
- `const V & operator[] (unsigned i) const`  
}
- `value_array ()`  
*Constructors.*
- `const mln::value::set< T > & vset () const`  
}

### 10.408.1 Detailed Description

`template<typename T, typename V> struct mln::value::value_array< T, V >`

Generic array class over indexed by a value set with type T.

### 10.408.2 Constructor & Destructor Documentation

**10.408.2.1** `template<typename T, typename V > mln::value::value_array< T, V >::value_array ( ) [inline]`

Constructors.

```
{
```

### 10.408.3 Member Function Documentation

**10.408.3.1** `template<typename T, typename V > const V & mln::value::value_array< T, V >::operator() ( const T & v ) const [inline]`

```
}
```

Access elements through a value of T. {

**10.408.3.2** `template<typename T, typename V > const V & mln::value::value_array< T, V >::operator[] ( unsigned i ) const [inline]`

```
}
```

Access elements through array indexes. {

**10.408.3.3** `template<typename T, typename V > const mln::value::set< T > & mln::value::value_array< T, V >::vset ( ) const [inline]`

```
}
```

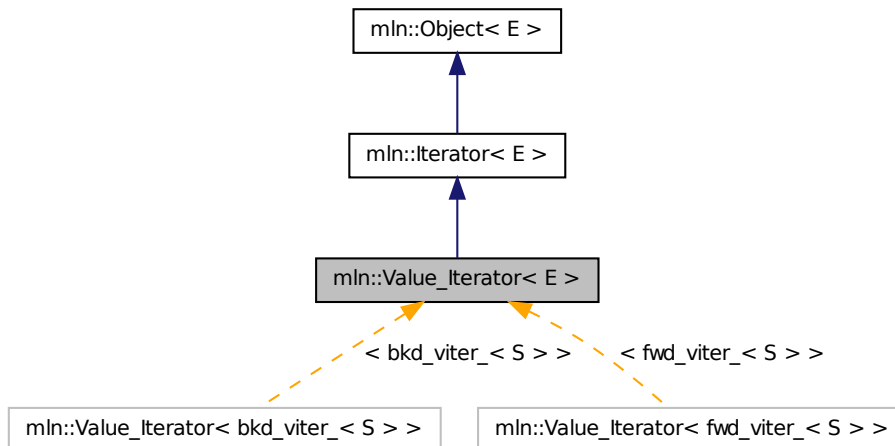
Reference to the set of T.

## 10.409 mln::Value\_Iterator< E > Struct Template Reference

Base class for implementation of classes of iterator on values.

```
#include <value_iterator.hh>
```

Inheritance diagram for mln::Value\_Iterator< E >:



### Public Member Functions

- void `next()`

*Go to the next element.*

### Related Functions

(Note that these are not member functions.)

- `template<typename E > std::ostream & operator<<< (std::ostream &ostr, const Value_Iterator< E > &v)`

*Print an iterator v on value set into the output stream ostr.*

### 10.409.1 Detailed Description

```
template<typename E> struct mln::Value_Iterator< E >
```

Base class for implementation of classes of iterator on values. An iterator on values is an iterator that browse over a set of values.

**See also**

[mln::doc::Value\\_Iterator](#) for a complete documentation of this class contents.

**10.409.2 Member Function Documentation****10.409.2.1** `template<typename E> void mln::Iterator< E >::next ( ) [inherited]`

Go to the next element.

**Warning**

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next\_* method.

**Precondition**

The iterator is valid.

**10.409.3 Friends And Related Function Documentation****10.409.3.1** `template<typename E> std::ostream & operator<< ( std::ostream & ostr, const Value_Iterator< E > & v ) [related]`

Print an iterator *v* on value set into the output stream *ostr*.

**Parameters**

[in, out] *ostr* An output stream.

[in] *v* An iterator on value set.

**Precondition**

*v* is a valid.

**Returns**

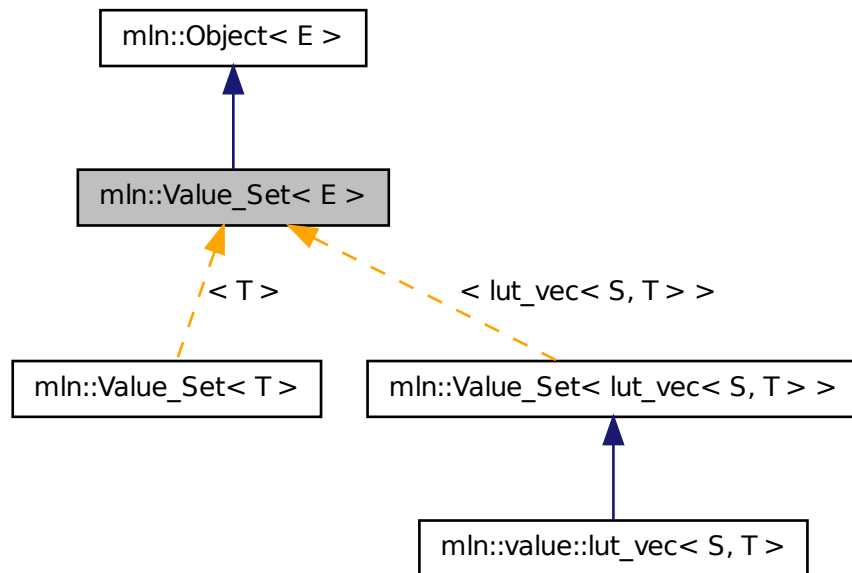
The modified output stream *ostr*.

**10.410 mln::Value\_Set< E > Struct Template Reference**

Base class for implementation classes of sets of values.

```
#include <value_set.hh>
```

Inheritance diagram for mln::Value\_Set< E >:



### 10.410.1 Detailed Description

```
template<typename E> struct mln::Value_Set< E >
```

Base class for implementation classes of sets of values.

See also

[mln::doc::Value\\_Set](#) for a complete documentation of this class contents.

## 10.411 mln::Vertex< E > Struct Template Reference

[Vertex](#) category flag type.

```
#include <vertex.hh>
```

### 10.411.1 Detailed Description

```
template<typename E> struct mln::Vertex< E >
```

[Vertex](#) category flag type.

## 10.412 mln::vertex\_image< P, V, G > Class Template Reference

[Image](#) based on graph vertices.

```
#include <vertex_image.hh>
```

Inherits [image\\_base< fun::i2v::array< V >, p\\_vertices< G, internal::vfsite\\_selector< P, G >::site\\_function\\_t >, vertex\\_image< P, V, G > >](#).

### Public Types

- typedef [G](#) [graph\\_t](#)  
*The type of the underlying graph.*
- typedef [vertex\\_nbh\\_t](#) [nbh\\_t](#)  
*Neighborhood type.*
- typedef [internal::vfsite\\_selector< P, G >::site\\_function\\_t](#) [site\\_function\\_t](#)  
*Function mapping graph elements to sites.*
- typedef [vertex\\_image< tag::psite\\_< P >, tag::value\\_< V >, tag::graph\\_< G > >](#) [skeleton](#)  
*Skeleton type.*
- typedef [graph\\_elt\\_neighborhood< G, S >](#) [vertex\\_nbh\\_t](#)  
*Vertex Neighborhood type.*
- typedef [graph\\_elt\\_window< G, S >](#) [vertex\\_win\\_t](#)  
*Vertex Window type.*
- typedef [vertex\\_win\\_t](#) [win\\_t](#)  
*Window type.*

### Public Member Functions

- [vertex\\_image](#) ()  
*Constructors.*
- rvalue [operator](#)() (unsigned v\_id) const  
*Value accessors/operators overloads.*

#### 10.412.1 Detailed Description

```
template<typename P, typename V, typename G = util::graph> class mln::vertex_image< P, V, G >
```

[Image](#) based on graph vertices.



## 10.412.2 Member Typedef Documentation

**10.412.2.1** `template<typename P, typename V, typename G = util::graph> typedef G mln::vertex_image< P, V, G >::graph_t`

The type of the underlying graph.

**10.412.2.2** `template<typename P, typename V, typename G = util::graph> typedef vertex_nbh_t mln::vertex_image< P, V, G >::nbh_t`

[Neighborhood](#) type.

**10.412.2.3** `template<typename P, typename V, typename G = util::graph> typedef internal::vfsite_selector<P,G>::site_function_t mln::vertex_image< P, V, G >::site_function_t`

[Function](#) mapping graph elements to sites.

**10.412.2.4** `template<typename P, typename V, typename G = util::graph> typedef vertex_image< tag::psite_<P>, tag::value_<V>, tag::graph_<G> > mln::vertex_image< P, V, G >::skeleton`

[Skeleton](#) type.

**10.412.2.5** `template<typename P, typename V, typename G = util::graph> typedef graph_elt_neighborhood<G,S> mln::vertex_image< P, V, G >::vertex_nbh_t`

[Vertex Neighborhood](#) type.

**10.412.2.6** `template<typename P, typename V, typename G = util::graph> typedef graph_elt_window<G,S> mln::vertex_image< P, V, G >::vertex_win_t`

[Vertex Window](#) type.

**10.412.2.7** `template<typename P, typename V, typename G = util::graph> typedef vertex_win_t mln::vertex_image< P, V, G >::win_t`

[Window](#) type.

## 10.412.3 Constructor & Destructor Documentation

**10.412.3.1** `template<typename P, typename V, typename G > mln::vertex_image< P, V, G >::vertex_image( ) [inline]`

Constructors.

### 10.412.4 Member Function Documentation

**10.412.4.1** `template<typename P, typename V, typename G> vertex_image< P, V, G >::rvalue  
mln::vertex_image< P, V, G >::operator() ( unsigned v_id ) const`

[Value](#) accessors/operators overloads.

## 10.413 mln::violent\_cast\_image< T, I > Struct Template Reference

Violently cast image values to a given type.

```
#include <violent_cast_image.hh>
```

Inherits `image_value_morpher< I, T, violent_cast_image< T, I > >`.

### Public Types

- typedef T [lvalue](#)  
*Return type of read-write access.*
- typedef T [rvalue](#)  
*Return type of read-only access.*
- typedef [violent\\_cast\\_image](#)< tag::value\_< T >, tag::image\_< I > > [skeleton](#)  
*Skeleton.*
- typedef T [value](#)  
*Value associated type.*

### Public Member Functions

- T [operator\(\)](#) (const typename I::psite &p) const  
*Read-only access of pixel value at point site p.*
- T [operator\(\)](#) (const typename I::psite &p)  
*Mutable access is only OK for reading (not writing).*
- [violent\\_cast\\_image](#) (const [Image](#)< I > &ima)  
*Constructor.*

### 10.413.1 Detailed Description

```
template<typename T, typename I> struct mln::violent_cast_image< T, I >
```

Violently cast image values to a given type.

### 10.413.2 Member Typedef Documentation

#### 10.413.2.1 `template<typename T, typename I> typedef T mln::violent_cast_image< T, I >::lvalue`

Return type of read-write access.

#### 10.413.2.2 `template<typename T, typename I> typedef T mln::violent_cast_image< T, I >::rvalue`

Return type of read-only access.

#### 10.413.2.3 `template<typename T, typename I> typedef violent_cast_image< tag::value_<T>, tag::image_<I> > mln::violent_cast_image< T, I >::skeleton`

Skeleton.

#### 10.413.2.4 `template<typename T, typename I> typedef T mln::violent_cast_image< T, I >::value`

[Value](#) associated type.

### 10.413.3 Constructor & Destructor Documentation

#### 10.413.3.1 `template<typename T , typename I > mln::violent_cast_image< T, I >::violent_cast_image( const Image< I > & ima ) [inline]`

Constructor.

### 10.413.4 Member Function Documentation

#### 10.413.4.1 `template<typename T , typename I > T mln::violent_cast_image< T, I >::operator()( const typename I::psite & p ) const [inline]`

Read-only access of pixel value at point site p.

#### 10.413.4.2 `template<typename T , typename I > T mln::violent_cast_image< T, I >::operator()( const typename I::psite & p ) [inline]`

Mutable access is only OK for reading (not writing).

## 10.414 mln::w\_window< D, W > Struct Template Reference

Generic [w\\_window](#) class.

```
#include <w_window.hh>
```

Inherits [weighted\\_window\\_base< mln::window< D >, w\\_window< D, W > >](#).

## Public Types

- typedef with\_w\_< [dpsites\\_bkd\\_piter](#)< [w\\_window](#)< D, W > >, W > [bkd\\_qiter](#)  
*Site\_Iterator type to browse (backward) the points of a generic [w\\_window](#).*
- typedef D [dpsite](#)  
*Dpsite associated type.*
- typedef with\_w\_< [dpsites\\_fwd\\_piter](#)< [w\\_window](#)< D, W > >, W > [fwd\\_qiter](#)  
*Site\_Iterator type to browse (forward) the points of a generic [w\\_window](#).*
- typedef W [weight](#)  
*Weight associated type.*

## Public Member Functions

- void [clear](#) ()  
*Clear this window.*
- [w\\_window](#)< D, W > & [insert](#) (const W &w, const D &d)  
*Insert a couple of weight w and delta-point d.*
- bool [is\\_symmetric](#) () const  
*Test if the window is symmetric.*
- const std::vector< D > & [std\\_vector](#) () const  
*Give access to the vector of delta-points.*
- void [sym](#) ()  
*Apply a central symmetry to the window.*
- W [w](#) (unsigned i) const  
*Give the i-th weight.*
- [w\\_window](#) ()  
*Constructor without argument.*
- const std::vector< W > & [weights](#) () const  
*Give access to the vector of weights.*
- const [mln::window](#)< D > & [win](#) () const  
*Give the corresponding window.*

## Related Functions

(Note that these are not member functions.)

- `template<typename D, typename W >`  
`std::ostream & operator<< (std::ostream &ostr, const w_window< D, W > &w_win)`  
*Print a weighted window `w_win` into an output stream `ostr`.*
- `template<typename D, typename Wl, typename Wr >`  
`bool operator== (const w_window< D, Wl > &lhs, const w_window< D, Wr > &rhs)`  
*Equality test between two weighted windows `lhs` and `rhs`.*

### 10.414.1 Detailed Description

`template<typename D, typename W> struct mln::w_window< D, W >`

Generic `w_window` class. This type of `w_window` is just like a set of delta-points. The parameter `D` is the type of delta-points; the parameter `W` is the type of weights.

### 10.414.2 Member Typedef Documentation

**10.414.2.1** `template<typename D, typename W> typedef with_w_< dpsites_bkd_piter< w_window<D, W> >, W > mln::w_window< D, W >::bkd_qiter`

[Site\\_Iterator](#) type to browse (backward) the points of a generic `w_window`.

**10.414.2.2** `template<typename D, typename W> typedef D mln::w_window< D, W >::dpsite`

Dpsite associated type.

**10.414.2.3** `template<typename D, typename W> typedef with_w_< dpsites_fwd_piter< w_window<D, W> >, W > mln::w_window< D, W >::fwd_qiter`

[Site\\_Iterator](#) type to browse (forward) the points of a generic `w_window`.

**10.414.2.4** `template<typename D, typename W> typedef W mln::w_window< D, W >::weight`

Weight associated type.

### 10.414.3 Constructor & Destructor Documentation

**10.414.3.1** `template<typename D, typename W > mln::w_window< D, W >::w_window ( )`  
`[inline]`

Constructor without argument.

### 10.414.4 Member Function Documentation

**10.414.4.1** `template<typename D , typename W > void mln::w_window< D, W >::clear ( )`  
**[inline]**

Clear this window.

**10.414.4.2** `template<typename D , typename W > w_window< D, W > & mln::w_window< D, W >::insert ( const W & w, const D & d )` **[inline]**

Insert a couple of weight *w* and delta-point *d*.

Referenced by `mln::w_window< D, W >::sym()`, `mln::make::w_window()`, `mln::make::w_window1d()`, `mln::make::w_window3d()`, and `mln::make::w_window_directional()`.

**10.414.4.3** `template<typename D , typename W > bool mln::w_window< D, W >::is_symmetric ( ) const` **[inline]**

Test if the window is symmetric.

References `mln::w_window< D, W >::sym()`.

**10.414.4.4** `template<typename D , typename W > const std::vector< D > & mln::w_window< D, W >::std_vector ( ) const` **[inline]**

Give access to the vector of delta-points.

**10.414.4.5** `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()`.

**10.414.4.6** `template<typename D , typename W > W mln::w_window< D, W >::w ( unsigned i ) const` **[inline]**

Give the *i*-th weight.

**10.414.4.7** `template<typename D , typename W > const std::vector< W > & mln::w_window< D, W >::weights ( ) const` **[inline]**

Give access to the vector of weights.

Referenced by `mln::w_window< D, W >::operator==( )`.

**10.414.4.8** `template<typename D , typename W > const mln::window< D > & mln::w_window< D, W >::win ( ) const [inline]`

Give the corresponding window.

Referenced by `mln::w_window< D, W >::operator==( )`.

## 10.414.5 Friends And Related Function Documentation

**10.414.5.1** `template<typename D , typename W > std::ostream & operator<< ( std::ostream & ostr, const w_window< D, W > & w_win ) [related]`

Print a weighted window `w_win` into an output stream `ostr`.

**10.414.5.2** `template<typename D , typename Wl , typename Wr > bool operator==( const w_window< D, Wl > & lhs, const w_window< D, Wr > & rhs ) [related]`

Equality test between two weighted windows `lhs` and `rhs`.

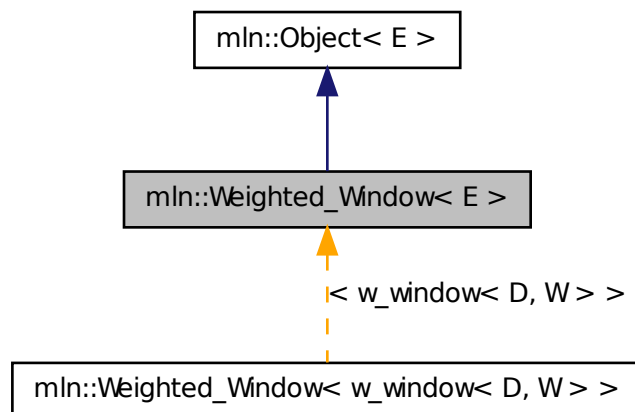
References `mln::w_window< D, W >::weights( )`, and `mln::w_window< D, W >::win( )`.

## 10.415 mln::Weighted\_Window< E > Struct Template Reference

Base class for implementation classes that are weighted\_windows.

```
#include <weighted_window.hh>
```

Inheritance diagram for `mln::Weighted_Window< E >`:



## Related Functions

(Note that these are not member functions.)

- `template<typename W >`  
`W operator-` (const [Weighted\\_Window](#)< W > &rhs)  
*Compute the symmetrical weighted window of rhs.*

### 10.415.1 Detailed Description

`template<typename E> struct mln::Weighted_Window< E >`

Base class for implementation classes that are `weighted_windows`.

See also

[mln::doc::Weighted\\_Window](#) for a complete documentation of this class contents.

### 10.415.2 Friends And Related Function Documentation

**10.415.2.1** `template<typename W > W operator-` ( const [Weighted\\_Window](#)< W > & rhs )  
**[related]**

Compute the symmetrical weighted window of rhs.

## 10.416 mln::win::backdiag2d Struct Reference

Diagonal line window defined on the 2D square grid.

```
#include <backdiag2d.hh>
```

Inherits `classical_window_base< dpoint2d, backdiag2d >`.

### Public Member Functions

- [backdiag2d](#) (unsigned length)  
*Constructor.*
- unsigned [length](#) () const  
*Give the diagonal length, that is, its width.*

### 10.416.1 Detailed Description

Diagonal line window defined on the 2D square grid. An [backdiag2d](#) is centered and symmetric. its width (length) is odd.

For instance:



```

*   o
*   o
*   x
*   o
*   o
*

```

is defined with length = 5.

## 10.416.2 Constructor & Destructor Documentation

### 10.416.2.1 mln::win::backdiag2d::backdiag2d ( unsigned *length* ) [inline]

Constructor.

#### Parameters

[in] *length* Length, thus width, of the diagonal line.

#### Precondition

*length* is odd.

## 10.416.3 Member Function Documentation

### 10.416.3.1 unsigned mln::win::backdiag2d::length ( ) const [inline]

Give the diagonal length, that is, its width.

## 10.417 mln::win::ball< G, C > Struct Template Reference

Generic ball window defined on a given grid.

```
#include <ball.hh>
```

Inherits classical\_window\_base< dpoint< G, C >, ball< G, C > >.

### Public Member Functions

- [ball](#) (unsigned diameter)  
*Constructor.*
- unsigned [diameter](#) () const  
*Give the ball diameter.*

### 10.417.1 Detailed Description

```
template<typename G, typename C> struct mln::win::ball< G, C >
```

Generic ball window defined on a given grid. A ball is centered and symmetric; so its diameter is odd.

G is the given grid on which the ball is defined and C is the type of coordinates.

## 10.417.2 Constructor & Destructor Documentation

**10.417.2.1** `template<typename G , typename C > mln::win::ball< G, C >::ball ( unsigned diameter ) [inline]`

Constructor.

### Parameters

[in] *diameter* Diameter of the ball.

### Precondition

`diameter` is odd.

References `mln::literal::origin`.

## 10.417.3 Member Function Documentation

**10.417.3.1** `template<typename G , typename C > unsigned mln::win::ball< G, C >::diameter ( ) const [inline]`

Give the ball diameter.

## 10.418 mln::win::cube3d Struct Reference

Cube window defined on the 3D grid.

```
#include <cube3d.hh>
```

Inherits `classical_window_base< dpoint3d, cube3d >`.

### Public Member Functions

- `cube3d` (unsigned length)  
*Constructor.*
- unsigned `length` () const  
*Give the cube length, that is, its height.*

### 10.418.1 Detailed Description

Cube window defined on the 3D grid. An `cube3d` is centered and symmetric; so its height (length) is odd.

For instance:

```
*   o o o
*   o o o
*   o o o

*   o o o
*   o x o
```

```

*   o   o   o
*
*   o   o   o
*   o   o   o
*   o   o   o
*
*

```

is defined with `length = 3`.

## 10.418.2 Constructor & Destructor Documentation

### 10.418.2.1 mln::win::cube3d::cube3d ( unsigned *length* ) [inline]

Constructor.

#### Parameters

[in] *length* Length, thus height, of the `cube3d`.

#### Precondition

`length` is odd.

## 10.418.3 Member Function Documentation

### 10.418.3.1 unsigned mln::win::cube3d::length ( ) const [inline]

Give the cube length, that is, its height.

## 10.419 mln::win::cuboid3d Struct Reference

Cuboid defined on the 3-D square grid.

```
#include <cuboid3d.hh>
```

Inherits `classical_window_base< dpoint3d, cuboid3d >`.

### Public Member Functions

- `cuboid3d` (unsigned depth, unsigned height, unsigned width)  
*Constructor.*
- unsigned `volume` () const  
*Return the volume of the cuboid.*
- unsigned `depth` () const  
*Accessors.*
- unsigned `height` () const  
*Return the height of the cuboid.*

- unsigned `width` () const  
Return the width of the cuboid.

### 10.419.1 Detailed Description

Cuboid defined on the 3-D square grid. A `cuboid3d` is a 3-D window with cuboid (also known as rectangular prism or rectangular parallelepiped) shape. It is centered and symmetric.

For instance:

```

  o o o o o o o
  o o o o o o o
  o o o o o o o
  o o o o o o o
  o o o o o o o
  o o o o o o o

  o o o o o o o
  o o o 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
  o o o o o o o

```

is defined with `depth = 3`, `height = 5` and `width = 7`.

Reference: <http://en.wikipedia.org/wiki/Cuboid>

### 10.419.2 Constructor & Destructor Documentation

#### 10.419.2.1 `mln::win::cuboid3d::cuboid3d ( unsigned depth, unsigned height, unsigned width ) [inline]`

Constructor.

#### Parameters

- [in] *depth* The depth of the `cuboid3d`.
- [in] *height* The height of the `cuboid3d`.
- [in] *width* The width of the `cuboid3d`.

#### Precondition

Argument *depth*, *height* and *width* must be odd.

### 10.419.3 Member Function Documentation

#### 10.419.3.1 `unsigned mln::win::cuboid3d::depth ( ) const [inline]`

Accessors.

Return the depth of the cuboid.

**10.419.3.2 unsigned mln::win::cuboid3d::height ( ) const [inline]**

Return the height of the cuboid.

**10.419.3.3 unsigned mln::win::cuboid3d::volume ( ) const [inline]**

Return the volume of the cuboid.

**10.419.3.4 unsigned mln::win::cuboid3d::width ( ) const [inline]**

Return the width of the cuboid.

**10.420 mln::win::diag2d Struct Reference**

Diagonal line window defined on the 2D square grid.

```
#include <diag2d.hh>
```

Inherits classical\_window\_base< dpoint2d, diag2d >.

**Public Member Functions**

- [diag2d](#) (unsigned length)  
*Constructor.*
- unsigned [length](#) () const  
*Give the diagonal length, that is, its width.*

**10.420.1 Detailed Description**

Diagonal line window defined on the 2D square grid. An [diag2d](#) is centered and symmetric. its width (length) is odd.

For instance:

```
*           o
*          o
*         x
*        o
*       o
*      o
*
```

is defined with length = 5.

**10.420.2 Constructor & Destructor Documentation****10.420.2.1 mln::win::diag2d::diag2d ( unsigned length ) [inline]**

Constructor.

**Parameters**

[in] *length* Length, thus width, of the diagonal line.

**Precondition**

*length* is odd.

**10.420.3 Member Function Documentation****10.420.3.1 unsigned mln::win::diag2d::length ( ) const [inline]**

Give the diagonal length, that is, its width.

**10.421 mln::win::line< M, i, C > Struct Template Reference**

Generic line window defined on a given grid in the given dimension.

```
#include <line.hh>
```

Inherits classical\_window\_base< dpoint< M, C >, line< M, i, C > >.

**Public Types**

- enum  
*Direction.*

**Public Member Functions**

- unsigned [length](#) ( ) const  
*Give the line length.*
- [line](#) (unsigned length)  
*Constructor.*
- unsigned [size](#) ( ) const  
*Give the line size, that is, its length.*

**10.421.1 Detailed Description**

```
template<typename M, unsigned i, typename C> struct mln::win::line< M, i, C >
```

Generic line window defined on a given grid in the given dimension. An line is centered and symmetric; so its length is odd.

M is the given grid on which the line is defined, i is the given dimension of the line end C is the type of the coordinates.

**See also**

mln::win::hline2d for an exemple of his use.

## 10.421.2 Member Enumeration Documentation

### 10.421.2.1 template<typename M , unsigned i, typename C > anonymous enum

Direction.

## 10.421.3 Constructor & Destructor Documentation

### 10.421.3.1 template<typename M , unsigned i, typename C > mln::win::line< M, i, C >::line ( unsigned *length* ) [inline]

Constructor.

#### Parameters

[in] *length* Length of the line.

#### Precondition

*length* is odd.

References mln::dpoint< G, C >::set\_all().

## 10.421.4 Member Function Documentation

### 10.421.4.1 template<typename M , unsigned i, typename C > unsigned mln::win::line< M, i, C >::length ( ) const [inline]

Give the line length.

### 10.421.4.2 template<typename M , unsigned i, typename C > unsigned mln::win::line< M, i, C >::size ( ) const [inline]

Give the line size, that is, its length.

## 10.422 mln::win::multiple< W, F > Class Template Reference

Multiple window.

```
#include <multiple.hh>
```

Inherits window\_base< W::dpsite, multiple< W, F > >.

### 10.422.1 Detailed Description

```
template<typename W, typename F> class mln::win::multiple< W, F >
```

Multiple window.

## 10.423 mln::win::multiple\_size< n, W, F > Class Template Reference

Definition of a multiple-size window.

```
#include <multiple_size.hh>
```

Inherits window\_base< W::dpsite, multiple\_size< n, W, F > >.

### 10.423.1 Detailed Description

```
template<unsigned n, typename W, typename F> class mln::win::multiple_size< n, W, F >
```

Definition of a multiple-size window.

## 10.424 mln::win::octagon2d Struct Reference

Octagon window defined on the 2D square grid.

```
#include <octagon2d.hh>
```

Inherits classical\_window\_base< dpoint2d, octagon2d >.

### Public Member Functions

- unsigned [area](#) () const  
*Give the area.*
- unsigned [length](#) () const  
*Give the octagon length, that is, its width.*
- [octagon2d](#) (unsigned length)  
*Constructor.*

### 10.424.1 Detailed Description

Octagon window defined on the 2D square grid. An [octagon2d](#) is centered and symmetric.

The length L of the octagon is such as  $L = 6 * l + 1$  where  $l \geq 0$ .

For instance:

```
*      o o o
*    o o o o o
*  o o o o o o o
* o o o x o o o
* o o o o o o o
*   o o o o o
*    o o o
*
```

is defined with  $L = 7$  ( $l = 1$ ).



## 10.424.2 Constructor & Destructor Documentation

### 10.424.2.1 mln::win::octagon2d::octagon2d ( unsigned *length* ) [inline]

Constructor.

#### Parameters

[in] *length* Length, of the octagon.

#### Precondition

*length* is such as  $length = 6 * x + 1$  where  $x \geq 0$ .

## 10.424.3 Member Function Documentation

### 10.424.3.1 unsigned mln::win::octagon2d::area ( ) const [inline]

Give the area.

### 10.424.3.2 unsigned mln::win::octagon2d::length ( ) const [inline]

Give the octagon length, that is, its width.

## 10.425 mln::win::rectangle2d Struct Reference

Rectangular window defined on the 2D square grid.

```
#include <rectangle2d.hh>
```

Inherits `classical_window_base< dpoint2d, rectangle2d >`.

### Public Member Functions

- unsigned [area](#) () const  
*Give the rectangle area.*
- unsigned [height](#) () const  
*Give the rectangle height.*
- [rectangle2d](#) (unsigned height, unsigned width)  
*Constructor.*
- const std::vector< [dpoint2d](#) > & [std\\_vector](#) () const  
*Give the std vector of delta-points.*
- unsigned [width](#) () const  
*Give the rectangle width.*

### 10.425.1 Detailed Description

Rectangular window defined on the 2D square grid. A [rectangle2d](#) is a 2D window with rectangular shape. It is centered and symmetric.

For instance:

```
*  o o o o o
*  o o x o o
*  o o o o o
*
```

is defined with height = 3 and width = 5.

### 10.425.2 Constructor & Destructor Documentation

#### 10.425.2.1 `mln::win::rectangle2d::rectangle2d ( unsigned height, unsigned width ) [inline]`

Constructor.

#### Parameters

[in] *height* Height of the [rectangle2d](#).

[in] *width* Width of the [rectangle2d](#).

#### Precondition

Height and width are odd.

### 10.425.3 Member Function Documentation

#### 10.425.3.1 `unsigned mln::win::rectangle2d::area ( ) const [inline]`

Give the rectangle area.

#### 10.425.3.2 `unsigned mln::win::rectangle2d::height ( ) const [inline]`

Give the rectangle height.

#### 10.425.3.3 `const std::vector< dpoint2d > & mln::win::rectangle2d::std_vector ( ) const [inline]`

Give the std vector of delta-points.

#### 10.425.3.4 `unsigned mln::win::rectangle2d::width ( ) const [inline]`

Give the rectangle width.



- typedef `window< D > regular`  
*Regular window associated type.*

## Public Member Functions

- void `clear ()`  
*Clear the window.*
- unsigned `delta () const`  
*Give the maximum coordinate gap between the window center and a window point.*
- const D & `dp (unsigned i) const`  
*Give the *i*-th delta-point.*
- bool `has (const D &dp) const`  
*Test if *dp* is in this window definition.*
- `window< D > & insert (const D &dp)`  
*Insert a delta-point *dp*.*
- template<typename W >  
`window< D > & insert (const Window< W > &win)`  
*Insert another window *win*.*
- bool `is_centered () const`  
*Test if the window is centered.*
- bool `is_empty () const`  
*Test if the window is empty (null size; no delta-point).*
- bool `is_symmetric () const`
- void `print (std::ostream &ostr) const`  
*Print the window definition into *ostr*.*
- unsigned `size () const`  
*Give the window size, i.e., the number of delta-sites.*
- const std::vector< D > & `std_vector () const`  
*Give the std vector of delta-points.*
- void `sym ()`  
*Apply a central symmetry to the target window.*
- `window ()`  
*Constructor without argument.*
- `window< D > & insert (const typename D::coord &dind)`

## Related Functions

(Note that these are not member functions.)

- `template<typename D >`  
`bool operator== (const window< D > &lhs, const window< D > &rhs)`  
*Equality comparison between windows `lhs` and `rhs`.*

### 10.427.1 Detailed Description

`template<typename D> class mln::window< D >`

Generic window class. This type of window is just like a set of delta-points. The parameter is `D`, type of delta-point.

### 10.427.2 Member Typedef Documentation

**10.427.2.1** `template<typename D> typedef dpsites_bkd_piter< window<D> > mln::window< D >::bkd_qiter`

[Site\\_Iterator](#) type to browse the points of a basic window w.r.t. the reverse ordering of delta-points.

**10.427.2.2** `template<typename D> typedef dpsites_fwd_piter< window<D> > mln::window< D >::fwd_qiter`

[Site\\_Iterator](#) type to browse the points of a basic window w.r.t. the ordering of delta-points.

**10.427.2.3** `template<typename D> typedef fwd_qiter mln::window< D >::qiter`

[Site\\_Iterator](#) type to browse the points of a basic window whatever the ordering of delta-points.

**10.427.2.4** `template<typename D> typedef window<D> mln::window< D >::regular`

Regular window associated type.

### 10.427.3 Constructor & Destructor Documentation

**10.427.3.1** `template<typename D > mln::window< D >::window ( ) [inline]`

Constructor without argument.

The constructed window is empty.

### 10.427.4 Member Function Documentation

**10.427.4.1** `template<typename D > void mln::window< D >::clear ( ) [inline]`

Clear the window.

**10.427.4.2** `template<typename D> unsigned mln::window< D >::delta ( ) const [inline]`

Give the maximum coordinate gap between the window center and a window point.

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

**10.427.4.3** `template<typename D> const D & mln::window< D >::dp ( unsigned i ) const [inline]`

Give the *i*-th delta-point.

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

Referenced by `mln::window< D >::delta()`, and `mln::window< D >::insert()`.

**10.427.4.4** `template<typename D> bool mln::window< D >::has ( const D & dp ) const [inline]`

Test if `dp` is in this window definition.

**10.427.4.5** `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::c2_3d_sli()`, `mln::c4_3d()`, `mln::c6()`, `mln::window< D >::insert()`, `mln::morpho::line_gradient()`, `mln::window< D >::sym()`, `mln::convert::to_upper_window()`, `mln::convert::to_window()`, `mln::win_c4p()`, `mln::win_c4p_3d()`, `mln::win_c8p()`, and `mln::win_c8p_3d()`.

**10.427.4.6** `template<typename D> template<typename W> window< D > & mln::window< D >::insert ( const Window< W > & win ) [inline]`

Insert another window `win`.

**10.427.4.7** `template<typename D> window< D > & mln::window< D >::insert ( const typename D::coord & dind ) [inline]`

Insertion of a delta-point with different numbers of arguments (coordinates) w.r.t. the dimension.

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

**10.427.4.8** `template<typename D> bool mln::window< D >::is_centered ( ) const [inline]`

Test if the window is centered.

**Returns**

True if the delta-point 0 belongs to the window.

References `mln::literal::zero`.

**10.427.4.9** `template<typename D> bool mln::window< D >::is_empty ( ) const [inline]`

Test if the window is empty (null size; no delta-point).

**10.427.4.10** `template<typename D> bool mln::window< D >::is_symmetric ( ) const [inline]`

Test if the window is symmetric.

**Returns**

True if for every dp of this window, -dp is also in this window.

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

**10.427.4.11** `template<typename D> void mln::window< D >::print ( std::ostream & ostr ) const [inline]`

Print the window definition into `ostr`.

**10.427.4.12** `template<typename D> unsigned mln::window< D >::size ( ) const [inline]`

Give the window size, i.e., the number of delta-sites.

Referenced by mln::window< D >::delta(), mln::window< D >::dp(), mln::window< D >::sym(), mln::win\_c4p(), mln::win\_c4p\_3d(), mln::win\_c8p(), and mln::win\_c8p\_3d().

**10.427.4.13** `template<typename D> const std::vector< D > & mln::window< D >::std_vector ( ) const [inline]`

Give the std vector of delta-points.

**10.427.4.14** `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().

**10.427.5 Friends And Related Function Documentation****10.427.5.1** `template<typename D> bool operator==( const window< D > & lhs, const window< D > & rhs ) [related]`

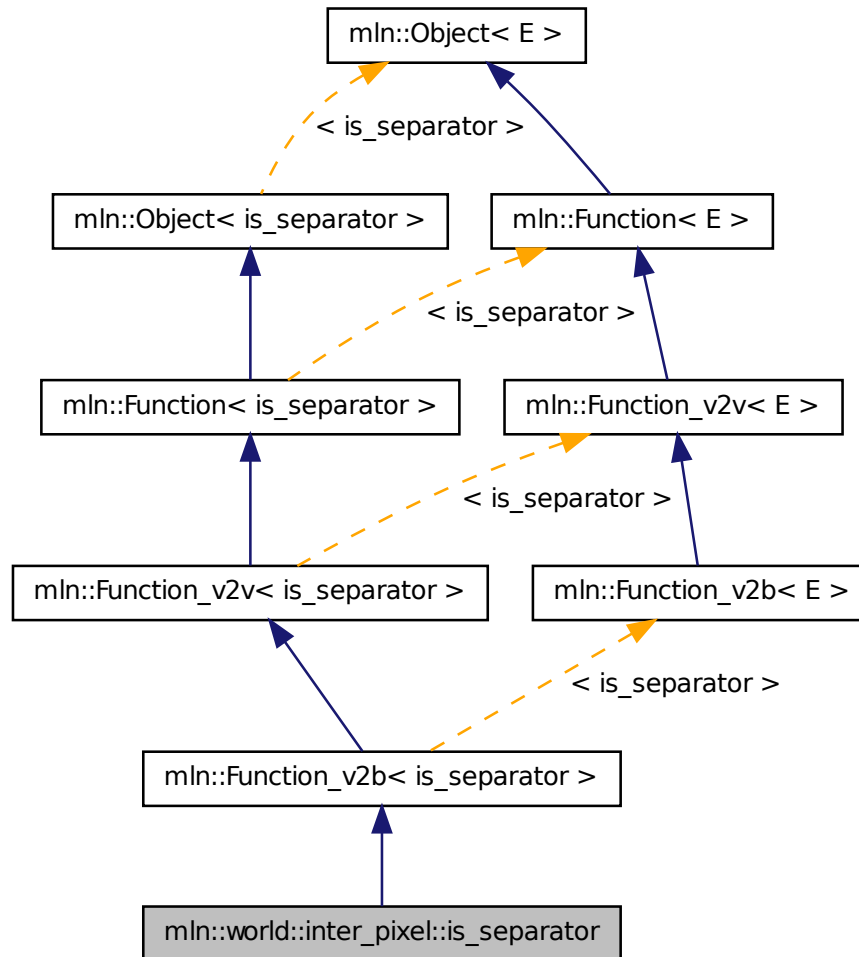
Equality comparison between windows `lhs` and `rhs`.

**10.428 mln::world::inter\_pixel::is\_separator Struct Reference**

Functor returning whether a site is a separator in an inter-pixel image.

```
#include <is_separator.hh>
```

Inheritance diagram for mln::world::inter\_pixel::is\_separator:



### 10.428.1 Detailed Description

Functor returning whether a site is a separator in an inter-pixel image.

## 10.429 trait::graph< I > Struct Template Reference

Graph traits.

```
#include <morpho.hh>
```



### 10.429.1 Detailed Description

```
template<typename I> struct trait::graph< I >
```

Graph traits.

## 10.430 `trait::graph< mln::complex_image< 1, G, V > >` Struct Template Reference

Graph traits for 1-complexes images.

```
#include <morpho.hh>
```

### 10.430.1 Detailed Description

```
template<typename G, typename V> struct trait::graph< mln::complex_image< 1, G, V > >
```

Graph traits for 1-complexes images.

## 10.431 `trait::graph< mln::image2d< T > >` Struct Template Reference

Graph traits for [mln::image2d](#).

```
#include <morpho.hh>
```

### 10.431.1 Detailed Description

```
template<typename T> struct trait::graph< mln::image2d< T > >
```

Graph traits for [mln::image2d](#).

# Index

- ~decorated\_image
  - mln::decorated\_image, [537](#)
- ~proxy
  - mln::value::proxy, [1032](#)
- ~soft\_heap
  - mln::util::soft\_heap, [993](#)
- ~tracked\_ptr
  - mln::util::tracked\_ptr, [996](#)
- \_1
  - mln::algebra::h\_mat, [488](#)
- 1D neighborhoods, [77](#)
- 1D windows, [88](#)
- 2D neighborhoods, [78](#)
- 2D windows, [89](#)
- 3D neighborhoods, [80](#)
- 3D windows, [91](#)
- a\_point\_of
  - mln, [123](#)
- abs
  - mln::data, [175](#)
  - mln::math, [313](#), [314](#)
- abs\_inplace
  - mln::data, [175](#)
- Accumulators, [75](#)
- add
  - mln::topo::n\_faces\_set, [934](#)
- add\_child
  - mln::util::tree\_node, [1001](#)
- add\_edge
  - mln::util::graph, [966](#)
- add\_face
  - mln::topo::complex, [915](#)
- add\_location
  - mln::geom::complex\_geometry, [661](#)
- add\_tree\_down
  - mln::util::tree, [998](#)
- add\_tree\_up
  - mln::util::tree, [998](#)
- add\_vertex
  - mln::util::graph, [966](#)
- add\_vertices
  - mln::util::graph, [967](#)
- addr
  - mln::topo::complex, [916](#)
- adj\_higher\_dim\_connected\_n\_face\_bkd\_iter
  - mln::topo::adj\_higher\_dim\_connected\_n\_face\_bkd\_iter, [890](#)
- adj\_higher\_dim\_connected\_n\_face\_fwd\_iter
  - mln::topo::adj\_higher\_dim\_connected\_n\_face\_fwd\_iter, [891](#)
- adj\_higher\_face\_bkd\_iter
  - mln::topo::adj\_higher\_face\_bkd\_iter, [892](#)
- adj\_higher\_face\_fwd\_iter
  - mln::topo::adj\_higher\_face\_fwd\_iter, [893](#)
- adj\_lower\_dim\_connected\_n\_face\_bkd\_iter
  - mln::topo::adj\_lower\_dim\_connected\_n\_face\_bkd\_iter, [894](#)
- adj\_lower\_dim\_connected\_n\_face\_fwd\_iter
  - mln::topo::adj\_lower\_dim\_connected\_n\_face\_fwd\_iter, [895](#)
- adj\_lower\_face\_bkd\_iter
  - mln::topo::adj\_lower\_face\_bkd\_iter, [896](#)
- adj\_lower\_face\_fwd\_iter
  - mln::topo::adj\_lower\_face\_fwd\_iter, [897](#)
- adj\_lower\_higher\_face\_bkd\_iter
  - mln::topo::adj\_lower\_higher\_face\_bkd\_iter, [898](#)
- adj\_lower\_higher\_face\_fwd\_iter
  - mln::topo::adj\_lower\_higher\_face\_fwd\_iter, [899](#)
- adj\_m\_face\_bkd\_iter
  - mln::topo::adj\_m\_face\_bkd\_iter, [900](#)
- adj\_m\_face\_fwd\_iter
  - mln::topo::adj\_m\_face\_fwd\_iter, [901](#)
- adjacency\_matrix
  - mln::util::adjacency\_matrix, [943](#)
- adjust
  - mln::border, [160](#)
  - mln::extension, [206](#), [207](#)
- adjust\_duplicate
  - mln::extension, [207](#)
- adjust\_fill
  - mln::extension, [207](#)
- algebraic\_face
  - mln::topo::algebraic\_face, [904](#)
- algebraic\_n\_face
  - mln::topo::algebraic\_n\_face, [908](#)
- and\_inplace
  - mln::logical, [287](#)

- and\_not
  - mln::logical, 287
- and\_not\_inplace
  - mln::logical, 287
- antialiased
  - mln::subsampling, 352
- apex
  - mln::util::branch, 950
- append
  - mln::p\_array, 776
  - mln::util::array, 947
- apply
  - mln::data, 176
- apply\_p2p
  - mln, 123
- area
  - mln::accu::site\_set::rectangularity, 458
  - mln::morpho::attribute::sharpness, 764
  - mln::morpho::attribute::volume, 768
  - mln::win::octagon2d, 1063
  - mln::win::rectangle2d, 1064
- argument
  - mln::accu::shape::height, 454
  - mln::accu::shape::volume, 456
  - mln::doc::Accumulator, 540
- array
  - mln::util::array, 946
- at
  - mln::opt, 345
- attachment
  - mln::make, 295
- backdiag2d
  - mln::win::backdiag2d, 1055
- background
  - mln::labeling, 260
- ball
  - mln::win::ball, 1056
- base\_level
  - mln::morpho::attribute::height, 763
- Basic types, 71, 84
- bbox
  - mln::accu::site\_set::rectangularity, 458
  - mln::Box, 504
  - mln::box, 498
  - mln::doc::Box, 542
  - mln::doc::Fastest\_Image, 549
  - mln::doc::Image, 558
  - mln::geom, 221
  - mln::image1d, 698
  - mln::image2d, 703
  - mln::image3d, 710
  - mln::labeled\_image, 720
  - mln::labeled\_image\_base, 723
  - mln::p\_line2d, 809
  - mln::p\_run, 832
- bbox\_t
  - mln::labeled\_image, 720
  - mln::labeled\_image\_base, 723
- bboxes
  - mln::labeled\_image, 720
  - mln::labeled\_image\_base, 724
- before
  - mln, 136
- begin
  - mln::p\_line2d, 809
- bin\_1complex\_image2d
  - mln, 119
- bin\_2complex\_image3df
  - mln, 119
- binarization
  - mln::binarization, 159
- bkd\_citer
  - mln::topo::complex, 915
- bkd\_eiter
  - mln::util::array, 946
  - mln::util::set, 987
- bkd\_niter
  - mln::doc::Neighborhood, 562
  - mln::graph\_elt\_mixed\_neighborhood, 669
  - mln::graph\_elt\_neighborhood, 674
  - mln::graph\_elt\_neighborhood\_if, 675
  - mln::mixed\_neighb, 759
  - mln::neighb, 770
- bkd\_piter
  - mln::box, 497
  - mln::doc::Box, 541
  - mln::doc::Fastest\_Image, 547
  - mln::doc::Image, 557
  - mln::doc::Site\_Set, 572
  - mln::hexa, 692
  - mln::image2d\_h, 706
  - mln::p\_array, 775
  - mln::p\_centered, 779
  - mln::p\_complex, 782
  - mln::p\_edges, 786
  - mln::p\_faces, 790
  - mln::p\_if, 795
  - mln::p\_image, 798
  - mln::p\_key, 804
  - mln::p\_line2d, 809
  - mln::p\_mutable\_array\_of, 812
  - mln::p\_priority, 818
  - mln::p\_queue, 823
  - mln::p\_queue\_fast, 827
  - mln::p\_run, 831
  - mln::p\_set, 836
  - mln::p\_set\_of, 839

- mln::p\_transformed, [842](#)
- mln::p\_vaccess, [846](#)
- mln::p\_vertices, [850](#)
- bkd\_pixter1d
  - mln::bkd\_pixter1d, [491](#)
- bkd\_pixter2d
  - mln::bkd\_pixter2d, [493](#)
- bkd\_pixter3d
  - mln::bkd\_pixter3d, [494](#)
- bkd\_qiter
  - mln::doc::Weighted\_Window, [578](#)
  - mln::doc::Window, [580](#)
  - mln::graph\_elt\_mixed\_window, [671](#)
  - mln::graph\_elt\_window, [678](#)
  - mln::graph\_elt\_window\_if, [681](#)
  - mln::w\_window, [1051](#)
  - mln::window, [1067](#)
- bkd\_viter
  - mln::doc::Value\_Set, [576](#)
  - mln::value::lut\_vec, [1029](#)
- black
  - mln::literal, [283](#)
- blobs
  - mln::canvas::labeling, [166](#)
  - mln::labeling, [260](#)
- blobs\_and\_compute
  - mln::labeling, [261](#)
- blue
  - mln::literal, [283](#)
- border
  - mln::doc::Fastest\_Image, [549](#)
  - mln::image1d, [698](#)
  - mln::image2d, [703](#)
  - mln::image3d, [710](#)
- box
  - mln::box, [498](#)
  - mln::draw, [202](#)
- box1d
  - mln, [119](#)
  - mln::make, [295](#)
- box2d
  - mln, [120](#)
  - mln::make, [296](#)
- box2d\_h
  - mln, [120](#)
  - mln::make, [296, 297](#)
- box3d
  - mln, [120](#)
  - mln::make, [297, 298](#)
- box\_plain
  - mln::draw, [202](#)
- box\_runend\_piter
  - mln::box\_runend\_piter, [507](#)
- box\_runstart\_piter
  - mln::box\_runstart\_piter, [508](#)
- branch
  - mln::util::branch, [950](#)
- brown
  - mln::literal, [284](#)
- buffer
  - mln::doc::Fastest\_Image, [549](#)
  - mln::image1d, [698](#)
  - mln::image2d, [703](#)
  - mln::image3d, [710](#)
- c18
  - modneighb3d, [81](#)
- c2
  - modneighb1d, [78](#)
- c26
  - modneighb3d, [81](#)
- c2\_3d\_sli
  - modneighb3d, [81](#)
- c2\_col
  - modneighb2d, [79](#)
- c2\_row
  - modneighb2d, [79](#)
- c4
  - modneighb2d, [79](#)
- c4\_3d
  - modneighb3d, [82](#)
- c6
  - modneighb3d, [82](#)
- c8
  - modneighb2d, [79](#)
- c8\_3d
  - modneighb3d, [83](#)
- can\_stop
  - mln::accu::logic::land\_basic, [400](#)
  - mln::accu::logic::lor\_basic, [402](#)
- Canvas, [75](#)
- card
  - mln::set, [350](#)
- cast
  - mln::value, [382](#)
- Category
  - mln::util::vertex, [1006](#)
- category
  - mln::util::edge, [958](#)
- cell
  - mln::make, [298](#)
- center
  - mln::p\_centered, [780](#)
- center\_only\_iter
  - mln::topo::center\_only\_iter, [911](#)
- center\_t
  - mln::graph\_elt\_mixed\_window, [671](#)
  - mln::graph\_elt\_window, [678](#)

- mln::graph\_window\_piter, 689
- center\_val
  - mln::dpoints\_bkd\_pixter, 587
  - mln::dpoints\_fwd\_pixter, 590
- centered\_bkd\_iter\_adapter
  - mln::topo::centered\_bkd\_iter\_adapter, 912
- centered\_fwd\_iter\_adapter
  - mln::topo::centered\_fwd\_iter\_adapter, 913
- chamfer
  - mln::geom, 221
- change
  - mln::p\_array, 776
- change\_both
  - mln::util::couple, 955
  - mln::util::ord\_pair, 982
- change\_extension
  - mln::extension\_val, 603
- change\_first
  - mln::util::couple, 955
  - mln::util::ord\_pair, 982
- change\_graph
  - mln::util::edge, 958
  - mln::util::vertex, 1006
- change\_key
  - mln::p\_key, 805
- change\_keys
  - mln::p\_key, 805
- change\_mask
  - mln::graph\_elt\_window\_if, 683
- change\_second
  - mln::util::couple, 955
  - mln::util::ord\_pair, 982
- change\_target
  - mln::complex\_psite, 531
  - mln::faces\_psite, 604
  - mln::p\_transformed\_piter, 844
- change\_target\_site\_set
  - mln::graph\_window\_piter, 690
- change\_to
  - mln::pixel, 855
- check\_consistency
  - mln::util::tree, 999
  - mln::util::tree\_node, 1001
- children
  - mln::util::tree\_node, 1002
- clear
  - mln::p\_array, 776
  - mln::p\_image, 799
  - mln::p\_key, 805
  - mln::p\_mutable\_array\_of, 813
  - mln::p\_priority, 819
  - mln::p\_queue, 823
  - mln::p\_queue\_fast, 828
  - mln::p\_set, 836
- mln::p\_set\_of, 840
- mln::util::array, 947
- mln::util::fibonacci\_heap, 961
- mln::util::set, 987
- mln::util::soft\_heap, 993
- mln::w\_window, 1052
- mln::window, 1067
- closing
  - mln::morpho::elementary, 327
- colorize
  - mln::labeling, 261, 262
- complementation
  - mln::morpho, 319
- complementation\_inplace
  - mln::morpho, 319
- complex
  - mln::topo::complex, 915
- Complex based, 85
- complex\_geometry
  - mln::geom::complex\_geometry, 661
- complex\_image
  - mln::complex\_image, 525
- complex\_neighborhood\_bkd\_piter
  - mln::complex\_neighborhood\_bkd\_piter, 527
- complex\_neighborhood\_fwd\_piter
  - mln::complex\_neighborhood\_fwd\_piter, 529
- complex\_psite
  - mln::complex\_psite, 531
- complex\_window\_bkd\_piter
  - mln::complex\_window\_bkd\_piter, 533
- complex\_window\_fwd\_piter
  - mln::complex\_window\_fwd\_piter, 535
- compose
  - mln, 124
- composed
  - mln::fun::x2x::composed, 639
- compute
  - mln::accu, 138
  - mln::data, 176
  - mln::graph, 230
  - mln::histo, 233
  - mln::labeling, 262, 263
  - mln::labeling::impl::generic, 273, 274
  - mln::set, 350
- compute\_attribute\_image
  - mln::morpho::tree, 331
- compute\_attribute\_image\_from
  - mln::morpho::tree, 332
- compute\_fastest
  - mln::labeling::impl, 272
- compute\_has
  - mln::p\_queue\_fast, 828
- compute\_image
  - mln::labeling, 264, 265

- compute\_parent
  - mln::morpho::tree, 332
- compute\_with\_weights
  - mln::set, 350
- contrast
  - mln::morpho, 319
- convert
  - mln::data, 177
- convolve
  - mln::linear::local, 279, 280
- coord
  - mln::def, 199
  - mln::doc::Dpoint, 544
  - mln::doc::Fastest\_Image, 547
  - mln::doc::Image, 557
  - mln::doc::Point\_Site, 567
  - mln::dpoint, 583
  - mln::point, 863
- coordf
  - mln::def, 199
- count
  - mln::accu::stat::mean, 464
- couple
  - mln::make, 298
- cplx
  - mln::p\_complex, 783
  - mln::p\_faces, 791
  - mln::topo::algebraic\_face, 904
  - mln::topo::algebraic\_n\_face, 909
  - mln::topo::face, 919
  - mln::topo::n\_face, 929
- crop\_wrt
  - mln::box, 499
- cube3d
  - mln::win::cube3d, 1057
- cuboid3d
  - mln::win::cuboid3d, 1058
- cyan
  - mln::literal, 284
- D
  - mln::topo::is\_simple\_cell, 927
- dark\_gray
  - mln::literal, 284
- dashed\_line
  - mln::draw, 202
- data
  - mln::topo::algebraic\_face, 904
  - mln::topo::algebraic\_n\_face, 909
  - mln::topo::face, 919
  - mln::topo::n\_face, 929
- data\_t
  - mln::fun::x2x::rotation, 643
  - mln::fun::x2x::translation, 646
- dec\_face\_id
  - mln::topo::algebraic\_face, 905
  - mln::topo::algebraic\_n\_face, 909
  - mln::topo::face, 919
  - mln::topo::n\_face, 929
- dec\_n
  - mln::topo::algebraic\_face, 905
  - mln::topo::face, 919
- decorated\_image
  - mln::decorated\_image, 537
- decoration
  - mln::decorated\_image, 537
- deepness
  - mln::util::branch\_iter, 951
  - mln::util::branch\_iter\_ind, 953
- delete\_tree\_node
  - mln::util::tree\_node, 1002
- delta
  - mln::doc::Weighted\_Window, 579
  - mln::geom, 221, 222
  - mln::graph\_elt\_mixed\_window, 672
  - mln::graph\_elt\_window, 679
  - mln::graph\_elt\_window\_if, 683
  - mln::graph\_window\_base, 685
  - mln::point, 863
  - mln::window, 1067
- delta\_index
  - mln::doc::Fastest\_Image, 549
  - mln::image1d, 699
  - mln::image2d, 703
  - mln::image3d, 710
- depth
  - mln::win::cuboid3d, 1058
- detach
  - mln::topo, 359
- detachment
  - mln::make, 299
- diag2d
  - mln::win::diag2d, 1059
- diameter
  - mln::win::ball, 1056
- diff
  - mln::Box, 504
  - mln::Site\_Set, 881
  - mln::win, 388
- diff\_abs
  - mln::arith, 149
- dilation
  - mln::morpho, 319
- dim
  - mln::complex\_image, 526
  - mln::doc::Dpoint, 544
  - mln::doc::Point\_Site, 568
  - mln::dpoint, 584

- mln::point, 864
- direct
  - mln::morpho::tree::filter, 337
- discrete\_plane\_1complex\_geometry
  - mln, 120
- discrete\_plane\_2complex\_geometry
  - mln, 120
- disk2d
  - modwin2d, 90
- display\_branch
  - mln::util, 372
- display\_tree
  - mln::util, 372
- distance\_and\_closest\_point\_geodesic
  - mln::transform, 365, 366
- distance\_and\_influence\_zone\_geodesic
  - mln::transform, 366
- distance\_front
  - mln::canvas, 164
  - mln::transform, 367
- distance\_geodesic
  - mln::canvas, 164
  - mln::transform, 367
- div
  - mln::arith, 149
- div\_cst
  - mln::arith, 149
- div\_inplace
  - mln::arith, 149
- domain
  - mln::complex\_image, 526
  - mln::doc::Fastest\_Image, 550
  - mln::doc::Image, 558
  - mln::extended, 597
  - mln::flat\_image, 607
  - mln::hexa, 693
  - mln::image1d, 699
  - mln::image2d, 703
  - mln::image2d\_h, 707
  - mln::image3d, 710
  - mln::image\_if, 713
  - mln::lazy\_image, 726
  - mln::p2p\_image, 773
  - mln::slice\_image, 884
  - mln::sub\_image, 885
  - mln::sub\_image\_if, 887
  - mln::tr\_image, 939
  - mln::transformed\_image, 941
  - mln::unproject\_image, 942
- Domain morphers, 73
- domain\_t
  - mln::value::stack\_image, 1040
- dp
  - mln::window, 1068
- dpoint
  - mln::doc::Dpoint, 544
  - mln::doc::Fastest\_Image, 547
  - mln::doc::Image, 557
  - mln::doc::Neighborhood, 562
  - mln::doc::Point\_Site, 567
  - mln::doc::Weighted\_Window, 578
  - mln::dpoint, 584, 585
- dpoint1d
  - mln, 120
- dpoint2d
  - mln, 120
- dpoint2d\_h
  - mln, 120
  - mln::make, 299
- dpoint3d
  - mln, 120
- dpoints\_bkd\_pixter
  - mln::dpoints\_bkd\_pixter, 587
- dpoints\_fwd\_pixter
  - mln::dpoints\_fwd\_pixter, 589
- dpsite
  - mln::point, 863
  - mln::w\_window, 1051
- dpsites\_bkd\_piter
  - mln::dpsites\_bkd\_piter, 591
- dpsites\_fwd\_piter
  - mln::dpsites\_fwd\_piter, 592
- draw\_graph
  - mln::debug, 194, 195
- dual\_input\_max\_tree
  - mln::morpho::tree, 333
- dummy\_p\_edges
  - mln::make, 299, 300
- dummy\_p\_vertices
  - mln::make, 300
- duplicate
  - mln, 124
  - mln::border, 160
  - mln::extension, 207
- e\_ith\_nbh\_edge
  - mln::util::graph, 967
  - mln::util::line\_graph, 974
- e\_nmax
  - mln::util::graph, 967
  - mln::util::line\_graph, 974
- e\_nmax\_nbh\_edges
  - mln::util::graph, 967
  - mln::util::line\_graph, 975
- edge
  - mln::p\_edges, 786
  - mln::topo, 359
  - mln::util::edge, 958

- mln::util::graph, 967
- mln::util::line\_graph, 975
- edge\_fwd\_iter
  - mln::util::graph, 965
  - mln::util::line\_graph, 974
- edge\_image
  - mln::edge\_image, 595
  - mln::make, 300–302
- edge\_nbh\_edge\_fwd\_iter
  - mln::util::graph, 965
  - mln::util::line\_graph, 974
- edge\_nbh\_t
  - mln::edge\_image, 594
- edge\_win\_t
  - mln::edge\_image, 594
- edge\_with
  - mln::util::vertex, 1006
- edges
  - mln::util::graph, 967
- edges\_set\_t
  - mln::util::graph, 965
- edges\_t
  - mln::util::graph, 965
  - mln::util::line\_graph, 974
- eiter
  - mln::util::array, 946
  - mln::util::set, 987
- element
  - mln::box, 497
  - mln::graph\_window\_if\_piter, 687
  - mln::graph\_window\_piter, 690
  - mln::image1d, 699
  - mln::image2d, 703
  - mln::image3d, 711
  - mln::p\_array, 775
  - mln::p\_centered, 779
  - mln::p\_complex, 782
  - mln::p\_edges, 786
  - mln::p\_faces, 790
  - mln::p\_if, 795
  - mln::p\_image, 798
  - mln::p\_key, 804
  - mln::p\_line2d, 809
  - mln::p\_mutable\_array\_of, 812
  - mln::p\_priority, 818
  - mln::p\_queue, 823
  - mln::p\_queue\_fast, 827
  - mln::p\_run, 831
  - mln::p\_set, 836
  - mln::p\_set\_of, 839
  - mln::p\_transformed, 842
  - mln::p\_vaccess, 846
  - mln::p\_vertices, 850
  - mln::util::array, 946
  - mln::util::set, 987
  - mln::util::soft\_heap, 993
- elt
  - mln::util::tree\_node, 1002
- empty
  - mln::p\_queue\_fast, 828
- enc
  - mln::value::float01, 1011
  - mln::value::label, 1026
  - mln::value::proxy, 1032
  - mln::value::sign, 1038
- end
  - mln::p\_line2d, 810
  - mln::p\_run, 832
- enlarge
  - mln::box, 499
- equalize
  - mln::border, 161
  - mln::histo, 233
- equiv
  - mln::value, 382
  - mln::value::float01, 1011
  - mln::value::proxy, 1032
  - mln::value::sign, 1038
- erosion
  - mln::morpho, 319
- exists\_key
  - mln::p\_key, 805
- exists\_priority
  - mln::p\_priority, 819
- extend
  - mln, 124
- extended
  - mln::extended, 596
- extension
  - mln::extension\_fun, 598
  - mln::extension\_ima, 600
  - mln::extension\_val, 603
- extension\_fun
  - mln::extension\_fun, 598
- extension\_ima
  - mln::extension\_ima, 600
- extension\_val
  - mln::extension\_val, 602
- f\_hsi\_to\_rgb\_3x8
  - mln::fun::v2v, 213
- f\_hsl\_to\_rgb\_3x8
  - mln::fun::v2v, 213
- f\_rgb\_to\_hsi\_f
  - mln::fun::v2v, 213
- f\_rgb\_to\_hsl\_f
  - mln::fun::v2v, 213
- face



- mln::complex\_psite, 531
- mln::faces\_psite, 605
- mln::topo::face, 918
- face\_bkd\_iter
  - mln::topo::face\_bkd\_iter, 921
- face\_fwd\_iter
  - mln::topo::face\_fwd\_iter, 922
- face\_id
  - mln::complex\_psite, 531
  - mln::faces\_psite, 605
  - mln::topo::algebraic\_face, 905
  - mln::topo::algebraic\_n\_face, 909
  - mln::topo::face, 919
  - mln::topo::n\_face, 929
- faces
  - mln::topo::n\_faces\_set, 934
- faces\_psite
  - mln::faces\_psite, 604
- faces\_type
  - mln::topo::n\_faces\_set, 934
- fast\_median
  - mln::data, 177
- fibonacci\_heap
  - mln::util::fibonacci\_heap, 961
- filename
  - mln::debug, 195
- fill
  - mln::border, 161
  - mln::data, 177
  - mln::extension, 207
  - mln::util::array, 947
- fill\_holes
  - mln::labeling, 265
- fill\_with\_image
  - mln::data, 178
  - mln::data::impl::generic, 189
- fill\_with\_value
  - mln::data, 178
  - mln::data::impl::generic, 189
- filter
  - mln::morpho::tree::filter, 338
- find
  - mln::border, 161
- first
  - mln::accu::pair, 449
  - mln::accu::stat::min\_max, 473
  - mln::util::couple, 955
  - mln::util::ord\_pair, 982
  - mln::util::site\_pair, 991
- first\_accu
  - mln::accu::pair, 449
  - mln::accu::stat::min\_max, 473
- first\_element
  - mln::util::set, 987
- flat\_image
  - mln::flat\_image, 607
- flat\_zones
  - mln::labeling, 266
- float01
  - mln::value::float01, 1011
- float01\_16
  - mln::value, 380
- float01\_8
  - mln::value, 380
- float01\_f
  - mln::value::float01\_f, 1012
- float\_2complex\_image3df
  - mln, 121
- flooding
  - mln::morpho::watershed, 340
- foreground
  - mln::labeling, 266
- format
  - mln::debug, 195
- from\_to
  - mln::convert, 170
- front
  - mln::p\_priority, 819
  - mln::p\_queue, 823
  - mln::p\_queue\_fast, 828
  - mln::util::fibonacci\_heap, 961
- fun
  - mln::p2p\_image, 773
- fun\_image
  - mln::fun\_image, 648, 649
- fun\_t
  - mln::p\_edges, 786
  - mln::p\_vertices, 850
- Function
  - mln::Function, 650
- function
  - mln::p\_edges, 788
  - mln::p\_transformed, 843
  - mln::p\_vertices, 852
- Functions, 76
- fwd\_citer
  - mln::topo::complex, 915
- fwd\_eiter
  - mln::util::array, 946
  - mln::util::set, 987
- fwd\_niter
  - mln::doc::Neighborhood, 563
  - mln::graph\_elt\_mixed\_neighborhood, 669
  - mln::graph\_elt\_neighborhood, 674
  - mln::graph\_elt\_neighborhood\_if, 675
  - mln::mixed\_neighb, 759
  - mln::neighb, 770
- fwd\_piter

- mln::box, 497
- mln::doc::Box, 541
- mln::doc::Fastest\_Image, 548
- mln::doc::Image, 557
- mln::doc::Site\_Set, 572
- mln::hexa, 692
- mln::image2d\_h, 706
- mln::p\_array, 775
- mln::p\_centered, 779
- mln::p\_complex, 782
- mln::p\_edges, 786
- mln::p\_faces, 790
- mln::p\_if, 795
- mln::p\_image, 798
- mln::p\_key, 804
- mln::p\_line2d, 809
- mln::p\_mutable\_array\_of, 812
- mln::p\_priority, 818
- mln::p\_queue, 823
- mln::p\_queue\_fast, 827
- mln::p\_run, 832
- mln::p\_set, 836
- mln::p\_set\_of, 839
- mln::p\_transformed, 842
- mln::p\_vaccess, 846
- mln::p\_vertices, 850
- fwd\_pixter1d
  - mln::fwd\_pixter1d, 655
- fwd\_pixter2d
  - mln::fwd\_pixter2d, 656
- fwd\_pixter3d
  - mln::fwd\_pixter3d, 657
- fwd\_qiter
  - mln::doc::Weighted\_Window, 578
  - mln::doc::Window, 580
  - mln::graph\_elt\_mixed\_window, 671
  - mln::graph\_elt\_window, 678
  - mln::graph\_elt\_window\_if, 681
  - mln::w\_window, 1051
  - mln::window, 1067
- fwd\_viter
  - mln::doc::Value\_Set, 576
  - mln::value::lut\_vec, 1029
- gaussian
  - mln::linear, 276
- gaussian\_1st\_derivative
  - mln::linear, 276, 277
- gaussian\_2nd\_derivative
  - mln::linear, 277
- gaussian\_subsampling
  - mln::subsampling, 352
- general
  - mln::morpho, 319
- geom
  - mln::complex\_image, 525
  - mln::p\_complex, 783
- get
  - mln::border, 162
  - mln::set, 350
- get\_header
  - mln::io::dicom, 237
  - mln::io::dump, 238
  - mln::io::raw, 255
- get\_rot
  - mln::registration, 347
- gl16
  - mln::value, 380
- gl8
  - mln::value, 380
- glf
  - mln::value, 380
- gradient
  - mln::morpho, 320
- gradient\_external
  - mln::morpho, 320
- gradient\_internal
  - mln::morpho, 320
- graph
  - mln::p\_edges, 788
  - mln::p\_graph\_piter, 793
  - mln::p\_vertices, 852
  - mln::util::edge, 958
  - mln::util::graph, 966
  - mln::util::line\_graph, 975
  - mln::util::vertex, 1006
- Graph based, 84
- graph\_element
  - mln::graph\_elt\_mixed\_window, 671
  - mln::graph\_elt\_window, 678
  - mln::graph\_window\_piter, 689
  - mln::p\_edges, 786
  - mln::p\_vertices, 850
- graph\_elt\_neighborhood\_if
  - mln::graph\_elt\_neighborhood\_if, 675
- graph\_elt\_window\_if
  - mln::graph\_elt\_window\_if, 682
- graph\_t
  - mln::edge\_image, 594
  - mln::p\_edges, 787
  - mln::p\_vertices, 851
  - mln::util::edge, 958
  - mln::util::vertex, 1006
  - mln::vertex\_image, 1047
- graph\_window\_if\_piter
  - mln::graph\_window\_if\_piter, 687
- graph\_window\_piter
  - mln::graph\_window\_piter, 689

- Graphes, [70](#)
- graylevel
  - mln::value::graylevel, [1014](#), [1015](#)
- graylevel\_f
  - mln::value::graylevel\_f, [1017](#)
- green
  - mln::literal, [284](#)
- grid
  - mln::dpoint, [583](#)
  - mln::point, [863](#)
- h\_mat
  - mln::algebra::h\_mat, [488](#)
  - mln::make, [302](#)
- h\_vec
  - mln::algebra::h\_vec, [490](#)
  - mln::point, [863](#)
- has
  - mln::box, [499](#)
  - mln::doc::Box, [542](#)
  - mln::doc::Fastest\_Image, [550](#)
  - mln::doc::Image, [558](#), [559](#)
  - mln::doc::Site\_Set, [572](#)
  - mln::doc::Value\_Set, [576](#)
  - mln::extension\_fun, [598](#)
  - mln::extension\_ima, [600](#)
  - mln::extension\_val, [603](#)
  - mln::flat\_image, [607](#)
  - mln::hexa, [693](#)
  - mln::image1d, [699](#)
  - mln::image2d, [703](#)
  - mln::image2d\_h, [707](#)
  - mln::image3d, [711](#)
  - mln::interpolated, [715](#)
  - mln::lazy\_image, [726](#)
  - mln::p\_array, [776](#)
  - mln::p\_centered, [780](#)
  - mln::p\_complex, [783](#)
  - mln::p\_edges, [788](#)
  - mln::p\_if, [795](#)
  - mln::p\_image, [799](#)
  - mln::p\_key, [806](#)
  - mln::p\_line2d, [810](#)
  - mln::p\_mutable\_array\_of, [813](#)
  - mln::p\_priority, [819](#)
  - mln::p\_queue, [823](#), [824](#)
  - mln::p\_queue\_fast, [828](#)
  - mln::p\_run, [833](#)
  - mln::p\_set, [836](#), [837](#)
  - mln::p\_set\_of, [840](#)
  - mln::p\_transformed, [843](#)
  - mln::p\_vaccess, [847](#)
  - mln::p\_vertices, [852](#)
  - mln::set, [350](#)
  - mln::tr\_image, [939](#)
  - mln::util::line\_graph, [975](#)
  - mln::util::set, [987](#)
  - mln::value::lut\_vec, [1030](#)
  - mln::window, [1068](#)
- has\_e
  - mln::util::graph, [968](#)
  - mln::util::line\_graph, [975](#)
- has\_index
  - mln::p\_run, [833](#)
- has\_v
  - mln::util::graph, [968](#)
  - mln::util::line\_graph, [976](#)
- height
  - mln::morpho::attribute::sharpness, [764](#)
  - mln::win::cuboid3d, [1058](#)
  - mln::win::rectangle2d, [1064](#)
- hexa
  - mln::hexa, [693](#)
- higher\_dim\_adj\_faces
  - mln::topo::algebraic\_face, [905](#)
  - mln::topo::algebraic\_n\_face, [909](#)
  - mln::topo::face, [919](#)
  - mln::topo::n\_face, [929](#)
- highest\_priority
  - mln::p\_priority, [819](#)
- hit\_or\_miss
  - mln::morpho, [320](#)
- hit\_or\_miss\_background\_closing
  - mln::morpho, [320](#)
- hit\_or\_miss\_background\_opening
  - mln::morpho, [321](#)
- hit\_or\_miss\_closing
  - mln::morpho, [321](#)
- hit\_or\_miss\_opening
  - mln::morpho, [321](#)
- hline2d
  - modwin2d, [90](#)
- horizontal\_symmetry
  - mln::geom, [222](#)
- hough
  - mln::transform, [367](#)
- i\_element
  - mln::p\_array, [775](#)
  - mln::p\_image, [798](#)
  - mln::p\_key, [804](#)
  - mln::p\_mutable\_array\_of, [812](#)
  - mln::p\_priority, [818](#)
  - mln::p\_queue, [823](#)
  - mln::p\_queue\_fast, [827](#)
  - mln::p\_set, [836](#)
  - mln::p\_set\_of, [839](#)
  - mln::p\_vaccess, [846](#)

- icp
  - mln::registration, 347
- id
  - mln::graph\_window\_if\_piter, 687
  - mln::graph\_window\_piter, 690
  - mln::p\_graph\_piter, 793
  - mln::util::edge, 958
  - mln::util::vertex, 1006
- id\_t
  - mln::util::edge, 958
  - mln::util::vertex, 1006
- id\_value\_t
  - mln::util::edge, 958
  - mln::util::vertex, 1006
- identity
  - mln::literal, 284
- Identity morphers, 74
- ima
  - mln::doc::Generalized\_Pixel, 554
  - mln::doc::Pixel\_Iterator, 566
  - mln::fun::x2x::linear, 641
  - mln::util::pix, 984
- image
  - mln::bkd\_pixter1d, 491
  - mln::bkd\_pixter2d, 492
  - mln::bkd\_pixter3d, 494
  - mln::doc::Generalized\_Pixel, 554
  - mln::doc::Pixel\_Iterator, 565
  - mln::fwd\_pixter1d, 655
  - mln::fwd\_pixter2d, 656
  - mln::fwd\_pixter3d, 657
  - mln::make, 302, 303
  - mln::pw::image, 872
- Image morphers, 72
- image1d
  - mln::image1d, 698
- image2d
  - mln::image2d, 702
  - mln::make, 303
- image2d\_h
  - mln::image2d\_h, 707
- image3d
  - mln::image3d, 710
  - mln::make, 303, 304
- image\_if
  - mln::image\_if, 713
- Images, 71
- implies
  - mln, 125
- inc\_face\_id
  - mln::topo::algebraic\_face, 905
  - mln::topo::algebraic\_n\_face, 909
  - mln::topo::face, 919
  - mln::topo::n\_face, 929
- inc\_n
  - mln::topo::algebraic\_face, 905
  - mln::topo::face, 919
- index
  - mln::p\_indexed\_bkd\_piter, 801
  - mln::p\_indexed\_fwd\_piter, 802
- index\_of
  - mln::doc::Value\_Set, 576
  - mln::value::lut\_vec, 1030
- influence\_zone\_adjacency\_graph
  - mln::make, 304
- influence\_zone\_front
  - mln::transform, 367
- influence\_zone\_geodesic
  - mln::transform, 368
- influence\_zone\_geodesic\_saturated
  - mln::transform, 368
- init
  - mln::accu::center, 390
  - mln::accu::convolve, 391
  - mln::accu::count\_adjacent\_vertices, 393
  - mln::accu::count\_value, 394
  - mln::accu::label\_used, 397
  - mln::accu::logic::land, 398
  - mln::accu::logic::land\_basic, 400
  - mln::accu::logic::lor, 401
  - mln::accu::logic::lor\_basic, 402
  - mln::accu::maj\_h, 404
  - mln::accu::math::count, 405
  - mln::accu::math::inf, 406
  - mln::accu::math::sum, 408
  - mln::accu::math::sup, 409
  - mln::accu::max\_site, 410
  - mln::accu::nil, 446
  - mln::accu::p, 447
  - mln::accu::pair, 449
  - mln::accu::rms, 451
  - mln::accu::shape::bbox, 452
  - mln::accu::shape::height, 454
  - mln::accu::shape::volume, 456
  - mln::accu::stat::deviation, 459
  - mln::accu::stat::max, 461
  - mln::accu::stat::max\_h, 462
  - mln::accu::stat::mean, 464
  - mln::accu::stat::median\_h, 467
  - mln::accu::stat::min, 469
  - mln::accu::stat::min\_h, 470
  - mln::accu::stat::min\_max, 473
  - mln::accu::stat::rank, 475
  - mln::accu::stat::rank < bool >, 476
  - mln::accu::stat::rank\_high\_quant, 477
  - mln::accu::stat::var, 479
  - mln::accu::stat::variance, 481
  - mln::accu::tuple, 483

- mln::accu::val, 485
- mln::doc::Accumulator, 540
- mln::morpho::attribute::card, 760
- mln::morpho::attribute::count\_adjacent\_vertices, 761
- mln::morpho::attribute::height, 763
- mln::morpho::attribute::sharpness, 765
- mln::morpho::attribute::sum, 766
- mln::morpho::attribute::volume, 768
- mln::p\_run, 833
- initialize
  - mln, 125
- insert
  - mln::p\_array, 777
  - mln::p\_image, 799
  - mln::p\_key, 806
  - mln::p\_mutable\_array\_of, 813
  - mln::p\_priority, 819
  - mln::p\_queue, 824
  - mln::p\_queue\_fast, 828
  - mln::p\_set, 837
  - mln::p\_set\_of, 840
  - mln::p\_vaccess, 847
  - mln::util::set, 988
  - mln::w\_window, 1052
  - mln::window, 1068
- int\_s
  - mln::value::int\_s, 1019
- int\_s16
  - mln::value, 380
- int\_s32
  - mln::value, 380
- int\_s8
  - mln::value, 381
- int\_u
  - mln::value::int\_u, 1021
- int\_u12
  - mln::value, 381
- int\_u16
  - mln::value, 381
- int\_u32
  - mln::value, 381
- int\_u8
  - mln::value, 381
- int\_u8\_1complex\_image2d
  - mln, 121
- int\_u8\_2complex\_image2d
  - mln, 121
- int\_u8\_2complex\_image3df
  - mln, 121
- int\_u\_sat
  - mln::value::int\_u\_sat, 1023
- inter
  - mln::Box, 504
  - mln::Site\_Set, 881
- interpolated
  - mln::interpolated, 715
- inv
  - mln::fun::x2x::rotation, 644
  - mln::fun::x2x::translation, 647
- invalidate
  - mln::complex\_psite, 531
  - mln::doc::Iterator, 561
  - mln::doc::Pixel\_Iterator, 566
  - mln::doc::Site\_Iterator, 570
  - mln::doc::Value\_Iterator, 574
  - mln::dpoints\_bkd\_pixter, 587
  - mln::dpoints\_fwd\_pixter, 590
  - mln::faces\_psite, 605
  - mln::p\_edges, 788
  - mln::p\_vertices, 853
  - mln::topo::algebraic\_face, 905
  - mln::topo::algebraic\_n\_face, 909
  - mln::topo::face, 920
  - mln::topo::n\_face, 929
  - mln::util::branch\_iter, 951
  - mln::util::branch\_iter\_ind, 953
  - mln::util::edge, 958
  - mln::util::vertex, 1007
- invert
  - mln::fun::x2x::rotation, 643
  - mln::fun::x2x::translation, 646
- iota
  - mln::debug, 196
- is\_centered
  - mln::doc::Weighted\_Window, 579
  - mln::graph\_elt\_mixed\_window, 672
  - mln::graph\_elt\_window, 679
  - mln::graph\_elt\_window\_if, 683
  - mln::graph\_window\_base, 685
  - mln::window, 1068
- is\_empty
  - mln::Box, 504
  - mln::box, 499
  - mln::doc::Weighted\_Window, 579
  - mln::graph\_elt\_mixed\_window, 672
  - mln::graph\_elt\_window, 679
  - mln::graph\_elt\_window\_if, 683
  - mln::graph\_window\_base, 685
  - mln::util::array, 947
  - mln::util::fibonacci\_heap, 961
  - mln::util::set, 988
  - mln::util::soft\_heap, 993
  - mln::window, 1068
- is\_facet
  - mln::topo, 359
- is\_simple\_2d
  - mln, 125

- is\_subgraph\_of
  - mln::util::graph, 968
  - mln::util::line\_graph, 976
- is\_symmetric
  - mln::graph\_elt\_mixed\_window, 672
  - mln::graph\_elt\_window, 679
  - mln::graph\_elt\_window\_if, 683
  - mln::graph\_window\_base, 685
  - mln::w\_window, 1052
  - mln::window, 1069
- is\_valid
  - mln::accu::center, 390
  - mln::accu::convolve, 391
  - mln::accu::count\_adjacent\_vertices, 393
  - mln::accu::count\_value, 394
  - mln::accu::histo, 396
  - mln::accu::label\_used, 397
  - mln::accu::logic::land, 398
  - mln::accu::logic::land\_basic, 400
  - mln::accu::logic::lor, 401
  - mln::accu::logic::lor\_basic, 402
  - mln::accu::maj\_h, 404
  - mln::accu::math::count, 405
  - mln::accu::math::inf, 406
  - mln::accu::math::sum, 408
  - mln::accu::math::sup, 409
  - mln::accu::max\_site, 410
  - mln::accu::nil, 446
  - mln::accu::p, 447
  - mln::accu::pair, 449
  - mln::accu::rms, 451
  - mln::accu::shape::bbox, 452
  - mln::accu::shape::height, 454
  - mln::accu::shape::volume, 456
  - mln::accu::stat::deviation, 459
  - mln::accu::stat::max, 461
  - mln::accu::stat::max\_h, 462
  - mln::accu::stat::mean, 464
  - mln::accu::stat::median\_alt, 465
  - mln::accu::stat::median\_h, 467
  - mln::accu::stat::min, 469
  - mln::accu::stat::min\_h, 470
  - mln::accu::stat::min\_max, 473
  - mln::accu::stat::rank, 475
  - mln::accu::stat::rank< bool >, 476
  - mln::accu::stat::rank\_high\_quant, 477
  - mln::accu::stat::var, 479
  - mln::accu::stat::variance, 481
  - mln::accu::tuple, 483
  - mln::accu::val, 485
  - mln::box, 499
  - mln::complex\_psite, 531
  - mln::doc::Fastest\_Image, 550
  - mln::doc::Image, 559
  - mln::doc::Iterator, 561
  - mln::doc::Pixel\_Iterator, 566
  - mln::doc::Site\_Iterator, 570
  - mln::doc::Value\_Iterator, 574
  - mln::dpoints\_bkd\_pixter, 587
  - mln::dpoints\_fwd\_pixter, 590
  - mln::faces\_psite, 605
  - mln::graph\_elt\_mixed\_window, 672
  - mln::graph\_elt\_window, 679
  - mln::graph\_elt\_window\_if, 683
  - mln::graph\_window\_base, 685
  - mln::interpolated, 715
  - mln::morpho::attribute::card, 760
  - mln::morpho::attribute::count\_adjacent\_vertices, 761
  - mln::morpho::attribute::height, 763
  - mln::morpho::attribute::sharpness, 765
  - mln::morpho::attribute::sum, 766
  - mln::morpho::attribute::volume, 768
  - mln::p\_array, 777
  - mln::p\_centered, 780
  - mln::p\_complex, 783
  - mln::p\_edges, 788
  - mln::p\_faces, 791
  - mln::p\_if, 795
  - mln::p\_image, 799
  - mln::p\_key, 806
  - mln::p\_line2d, 810
  - mln::p\_mutable\_array\_of, 813
  - mln::p\_priority, 820
  - mln::p\_queue, 824
  - mln::p\_queue\_fast, 828
  - mln::p\_run, 833
  - mln::p\_set, 837
  - mln::p\_set\_of, 840
  - mln::p\_transformed, 843
  - mln::p\_vaccess, 847
  - mln::p\_vertices, 853
  - mln::pixel, 855
  - mln::topo::algebraic\_face, 905
  - mln::topo::algebraic\_n\_face, 910
  - mln::topo::face, 920
  - mln::topo::n\_face, 929
  - mln::tr\_image, 939
  - mln::util::branch\_iter, 951
  - mln::util::branch\_iter\_ind, 953
  - mln::util::edge, 959
  - mln::util::fibonacci\_heap, 962
  - mln::util::soft\_heap, 993
  - mln::util::vertex, 1007
  - mln::value::stack\_image, 1041
- iter
  - mln::complex\_neighborhood\_bkd\_piter, 528
  - mln::complex\_neighborhood\_fwd\_piter, 529

- mln::complex\_window\_bkd\_piter, 533
- mln::complex\_window\_fwd\_piter, 535
- iter\_type
  - mln::complex\_neighborhood\_bkd\_piter, 527
  - mln::complex\_neighborhood\_fwd\_piter, 529
  - mln::complex\_window\_bkd\_piter, 533
  - mln::complex\_window\_fwd\_piter, 535
- ith\_nbh\_edge
  - mln::util::edge, 959
  - mln::util::vertex, 1007
- ith\_nbh\_vertex
  - mln::util::vertex, 1007
- k
  - mln::accu::stat::rank, 475
- key
  - mln::p\_key, 806
- keys
  - mln::p\_key, 806
- l1
  - mln::norm, 343
- l1\_distance
  - mln::norm, 343
- l2
  - mln::norm, 343
- l2\_distance
  - mln::norm, 343
- label
  - mln::value::label, 1027
- label\_16
  - mln::value, 381
- label\_32
  - mln::value, 381
- label\_8
  - mln::value, 381
- labeled\_image
  - mln::labeled\_image, 720
- labeled\_image\_base
  - mln::labeled\_image\_base, 723
- labeling
  - mln::graph, 230
- laplacian
  - mln::morpho, 321
- larger\_than
  - mln, 125
- last\_coord
  - mln::point, 864
- last\_element
  - mln::util::set, 988
- lazy\_image
  - mln::lazy\_image, 726
- ldlt\_decomp
  - mln::algebra, 145
- ldlt\_solve
  - mln::algebra, 145
- lemmings
  - mln::util, 373
- len
  - mln::Box, 504
  - mln::box, 500
- length
  - mln::p\_run, 833
  - mln::win::backdiag2d, 1055
  - mln::win::cube3d, 1057
  - mln::win::diag2d, 1060
  - mln::win::line, 1061
  - mln::win::octagon2d, 1063
- light\_gray
  - mln::literal, 284
- lime
  - mln::literal, 284
- line
  - mln::accu, 138
  - mln::draw, 203
  - mln::win::line, 1061
- line\_gradient
  - mln::morpho, 321
- linear
  - mln::fun::x2x::linear, 640
- linfty
  - mln::norm, 343
- linfty\_distance
  - mln::norm, 343
- load
  - mln::io::cloud, 236
  - mln::io::dicom, 237
  - mln::io::dump, 238
  - mln::io::fits, 239
  - mln::io::fld, 240
  - mln::io::magick, 241
  - mln::io::off, 242
  - mln::io::pbm, 243
  - mln::io::pbms, 244
  - mln::io::pfm, 245, 246
  - mln::io::pgm, 247
  - mln::io::pgms, 248
  - mln::io::plot, 249
  - mln::io::pnm, 250, 251
  - mln::io::pnms, 252
  - mln::io::ppm, 253
  - mln::io::ppms, 254
  - mln::io::raw, 255
  - mln::io::tiff, 256
- load\_ascii\_builtin
  - mln::io::pnm, 251
- load\_ascii\_value
  - mln::io::pnm, 251

- load\_raw\_2d
  - mln::io::pnm, 251
- lower\_dim\_adj\_faces
  - mln::topo::algebraic\_face, 906
  - mln::topo::algebraic\_n\_face, 910
  - mln::topo::face, 920
  - mln::topo::n\_face, 930
- lowest\_priority
  - mln::p\_priority, 820
- lut\_vec
  - mln::value::lut\_vec, 1030
- lvalue
  - mln::complex\_image, 525
  - mln::decorated\_image, 537
  - mln::doc::Fastest\_Image, 548
  - mln::doc::Image, 557
  - mln::doc::Pixel\_Iterator, 565
  - mln::flat\_image, 607
  - mln::fun\_image, 648
  - mln::hexa, 692
  - mln::image1d, 697
  - mln::image2d, 702
  - mln::image2d\_h, 706
  - mln::image3d, 709
  - mln::interpolated, 714
  - mln::lazy\_image, 726
  - mln::tr\_image, 938
  - mln::value::stack\_image, 1040
  - mln::violent\_cast\_image, 1049
- magenta
  - mln::literal, 284
- main\_branch
  - mln::util::tree, 999
- make\_algebraic\_face
  - mln::topo, 360
- make\_debug\_graph\_image
  - mln, 125
- make\_greater\_point
  - mln::util, 373
- make\_greater\_psite
  - mln::util, 373
- mask
  - mln::graph\_elt\_neighborhood\_if, 676
  - mln::graph\_elt\_window\_if, 683
- mask\_t
  - mln::graph\_elt\_window\_if, 682
- mat
  - mln::make, 304
- max
  - mln::literal, 284
  - mln::morpho::tree::filter, 338
- max\_col
  - mln::geom, 222
- max\_component
  - mln::io::pnm, 251
- max\_ind
  - mln::geom, 222
- max\_row
  - mln::geom, 222
- max\_sli
  - mln::geom, 222
- max\_tree
  - mln::morpho::tree, 334
- mean
  - mln::accu::stat::var, 479
  - mln::accu::stat::variance, 481
  - mln::estim, 204
- mean\_t
  - mln::accu::stat::var, 479
- median
  - mln::data, 178
  - mln::data::approx, 184, 185
  - mln::data::naive, 192
- medium\_gray
  - mln::literal, 284
- memory\_size
  - mln::box, 500
  - mln::p\_array, 777
  - mln::p\_centered, 780
  - mln::p\_edges, 789
  - mln::p\_if, 796
  - mln::p\_image, 799
  - mln::p\_key, 806
  - mln::p\_line2d, 810
  - mln::p\_mutable\_array\_of, 813
  - mln::p\_priority, 820
  - mln::p\_queue, 824
  - mln::p\_queue\_fast, 829
  - mln::p\_run, 833
  - mln::p\_set, 837
  - mln::p\_set\_of, 840
  - mln::p\_transformed, 843
  - mln::p\_vaccess, 848
  - mln::p\_vertices, 853
  - mln::util::array, 947
  - mln::util::set, 989
- merge
  - mln::box, 500
- mesh
  - mln::doc::Point\_Site, 567
- mesh\_corner\_point\_area
  - mln::geom, 223
- mesh\_curvature
  - mln::geom, 223
- mesh\_normal
  - mln::geom, 223
- meyer\_wst



- mln::morpho, 321, 322
- min
  - mln::arith, 150
  - mln::literal, 284
  - mln::morpho, 322
  - mln::morpho::tree::filter, 338
- min\_col
  - mln::geom, 224
- min\_ind
  - mln::geom, 224
- min\_inplace
  - mln::arith, 150
  - mln::morpho, 322
- min\_max
  - mln::estim, 205
- min\_row
  - mln::geom, 224
- min\_sli
  - mln::geom, 224
- min\_tree
  - mln::morpho::tree, 334
- minus
  - mln::arith, 150, 151
  - mln::morpho, 322
- minus\_cst
  - mln::arith, 152
- minus\_cst\_inplace
  - mln::arith, 152
- minus\_infty
  - mln::point, 865
- minus\_inplace
  - mln::arith, 153
- mirror
  - mln::border, 162
- mixed\_neighb
  - mln::mixed\_neighb, 759
- mln, 95
  - a\_point\_of, 123
  - apply\_p2p, 123
  - before, 136
  - bin\_1complex\_image2d, 119
  - bin\_2complex\_image3df, 119
  - box1d, 119
  - box2d, 120
  - box2d\_h, 120
  - box3d, 120
  - compose, 124
  - discrete\_plane\_1complex\_geometry, 120
  - discrete\_plane\_2complex\_geometry, 120
  - dpoint1d, 120
  - dpoint2d, 120
  - dpoint2d\_h, 120
  - dpoint3d, 120
  - duplicate, 124
  - extend, 124
  - float\_2complex\_image3df, 121
  - implies, 125
  - initialize, 125
  - int\_u8\_1complex\_image2d, 121
  - int\_u8\_2complex\_image2d, 121
  - int\_u8\_2complex\_image3df, 121
  - is\_simple\_2d, 125
  - larger\_than, 125
  - make\_debug\_graph\_image, 125
  - mln\_exact, 126
  - mln\_gen\_complex\_neighborhood, 126, 127
  - mln\_gen\_complex\_window, 127
  - mln\_gen\_complex\_window\_p, 127, 128
  - mln\_regular, 128
  - mln\_trait\_op\_geq, 128
  - mln\_trait\_op\_greater, 128
  - mln\_trait\_op\_leq, 129
  - mln\_trait\_op\_neq, 129
  - operator<, 131
  - operator<<, 131, 132
  - operator<=, 132
  - operator\*, 130
  - operator++, 130
  - operator-, 130
  - operator--, 131
  - operator==, 133, 134
  - p\_run2d, 121
  - p\_runs2d, 121
  - point1d, 121
  - point1df, 121
  - point2d, 121
  - point2d\_h, 121
  - point2df, 122
  - point3d, 122
  - point3df, 122
  - primary, 135
  - ptransform, 135
  - rgb8\_2complex\_image3df, 122
  - sagittal\_dec, 136
  - space\_2complex\_geometry, 122
  - unsigned\_2complex\_image3df, 122
  - up, 136
  - vec2d\_d, 122
  - vec2d\_f, 122
  - vec3d\_d, 122
  - vec3d\_f, 122
  - w\_window1d\_float, 123
  - w\_window1d\_int, 123
  - w\_window2d\_float, 123
  - w\_window2d\_int, 123
  - w\_window3d\_float, 123
  - w\_window3d\_int, 123
- mln::accu, 136

- compute, 138
- line, 138
- mln\_meta\_accu\_result, 138
- take, 139
- mln::accu::center, 389
  - init, 390
  - is\_valid, 390
  - nsites, 390
  - take\_as\_init, 390
  - take\_n\_times, 390
  - to\_result, 390
- mln::accu::convolve, 390
  - init, 391
  - is\_valid, 391
  - take\_as\_init, 391
  - take\_n\_times, 392
  - to\_result, 392
- mln::accu::count\_adjacent\_vertices, 392
  - init, 393
  - is\_valid, 393
  - set\_value, 393
  - take\_as\_init, 393
  - take\_n\_times, 393
  - to\_result, 393
- mln::accu::count\_value, 393
  - init, 394
  - is\_valid, 394
  - set\_value, 394
  - take\_as\_init, 394
  - take\_n\_times, 395
  - to\_result, 395
- mln::accu::histo, 395
  - is\_valid, 396
  - take, 396
  - take\_as\_init, 396
  - take\_n\_times, 396
  - vect, 396
- mln::accu::image, 139
- mln::accu::impl, 139
- mln::accu::label\_used, 396
  - init, 397
  - is\_valid, 397
  - take, 397
  - take\_as\_init, 397
  - take\_n\_times, 397
  - to\_result, 398
- mln::accu::logic, 140
- mln::accu::logic::land, 398
  - init, 398
  - is\_valid, 398
  - take\_as\_init, 399
  - take\_n\_times, 399
  - to\_result, 399
- mln::accu::logic::land\_basic, 399
  - can\_stop, 400
  - init, 400
  - is\_valid, 400
  - take\_as\_init, 400
  - take\_n\_times, 400
  - to\_result, 400
- mln::accu::logic::lor, 400
  - init, 401
  - is\_valid, 401
  - take\_as\_init, 401
  - take\_n\_times, 401
  - to\_result, 401
- mln::accu::logic::lor\_basic, 402
  - can\_stop, 402
  - init, 402
  - is\_valid, 402
  - take\_as\_init, 403
  - take\_n\_times, 403
  - to\_result, 403
- mln::accu::maj\_h, 403
  - init, 404
  - is\_valid, 404
  - take\_as\_init, 404
  - take\_n\_times, 404
  - to\_result, 404
- mln::accu::math, 140
- mln::accu::math::count, 404
  - init, 405
  - is\_valid, 405
  - set\_value, 405
  - take\_as\_init, 405
  - take\_n\_times, 405
  - to\_result, 406
- mln::accu::math::inf, 406
  - init, 406
  - is\_valid, 406
  - take\_as\_init, 407
  - take\_n\_times, 407
  - to\_result, 407
- mln::accu::math::sum, 407
  - init, 408
  - is\_valid, 408
  - take\_as\_init, 408
  - take\_n\_times, 408
  - to\_result, 408
- mln::accu::math::sup, 408
  - init, 409
  - is\_valid, 409
  - take\_as\_init, 409
  - take\_n\_times, 409
  - to\_result, 409
- mln::accu::max\_site, 410
  - init, 410
  - is\_valid, 410

- take\_as\_init, 410
- take\_n\_times, 411
- to\_result, 411
- mln::accu::meta::center, 411
- mln::accu::meta::count\_adjacent\_vertices, 412
- mln::accu::meta::count\_labels, 413
- mln::accu::meta::count\_value, 414
- mln::accu::meta::histo, 415
- mln::accu::meta::label\_used, 416
- mln::accu::meta::logic, 141
- mln::accu::meta::logic::land, 417
- mln::accu::meta::logic::land\_basic, 418
- mln::accu::meta::logic::lor, 419
- mln::accu::meta::logic::lor\_basic, 420
- mln::accu::meta::maj\_h, 421
- mln::accu::meta::math, 141
- mln::accu::meta::math::count, 422
- mln::accu::meta::math::inf, 423
- mln::accu::meta::math::sum, 424
- mln::accu::meta::math::sup, 425
- mln::accu::meta::max\_site, 426
- mln::accu::meta::nil, 427
- mln::accu::meta::p, 428
- mln::accu::meta::pair, 429
- mln::accu::meta::rms, 430
- mln::accu::meta::shape, 142
- mln::accu::meta::shape::bbox, 431
- mln::accu::meta::shape::height, 432
- mln::accu::meta::shape::volume, 433
- mln::accu::meta::stat, 142
- mln::accu::meta::stat::max, 434
- mln::accu::meta::stat::max\_h, 435
- mln::accu::meta::stat::mean, 436
- mln::accu::meta::stat::median\_alt, 437
- mln::accu::meta::stat::median\_h, 438
- mln::accu::meta::stat::min, 439
- mln::accu::meta::stat::min\_h, 440
- mln::accu::meta::stat::rank, 441
- mln::accu::meta::stat::rank\_high\_quant, 442
- mln::accu::meta::tuple, 443
- mln::accu::meta::val, 444
- mln::accu::nil, 445
  - init, 446
  - is\_valid, 446
  - take\_as\_init, 446
  - take\_n\_times, 446
  - to\_result, 446
- mln::accu::p, 447
  - init, 447
  - is\_valid, 447
  - take\_as\_init, 447
  - take\_n\_times, 447
  - to\_result, 448
- mln::accu::pair, 448
  - first, 449
  - first\_accu, 449
  - init, 449
  - is\_valid, 449
  - second, 450
  - second\_accu, 450
  - take\_as\_init, 450
  - take\_n\_times, 450
  - to\_result, 450
- mln::accu::rms, 450
  - init, 451
  - is\_valid, 451
  - take\_as\_init, 451
  - take\_n\_times, 451
  - to\_result, 451
- mln::accu::shape, 143
- mln::accu::shape::bbox, 452
  - init, 452
  - is\_valid, 452
  - take\_as\_init, 452
  - take\_n\_times, 453
  - to\_result, 453
- mln::accu::shape::height, 453
  - argument, 454
  - init, 454
  - is\_valid, 454
  - set\_value, 454
  - take\_as\_init, 454
  - take\_n\_times, 455
  - to\_result, 455
  - value, 454
- mln::accu::shape::volume, 455
  - argument, 456
  - init, 456
  - is\_valid, 456
  - set\_value, 456
  - take\_as\_init, 456
  - take\_n\_times, 457
  - to\_result, 457
  - value, 456
- mln::accu::site\_set::rectangularity, 457
  - area, 458
  - bbox, 458
  - rectangularity, 458
  - take\_as\_init, 458
  - take\_n\_times, 458
  - to\_result, 458
- mln::accu::stat, 143
- mln::accu::stat::deviation, 459
  - init, 459
  - is\_valid, 459
  - take\_as\_init, 459
  - take\_n\_times, 460
  - to\_result, 460

- mln::accu::stat::max, 460
  - init, 461
  - is\_valid, 461
  - set\_value, 461
  - take\_as\_init, 461
  - take\_n\_times, 461
  - to\_result, 461
- mln::accu::stat::max\_h, 461
  - init, 462
  - is\_valid, 462
  - take\_as\_init, 462
  - take\_n\_times, 462
  - to\_result, 462
- mln::accu::stat::mean, 463
  - count, 464
  - init, 464
  - is\_valid, 464
  - sum, 464
  - take\_as\_init, 464
  - take\_n\_times, 464
  - to\_result, 464
- mln::accu::stat::median\_alt, 464
  - is\_valid, 465
  - take, 465
  - take\_as\_init, 466
  - take\_n\_times, 466
  - to\_result, 466
- mln::accu::stat::median\_h, 466
  - init, 467
  - is\_valid, 467
  - take\_as\_init, 467
  - take\_n\_times, 467
  - to\_result, 467
- mln::accu::stat::meta::deviation, 468
- mln::accu::stat::min, 468
  - init, 469
  - is\_valid, 469
  - set\_value, 469
  - take\_as\_init, 469
  - take\_n\_times, 469
  - to\_result, 470
- mln::accu::stat::min\_h, 470
  - init, 470
  - is\_valid, 470
  - take\_as\_init, 471
  - take\_n\_times, 471
  - to\_result, 471
- mln::accu::stat::min\_max, 471
  - first, 473
  - first\_accu, 473
  - init, 473
  - is\_valid, 473
  - second, 473
  - second\_accu, 473
  - take\_as\_init, 473
  - take\_n\_times, 474
  - to\_result, 474
- mln::accu::stat::rank, 474
  - init, 475
  - is\_valid, 475
  - k, 475
  - take\_as\_init, 475
  - take\_n\_times, 475
  - to\_result, 475
- mln::accu::stat::rank< bool >, 475
  - init, 476
  - is\_valid, 476
  - take\_as\_init, 476
  - take\_n\_times, 476
  - to\_result, 476
- mln::accu::stat::rank\_high\_quant, 477
  - init, 477
  - is\_valid, 477
  - take\_as\_init, 478
  - take\_n\_times, 478
  - to\_result, 478
- mln::accu::stat::var, 478
  - init, 479
  - is\_valid, 479
  - mean, 479
  - mean\_t, 479
  - n\_items, 479
  - take\_as\_init, 480
  - take\_n\_times, 480
  - to\_result, 480
  - variance, 480
- mln::accu::stat::variance, 480
  - init, 481
  - is\_valid, 481
  - mean, 481
  - n\_items, 482
  - standard\_deviation, 482
  - sum, 482
  - take\_as\_init, 482
  - take\_n\_times, 482
  - to\_result, 482
  - var, 482
- mln::accu::tuple, 483
  - init, 483
  - is\_valid, 483
  - take\_as\_init, 483
  - take\_n\_times, 484
  - to\_result, 484
- mln::accu::val, 484
  - init, 485
  - is\_valid, 485
  - take\_as\_init, 485
  - take\_n\_times, 485

- to\_result, 485
- mln::Accumulator, 485
  - take\_as\_init, 487
  - take\_n\_times, 487
- mln::algebra, 144
  - ldlt\_decomp, 145
  - ldlt\_solve, 145
  - operator\*, 146
  - vprod, 146
- mln::algebra::h\_mat, 487
  - \_1, 488
  - h\_mat, 488
  - t, 489
- mln::algebra::h\_vec, 489
  - h\_vec, 490
  - operator mat< n, 1, U >, 490
  - origin, 490
  - t, 490
  - to\_vec, 490
  - zero, 490
- mln::arith, 146
  - diff\_abs, 149
  - div, 149
  - div\_cst, 149
  - div\_inplace, 149
  - min, 150
  - min\_inplace, 150
  - minus, 150, 151
  - minus\_cst, 152
  - minus\_cst\_inplace, 152
  - minus\_inplace, 153
  - plus, 153, 154
  - plus\_cst, 154, 155
  - plus\_cst\_inplace, 155
  - plus\_inplace, 155
  - revert, 156
  - revert\_inplace, 156
  - times, 157
  - times\_cst, 157
  - times\_inplace, 157
- mln::arith::impl, 158
- mln::arith::impl::generic, 158
- mln::binarization, 158
  - binarization, 159
  - threshold, 159
- mln::bkd\_pixter1d, 491
  - bkd\_pixter1d, 491
  - image, 491
  - next, 492
- mln::bkd\_pixter2d, 492
  - bkd\_pixter2d, 493
  - image, 492
  - next, 493
- mln::bkd\_pixter3d, 493
  - bkd\_pixter3d, 494
  - image, 494
  - next, 494
- mln::border, 159
  - adjust, 160
  - duplicate, 160
  - equalize, 161
  - fill, 161
  - find, 161
  - get, 162
  - mirror, 162
  - resize, 162
- mln::border::impl, 163
- mln::border::impl::generic, 163
- mln::Box, 501
  - bbox, 504
  - diff, 504
  - inter, 504
  - is\_empty, 504
  - len, 504
  - nsites, 504
  - operator<, 505
  - operator<<, 505
  - operator<=, 505
  - operator==, 506
  - sym\_diff, 506
  - uni, 506
  - unique, 506
- mln::box, 494
  - bbox, 498
  - bkd\_piter, 497
  - box, 498
  - crop\_wrt, 499
  - element, 497
  - enlarge, 499
  - fwd\_piter, 497
  - has, 499
  - is\_empty, 499
  - is\_valid, 499
  - len, 500
  - memory\_size, 500
  - merge, 500
  - nsites, 500
  - operator<<, 501
  - pcenter, 500
  - piter, 498
  - pmax, 500
  - pmin, 501
  - psite, 498
  - site, 498
  - to\_larger, 501
- mln::box\_runend\_piter, 506
  - box\_runend\_piter, 507
  - next, 507

- run\_length, 507
- mln::box\_runstart\_piter, 507
  - box\_runstart\_piter, 508
  - next, 508
  - run\_length, 508
- mln::Browsing, 509
- mln::canvas, 163
  - distance\_front, 164
  - distance\_geodesic, 164
- mln::canvas::browsing, 165
- mln::canvas::browsing::backdiagonal2d\_t, 509
- mln::canvas::browsing::breadth\_first\_search\_t, 511
- mln::canvas::browsing::depth\_first\_search\_t, 511
- mln::canvas::browsing::diagonal2d\_t, 511
- mln::canvas::browsing::dir\_struct\_elt\_incr\_update\_t, 513
- mln::canvas::browsing::directional\_t, 514
- mln::canvas::browsing::fwd\_t, 516
- mln::canvas::browsing::hyper\_directional\_t, 517
- mln::canvas::browsing::snake\_fwd\_t, 519
- mln::canvas::browsing::snake\_generic\_t, 520
- mln::canvas::browsing::snake\_vert\_t, 522
- mln::canvas::chamfer, 523
- mln::canvas::impl, 165
- mln::canvas::labeling, 166
  - blobs, 166
- mln::canvas::labeling::impl, 167
- mln::canvas::morpho, 167
- mln::category< R(\*) (A) >, 523
- mln::complex\_image, 523
  - complex\_image, 525
  - dim, 526
  - domain, 526
  - geom, 525
  - lvalue, 525
  - operator(), 526
  - rvalue, 525
  - skeleton, 525
  - value, 525
  - values, 526
- mln::complex\_neighborhood\_bkd\_piter, 526
  - complex\_neighborhood\_bkd\_piter, 527
  - iter, 528
  - iter\_type, 527
  - next, 528
  - psite, 527
- mln::complex\_neighborhood\_fwd\_piter, 528
  - complex\_neighborhood\_fwd\_piter, 529
  - iter, 529
  - iter\_type, 529
  - next, 529
  - psite, 529
- mln::complex\_psite, 530
  - change\_target, 531
  - complex\_psite, 531
  - face, 531
  - face\_id, 531
  - invalidate, 531
  - is\_valid, 531
  - n, 532
  - site\_set, 532
- mln::complex\_window\_bkd\_piter, 532
  - complex\_window\_bkd\_piter, 533
  - iter, 533
  - iter\_type, 533
  - next, 533
  - psite, 533
- mln::complex\_window\_fwd\_piter, 534
  - complex\_window\_fwd\_piter, 535
  - iter, 535
  - iter\_type, 535
  - next, 535
  - psite, 535
- mln::convert, 167
  - from\_to, 170
  - mln\_image\_from\_grid, 170
  - mln\_window, 170
  - to, 170
  - to\_dpoint, 171
  - to\_fun, 171, 173
  - to\_image, 171
  - to\_p\_array, 171
  - to\_p\_set, 171, 172
  - to\_qimage, 172
  - to\_upper\_window, 172
  - to\_window, 172, 173
- mln::data, 173
  - abs, 175
  - abs\_inplace, 175
  - apply, 176
  - compute, 176
  - convert, 177
  - fast\_median, 177
  - fill, 177
  - fill\_with\_image, 178
  - fill\_with\_value, 178
  - median, 178
  - mln\_meta\_accu\_result, 178
  - paste, 179
  - paste\_without\_localization, 179
  - replace, 179
  - saturate, 180
  - saturate\_inplace, 180
  - sort\_offsets\_increasing, 180
  - sort\_psites\_decreasing, 181
  - sort\_psites\_increasing, 181
  - stretch, 181
  - to\_enc, 181

- transform, 182
- transform\_inplace, 182, 183
- update, 183
- wrap, 183
- mln::data::approx, 184
  - median, 184, 185
- mln::data::approx::impl, 185
- mln::data::impl, 185
  - paste\_without\_localization\_fast, 186
  - paste\_without\_localization\_fastest, 187
  - paste\_without\_localization\_lines, 187
  - stretch, 187
  - transform\_inplace\_lowq, 188
  - update\_fastest, 188
- mln::data::impl::generic, 188
  - fill\_with\_image, 189
  - fill\_with\_value, 189
  - paste, 190
  - transform, 190
  - transform\_inplace, 190, 191
  - update, 191
- mln::data::naive, 191
  - median, 192
- mln::data::naive::impl, 192
- mln::debug, 192
  - draw\_graph, 194, 195
  - filename, 195
  - format, 195
  - iota, 196
  - mosaic, 196
  - println, 196
  - println\_with\_border, 196
  - put\_word, 196
  - slices\_2d, 197
  - superpose, 197
  - z\_order, 198
- mln::debug::impl, 198
- mln::decorated\_image, 535
  - ~decorated\_image, 537
  - decorated\_image, 537
  - decoration, 537
  - lvalue, 537
  - operator decorated\_image< const I, D >, 537
  - operator(), 538
  - psite, 537
  - rvalue, 537
  - skeleton, 537
- mln::def, 198
  - coord, 199
  - coordf, 199
- mln::Delta\_Point\_Site, 538
- mln::Delta\_Point\_Site< void >, 539
- mln::display, 199
- mln::display::impl, 199
- mln::display::impl::generic, 200
- mln::doc, 200
- mln::doc::Accumulator, 539
  - argument, 540
  - init, 540
  - take, 540
- mln::doc::Box, 540
  - bbox, 542
  - bkd\_piter, 541
  - fwd\_piter, 541
  - has, 542
  - nsites, 542
  - pmax, 542
  - pmin, 543
  - psite, 542
  - site, 542
- mln::doc::Dpoint, 543
  - coord, 544
  - dim, 544
  - dpoint, 544
  - point, 544
- mln::doc::Fastest\_Image, 545
  - bbox, 549
  - bkd\_piter, 547
  - border, 549
  - buffer, 549
  - coord, 547
  - delta\_index, 549
  - domain, 550
  - dpoint, 547
  - fwd\_piter, 548
  - has, 550
  - is\_valid, 550
  - lvalue, 548
  - nelements, 550
  - nsites, 551
  - operator(), 551
  - point, 548
  - point\_at\_index, 552
  - pset, 548
  - psite, 548
  - rvalue, 548
  - skeleton, 548
  - value, 549
  - values, 552
  - vset, 549
- mln::doc::Generalized\_Pixel, 553
  - ima, 554
  - image, 554
  - rvalue, 554
  - val, 554
  - value, 554
- mln::doc::Image, 554
  - bbox, 558

- bkd\_piter, 557
- coord, 557
- domain, 558
- dpoint, 557
- fwd\_piter, 557
- has, 558, 559
- is\_valid, 559
- lvalue, 557
- nsites, 559
- operator(), 559
- point, 557
- pset, 557
- psite, 557
- rvalue, 558
- skeleton, 558
- value, 558
- values, 560
- vset, 558
- mln::doc::Iterator, 560
  - invalidate, 561
  - is\_valid, 561
  - start, 561
- mln::doc::Neighborhood, 561
  - bkd\_niter, 562
  - dpoint, 562
  - fwd\_niter, 563
  - niter, 563
  - point, 563
- mln::doc::Object, 563
- mln::doc::Pixel\_Iterator, 564
  - ima, 566
  - image, 565
  - invalidate, 566
  - is\_valid, 566
  - lvalue, 565
  - rvalue, 565
  - start, 566
  - val, 566
  - value, 565
- mln::doc::Point\_Site
  - dim, 568
- mln::doc::Point\_Site, 566
  - coord, 567
  - dpoint, 567
  - mesh, 567
  - point, 567
  - to\_point, 568
- mln::doc::Site\_Iterator, 568
  - invalidate, 570
  - is\_valid, 570
  - operator psite, 570
  - psite, 570
  - start, 570
- mln::doc::Site\_Set, 570
  - bkd\_piter, 572
  - fwd\_piter, 572
  - has, 572
  - psite, 572
  - site, 572
- mln::doc::Value\_Iterator, 572
  - invalidate, 574
  - is\_valid, 574
  - operator value, 574
  - start, 574
  - value, 574
- mln::doc::Value\_Set, 574
  - bkd\_viter, 576
  - fwd\_viter, 576
  - has, 576
  - index\_of, 576
  - nvalues, 576
  - value, 576
- mln::doc::Weighted\_Window, 577
  - bkd\_qiter, 578
  - delta, 579
  - dpoint, 578
  - fwd\_qiter, 578
  - is\_centered, 579
  - is\_empty, 579
  - point, 578
  - sym, 579
  - weight, 578
  - win, 579
  - window, 578
- mln::doc::Window, 579
  - bkd\_qiter, 580
  - fwd\_qiter, 580
  - qiter, 580
- mln::Dpoint, 580
  - to\_dpoint, 581
- mln::dpoint, 581
  - coord, 583
  - dim, 584
  - dpoint, 584, 585
  - grid, 583
  - operator mln::algebra::vec< dpoint< G, C >::dim, Q >, 585
  - psite, 584
  - set\_all, 585
  - site, 584
  - to\_vec, 585
  - vec, 584
- mln::dpoints\_bkd\_pixter, 586
  - center\_val, 587
  - dpoints\_bkd\_pixter, 587
  - invalidate, 587
  - is\_valid, 587
  - next, 587



- start, 588
- update, 588
- mln::dpoints\_fwd\_pixter, 588
  - center\_val, 590
  - dpoints\_fwd\_pixter, 589
  - invalidate, 590
  - is\_valid, 590
  - next, 590
  - start, 590
  - update, 590
- mln::dpsites\_bkd\_piter, 590
  - dpsites\_bkd\_piter, 591
  - next, 591
- mln::dpsites\_fwd\_piter, 592
  - dpsites\_fwd\_piter, 592
  - next, 593
- mln::draw, 201
  - box, 202
  - box\_plain, 202
  - dashed\_line, 202
  - line, 203
  - plot, 203
- mln::Edge, 593
- mln::edge\_image, 593
  - edge\_image, 595
  - edge\_nbh\_t, 594
  - edge\_win\_t, 594
  - graph\_t, 594
  - nbh\_t, 594
  - operator(), 595
  - site\_function\_t, 595
  - skeleton, 595
  - win\_t, 595
- mln::estim, 203
  - mean, 204
  - min\_max, 205
  - sum, 205
- mln::extended, 595
  - domain, 597
  - extended, 596
  - skeleton, 596
  - value, 596
- mln::extension, 205
  - adjust, 206, 207
  - adjust\_duplicate, 207
  - adjust\_fill, 207
  - duplicate, 207
  - fill, 207
- mln::extension\_fun, 597
  - extension, 598
  - extension\_fun, 598
  - has, 598
  - operator(), 598, 599
  - rvalue, 598
  - skeleton, 598
  - value, 598
- mln::extension\_ima, 599
  - extension, 600
  - extension\_ima, 600
  - has, 600
  - operator(), 601
  - rvalue, 600
  - skeleton, 600
  - value, 600
- mln::extension\_val, 601
  - change\_extension, 603
  - extension, 603
  - extension\_val, 602
  - has, 603
  - operator(), 603
  - rvalue, 602
  - skeleton, 602
  - value, 602
- mln::faces\_psite, 603
  - change\_target, 604
  - face, 605
  - face\_id, 605
  - faces\_psite, 604
  - invalidate, 605
  - is\_valid, 605
  - n, 605
  - site\_set, 605
- mln::flat\_image, 606
  - domain, 607
  - flat\_image, 607
  - has, 607
  - lvalue, 607
  - operator(), 607
  - rvalue, 607
  - skeleton, 607
  - value, 607
- mln::fun, 208
- mln::fun::access, 209
- mln::fun::from\_accu, 608
- mln::fun::i2v, 209
  - operator<<, 210
- mln::fun::n2v, 210
- mln::fun::n2v::white\_gaussian, 608
- mln::fun::p2b, 210
- mln::fun::p2b::antilogy, 609
- mln::fun::p2b::tautology, 610
- mln::fun::p2p, 210
- mln::fun::p2v, 211
- mln::fun::stat, 211
- mln::fun::v2b, 211
- mln::fun::v2b::lnot, 611
- mln::fun::v2b::threshold, 612
- mln::fun::v2i, 211

- mln::fun::v2v, 212
  - f\_hsi\_to\_rgb\_3x8, 213
  - f\_hsl\_to\_rgb\_3x8, 213
  - f\_rgb\_to\_hsi\_f, 213
  - f\_rgb\_to\_hsl\_f, 213
- mln::fun::v2v::ch\_function\_value, 613
- mln::fun::v2v::component, 614
- mln::fun::v2v::l1\_norm, 615
- mln::fun::v2v::l2\_norm, 616
- mln::fun::v2v::linear, 617
- mln::fun::v2v::linfty\_norm, 618
- mln::fun::v2w2v, 213
- mln::fun::v2w2v::cos, 619
- mln::fun::v2w\_w2v, 213
- mln::fun::v2w\_w2v::l1\_norm, 620
- mln::fun::v2w\_w2v::l2\_norm, 621
- mln::fun::v2w\_w2v::linfty\_norm, 622
- mln::fun::vv2b, 214
- mln::fun::vv2b::eq, 623
- mln::fun::vv2b::ge, 624
- mln::fun::vv2b::gt, 625
- mln::fun::vv2b::implies, 626
- mln::fun::vv2b::le, 627
- mln::fun::vv2b::lt, 628
- mln::fun::vv2v, 214
- mln::fun::vv2v::diff\_abs, 629
- mln::fun::vv2v::land, 630
- mln::fun::vv2v::land\_not, 631
- mln::fun::vv2v::lor, 632
- mln::fun::vv2v::lxor, 633
- mln::fun::vv2v::max, 634
- mln::fun::vv2v::min, 635
- mln::fun::vv2v::vec, 636
- mln::fun::x2p, 215
- mln::fun::x2p::closest\_point, 637
- mln::fun::x2v, 216
- mln::fun::x2v::bilinear, 638
  - operator(), 638
- mln::fun::x2v::trilinear, 639
- mln::fun::x2x, 216
- mln::fun::x2x::composed, 639
  - composed, 639
- mln::fun::x2x::linear, 640
  - ima, 641
  - linear, 640
  - operator(), 641
- mln::fun::x2x::rotation, 641
  - data\_t, 643
  - inv, 644
  - invert, 643
  - operator(), 644
  - rotation, 643, 644
  - set\_alpha, 644
  - set\_axis, 644
- mln::fun::x2x::translation, 644
  - data\_t, 646
  - inv, 647
  - invert, 646
  - operator(), 647
  - set\_t, 647
  - t, 647
  - translation, 646
- mln::fun\_image, 647
  - fun\_image, 648, 649
  - lvalue, 648
  - operator(), 649
  - rvalue, 648
  - skeleton, 648
  - value, 648
- mln::Function, 649
  - Function, 650
- mln::Function< void >, 650
- mln::Function\_n2v, 650
- mln::Function\_v2b, 651
- mln::Function\_v2v, 652
- mln::Function\_vv2b, 653
- mln::Function\_vv2v, 653
- mln::fwd\_pixter1d, 654
  - fwd\_pixter1d, 655
  - image, 655
  - next, 655
- mln::fwd\_pixter2d, 655
  - fwd\_pixter2d, 656
  - image, 656
  - next, 656
- mln::fwd\_pixter3d, 657
  - fwd\_pixter3d, 657
  - image, 657
  - next, 658
- mln::Gdpoint, 658
- mln::Gdpoint< void >, 659
- mln::Generalized\_Pixel, 659
- mln::geom, 216
  - bbox, 221
  - chamfer, 221
  - delta, 221, 222
  - horizontal\_symmetry, 222
  - max\_col, 222
  - max\_ind, 222
  - max\_row, 222
  - max\_sli, 222
  - mesh\_corner\_point\_area, 223
  - mesh\_curvature, 223
  - mesh\_normal, 223
  - min\_col, 224
  - min\_ind, 224
  - min\_row, 224
  - min\_sli, 224

- ncols, 224, 225
- ninds, 225
- nrows, 225
- nsites, 225
- nslices, 225
- pmin\_pmax, 225, 226
- rotate, 226, 227
- seeds2tiling, 227
- seeds2tiling\_roundness, 227
- translate, 228
- vertical\_symmetry, 229
- mln::geom::complex\_geometry, 660
  - add\_location, 661
  - complex\_geometry, 661
  - operator(), 661
- mln::geom::impl, 229
  - seeds2tiling, 229
- mln::Gpoint, 661
  - operator<<, 665
  - operator+, 663
  - operator+=", 663
  - operator-, 664
  - operator-=, 664
  - operator/, 665
  - operator==, 665
- mln::Graph, 666
- mln::graph, 230
  - compute, 230
  - labeling, 230
  - to\_neighb, 231
  - to\_win, 231
- mln::graph::attribute::card\_t, 666
  - result, 667
- mln::graph::attribute::representative\_t, 667
  - result, 667
- mln::graph\_elt\_mixed\_neighborhood, 667
  - bkd\_niter, 669
  - fwd\_niter, 669
  - niter, 669
- mln::graph\_elt\_mixed\_window, 669
  - bkd\_qiter, 671
  - center\_t, 671
  - delta, 672
  - fwd\_qiter, 671
  - graph\_element, 671
  - is\_centered, 672
  - is\_empty, 672
  - is\_symmetric, 672
  - is\_valid, 672
  - psite, 671
  - qiter, 671
  - site, 671
  - sym, 672
  - target, 671
- mln::graph\_elt\_neighborhood, 672
  - bkd\_niter, 674
  - fwd\_niter, 674
  - niter, 674
- mln::graph\_elt\_neighborhood\_if, 674
  - bkd\_niter, 675
  - fwd\_niter, 675
  - graph\_elt\_neighborhood\_if, 675
  - mask, 676
  - niter, 675
- mln::graph\_elt\_window, 676
  - bkd\_qiter, 678
  - center\_t, 678
  - delta, 679
  - fwd\_qiter, 678
  - graph\_element, 678
  - is\_centered, 679
  - is\_empty, 679
  - is\_symmetric, 679
  - is\_valid, 679
  - psite, 678
  - qiter, 678
  - site, 678
  - sym, 679
  - target, 678
- mln::graph\_elt\_window\_if, 679
  - bkd\_qiter, 681
  - change\_mask, 683
  - delta, 683
  - fwd\_qiter, 681
  - graph\_elt\_window\_if, 682
  - is\_centered, 683
  - is\_empty, 683
  - is\_symmetric, 683
  - is\_valid, 683
  - mask, 683
  - mask\_t, 682
  - psite, 682
  - qiter, 682
  - site, 682
  - sym, 684
  - target, 682
- mln::graph\_window\_base, 684
  - delta, 685
  - is\_centered, 685
  - is\_empty, 685
  - is\_symmetric, 685
  - is\_valid, 685
  - site, 685
  - sym, 685
- mln::graph\_window\_if\_piter, 686
  - element, 687
  - graph\_window\_if\_piter, 687
  - id, 687

- next, 687
- P, 686
- mln::graph\_window\_piter, 687
  - center\_t, 689
  - change\_target\_site\_set, 690
  - element, 690
  - graph\_element, 689
  - graph\_window\_piter, 689
  - id, 690
  - next, 690
  - P, 689
  - target\_site\_set, 690
- mln::grid, 232
- mln::hexa, 690
  - bkd\_piter, 692
  - domain, 693
  - fwd\_piter, 692
  - has, 693
  - hexa, 693
  - lvalue, 692
  - operator(), 693
  - psite, 692
  - rvalue, 692
  - skeleton, 693
  - value, 693
- mln::histo, 232
  - compute, 233
  - equalize, 233
- mln::histo::array, 694
- mln::histo::impl, 233
- mln::histo::impl::generic, 233
- mln::Image, 694
- mln::image1d, 696
  - bbox, 698
  - border, 698
  - buffer, 698
  - delta\_index, 699
  - domain, 699
  - element, 699
  - has, 699
  - image1d, 698
  - lvalue, 697
  - nelements, 699
  - ninds, 699
  - operator(), 699, 700
  - point\_at\_index, 700
  - rvalue, 697
  - skeleton, 698
  - value, 698
- mln::image2d, 700
  - bbox, 703
  - border, 703
  - buffer, 703
  - delta\_index, 703
  - domain, 703
  - element, 703
  - has, 703
  - image2d, 702
  - lvalue, 702
  - ncols, 704
  - nelements, 704
  - nrows, 704
  - operator(), 704
  - point\_at\_index, 704
  - rvalue, 702
  - skeleton, 702
  - value, 702
- mln::image2d\_h, 704
  - bkd\_piter, 706
  - domain, 707
  - fwd\_piter, 706
  - has, 707
  - image2d\_h, 707
  - lvalue, 706
  - operator(), 707
  - psite, 706
  - rvalue, 706
  - skeleton, 706
  - value, 707
- mln::image3d, 707
  - bbox, 710
  - border, 710
  - buffer, 710
  - delta\_index, 710
  - domain, 710
  - element, 711
  - has, 711
  - image3d, 710
  - lvalue, 709
  - ncols, 711
  - nelements, 711
  - nrows, 711
  - nslices, 711
  - operator(), 711, 712
  - point\_at\_index, 712
  - rvalue, 709
  - skeleton, 709
  - value, 709
- mln::image\_if, 712
  - domain, 713
  - image\_if, 713
  - operator image\_if< const I, F >, 713
  - skeleton, 713
- mln::impl, 234
- mln::interpolated, 713
  - has, 715
  - interpolated, 715
  - is\_valid, 715

- lvalue, 714
- psite, 714
- rvalue, 714
- skeleton, 714
- value, 714
- mln::io, 234
- mln::io::cloud, 235
  - load, 236
  - save, 236
- mln::io::dicom, 236
  - get\_header, 237
  - load, 237
- mln::io::dicom::dicom\_header, 715
- mln::io::dump, 237
  - get\_header, 238
  - load, 238
  - save, 238
- mln::io::dump::dump\_header, 715
- mln::io::fits, 238
  - load, 239
- mln::io::fld, 239
  - load, 240
  - read\_header, 240
  - write\_header, 240
- mln::io::fld::fld\_header, 716
- mln::io::magick, 240
  - load, 241
  - save, 241
- mln::io::off, 241
  - load, 242
  - save, 242
  - save\_bin\_alt, 242
- mln::io::pbm, 243
  - load, 243
  - save, 243
- mln::io::pbm::impl, 244
- mln::io::pbms, 244
  - load, 244
- mln::io::pbms::impl, 245
- mln::io::pfm, 245
  - load, 245, 246
  - save, 246
- mln::io::pfm::impl, 246
- mln::io::pgm, 246
  - load, 247
  - save, 247
- mln::io::pgms, 247
  - load, 248
- mln::io::plot, 248
  - load, 249
  - save, 249
- mln::io::pnm, 250
  - load, 250, 251
  - load\_ascii\_builtin, 251
  - load\_ascii\_value, 251
  - load\_raw\_2d, 251
  - max\_component, 251
  - save, 251
- mln::io::pnm::impl, 251
- mln::io::pnms, 252
  - load, 252
- mln::io::ppm, 253
  - load, 253
  - save, 253
- mln::io::ppms, 254
  - load, 254
- mln::io::raw, 254
  - get\_header, 255
  - load, 255
  - save, 255
- mln::io::raw::raw\_header, 716
- mln::io::tiff, 256
  - load, 256
- mln::io::txt, 256
  - save, 256
- mln::Iterator, 716
  - next, 718
- mln::labeled\_image, 718
  - bbox, 720
  - bbox\_t, 720
  - bboxes, 720
  - labeled\_image, 720
  - nlabels, 721
  - relabel, 721
  - skeleton, 720
  - subdomain, 721
  - update\_data, 721
- mln::labeled\_image\_base, 721
  - bbox, 723
  - bbox\_t, 723
  - bboxes, 724
  - labeled\_image\_base, 723
  - nlabels, 724
  - relabel, 724
  - subdomain, 724
  - update\_data, 724
- mln::labeling, 257
  - background, 260
  - blobs, 260
  - blobs\_and\_compute, 261
  - colorize, 261, 262
  - compute, 262, 263
  - compute\_image, 264, 265
  - fill\_holes, 265
  - flat\_zones, 266
  - foreground, 266
  - pack, 266, 267
  - pack\_inplace, 267

- regional\_maxima, 267
- regional\_minima, 268
- relabel, 268
- relabel\_inplace, 269
- superpose, 269
- value, 270
- value\_and\_compute, 270
- wrap, 271
- mln::labeling::impl, 271
  - compute\_fastest, 272
- mln::labeling::impl::generic, 273
  - compute, 273, 274
- mln::lazy\_image, 725
  - domain, 726
  - has, 726
  - lazy\_image, 726
  - lvalue, 726
  - operator(), 727
  - rvalue, 726
  - skeleton, 726
- mln::linear, 275
  - gaussian, 276
  - gaussian\_1st\_derivative, 276, 277
  - gaussian\_2nd\_derivative, 277
  - mln\_ch\_convolve, 277, 278
  - mln\_ch\_convolve\_grad, 278
- mln::linear::impl, 279
- mln::linear::local, 279
  - convolve, 279, 280
- mln::linear::local::impl, 280
- mln::Literal, 727
- mln::literal, 280
  - black, 283
  - blue, 283
  - brown, 284
  - cyan, 284
  - dark\_gray, 284
  - green, 284
  - identity, 284
  - light\_gray, 284
  - lime, 284
  - magenta, 284
  - max, 284
  - medium\_gray, 284
  - min, 284
  - olive, 285
  - one, 285
  - orange, 285
  - origin, 285
  - pink, 285
  - purple, 285
  - red, 285
  - teal, 285
  - violet, 285
  - white, 285
  - yellow, 286
  - zero, 286
- mln::literal::black\_t, 729
- mln::literal::blue\_t, 730
- mln::literal::brown\_t, 730
- mln::literal::cyan\_t, 731
- mln::literal::green\_t, 732
- mln::literal::identity\_t, 733
- mln::literal::light\_gray\_t, 734
- mln::literal::lime\_t, 735
- mln::literal::magenta\_t, 736
- mln::literal::max\_t, 737
- mln::literal::min\_t, 738
- mln::literal::olive\_t, 739
- mln::literal::one\_t, 740
- mln::literal::orange\_t, 741
- mln::literal::origin\_t, 742
- mln::literal::pink\_t, 743
- mln::literal::purple\_t, 744
- mln::literal::red\_t, 745
- mln::literal::teal\_t, 746
- mln::literal::violet\_t, 747
- mln::literal::white\_t, 748
- mln::literal::yellow\_t, 749
- mln::literal::zero\_t, 750
- mln::logical, 286
  - and\_inplace, 287
  - and\_not, 287
  - and\_not\_inplace, 287
  - not\_inplace, 288
  - or\_inplace, 288
  - xor\_inplace, 289
- mln::logical::impl, 289
- mln::logical::impl::generic, 289
- mln::make, 290
  - attachment, 295
  - box1d, 295
  - box2d, 296
  - box2d\_h, 296, 297
  - box3d, 297, 298
  - cell, 298
  - couple, 298
  - detachment, 299
  - dpoint2d\_h, 299
  - dummy\_p\_edges, 299, 300
  - dummy\_p\_vertices, 300
  - edge\_image, 300–302
  - h\_mat, 302
  - image, 302, 303
  - image2d, 303
  - image3d, 303, 304
  - influence\_zone\_adjacency\_graph, 304
  - mat, 304

- ord\_pair, 304
- p\_edges\_with\_mass\_centers, 304
- p\_vertices\_with\_mass\_centers, 305
- pix, 305
- pixel, 306
- point2d\_h, 306
- rag\_and\_labeled\_wsl, 306
- region\_adjacency\_graph, 307
- relabelfun, 307
- vec, 308, 309
- vertex\_image, 309
- voronoi, 310
- w\_window, 310
- w\_window1d, 310
- w\_window1d\_int, 311
- w\_window2d, 311
- w\_window2d\_int, 311
- w\_window3d, 312
- w\_window3d\_int, 312
- w\_window\_directional, 312
- mln::math, 313
  - abs, 313, 314
- mln::Mesh, 751
- mln::Meta\_Accumulator, 752
- mln::Meta\_Function, 754
- mln::Meta\_Function\_v2v, 754
- mln::Meta\_Function\_vv2v, 755
- mln::metal, 314
- mln::metal::ands, 756
- mln::metal::converts\_to, 756
- mln::metal::equal, 756
- mln::metal::goes\_to, 757
- mln::metal::impl, 315
- mln::metal::is, 757
- mln::metal::is\_a, 757
- mln::metal::is\_not, 757
- mln::metal::is\_not\_a, 758
- mln::metal::math, 315
- mln::metal::math::impl, 315
- mln::mixed\_neighb, 758
  - bkd\_niter, 759
  - fwd\_niter, 759
  - mixed\_neighb, 759
  - niter, 759
- mln::morpho, 315
  - complementation, 319
  - complementation\_inplace, 319
  - contrast, 319
  - dilation, 319
  - erosion, 319
  - general, 319
  - gradient, 320
  - gradient\_external, 320
  - gradient\_internal, 320
  - hit\_or\_miss, 320
  - hit\_or\_miss\_background\_closing, 320
  - hit\_or\_miss\_background\_opening, 321
  - hit\_or\_miss\_closing, 321
  - hit\_or\_miss\_opening, 321
  - laplacian, 321
  - line\_gradient, 321
  - meyer\_wst, 321, 322
  - min, 322
  - min\_inplace, 322
  - minus, 322
  - plus, 323
  - rank\_filter, 323
  - thick\_miss, 323
  - thickening, 323
  - thin\_fit, 323
  - thinning, 324
  - top\_hat\_black, 324
  - top\_hat\_self\_complementary, 324
  - top\_hat\_white, 324
- mln::morpho::approx, 324
- mln::morpho::attribute, 325
- mln::morpho::attribute::card, 759
  - init, 760
  - is\_valid, 760
  - take\_as\_init, 760
  - take\_n\_times, 760
  - to\_result, 760
- mln::morpho::attribute::count\_adjacent\_vertices, 761
  - init, 761
  - is\_valid, 761
  - take\_as\_init, 761
  - take\_n\_times, 762
  - to\_result, 762
- mln::morpho::attribute::height, 762
  - base\_level, 763
  - init, 763
  - is\_valid, 763
  - take\_as\_init, 763
  - take\_n\_times, 763
  - to\_result, 763
- mln::morpho::attribute::sharpness, 764
  - area, 764
  - height, 764
  - init, 765
  - is\_valid, 765
  - take\_as\_init, 765
  - take\_n\_times, 765
  - to\_result, 765
  - volume, 765
- mln::morpho::attribute::sum, 765
  - init, 766
  - is\_valid, 766

- set\_value, 766
- take\_as\_init, 767
- take\_n\_times, 767
- to\_result, 767
- untake, 767
- mln::morpho::attribute::volume, 767
  - area, 768
  - init, 768
  - is\_valid, 768
  - take\_as\_init, 768
  - take\_n\_times, 768
  - to\_result, 768
- mln::morpho::closing::approx, 325
  - structural, 326
- mln::morpho::elementary, 326
  - closing, 327
  - mln\_trait\_op\_minus\_twice, 327
  - opening, 327
  - top\_hat\_black, 327
  - top\_hat\_self\_complementary, 327
  - top\_hat\_white, 328
- mln::morpho::impl, 328
- mln::morpho::impl::generic, 328
- mln::morpho::opening::approx, 328
  - structural, 329
- mln::morpho::reconstruction, 329
- mln::morpho::reconstruction::by\_dilation, 329
- mln::morpho::reconstruction::by\_erosion, 329
- mln::morpho::tree, 330
  - compute\_attribute\_image, 331
  - compute\_attribute\_image\_from, 332
  - compute\_parent, 332
  - dual\_input\_max\_tree, 333
  - max\_tree, 334
  - min\_tree, 334
  - propagate\_if, 334, 335
  - propagate\_if\_value, 335
  - propagate\_node\_to\_ancestors, 335, 336
  - propagate\_node\_to\_descendants, 336
  - propagate\_representative, 336
- mln::morpho::tree::filter, 337
  - direct, 337
  - filter, 338
  - max, 338
  - min, 338
  - subtractive, 338
- mln::morpho::watershed, 339
  - flooding, 340
  - superpose, 340, 341
  - topological, 341
- mln::morpho::watershed::watershed, 341
- mln::morpho::watershed::watershed::generic, 341
- mln::neighb, 769
  - bkd\_niter, 770
  - fwd\_niter, 770
  - neighb, 770
  - niter, 770
- mln::Neighborhood, 770
- mln::Neighborhood< void >, 771
- mln::norm, 342
  - l1, 343
  - l1\_distance, 343
  - l2, 343
  - l2\_distance, 343
  - linfty, 343
  - linfty\_distance, 343
  - sqr\_l2, 343
- mln::norm::impl, 344
- mln::Object, 771
- mln::opt, 344
  - at, 345
- mln::opt::impl, 345
- mln::p2p\_image, 771
  - domain, 773
  - fun, 773
  - operator(), 773
  - p2p\_image, 772
  - skeleton, 772
- mln::p\_array, 773
  - append, 776
  - bkd\_piter, 775
  - change, 776
  - clear, 776
  - element, 775
  - fwd\_piter, 775
  - has, 776
  - i\_element, 775
  - insert, 777
  - is\_valid, 777
  - memory\_size, 777
  - nsites, 777
  - p\_array, 776
  - piter, 775
  - psite, 775
  - reserve, 777
  - resize, 778
  - std\_vector, 778
- mln::p\_centered, 778
  - bkd\_piter, 779
  - center, 780
  - element, 779
  - fwd\_piter, 779
  - has, 780
  - is\_valid, 780
  - memory\_size, 780
  - p\_centered, 780
  - piter, 779
  - psite, 780



- site, 780
- window, 780
- mln::p\_complex, 781
  - bkd\_piter, 782
  - cplx, 783
  - element, 782
  - fwd\_piter, 782
  - geom, 783
  - has, 783
  - is\_valid, 783
  - nfaces, 784
  - nfaces\_of\_dim, 784
  - nsites, 784
  - p\_complex, 783
  - piter, 782
  - psite, 782
- mln::p\_edges, 784
  - bkd\_piter, 786
  - edge, 786
  - element, 786
  - fun\_t, 786
  - function, 788
  - fwd\_piter, 786
  - graph, 788
  - graph\_element, 786
  - graph\_t, 787
  - has, 788
  - invalidate, 788
  - is\_valid, 788
  - memory\_size, 789
  - nedges, 789
  - nsites, 789
  - p\_edges, 787
  - piter, 787
  - psite, 787
- mln::p\_faces, 789
  - bkd\_piter, 790
  - cplx, 791
  - element, 790
  - fwd\_piter, 790
  - is\_valid, 791
  - nfaces, 792
  - nsites, 792
  - p\_faces, 791
  - piter, 790
  - psite, 791
- mln::p\_graph\_piter, 792
  - graph, 793
  - id, 793
  - mln\_q\_subject, 793
  - next, 793
  - p\_graph\_piter, 793
- mln::p\_if, 793
  - bkd\_piter, 795
  - element, 795
  - fwd\_piter, 795
  - has, 795
  - is\_valid, 795
  - memory\_size, 796
  - overset, 796
  - p\_if, 795
  - piter, 795
  - pred, 796
  - predicate, 796
  - psite, 795
- mln::p\_image, 796
  - bkd\_piter, 798
  - clear, 799
  - element, 798
  - fwd\_piter, 798
  - has, 799
  - i\_element, 798
  - insert, 799
  - is\_valid, 799
  - memory\_size, 799
  - nsites, 799
  - operator typename internal::p\_image\_site\_-  
set< I >::ret, 799
  - p\_image, 798
  - piter, 798
  - psite, 798
  - r\_element, 798
  - remove, 799
  - S, 798
  - toggle, 800
- mln::p\_indexed\_bkd\_piter, 800
  - index, 801
  - next, 801
  - p\_indexed\_bkd\_piter, 800
- mln::p\_indexed\_fwd\_piter, 801
  - index, 802
  - p\_indexed\_fwd\_piter, 802
- mln::p\_indexed\_psite, 802
- mln::p\_key, 802
  - bkd\_piter, 804
  - change\_key, 805
  - change\_keys, 805
  - clear, 805
  - element, 804
  - exists\_key, 805
  - fwd\_piter, 804
  - has, 806
  - i\_element, 804
  - insert, 806
  - is\_valid, 806
  - key, 806
  - keys, 806
  - memory\_size, 806

- nsites, 806
- operator(), 807
- p\_key, 805
- piter, 805
- psite, 805
- r\_element, 805
- remove, 807
- remove\_key, 807
- mln::p\_line2d, 807
  - bbox, 809
  - begin, 809
  - bkd\_piter, 809
  - element, 809
  - end, 810
  - fwd\_piter, 809
  - has, 810
  - is\_valid, 810
  - memory\_size, 810
  - nsites, 810
  - p\_line2d, 809
  - piter, 809
  - psite, 809
  - q\_box, 809
  - std\_vector, 810
- mln::p\_mutable\_array\_of, 811
  - bkd\_piter, 812
  - clear, 813
  - element, 812
  - fwd\_piter, 812
  - has, 813
  - i\_element, 812
  - insert, 813
  - is\_valid, 813
  - memory\_size, 813
  - nelements, 813
  - p\_mutable\_array\_of, 813
  - piter, 812
  - psite, 812
  - reserve, 814
- mln::p\_n\_faces\_bkd\_piter, 814
  - n, 814
  - p\_n\_faces\_bkd\_piter, 814
- mln::p\_n\_faces\_fwd\_piter, 815
  - n, 815
  - next, 815
  - p\_n\_faces\_fwd\_piter, 815
- mln::p\_priority, 816
  - bkd\_piter, 818
  - clear, 819
  - element, 818
  - exists\_priority, 819
  - front, 819
  - fwd\_piter, 818
  - has, 819
  - highest\_priority, 819
  - i\_element, 818
  - insert, 819
  - is\_valid, 820
  - lowest\_priority, 820
  - memory\_size, 820
  - nsites, 820
  - operator(), 820
  - p\_priority, 818
  - piter, 818
  - pop, 820
  - pop\_front, 820
  - priorities, 821
  - psite, 818
  - push, 821
- mln::p\_queue, 821
  - bkd\_piter, 823
  - clear, 823
  - element, 823
  - front, 823
  - fwd\_piter, 823
  - has, 823, 824
  - i\_element, 823
  - insert, 824
  - is\_valid, 824
  - memory\_size, 824
  - nsites, 824
  - p\_queue, 823
  - piter, 823
  - pop, 824
  - pop\_front, 824
  - psite, 823
  - push, 825
  - std\_deque, 825
- mln::p\_queue\_fast, 825
  - bkd\_piter, 827
  - clear, 828
  - compute\_has, 828
  - element, 827
  - empty, 828
  - front, 828
  - fwd\_piter, 827
  - has, 828
  - i\_element, 827
  - insert, 828
  - is\_valid, 828
  - memory\_size, 829
  - nsites, 829
  - p\_queue\_fast, 828
  - piter, 827
  - pop, 829
  - pop\_front, 829
  - psite, 827
  - purge, 829

- push, 829
- reserve, 829
- std\_vector, 830
- mln::p\_run, 830
  - bbox, 832
  - bkd\_piter, 831
  - element, 831
  - end, 832
  - fwd\_piter, 832
  - has, 833
  - has\_index, 833
  - init, 833
  - is\_valid, 833
  - length, 833
  - memory\_size, 833
  - nsites, 833
  - p\_run, 832
  - piter, 832
  - psite, 832
  - q\_box, 832
  - start, 834
- mln::p\_set, 834
  - bkd\_piter, 836
  - clear, 836
  - element, 836
  - fwd\_piter, 836
  - has, 836, 837
  - i\_element, 836
  - insert, 837
  - is\_valid, 837
  - memory\_size, 837
  - nsites, 837
  - p\_set, 836
  - piter, 836
  - psite, 836
  - r\_element, 836
  - remove, 837
  - std\_vector, 837
  - util\_set, 838
- mln::p\_set\_of, 838
  - bkd\_piter, 839
  - clear, 840
  - element, 839
  - fwd\_piter, 839
  - has, 840
  - i\_element, 839
  - insert, 840
  - is\_valid, 840
  - memory\_size, 840
  - nelements, 840
  - p\_set\_of, 840
  - piter, 839
  - psite, 839
- mln::p\_transformed, 841
  - bkd\_piter, 842
  - element, 842
  - function, 843
  - fwd\_piter, 842
  - has, 843
  - is\_valid, 843
  - memory\_size, 843
  - p\_transformed, 842
  - piter, 842
  - primary\_set, 843
  - psite, 842
- mln::p\_transformed\_piter, 843
  - change\_target, 844
  - next, 844
  - p\_transformed\_piter, 844
- mln::p\_vaccess, 845
  - bkd\_piter, 846
  - element, 846
  - fwd\_piter, 846
  - has, 847
  - i\_element, 846
  - insert, 847
  - is\_valid, 847
  - memory\_size, 848
  - operator(), 848
  - p\_vaccess, 847
  - piter, 846
  - pset, 846
  - psite, 847
  - value, 847
  - values, 848
  - vset, 847
- mln::p\_vertices, 848
  - bkd\_piter, 850
  - element, 850
  - fun\_t, 850
  - function, 852
  - fwd\_piter, 850
  - graph, 852
  - graph\_element, 850
  - graph\_t, 851
  - has, 852
  - invalidate, 853
  - is\_valid, 853
  - memory\_size, 853
  - nsites, 853
  - nvertices, 853
  - operator(), 853
  - p\_vertices, 851, 852
  - piter, 851
  - psite, 851
  - vertex, 851
- mln::pixel, 853
  - change\_to, 855

- is\_valid, 855
- pixel, 854
- mln::Pixel\_Iterator, 855
  - next, 856
- mln::plain, 856
  - operator I, 857
  - operator=, 857, 858
  - plain, 857
  - skeleton, 857
- mln::Point, 858
  - operator+=, 859
  - operator-=, 860
  - operator/, 860
  - point, 859
  - to\_point, 859
- mln::point, 860
  - coord, 863
  - delta, 863
  - dim, 864
  - dpsite, 863
  - grid, 863
  - h\_vec, 863
  - last\_coord, 864
  - minus\_infty, 865
  - operator+=, 865
  - operator-=, 865
  - origin, 866
  - plus\_infty, 865
  - point, 864
  - set\_all, 865
  - to\_h\_vec, 866
  - to\_vec, 866
  - vec, 863
- mln::Point\_Site, 866
  - operator<<, 869
  - operator+, 868
  - operator-, 868, 869
  - operator==, 869
- mln::Point\_Site< void >, 870
- mln::Proxy, 870
- mln::Proxy< void >, 870
- mln::Pseudo\_Site, 871
- mln::Pseudo\_Site< void >, 871
- mln::pw, 345
- mln::pw::image, 871
  - image, 872
  - skeleton, 872
- mln::registration, 346
  - get\_rot, 347
  - icp, 347
  - registration1, 348
  - registration2, 348
  - registration3, 348
- mln::registration::closest\_point\_basic, 873
- mln::registration::closest\_point\_with\_map, 873
- mln::Regular\_Grid, 873
- mln::safe\_image, 874
  - operator safe\_image< const I >, 874
  - skeleton, 874
- mln::select, 348
- mln::select::p\_of, 875
- mln::set, 349
  - card, 350
  - compute, 350
  - compute\_with\_weights, 350
  - get, 350
  - has, 350
  - mln\_meta\_accu\_result, 350, 351
- mln::Site, 875
- mln::Site< void >, 876
- mln::Site\_Iterator, 876
  - next, 878
- mln::Site\_Proxy, 878
- mln::Site\_Proxy< void >, 878
- mln::Site\_Set, 879
  - diff, 881
  - inter, 881
  - operator<, 881
  - operator<<, 881
  - operator<=, 881
  - operator==, 881
  - sym\_diff, 882
  - uni, 882
  - unique, 882
- mln::Site\_Set< void >, 882
- mln::slice\_image, 882
  - domain, 884
  - operator slice\_image< const I >, 884
  - operator(), 884
  - skeleton, 883
  - sli, 884
  - slice\_image, 883
- mln::sub\_image, 884
  - domain, 885
  - operator sub\_image< const I, S >, 885
  - skeleton, 885
  - sub\_image, 885
- mln::sub\_image\_if, 886
  - domain, 887
  - skeleton, 886
  - sub\_image\_if, 887
- mln::subsampling, 351
  - antialiased, 352
  - gaussian\_subsampling, 352
  - subsampling, 352
- mln::tag, 352
- mln::test, 353
  - positive, 353

- predicate, 353, 354
- mln::test::impl, 354
- mln::thru\_image, 887
  - operator thru\_image< const I, F >, 887
- mln::thrubin\_image, 887
  - operator thrubin\_image< const I1, const I2, F >, 889
- psite, 888
- rvalue, 888
- skeleton, 888
- value, 888
- mln::topo, 354
  - detach, 359
  - edge, 359
  - is\_facet, 359
  - make\_algebraic\_face, 360
  - operator<, 361, 362
  - operator<<, 362
  - operator+, 360
  - operator-, 361
  - operator==, 362, 363
- mln::topo::adj\_higher\_dim\_connected\_n\_face\_-bkd\_iter, 889
  - adj\_higher\_dim\_connected\_n\_face\_bkd\_iter, 890
  - next, 890
- mln::topo::adj\_higher\_dim\_connected\_n\_face\_-fwd\_iter, 890
  - adj\_higher\_dim\_connected\_n\_face\_fwd\_iter, 891
  - next, 891
- mln::topo::adj\_higher\_face\_bkd\_iter, 891
  - adj\_higher\_face\_bkd\_iter, 892
  - next, 892
- mln::topo::adj\_higher\_face\_fwd\_iter, 892
  - adj\_higher\_face\_fwd\_iter, 893
  - next, 893
- mln::topo::adj\_lower\_dim\_connected\_n\_face\_-bkd\_iter, 893
  - adj\_lower\_dim\_connected\_n\_face\_bkd\_iter, 894
  - next, 894
- mln::topo::adj\_lower\_dim\_connected\_n\_face\_-fwd\_iter, 894
  - adj\_lower\_dim\_connected\_n\_face\_fwd\_iter, 895
  - next, 895
- mln::topo::adj\_lower\_face\_bkd\_iter, 895
  - adj\_lower\_face\_bkd\_iter, 896
  - next, 896
- mln::topo::adj\_lower\_face\_fwd\_iter, 896
  - adj\_lower\_face\_fwd\_iter, 897
  - next, 897
- mln::topo::adj\_lower\_higher\_face\_bkd\_iter, 897
  - adj\_lower\_higher\_face\_bkd\_iter, 898
  - next, 898
- mln::topo::adj\_lower\_higher\_face\_fwd\_iter, 898
  - adj\_lower\_higher\_face\_fwd\_iter, 899
  - next, 899
- mln::topo::adj\_m\_face\_bkd\_iter, 899
  - adj\_m\_face\_bkd\_iter, 900
  - next, 900
- mln::topo::adj\_m\_face\_fwd\_iter, 901
  - adj\_m\_face\_fwd\_iter, 901
  - next, 902
- mln::topo::algebraic\_face, 902
  - algebraic\_face, 904
  - cplx, 904
  - data, 904
  - dec\_face\_id, 905
  - dec\_n, 905
  - face\_id, 905
  - higher\_dim\_adj\_faces, 905
  - inc\_face\_id, 905
  - inc\_n, 905
  - invalidate, 905
  - is\_valid, 905
  - lower\_dim\_adj\_faces, 906
  - n, 906
  - set\_cplx, 906
  - set\_face\_id, 906
  - set\_n, 906
  - set\_sign, 906
  - sign, 906
- mln::topo::algebraic\_n\_face, 907
  - algebraic\_n\_face, 908
  - cplx, 909
  - data, 909
  - dec\_face\_id, 909
  - face\_id, 909
  - higher\_dim\_adj\_faces, 909
  - inc\_face\_id, 909
  - invalidate, 909
  - is\_valid, 910
  - lower\_dim\_adj\_faces, 910
  - n, 910
  - set\_cplx, 910
  - set\_face\_id, 910
  - set\_sign, 910
  - sign, 910
- mln::topo::center\_only\_iter, 911
  - center\_only\_iter, 911
  - next, 912
- mln::topo::centered\_bkd\_iter\_adapter, 912
  - centered\_bkd\_iter\_adapter, 912
  - next, 913
- mln::topo::centered\_fwd\_iter\_adapter, 913
  - centered\_fwd\_iter\_adapter, 913

- next, 914
- mln::topo::complex, 914
  - add\_face, 915
  - addr, 916
  - bkd\_citer, 915
  - complex, 915
  - fwd\_citer, 915
  - nfaces, 916
  - nfaces\_of\_dim, 916
  - nfaces\_of\_static\_dim, 916
  - print, 916
  - print\_faces, 916
- mln::topo::face, 917
  - cplx, 919
  - data, 919
  - dec\_face\_id, 919
  - dec\_n, 919
  - face, 918
  - face\_id, 919
  - higher\_dim\_adj\_faces, 919
  - inc\_face\_id, 919
  - inc\_n, 919
  - invalidate, 920
  - is\_valid, 920
  - lower\_dim\_adj\_faces, 920
  - n, 920
  - set\_cplx, 920
  - set\_face\_id, 920
  - set\_n, 920
- mln::topo::face\_bkd\_iter, 920
  - face\_bkd\_iter, 921
  - next, 921
  - start, 921
- mln::topo::face\_fwd\_iter, 922
  - face\_fwd\_iter, 922
  - next, 922
  - start, 923
- mln::topo::is\_n\_face, 923
- mln::topo::is\_simple\_cell, 924
  - D, 927
  - mln\_geom, 926
  - operator(), 926
  - psite, 926
  - result, 926
  - set\_image, 926
- mln::topo::n\_face, 927
  - cplx, 929
  - data, 929
  - dec\_face\_id, 929
  - face\_id, 929
  - higher\_dim\_adj\_faces, 929
  - inc\_face\_id, 929
  - invalidate, 929
  - is\_valid, 929
  - lower\_dim\_adj\_faces, 930
  - n, 930
  - n\_face, 928
  - set\_cplx, 930
  - set\_face\_id, 930
- mln::topo::n\_face\_bkd\_iter, 930
  - n, 931
  - n\_face\_bkd\_iter, 931
  - next, 931
  - start, 931
- mln::topo::n\_face\_fwd\_iter, 932
  - n, 933
  - n\_face\_fwd\_iter, 932
  - next, 933
  - start, 933
- mln::topo::n\_faces\_set, 933
  - add, 934
  - faces, 934
  - faces\_type, 934
  - reserve, 934
- mln::topo::static\_n\_face\_bkd\_iter, 934
  - next, 935
  - start, 935
  - static\_n\_face\_bkd\_iter, 935
- mln::topo::static\_n\_face\_fwd\_iter, 936
  - next, 936
  - start, 937
  - static\_n\_face\_fwd\_iter, 936
- mln::tr\_image, 937
  - domain, 939
  - has, 939
  - is\_valid, 939
  - lvalue, 938
  - operator(), 939
  - psite, 938
  - rvalue, 938
  - set\_tr, 939
  - site, 938
  - skeleton, 938
  - tr, 939
  - tr\_image, 939
  - value, 938
- mln::trace, 364
- mln::trait, 364
- mln::transform, 364
  - distance\_and\_closest\_point\_geodesic, 365, 366
  - distance\_and\_influence\_zone\_geodesic, 366
  - distance\_front, 367
  - distance\_geodesic, 367
  - hough, 367
  - influence\_zone\_front, 367
  - influence\_zone\_geodesic, 368
  - influence\_zone\_geodesic\_saturated, 368

- mln::transformed\_image, 940
  - domain, 941
  - operator transformed\_image< const I, F >, 941
  - operator(), 941
  - skeleton, 940
  - transformed\_image, 941
- mln::unproject\_image, 941
  - domain, 942
  - operator(), 942
  - unproject\_image, 942
- mln::util, 369
  - display\_branch, 372
  - display\_tree, 372
  - lemmings, 373
  - make\_greater\_point, 373
  - make\_greater\_psite, 373
  - operator<, 373
  - operator<<, 373
  - operator==, 374
  - ord\_strict, 374
  - ord\_weak, 374
  - tree\_fast\_to\_image, 374
  - tree\_to\_fast, 374
  - tree\_to\_image, 375
  - vertex\_id\_t, 372
- mln::util::adjacency\_matrix, 943
  - adjacency\_matrix, 943
- mln::util::array, 943
  - append, 947
  - array, 946
  - bkd\_eiter, 946
  - clear, 947
  - eiter, 946
  - element, 946
  - fill, 947
  - fwd\_eiter, 946
  - is\_empty, 947
  - memory\_size, 947
  - nelements, 948
  - operator(), 948
  - reserve, 949
  - resize, 949
  - result, 946
  - size, 949
  - std\_vector, 949
- mln::util::branch, 949
  - apex, 950
  - branch, 950
  - util\_tree, 950
- mln::util::branch\_iter, 951
  - deepness, 951
  - invalidate, 951
  - is\_valid, 951
  - next, 952
  - operator util::tree\_node< T > &, 952
  - start, 952
- mln::util::branch\_iter\_ind, 952
  - deepness, 953
  - invalidate, 953
  - is\_valid, 953
  - next, 953
  - operator util::tree\_node< T > &, 953
  - start, 953
- mln::util::couple, 953
  - change\_both, 955
  - change\_first, 955
  - change\_second, 955
  - first, 955
  - second, 955
- mln::util::eat, 955
- mln::util::edge, 956
  - category, 958
  - change\_graph, 958
  - edge, 958
  - graph, 958
  - graph\_t, 958
  - id, 958
  - id\_t, 958
  - id\_value\_t, 958
  - invalidate, 958
  - is\_valid, 959
  - ith\_nbh\_edge, 959
  - nmax\_nbh\_edges, 959
  - operator edge\_id\_t, 959
  - update\_id, 959
  - v1, 959
  - v2, 959
  - v\_other, 959
- mln::util::fibonacci\_heap, 960
  - clear, 961
  - fibonacci\_heap, 961
  - front, 961
  - is\_empty, 961
  - is\_valid, 962
  - nelements, 962
  - operator=, 962
  - pop\_front, 962
  - push, 962
- mln::util::graph, 963
  - add\_edge, 966
  - add\_vertex, 966
  - add\_vertices, 967
  - e\_ith\_nbh\_edge, 967
  - e\_nmax, 967
  - e\_nmax\_nbh\_edges, 967
  - edge, 967
  - edge\_fwd\_iter, 965

- edge\_nbh\_edge\_fwd\_iter, 965
- edges, 967
- edges\_set\_t, 965
- edges\_t, 965
- graph, 966
- has\_e, 968
- has\_v, 968
- is\_subgraph\_of, 968
- v1, 968
- v2, 968
- v\_ith\_nbh\_edge, 968
- v\_ith\_nbh\_vertex, 968
- v\_nmax, 968
- v\_nmax\_nbh\_edges, 969
- v\_nmax\_nbh\_vertices, 969
- vertex, 969
- vertex\_fwd\_iter, 965
- vertex\_nbh\_edge\_fwd\_iter, 965
- vertex\_nbh\_vertex\_fwd\_iter, 966
- vertices\_t, 966
- mln::util::greater\_point, 969
- operator(), 970
- mln::util::greater\_psite, 970
- operator(), 970
- mln::util::head, 970
- mln::util::ignore, 971
- mln::util::ilcell, 971
- mln::util::impl, 375
- mln::util::line\_graph, 971
- e\_ith\_nbh\_edge, 974
- e\_nmax, 974
- e\_nmax\_nbh\_edges, 975
- edge, 975
- edge\_fwd\_iter, 974
- edge\_nbh\_edge\_fwd\_iter, 974
- edges\_t, 974
- graph, 975
- has, 975
- has\_e, 975
- has\_v, 976
- is\_subgraph\_of, 976
- v1, 976
- v2, 976
- v\_ith\_nbh\_edge, 976
- v\_ith\_nbh\_vertex, 976
- v\_nmax, 977
- v\_nmax\_nbh\_edges, 977
- v\_nmax\_nbh\_vertices, 977
- vertex, 977
- vertex\_fwd\_iter, 974
- vertex\_nbh\_edge\_fwd\_iter, 974
- vertex\_nbh\_vertex\_fwd\_iter, 974
- vertices\_t, 974
- mln::util::nil, 977
- mln::util::node, 978
- mln::util::object\_id, 978
- object\_id, 980
- value\_t, 980
- mln::util::ord, 980
- mln::util::ord\_pair, 980
- change\_both, 982
- change\_first, 982
- change\_second, 982
- first, 982
- second, 982
- mln::util::pix, 982
- ima, 984
- p, 984
- pix, 983
- psite, 983
- v, 984
- value, 983
- mln::util::set, 984
- bkd\_eiter, 987
- clear, 987
- eiter, 987
- element, 987
- first\_element, 987
- fwd\_eiter, 987
- has, 987
- insert, 988
- is\_empty, 988
- last\_element, 988
- memory\_size, 989
- nelements, 989
- remove, 989
- set, 987
- std\_vector, 989
- mln::util::site\_pair, 990
- first, 991
- pair, 991
- second, 991
- mln::util::soft\_heap, 991
- ~soft\_heap, 993
- clear, 993
- element, 993
- is\_empty, 993
- is\_valid, 993
- nelements, 994
- pop\_front, 994
- push, 994
- soft\_heap, 993
- mln::util::timer, 994
- mln::util::tracked\_ptr, 995
- ~tracked\_ptr, 996
- operator bool, 996
- operator->, 997
- operator=, 997



- tracked\_ptr, 996
- mln::util::tree, 997
  - add\_tree\_down, 998
  - add\_tree\_up, 998
  - check\_consistency, 999
  - main\_branch, 999
  - root, 999
  - tree, 998
- mln::util::tree\_node, 999
  - add\_child, 1001
  - check\_consistency, 1001
  - children, 1002
  - delete\_tree\_node, 1002
  - elt, 1002
  - parent, 1002
  - print, 1003
  - search, 1003
  - search\_rec, 1003
  - set\_parent, 1003
  - tree\_node, 1001
- mln::util::vertex, 1004
  - Category, 1006
  - change\_graph, 1006
  - edge\_with, 1006
  - graph, 1006
  - graph\_t, 1006
  - id, 1006
  - id\_t, 1006
  - id\_value\_t, 1006
  - invalidate, 1007
  - is\_valid, 1007
  - ith\_nbh\_edge, 1007
  - ith\_nbh\_vertex, 1007
  - nmax\_nbh\_edges, 1007
  - nmax\_nbh\_vertices, 1007
  - operator vertex\_id\_t, 1007
  - other, 1007
  - update\_id, 1008
  - vertex, 1006
- mln::util::yes, 1008
- mln::Value, 1008
- mln::value, 375
  - cast, 382
  - equiv, 382
  - float01\_16, 380
  - float01\_8, 380
  - gl16, 380
  - gl8, 380
  - glf, 380
  - int\_s16, 380
  - int\_s32, 380
  - int\_s8, 381
  - int\_u12, 381
  - int\_u16, 381
  - int\_u32, 381
  - int\_u8, 381
  - label\_16, 381
  - label\_32, 381
  - label\_8, 381
  - operator<<, 383–385
  - operator\*, 382
  - operator+, 382
  - operator-, 382
  - operator/, 382, 383
  - operator==, 385
  - other, 385
  - rgb16, 381
  - rgb8, 381
  - stack, 385
- mln::value::float01, 1010
  - enc, 1011
  - equiv, 1011
  - float01, 1011
  - nbits, 1011
  - operator float, 1011
  - set\_nbits, 1011
  - to\_nbits, 1011
  - value, 1011
  - value\_ind, 1012
- mln::value::float01\_f, 1012
  - float01\_f, 1012
  - operator float, 1013
  - operator=, 1013
  - value, 1013
- mln::value::graylevel, 1013
  - graylevel, 1014, 1015
  - operator=, 1015
  - to\_float, 1015
  - value, 1015
- mln::value::graylevel\_f, 1016
  - graylevel\_f, 1017
  - operator graylevel< n >, 1017
  - operator=, 1017, 1018
  - value, 1018
- mln::value::impl, 386
- mln::value::int\_s, 1018
  - int\_s, 1019
  - one, 1020
  - operator int, 1020
  - operator=, 1020
  - zero, 1020
- mln::value::int\_u, 1020
  - int\_u, 1021
  - next, 1022
  - operator unsigned, 1022
  - operator-, 1022
  - operator=, 1022
- mln::value::int\_u\_sat, 1022

- int\_u\_sat, 1023
- one, 1024
- operator int, 1024
- operator+=, 1024
- operator-=, 1024
- operator=, 1024
- zero, 1024
- mln::value::Integer, 1024
- mln::value::Integer< void >, 1025
- mln::value::label, 1025
  - enc, 1026
  - label, 1027
  - next, 1027
  - operator unsigned, 1027
  - operator++, 1027
  - operator--, 1027
  - operator=, 1027
  - prev, 1027
- mln::value::lut\_vec, 1028
  - bkd\_viter, 1029
  - fwd\_viter, 1029
  - has, 1030
  - index\_of, 1030
  - lut\_vec, 1030
  - nvalues, 1030
  - value, 1029
- mln::value::proxy, 1031
  - ~proxy, 1032
  - enc, 1032
  - equiv, 1032
  - operator=, 1033
  - proxy, 1032
  - to\_value, 1033
- mln::value::qt::rgb32, 1033
  - operator=, 1034
  - red, 1034
  - rgb32, 1034
  - zero, 1034
- mln::value::rgb, 1035
  - operator=, 1036
  - red, 1036
  - rgb, 1035, 1036
  - zero, 1036
- mln::value::set, 1036
  - the, 1037
- mln::value::sign, 1037
  - enc, 1038
  - equiv, 1038
  - one, 1039
  - operator int, 1038
  - operator=, 1038
  - sign, 1038
  - zero, 1039
- mln::value::stack\_image, 1039
  - domain\_t, 1040
  - is\_valid, 1041
  - lvalue, 1040
  - operator(), 1041
  - psite, 1040
  - rvalue, 1040
  - skeleton, 1040
  - stack\_image, 1041
  - value, 1040
- mln::value::super\_value< sign >, 1041
- mln::value::value\_array, 1041
  - operator(), 1042
  - value\_array, 1042
  - vset, 1042
- mln::Value\_Iterator, 1043
  - next, 1044
  - operator<<, 1044
- mln::Value\_Set, 1044
- mln::Vertex, 1045
- mln::vertex\_image, 1046
  - graph\_t, 1047
  - nbh\_t, 1047
  - operator(), 1048
  - site\_function\_t, 1047
  - skeleton, 1047
  - vertex\_image, 1047
  - vertex\_nbh\_t, 1047
  - vertex\_win\_t, 1047
  - win\_t, 1047
- mln::violent\_cast\_image, 1048
  - lvalue, 1049
  - operator(), 1049
  - rvalue, 1049
  - skeleton, 1049
  - value, 1049
  - violent\_cast\_image, 1049
- mln::w\_window, 1049
  - bkd\_qiter, 1051
  - clear, 1052
  - dpsite, 1051
  - fwd\_qiter, 1051
  - insert, 1052
  - is\_symmetric, 1052
  - operator<<, 1053
  - operator==, 1053
  - std\_vector, 1052
  - sym, 1052
  - w, 1052
  - w\_window, 1051
  - weight, 1051
  - weights, 1052
  - win, 1052
- mln::Weighted\_Window, 1053
  - operator-, 1054

- mln::win, 386
  - diff, 388
  - mln\_regular, 388
  - sym, 388
- mln::win::backdiag2d, 1054
  - backdiag2d, 1055
  - length, 1055
- mln::win::ball, 1055
  - ball, 1056
  - diameter, 1056
- mln::win::cube3d, 1056
  - cube3d, 1057
  - length, 1057
- mln::win::cuboid3d, 1057
  - cuboid3d, 1058
  - depth, 1058
  - height, 1058
  - volume, 1059
  - width, 1059
- mln::win::diag2d, 1059
  - diag2d, 1059
  - length, 1060
- mln::win::line, 1060
  - length, 1061
  - line, 1061
  - size, 1061
- mln::win::multiple, 1061
- mln::win::multiple\_size, 1062
- mln::win::octagon2d, 1062
  - area, 1063
  - length, 1063
  - octagon2d, 1063
- mln::win::rectangle2d, 1063
  - area, 1064
  - height, 1064
  - rectangle2d, 1064
  - std\_vector, 1064
  - width, 1064
- mln::Window, 1065
- mln::window, 1065
  - bkd\_qiter, 1067
  - clear, 1067
  - delta, 1067
  - dp, 1068
  - fwd\_qiter, 1067
  - has, 1068
  - insert, 1068
  - is\_centered, 1068
  - is\_empty, 1068
  - is\_symmetric, 1069
  - operator==, 1069
  - print, 1069
  - qiter, 1067
  - regular, 1067
    - size, 1069
    - std\_vector, 1069
    - sym, 1069
    - window, 1067
- mln::world::inter\_pixel::is\_separator, 1069
- mln\_ch\_convolve
  - mln::linear, 277, 278
- mln\_ch\_convolve\_grad
  - mln::linear, 278
- mln\_exact
  - mln, 126
- mln\_gen\_complex\_neighborhood
  - mln, 126, 127
- mln\_gen\_complex\_window
  - mln, 127
- mln\_gen\_complex\_window\_p
  - mln, 127, 128
- mln\_geom
  - mln::topo::is\_simple\_cell, 926
- mln\_image\_from\_grid
  - mln::convert, 170
- mln\_meta\_accu\_result
  - mln::accu, 138
  - mln::data, 178
  - mln::set, 350, 351
- mln\_q\_subject
  - mln::p\_graph\_piter, 793
- mln\_regular
  - mln, 128
  - mln::win, 388
- mln\_trait\_op\_geq
  - mln, 128
- mln\_trait\_op\_greater
  - mln, 128
- mln\_trait\_op\_leq
  - mln, 129
- mln\_trait\_op\_minus\_twice
  - mln::morpho::elementary, 327
- mln\_trait\_op\_neq
  - mln, 129
- mln\_window
  - mln::convert, 170
- modneighb1d
  - c2, 78
  - neighb1d, 78
- modneighb2d
  - c2\_col, 79
  - c2\_row, 79
  - c4, 79
  - c8, 79
  - neighb2d, 79
- modneighb3d
  - c18, 81
  - c26, 81

- c2\_3d\_sli, 81
- c4\_3d, 82
- c6, 82
- c8\_3d, 83
- neighb3d, 81
- modwin1d
  - segment1d, 88
  - window1d, 88
- modwin2d
  - disk2d, 90
  - hline2d, 90
  - vline2d, 90
  - win\_c4p, 90
  - win\_c8p, 91
  - window2d, 90
- modwin3d
  - sline3d, 92
  - sphere3d, 92
  - win\_c4p\_3d, 92
  - win\_c8p\_3d, 93
  - window3d, 92
- mosaic
  - mln::debug, 196
- Multiple accumulators, 70
- Multiple windows, 94
- n
  - mln::complex\_psite, 532
  - mln::faces\_psite, 605
  - mln::p\_n\_faces\_bkd\_piter, 814
  - mln::p\_n\_faces\_fwd\_piter, 815
  - mln::topo::algebraic\_face, 906
  - mln::topo::algebraic\_n\_face, 910
  - mln::topo::face, 920
  - mln::topo::n\_face, 930
  - mln::topo::n\_face\_bkd\_iter, 931
  - mln::topo::n\_face\_fwd\_iter, 933
- N-D windows, 93
- n\_face
  - mln::topo::n\_face, 928
- n\_face\_bkd\_iter
  - mln::topo::n\_face\_bkd\_iter, 931
- n\_face\_fwd\_iter
  - mln::topo::n\_face\_fwd\_iter, 932
- n\_items
  - mln::accu::stat::var, 479
  - mln::accu::stat::variance, 482
- nbh\_t
  - mln::edge\_image, 594
  - mln::vertex\_image, 1047
- nbits
  - mln::value::float01, 1011
- ncols
  - mln::geom, 224, 225
  - mln::image2d, 704
  - mln::image3d, 711
- nedges
  - mln::p\_edges, 789
- neighb
  - mln::neighb, 770
- neighb1d
  - modneighb1d, 78
- neighb2d
  - modneighb2d, 79
- neighb3d
  - modneighb3d, 81
- Neighborhoods, 77
- nelements
  - mln::doc::Fastest\_Image, 550
  - mln::image1d, 699
  - mln::image2d, 704
  - mln::image3d, 711
  - mln::p\_mutable\_array\_of, 813
  - mln::p\_set\_of, 840
  - mln::util::array, 948
  - mln::util::fibonacci\_heap, 962
  - mln::util::set, 989
  - mln::util::soft\_heap, 994
- next
  - mln::bkd\_pixter1d, 492
  - mln::bkd\_pixter2d, 493
  - mln::bkd\_pixter3d, 494
  - mln::box\_runend\_piter, 507
  - mln::box\_runstart\_piter, 508
  - mln::complex\_neighborhood\_bkd\_piter, 528
  - mln::complex\_neighborhood\_fwd\_piter, 529
  - mln::complex\_window\_bkd\_piter, 533
  - mln::complex\_window\_fwd\_piter, 535
  - mln::dpoints\_bkd\_pixter, 587
  - mln::dpoints\_fwd\_pixter, 590
  - mln::dpsites\_bkd\_piter, 591
  - mln::dpsites\_fwd\_piter, 593
  - mln::fwd\_pixter1d, 655
  - mln::fwd\_pixter2d, 656
  - mln::fwd\_pixter3d, 658
  - mln::graph\_window\_if\_piter, 687
  - mln::graph\_window\_piter, 690
  - mln::Iterator, 718
  - mln::p\_graph\_piter, 793
  - mln::p\_indexed\_bkd\_piter, 801
  - mln::p\_n\_faces\_fwd\_piter, 815
  - mln::p\_transformed\_piter, 844
  - mln::Pixel\_Iterator, 856
  - mln::Site\_Iterator, 878
  - mln::topo::adj\_higher\_dim\_connected\_n\_-  
face\_bkd\_iter, 890
  - mln::topo::adj\_higher\_dim\_connected\_n\_-  
face\_fwd\_iter, 891

- mln::topo::adj\_higher\_face\_bkd\_iter, 892
- mln::topo::adj\_higher\_face\_fwd\_iter, 893
- mln::topo::adj\_lower\_dim\_connected\_n\_-  
face\_bkd\_iter, 894
- mln::topo::adj\_lower\_dim\_connected\_n\_-  
face\_fwd\_iter, 895
- mln::topo::adj\_lower\_face\_bkd\_iter, 896
- mln::topo::adj\_lower\_face\_fwd\_iter, 897
- mln::topo::adj\_lower\_higher\_face\_bkd\_iter,  
898
- mln::topo::adj\_lower\_higher\_face\_fwd\_iter,  
899
- mln::topo::adj\_m\_face\_bkd\_iter, 900
- mln::topo::adj\_m\_face\_fwd\_iter, 902
- mln::topo::center\_only\_iter, 912
- mln::topo::centered\_bkd\_iter\_adapter, 913
- mln::topo::centered\_fwd\_iter\_adapter, 914
- mln::topo::face\_bkd\_iter, 921
- mln::topo::face\_fwd\_iter, 922
- mln::topo::n\_face\_bkd\_iter, 931
- mln::topo::n\_face\_fwd\_iter, 933
- mln::topo::static\_n\_face\_bkd\_iter, 935
- mln::topo::static\_n\_face\_fwd\_iter, 936
- mln::util::branch\_iter, 952
- mln::util::branch\_iter\_ind, 953
- mln::value::int\_u, 1022
- mln::value::label, 1027
- mln::Value\_Iterator, 1044
- nfaces
  - mln::p\_complex, 784
  - mln::p\_faces, 792
  - mln::topo::complex, 916
- nfaces\_of\_dim
  - mln::p\_complex, 784
  - mln::topo::complex, 916
- nfaces\_of\_static\_dim
  - mln::topo::complex, 916
- ninds
  - mln::geom, 225
  - mln::image1d, 699
- niter
  - mln::doc::Neighborhood, 563
  - mln::graph\_elt\_mixed\_neighborhood, 669
  - mln::graph\_elt\_neighborhood, 674
  - mln::graph\_elt\_neighborhood\_if, 675
  - mln::mixed\_neighb, 759
  - mln::neighb, 770
- nlabels
  - mln::labeled\_image, 721
  - mln::labeled\_image\_base, 724
- nmax\_nbh\_edges
  - mln::util::edge, 959
  - mln::util::vertex, 1007
- nmax\_nbh\_vertices
  - mln::util::vertex, 1007
- not\_inplace
  - mln::logical, 288
- nrows
  - mln::geom, 225
  - mln::image2d, 704
  - mln::image3d, 711
- nsites
  - mln::accu::center, 390
  - mln::Box, 504
  - mln::box, 500
  - mln::doc::Box, 542
  - mln::doc::Fastest\_Image, 551
  - mln::doc::Image, 559
  - mln::geom, 225
  - mln::p\_array, 777
  - mln::p\_complex, 784
  - mln::p\_edges, 789
  - mln::p\_faces, 792
  - mln::p\_image, 799
  - mln::p\_key, 806
  - mln::p\_line2d, 810
  - mln::p\_priority, 820
  - mln::p\_queue, 824
  - mln::p\_queue\_fast, 829
  - mln::p\_run, 833
  - mln::p\_set, 837
  - mln::p\_vertices, 853
- nslices
  - mln::geom, 225
  - mln::image3d, 711
- nvalues
  - mln::doc::Value\_Set, 576
  - mln::value::lut\_vec, 1030
- nvertices
  - mln::p\_vertices, 853
- object\_id
  - mln::util::object\_id, 980
- octagon2d
  - mln::win::octagon2d, 1063
- olive
  - mln::literal, 285
- On images, 67
- On site sets, 67
- On values, 68
- one
  - mln::literal, 285
  - mln::value::int\_s, 1020
  - mln::value::int\_u\_sat, 1024
  - mln::value::sign, 1039
- opening
  - mln::morpho::elementary, 327
- operator bool

- mln::util::tracked\_ptr, 996
- operator decorated\_image< const I, D >
  - mln::decorated\_image, 537
- operator edge\_id\_t
  - mln::util::edge, 959
- operator float
  - mln::value::float01, 1011
  - mln::value::float01\_f, 1013
- operator graylevel< n >
  - mln::value::graylevel\_f, 1017
- operator I
  - mln::plain, 857
- operator image\_if< const I, F >
  - mln::image\_if, 713
- operator int
  - mln::value::int\_s, 1020
  - mln::value::int\_u\_sat, 1024
  - mln::value::sign, 1038
- operator mat< n, I, U >
  - mln::algebra::h\_vec, 490
- operator mln::algebra::vec< dpoint< G, C >::dim, Q >
  - mln::dpoint, 585
- operator psite
  - mln::doc::Site\_Iterator, 570
- operator safe\_image< const I >
  - mln::safe\_image, 874
- operator slice\_image< const I >
  - mln::slice\_image, 884
- operator sub\_image< const I, S >
  - mln::sub\_image, 885
- operator thru\_image< const I, F >
  - mln::thru\_image, 887
- operator thrubin\_image< const I1, const I2, F >
  - mln::thrubin\_image, 889
- operator transformed\_image< const I, F >
  - mln::transformed\_image, 941
- operator typename internal::p\_image\_site\_set< I >::ret
  - mln::p\_image, 799
- operator unsigned
  - mln::value::int\_u, 1022
  - mln::value::label, 1027
- operator util::tree\_node< T > &
  - mln::util::branch\_iter, 952
  - mln::util::branch\_iter\_ind, 953
- operator value
  - mln::doc::Value\_Iterator, 574
- operator vertex\_id\_t
  - mln::util::vertex, 1007
- operator<
  - mln, 131
  - mln::Box, 505
  - mln::Site\_Set, 881
  - mln::topo, 361, 362
  - mln::util, 373
- operator<<
  - mln, 131, 132
  - mln::Box, 505
  - mln::box, 501
  - mln::fun::i2v, 210
  - mln::Gpoint, 665
  - mln::Point\_Site, 869
  - mln::Site\_Set, 881
  - mln::topo, 362
  - mln::util, 373
  - mln::value, 383–385
  - mln::Value\_Iterator, 1044
  - mln::w\_window, 1053
- operator<=
  - mln, 132
  - mln::Box, 505
  - mln::Site\_Set, 881
- operator\*
  - mln, 130
  - mln::algebra, 146
  - mln::value, 382
- operator()
  - mln::complex\_image, 526
  - mln::decorated\_image, 538
  - mln::doc::Fastest\_Image, 551
  - mln::doc::Image, 559
  - mln::edge\_image, 595
  - mln::extension\_fun, 598, 599
  - mln::extension\_ima, 601
  - mln::extension\_val, 603
  - mln::flat\_image, 607
  - mln::fun::x2v::bilinear, 638
  - mln::fun::x2x::linear, 641
  - mln::fun::x2x::rotation, 644
  - mln::fun::x2x::translation, 647
  - mln::fun\_image, 649
  - mln::geom::complex\_geometry, 661
  - mln::hexa, 693
  - mln::image1d, 699, 700
  - mln::image2d, 704
  - mln::image2d\_h, 707
  - mln::image3d, 711, 712
  - mln::lazy\_image, 727
  - mln::p2p\_image, 773
  - mln::p\_key, 807
  - mln::p\_priority, 820
  - mln::p\_vaccess, 848
  - mln::p\_vertices, 853
  - mln::slice\_image, 884
  - mln::topo::is\_simple\_cell, 926
  - mln::tr\_image, 939
  - mln::transformed\_image, 941

- mln::unproject\_image, 942
- mln::util::array, 948
- mln::util::greater\_point, 970
- mln::util::greater\_psite, 970
- mln::value::stack\_image, 1041
- mln::value::value\_array, 1042
- mln::vertex\_image, 1048
- mln::violent\_cast\_image, 1049
- operator+
  - mln::Gpoint, 663
  - mln::Point\_Site, 868
  - mln::topo, 360
  - mln::value, 382
- operator++
  - mln, 130
  - mln::value::label, 1027
- operator+=
  - mln::Gpoint, 663
  - mln::Point, 859
  - mln::point, 865
  - mln::value::int\_u\_sat, 1024
- operator-
  - mln, 130
  - mln::Gpoint, 664
  - mln::Point\_Site, 868, 869
  - mln::topo, 361
  - mln::value, 382
  - mln::value::int\_u, 1022
  - mln::Weighted\_Window, 1054
- operator->
  - mln::util::tracked\_ptr, 997
- operator--
  - mln, 131
  - mln::value::label, 1027
- operator-=
  - mln::Gpoint, 664
  - mln::Point, 860
  - mln::point, 865
  - mln::value::int\_u\_sat, 1024
- operator/
  - mln::Gpoint, 665
  - mln::Point, 860
  - mln::value, 382, 383
- operator=
  - mln::plain, 857, 858
  - mln::util::fibonacci\_heap, 962
  - mln::util::tracked\_ptr, 997
  - mln::value::float01\_f, 1013
  - mln::value::graylevel, 1015
  - mln::value::graylevel\_f, 1017, 1018
  - mln::value::int\_s, 1020
  - mln::value::int\_u, 1022
  - mln::value::int\_u\_sat, 1024
  - mln::value::label, 1027
  - mln::value::proxy, 1033
  - mln::value::qt::rgb32, 1034
  - mln::value::rgb, 1036
  - mln::value::sign, 1038
- operator==
  - mln, 133, 134
  - mln::Box, 506
  - mln::Gpoint, 665
  - mln::Point\_Site, 869
  - mln::Site\_Set, 881
  - mln::topo, 362, 363
  - mln::util, 374
  - mln::value, 385
  - mln::w\_window, 1053
  - mln::window, 1069
- or\_inplace
  - mln::logical, 288
- orange
  - mln::literal, 285
- ord\_pair
  - mln::make, 304
- ord\_strict
  - mln::util, 374
- ord\_weak
  - mln::util, 374
- origin
  - mln::algebra::h\_vec, 490
  - mln::literal, 285
  - mln::point, 866
- other
  - mln::util::vertex, 1007
  - mln::value, 385
- overset
  - mln::p\_if, 796
- P
  - mln::graph\_window\_if\_piter, 686
  - mln::graph\_window\_piter, 689
- p
  - mln::util::pix, 984
- p2p\_image
  - mln::p2p\_image, 772
- p\_array
  - mln::p\_array, 776
- p\_centered
  - mln::p\_centered, 780
- p\_complex
  - mln::p\_complex, 783
- p\_edges
  - mln::p\_edges, 787
- p\_edges\_with\_mass\_centers
  - mln::make, 304
- p\_faces
  - mln::p\_faces, 791

- p\_graph\_piter
  - mln::p\_graph\_piter, 793
- p\_if
  - mln::p\_if, 795
- p\_image
  - mln::p\_image, 798
- p\_indexed\_bkd\_piter
  - mln::p\_indexed\_bkd\_piter, 800
- p\_indexed\_fwd\_piter
  - mln::p\_indexed\_fwd\_piter, 802
- p\_key
  - mln::p\_key, 805
- p\_line2d
  - mln::p\_line2d, 809
- p\_mutable\_array\_of
  - mln::p\_mutable\_array\_of, 813
- p\_n\_faces\_bkd\_piter
  - mln::p\_n\_faces\_bkd\_piter, 814
- p\_n\_faces\_fwd\_piter
  - mln::p\_n\_faces\_fwd\_piter, 815
- p\_priority
  - mln::p\_priority, 818
- p\_queue
  - mln::p\_queue, 823
- p\_queue\_fast
  - mln::p\_queue\_fast, 828
- p\_run
  - mln::p\_run, 832
- p\_run2d
  - mln, 121
- p\_runs2d
  - mln, 121
- p\_set
  - mln::p\_set, 836
- p\_set\_of
  - mln::p\_set\_of, 840
- p\_transformed
  - mln::p\_transformed, 842
- p\_transformed\_piter
  - mln::p\_transformed\_piter, 844
- p\_vaccess
  - mln::p\_vaccess, 847
- p\_vertices
  - mln::p\_vertices, 851, 852
- p\_vertices\_with\_mass\_centers
  - mln::make, 305
- pack
  - mln::labeling, 266, 267
- pack\_inplace
  - mln::labeling, 267
- pair
  - mln::util::site\_pair, 991
- parent
  - mln::util::tree\_node, 1002
- paste
  - mln::data, 179
  - mln::data::impl::generic, 190
- paste\_without\_localization
  - mln::data, 179
- paste\_without\_localization\_fast
  - mln::data::impl, 186
- paste\_without\_localization\_fastest
  - mln::data::impl, 187
- paste\_without\_localization\_lines
  - mln::data::impl, 187
- pcenter
  - mln::box, 500
- pink
  - mln::literal, 285
- piter
  - mln::box, 498
  - mln::p\_array, 775
  - mln::p\_centered, 779
  - mln::p\_complex, 782
  - mln::p\_edges, 787
  - mln::p\_faces, 790
  - mln::p\_if, 795
  - mln::p\_image, 798
  - mln::p\_key, 805
  - mln::p\_line2d, 809
  - mln::p\_mutable\_array\_of, 812
  - mln::p\_priority, 818
  - mln::p\_queue, 823
  - mln::p\_queue\_fast, 827
  - mln::p\_run, 832
  - mln::p\_set, 836
  - mln::p\_set\_of, 839
  - mln::p\_transformed, 842
  - mln::p\_vaccess, 846
  - mln::p\_vertices, 851
- pix
  - mln::make, 305
  - mln::util::pix, 983
- pixel
  - mln::make, 306
  - mln::pixel, 854
- plain
  - mln::plain, 857
- plot
  - mln::draw, 203
- plus
  - mln::arith, 153, 154
  - mln::morpho, 323
- plus\_cst
  - mln::arith, 154, 155
- plus\_cst\_inplace
  - mln::arith, 155
- plus\_infty



- mln::point, 865
- plus\_inplace
  - mln::arith, 155
- pmax
  - mln::box, 500
  - mln::doc::Box, 542
- pmin
  - mln::box, 501
  - mln::doc::Box, 543
- pmin\_pmax
  - mln::geom, 225, 226
- point
  - mln::doc::Dpoint, 544
  - mln::doc::Fastest\_Image, 548
  - mln::doc::Image, 557
  - mln::doc::Neighborhood, 563
  - mln::doc::Point\_Site, 567
  - mln::doc::Weighted\_Window, 578
  - mln::Point, 859
  - mln::point, 864
- point1d
  - mln, 121
- point1df
  - mln, 121
- point2d
  - mln, 121
- point2d\_h
  - mln, 121
  - mln::make, 306
- point2df
  - mln, 122
- point3d
  - mln, 122
- point3df
  - mln, 122
- point\_at\_index
  - mln::doc::Fastest\_Image, 552
  - mln::image1d, 700
  - mln::image2d, 704
  - mln::image3d, 712
- pop
  - mln::p\_priority, 820
  - mln::p\_queue, 824
  - mln::p\_queue\_fast, 829
- pop\_front
  - mln::p\_priority, 820
  - mln::p\_queue, 824
  - mln::p\_queue\_fast, 829
  - mln::util::fibonacci\_heap, 962
  - mln::util::soft\_heap, 994
- positive
  - mln::test, 353
- pred
  - mln::p\_if, 796
- predicate
  - mln::p\_if, 796
  - mln::test, 353, 354
- prev
  - mln::value::label, 1027
- primary
  - mln, 135
- primary\_set
  - mln::p\_transformed, 843
- print
  - mln::topo::complex, 916
  - mln::util::tree\_node, 1003
  - mln::window, 1069
- print\_faces
  - mln::topo::complex, 916
- println
  - mln::debug, 196
- println\_with\_border
  - mln::debug, 196
- priorities
  - mln::p\_priority, 821
- propagate\_if
  - mln::morpho::tree, 334, 335
- propagate\_if\_value
  - mln::morpho::tree, 335
- propagate\_node\_to\_ancestors
  - mln::morpho::tree, 335, 336
- propagate\_node\_to\_descendants
  - mln::morpho::tree, 336
- propagate\_representative
  - mln::morpho::tree, 336
- proxy
  - mln::value::proxy, 1032
- pset
  - mln::doc::Fastest\_Image, 548
  - mln::doc::Image, 557
  - mln::p\_vaccess, 846
- psite
  - mln::box, 498
  - mln::complex\_neighborhood\_bkd\_piter, 527
  - mln::complex\_neighborhood\_fwd\_piter, 529
  - mln::complex\_window\_bkd\_piter, 533
  - mln::complex\_window\_fwd\_piter, 535
  - mln::decorated\_image, 537
  - mln::doc::Box, 542
  - mln::doc::Fastest\_Image, 548
  - mln::doc::Image, 557
  - mln::doc::Site\_Iterator, 570
  - mln::doc::Site\_Set, 572
  - mln::dpoint, 584
  - mln::graph\_elt\_mixed\_window, 671
  - mln::graph\_elt\_window, 678
  - mln::graph\_elt\_window\_if, 682
  - mln::hexa, 692

- mln::image2d\_h, 706
- mln::interpolated, 714
- mln::p\_array, 775
- mln::p\_centered, 780
- mln::p\_complex, 782
- mln::p\_edges, 787
- mln::p\_faces, 791
- mln::p\_if, 795
- mln::p\_image, 798
- mln::p\_key, 805
- mln::p\_line2d, 809
- mln::p\_mutable\_array\_of, 812
- mln::p\_priority, 818
- mln::p\_queue, 823
- mln::p\_queue\_fast, 827
- mln::p\_run, 832
- mln::p\_set, 836
- mln::p\_set\_of, 839
- mln::p\_transformed, 842
- mln::p\_vaccess, 847
- mln::p\_vertices, 851
- mln::thrubin\_image, 888
- mln::topo::is\_simple\_cell, 926
- mln::tr\_image, 938
- mln::util::pix, 983
- mln::value::stack\_image, 1040
- ptransform
  - mln, 135
- purge
  - mln::p\_queue\_fast, 829
- purple
  - mln::literal, 285
- push
  - mln::p\_priority, 821
  - mln::p\_queue, 825
  - mln::p\_queue\_fast, 829
  - mln::util::fibonacci\_heap, 962
  - mln::util::soft\_heap, 994
- put\_word
  - mln::debug, 196
- q\_box
  - mln::p\_line2d, 809
  - mln::p\_run, 832
- qiter
  - mln::doc::Window, 580
  - mln::graph\_elt\_mixed\_window, 671
  - mln::graph\_elt\_window, 678
  - mln::graph\_elt\_window\_if, 682
  - mln::window, 1067
- Queue based, 86
- r\_element
  - mln::p\_image, 798
  - mln::p\_key, 805
  - mln::p\_set, 836
- rag\_and\_labeled\_wsl
  - mln::make, 306
- rank\_filter
  - mln::morpho, 323
- read\_header
  - mln::io::fld, 240
- rectangle2d
  - mln::win::rectangle2d, 1064
- rectangularity
  - mln::accu::site\_set::rectangularity, 458
- red
  - mln::literal, 285
  - mln::value::qt::rgb32, 1034
  - mln::value::rgb, 1036
- region\_adjacency\_graph
  - mln::make, 307
- regional\_maxima
  - mln::labeling, 267
- regional\_minima
  - mln::labeling, 268
- registration1
  - mln::registration, 348
- registration2
  - mln::registration, 348
- registration3
  - mln::registration, 348
- regular
  - mln::window, 1067
- relabel
  - mln::labeled\_image, 721
  - mln::labeled\_image\_base, 724
  - mln::labeling, 268
- relabel\_inplace
  - mln::labeling, 269
- relabelfun
  - mln::make, 307
- remove
  - mln::p\_image, 799
  - mln::p\_key, 807
  - mln::p\_set, 837
  - mln::util::set, 989
- remove\_key
  - mln::p\_key, 807
- replace
  - mln::data, 179
- reserve
  - mln::p\_array, 777
  - mln::p\_mutable\_array\_of, 814
  - mln::p\_queue\_fast, 829
  - mln::topo::n\_faces\_set, 934
  - mln::util::array, 949
- resize

- mln::border, 162
- mln::p\_array, 778
- mln::util::array, 949
- result
  - mln::graph::attribute::card\_t, 667
  - mln::graph::attribute::representative\_t, 667
  - mln::topo::is\_simple\_cell, 926
  - mln::util::array, 946
- revert
  - mln::arith, 156
- revert\_inplace
  - mln::arith, 156
- rgb
  - mln::value::rgb, 1035, 1036
- rgb16
  - mln::value, 381
- rgb32
  - mln::value::qt::rgb32, 1034
- rgb8
  - mln::value, 381
- rgb8\_2complex\_image3df
  - mln, 122
- root
  - mln::util::tree, 999
- rotate
  - mln::geom, 226, 227
- rotation
  - mln::fun::x2x::rotation, 643, 644
- Routines, 75
- run\_length
  - mln::box\_runend\_piter, 507
  - mln::box\_runstart\_piter, 508
- rvalue
  - mln::complex\_image, 525
  - mln::decorated\_image, 537
  - mln::doc::Fastest\_Image, 548
  - mln::doc::Generalized\_Pixel, 554
  - mln::doc::Image, 558
  - mln::doc::Pixel\_Iterator, 565
  - mln::extension\_fun, 598
  - mln::extension\_ima, 600
  - mln::extension\_val, 602
  - mln::flat\_image, 607
  - mln::fun\_image, 648
  - mln::hexa, 692
  - mln::image1d, 697
  - mln::image2d, 702
  - mln::image2d\_h, 706
  - mln::image3d, 709
  - mln::interpolated, 714
  - mln::lazy\_image, 726
  - mln::thrubin\_image, 888
  - mln::tr\_image, 938
  - mln::value::stack\_image, 1040
  - mln::violent\_cast\_image, 1049
- S
  - mln::p\_image, 798
- sagittal\_dec
  - mln, 136
- saturate
  - mln::data, 180
- saturate\_inplace
  - mln::data, 180
- save
  - mln::io::cloud, 236
  - mln::io::dump, 238
  - mln::io::magick, 241
  - mln::io::off, 242
  - mln::io::pbm, 243
  - mln::io::pfm, 246
  - mln::io::pgm, 247
  - mln::io::plot, 249
  - mln::io::pnm, 251
  - mln::io::ppm, 253
  - mln::io::raw, 255
  - mln::io::txt, 256
- save\_bin\_alt
  - mln::io::off, 242
- search
  - mln::util::tree\_node, 1003
- search\_rec
  - mln::util::tree\_node, 1003
- second
  - mln::accu::pair, 450
  - mln::accu::stat::min\_max, 473
  - mln::util::couple, 955
  - mln::util::ord\_pair, 982
  - mln::util::site\_pair, 991
- second\_accu
  - mln::accu::pair, 450
  - mln::accu::stat::min\_max, 473
- seeds2tiling
  - mln::geom, 227
  - mln::geom::impl, 229
- seeds2tiling\_roundness
  - mln::geom, 227
- segment1d
  - modwin1d, 88
- set
  - mln::util::set, 987
- set\_all
  - mln::dpoint, 585
  - mln::point, 865
- set\_alpha
  - mln::fun::x2x::rotation, 644
- set\_axis
  - mln::fun::x2x::rotation, 644

- set\_cplx
  - mln::topo::algebraic\_face, 906
  - mln::topo::algebraic\_n\_face, 910
  - mln::topo::face, 920
  - mln::topo::n\_face, 930
- set\_face\_id
  - mln::topo::algebraic\_face, 906
  - mln::topo::algebraic\_n\_face, 910
  - mln::topo::face, 920
  - mln::topo::n\_face, 930
- set\_image
  - mln::topo::is\_simple\_cell, 926
- set\_n
  - mln::topo::algebraic\_face, 906
  - mln::topo::face, 920
- set\_nbits
  - mln::value::float01, 1011
- set\_parent
  - mln::util::tree\_node, 1003
- set\_sign
  - mln::topo::algebraic\_face, 906
  - mln::topo::algebraic\_n\_face, 910
- set\_t
  - mln::fun::x2x::translation, 647
- set\_tr
  - mln::tr\_image, 939
- set\_value
  - mln::accu::count\_adjacent\_vertices, 393
  - mln::accu::count\_value, 394
  - mln::accu::math::count, 405
  - mln::accu::shape::height, 454
  - mln::accu::shape::volume, 456
  - mln::accu::stat::max, 461
  - mln::accu::stat::min, 469
  - mln::morpho::attribute::sum, 766
- sign
  - mln::topo::algebraic\_face, 906
  - mln::topo::algebraic\_n\_face, 910
  - mln::value::sign, 1038
- site
  - mln::box, 498
  - mln::doc::Box, 542
  - mln::doc::Site\_Set, 572
  - mln::dpoint, 584
  - mln::graph\_elt\_mixed\_window, 671
  - mln::graph\_elt\_window, 678
  - mln::graph\_elt\_window\_if, 682
  - mln::graph\_window\_base, 685
  - mln::p\_centered, 780
  - mln::tr\_image, 938
- Site sets, 83
- site\_function\_t
  - mln::edge\_image, 595
  - mln::vertex\_image, 1047
- site\_set
  - mln::complex\_psite, 532
  - mln::faces\_psite, 605
- size
  - mln::util::array, 949
  - mln::win::line, 1061
  - mln::window, 1069
- skeleton
  - mln::complex\_image, 525
  - mln::decorated\_image, 537
  - mln::doc::Fastest\_Image, 548
  - mln::doc::Image, 558
  - mln::edge\_image, 595
  - mln::extended, 596
  - mln::extension\_fun, 598
  - mln::extension\_ima, 600
  - mln::extension\_val, 602
  - mln::flat\_image, 607
  - mln::fun\_image, 648
  - mln::hexa, 693
  - mln::image1d, 698
  - mln::image2d, 702
  - mln::image2d\_h, 706
  - mln::image3d, 709
  - mln::image\_if, 713
  - mln::interpolated, 714
  - mln::labeled\_image, 720
  - mln::lazy\_image, 726
  - mln::p2p\_image, 772
  - mln::plain, 857
  - mln::pw::image, 872
  - mln::safe\_image, 874
  - mln::slice\_image, 883
  - mln::sub\_image, 885
  - mln::sub\_image\_if, 886
  - mln::thrubin\_image, 888
  - mln::tr\_image, 938
  - mln::transformed\_image, 940
  - mln::value::stack\_image, 1040
  - mln::vertex\_image, 1047
  - mln::violent\_cast\_image, 1049
- sli
  - mln::slice\_image, 884
- slice\_image
  - mln::slice\_image, 883
- slices\_2d
  - mln::debug, 197
- sline3d
  - modwin3d, 92
- soft\_heap
  - mln::util::soft\_heap, 993
- sort\_offsets\_increasing
  - mln::data, 180
- sort\_psites\_decreasing

- mln::data, 181
- sort\_psites\_increasing
  - mln::data, 181
- space\_2complex\_geometry
  - mln, 122
- Sparse types, 85
- sphere3d
  - modwin3d, 92
- sqr\_l2
  - mln::norm, 343
- stack
  - mln::value, 385
- stack\_image
  - mln::value::stack\_image, 1041
- standard\_deviation
  - mln::accu::stat::variance, 482
- start
  - mln::doc::Iterator, 561
  - mln::doc::Pixel\_Iterator, 566
  - mln::doc::Site\_Iterator, 570
  - mln::doc::Value\_Iterator, 574
  - mln::dpoints\_bkd\_pixter, 588
  - mln::dpoints\_fwd\_pixter, 590
  - mln::p\_run, 834
  - mln::topo::face\_bkd\_iter, 921
  - mln::topo::face\_fwd\_iter, 923
  - mln::topo::n\_face\_bkd\_iter, 931
  - mln::topo::n\_face\_fwd\_iter, 933
  - mln::topo::static\_n\_face\_bkd\_iter, 935
  - mln::topo::static\_n\_face\_fwd\_iter, 937
  - mln::util::branch\_iter, 952
  - mln::util::branch\_iter\_ind, 953
- static\_n\_face\_bkd\_iter
  - mln::topo::static\_n\_face\_bkd\_iter, 935
- static\_n\_face\_fwd\_iter
  - mln::topo::static\_n\_face\_fwd\_iter, 936
- std\_deque
  - mln::p\_queue, 825
- std\_vector
  - mln::p\_array, 778
  - mln::p\_line2d, 810
  - mln::p\_queue\_fast, 830
  - mln::p\_set, 837
  - mln::util::array, 949
  - mln::util::set, 989
  - mln::w\_window, 1052
  - mln::win::rectangle2d, 1064
  - mln::window, 1069
- stretch
  - mln::data, 181
  - mln::data::impl, 187
- structural
  - mln::morpho::closing::approx, 326
  - mln::morpho::opening::approx, 329
- sub\_image
  - mln::sub\_image, 885
- sub\_image\_if
  - mln::sub\_image\_if, 887
- subdomain
  - mln::labeled\_image, 721
  - mln::labeled\_image\_base, 724
- subsampling
  - mln::subsampling, 352
- subtractive
  - mln::morpho::tree::filter, 338
- sum
  - mln::accu::stat::mean, 464
  - mln::accu::stat::variance, 482
  - mln::estim, 205
- superpose
  - mln::debug, 197
  - mln::labeling, 269
  - mln::morpho::watershed, 340, 341
- sym
  - mln::doc::Weighted\_Window, 579
  - mln::graph\_elt\_mixed\_window, 672
  - mln::graph\_elt\_window, 679
  - mln::graph\_elt\_window\_if, 684
  - mln::graph\_window\_base, 685
  - mln::w\_window, 1052
  - mln::win, 388
  - mln::window, 1069
- sym\_diff
  - mln::Box, 506
  - mln::Site\_Set, 882
- t
  - mln::algebra::h\_mat, 489
  - mln::algebra::h\_vec, 490
  - mln::fun::x2x::translation, 647
- take
  - mln::accu, 139
  - mln::accu::histo, 396
  - mln::accu::label\_used, 397
  - mln::accu::stat::median\_alt, 465
  - mln::doc::Accumulator, 540
- take\_as\_init
  - mln::accu::center, 390
  - mln::accu::convolve, 391
  - mln::accu::count\_adjacent\_vertices, 393
  - mln::accu::count\_value, 394
  - mln::accu::histo, 396
  - mln::accu::label\_used, 397
  - mln::accu::logic::land, 399
  - mln::accu::logic::land\_basic, 400
  - mln::accu::logic::lor, 401
  - mln::accu::logic::lor\_basic, 403
  - mln::accu::maj\_h, 404

- mln::accu::math::count, 405
- mln::accu::math::inf, 407
- mln::accu::math::sum, 408
- mln::accu::math::sup, 409
- mln::accu::max\_site, 410
- mln::accu::nil, 446
- mln::accu::p, 447
- mln::accu::pair, 450
- mln::accu::rms, 451
- mln::accu::shape::bbox, 452
- mln::accu::shape::height, 454
- mln::accu::shape::volume, 456
- mln::accu::site\_set::rectangularity, 458
- mln::accu::stat::deviation, 459
- mln::accu::stat::max, 461
- mln::accu::stat::max\_h, 462
- mln::accu::stat::mean, 464
- mln::accu::stat::median\_alt, 466
- mln::accu::stat::median\_h, 467
- mln::accu::stat::min, 469
- mln::accu::stat::min\_h, 471
- mln::accu::stat::min\_max, 473
- mln::accu::stat::rank, 475
- mln::accu::stat::rank < bool >, 476
- mln::accu::stat::rank\_high\_quant, 478
- mln::accu::stat::var, 480
- mln::accu::stat::variance, 482
- mln::accu::tuple, 483
- mln::accu::val, 485
- mln::Accumulator, 487
- mln::morpho::attribute::card, 760
- mln::morpho::attribute::count\_adjacent\_vertices, 761
- mln::morpho::attribute::height, 763
- mln::morpho::attribute::sharpness, 765
- mln::morpho::attribute::sum, 767
- mln::morpho::attribute::volume, 768
- take\_n\_times
  - mln::accu::center, 390
  - mln::accu::convolve, 392
  - mln::accu::count\_adjacent\_vertices, 393
  - mln::accu::count\_value, 395
  - mln::accu::histo, 396
  - mln::accu::label\_used, 397
  - mln::accu::logic::land, 399
  - mln::accu::logic::land\_basic, 400
  - mln::accu::logic::lor, 401
  - mln::accu::logic::lor\_basic, 403
  - mln::accu::maj\_h, 404
  - mln::accu::math::count, 405
  - mln::accu::math::inf, 407
  - mln::accu::math::sum, 408
  - mln::accu::math::sup, 409
  - mln::accu::max\_site, 411
  - mln::accu::nil, 446
  - mln::accu::p, 447
  - mln::accu::pair, 450
  - mln::accu::rms, 451
  - mln::accu::shape::bbox, 453
  - mln::accu::shape::height, 455
  - mln::accu::shape::volume, 457
  - mln::accu::site\_set::rectangularity, 458
  - mln::accu::stat::deviation, 460
  - mln::accu::stat::max, 461
  - mln::accu::stat::max\_h, 462
  - mln::accu::stat::mean, 464
  - mln::accu::stat::median\_alt, 466
  - mln::accu::stat::median\_h, 467
  - mln::accu::stat::min, 469
  - mln::accu::stat::min\_h, 471
  - mln::accu::stat::min\_max, 474
  - mln::accu::stat::rank, 475
  - mln::accu::stat::rank < bool >, 476
  - mln::accu::stat::rank\_high\_quant, 478
  - mln::accu::stat::var, 480
  - mln::accu::stat::variance, 482
  - mln::accu::tuple, 484
  - mln::accu::val, 485
  - mln::Accumulator, 487
  - mln::morpho::attribute::card, 760
  - mln::morpho::attribute::count\_adjacent\_vertices, 762
  - mln::morpho::attribute::height, 763
  - mln::morpho::attribute::sharpness, 765
  - mln::morpho::attribute::sum, 767
  - mln::morpho::attribute::volume, 768
- target
  - mln::graph\_elt\_mixed\_window, 671
  - mln::graph\_elt\_window, 678
  - mln::graph\_elt\_window\_if, 682
- target\_site\_set
  - mln::graph\_window\_piter, 690
- teal
  - mln::literal, 285
- the
  - mln::value::set, 1037
- thick\_miss
  - mln::morpho, 323
- thickening
  - mln::morpho, 323
- thin\_fit
  - mln::morpho, 323
- thinning
  - mln::morpho, 324
- threshold
  - mln::binarization, 159
- times
  - mln::arith, 157

- times\_cst
  - mln::arith, 157
- times\_inplace
  - mln::arith, 157
- to
  - mln::convert, 170
- to\_dpoint
  - mln::convert, 171
  - mln::Dpoint, 581
- to\_enc
  - mln::data, 181
- to\_float
  - mln::value::graylevel, 1015
- to\_fun
  - mln::convert, 171, 173
- to\_h\_vec
  - mln::point, 866
- to\_image
  - mln::convert, 171
- to\_larger
  - mln::box, 501
- to\_nbits
  - mln::value::float01, 1011
- to\_neighb
  - mln::graph, 231
- to\_p\_array
  - mln::convert, 171
- to\_p\_set
  - mln::convert, 171, 172
- to\_point
  - mln::doc::Point\_Site, 568
  - mln::Point, 859
- to\_qimage
  - mln::convert, 172
- to\_result
  - mln::accu::center, 390
  - mln::accu::convolve, 392
  - mln::accu::count\_adjacent\_vertices, 393
  - mln::accu::count\_value, 395
  - mln::accu::label\_used, 398
  - mln::accu::logic::land, 399
  - mln::accu::logic::land\_basic, 400
  - mln::accu::logic::lor, 401
  - mln::accu::logic::lor\_basic, 403
  - mln::accu::maj\_h, 404
  - mln::accu::math::count, 406
  - mln::accu::math::inf, 407
  - mln::accu::math::sum, 408
  - mln::accu::math::sup, 409
  - mln::accu::max\_site, 411
  - mln::accu::nil, 446
  - mln::accu::p, 448
  - mln::accu::pair, 450
  - mln::accu::rms, 451
  - mln::accu::shape::bbox, 453
  - mln::accu::shape::height, 455
  - mln::accu::shape::volume, 457
  - mln::accu::site\_set::rectangularity, 458
  - mln::accu::stat::deviation, 460
  - mln::accu::stat::max, 461
  - mln::accu::stat::max\_h, 462
  - mln::accu::stat::mean, 464
  - mln::accu::stat::median\_alt, 466
  - mln::accu::stat::median\_h, 467
  - mln::accu::stat::min, 470
  - mln::accu::stat::min\_h, 471
  - mln::accu::stat::min\_max, 474
  - mln::accu::stat::rank, 475
  - mln::accu::stat::rank< bool >, 476
  - mln::accu::stat::rank\_high\_quant, 478
  - mln::accu::stat::var, 480
  - mln::accu::stat::variance, 482
  - mln::accu::tuple, 484
  - mln::accu::val, 485
  - mln::morpho::attribute::card, 760
  - mln::morpho::attribute::count\_adjacent\_vertices, 762
  - mln::morpho::attribute::height, 763
  - mln::morpho::attribute::sharpness, 765
  - mln::morpho::attribute::sum, 767
  - mln::morpho::attribute::volume, 768
- to\_upper\_window
  - mln::convert, 172
- to\_value
  - mln::value::proxy, 1033
- to\_vec
  - mln::algebra::h\_vec, 490
  - mln::dpoint, 585
  - mln::point, 866
- to\_win
  - mln::graph, 231
- to\_window
  - mln::convert, 172, 173
- toggle
  - mln::p\_image, 800
- top\_hat\_black
  - mln::morpho, 324
  - mln::morpho::elementary, 327
- top\_hat\_self\_complementary
  - mln::morpho, 324
  - mln::morpho::elementary, 327
- top\_hat\_white
  - mln::morpho, 324
  - mln::morpho::elementary, 328
- topological
  - mln::morpho::watershed, 341
- tr
  - mln::tr\_image, 939

- tr\_image
  - mln::tr\_image, 939
- tracked\_ptr
  - mln::util::tracked\_ptr, 996
- trait::graph, 1070
- trait::graph< mln::complex\_image< 1, G, V > >, 1071
- trait::graph< mln::image2d< T > >, 1071
- transform
  - mln::data, 182
  - mln::data::impl::generic, 190
- transform\_inplace
  - mln::data, 182, 183
  - mln::data::impl::generic, 190, 191
- transform\_inplace\_lowq
  - mln::data::impl, 188
- transformed\_image
  - mln::transformed\_image, 941
- translate
  - mln::geom, 228
- translation
  - mln::fun::x2x::translation, 646
- tree
  - mln::util::tree, 998
- tree\_fast\_to\_image
  - mln::util, 374
- tree\_node
  - mln::util::tree\_node, 1001
- tree\_to\_fast
  - mln::util, 374
- tree\_to\_image
  - mln::util, 375
- Types, 74
- uni
  - mln::Box, 506
  - mln::Site\_Set, 882
- unique
  - mln::Box, 506
  - mln::Site\_Set, 882
- unproject\_image
  - mln::unproject\_image, 942
- unsigned\_2complex\_image3df
  - mln, 122
- untake
  - mln::morpho::attribute::sum, 767
- up
  - mln, 136
- update
  - mln::data, 183
  - mln::data::impl::generic, 191
  - mln::dpoints\_bkd\_pixter, 588
  - mln::dpoints\_fwd\_pixter, 590
- update\_data
  - mln::labeled\_image, 721
  - mln::labeled\_image\_base, 724
- update\_fastest
  - mln::data::impl, 188
- update\_id
  - mln::util::edge, 959
  - mln::util::vertex, 1008
- util\_set
  - mln::p\_set, 838
- util\_tree
  - mln::util::branch, 950
- Utilities, 86
- v
  - mln::util::pix, 984
- v1
  - mln::util::edge, 959
  - mln::util::graph, 968
  - mln::util::line\_graph, 976
- v2
  - mln::util::edge, 959
  - mln::util::graph, 968
  - mln::util::line\_graph, 976
- v2w2v functions, 94
- v2w\_w2v functions, 94
- v\_ith\_nbh\_edge
  - mln::util::graph, 968
  - mln::util::line\_graph, 976
- v\_ith\_nbh\_vertex
  - mln::util::graph, 968
  - mln::util::line\_graph, 976
- v\_nmax
  - mln::util::graph, 968
  - mln::util::line\_graph, 977
- v\_nmax\_nbh\_edges
  - mln::util::graph, 969
  - mln::util::line\_graph, 977
- v\_nmax\_nbh\_vertices
  - mln::util::graph, 969
  - mln::util::line\_graph, 977
- v\_other
  - mln::util::edge, 959
- val
  - mln::doc::Generalized\_Pixel, 554
  - mln::doc::Pixel\_Iterator, 566
- value
  - mln::accu::shape::height, 454
  - mln::accu::shape::volume, 456
  - mln::complex\_image, 525
  - mln::doc::Fastest\_Image, 549
  - mln::doc::Generalized\_Pixel, 554
  - mln::doc::Image, 558
  - mln::doc::Pixel\_Iterator, 565
  - mln::doc::Value\_Iterator, 574



- mln::doc::Value\_Set, 576
- mln::extended, 596
- mln::extension\_fun, 598
- mln::extension\_ima, 600
- mln::extension\_val, 602
- mln::flat\_image, 607
- mln::fun\_image, 648
- mln::hexa, 693
- mln::image1d, 698
- mln::image2d, 702
- mln::image2d\_h, 707
- mln::image3d, 709
- mln::interpolated, 714
- mln::labeling, 270
- mln::p\_vaccess, 847
- mln::thrubin\_image, 888
- mln::tr\_image, 938
- mln::util::pix, 983
- mln::value::float01, 1011
- mln::value::float01\_f, 1013
- mln::value::graylevel, 1015
- mln::value::graylevel\_f, 1018
- mln::value::lut\_vec, 1029
- mln::value::stack\_image, 1040
- mln::violent\_cast\_image, 1049
- value\_and\_compute
  - mln::labeling, 270
- value\_array
  - mln::value::value\_array, 1042
- value\_ind
  - mln::value::float01, 1012
- value\_t
  - mln::util::object\_id, 980
- values
  - mln::complex\_image, 526
  - mln::doc::Fastest\_Image, 552
  - mln::doc::Image, 560
  - mln::p\_vaccess, 848
- Values morphers, 72
- var
  - mln::accu::stat::variance, 482
- variance
  - mln::accu::stat::var, 480
- vec
  - mln::dpoint, 584
  - mln::make, 308, 309
  - mln::point, 863
- vec2d\_d
  - mln, 122
- vec2d\_f
  - mln, 122
- vec3d\_d
  - mln, 122
- vec3d\_f
  - mln, 122
- vect
  - mln::accu::histo, 396
- vertex
  - mln::p\_vertices, 851
  - mln::util::graph, 969
  - mln::util::line\_graph, 977
  - mln::util::vertex, 1006
- vertex\_fwd\_iter
  - mln::util::graph, 965
  - mln::util::line\_graph, 974
- vertex\_id\_t
  - mln::util, 372
- vertex\_image
  - mln::make, 309
  - mln::vertex\_image, 1047
- vertex\_nbh\_edge\_fwd\_iter
  - mln::util::graph, 965
  - mln::util::line\_graph, 974
- vertex\_nbh\_t
  - mln::vertex\_image, 1047
- vertex\_nbh\_vertex\_fwd\_iter
  - mln::util::graph, 966
  - mln::util::line\_graph, 974
- vertex\_win\_t
  - mln::vertex\_image, 1047
- vertical\_symmetry
  - mln::geom, 229
- vertices\_t
  - mln::util::graph, 966
  - mln::util::line\_graph, 974
- violent\_cast\_image
  - mln::violent\_cast\_image, 1049
- violet
  - mln::literal, 285
- vline2d
  - modwin2d, 90
- volume
  - mln::morpho::attribute::sharpness, 765
  - mln::win::cuboid3d, 1059
- voronoi
  - mln::make, 310
- vprod
  - mln::algebra, 146
- vset
  - mln::doc::Fastest\_Image, 549
  - mln::doc::Image, 558
  - mln::p\_vaccess, 847
  - mln::value::value\_array, 1042
- vv2b functions, 94
- w
  - mln::w\_window, 1052
- w\_window

- mln::make, 310
  - mln::w\_window, 1051
- w\_window1d
  - mln::make, 310
- w\_window1d\_float
  - mln, 123
- w\_window1d\_int
  - mln, 123
  - mln::make, 311
- w\_window2d
  - mln::make, 311
- w\_window2d\_float
  - mln, 123
- w\_window2d\_int
  - mln, 123
  - mln::make, 311
- w\_window3d
  - mln::make, 312
- w\_window3d\_float
  - mln, 123
- w\_window3d\_int
  - mln, 123
  - mln::make, 312
- w\_window\_directional
  - mln::make, 312
- weight
  - mln::doc::Weighted\_Window, 578
  - mln::w\_window, 1051
- weights
  - mln::w\_window, 1052
- white
  - mln::literal, 285
- width
  - mln::win::cuboid3d, 1059
  - mln::win::rectangle2d, 1064
- win
  - mln::doc::Weighted\_Window, 579
  - mln::w\_window, 1052
- win\_c4p
  - modwin2d, 90
- win\_c4p\_3d
  - modwin3d, 92
- win\_c8p
  - modwin2d, 91
- win\_c8p\_3d
  - modwin3d, 93
- win\_t
  - mln::edge\_image, 595
  - mln::vertex\_image, 1047
- window
  - mln::doc::Weighted\_Window, 578
  - mln::p\_centered, 780
  - mln::window, 1067
- window1d
  - modwin1d, 88
- window2d
  - modwin2d, 90
- window3d
  - modwin3d, 92
- Windows, 87
- wrap
  - mln::data, 183
  - mln::labeling, 271
- write\_header
  - mln::io::fld, 240
- xor\_inplace
  - mln::logical, 289
- yellow
  - mln::literal, 286
- z\_order
  - mln::debug, 198
- zero
  - mln::algebra::h\_vec, 490
  - mln::literal, 286
  - mln::value::int\_s, 1020
  - mln::value::int\_u\_sat, 1024
  - mln::value::qt::rgb32, 1034
  - mln::value::rgb, 1036
  - mln::value::sign, 1039