

# spot Reference Manual

0.4

Generated by Doxygen 1.5.2

Tue Jul 17 15:13:12 2007

## Contents

<b>1</b>	<b>spot Main Page</b>	<b>1</b>
<b>2</b>	<b>spot Module Index</b>	<b>2</b>
<b>3</b>	<b>spot Directory Hierarchy</b>	<b>3</b>
<b>4</b>	<b>spot Namespace Index</b>	<b>4</b>
<b>5</b>	<b>spot Hierarchical Index</b>	<b>4</b>
<b>6</b>	<b>spot Class Index</b>	<b>9</b>
<b>7</b>	<b>spot File Index</b>	<b>14</b>
<b>8</b>	<b>spot Page Index</b>	<b>18</b>
<b>9</b>	<b>spot Module Documentation</b>	<b>18</b>
<b>10</b>	<b>spot Directory Documentation</b>	<b>56</b>
<b>11</b>	<b>spot Namespace Documentation</b>	<b>64</b>
<b>12</b>	<b>spot Class Documentation</b>	<b>86</b>
<b>13</b>	<b>spot File Documentation</b>	<b>465</b>
<b>14</b>	<b>spot Page Documentation</b>	<b>559</b>

## 1 spot Main Page

### 1.1 The Spot Library

Spot is a model-checking library. It provides algorithms and data structures to implement the automata-theoretic approach to model-checking.

See [spot.lip6.fr](http://spot.lip6.fr) for more information about this project.

### 1.2 This Document

This document describes all the public data structures and functions of Spot. This aims to be a reference manual, not a tutorial.

If you are new to this manual, start with [the module page](#). This is what looks the closest to a table of contents.

## 1.3 Handy starting points

- [spot::ltl::formula](#) Base class for an LTL formulae.
- [spot::ltl::parse](#) Parsing a text string into a [spot::ltl::formula](#).
- [spot::tgba](#) Base class for Transition-based Generalized Büchi Automaton.
- [spot::ltl\\_to\\_tgba\\_fm](#) Convert a [spot::ltl::formula](#) into a [spot::tgba](#).
- [spot::tgba\\_product](#) On-the-fly product of two [spot::tgba](#).
- [spot::emptiness\\_check](#) Base class for all emptiness-check algorithms (see also module [Emptiness-checks](#))

## 2 spot Module Index

### 2.1 spot Modules

Here is a list of all modules:

<b>LTL formulae</b>	<b>18</b>
<b>Essential LTL types</b>	<b>19</b>
<b>LTL Abstract Syntax Tree</b>	<b>20</b>
<b>LTL environments</b>	<b>20</b>
<b>Algorithms for LTL formulae</b>	<b>21</b>
<b>Input/Output of LTL formulae</b>	<b>21</b>
<b>Derivable visitors</b>	<b>24</b>
<b>Rewriting LTL formulae</b>	<b>24</b>
<b>Miscellaneous algorithms for LTL formulae</b>	<b>26</b>
<b>TGBA (Transition-based Generalized Büchi Automata)</b>	<b>18</b>
<b>Essential TGBA types</b>	<b>33</b>
<b>TGBA representations</b>	<b>34</b>
<b>TGBA algorithms</b>	<b>34</b>
<b>TGBA on-the-fly algorithms</b>	<b>34</b>
<b>Input/Output of TGBA</b>	<b>35</b>
<b>Decorating the dot output</b>	<b>44</b>
<b>Translating LTL formulae into TGBA</b>	<b>37</b>
<b>Algorithm patterns</b>	<b>39</b>

TGBA simplifications	40
Miscellaneous algorithms on TGBA	42
Emptiness-checks	44
Emptiness-check algorithms for SSP	19
Emptiness-check algorithms	45
TGBA runs and supporting functions	54
Emptiness-check statistics	56
Miscellaneous helper algorithms	29
Hashing functions	31
Random functions	31

### 3 spot Directory Hierarchy

#### 3.1 spot Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

evtgba	56
evtgbalogs	57
evtgbaparse	57
gspn	58
ltlast	59
ltlenv	60
ltlparse	60
ltlvisit	61
misc	61
sautparse	62
tgba	62
tgbaalogs	63
gtec	58
tgbaparse	64

## 4 spot Namespace Index

### 4.1 spot Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">ltlyy</a>	64
<a href="#">sautyy</a>	66
<a href="#">spot</a>	68
<a href="#">spot::ltl</a>	82

## 5 spot Hierarchical Index

### 5.1 spot Class Hierarchy

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

<a href="#">spot::barand&lt; gen &gt;</a>	97
<a href="#">spot::bdd_less_than</a>	117
<a href="#">spot::bfs_steps</a>	117
<a href="#">spot::char_ptr_less_than</a>	127
<a href="#">spot::ltl::const_visitor</a>	134
<a href="#">spot::couvreur99_check_shy::successor</a>	159
<a href="#">spot::couvreur99_check_shy::todo_item</a>	160
<a href="#">spot::couvreur99_check_status</a>	161
<a href="#">spot::delayed_simulation_relation</a>	167
<a href="#">spot::direct_simulation_relation</a>	167
<a href="#">spot::dotty_decorator</a>	167
<a href="#">spot::tgba_run_dotty_decorator</a>	413
<a href="#">spot::emptiness_check</a>	182
<a href="#">spot::couvreur99_check</a>	140
<a href="#">spot::couvreur99_check_shy</a>	151
<a href="#">spot::emptiness_check_instantiator</a>	186
<a href="#">spot::emptiness_check_result</a>	189
<a href="#">spot::couvreur99_check_result</a>	147

spot::ltl::environment	192
spot::ltl::declarative_environment	163
spot::ltl::default_environment	165
spot::ltl::read_only_environment	294
spot::evtgba	193
spot::evtgba_explicit	196
spot::evtgba_product	202
spot::evtgba_explicit::state	200
spot::evtgba_explicit::transition	200
spot::evtgba_iterator	201
spot::evtgba_reachable_iterator	205
spot::evtgba_reachable_iterator_breadth_first	209
spot::evtgba_reachable_iterator_depth_first	213
spot::explicit_connected_component_factory	218
spot::connected_component_hash_set_factory	133
spot::ltl::formula	219
spot::ltl::constant	135
spot::ltl::ref_formula	297
spot::ltl::atomic_prop	91
spot::ltl::binop	120
spot::ltl::multop	246
spot::ltl::unop	453
spot::ltl::formula_ptr_hash	222
spot::ltl::formula_ptr_less_than	223
spot::free_list	223
spot::bdd_allocator	98
spot::bdd_dict	103
spot::bdd_dict::anon_free_list	112
spot::gspn_exception	227

spot::gspn_interface	227
spot::gspn_ssp_interface	229
spot::ltl::language_containment_checker	231
spot::ltl::language_containment_checker::record_	235
sautyy::location	236
ltlyy::location	238
spot::loopless_modular_mixed_radix_gray_code	240
spot::minato_isop	243
spot::minato_isop::local_vars	244
spot::ltl::multop::pairemp	253
spot::numbered_state_heap	254
spot::numbered_state_heap_hash_map	258
spot::numbered_state_heap_const_iterator	256
spot::numbered_state_heap_factory	257
spot::numbered_state_heap_hash_map_factory	262
spot::option_map	263
ltlyy::position	284
sautyy::position	286
spot::ptr_hash< T >	290
spot::ltl::random_ltl	290
spot::ltl::random_ltl::op_proba	293
spot::rsymbol	301
spot::scc_stack	302
spot::scc_stack::connected_component	304
spot::explicit_connected_component	216
spot::connected_component_hash_set	130
ltlyy::slice< T, S >	308
sautyy::slice< T, S >	309
spot::spoiler_node	310

spot::duplicator_node	169
spot::duplicator_node_delayed	174
spot::spoiler_node_delayed	313
sautyy::stack< T, S >	317
ltlyy::stack< T, S >	319
spot::state	321
spot::state_bdd	323
spot::state_evtgba_explicit	325
spot::state_explicit	328
spot::state_product	330
spot::state_ptr_equal	333
spot::state_ptr_hash	334
spot::state_ptr_less_than	334
spot::string_hash	335
spot::symbol	335
spot::tgba	337
spot::tgba_bdd_concrete	344
spot::tgba_explicit	362
spot::tgba_reduc	396
spot::tgba_product	375
spot::tgba_tba_proxy	435
spot::tgba_sba_proxy	416
spot::tgba_bdd_core_data	355
spot::tgba_bdd_factory	361
spot::tgba_bdd_concrete_factory	351
spot::tgba_explicit::transition	371
spot::tgba_reachable_iterator	383
spot::tgba_reachable_iterator_breadth_first	387
spot::parity_game_graph	265



spot::parity_game_graph_delayed	271
spot::parity_game_graph_direct	278
spot::tgba_reduc	396
spot::tgba_reachable_iterator_depth_first	392
spot::tgba_run	411
spot::tgba_run::step	412
spot::tgba_statistics	422
spot::tgba_succ_iterator	422
spot::tgba_explicit_succ_iterator	372
spot::tgba_succ_iterator_concrete	425
spot::tgba_succ_iterator_product	431
spot::time_info	442
spot::timer	443
spot::timer_map	444
spot::unsigned_statistics	459
spot::ars_statistics	89
spot::acss_statistics	86
spot::couvreur99_check_result	147
spot::ec_statistics	178
spot::couvreur99_check	140
spot::unsigned_statistics_copy	460
spot::ltl::visitor	461
spot::ltl::clone_visitor	128
spot::ltl::simplify_f_g_visitor	305
spot::ltl::unabbreviate_logic_visitor	446
spot::ltl::unabbreviate_ltl_visitor	450
spot::ltl::postfix_visitor	287
spot::weight	463

## 6 spot Class Index

### 6.1 spot Class List

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

<a href="#"><code>spot::acss_statistics</code></a> (Accepting Cycle Search Space statistics )	86
<a href="#"><code>spot::ars_statistics</code></a> (Accepting Run Search statistics )	89
<a href="#"><code>spot::ltl::atomic_prop</code></a> (Atomic propositions )	91
<a href="#"><code>spot::barand&lt; gen &gt;</code></a> (Compute pseudo-random integer value between 0 and $n$ included, following a binomial distribution for probability $p$ )	97
<a href="#"><code>spot::bdd_allocator</code></a> (Manage ranges of variables )	98
<a href="#"><code>spot::bdd_dict</code></a>	103
<a href="#"><code>spot::bdd_dict::anon_free_list</code></a>	112
<a href="#"><code>spot::bdd_less_than</code></a> (Comparison functor for BDDs )	117
<a href="#"><code>spot::bfs_steps</code></a> (Make a BFS in a <a href="#"><code>spot::tgba</code></a> to compute a <a href="#"><code>tgba_run::steps</code></a> )	117
<a href="#"><code>spot::ltl::binop</code></a> (Binary operator )	120
<a href="#"><code>spot::char_ptr_less_than</code></a> (Strict Weak Ordering for <code>char*</code> )	127
<a href="#"><code>spot::ltl::clone_visitor</code></a> (Clone a <a href="#"><code>formula</code></a> )	128
<a href="#"><code>spot::connected_component_hash_set</code></a>	130
<a href="#"><code>spot::connected_component_hash_set_factory</code></a> (Factory for <a href="#"><code>connected_component_hash_set</code></a> )	133
<a href="#"><code>spot::ltl::const_visitor</code></a> (Formula <a href="#"><code>visitor</code></a> that cannot modify the <a href="#"><code>formula</code></a> )	134
<a href="#"><code>spot::ltl::constant</code></a> (A <a href="#"><code>constant</code></a> (True or False) )	135
<a href="#"><code>spot::couvreur99_check</code></a> (An implementation of the Couvreur99 emptiness-check algorithm )	140
<a href="#"><code>spot::couvreur99_check_result</code></a> (Compute a counter example from a <a href="#"><code>spot::couvreur99_check_status</code></a> )	147
<a href="#"><code>spot::couvreur99_check_shy</code></a> (A version of <a href="#"><code>spot::couvreur99_check</code></a> that tries to visit known states first )	151
<a href="#"><code>spot::couvreur99_check_shy::successor</code></a>	159
<a href="#"><code>spot::couvreur99_check_shy::todo_item</code></a>	160
<a href="#"><code>spot::couvreur99_check_status</code></a> (The status of the emptiness-check on success )	161
<a href="#"><code>spot::ltl::declarative_environment</code></a> (A declarative <a href="#"><code>environment</code></a> )	163
<a href="#"><code>spot::ltl::default_environment</code></a> (A laxist <a href="#"><code>environment</code></a> )	165

<a href="#">spot::delayed_simulation_relation</a>	167
<a href="#">spot::direct_simulation_relation</a>	167
<a href="#">spot::dotty_decorator</a> (Choose <a href="#">state</a> and link styles for <a href="#">spot::dotty_reachable</a> )	167
<a href="#">spot::duplicator_node</a> (Duplicator node of parity game graph )	169
<a href="#">spot::duplicator_node_delayed</a> (Duplicator node of parity game graph for delayed simulation )	174
<a href="#">spot::ec_statistics</a> (Emptiness-check statistics )	178
<a href="#">spot::emptiness_check</a> (Common interface to emptiness check algorithms )	182
<a href="#">spot::emptiness_check_instantiator</a>	186
<a href="#">spot::emptiness_check_result</a> (The result of an emptiness check )	189
<a href="#">spot::ltl::environment</a> (An <a href="#">environment</a> that describes atomic propositions )	192
<a href="#">spot::evtgba</a>	193
<a href="#">spot::evtgba_explicit</a>	196
<a href="#">spot::evtgba_explicit::state</a>	200
<a href="#">spot::evtgba_explicit::transition</a> (Explicit transitions (used by <a href="#">spot::evtgba_explicit</a> ) )	200
<a href="#">spot::evtgba_iterator</a>	201
<a href="#">spot::evtgba_product</a>	202
<a href="#">spot::evtgba_reachable_iterator</a> (Iterate over all reachable states of a <a href="#">spot::evtgba</a> )	205
<a href="#">spot::evtgba_reachable_iterator_breadth_first</a> (An implementation of <a href="#">spot::evtgba_reachable_iterator</a> that browses states breadth first )	209
<a href="#">spot::evtgba_reachable_iterator_depth_first</a> (An implementation of <a href="#">spot::evtgba_reachable_iterator</a> that browses states depth first )	213
<a href="#">spot::explicit_connected_component</a> (An SCC storing all its states explicitly )	216
<a href="#">spot::explicit_connected_component_factory</a> (Abstract factory for <a href="#">explicit_connected_component</a> )	218
<a href="#">spot::ltl::formula</a> (An LTL formula )	219
<a href="#">spot::ltl::formula_ptr_hash</a> (Hash Function for <code>const formula*</code> )	222
<a href="#">spot::ltl::formula_ptr_less_than</a> (Strict Weak Ordering for <code>const formula*</code> )	223
<a href="#">spot::free_list</a> (Manage list of free integers )	223
<a href="#">spot::gspn_exception</a> (An exception used to forward GSPN errors )	227
<a href="#">spot::gspn_interface</a>	227

<a href="#">spot::gspn_ssp_interface</a>	229
<a href="#">spot::ltl::language_containment_checker</a>	231
<a href="#">spot::ltl::language_containment_checker::record_</a>	235
<a href="#">sauty::location</a> (Abstract a <a href="#">location</a> )	236
<a href="#">ltly::location</a> (Abstract a <a href="#">location</a> )	238
<a href="#">spot::loopless_modular_mixed_radix_gray_code</a> (Loopless modular mixed radix Gray code iteration )	240
<a href="#">spot::minato_isop</a> (Generate an irredundant sum-of-products (ISOP) form of a BDD function )	243
<a href="#">spot::minato_isop::local_vars</a> (Internal variables for <a href="#">minato_isop</a> )	244
<a href="#">spot::ltl::multop</a> (Multi-operand operators )	246
<a href="#">spot::ltl::multop::pairemp</a> (Comparison functor used internally by <a href="#">ltl::multop</a> )	253
<a href="#">spot::numbered_state_heap</a> (Keep track of a large quantity of indexed states )	254
<a href="#">spot::numbered_state_heap_const_iterator</a> (Iterator on <a href="#">numbered_state_heap</a> objects )	256
<a href="#">spot::numbered_state_heap_factory</a> (Abstract factory for <a href="#">numbered_state_heap</a> )	257
<a href="#">spot::numbered_state_heap_hash_map</a> (A straightforward implementation of <a href="#">numbered_state_heap</a> with a hash map )	258
<a href="#">spot::numbered_state_heap_hash_map_factory</a> (Factory for <a href="#">numbered_state_heap_hash_map</a> )	262
<a href="#">spot::option_map</a> (Manage a map of options )	263
<a href="#">spot::parity_game_graph</a> (Parity game graph which compute a simulation relation )	265
<a href="#">spot::parity_game_graph_delayed</a>	271
<a href="#">spot::parity_game_graph_direct</a> (Parity game graph which compute the direct simulation relation )	278
<a href="#">ltly::position</a> (Abstract a <a href="#">position</a> )	284
<a href="#">sauty::position</a> (Abstract a <a href="#">position</a> )	286
<a href="#">spot::ltl::postfix_visitor</a> (Apply an algorithm on each node of an AST, during a postfix traversal )	287
<a href="#">spot::ptr_hash&lt; T &gt;</a> (A hash function for pointers )	290
<a href="#">spot::ltl::random_ltl</a> (Generate random LTL formulae )	290
<a href="#">spot::ltl::random_ltl::op_proba</a>	293
<a href="#">spot::ltl::read_only_environment</a> (A read only <a href="#">environment</a> )	294

<a href="#">spot::ltl::ref_formula</a> (A reference-counted LTL <a href="#">formula</a> )	297
<a href="#">spot::rsymbol</a>	301
<a href="#">spot::scc_stack</a>	302
<a href="#">spot::scc_stack::connected_component</a>	304
<a href="#">spot::ltl::simplify_f_g_visitor</a> (Replace <code>true U f</code> and <code>false R g</code> by <code>F f</code> and <code>G g</code> )	305
<a href="#">ltl::slice&lt; T, S &gt;</a> (Present a <a href="#">slice</a> of the top of a <a href="#">stack</a> )	308
<a href="#">sauty::slice&lt; T, S &gt;</a> (Present a <a href="#">slice</a> of the top of a <a href="#">stack</a> )	309
<a href="#">spot::spoiler_node</a> (Spoiler node of parity game graph )	310
<a href="#">spot::spoiler_node_delayed</a> (Spoiler node of parity game graph for delayed simulation )	313
<a href="#">sauty::stack&lt; T, S &gt;</a>	317
<a href="#">ltl::stack&lt; T, S &gt;</a>	319
<a href="#">spot::state</a> (Abstract class for states )	321
<a href="#">spot::state_bdd</a>	323
<a href="#">spot::state_evtgba_explicit</a> (States used by <a href="#">spot::tgba_evtgba_explicit</a> )	325
<a href="#">spot::state_explicit</a>	328
<a href="#">spot::state_product</a> (A <a href="#">state</a> for <a href="#">spot::tgba_product</a> )	330
<a href="#">spot::state_ptr_equal</a> (An Equivalence Relation for <code>state*</code> )	333
<a href="#">spot::state_ptr_hash</a> (Hash Function for <code>state*</code> )	334
<a href="#">spot::state_ptr_less_than</a> (Strict Weak Ordering for <code>state*</code> )	334
<a href="#">spot::string_hash</a> (A hash function for strings )	335
<a href="#">spot::symbol</a>	335
<a href="#">spot::tgba</a> (A Transition-based Generalized Büchi Automaton )	337
<a href="#">spot::tgba_bdd_concrete</a> (A concrete <a href="#">spot::tgba</a> implemented using BDDs )	344
<a href="#">spot::tgba_bdd_concrete_factory</a> (Helper class to build a <a href="#">spot::tgba_bdd_concrete</a> object )	351
<a href="#">spot::tgba_bdd_core_data</a> (Core data for a TGBA encoded using BDDs )	355
<a href="#">spot::tgba_bdd_factory</a> (Abstract class for <a href="#">spot::tgba_bdd_concrete</a> factories )	361
<a href="#">spot::tgba_explicit</a>	362
<a href="#">spot::tgba_explicit::transition</a> (Explicit transitions (used by <a href="#">spot::tgba_explicit</a> ) )	371
<a href="#">spot::tgba_explicit_succ_iterator</a>	372

<a href="#">spot::tgba_product</a> (A lazy product. (States are computed on the fly.) )	375
<a href="#">spot::tgba_reachable_iterator</a> (Iterate over all reachable states of a <a href="#">spot::tgba</a> )	383
<a href="#">spot::tgba_reachable_iterator_breadth_first</a> (An implementation of <a href="#">spot::tgba_reachable_iterator</a> that browses states breadth first )	387
<a href="#">spot::tgba_reachable_iterator_depth_first</a> (An implementation of <a href="#">spot::tgba_reachable_iterator</a> that browses states depth first )	392
<a href="#">spot::tgba_reduc</a>	396
<a href="#">spot::tgba_run</a> (An accepted run, for a <a href="#">tgba</a> )	411
<a href="#">spot::tgba_run::step</a>	412
<a href="#">spot::tgba_run_dotty_decorator</a> (Highlight a <a href="#">spot::tgba_run</a> on a <a href="#">spot::tgba</a> )	413
<a href="#">spot::tgba_sba_proxy</a> (Degeneralize a <a href="#">spot::tgba</a> on the fly, producing an SBA )	416
<a href="#">spot::tgba_statistics</a>	422
<a href="#">spot::tgba_succ_iterator</a> (Iterate over the successors of a <a href="#">state</a> )	422
<a href="#">spot::tgba_succ_iterator_concrete</a>	425
<a href="#">spot::tgba_succ_iterator_product</a> (Iterate over the successors of a product computed on the fly )	431
<a href="#">spot::tgba_tba_proxy</a> (Degeneralize a <a href="#">spot::tgba</a> on the fly, producing a TBA )	435
<a href="#">spot::time_info</a> (A structure to record elapsed time in clock ticks )	442
<a href="#">spot::timer</a> (A timekeeper that accumulate interval of time )	443
<a href="#">spot::timer_map</a> (A map of <a href="#">timer</a> , where each <a href="#">timer</a> has a name )	444
<a href="#">spot::ltl::unabbreviate_logic_visitor</a> (Clone and rewrite a <a href="#">formula</a> to remove most of the abbreviated logical operators )	446
<a href="#">spot::ltl::unabbreviate_ltl_visitor</a> (Clone and rewrite a <a href="#">formula</a> to remove most of the abbreviated LTL and logical operators )	450
<a href="#">spot::ltl::unop</a> (Unary operators )	453
<a href="#">spot::unsigned_statistics</a>	459
<a href="#">spot::unsigned_statistics_copy</a> (Comparable statistics )	460
<a href="#">spot::ltl::visitor</a> (Formula <a href="#">visitor</a> that can modify the <a href="#">formula</a> )	461
<a href="#">spot::weight</a> (Manage for a given automaton a vector of counter indexed by its acceptance condition )	463

## 7 spot File Index

### 7.1 spot File List

Here is a list of all files with brief descriptions:

<a href="#">gspn/common.hh</a>	465
<a href="#">gspn/gspn.hh</a>	466
<a href="#">gspn/ssp.hh</a>	467
<a href="#">evtgba/evtgba.hh</a>	468
<a href="#">evtgba/evtgbaiter.hh</a>	469
<a href="#">evtgba/explicit.hh</a>	469
<a href="#">evtgba/product.hh</a>	470
<a href="#">evtgba/symbol.hh</a>	471
<a href="#">evtgbaalgorithms/dotty.hh</a>	472
<a href="#">evtgbaalgorithms/reachiter.hh</a>	474
<a href="#">evtgbaalgorithms/save.hh</a>	476
<a href="#">evtgbaalgorithms/tgba2evtgba.hh</a>	477
<a href="#">evtgbaparse/public.hh</a>	477
<a href="#">ltlast/allnodes.hh</a> (Define all LTL node types )	482
<a href="#">ltlast/atomic_prop.hh</a> (LTL atomic propositions )	482
<a href="#">ltlast/binop.hh</a> (LTL binary operators )	483
<a href="#">ltlast/constant.hh</a> (LTL constants )	484
<a href="#">ltlast/formula.hh</a> (LTL formula interface )	485
<a href="#">ltlast/multop.hh</a> (LTL multi-operand operators )	487
<a href="#">ltlast/predecl.hh</a> (Predeclare all LTL node types )	488
<a href="#">ltlast/refformula.hh</a> (Reference-counted LTL formulae )	489
<a href="#">ltlast/unop.hh</a> (LTL unary operators )	489
<a href="#">ltlast/visitor.hh</a> (LTL visitor interface )	490
<a href="#">ltlenv/declenv.hh</a>	491
<a href="#">ltlenv/defaultenv.hh</a>	492
<a href="#">ltlenv/environment.hh</a>	492

<a href="#">ltlenv/rodeco.hh</a>	493
<a href="#">ltlparse/location.hh</a>	494
<a href="#">ltlparse/position.hh</a>	496
<a href="#">ltlparse/public.hh</a>	478
<a href="#">ltlparse/stack.hh</a>	498
<a href="#">ltlvisit/apcollect.hh</a>	499
<a href="#">ltlvisit/basicreduce.hh</a>	499
<a href="#">ltlvisit/clone.hh</a>	500
<a href="#">ltlvisit/contain.hh</a>	501
<a href="#">ltlvisit/destroy.hh</a>	502
<a href="#">ltlvisit/dotty.hh</a>	473
<a href="#">ltlvisit/dump.hh</a>	503
<a href="#">ltlvisit/length.hh</a>	503
<a href="#">ltlvisit/lunabbrev.hh</a>	504
<a href="#">ltlvisit/nenoform.hh</a>	504
<a href="#">ltlvisit/postfix.hh</a>	505
<a href="#">ltlvisit/randomltl.hh</a>	505
<a href="#">ltlvisit/reduce.hh</a>	506
<a href="#">ltlvisit/simpfg.hh</a>	507
<a href="#">ltlvisit/syntimpl.hh</a>	508
<a href="#">ltlvisit/tostring.hh</a>	508
<a href="#">ltlvisit/tunabbrev.hh</a>	509
<a href="#">misc/bareword.hh</a>	509
<a href="#">misc/bddalloc.hh</a>	510
<a href="#">misc/bddlt.hh</a>	511
<a href="#">misc/escape.hh</a>	511
<a href="#">misc/freelist.hh</a>	512
<a href="#">misc/hash.hh</a>	513
<a href="#">misc/hashfunc.hh</a>	514



<a href="#">misc/ltstr.hh</a>	514
<a href="#">misc/memusage.hh</a>	515
<a href="#">misc/minato.hh</a>	515
<a href="#">misc/modgray.hh</a>	516
<a href="#">misc/optionmap.hh</a>	516
<a href="#">misc/random.hh</a>	517
<a href="#">misc/timer.hh</a>	518
<a href="#">misc/version.hh</a>	519
<a href="#">sautparse/location.hh</a>	495
<a href="#">sautparse/position.hh</a>	497
<a href="#">sautparse/stack.hh</a>	498
<a href="#">tgba/bdddict.hh</a>	519
<a href="#">tgba/bddprint.hh</a>	520
<a href="#">tgba/formula2bdd.hh</a>	522
<a href="#">tgba/public.hh</a>	480
<a href="#">tgba/state.hh</a>	522
<a href="#">tgba/statebdd.hh</a>	524
<a href="#">tgba/succiter.hh</a>	524
<a href="#">tgba/succiterconcrete.hh</a>	525
<a href="#">tgba/tgba.hh</a>	526
<a href="#">tgba/tgababddconcrete.hh</a>	528
<a href="#">tgba/tgababddconcretefactory.hh</a>	529
<a href="#">tgba/tgababddconcreteproduct.hh</a>	529
<a href="#">tgba/tgababddcoredata.hh</a>	530
<a href="#">tgba/tgababddfactory.hh</a>	531
<a href="#">tgba/tgbaexplicit.hh</a>	532
<a href="#">tgba/tgabproduct.hh</a>	534
<a href="#">tgba/tgbareduc.hh</a>	534
<a href="#">tgba/tgbatba.hh</a>	536

<a href="#">tgbaalgos/bfssteps.hh</a>	536
<a href="#">tgbaalgos/dotty.hh</a>	473
<a href="#">tgbaalgos/dottydec.hh</a>	537
<a href="#">tgbaalgos/duexp.hh</a>	538
<a href="#">tgbaalgos/emptiness.hh</a>	538
<a href="#">tgbaalgos/emptiness_stats.hh</a>	540
<a href="#">tgbaalgos/gv04.hh</a>	547
<a href="#">tgbaalgos/lbtt.hh</a>	547
<a href="#">tgbaalgos/ltl2tgba_fm.hh</a>	548
<a href="#">tgbaalgos/ltl2tgba_lacim.hh</a>	549
<a href="#">tgbaalgos/magic.hh</a>	549
<a href="#">tgbaalgos/neverclaim.hh</a>	550
<a href="#">tgbaalgos/powerset.hh</a>	551
<a href="#">tgbaalgos/projrun.hh</a>	552
<a href="#">tgbaalgos/randomgraph.hh</a>	553
<a href="#">tgbaalgos/reachiter.hh</a>	475
<a href="#">tgbaalgos/reducerun.hh</a>	553
<a href="#">tgbaalgos/reductgba_sim.hh</a>	553
<a href="#">tgbaalgos/replayrun.hh</a>	555
<a href="#">tgbaalgos/rundotdec.hh</a>	556
<a href="#">tgbaalgos/save.hh</a>	476
<a href="#">tgbaalgos/se05.hh</a>	556
<a href="#">tgbaalgos/stats.hh</a>	557
<a href="#">tgbaalgos/tau03.hh</a>	558
<a href="#">tgbaalgos/tau03opt.hh</a>	558
<a href="#">tgbaalgos/weight.hh</a>	559
<a href="#">tgbaalgos/gtec/ce.hh</a>	541
<a href="#">tgbaalgos/gtec/explscg.hh</a>	542
<a href="#">tgbaalgos/gtec/gtec.hh</a>	543

<a href="#">tgbaalgos/gtec/nsheap.hh</a>	544
<a href="#">tgbaalgos/gtec/scstack.hh</a>	545
<a href="#">tgbaalgos/gtec/status.hh</a>	546
<a href="#">tgbaparse/public.hh</a>	480

## 8 spot Page Index

### 8.1 spot Related Pages

Here is a list of all related documentation pages:

<a href="#">Bug List</a>	559
--------------------------	-----

## 9 spot Module Documentation

### 9.1 LTL formulae

#### Modules

- [Essential LTL types](#)
- [LTL Abstract Syntax Tree](#)
- [LTL environments](#)
- [Algorithms for LTL formulae](#)

#### 9.1.1 Detailed Description

This module gathers types and definitions related to LTL formulae.

### 9.2 TGBA (Transition-based Generalized Büchi Automata)

#### Modules

- [Essential TGBA types](#)
- [TGBA representations](#)
- [TGBA algorithms](#)

#### 9.2.1 Detailed Description

Spot is centered around the `spot::tgba` type. This type and its cousins are listed [here](#). This is an abstract interface. Its implementations are either [concrete representations](#), or [on-the-fly algorithms](#). Other algorithms that work on `spot::tgba` are [listed separately](#).

## 9.3 Emptiness-check algorithms for SSP

### Functions

- `couvreur99_check * spot::couvreur99_check_ssp_semi (const tgba *ssp_automata)`
- `couvreur99_check * spot::couvreur99_check_ssp_shy_semi (const tgba *ssp_automata)`
- `couvreur99_check * spot::couvreur99_check_ssp_shy (const tgba *ssp_automata, bool stack_inclusion=true)`

### 9.3.1 Function Documentation

**9.3.1.1** `couvreur99_check* spot::couvreur99_check_ssp_semi (const tgba * ssp_automata)`

**9.3.1.2** `couvreur99_check* spot::couvreur99_check_ssp_shy (const tgba * ssp_automata, bool stack_inclusion = true)`

**9.3.1.3** `couvreur99_check* spot::couvreur99_check_ssp_shy_semi (const tgba * ssp_automata)`

## 9.4 Essential LTL types

### Classes

- class `spot::ltl::formula`  
*An LTL formula.*
- struct `spot::ltl::visitor`  
*Formula visitor that can modify the formula.*
- class `spot::ltl::environment`  
*An environment that describes atomic propositions.*

### Functions

- `formula * spot::ltl::clone (const formula *f)`  
*Clone a formula.*
- `void spot::ltl::destroy (const formula *f)`  
*Destroys a formula.*

### 9.4.1 Function Documentation

**9.4.1.1** `formula* spot::ltl::clone (const formula *f)`

Clone a `formula`.

#### 9.4.1.2 void spot::ltl::destroy (const formula \*f)

Destroys a [formula](#).

## 9.5 LTL Abstract Syntax Tree

### Classes

- class [spot::ltl::atomic\\_prop](#)  
*Atomic propositions.*
- class [spot::ltl::binop](#)  
*Binary operator.*
- class [spot::ltl::constant](#)  
*A [constant](#) (True or False).*
- class [spot::ltl::formula](#)  
*An LTL [formula](#).*
- class [spot::ltl::multop](#)  
*Multi-operand operators.*
- class [spot::ltl::ref\\_formula](#)  
*A reference-counted LTL [formula](#).*
- class [spot::ltl::unop](#)  
*Unary operators.*

## 9.6 LTL environments

### Classes

- class [spot::ltl::declarative\\_environment](#)  
*A declarative [environment](#).*
- class [spot::ltl::default\\_environment](#)  
*A laxist [environment](#).*
- class [spot::ltl::read\\_only\\_environment](#)  
*A read only [environment](#).*

#### 9.6.1 Detailed Description

LTL [environment](#) implementations.

## 9.7 Algorithms for LTL formulae

### Modules

- [Input/Output of LTL formulae](#)
- [Derivable visitors](#)
- [Rewriting LTL formulae](#)
- [Miscellaneous algorithms for LTL formulae](#)

## 9.8 Input/Output of LTL formulae

### Classes

- class [spot::ltl::random\\_ltl](#)  
*Generate random LTL formulae.*

### Typedefs

- typedef std::pair< [ltl::location](#), std::string > [spot::ltl::parse\\_error](#)  
*A parse diagnostic with its location.*
- typedef std::list< parse\_error > [spot::ltl::parse\\_error\\_list](#)  
*A list of parser diagnostics, as filled by parse.*

### Functions

- formula \* [spot::ltl::parse](#) (const std::string &ltl\_string, parse\_error\_list &error\_list, environment &env=default\_environment::instance(), bool debug=false)  
*Build a [formula](#) from an LTL string.*
- bool [spot::ltl::format\\_parse\\_errors](#) (std::ostream &os, const std::string &ltl\_string, parse\_error\_list &error\_list)  
*Format diagnostics produced by [spot::ltl::parse](#).*
- std::ostream & [spot::ltl::dotty](#) (std::ostream &os, const formula \*f)  
*Write a [formula](#) tree using dot's syntax.*
- std::ostream & [spot::ltl::dump](#) (std::ostream &os, const formula \*f)  
*Dump a [formula](#) tree.*
- std::ostream & [spot::ltl::to\\_string](#) (const formula \*f, std::ostream &os)  
*Output a [formula](#) as a (parsable) string.*
- std::string [spot::ltl::to\\_string](#) (const formula \*f)  
*Convert a [formula](#) into a (parsable) string.*
- std::ostream & [spot::ltl::to\\_spin\\_string](#) (const formula \*f, std::ostream &os)

Output a *formula* as a (parsable by Spin) string.

- `std::string spot::ltl::to_spin_string (const formula *f)`  
Convert a *formula* into a (parsable by Spin) string.

### 9.8.1 Typedef Documentation

#### 9.8.1.1 `typedef std::pair<ltl::location, std::string> spot::ltl::parse_error`

A parse diagnostic with its location.

#### 9.8.1.2 `typedef std::list<parse_error> spot::ltl::parse_error_list`

A list of parser diagnostics, as filled by parse.

### 9.8.2 Function Documentation

#### 9.8.2.1 `std::ostream& spot::ltl::dotty (std::ostream & os, const formula *f)`

Write a *formula* tree using dot's syntax.

##### Parameters:

- os* The stream where it should be output.
- f* The *formula* to translate.

dot is part of the GraphViz package <http://www.research.att.com/sw/tools/graphviz/>

#### 9.8.2.2 `std::ostream& spot::ltl::dump (std::ostream & os, const formula *f)`

Dump a *formula* tree.

##### Parameters:

- os* The stream where it should be output.
- f* The *formula* to dump.

This is useful to display a *formula* when debugging.

#### 9.8.2.3 `bool spot::ltl::format_parse_errors (std::ostream & os, const std::string & ltl_string, parse_error_list & error_list)`

Format diagnostics produced by `spot::ltl::parse`.

##### Parameters:

- os* Where diagnostics should be output.
- ltl\_string* The string that were parsed.
- error\_list* The error list filled by `spot::ltl::parse` while parsing *ltl\_string*.

##### Returns:

- `true` iff any diagnostic was output.

**9.8.2.4** `formula* spot::ltl::parse (const std::string & ltl_string, parse_error_list & error_list, environment & env = default_environment::instance(), bool debug = false)`

Build a [formula](#) from an LTL string.

**Parameters:**

*ltl\_string* The string to parse.

*error\_list* A list that will be filled with parse errors that occurred during parsing.

*env* The [environment](#) into which parsing should take place.

*debug* When true, causes the parser to trace its execution.

**Returns:**

A pointer to the [formula](#) built from *ltl\_string*, or 0 if the input was unparseable.

Note that the parser usually tries to recover from errors. It can return a non zero value even if it encountered an error during the parsing of *ltl\_string*. If you want to make sure *ltl\_string* was parsed successfully, check *error\_list* for emptiness.

**Warning:**

This function is not reentrant.

**9.8.2.5** `std::string spot::ltl::to_spin_string (const formula * f)`

Convert a [formula](#) into a (parsable by Spin) string.

**Parameters:**

*f* The [formula](#) to translate.

**9.8.2.6** `std::ostream& spot::ltl::to_spin_string (const formula * f, std::ostream & os)`

Output a [formula](#) as a (parsable by Spin) string.

**Parameters:**

*f* The [formula](#) to translate.

*os* The stream where it should be output.

**9.8.2.7** `std::string spot::ltl::to_string (const formula * f)`

Convert a [formula](#) into a (parsable) string.

**Parameters:**

*f* The [formula](#) to translate.



### 9.8.2.8 `std::ostream& spot::ltl::to_string (const formula *f, std::ostream & os)`

Output a `formula` as a (parsable) string.

#### Parameters:

- f* The `formula` to translate.
- os* The stream where it should be output.

## 9.9 Derivable visitors

### Classes

- class `spot::ltl::clone_visitor`  
*Clone a `formula`.*
- class `spot::ltl::unabbreviate_logic_visitor`  
*Clone and rewrite a `formula` to remove most of the abbreviated logical operators.*
- class `spot::ltl::postfix_visitor`  
*Apply an algorithm on each node of an AST, during a postfix traversal.*
- class `spot::ltl::simplify_f_g_visitor`  
*Replace `true`  $\cup$  `f` and `false`  $\cap$  `g` by `F` `f` and `G` `g`.*
- class `spot::ltl::unabbreviate_ltl_visitor`  
*Clone and rewrite a `formula` to remove most of the abbreviated LTL and logical operators.*

## 9.10 Rewriting LTL formulae

### Enumerations

- enum `spot::ltl::reduce_options` {  
`spot::ltl::Reduce_None` = 0, `spot::ltl::Reduce_Basics` = 1, `spot::ltl::Reduce_Syntactic_Implications` = 2, `spot::ltl::Reduce_Eventuality_And_Universality` = 4,  
`spot::ltl::Reduce_Containment_Checks` = 8, `spot::ltl::Reduce_Containment_Checks_Stronger` = 16,  
`spot::ltl::Reduce_All` = -1U }  
*Options for `spot::ltl::reduce`.*

### Functions

- `formula * spot::ltl::basic_reduce (const formula *f)`  
*Basic rewritings.*
- `formula * spot::ltl::unabbreviate_logic (const formula *f)`  
*Clone and rewrite a `formula` to remove most of the abbreviated logical operators.*
- `formula * spot::ltl::negative_normal_form (const formula *f, bool negated=false)`

*Build the negative normal form of  $f$ .*

- `formula * spot::ltl::reduce (const formula *f, int opt=Reduce_All)`

*Reduce a [formula](#)  $f$ .*

- `formula * spot::ltl::simplify_f_g (const formula *f)`

*Replace `true`  $\cup f$  and `false`  $\cap g$  by `F f` and `G g`.*

### 9.10.1 Enumeration Type Documentation

#### 9.10.1.1 enum `spot::ltl::reduce_options`

Options for [spot::ltl::reduce](#).

##### Enumerator:

***Reduce\_None*** No reduction.

***Reduce\_Basics*** Basic reductions.

***Reduce\_Syntactic\_Implications*** Somenzi & Bloem syntactic implication.

***Reduce\_Eventuality\_And\_Universality*** Etessami & Holzmann eventuality and universality reductions.

***Reduce\_Containment\_Checks*** Tauriainen containment checks.

***Reduce\_Containment\_Checks\_Stronger*** Tauriainen containment checks (stronger version).

***Reduce\_All*** All reductions.

### 9.10.2 Function Documentation

#### 9.10.2.1 `formula* spot::ltl::basic_reduce (const formula *f)`

Basic rewritings.

#### 9.10.2.2 `formula* spot::ltl::negative_normal_form (const formula *f, bool negated = false)`

Build the negative normal form of  $f$ .

All negations of the [formula](#) are pushed in front of the atomic propositions.

##### Parameters:

***f*** The [formula](#) to normalize.

***negated*** If `true`, return the negative normal form of  $\neg f$

Note that this will not remove abbreviated operators. If you want to remove abbreviations, call [spot::ltl::unabbreviate\\_logic](#) or [spot::ltl::unabbreviate\\_ltl](#) first. (Calling these functions after [spot::ltl::negative\\_normal\\_form](#) would likely produce a [formula](#) which is not in negative normal form.)

**9.10.2.3 formula\* spot::ltl::reduce (const formula \*f, int opt = Reduce\_All)**

Reduce a [formula](#) *f*.

**Parameters:**

*f* the [formula](#) to reduce

*opt* a conjunction of [spot::ltl::reduce\\_options](#) specifying which optimizations to apply.

**Returns:**

the reduced [formula](#)

**9.10.2.4 formula\* spot::ltl::simplify\_f\_g (const formula \*f)**

Replace `true U f` and `false R g` by `F f` and `G g`.

**9.10.2.5 formula\* spot::ltl::unabbreviate\_logic (const formula \*f)**

Clone and rewrite a [formula](#) to remove most of the abbreviated logical operators.

This will rewrite binary operators such as [binop::Implies](#), [binop::Equals](#), and [binop::Xor](#), using only [unop::Not](#), [multop::Or](#), and [multop::And](#).

**9.11 Miscellaneous algorithms for LTL formulae****Typedefs**

- `typedef std::set< atomic_prop *, formula_ptr_less_than > spot::ltl::atomic_prop_set`  
*Set of atomic propositions.*

**Functions**

- `atomic_prop_set * spot::ltl::atomic_prop_collect (const formula *f, atomic_prop_set *s=0)`  
*Return the set of atomic propositions occurring in a [formula](#).*
- `bool spot::ltl::is_GF (const formula *f)`  
*Whether a [formula](#) starts with GF.*
- `bool spot::ltl::is_FG (const formula *f)`  
*Whether a [formula](#) starts with FG.*
- `int spot::ltl::length (const formula *f)`  
*Compute the length of a [formula](#).*
- `bool spot::ltl::is_eventual (const formula *f)`  
*Check whether a [formula](#) is a pure eventuality.*
- `bool spot::ltl::is_universal (const formula *f)`  
*Check whether a [formula](#) is purely universal.*

- bool `spot::ltl::syntactic_implication` (const formula \*f1, const formula \*f2)  
*Syntactic implication.*
- bool `spot::ltl::syntactic_implication_neg` (const formula \*f1, const formula \*f2, bool right)  
*Syntactic implication.*

### 9.11.1 Typedef Documentation

#### 9.11.1.1 typedef std::set<atomic\_prop\*, formula\_ptr\_less\_than> spot::ltl::atomic\_prop\_set

Set of atomic propositions.

### 9.11.2 Function Documentation

#### 9.11.2.1 atomic\_prop\_set\* spot::ltl::atomic\_prop\_collect (const formula \*f, atomic\_prop\_set \*s = 0)

Return the set of atomic propositions occurring in a [formula](#).

##### Parameters:

*f* the [formula](#) to inspect

*s* an existing set to fill with atomic propositions discovered, or 0 if the set should be allocated by the function.

##### Returns:

A pointer to the supplied set, *s*, augmented with atomic propositions occurring in *f*; or a newly allocated set containing all these atomic propositions if *s* is 0.

#### 9.11.2.2 bool spot::ltl::is\_eventual (const formula \*f)

Check whether a [formula](#) is a pure eventuality.

Pure eventuality formulae are defined in

```

/// @InProceedings{ etessami.00.concur,
/// author   = {Kousha Etessami and Gerard J. Holzmann},
/// title    = {Optimizing {B\^u}chi Automata},
/// booktitle = {Proceedings of the 11th International Conference on
///   Concurrency Theory (Concur'2000)},
/// pages    = {153--167},
/// year     = {2000},
/// editor   = {C. Palamidessi},
/// volume   = {1877},
/// series   = {Lecture Notes in Computer Science},
/// publisher = {Springer-Verlag}
/// }
///

```

A word that satisfies a pure eventuality can be prefixed by anything and still satisfies the [formula](#).

**9.11.2.3 bool spot::ltl::is\_FG (const formula \*f)**

Whether a [formula](#) starts with FG.

**9.11.2.4 bool spot::ltl::is\_GF (const formula \*f)**

Whether a [formula](#) starts with GF.

**9.11.2.5 bool spot::ltl::is\_universal (const formula \*f)**

Check whether a [formula](#) is purely universal.

Purely universal formulae are defined in

```

/// @InProceedings{ etessami.00.concur,
/// author   = {Kousha Etessami and Gerard J. Holzmann},
/// title    = {Optimizing {B\"u}chi Automata},
/// booktitle = {Proceedings of the 11th International Conference on
///               Concurrency Theory (Concur'2000)},
/// pages    = {153--167},
/// year     = {2000},
/// editor    = {C. Palamidessi},
/// volume   = {1877},
/// series    = {Lecture Notes in Computer Science},
/// publisher = {Springer-Verlag}
/// }
///

```

Any (non-empty) suffix of a word that satisfies if purely universal [formula](#) also satisfies the [formula](#).

**9.11.2.6 int spot::ltl::length (const formula \*f)**

Compute the length of a [formula](#).

The length of a [formula](#) is the number of atomic properties, constants, and operators (logical and temporal) occurring in the [formula](#). spot::ltl::multops count only for 1, even if they have more than two operands (e.g.  $a \mid b \mid c$  has length 4, because  $\mid$  is represented once internally).

**9.11.2.7 bool spot::ltl::syntactic\_implication (const formula \*f1, const formula \*f2)**

Syntactic implication.

This comes from

```

/// @InProceedings{ somenzi.00.cav,
/// author   = {Fabio Somenzi and Roderick Bloem},
/// title    = {Efficient {B\"u}chi Automata for {LTL} Formulae},
/// booktitle = {Proceedings of the 12th International Conference on
///               Computer Aided Verification (CAV'00)},
/// pages    = {247--263},
/// year     = {2000},
/// volume   = {1855},
/// series    = {Lecture Notes in Computer Science},
/// publisher = {Springer-Verlag}
/// }
///

```

**9.11.2.8 bool spot::ltl::syntactic\_implication\_neg (const formula \*f1, const formula \*f2, bool right)**

Syntactic implication.

If right==false, true if !f1 < f2, false otherwise. If right==true, true if f1 < !f2, false otherwise.

See also:

[syntactic\\_implication](#)

**9.12 Miscellaneous helper algorithms**

Whether a word is bare.

**Modules**

- [Hashing functions](#)
- [Random functions](#)

**Classes**

- class [spot::bdd\\_allocator](#)  
*Manage ranges of variables.*
- struct [spot::bdd\\_less\\_than](#)  
*Comparison functor for BDDs.*
- class [spot::free\\_list](#)  
*Manage list of free integers.*
- struct [spot::char\\_ptr\\_less\\_than](#)  
*Strict Weak Ordering for char\*.*
- class [spot::minato\\_isop](#)  
*Generate an irredundant sum-of-products (ISOP) form of a BDD function.*
- class [spot::loopless\\_modular\\_mixed\\_radix\\_gray\\_code](#)  
*Loopless modular mixed radix Gray code iteration.*
- class [spot::option\\_map](#)  
*Manage a map of options.*
- struct [spot::time\\_info](#)  
*A structure to record elapsed time in clock ticks.*
- class [spot::timer](#)  
*A timekeeper that accumulate interval of time.*
- class [spot::timer\\_map](#)  
*A map of [timer](#), where each [timer](#) has a name.*

## Functions

- bool [spot::is\\_bare\\_word](#) (const char \*str)
  - std::string [spot::quote\\_unless\\_bare\\_word](#) (const std::string &str)  
*Double-quote words that are not bare.*
- std::ostream & [spot::escape\\_str](#) (std::ostream &os, const std::string &str)  
*Escape " and \ characters in str.*
- std::string [spot::escape\\_str](#) (const std::string &str)  
*Escape " and \ characters in str.*
- const char \* [spot::version](#) ()  
*Return Spot's version.*

### 9.12.1 Detailed Description

Whether a word is bare.

Bare words should start with a letter or an underscore, and consist solely of alphanumeric characters and underscores.

### 9.12.2 Function Documentation

#### 9.12.2.1 std::string spot::escape\_str (const std::string & str)

Escape " and \ characters in *str*.

#### 9.12.2.2 std::ostream& spot::escape\_str (std::ostream & os, const std::string & str)

Escape " and \ characters in *str*.

#### 9.12.2.3 bool spot::is\_bare\_word (const char \* str)

#### 9.12.2.4 std::string spot::quote\_unless\_bare\_word (const std::string & str)

Double-quote words that are not bare.

See also:

[is\\_bare\\_word](#)

#### 9.12.2.5 const char\* spot::version ()

Return Spot's version.

## 9.13 Hashing functions

### Classes

- struct `spot::ltd::formula_ptr_hash`  
*Hash Function for `const formula*`.*
- struct `spot::ptr_hash< T >`  
*A hash function for pointers.*
- struct `spot::string_hash`  
*A hash function for strings.*
- struct `spot::state_ptr_hash`  
*Hash Function for `state*`.*

### Functions

- `size_t spot::wang32_hash (size_t key)`  
*Thomas Wang's 32 bit hash function.*
- `size_t spot::knuth32_hash (size_t key)`  
*Knuth's Multiplicative hash function.*

#### 9.13.1 Function Documentation

##### 9.13.1.1 `size_t spot::knuth32_hash (size_t key)` [inline]

Knuth's Multiplicative hash function.

This function is suitable for hashing values whose high order bits do not vary much (ex. addresses of memory objects). Prefer `spot::wang32_hash()` otherwise.  
<http://www.concentric.net/~Ttwang/tech/addrhash.htm>

##### 9.13.1.2 `size_t spot::wang32_hash (size_t key)` [inline]

Thomas Wang's 32 bit hash function.

Hash an integer amongst the integers. <http://www.concentric.net/~Ttwang/tech/inthash.htm>

## 9.14 Random functions

### Classes

- class `spot::barand< gen >`  
*Compute pseudo-random integer value between 0 and n included, following a binomial distribution for probability p.*



## Functions

- void `spot::srand` (unsigned int seed)  
*Reset the seed of the pseudo-random number generator.*
- int `spot::rrand` (int min, int max)  
*Compute a pseudo-random integer value between min and max included.*
- int `spot::mrand` (int max)  
*Compute a pseudo-random integer value between 0 and max-1 included.*
- double `spot::drand` ()  
*Compute a pseudo-random double value between 0.0 and 1.0 (1.0 excluded).*
- double `spot::nrand` ()  
*Compute a pseudo-random double value following a standard normal distribution. (Odeh & Evans).*
- double `spot::bmrnd` ()  
*Compute a pseudo-random double value following a standard normal distribution. (Box-Muller).*
- int `spot::prand` (double p)  
*Return a pseudo-random positive integer value following a Poisson distribution with parameter p.*

### 9.14.1 Function Documentation

#### 9.14.1.1 double `spot::bmrnd` ()

Compute a pseudo-random double value following a standard normal distribution. (Box-Muller).

This uses the polar form of the Box-Muller transform to generate random values.

#### 9.14.1.2 double `spot::drand` ()

Compute a pseudo-random double value between 0.0 and 1.0 (1.0 excluded).

See also:

`mrand`, `rrand`, `srand`

#### 9.14.1.3 int `spot::mrand` (int *max*)

Compute a pseudo-random integer value between 0 and *max-1* included.

See also:

`drand`, `rrand`, `srand`

#### 9.14.1.4 double `spot::nrand` ()

Compute a pseudo-random double value following a standard normal distribution. (Odeh & Evans).

This uses a polynomial approximation of the inverse cumulated density function from Odeh & Evans, Journal of Applied Statistics, 1974, vol 23, pp 96-97.

#### 9.14.1.5 `int spot::prand (double p)`

Return a pseudo-random positive integer value following a Poisson distribution with parameter *p*.

**Precondition:**

$$p > 0$$

#### 9.14.1.6 `int spot::rrand (int min, int max)`

Compute a pseudo-random integer value between *min* and *max* included.

**See also:**

[drand](#), [mrand](#), [srand](#)

#### 9.14.1.7 `void spot::srand (unsigned int seed)`

Reset the seed of the pseudo-random number generator.

**See also:**

[drand](#), [mrand](#), [rrand](#)

## 9.15 Essential TGBA types

### Classes

- class [spot::bdd\\_dict](#)
- class [spot::state](#)  
*Abstract class for states.*
- struct [spot::state\\_ptr\\_less\\_than](#)  
*Strict Weak Ordering for state\*.*
- struct [spot::state\\_ptr\\_equal](#)  
*An Equivalence Relation for state\*.*
- struct [spot::state\\_ptr\\_hash](#)  
*Hash Function for state\*.*
- class [spot::tgba\\_succ\\_iterator](#)  
*Iterate over the successors of a [state](#).*
- class [spot::tgba](#)  
*A Transition-based Generalized Büchi Automaton.*

## 9.16 TGBA representations

### Classes

- class `spot::state_bdd`
- class `spot::tgb succ_iterator_concrete`
- class `spot::tgb bdd_concrete`  
*A concrete `spot::tgb` implemented using BDDs.*
- class `spot::tgb explicit`
- class `spot::state_explicit`
- class `spot::tgb explicit_succ_iterator`
- class `spot::tgb_reduc`

## 9.17 TGBA algorithms

### Modules

- TGBA on-the-fly algorithms
- Input/Output of TGBA
- Translating LTL formulae into TGBA
- Algorithm patterns
- TGBA simplifications
- Miscellaneous algorithms on TGBA
- Emptiness-checks

### Functions

- `tgb bdd_concrete * spot::product (const tgb bdd_concrete *left, const tgb bdd_concrete *right)`  
*Multiplies two `spot::tgb bdd_concrete` automata.*

### 9.17.1 Function Documentation

#### 9.17.1.1 `tgb bdd_concrete* spot::product (const tgb bdd_concrete * left, const tgb bdd_concrete * right)`

Multiplies two `spot::tgb bdd_concrete` automata.

This function builds the resulting product as another `spot::tgb bdd_concrete` automaton.

## 9.18 TGBA on-the-fly algorithms

### Classes

- class `spot::state_product`  
*A state for `spot::tgb product`.*
- class `spot::tgb_tba_proxy`

*Degeneralize a `spot::tgba` on the fly, producing a TBA.*

- class `spot::tgba_sba_proxy`  
*Degeneralize a `spot::tgba` on the fly, producing an SBA.*

## 9.19 Input/Output of TGBA

### Modules

- `Decorating the dot output`

### Typedefs

- typedef `std::pair< tgba::location, std::string >` `spot::tgba_parse_error`  
*A parse diagnostic with its location.*
- typedef `std::list< tgba_parse_error >` `spot::tgba_parse_error_list`  
*A list of parser diagnostics, as filled by parse.*

### Functions

- `std::ostream & spot::dotty_reachable` (`std::ostream &os`, `const tgba *g`, `dotty_decorator *dd=dotty_decorator::instance()`)  
*Print reachable states in dot format.*
- `std::ostream & spot::lbt_reachable` (`std::ostream &os`, `const tgba *g`)  
*Print reachable states in LBTT format.*
- `std::ostream & spot::never_claim_reachable` (`std::ostream &os`, `const tgba_sba_proxy *g`, `const ltl::formula *f=0`)  
*Print reachable states in Spin never claim format.*
- `std::ostream & spot::tgba_save_reachable` (`std::ostream &os`, `const tgba *g`)  
*Save reachable states in text format.*
- `tgba_explicit * spot::tgba_parse` (`const std::string &filename`, `tgba_parse_error_list &error_list`, `bdd_dict *dict`, `ltl::environment &env=ltl::default_environment::instance()`, `ltl::environment &envacc=ltl::default_environment::instance()`, `bool debug=false`)  
*Build a `spot::tgba_explicit` from a text file.*
- `bool spot::format_tgba_parse_errors` (`std::ostream &os`, `const std::string &filename`, `tgba_parse_error_list &error_list`)  
*Format diagnostics produced by `spot::tgba_parse`.*

### 9.19.1 Typedef Documentation

#### 9.19.1.1 `typedef std::pair<tgba::location, std::string> spot::tgba_parse_error`

A parse diagnostic with its location.

#### 9.19.1.2 `typedef std::list<tgba_parse_error> spot::tgba_parse_error_list`

A list of parser diagnostics, as filled by `parse`.

### 9.19.2 Function Documentation

#### 9.19.2.1 `std::ostream& spot::dotty_reachable (std::ostream & os, const tgba * g, dotty_decorator * dd = dotty_decorator::instance())`

Print reachable states in dot format.

The `dd` argument allows to customize the output in various ways. See [this page](#) for a list of available decorators.

#### 9.19.2.2 `bool spot::format_tgba_parse_errors (std::ostream & os, const std::string & filename, tgba_parse_error_list & error_list)`

Format diagnostics produced by [spot::tgba\\_parse](#).

##### Parameters:

*os* Where diagnostics should be output.

*filename* The filename that should appear in the diagnostics.

*error\_list* The error list filled by [spot::ltl::parse](#) while parsing *ltl\_string*.

##### Returns:

`true` iff any diagnostic was output.

#### 9.19.2.3 `std::ostream& spot::lbtt_reachable (std::ostream & os, const tgba * g)`

Print reachable states in LBTT format.

##### Parameters:

*g* The automata to print.

*os* Where to print.

#### 9.19.2.4 `std::ostream& spot::never_claim_reachable (std::ostream & os, const tgba_sba_proxy * g, const ltl::formula * f = 0)`

Print reachable states in Spin never claim format.

##### Parameters:

*os* The output stream to print on.

*g* The degeneralized automaton to output.

*f* The (optional) formula associated to the automaton. If given it will be output as a comment.

**9.19.2.5** `tgba_explicit*` `spot::tgba_parse` (`const std::string & filename`, `tgba_parse_error_list & error_list`, `bdd_dict * dict`, `ltl::environment & env = ltl::default_environment::instance()`, `ltl::environment & envacc = ltl::default_environment::instance()`, `bool debug = false`)

Build a `spot::tgba_explicit` from a text file.

#### Parameters:

*filename* The name of the file to parse.

*error\_list* A list that will be filled with parse errors that occurred during parsing.

*dict* The BDD dictionary where to use.

*env* The environment of atomic proposition into which parsing should take place.

*envacc* The environment of acceptance conditions into which parsing should take place.

*debug* When true, causes the parser to trace its execution.

#### Returns:

A pointer to the `tgba` built from *filename*, or 0 if the file could not be opened.

Note that the parser usually tries to recover from errors. It can return a non zero value even if it encountered error during the parsing of *filename*. If you want to make sure *filename* was parsed successfully, check *error\_list* for emptiness.

#### Warning:

This function is not reentrant.

**9.19.2.6** `std::ostream& spot::tgba_save_reachable` (`std::ostream & os`, `const tgba * g`)

Save reachable states in text format.

## 9.20 Translating LTL formulae into TGBA

### Functions

- `tgba_explicit *` `spot::ltl_to_tgba_fm` (`const ltl::formula *f`, `bdd_dict *dict`, `bool exprop=false`, `bool symb_merge=true`, `bool branching_postponement=false`, `bool fair_loop_approx=false`, `const ltl::atomic_prop_set *unobs=0`, `int reduce_ltl=ltl::Reduce_None`, `bool containment_checks=false`)

Build a `spot::tgba_explicit*` from an LTL formula.

- `tgba_bdd_concrete *` `spot::ltl_to_tgba_lacim` (`const ltl::formula *f`, `bdd_dict *dict`)

Build a `spot::tgba_bdd_concrete` from an LTL formula.

### 9.20.1 Function Documentation

**9.20.1.1** `tgba_explicit*` `spot::ltl_to_tgba_fm` (`const ltl::formula *f`, `bdd_dict *dict`, `bool exprop = false`, `bool symb_merge = true`, `bool branching_postponement = false`, `bool fair_loop_approx = false`, `const ltl::atomic_prop_set * unobs = 0`, `int reduce_ltl = ltl::Reduce_None`, `bool containment_checks = false`)

Build a `spot::tgba_explicit*` from an LTL formula.

This is based on the following paper.

```

/// @InProceedings{couvreur.99.fm,
///   author   = {Jean-Michel Couvreur},
///   title    = {On-the-fly Verification of Temporal Logic},
///   pages    = {253--271},
///   editor   = {Jeannette M. Wing and Jim Woodcock and Jim Davies},
///   booktitle = {Proceedings of the World Congress on Formal Methods in the
///     Development of Computing Systems (FM'99)},
///   publisher = {Springer-Verlag},
///   series   = {Lecture Notes in Computer Science},
///   volume   = {1708},
///   year     = {1999},
///   address  = {Toulouse, France},
///   month    = {September},
///   isbn     = {3-540-66587-0}
/// }
///

```

#### Parameters:

**f** The formula to translate into an automaton.

**dict** The `spot::bdd_dict` the constructed automata should use.

**exprop** When set, the algorithm will consider all properties combinations possible on each `state`, in an attempt to reduce the non-determinism. The automaton will have the same size as without this option, but because the transition will be more deterministic, the product automaton will be smaller (or, at worse, equal).

**symb\_merge** When false, states with the same symbolic representation (these are equivalent formulae) will not be merged.

**branching\_postponement** When set, several transitions leaving from the same `state` with the same label (i.e., condition + acceptance conditions) will be merged. This correspond to an optimization described in the following paper.

```

/// @InProceedings{sebastiani.03.charme,
///   author   = {Roberto Sebastiani and Stefano Tonetta},
///   title    = {"More Deterministic" vs. "Smaller" B{\u}chi Automata for
///     Efficient LTL Model Checking},
///   booktitle = {Proceedings for the 12th Advanced Research Working
///     Conference on Correct Hardware Design and Verification
///     Methods (CHARME'03)},
///   pages    = {126--140},
///   year     = {2003},
///   editor   = {G. Goos and J. Hartmanis and J. van Leeuwen},
///   volume   = {2860},
///   series   = {Lectures Notes in Computer Science},
///   month    = {October},
///   publisher = {Springer-Verlag}
/// }
///

```

**fair\_loop\_approx** When set, a really simple characterization of unstable `state` is used to suppress all acceptance conditions from incoming transitions.

**unobs** When non-zero, the atomic propositions in the LTL formula are interpreted as events that exclude each other. The events in the formula are observable events, and `unobs` can be filled with additional unobservable events.

**reduce\_ltl** If this parameter is set, the LTL formulae representing each `state` of the automaton will be simplified using `spot::ltl::reduce()` before computing the successor. `reduce_ltl` should specify the type of reduction to apply as documented for `spot::ltl::reduce()`. This idea is taken from the following paper.

```

/// @InProceedings{ thirioux.02.fmics,
///   author   = {Xavier Thirioux},
///   title    = {Simple and Efficient Translation from {LTL} Formulas to
///               {B\"u}chi Automata},
///   booktitle = {Proceedings of the 7th International ERCIM Workshop in
///               Formal Methods for Industrial Critical Systems (FMICS'02)},
///   series   = {Electronic Notes in Theoretical Computer Science},
///   volume   = {66(2)},
///   publisher = {Elsevier},
///   editor   = {Rance Cleaveland and Hubert Garavel},
///   year     = {2002},
///   month    = jul,
///   address  = {M{\`a}laga, Spain}
/// }
///

```

**Returns:**

A [spot::tgba\\_explicit](#) that recognizes the language of  $f$ .

**9.20.1.2 tgba\_bdd\_concrete\* spot::ltl\_to\_tgba\_lacim (const ltl::formula \*f, bdd\_dict \*dict)**

Build a [spot::tgba\\_bdd\\_concrete](#) from an LTL formula.

This is based on the following paper.

```

/// @InProceedings{ couvreur.00.lacim,
///   author   = {Jean-Michel Couvreur},
///   title    = {Un point de vue symbolique sur la logique temporelle
///               lin{\`e}aire},
///   booktitle = {Actes du Colloque LaCIM 2000},
///   month    = {August},
///   year     = {2000},
///   pages    = {131--140},
///   volume   = {27},
///   series   = {Publications du LaCIM},
///   publisher = {Universit{\`e} du Qu{\`e}bec {\`a} Montr{\`e}al},
///   editor   = {Pierre Leroux}
/// }
///

```

**Parameters:**

$f$  The formula to translate into an automaton.

$dict$  The [spot::bdd\\_dict](#) the constructed automata should use.

**Returns:**

A [spot::tgba\\_bdd\\_concrete](#) that recognizes the language of  $f$ .

**9.21 Algorithm patterns****Classes**

- class [spot::tgba\\_reachable\\_iterator](#)  
Iterate over all reachable states of a [spot::tgba](#).
- class [spot::tgba\\_reachable\\_iterator\\_depth\\_first](#)  
An implementation of [spot::tgba\\_reachable\\_iterator](#) that browses states depth first.



- class `spot::tgba_reachable_iterator_breadth_first`

*An implementation of `spot::tgba_reachable_iterator` that browses states breadth first.*

## 9.22 TGBA simplifications

### Classes

- class `spot::parity_game_graph`  
*Parity game graph which compute a simulation relation.*
- class `spot::spoiler_node`  
*Spoiler node of parity game graph.*
- class `spot::duplicator_node`  
*Duplicator node of parity game graph.*
- class `spot::parity_game_graph_direct`  
*Parity game graph which compute the direct simulation relation.*
- class `spot::spoiler_node_delayed`  
*Spoiler node of parity game graph for delayed simulation.*
- class `spot::duplicator_node_delayed`  
*Duplicator node of parity game graph for delayed simulation.*
- class `spot::parity_game_graph_delayed`

### Typedefs

- typedef `std::vector< spoiler_node * >` `spot::sn_v`
- typedef `std::vector< duplicator_node * >` `spot::dn_v`
- typedef `std::vector< const state * >` `spot::s_v`

### Enumerations

- enum `spot::reduce_tgba_options` {  
`spot::Reduce_None = 0`, `spot::Reduce_quotient_Dir_Sim = 1`, `spot::Reduce_transition_Dir_Sim = 2`,  
`spot::Reduce_quotient_Del_Sim = 4`,  
`spot::Reduce_transition_Del_Sim = 8`, `spot::Reduce_Scc = 16`, `spot::Reduce_All = -1U` }  
*Options for reduce.*

## Functions

- `tgba * spot::reduc_tgba_sim` (`const tgba *a`, `int opt=Reduce_All`)  
*Remove some node of the automata using a simulation relation.*
- `direct_simulation_relation * spot::get_direct_relation_simulation` (`const tgba *a`, `std::ostream &os`, `int opt=-1`)  
*Compute a direct simulation relation on [state](#) of *tgba* f.*
- `delayed_simulation_relation * spot::get_delayed_relation_simulation` (`const tgba *a`, `std::ostream &os`, `int opt=-1`)
- `void spot::free_relation_simulation` (`direct_simulation_relation *rel`)  
*To free a simulation relation.*
- `void spot::free_relation_simulation` (`delayed_simulation_relation *rel`)  
*To free a simulation relation.*

### 9.22.1 Typedef Documentation

9.22.1.1 `typedef std::vector<duplicator_node*> spot::dn_v`

9.22.1.2 `typedef std::vector<const state*> spot::s_v`

9.22.1.3 `typedef std::vector<spoiler_node*> spot::sn_v`

### 9.22.2 Enumeration Type Documentation

9.22.2.1 `enum spot::reduce_tgba_options`

Options for reduce.

#### Enumerator:

*Reduce\_None* No reduction.

*Reduce\_quotient\_Dir\_Sim* Reduction of [state](#) using direct simulation relation.

*Reduce\_transition\_Dir\_Sim* Reduction of transitions using direct simulation relation.

*Reduce\_quotient\_Del\_Sim* Reduction of [state](#) using delayed simulation relation.

*Reduce\_transition\_Del\_Sim* Reduction of transition using delayed simulation relation.

*Reduce\_Scc* Reduction using SCC.

*Reduce\_All* All reductions.

### 9.22.3 Function Documentation

9.22.3.1 `void spot::free_relation_simulation` (`delayed_simulation_relation * rel`)

To free a simulation relation.

**9.22.3.2 void spot::free\_relation\_simulation (direct\_simulation\_relation \* rel)**

To free a simulation relation.

**9.22.3.3 delayed\_simulation\_relation\* spot::get\_delayed\_relation\_simulation (const tgba \* a, std::ostream & os, int opt = -1)**

Compute a delayed simulation relation on [state](#) of [tgba](#) *f*.

**Bug**

Does not work for generalized automata.

**9.22.3.4 direct\_simulation\_relation\* spot::get\_direct\_relation\_simulation (const tgba \* a, std::ostream & os, int opt = -1)**

Compute a direct simulation relation on [state](#) of [tgba](#) *f*.

**9.22.3.5 tgba\* spot::reduc\_tgba\_sim (const tgba \* a, int opt = Reduce\_All)**

Remove some node of the automata using a simulation relation.

**Parameters:**

*a* the automata to reduce.

*opt* a conjunction of [spot::reduce\\_tgba\\_options](#) specifying which optimizations to apply.

**Returns:**

the reduced automata.

**9.23 Miscellaneous algorithms on TGBA****Classes**

- class [spot::bfs\\_steps](#)  
*Make a BFS in a [spot::tgba](#) to compute a [tgba\\_run::steps](#).*
- struct [spot::tgba\\_statistics](#)

**Functions**

- tgba\_explicit \* [spot::tgba\\_dupexp\\_bfs](#) (const tgba \* aut)  
*Build an explicit automata from all states of aut, numbering states in bread first order as they are processed.*
- tgba\_explicit \* [spot::tgba\\_dupexp\\_dfs](#) (const tgba \* aut)  
*Build an explicit automata from all states of aut, numbering states in depth first order as they are processed.*
- tgba\_explicit \* [spot::tgba\\_powerset](#) (const tgba \* aut)  
*Build a deterministic automaton, ignoring acceptance conditions.*

- `tgba * spot::random_graph (int n, float d, const ltl::atomic_prop_set *ap, bdd_dict *dict, int n_acc=0, float a=0.1, float t=0.5, ltl::environment *env=&ltl::default_environment::instance())`

*Construct a `tgba` randomly.*

- `tgba_statistics spot::stats_reachable (const tgba *g)`

*Compute statistics for an automaton.*

### 9.23.1 Function Documentation

**9.23.1.1 `tgba* spot::random_graph (int n, float d, const ltl::atomic_prop_set * ap, bdd_dict * dict, int n_acc = 0, float a = 0.1, float t = 0.5, ltl::environment * env = &ltl::default_environment::instance())`**

Construct a `tgba` randomly.

#### Parameters:

***n*** The number of states wanted in the automata ( $>0$ ). All states will be connected, and there will be no dead `state`.

***d*** The density of the automata. This is the probability (between 0.0 and 1.0), to add a transition between two states. All states have at least one outgoing transition, so *d* is considered only when adding the remaining transition. A density of 1 means all states will be connected to each other.

***ap*** The list of atomic property that should label the transition.

***dict*** The `bdd_dict` to used for this automata.

***n\_acc*** The number of acceptance sets to use.

***a*** The probability (between 0.0 and 1.0) that a transition belongs to an acceptance set.

***t*** The probability (between 0.0 and 1.0) that an atomic proposition is true.

***env*** The environment in which to declare the acceptance conditions.

This algorithms is adapted from the one in Fig 6.2 page 48 of

```

/// @TechReport{ tauriainen.00.a66,
///   author = {Heikki Tauriainen},
///   title   = {Automated Testing of {\u}chi Automata Translators for
///             {\L}inear {\T}emporal {\L}ogic},
///   address = {Espoo, Finland},
///   institution = {Helsinki University of Technology, Laboratory for
///                 Theoretical Computer Science},
///   number = {A66},
///   year = {2000},
///   url = {http://citeseer.nj.nec.com/tauriainen00automated.html},
///   type = {Research Report},
///   note = {Reprint of Master's thesis}
/// }
///

```

Although the intent is similar, there are some differences with between the above published algorithm and this implementation . First labels are on transitions, and acceptance conditions are generated too. Second, the number of successors of a node is chosen in  $[1, n]$  following a normal distribution with mean  $1+(n-1)d$  and variance  $(n-1)d(1-d)$ . (This is less accurate, but faster than considering all possible  $n$  successors one by one.)

**9.23.1.2 tgba\_statistics spot::stats\_reachable (const tgba \* g)**

Compute statistics for an automaton.

**9.23.1.3 tgba\_explicit\* spot::tgba\_dupexp\_bfs (const tgba \* aut)**

Build an explicit automata from all states of *aut*, numbering states in bread first order as they are processed.

**9.23.1.4 tgba\_explicit\* spot::tgba\_dupexp\_dfs (const tgba \* aut)**

Build an explicit automata from all states of *aut*, numbering states in depth first order as they are processed.

**9.23.1.5 tgba\_explicit\* spot::tgba\_powerset (const tgba \* aut)**

Build a deterministic automaton, ignoring acceptance conditions.

This create a deterministic automaton that recognize the same language as *aut* would if its acceptance conditions were ignored. This is the classical powerset algorithm.

**9.24 Decorating the dot output****Classes**

- class [spot::dotty\\_decorator](#)  
*Choose [state](#) and link styles for [spot::dotty\\_reachable](#).*
- class [spot::tgba\\_run\\_dotty\\_decorator](#)  
*Highlight a [spot::tgba\\_run](#) on a [spot::tgba](#).*

**9.25 Emptiness-checks****Modules**

- [Emptiness-check algorithms for SSP](#)
- [Emptiness-check algorithms](#)
- [TGBA runs and supporting functions](#)
- [Emptiness-check statistics](#)

**Classes**

- class [spot::emptiness\\_check\\_result](#)  
*The result of an emptiness check.*
- class [spot::emptiness\\_check](#)  
*Common interface to emptiness check algorithms.*
- class [spot::emptiness\\_check\\_instantiator](#)

### 9.25.1 Detailed Description

All emptiness-check algorithms follow the same interface. Basically once you have constructed an instance of `spot::emptiness_check` (by instantiating a subclass, or calling a function to construct such instance; see [this list](#)), you should call `spot::emptiness_check::check()` to check the automaton.

If `spot::emptiness_check::check()` returns 0, then the automaton was found empty. Otherwise the automaton accepts some run. (Beware that some algorithms—those using bit-state hashing—may find the automaton to be empty even if it is not actually empty.)

When `spot::emptiness_check::check()` does not return 0, it returns an instance of `spot::emptiness_check_result`. You can try to call `spot::emptiness_check_result::accepting_run()` to obtain an accepting run. For some emptiness-check algorithms, `spot::emptiness_check_result::accepting_run()` will require some extra computation. Most emptiness-check algorithms are able to return such an accepting run, however this is not mandatory and `spot::emptiness_check_result::accepting_run()` can return 0 (this does not mean by anyway that no accepting run exist).

The acceptance run returned by `spot::emptiness_check_result::accepting_run()`, if any, is of type `spot::tgba_run`. [This page](#) gathers existing operations on these objects.

## 9.26 Emptiness-check algorithms

### Classes

- class `spot::couvreur99_check`  
*An implementation of the Couvreur99 emptiness-check algorithm.*
- class `spot::couvreur99_check_shy`  
*A version of `spot::couvreur99_check` that tries to visit known states first.*

### Functions

- `emptiness_check * spot::couvreur99` (const tgba \*a, option\_map options=option\_map(), const numbered\_state\_heap\_factory \*nshf=numbered\_state\_heap\_hash\_map\_factory::instance())  
*Check whether the language of an automate is empty.*
- `emptiness_check * spot::explicit_gv04_check` (const tgba \*a, option\_map o=option\_map())  
*Emptiness check based on Geldenhuys and Valmari's TACAS'04 paper.*
- `emptiness_check * spot::explicit_magic_search` (const tgba \*a, option\_map o=option\_map())  
*Returns an emptiness checker on the `spot::tgba` automaton a.*
- `emptiness_check * spot::bit_state_hashing_magic_search` (const tgba \*a, size\_t size, option\_map o=option\_map())  
*Returns an emptiness checker on the `spot::tgba` automaton a.*
- `emptiness_check * spot::magic_search` (const tgba \*a, option\_map o=option\_map())  
*Wrapper for the two magic\_search implementations.*
- `emptiness_check * spot::explicit_sc05_search` (const tgba \*a, option\_map o=option\_map())  
*Returns an emptiness check on the `spot::tgba` automaton a.*

- `emptiness_check * spot::bit_state_hashing_se05_search` (`const tgba *a`, `size_t size`, `option_map o=option_map()`)  
Returns an emptiness checker on the `spot::tgba` automaton `a`.
- `emptiness_check * spot::se05` (`const tgba *a`, `option_map o`)  
Wrapper for the two `se05` implementations.
- `emptiness_check * spot::explicit_tau03_search` (`const tgba *a`, `option_map o=option_map()`)  
Returns an emptiness checker on the `spot::tgba` automaton `a`.
- `emptiness_check * spot::explicit_tau03_opt_search` (`const tgba *a`, `option_map o=option_map()`)  
Returns an emptiness checker on the `spot::tgba` automaton `a`.

### 9.26.1 Function Documentation

#### 9.26.1.1 `emptiness_check* spot::bit_state_hashing_magic_search` (`const tgba * a`, `size_t size`, `option_map o = option_map()`)

Returns an emptiness checker on the `spot::tgba` automaton `a`.

##### Precondition:

The automaton `a` must have at most one acceptance condition (i.e. it is a TBA).

During the visit of `a`, the returned checker does not store explicitly the traversed states but uses the bit-state hashing technic presented in:

```
/// @book{Holzmann91,
///   author = {G.J. Holzmann},
///   title = {Design and Validation of Computer Protocols},
///   publisher = {Prentice-Hall},
///   address = {Englewood Cliffs, New Jersey},
///   year = {1991}
/// }
```

Consequently, the detection of an acceptance cycle is not ensured.

The size of the heap is limited to

size bytes.

The implemented algorithm is the same as the one of `spot::explicit_magic_search`.

##### See also:

[spot::explicit\\_magic\\_search](#)

#### 9.26.1.2 `emptiness_check* spot::bit_state_hashing_se05_search` (`const tgba *a`, `size_t size`, `option_map o = option_map()`)

Returns an emptiness checker on the `spot::tgba` automaton `a`.

**Precondition:**

The automaton  $a$  must have at most one acceptance condition (i.e. it is a TBA).

During the visit of  $a$ , the returned checker does not store explicitly the traversed states but uses the bit-state hashing technic presented in:

```
/// @book{Holzmann91,
///   author = {G.J. Holzmann},
///   title = {Design and Validation of Computer Protocols},
///   publisher = {Prentice-Hall},
///   address = {Englewood Cliffs, New Jersey},
///   year = {1991}
/// }
```

Consequently, the detection of an acceptance cycle is not ensured.

The size of the heap is limited to  
size bytes.

The implemented algorithm is the same as the one of [spot::explicit\\_se05\\_search](#).

**See also:**

[spot::explicit\\_se05\\_search](#)

**9.26.1.3 emptiness\_check\*** `spot::couvreur99 (const tgba * a, option_map options = option_map(), const numbered_state_heap_factory * nshf = numbered_state_heap_hash_map_factory::instance())`

Check whether the language of an automate is empty.

This is based on the following paper.

```
/// @InProceedings{couvreur.99.fm,
///   author = {Jean-Michel Couvreur},
///   title = {On-the-fly Verification of Temporal Logic},
///   pages = {253--271},
///   editor = {Jeannette M. Wing and Jim Woodcock and Jim Davies},
///   booktitle = {Proceedings of the World Congress on Formal Methods in
///     the Development of Computing Systems (FM'99)},
///   publisher = {Springer-Verlag},
///   series = {Lecture Notes in Computer Science},
///   volume = {1708},
///   year = {1999},
///   address = {Toulouse, France},
///   month = {September},
///   isbn = {3-540-66587-0}
/// }
```

A recursive definition of the algorithm would look as follows, but the implementation is of course not recursive. ( $\langle \Sigma, Q, \delta, q, F \rangle$  is the automaton to check,  $H$  is an associative array mapping each [state](#) to its positive DFS order or 0 if it is dead,  $SCC$  is and  $ACC$  are two stacks.)

```
/// check( $\langle \Sigma, Q, \delta, q, F \rangle$ ,  $H$ ,  $SCC$ ,  $ACC$ )
///   if  $q$  is not in  $H$  // new state
///      $H[q] = H.size + 1$ 
```



```

///      SCC.push(<H[q], {}>)
///      forall <a, s> : <q, _, a, s> in delta
///          ACC.push(a)
///          res = check(<Sigma, Q, delta, s, F>, H, SCC, ACC)
///          if res
///              return res
///      <n, _> = SCC.top()
///      if n = H[q]
///          SCC.pop()
///          mark_reachable_states_as_dead(<Sigma, Q, delta, q, F>, H$)
///      return 0
///  else
///      if H[q] = 0 // dead state
///          ACC.pop()
///          return true
///      else // state in stack: merge SCC
///          all = {}
///          do
///              <n, a> = SCC.pop()
///              all = all union a union { ACC.pop() }
///          until n <= H[q]
///          SCC.push(<n, all>)
///          if all != F
///              return 0
///          return new emptiness_check_result(necessary data)
///

```

check() returns 0 iff the automaton's language is empty. It returns an instance of `emptiness_check_result`. If the automaton accept a word. (Use `emptiness_check_result::accepting_run()` to extract an accepting run.)

There are two variants of this algorithm: `spot::couvreur99_check` and `spot::couvreur99_check_shy`. They differ in their memory usage, the number for successors computed before they are used and the way the depth first search is directed.

`spot::couvreur99_check` performs a straightforward depth first search. The DFS stacks store `tgba_succ_` iterators, so that only the iterators which really are explored are computed.

`spot::couvreur99_check_shy` tries to explore successors which are visited states first. this helps to merge SCCs and generally helps to produce shorter counter-examples. However this algorithm cannot stores unprocessed successors as `tgba_succ_iterators`: it must compute all successors of a `state` at once in order to decide which to explore first, and must keep a list of all unexplored successors in its DFS stack.

The `couvreur99()` function is a wrapper around these two flavors of the algorithm. *options* is an option map that specifies which algorithms should be used, and how.

The following options are available.

- "shy" : if non zero, then `spot::couvreur99_check_shy` is used, otherwise (and by default) `spot::couvreur99_check` is used.
- "poprem" : specifies how the algorithm should handle the destruction of non-accepting maximal strongly connected components. If `poprem` is non null, the algorithm will keep a list of all states of a SCC that are fully processed and should be removed once the MSCC is popped. If `poprem` is null (the default), the MSCC will be traversed again (i.e. generating the successors of the root recursively) for deletion. This is a choice between memory and speed.
- "group" : this options is used only by `spot::couvreur99_check_shy`. If non null (the default), the successors of all the states that belong to the same SCC will be considered when choosing a successor. Otherwise, only the successor of the topmost `state` on the DFS stack are considered.

#### 9.26.1.4 `emptiness_check* spot::explicit_gv04_check (const tgba * a, option_map o = option_map())`

Emptiness check based on Geldenhuys and Valmari's TACAS'04 paper.

##### Precondition:

The automaton  $a$  must have at most one acceptance condition.

The original algorithm, coming from the following paper, has only been slightly modified to work on transition-based automata.

```

/// @InProceedings{geldenhuys.04.tacas,
///   author   = {Jaco Geldenhuys and Antti Valmari},
///   title    = {Tarjan's Algorithm Makes On-the-Fly {LTL} Verification
///               More Efficient},
///   booktitle = {Proceedings of the 10th International Conference on Tools
///               and Algorithms for the Construction and Analysis of Systems
///               (TACAS'04)},
///   editor   = {Kurt Jensen and Andreas Podelski},
///   pages    = {205--219},
///   year     = {2004},
///   publisher = {Springer-Verlag},
///   series   = {Lecture Notes in Computer Science},
///   volume   = {2988},
///   isbn     = {3-540-21299-X}
/// }
///

```

#### 9.26.1.5 `emptiness_check* spot::explicit_magic_search (const tgba * a, option_map o = option_map())`

Returns an emptiness checker on the `spot::tgba` automaton  $a$ .

##### Precondition:

The automaton  $a$  must have at most one acceptance condition (i.e. it is a TBA).

During the visit of  $a$ , the returned checker stores explicitly all the traversed states. The method `check()` of the checker can be called several times (until it returns a null pointer) to enumerate all the visited acceptance paths. The implemented algorithm is the following:

```

/// procedure check ()
/// begin
///   call dfs_blue(s0);
/// end;
///
/// procedure dfs_blue (s)
/// begin
///   s.color = blue;
///   for all t in post(s) do
///     if t.color == white then
///       call dfs_blue(t);
///     end if;
///     if (the edge (s,t) is accepting) then
///       target = s;
///       call dfs_red(t);
///     end if;
///   end for;
/// end;

```

```

///
/// procedure dfs_red(s)
/// begin
///   s.color = red;
///   if s == target then
///     report cycle
///   end if;
///   for all t in post(s) do
///     if t.color == blue then
///       call dfs_red(t);
///     end if;
///   end for;
/// end;
///

```

This algorithm is an adaptation to TBA of the one (which deals with accepting states) presented in

```

/// Article{      courcoubetis.92.fmsd,
///   author      = {Costas Courcoubetis and Moshe Y. Vardi and Pierre
///                  Wolper and Mihalis Yannakakis},
///   title       = {Memory-Efficient Algorithm for the Verification of
///                  Temporal Properties},
///   journal     = {Formal Methods in System Design},
///   pages       = {275--288},
///   year        = {1992},
///   volume      = {1}
/// }
///

```

### Bug

The name is misleading. Magic-search is the algorithm from `godefroid.93.pstv`, not `courcoubetis.92.fmsd`.

**9.26.1.6 emptiness\_check\* spot::explicit\_se05\_search (const tgba \* *a*, option\_map *o* = option\_map())**

Returns an emptiness check on the `spot::tgba` automaton *a*.

### Precondition:

The automaton *a* must have at most one acceptance condition (i.e. it is a TBA).

During the visit of *a*, the returned checker stores explicitly all the traversed states. The method *check()* of the checker can be called several times (until it returns a null pointer) to enumerate all the visited accepting paths. The implemented algorithm is an optimization of `spot::explicit_magic_search` and is the following:

```

/// procedure check ()
/// begin
///   call dfs_blue(s0);
/// end;
///
/// procedure dfs_blue (s)
/// begin
///   s.color = cyan;
///   for all t in post(s) do
///     if t.color == white then
///       call dfs_blue(t);
///     else if t.color == cyan and
///       (the edge (s,t) is accepting or

```

```

///          (it exists a predecessor p of s in st_blue and s != t and
///          the arc between p and s is accepting)) then
///      report cycle;
///  end if;
///  if the edge (s,t) is accepting then
///      call dfs_red(t);
///  end if;
/// end for;
/// s.color = blue;
/// end;
///
/// procedure dfs_red(s)
/// begin
///   if s.color == cyan then
///     report cycle;
///   end if;
///   s.color = red;
///   for all t in post(s) do
///     if t.color == blue then
///       call dfs_red(t);
///     end if;
///   end for;
/// end;
///

```

It is an adaptation to TBA of the one presented in

```

/// @techreport{SE04,
///   author = {Stefan Schwoon and Javier Esparza},
///   institution = {Universit{"a"}t Stuttgart, Fakult{"a"}t Informatik,
///   Elektrotechnik und Informationstechnik},
///   month = {November},
///   number = {2004/06},
///   title = {A Note on On-The-Fly Verification Algorithms},
///   year = {2004},
///   url =
/// {http://www.fmi.uni-stuttgart.de/szs/publications/info/schwoosn.SE04.shtml}
/// }
///

```

**See also:**

[spot::explicit\\_magic\\_search](#)

**9.26.1.7 emptiness\_check\*** [spot::explicit\\_tau03\\_opt\\_search](#) (const tgba \* *a*, option\_map *o* = option\_map())

Returns an emptiness checker on the [spot::tgba](#) automaton *a*.

**Precondition:**

The automaton *a* must have at least one acceptance condition.

During the visit of *a*, the returned checker stores explicitly all the traversed states. The implemented algorithm is the following:

```

/// procedure check ()
/// begin
///   weight = 0; // the null vector
///   call dfs_blue(s0);
/// end;

```

```

///
/// procedure dfs_blue (s)
/// begin
///   s.color = cyan;
///   s.acc = emptyset;
///   s.weight = weight;
///   for all t in post(s) do
///     let (s, l, a, t) be the edge from s to t;
///     if t.color == white then
///       for all b in a do
///         weight[b] = weight[b] + 1;
///       end for;
///       call dfs_blue(t);
///       for all b in a do
///         weight[b] = weight[b] - 1;
///       end for;
///     end if;
///     Acc = s.acc U a;
///     if t.color == cyan &&
///        (Acc U support(weight - t.weight) U t.acc) == all_acc then
///       report a cycle;
///     else if Acc not included in t.acc then
///       t.acc := t.acc U Acc;
///       call dfs_red(t, Acc);
///     end if;
///   end for;
///   s.color = blue;
/// end;
///
/// procedure dfs_red(s, Acc)
/// begin
///   for all t in post(s) do
///     let (s, l, a, t) be the edge from s to t;
///     if t.color == cyan &&
///        (Acc U support(weight - t.weight) U t.acc) == all_acc then
///       report a cycle;
///     else if t.color != white and Acc not included in t.acc then
///       t.acc := t.acc U Acc;
///       call dfs_red(t, Acc);
///     end if;
///   end for;
/// end;
///

```

This algorithm is a generalisation to TGBA of the one implemented in [spot::explicit\\_se05\\_search](#). It is based on the acceptance set labelling of states used in [spot::explicit\\_tau03\\_search](#). Moreover, it introduces a slight optimisation based on vectors of integers counting for each acceptance condition how many times the condition has been visited in the path stored in the blue stack. Such a vector is associated to each [state](#) of this stack.

#### 9.26.1.8 emptiness\_check\* spot::explicit\_tau03\_search (const tgba \* a, option\_map o = option\_map())

Returns an emptiness checker on the [spot::tgba](#) automaton *a*.

##### Precondition:

The automaton *a* must have at least one acceptance condition.

During the visit of *a*, the returned checker stores explicitly all the traversed states. The implemented algorithm is the following:

```

/// procedure check ()

```

```

/// begin
///   call dfs_blue(s0);
/// end;
///
/// procedure dfs_blue (s)
/// begin
///   s.color = blue;
///   s.acc = emptyset;
///   for all t in post(s) do
///     if t.color == white then
///       call dfs_blue(t);
///     end if;
///   end for;
///   for all t in post(s) do
///     let (s, l, a, t) be the edge from s to t;
///     if s.acc U a not included in t.acc then
///       call dfs_red(t, a U s.acc);
///     end if;
///   end for;
///   if s.acc == all_acc then
///     report a cycle;
///   end if;
/// end;
///
/// procedure dfs_red(s, A)
/// begin
///   s.acc = s.acc U A;
///   for all t in post(s) do
///     if t.color != white and A not included in t.acc then
///       call dfs_red(t, A);
///     end if;
///   end for;
/// end;
///

```

This algorithm is the one presented in

```

/// @techreport{HUT-TCS-A83,
///   address = {Espoo, Finland},
///   author = {Heikki Tauriainen},
///   institution = {Helsinki University of Technology, Laboratory for
///   Theoretical Computer Science},
///   month = {December},
///   number = {A83},
///   pages = {132},
///   title = {On Translating Linear Temporal Logic into Alternating and
///   Nondeterministic Automata},
///   type = {Research Report},
///   year = {2003},
///   url = {http://www.tcs.hut.fi/Publications/info/bibdb.HUT-TCS-A83.shtml}
/// }
///

```

#### 9.26.1.9 emptiness\_check\* spot::magic\_search (const tgba \* a, option\_map o = option\_map())

Wrapper for the two magic\_search implementations.

This wrapper calls explicit\_magic\_search\_search() or bit\_state\_hashing\_magic\_search() according to the "bsh" option in the option\_map. If "bsh" is set and non null, its value is used as the size of the hash map.

#### 9.26.1.10 emptiness\_check\* spot::se05 (const tgba \* a, option\_map o)

Wrapper for the two se05 implementations.

This wrapper calls `explicit_se05_search()` or `bit_state_hashing_se05_search()` according to the "bsh" option in the `option_map`. If "bsh" is set and non null, its value is used as the size of the hash map.

## 9.27 TGBA runs and supporting functions

### Classes

- struct `spot::tgba_run`  
*An accepted run, for a `tgba`.*

### Functions

- `std::ostream & spot::print_tgba_run` (`std::ostream &os`, `const tgba *a`, `const tgba_run *run`)  
*Display a `tgba_run`.*
- `tgba * spot::tgba_run_to_tgba` (`const tgba *a`, `const tgba_run *run`)  
*Return an `explicit_tgba` corresponding to run (i.e. comparable states are merged).*
- `tgba_run * spot::project_tgba_run` (`const tgba *a_run`, `const tgba *a_proj`, `const tgba_run *run`)  
*Project a `tgba_run` on a `tgba`.*
- `tgba_run * spot::reduce_run` (`const tgba *a`, `const tgba_run *org`)  
*Reduce an accepting run.*
- `bool spot::replay_tgba_run` (`std::ostream &os`, `const tgba *a`, `const tgba_run *run`, `bool debug=false`)  
*Replay a `tgba_run` on a `tgba`.*

### 9.27.1 Function Documentation

#### 9.27.1.1 `std::ostream& spot::print_tgba_run` (`std::ostream & os`, `const tgba * a`, `const tgba_run * run`)

Display a `tgba_run`.

Output the prefix and cycle of the `tgba_run run`, even if it does not corresponds to an actual run of the automaton `a`. This is unlike `replay_tgba_run()`, which will ensure the run actually exist in the automaton (and will display any transition annotation).

(`a` is used here only to format states and transitions.)

Output the prefix and cycle of the `tgba_run run`, even if it does not corresponds to an actual run of the automaton `a`. This is unlike `replay_tgba_run()`, which will ensure the run actually exist in the automaton (and will display any transition annotation).

### 9.27.1.2 `tgba_run* spot::project_tgba_run (const tgba * a_run, const tgba * a_proj, const tgba_run * run)`

Project a `tgba_run` on a `tgba`.

If a `tgba_run` has been generated on a product, or any other on-the-fly algorithm with `tgba` operands,

#### Parameters:

- run* the run to replay
- a\_run* the automata on which the run was generated
- a\_proj* the automata on which to project the run

#### Returns:

true iff the run could be completed

### 9.27.1.3 `tgba_run* spot::reduce_run (const tgba * a, const tgba_run * org)`

Reduce an accepting run.

Return a run which is accepting for *and* that is no longer that *org*.

### 9.27.1.4 `bool spot::replay_tgba_run (std::ostream & os, const tgba * a, const tgba_run * run, bool debug = false)`

Replay a `tgba_run` on a `tgba`.

This is similar to `print_tgba_run()`, except that the run is actually replayed on the automaton while it is printed. Doing so makes it possible to display transition annotations (returned by `spot::tgba::transition_annotation()`). The output will stop if the run cannot be completed.

#### Parameters:

- run* the run to replay
- a* the automata on which to replay that run
- os* the stream on which the replay should be traced
- debug* if set the output will be more verbose and extra debugging informations will be output on failure

#### Returns:

true iff the run could be completed

### 9.27.1.5 `tgba* spot::tgba_run_to_tgba (const tgba * a, const tgba_run * run)`

Return an `explicit_tgba` corresponding to *run* (i.e. comparable states are merged).

#### Precondition:

*run* must correspond to an actual run of the automaton *a*.



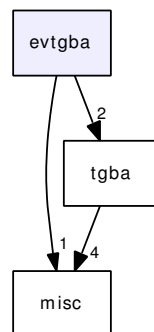
## 9.28 Emptiness-check statistics

### Classes

- struct [spot::unsigned\\_statistics](#)
- class [spot::unsigned\\_statistics\\_copy](#)  
*comparable statistics*
- class [spot::ec\\_statistics](#)  
*Emptiness-check statistics.*
- class [spot::ars\\_statistics](#)  
*Accepting Run Search statistics.*
- class [spot::acss\\_statistics](#)  
*Accepting Cycle Search Space statistics.*

## 10 spot Directory Documentation

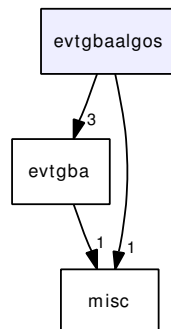
### 10.1 evtgba/ Directory Reference



### Files

- file [evtgba.hh](#)
- file [evtgbaiter.hh](#)
- file [explicit.hh](#)
- file [product.hh](#)
- file [symbol.hh](#)

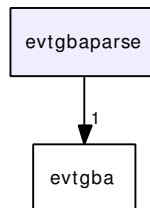
## 10.2 evtgbaalgos/ Directory Reference



### Files

- file [dotty.hh](#)
- file [reachiter.hh](#)
- file [save.hh](#)
- file [tgba2evtgba.hh](#)

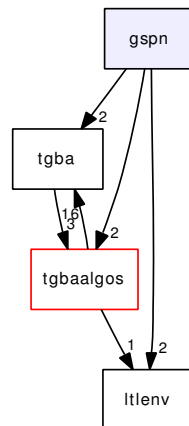
## 10.3 evtgbaparse/ Directory Reference



### Files

- file [public.hh](#)

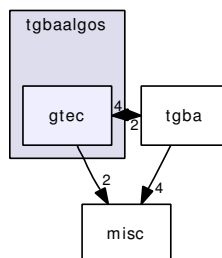
## 10.4 gspn/ Directory Reference



### Files

- file [common.hh](#)
- file [gspn.hh](#)
- file [ssp.hh](#)

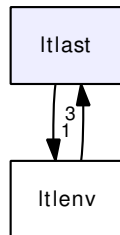
## 10.5 tgbaalgos/gtec/ Directory Reference



### Files

- file [ce.hh](#)
- file [explscc.hh](#)
- file [gtec.hh](#)
- file [nsheap.hh](#)
- file [sccstack.hh](#)
- file [status.hh](#)

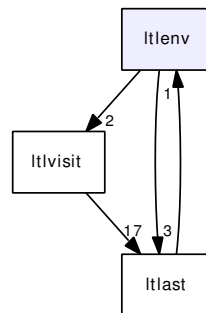
## 10.6 Itlast/ Directory Reference



### Files

- file [allnodes.hh](#)  
*Define all LTL node types.*
- file [atomic\\_prop.hh](#)  
*LTL atomic propositions.*
- file [binop.hh](#)  
*LTL binary operators.*
- file [constant.hh](#)  
*LTL constants.*
- file [formula.hh](#)  
*LTL formula interface.*
- file [multop.hh](#)  
*LTL multi-operand operators.*
- file [predecl.hh](#)  
*Predeclare all LTL node types.*
- file [reformula.hh](#)  
*Reference-counted LTL formulae.*
- file [unop.hh](#)  
*LTL unary operators.*
- file [visitor.hh](#)  
*LTL visitor interface.*

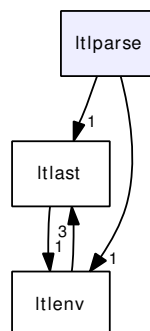
## 10.7 Itlenv/ Directory Reference



### Files

- file [declenv.hh](#)
- file [defaultenv.hh](#)
- file [environment.hh](#)
- file [rodeco.hh](#)

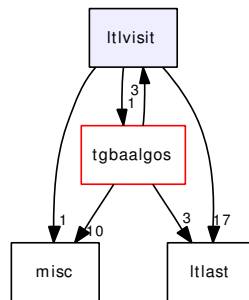
## 10.8 Itlparse/ Directory Reference



### Files

- file [location.hh](#)
- file [position.hh](#)
- file [public.hh](#)
- file [stack.hh](#)

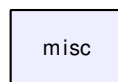
## 10.9 Itlvisit/ Directory Reference



### Files

- file [apcollect.hh](#)
- file [basicreduce.hh](#)
- file [clone.hh](#)
- file [contain.hh](#)
- file [destroy.hh](#)
- file [dotty.hh](#)
- file [dump.hh](#)
- file [length.hh](#)
- file [lunabbrev.hh](#)
- file [nenofrm.hh](#)
- file [postfix.hh](#)
- file [randomltl.hh](#)
- file [reduce.hh](#)
- file [simpfg.hh](#)
- file [syntimpl.hh](#)
- file [tostring.hh](#)
- file [tunabbrev.hh](#)

## 10.10 misc/ Directory Reference

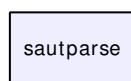


### Files

- file [bareword.hh](#)
- file [bddalloc.hh](#)
- file [bddlt.hh](#)
- file [escape.hh](#)
- file [freelist.hh](#)
- file [hash.hh](#)
- file [hashfunc.hh](#)

- file [ltstr.hh](#)
- file [memusage.hh](#)
- file [minato.hh](#)
- file [modgray.hh](#)
- file [optionmap.hh](#)
- file [random.hh](#)
- file [timer.hh](#)
- file [version.hh](#)

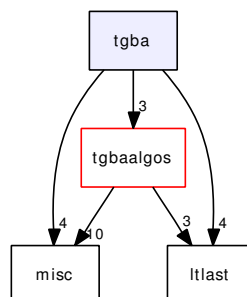
## 10.11 sautparse/ Directory Reference



### Files

- file [location.hh](#)
- file [position.hh](#)
- file [stack.hh](#)

## 10.12 tgba/ Directory Reference

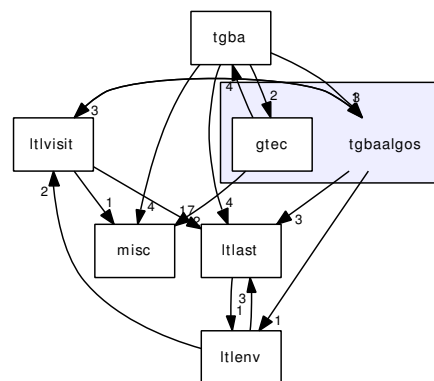


### Files

- file [bdddict.hh](#)
- file [bddprint.hh](#)
- file [formula2bdd.hh](#)
- file [public.hh](#)
- file [state.hh](#)
- file [statebdd.hh](#)
- file [succiter.hh](#)
- file [succiterconcrete.hh](#)
- file [tgba.hh](#)
- file [tgbabddconcrete.hh](#)
- file [tgbabddconcretefactory.hh](#)
- file [tgbabddconcreteproduct.hh](#)

- file [tgbaabddcoredata.hh](#)
- file [tgbaabddfactory.hh](#)
- file [tgbaexplicit.hh](#)
- file [tgbaproduct.hh](#)
- file [tgbaeduc.hh](#)
- file [tgbatba.hh](#)

## 10.13 tgbaalgorithms/ Directory Reference



### Directories

- directory [gtec](#)

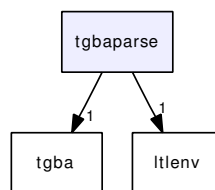
### Files

- file [bfssteps.hh](#)
- file [dotty.hh](#)
- file [dottydec.hh](#)
- file [dupexp.hh](#)
- file [emptiness.hh](#)
- file [emptiness\\_stats.hh](#)
- file [gv04.hh](#)
- file [lbtt.hh](#)
- file [ltl2tgba\\_fm.hh](#)
- file [ltl2tgba\\_lacim.hh](#)
- file [magic.hh](#)
- file [neverclaim.hh](#)
- file [powerset.hh](#)
- file [projrun.hh](#)
- file [randomgraph.hh](#)
- file [reachiter.hh](#)
- file [reducerun.hh](#)
- file [reductgba\\_sim.hh](#)
- file [replayrun.hh](#)
- file [rundotdec.hh](#)



- file [save.hh](#)
- file [se05.hh](#)
- file [stats.hh](#)
- file [tau03.hh](#)
- file [tau03opt.hh](#)
- file [weight.hh](#)

## 10.14 tgbaparse/ Directory Reference



### Files

- file [public.hh](#)

## 11 spot Namespace Documentation

### 11.1 Itlly Namespace Reference

#### Classes

- class [location](#)  
*Abstract a [location](#).*
- class [position](#)  
*Abstract a [position](#).*
- class [stack](#)
- class [slice](#)  
*Present a [slice](#) of the top of a [stack](#).*

#### Functions

- `const location operator+ (const location &begin, const location &end)`  
*Join two [location](#) objects to create a [location](#).*
- `const location operator+ (const location &begin, unsigned int width)`  
*Add two [location](#) objects.*
- `location & operator+= (location &res, unsigned int width)`  
*Add and assign a [location](#).*

- `std::ostream & operator<< (std::ostream &ostr, const location &loc)`  
*Intercept output stream redirection.*
- `const position & operator+= (position &res, const int width)`  
*Add and assign a [position](#).*
- `const position operator+ (const position &begin, const int width)`  
*Add two [position](#) objects.*
- `const position & operator-= (position &res, const int width)`  
*Add and assign a [position](#).*
- `const position operator- (const position &begin, const int width)`  
*Add two [position](#) objects.*
- `std::ostream & operator<< (std::ostream &ostr, const position &pos)`  
*Intercept output stream redirection.*

### 11.1.1 Function Documentation

**11.1.1.1** `const position Itlly::operator+ (const position & begin, const int width)` `[inline]`

Add two [position](#) objects.

**11.1.1.2** `const location Itlly::operator+ (const location & begin, unsigned int width)` `[inline]`

Add two [location](#) objects.

**11.1.1.3** `const location Itlly::operator+ (const location & begin, const location & end)` `[inline]`

Join two [location](#) objects to create a [location](#).

**11.1.1.4** `const position& Itlly::operator+= (position & res, const int width)` `[inline]`

Add and assign a [position](#).

**11.1.1.5** `location& Itlly::operator+= (location & res, unsigned int width)` `[inline]`

Add and assign a [location](#).

**11.1.1.6** `const position Itlly::operator- (const position & begin, const int width)` `[inline]`

Add two [position](#) objects.

**11.1.1.7** `const position& Itlly::operator-= (position & res, const int width)` `[inline]`

Add and assign a [position](#).

**11.1.1.8** `std::ostream& ltlyy::operator<< (std::ostream & ostr, const position & pos)` [inline]

Intercept output stream redirection.

**Parameters:**

- ostr* the destination output stream
- pos* a reference to the [position](#) to redirect

**11.1.1.9** `std::ostream& ltlyy::operator<< (std::ostream & ostr, const location & loc)` [inline]

Intercept output stream redirection.

**Parameters:**

- ostr* the destination output stream
- loc* a reference to the [location](#) to redirect

Avoid duplicate information.

**11.2 sautyy Namespace Reference****Classes**

- class [location](#)  
*Abstract a [location](#).*
- class [position](#)  
*Abstract a [position](#).*
- class [stack](#)
- class [slice](#)  
*Present a [slice](#) of the top of a [stack](#).*

**Functions**

- const [location](#) [operator+](#) (const [location](#) &begin, const [location](#) &end)  
*Join two [location](#) objects to create a [location](#).*
- const [location](#) [operator+](#) (const [location](#) &begin, unsigned int width)  
*Add two [location](#) objects.*
- [location](#) & [operator+=](#) ([location](#) &res, unsigned int width)  
*Add and assign a [location](#).*
- std::ostream & [operator<<](#) (std::ostream &ostr, const [location](#) &loc)  
*Intercept output stream redirection.*

- `const position & operator+= (position &res, const int width)`  
*Add and assign a [position](#).*
- `const position operator+ (const position &begin, const int width)`  
*Add two [position](#) objects.*
- `const position & operator-= (position &res, const int width)`  
*Add and assign a [position](#).*
- `const position operator- (const position &begin, const int width)`  
*Add two [position](#) objects.*
- `std::ostream & operator<< (std::ostream &ostr, const position &pos)`  
*Intercept output stream redirection.*

### 11.2.1 Function Documentation

#### 11.2.1.1 `const position sauty::operator+ (const position & begin, const int width)` [inline]

Add two [position](#) objects.

#### 11.2.1.2 `const location sauty::operator+ (const location & begin, unsigned int width)` [inline]

Add two [location](#) objects.

#### 11.2.1.3 `const location sauty::operator+ (const location & begin, const location & end)` [inline]

Join two [location](#) objects to create a [location](#).

#### 11.2.1.4 `const position& sauty::operator+= (position & res, const int width)` [inline]

Add and assign a [position](#).

#### 11.2.1.5 `location& sauty::operator+= (location & res, unsigned int width)` [inline]

Add and assign a [location](#).

#### 11.2.1.6 `const position sauty::operator- (const position & begin, const int width)` [inline]

Add two [position](#) objects.

#### 11.2.1.7 `const position& sauty::operator-= (position & res, const int width)` [inline]

Add and assign a [position](#).

**11.2.1.8** `std::ostream& sautyy::operator<< (std::ostream & ostr, const position & pos)`  
`[inline]`

Intercept output stream redirection.

**Parameters:**

- ostr* the destination output stream
- pos* a reference to the [position](#) to redirect

**11.2.1.9** `std::ostream& sautyy::operator<< (std::ostream & ostr, const location & loc)`  
`[inline]`

Intercept output stream redirection.

**Parameters:**

- ostr* the destination output stream
- loc* a reference to the [location](#) to redirect

Avoid duplicate information.

## 11.3 spot Namespace Reference

### Classes

- class [evtgba](#)
- class [evtgba\\_iterator](#)
- class [evtgba\\_explicit](#)
- class [state\\_evtgba\\_explicit](#)  
*States used by `spot::tgba_evtgba_explicit`.*
- class [evtgba\\_product](#)
- class [symbol](#)
- class [rsymbol](#)
- class [evtgba\\_reachable\\_iterator](#)  
*Iterate over all reachable states of a `spot::evtgba`.*
- class [evtgba\\_reachable\\_iterator\\_depth\\_first](#)  
*An implementation of `spot::evtgba_reachable_iterator` that browses states depth first.*
- class [evtgba\\_reachable\\_iterator\\_breadth\\_first](#)  
*An implementation of `spot::evtgba_reachable_iterator` that browses states breadth first.*
- class [bdd\\_allocator](#)  
*Manage ranges of variables.*
- struct [bdd\\_less\\_than](#)  
*Comparison functor for BDDs.*
- class [free\\_list](#)

*Manage list of free integers.*

- struct [ptr\\_hash](#)  
*A hash function for pointers.*
- struct [string\\_hash](#)  
*A hash function for strings.*
- struct [char\\_ptr\\_less\\_than](#)  
*Strict Weak Ordering for `char*`.*
- class [minato\\_isop](#)  
*Generate an irredundant sum-of-products (ISOP) form of a BDD function.*
- class [loopless\\_modular\\_mixed\\_radix\\_gray\\_code](#)  
*Loopless modular mixed radix Gray code iteration.*
- class [option\\_map](#)  
*Manage a map of options.*
- class [barand](#)  
*Compute pseudo-random integer value between 0 and `n` included, following a binomial distribution for probability `p`.*
- struct [time\\_info](#)  
*A structure to record elapsed time in clock ticks.*
- class [timer](#)  
*A timekeeper that accumulate interval of time.*
- class [timer\\_map](#)  
*A map of [timer](#), where each [timer](#) has a name.*
- class [bdd\\_dict](#)
- class [state](#)  
*Abstract class for states.*
- struct [state\\_ptr\\_less\\_than](#)  
*Strict Weak Ordering for `state*`.*
- struct [state\\_ptr\\_equal](#)  
*An Equivalence Relation for `state*`.*
- struct [state\\_ptr\\_hash](#)  
*Hash Function for `state*`.*
- class [state\\_bdd](#)
- class [tgba\\_succ\\_iterator](#)  
*Iterate over the successors of a [state](#).*

- class [tgba\\_succ\\_iterator\\_concrete](#)
- class [tgba](#)  
*A Transition-based Generalized Büchi Automaton.*
- class [tgba\\_bdd\\_concrete](#)  
*A concrete [spot::tgba](#) implemented using BDDs.*
- class [tgba\\_bdd\\_concrete\\_factory](#)  
*Helper class to build a [spot::tgba\\_bdd\\_concrete](#) object.*
- struct [tgba\\_bdd\\_core\\_data](#)  
*Core data for a TGBA encoded using BDDs.*
- class [tgba\\_bdd\\_factory](#)  
*Abstract class for [spot::tgba\\_bdd\\_concrete](#) factories.*
- class [tgba\\_explicit](#)
- class [state\\_explicit](#)
- class [tgba\\_explicit\\_succ\\_iterator](#)
- class [state\\_product](#)  
*A state for [spot::tgba\\_product](#).*
- class [tgba\\_succ\\_iterator\\_product](#)  
*Iterate over the successors of a product computed on the fly.*
- class [tgba\\_product](#)  
*A lazy product. (States are computed on the fly.).*
- class [direct\\_simulation\\_relation](#)
- class [delayed\\_simulation\\_relation](#)
- class [tgba\\_reduc](#)
- class [tgba\\_tba\\_proxy](#)  
*Degeneralize a [spot::tgba](#) on the fly, producing a TBA.*
- class [tgba\\_sba\\_proxy](#)  
*Degeneralize a [spot::tgba](#) on the fly, producing an SBA.*
- class [bfs\\_steps](#)  
*Make a BFS in a [spot::tgba](#) to compute a [tgba\\_run::steps](#).*
- class [dotty\\_decorator](#)  
*Choose *state* and link styles for [spot::dotty\\_reachable](#).*
- class [emptiness\\_check\\_result](#)  
*The result of an emptiness check.*
- class [emptiness\\_check](#)  
*Common interface to emptiness check algorithms.*
- class [emptiness\\_check\\_instantiator](#)

- struct [tgba\\_run](#)  
*An accepted run, for a [tgba](#).*
- struct [unsigned\\_statistics](#)
- class [unsigned\\_statistics\\_copy](#)  
*comparable statistics*
- class [ec\\_statistics](#)  
*Emptiness-check statistics.*
- class [ars\\_statistics](#)  
*Accepting Run Search statistics.*
- class [acss\\_statistics](#)  
*Accepting Cycle Search Space statistics.*
- class [couvreur99\\_check\\_result](#)  
*Compute a counter example from a [spot::couvreur99\\_check\\_status](#).*
- class [explicit\\_connected\\_component](#)  
*An SCC storing all its states explicitly.*
- class [connected\\_component\\_hash\\_set](#)
- class [explicit\\_connected\\_component\\_factory](#)  
*Abstract factory for [explicit\\_connected\\_component](#).*
- class [connected\\_component\\_hash\\_set\\_factory](#)  
*Factory for [connected\\_component\\_hash\\_set](#).*
- class [couvreur99\\_check](#)  
*An implementation of the Couvreur99 emptiness-check algorithm.*
- class [couvreur99\\_check\\_shy](#)  
*A version of [spot::couvreur99\\_check](#) that tries to visit known states first.*
- class [numbered\\_state\\_heap\\_const\\_iterator](#)  
*Iterator on [numbered\\_state\\_heap](#) objects.*
- class [numbered\\_state\\_heap](#)  
*Keep track of a large quantity of indexed states.*
- class [numbered\\_state\\_heap\\_factory](#)  
*Abstract factory for [numbered\\_state\\_heap](#).*
- class [numbered\\_state\\_heap\\_hash\\_map](#)  
*A straightforward implementation of [numbered\\_state\\_heap](#) with a hash map.*
- class [numbered\\_state\\_heap\\_hash\\_map\\_factory](#)  
*Factory for [numbered\\_state\\_heap\\_hash\\_map](#).*



- class [scc\\_stack](#)
- class [couvreur99\\_check\\_status](#)  
*The status of the emptiness-check on success.*
- class [tgba\\_reachable\\_iterator](#)  
*Iterate over all reachable states of a [spot::tgba](#).*
- class [tgba\\_reachable\\_iterator\\_depth\\_first](#)  
*An implementation of [spot::tgba\\_reachable\\_iterator](#) that browses states depth first.*
- class [tgba\\_reachable\\_iterator\\_breadth\\_first](#)  
*An implementation of [spot::tgba\\_reachable\\_iterator](#) that browses states breadth first.*
- class [parity\\_game\\_graph](#)  
*Parity game graph which compute a simulation relation.*
- class [spoiler\\_node](#)  
*Spoiler node of parity game graph.*
- class [duplicator\\_node](#)  
*Duplicator node of parity game graph.*
- class [parity\\_game\\_graph\\_direct](#)  
*Parity game graph which compute the direct simulation relation.*
- class [spoiler\\_node\\_delayed](#)  
*Spoiler node of parity game graph for delayed simulation.*
- class [duplicator\\_node\\_delayed](#)  
*Duplicator node of parity game graph for delayed simulation.*
- class [parity\\_game\\_graph\\_delayed](#)
- class [tgba\\_run\\_dotty\\_decorator](#)  
*Highlight a [spot::tgba\\_run](#) on a [spot::tgba](#).*
- struct [tgba\\_statistics](#)
- class [weight](#)  
*Manage for a given automaton a vector of counter indexed by its acceptance condition.*
- class [gspn\\_exception](#)  
*An exception used to forward GSPN errors.*
- class [gspn\\_interface](#)
- class [gspn\\_ssp\\_interface](#)

## Namespaces

- namespace [ltl](#)

## Typedefs

- typedef std::set< const [symbol](#) \* > [symbol\\_set](#)
- typedef std::set< [rsymbol](#) > [rsymbol\\_set](#)
- typedef std::pair< [evtgba::location](#), std::string > [evtgba\\_parse\\_error](#)  
*A parse diagnostic with its location.*
- typedef std::list< [evtgba\\_parse\\_error](#) > [evtgba\\_parse\\_error\\_list](#)  
*A list of parser diagnostics, as filled by parse.*
- typedef std::pair< const [spot::state](#) \*, const [spot::state](#) \* > [state\\_couple](#)
- typedef std::vector< [state\\_couple](#) \* > [simulation\\_relation](#)
- typedef std::vector< [spoiler\\_node](#) \* > [sn\\_v](#)
- typedef std::vector< [duplicator\\_node](#) \* > [dn\\_v](#)
- typedef std::vector< const [state](#) \* > [s\\_v](#)
- typedef std::pair< [tgba::location](#), std::string > [tgba\\_parse\\_error](#)  
*A parse diagnostic with its location.*
- typedef std::list< [tgba\\_parse\\_error](#) > [tgba\\_parse\\_error\\_list](#)  
*A list of parser diagnostics, as filled by parse.*

## Enumerations

- enum [reduce\\_tgba\\_options](#) {  
[Reduce\\_None](#) = 0, [Reduce\\_quotient\\_Dir\\_Sim](#) = 1, [Reduce\\_transition\\_Dir\\_Sim](#) = 2, [Reduce\\_quotient\\_Del\\_Sim](#) = 4,  
[Reduce\\_transition\\_Del\\_Sim](#) = 8, [Reduce\\_Scc](#) = 16, [Reduce\\_All](#) = -1U }  
*Options for reduce.*

## Functions

- std::ostream & [dotty\\_reachable](#) (std::ostream &os, const [evtgba](#) \*g)  
*Print reachable states in dot format.*
- std::ostream & [evtgba\\_save\\_reachable](#) (std::ostream &os, const [evtgba](#) \*g)  
*Save reachable states in text format.*
- [evtgba\\_explicit](#) \* [tgba\\_to\\_evtgba](#) (const [tgba](#) \*a)  
*Convert a [tgba](#) into an [evtgba](#).*
- [evtgba\\_explicit](#) \* [evtgba\\_parse](#) (const std::string &filename, [evtgba\\_parse\\_error\\_list](#) &error\_list, bool debug=false)  
*Build a [spot::evtgba\\_explicit](#) from a text file.*
- bool [format\\_evtgba\\_parse\\_errors](#) (std::ostream &os, const std::string &filename, [evtgba\\_parse\\_error\\_list](#) &error\_list)  
*Format diagnostics produced by [spot::evtgba\\_parse](#).*

- bool `is_bare_word` (const char \*str)
  - std::string `quote_unless_bare_word` (const std::string &str)  
*Double-quote words that are not bare.*
- std::ostream & `escape_str` (std::ostream &os, const std::string &str)  
*Escape " and \ characters in str.*
- std::string `escape_str` (const std::string &str)  
*Escape " and \ characters in str.*
- size\_t `wang32_hash` (size\_t key)  
*Thomas Wang's 32 bit hash function.*
- size\_t `knuth32_hash` (size\_t key)  
*Knuth's Multiplicative hash function.*
- int `memusage` ()  
*Total number of pages in use by the program.*
- void `srand` (unsigned int seed)  
*Reset the seed of the pseudo-random number generator.*
- int `rrand` (int min, int max)  
*Compute a pseudo-random integer value between min and max included.*
- int `mrnd` (int max)  
*Compute a pseudo-random integer value between 0 and max-1 included.*
- double `drand` ()  
*Compute a pseudo-random double value between 0.0 and 1.0 (1.0 excluded).*
- double `nrnd` ()  
*Compute a pseudo-random double value following a standard normal distribution. (Odeh & Evans).*
- double `bmrand` ()  
*Compute a pseudo-random double value following a standard normal distribution. (Box-Muller).*
- int `prand` (double p)  
*Return a pseudo-random positive integer value following a Poisson distribution with parameter p.*
- const char \* `version` ()  
*Return Spot's version.*
- std::ostream & `bdd_print_sat` (std::ostream &os, const `bdd_dict` \*dict, bdd b)  
*Print a BDD as a list of literals.*
- std::string `bdd_format_sat` (const `bdd_dict` \*dict, bdd b)  
*Format a BDD as a list of literals.*

- `std::ostream & bdd_print_acc` (`std::ostream &os`, `const bdd_dict *dict`, `bdd b`)  
*Print a BDD as a list of acceptance conditions.*
- `std::ostream & bdd_print_accset` (`std::ostream &os`, `const bdd_dict *dict`, `bdd b`)  
*Print a BDD as a set of acceptance conditions.*
- `std::string bdd_format_accset` (`const bdd_dict *dict`, `bdd b`)  
*Format a BDD as a set of acceptance conditions.*
- `std::ostream & bdd_print_set` (`std::ostream &os`, `const bdd_dict *dict`, `bdd b`)  
*Print a BDD as a set.*
- `std::string bdd_format_set` (`const bdd_dict *dict`, `bdd b`)  
*Format a BDD as a set.*
- `std::ostream & bdd_print_formula` (`std::ostream &os`, `const bdd_dict *dict`, `bdd b`)  
*Print a BDD as a formula.*
- `std::string bdd_format_formula` (`const bdd_dict *dict`, `bdd b`)  
*Format a BDD as a formula.*
- `std::ostream & bdd_print_dot` (`std::ostream &os`, `const bdd_dict *dict`, `bdd b`)  
*Print a BDD as a diagram in doty format.*
- `std::ostream & bdd_print_table` (`std::ostream &os`, `const bdd_dict *dict`, `bdd b`)  
*Print a BDD as a table.*
- `bdd formula_to_bdd` (`const ltl::formula *f`, `bdd_dict *d`, `void *for_me`)
- `const ltl::formula * bdd_to_formula` (`bdd f`, `const bdd_dict *d`)
- `tgba_bdd_concrete * product` (`const tgba_bdd_concrete *left`, `const tgba_bdd_concrete *right`)  
*Multiplies two spot::tgba\_bdd\_concrete automata.*
- `std::ostream & dotty_reachable` (`std::ostream &os`, `const tgba *g`, `dotty_decorator *dd=dotty_decorator::instance()`)  
*Print reachable states in dot format.*
- `tgba_explicit * tgba_dupexp_bfs` (`const tgba *aut`)  
*Build an explicit automata from all states of aut, numbering states in bread first order as they are processed.*
- `tgba_explicit * tgba_dupexp_dfs` (`const tgba *aut`)  
*Build an explicit automata from all states of aut, numbering states in depth first order as they are processed.*
- `std::ostream & print_tgba_run` (`std::ostream &os`, `const tgba *a`, `const tgba_run *run`)  
*Display a tgba\_run.*
- `tgba * tgba_run_to_tgba` (`const tgba *a`, `const tgba_run *run`)  
*Return an explicit\_tgba corresponding to run (i.e. comparable states are merged).*
- `emptiness_check * couvreur99` (`const tgba *a`, `option_map options=option_map()`, `const numbered_state_heap_factory *nshf=numbered_state_heap_hash_map_factory::instance()`)

*Check whether the language of an automaton is empty.*

- `emptiness_check * explicit_gv04_check` (const `tgba` \*a, `option_map` o=`option_map`())  
*Emptiness check based on Geldenhuys and Valmari's TACAS'04 paper.*
- `std::ostream & lbtt_reachable` (std::ostream &os, const `tgba` \*g)  
*Print reachable states in LBTT format.*
- `tgba_explicit * ltl_to_tgba_fm` (const `ltl::formula` \*f, `bdd_dict` \*dict, bool `exprop`=false, bool `symb_merge`=true, bool `branching_postponement`=false, bool `fair_loop_approx`=false, const `ltl::atomic_prop_set` \*unobs=0, int `reduce_ltl`=`ltl::Reduce_None`, bool `containment_checks`=false)  
*Build a `spot::tgba_explicit` from an LTL formula.*
- `tgba_bdd_concrete * ltl_to_tgba_lacim` (const `ltl::formula` \*f, `bdd_dict` \*dict)  
*Build a `spot::tgba_bdd_concrete` from an LTL formula.*
- `emptiness_check * explicit_magic_search` (const `tgba` \*a, `option_map` o=`option_map`())  
*Returns an emptiness checker on the `spot::tgba` automaton a.*
- `emptiness_check * bit_state_hashing_magic_search` (const `tgba` \*a, `size_t` size, `option_map` o=`option_map`())  
*Returns an emptiness checker on the `spot::tgba` automaton a.*
- `emptiness_check * magic_search` (const `tgba` \*a, `option_map` o=`option_map`())  
*Wrapper for the two `magic_search` implementations.*
- `std::ostream & never_claim_reachable` (std::ostream &os, const `tgba_sba_proxy` \*g, const `ltl::formula` \*f=0)  
*Print reachable states in Spin never claim format.*
- `tgba_explicit * tgba_powerset` (const `tgba` \*aut)  
*Build a deterministic automaton, ignoring acceptance conditions.*
- `tgba_run * project_tgba_run` (const `tgba` \*a\_run, const `tgba` \*a\_proj, const `tgba_run` \*run)  
*Project a `tgba_run` on a `tgba`.*
- `tgba * random_graph` (int n, float d, const `ltl::atomic_prop_set` \*ap, `bdd_dict` \*dict, int n\_acc=0, float a=0.1, float t=0.5, `ltl::environment` \*env=&`ltl::default_environment::instance`())  
*Construct a `tgba` randomly.*
- `tgba_run * reduce_run` (const `tgba` \*a, const `tgba_run` \*org)  
*Reduce an accepting run.*
- `tgba * reduc_tgba_sim` (const `tgba` \*a, int opt=`Reduce_All`)  
*Remove some node of the automata using a simulation relation.*
- `direct_simulation_relation * get_direct_relation_simulation` (const `tgba` \*a, std::ostream &os, int opt=-1)  
*Compute a direct simulation relation on *state* of `tgba` f.*

- `delayed_simulation_relation * get_delayed_relation_simulation` (const `tgba` \*a, `std::ostream` &os, int opt=-1)
- `void free_relation_simulation` (`direct_simulation_relation` \*rel)  
*To free a simulation relation.*
- `void free_relation_simulation` (`delayed_simulation_relation` \*rel)  
*To free a simulation relation.*
- `bool replay_tgba_run` (`std::ostream` &os, const `tgba` \*a, const `tgba_run` \*run, bool debug=false)  
*Replay a `tgba_run` on a `tgba`.*
- `std::ostream & tgba_save_reachable` (`std::ostream` &os, const `tgba` \*g)  
*Save reachable states in text format.*
- `emptiness_check * explicit_se05_search` (const `tgba` \*a, `option_map` o=`option_map`())  
*Returns an emptiness check on the `spot::tgba` automaton a.*
- `emptiness_check * bit_state_hashing_se05_search` (const `tgba` \*a, `size_t` size, `option_map` o=`option_map`())  
*Returns an emptiness checker on the `spot::tgba` automaton a.*
- `emptiness_check * se05` (const `tgba` \*a, `option_map` o)  
*Wrapper for the two se05 implementations.*
- `tgba_statistics stats_reachable` (const `tgba` \*g)  
*Compute statistics for an automaton.*
- `emptiness_check * explicit_tau03_search` (const `tgba` \*a, `option_map` o=`option_map`())  
*Returns an emptiness checker on the `spot::tgba` automaton a.*
- `emptiness_check * explicit_tau03_opt_search` (const `tgba` \*a, `option_map` o=`option_map`())  
*Returns an emptiness checker on the `spot::tgba` automaton a.*
- `tgba_explicit * tgba_parse` (const `std::string` &filename, `tgba_parse_error_list` &error\_list, `bdd_dict` \*dict, `ltl::environment` &env=`ltl::default_environment::instance`(), `ltl::environment` &envacc=`ltl::default_environment::instance`(), bool debug=false)  
*Build a `spot::tgba_explicit` from a text file.*
- `bool format_tgba_parse_errors` (`std::ostream` &os, const `std::string` &filename, `tgba_parse_error_list` &error\_list)  
*Format diagnostics produced by `spot::tgba_parse`.*
- `std::ostream & operator<<` (`std::ostream` &os, const `gspn_exception` &e)
- `couvreur99_check * couvreur99_check_ssp_semi` (const `tgba` \*ssp\_automata)
- `couvreur99_check * couvreur99_check_ssp_shy_semi` (const `tgba` \*ssp\_automata)
- `couvreur99_check * couvreur99_check_ssp_shy` (const `tgba` \*ssp\_automata, bool stack\_inclusion=true)

### 11.3.1 Typedef Documentation

#### 11.3.1.1 `typedef std::pair<evtgba::location, std::string> spot::evtgba_parse_error`

A parse diagnostic with its location.

#### 11.3.1.2 `typedef std::list<evtgba_parse_error> spot::evtgba_parse_error_list`

A list of parser diagnostics, as filled by parse.

#### 11.3.1.3 `typedef std::set<rsymbol> spot::rsymbol_set`

#### 11.3.1.4 `typedef std::vector<state_couple*> spot::simulation_relation`

#### 11.3.1.5 `typedef std::pair<const spot::state*, const spot::state*> spot::state_couple`

#### 11.3.1.6 `typedef std::set<const symbol*> spot::symbol_set`

### 11.3.2 Function Documentation

#### 11.3.2.1 `std::string spot::bdd_format_accset (const bdd_dict * dict, bdd b)`

Format a BDD as a set of acceptance conditions.

This is used when saving a TGBA.

##### Parameters:

*dict* The dictionary to use, to lookup variables.

*b* The BDD to print.

##### Returns:

The BDD formatted as a string.

#### 11.3.2.2 `std::string spot::bdd_format_formula (const bdd_dict * dict, bdd b)`

Format a BDD as a formula.

##### Parameters:

*dict* The dictionary to use, to lookup variables.

*b* The BDD to print.

##### Returns:

The BDD formatted as a string.

**11.3.2.3 std::string spot::bdd\_format\_sat (const bdd\_dict \* *dict*, bdd *b*)**

Format a BDD as a list of literals.

This assumes that *b* is a conjunction of literals.

**Parameters:**

*dict* The dictionary to use, to lookup variables.

*b* The BDD to print.

**Returns:**

The BDD formatted as a string.

**11.3.2.4 std::string spot::bdd\_format\_set (const bdd\_dict \* *dict*, bdd *b*)**

Format a BDD as a set.

**Parameters:**

*dict* The dictionary to use, to lookup variables.

*b* The BDD to print.

**Returns:**

The BDD formatted as a string.

**11.3.2.5 std::ostream& spot::bdd\_print\_acc (std::ostream & *os*, const bdd\_dict \* *dict*, bdd *b*)**

Print a BDD as a list of acceptance conditions.

This is used when saving a TGBA.

**Parameters:**

*os* The output stream.

*dict* The dictionary to use, to lookup variables.

*b* The BDD to print.

**Returns:**

The BDD formatted as a string.

**11.3.2.6 std::ostream& spot::bdd\_print\_accset (std::ostream & *os*, const bdd\_dict \* *dict*, bdd *b*)**

Print a BDD as a set of acceptance conditions.

This is used when saving a TGBA.

**Parameters:**

*os* The output stream.

*dict* The dictionary to use, to lookup variables.

*b* The BDD to print.

**Returns:**

The BDD formatted as a string.



**11.3.2.7 `std::ostream& spot::bdd_print_dot (std::ostream & os, const bdd_dict * dict, bdd b)`**

Print a BDD as a diagram in dotty format.

**Parameters:**

*os* The output stream.

*dict* The dictionary to use, to lookup variables.

*b* The BDD to print.

**11.3.2.8 `std::ostream& spot::bdd_print_formula (std::ostream & os, const bdd_dict * dict, bdd b)`**

Print a BDD as a formula.

**Parameters:**

*os* The output stream.

*dict* The dictionary to use, to lookup variables.

*b* The BDD to print.

**11.3.2.9 `std::ostream& spot::bdd_print_sat (std::ostream & os, const bdd_dict * dict, bdd b)`**

Print a BDD as a list of literals.

This assumes that *b* is a conjunction of literals.

**Parameters:**

*os* The output stream.

*dict* The dictionary to use, to lookup variables.

*b* The BDD to print.

**11.3.2.10 `std::ostream& spot::bdd_print_set (std::ostream & os, const bdd_dict * dict, bdd b)`**

Print a BDD as a set.

**Parameters:**

*os* The output stream.

*dict* The dictionary to use, to lookup variables.

*b* The BDD to print.

**11.3.2.11 `std::ostream& spot::bdd_print_table (std::ostream & os, const bdd_dict * dict, bdd b)`**

Print a BDD as a table.

**Parameters:**

*os* The output stream.

*dict* The dictionary to use, to lookup variables.

*b* The BDD to print.

**11.3.2.12** `const ltl::formula* spot::bdd_to_formula (bdd f, const bdd_dict * d)`

**11.3.2.13** `std::ostream& spot::dotty_reachable (std::ostream & os, const evtgba * g)`

Print reachable states in dot format.

**11.3.2.14** `evtgba_explicit* spot::evtgba_parse (const std::string & filename, evtgba_parse_error_list & error_list, bool debug = false)`

Build a [spot::evtgba\\_explicit](#) from a text file.

**Parameters:**

*filename* The name of the file to parse.

*error\_list* A list that will be filled with parse errors that occurred during parsing.

*debug* When true, causes the parser to trace its execution.

**Returns:**

A pointer to the [evtgba](#) built from *filename*, or 0 if the file could not be opened.

Note that the parser usually tries to recover from errors. It can return a non zero value even if it encountered an error during the parsing of *filename*. If you want to make sure *filename* was parsed successfully, check *error\_list* for emptiness.

**Warning:**

This function is not reentrant.

**11.3.2.15** `std::ostream& spot::evtgba_save_reachable (std::ostream & os, const evtgba * g)`

Save reachable states in text format.

**11.3.2.16** `bool spot::format_evtgba_parse_errors (std::ostream & os, const std::string & filename, evtgba_parse_error_list & error_list)`

Format diagnostics produced by [spot::evtgba\\_parse](#).

**Parameters:**

*os* Where diagnostics should be output.

*filename* The filename that should appear in the diagnostics.

*error\_list* The error list filled by [spot::ltl::parse](#) while parsing *ltl\_string*.

**Returns:**

true iff any diagnostic was output.

**11.3.2.17** `bdd spot::formula_to_bdd (const ltl::formula * f, bdd_dict * d, void * for_me)`

### 11.3.2.18 int spot::memusage ()

Total number of pages in use by the program.

#### Returns:

The total number of pages in use by the program if known. -1 otherwise.

### 11.3.2.19 std::ostream& spot::operator<< (std::ostream & *os*, const gspn\_exception & *e*)

### 11.3.2.20 evtgba\_explicit\* spot::tgba\_to\_evtgba (const tgba \* *a*)

Convert a [tgba](#) into an [evtgba](#).

(This cannot be done on-the-fly because the alphabet of a [tgba](#) is unknown beforehand.)

## 11.4 spot::ltl Namespace Reference

### Classes

- class [atomic\\_prop](#)  
*Atomic propositions.*
- class [binop](#)  
*Binary operator.*
- class [constant](#)  
*A [constant](#) (True or False).*
- class [formula](#)  
*An LTL [formula](#).*
- struct [formula\\_ptr\\_less\\_than](#)  
*Strict Weak Ordering for `const formula*`.*
- struct [formula\\_ptr\\_hash](#)  
*Hash Function for `const formula*`.*
- class [multop](#)  
*Multi-operand operators.*
- class [ref\\_formula](#)  
*A reference-counted LTL [formula](#).*
- class [unop](#)  
*Unary operators.*
- struct [visitor](#)  
*Formula [visitor](#) that can modify the [formula](#).*

- struct [const\\_visitor](#)  
*Formula [visitor](#) that cannot modify the [formula](#).*
- class [declarative\\_environment](#)  
*A declarative [environment](#).*
- class [default\\_environment](#)  
*A laxist [environment](#).*
- class [environment](#)  
*An [environment](#) that describes atomic propositions.*
- class [read\\_only\\_environment](#)  
*A read only [environment](#).*
- class [clone\\_visitor](#)  
*Clone a [formula](#).*
- class [language\\_containment\\_checker](#)
- class [unabbreviate\\_logic\\_visitor](#)  
*Clone and rewrite a [formula](#) to remove most of the abbreviated logical operators.*
- class [postfix\\_visitor](#)  
*Apply an algorithm on each node of an AST, during a postfix traversal.*
- class [random\\_ltl](#)  
*Generate random LTL formulae.*
- class [simplify\\_f\\_g\\_visitor](#)  
*Replace `true U f` and `false R g` by `F f` and `G g`.*
- class [unabbreviate\\_ltl\\_visitor](#)  
*Clone and rewrite a [formula](#) to remove most of the abbreviated LTL and logical operators.*

## Typedefs

- typedef std::pair< [ltl::location](#), std::string > [parse\\_error](#)  
*A parse diagnostic with its location.*
- typedef std::list< [parse\\_error](#) > [parse\\_error\\_list](#)  
*A list of parser diagnostics, as filled by [parse](#).*
- typedef std::set< [atomic\\_prop](#) \*, [formula\\_ptr\\_less\\_than](#) > [atomic\\_prop\\_set](#)  
*Set of atomic propositions.*

## Enumerations

- enum `reduce_options` {  
`Reduce_None` = 0, `Reduce_Basics` = 1, `Reduce_Syntactic_Implications` = 2, `Reduce_Eventuality_And_Universality` = 4,  
`Reduce_Containment_Checks` = 8, `Reduce_Containment_Checks_Stronger` = 16, `Reduce_All` = -1U  
 }  
*Options for `spot::ltl::reduce`.*

## Functions

- `formula * parse` (const std::string &ltl\_string, `parse_error_list` &error\_list, `environment` &env=default\_environment::instance(), bool debug=false)  
*Build a `formula` from an LTL string.*
- `bool format_parse_errors` (std::ostream &os, const std::string &ltl\_string, `parse_error_list` &error\_list)  
*Format diagnostics produced by `spot::ltl::parse`.*
- `atomic_prop_set * atomic_prop_collect` (const `formula` \*f, `atomic_prop_set` \*s=0)  
*Return the set of atomic propositions occurring in a `formula`.*
- `formula * basic_reduce` (const `formula` \*f)  
*Basic rewritings.*
- `bool is_GF` (const `formula` \*f)  
*Whether a `formula` starts with GF.*
- `bool is_FG` (const `formula` \*f)  
*Whether a `formula` starts with FG.*
- `formula * clone` (const `formula` \*f)  
*Clone a `formula`.*
- `formula * reduce_tau03` (const `formula` \*f, bool stronger=true)  
*Reduce a `formula` using language containment relationships.*
- `void destroy` (const `formula` \*f)  
*Destroys a `formula`.*
- `std::ostream & dotty` (std::ostream &os, const `formula` \*f)  
*Write a `formula` tree using dot's syntax.*
- `std::ostream & dump` (std::ostream &os, const `formula` \*f)  
*Dump a `formula` tree.*
- `int length` (const `formula` \*f)  
*Compute the length of a `formula`.*

- `formula * unabbreviate_logic` (const `formula` \*f)  
*Clone and rewrite a `formula` to remove most of the abbreviated logical operators.*
- `formula * negative_normal_form` (const `formula` \*f, bool negated=false)  
*Build the negative normal form of f.*
- `formula * reduce` (const `formula` \*f, int opt=Reduce\_All)  
*Reduce a `formula` f.*
- `bool is_eventual` (const `formula` \*f)  
*Check whether a `formula` is a pure eventuality.*
- `bool is_universal` (const `formula` \*f)  
*Check whether a `formula` is purely universal.*
- `formula * simplify_f_g` (const `formula` \*f)  
*Replace `true U f` and `false R g` by `F f` and `G g`.*
- `bool syntactic_implication` (const `formula` \*f1, const `formula` \*f2)  
*Syntactic implication.*
- `bool syntactic_implication_neg` (const `formula` \*f1, const `formula` \*f2, bool right)  
*Syntactic implication.*
- `std::ostream & to_string` (const `formula` \*f, std::ostream &os)  
*Output a `formula` as a (parsable) string.*
- `std::string to_string` (const `formula` \*f)  
*Convert a `formula` into a (parsable) string.*
- `std::ostream & to_spin_string` (const `formula` \*f, std::ostream &os)  
*Output a `formula` as a (parsable by Spin) string.*
- `std::string to_spin_string` (const `formula` \*f)  
*Convert a `formula` into a (parsable by Spin) string.*
- `formula * unabbreviate_ltl` (const `formula` \*f)  
*Clone and rewrite a `formula` to remove most of the abbreviated LTL and logical operators.*

### 11.4.1 Function Documentation

#### 11.4.1.1 `formula* spot::ltl::reduce_tau03` (const `formula` \*f, bool *stronger* = true)

Reduce a `formula` using language containment relationships.

The method is taken from table 4.1 in

```

///@TechReport{    tauriainen.03.a83,
///  author = {Heikki Tauriainen},
///  title = {On Translating Linear Temporal Logic into Alternating and
///    Nondeterministic Automata},

```

```

/// institution = {Helsinki University of Technology, Laboratory for
///               Theoretical Computer Science},
/// address = {Espoo, Finland},
/// month = dec,
/// number      = {A83},
/// pages = {132},
/// type = {Research Report},
/// year = {2003},
/// note = {Reprint of Licentiate's thesis}
///}
///
/// (The "dagged" cells in the tables are not handled here.)
///
/// If \a stronger is set, additional rules are used to further
/// reduce some U, R, and X usages.
///

```

#### 11.4.1.2 formula\* spot::ltl::unabbreviate\_ltl (const formula \*f)

Clone and rewrite a [formula](#) to remove most of the abbreviated LTL and logical operators.

The rewriting performed on logical operator is the same as the one done by [spot::ltl::unabbreviate\\_logic](#).

This will also rewrite unary operators such as [unop::F](#), and [unop::G](#), using only [binop::U](#), and [binop::R](#).

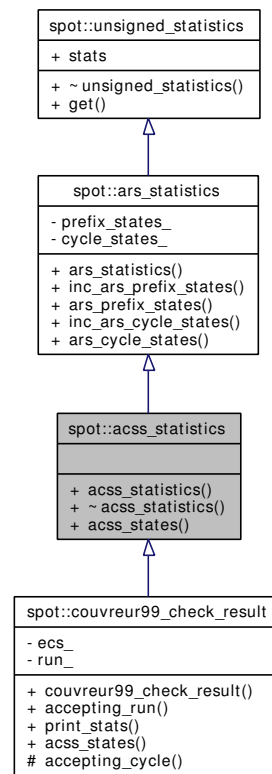
## 12 spot Class Documentation

### 12.1 spot::acss\_statistics Class Reference

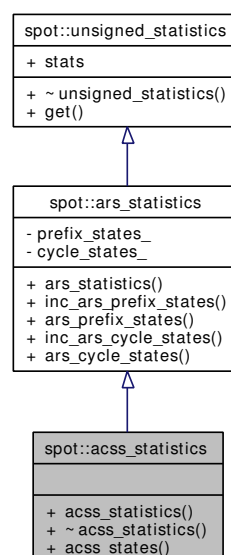
Accepting Cycle Search Space statistics.

```
#include <tgbaaalgos/emptiness_stats.hh>
```

Inheritance diagram for spot::acss\_statistics:



Collaboration diagram for spot::acss\_statistics:



## Public Types

- `typedef unsigned(unsigned_statistics::*) unsigned\_fun () const`



- typedef std::map< const char \*, [unsigned\\_fun](#), [char\\_ptr\\_less\\_than](#) > [stats\\_map](#)

## Public Member Functions

- [acss\\_statistics](#) ()
- virtual [~acss\\_statistics](#) ()
- virtual unsigned [acss\\_states](#) () const=0  
*Number of states in the search space for the accepting cycle.*
- void [inc\\_ars\\_prefix\\_states](#) ()
- unsigned [ars\\_prefix\\_states](#) () const
- void [inc\\_ars\\_cycle\\_states](#) ()
- unsigned [ars\\_cycle\\_states](#) () const
- unsigned [get](#) (const char \*str) const

## Public Attributes

- [stats\\_map](#) stats

### 12.1.1 Detailed Description

Accepting Cycle Search Space statistics.

Implementations of [spot::emptiness\\_check\\_result](#) may also implement this interface. Try to dynamic\_cast the [spot::emptiness\\_check\\_result](#) pointer to know whether these statistics are available.

### 12.1.2 Member Typedef Documentation

**12.1.2.1** typedef unsigned(unsigned\_statistics::\*) spot::unsigned\_statistics::unsigned\_fun() const [inherited]

**12.1.2.2** typedef std::map<const char\*, unsigned\_fun, char\_ptr\_less\_than> spot::unsigned\_statistics::stats\_map [inherited]

### 12.1.3 Constructor & Destructor Documentation

**12.1.3.1** spot::acss\_statistics::acss\_statistics () [inline]

**12.1.3.2** virtual spot::acss\_statistics::~~acss\_statistics () [inline, virtual]

### 12.1.4 Member Function Documentation

**12.1.4.1** virtual unsigned spot::acss\_statistics::acss\_states () const [pure virtual]

Number of states in the search space for the accepting cycle.

Implemented in [spot::couvereur99\\_check\\_result](#).

**12.1.4.2** void spot::ars\_statistics::inc\_ars\_prefix\_states () [inline, inherited]

**12.1.4.3** unsigned spot::ars\_statistics::ars\_prefix\_states () const [inline, inherited]

**12.1.4.4** void spot::ars\_statistics::inc\_ars\_cycle\_states () [inline, inherited]

**12.1.4.5** unsigned spot::ars\_statistics::ars\_cycle\_states () const [inline, inherited]

**12.1.4.6** unsigned spot::unsigned\_statistics::get (const char \* str) const [inline, inherited]

### 12.1.5 Member Data Documentation

**12.1.5.1** stats\_map spot::unsigned\_statistics::stats [inherited]

The documentation for this class was generated from the following file:

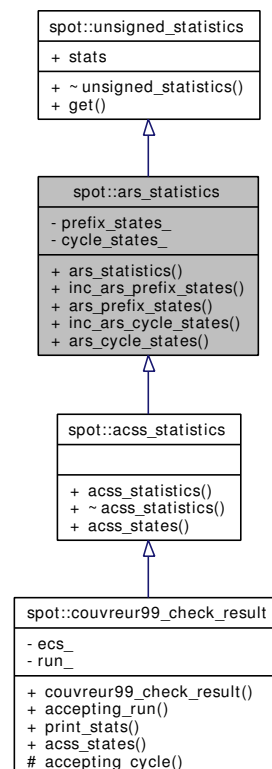
- [tgbaalgos/emptiness\\_stats.hh](#)

## 12.2 spot::ars\_statistics Class Reference

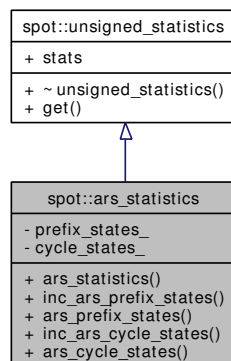
Accepting Run Search statistics.

```
#include <tgbaalgos/emptiness_stats.hh>
```

Inheritance diagram for spot::ars\_statistics:



Collaboration diagram for spot::ars\_statistics:



## Public Types

- typedef unsigned(unsigned\_statistics::\*) [unsigned\\_fun](#) () const
- typedef std::map< const char \*, [unsigned\\_fun](#), [char\\_ptr\\_less\\_than](#) > [stats\\_map](#)

## Public Member Functions

- [ars\\_statistics](#) ()
- void [inc\\_ars\\_prefix\\_states](#) ()
- unsigned [ars\\_prefix\\_states](#) () const
- void [inc\\_ars\\_cycle\\_states](#) ()
- unsigned [ars\\_cycle\\_states](#) () const
- unsigned [get](#) (const char \*str) const

## Public Attributes

- [stats\\_map](#) stats

## Private Attributes

- unsigned [prefix\\_states\\_](#)
- unsigned [cycle\\_states\\_](#)  
*states visited to construct the prefix*

### 12.2.1 Detailed Description

Accepting Run Search statistics.

Implementations of [spot::emptiness\\_check\\_result](#) may also implement this interface. Try to dynamic\_cast the [spot::emptiness\\_check\\_result](#) pointer to know whether these statistics are available.

### 12.2.2 Member Typedef Documentation

**12.2.2.1** typedef unsigned(unsigned\_statistics::\*) [spot::unsigned\\_statistics::unsigned\\_fun](#)() const  
[inherited]

**12.2.2.2** `typedef std::map<const char*, unsigned_fun, char_ptr_less_than> spot::unsigned_statistics::stats_map` [inherited]

### 12.2.3 Constructor & Destructor Documentation

**12.2.3.1** `spot::ars_statistics::ars_statistics ()` [inline]

### 12.2.4 Member Function Documentation

**12.2.4.1** `void spot::ars_statistics::inc_ars_prefix_states ()` [inline]

**12.2.4.2** `unsigned spot::ars_statistics::ars_prefix_states () const` [inline]

**12.2.4.3** `void spot::ars_statistics::inc_ars_cycle_states ()` [inline]

**12.2.4.4** `unsigned spot::ars_statistics::ars_cycle_states () const` [inline]

**12.2.4.5** `unsigned spot::unsigned_statistics::get (const char * str) const` [inline, inherited]

### 12.2.5 Member Data Documentation

**12.2.5.1** `unsigned spot::ars_statistics::prefix_states_` [private]

**12.2.5.2** `unsigned spot::ars_statistics::cycle_states_` [private]

states visited to construct the prefix

**12.2.5.3** `stats_map spot::unsigned_statistics::stats` [inherited]

The documentation for this class was generated from the following file:

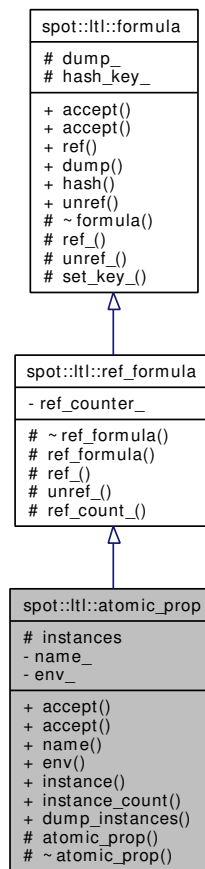
- [tgbaalgos/emptiness\\_stats.hh](#)

## 12.3 spot::ltl::atomic\_prop Class Reference

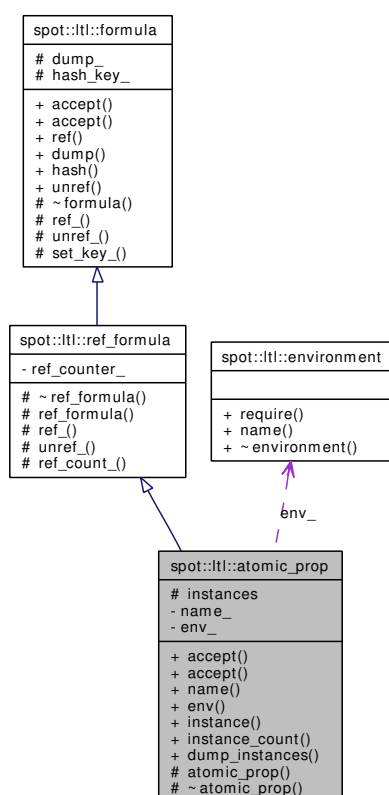
Atomic propositions.

```
#include <ltlast/atomic_prop.hh>
```

Inheritance diagram for spot::ltl::atomic\_prop:



Collaboration diagram for spot::ltl::atomic\_prop:



## Public Member Functions

- virtual void **accept** (visitor &visitor)  
*Entry point for vspot::ltl::visitor instances.*
- virtual void **accept** (const\_visitor &visitor) const  
*Entry point for vspot::ltl::const\_visitor instances.*
- const std::string & **name** () const  
*Get the name of the atomic proposition.*
- **environment** & **env** () const  
*Get the **environment** of the atomic proposition.*
- **formula** \* **ref** ()  
*clone this node*
- const std::string & **dump** () const  
*Return a canonic representation of the **formula**.*
- const size\_t **hash** () const  
*Return a hash\_key for the **formula**.*

### Static Public Member Functions

- static [atomic\\_prop](#) \* [instance](#) (const std::string &name, [environment](#) &env)
- static unsigned [instance\\_count](#) ()  
*Number of instantiated atomic propositions. For debugging.*
- static std::ostream & [dump\\_instances](#) (std::ostream &os)  
*List all instances of atomic propositions. For debugging.*
- static void [unref](#) ([formula](#) \*f)  
*release this node*

### Protected Types

- typedef std::pair< std::string, [environment](#) \* > [pair](#)
- typedef std::map< [pair](#), [atomic\\_prop](#) \* > [map](#)

### Protected Member Functions

- [atomic\\_prop](#) (const std::string &name, [environment](#) &env)
- virtual [~atomic\\_prop](#) ()
- void [ref\\_](#) ()  
*increment reference counter if any*
- bool [unref\\_](#) ()  
*decrement reference counter if any, return true when the instance must be deleted (usually when the counter hits 0).*
- unsigned [ref\\_count\\_](#) ()  
*Number of references to this [formula](#).*
- void [set\\_key\\_](#) ()  
*Compute key\_ from dump\_.*

### Protected Attributes

- std::string [dump\\_](#)  
*The canonic representation of the [formula](#).*
- size\_t [hash\\_key\\_](#)  
*The hash key of this [formula](#).*

### Static Protected Attributes

- static [map](#) [instances](#)

**Private Attributes**

- std::string [name\\_](#)
- [environment](#) \* [env\\_](#)

**12.3.1 Detailed Description**

Atomic propositions.

**12.3.2 Member Typedef Documentation**

**12.3.2.1** `typedef std::pair<std::string, environment*> spot::ltl::atomic_prop::pair` [protected]

**12.3.2.2** `typedef std::map<pair, atomic_prop*> spot::ltl::atomic_prop::map` [protected]

**12.3.3 Constructor & Destructor Documentation**

**12.3.3.1** `spot::ltl::atomic_prop::atomic_prop (const std::string & name, environment & env)` [protected]

**12.3.3.2** `virtual spot::ltl::atomic_prop::~~atomic_prop ()` [protected, virtual]

**12.3.4 Member Function Documentation**

**12.3.4.1** `static atomic_prop* spot::ltl::atomic_prop::instance (const std::string & name, environment & env)` [static]

Build an atomic proposition with name *name* in [environment](#) *env*.

**12.3.4.2** `virtual void spot::ltl::atomic_prop::accept (visitor & v)` [virtual]

Entry point for `vspot::ltl::visitor` instances.

Implements [spot::ltl::formula](#).

**12.3.4.3** `virtual void spot::ltl::atomic_prop::accept (const_visitor & v) const` [virtual]

Entry point for `vspot::ltl::const_visitor` instances.

Implements [spot::ltl::formula](#).

**12.3.4.4** `const std::string& spot::ltl::atomic_prop::name () const`

Get the name of the atomic proposition.

**12.3.4.5** `environment& spot::ltl::atomic_prop::env () const`

Get the [environment](#) of the atomic proposition.



**12.3.4.6 static unsigned spot::ltl::atomic\_prop::instance\_count ()** [static]

Number of instantiated atomic propositions. For debugging.

**12.3.4.7 static std::ostream& spot::ltl::atomic\_prop::dump\_instances (std::ostream & os)** [static]

List all instances of atomic propositions. For debugging.

**12.3.4.8 void spot::ltl::ref\_formula::ref\_ ()** [protected, virtual, inherited]

increment reference counter if any

Reimplemented from [spot::ltl::formula](#).

**12.3.4.9 bool spot::ltl::ref\_formula::unref\_ ()** [protected, virtual, inherited]

decrement reference counter if any, return true when the instance must be deleted (usually when the counter hits 0).

Reimplemented from [spot::ltl::formula](#).

**12.3.4.10 unsigned spot::ltl::ref\_formula::ref\_count\_ ()** [protected, inherited]

Number of references to this [formula](#).

**12.3.4.11 formula\* spot::ltl::formula::ref ()** [inherited]

clone this node

This increments the reference counter of this node (if one is used). You should almost never use this method directly as it doesn't touch the children. If you want to clone a whole [formula](#), use [spot::ltl::clone\(\)](#) instead.

**12.3.4.12 static void spot::ltl::formula::unref (formula \*f)** [static, inherited]

release this node

This decrements the reference counter of this node (if one is used) and can free the object. You should almost never use this method directly as it doesn't touch the children. If you want to release a whole [formula](#), use [spot::ltl::destroy\(\)](#) instead.

**12.3.4.13 const std::string& spot::ltl::formula::dump () const** [inherited]

Return a canonic representation of the [formula](#).

**12.3.4.14 const size\_t spot::ltl::formula::hash () const** [inline, inherited]

Return a hash\_key for the [formula](#).

**12.3.4.15 void spot::ltl::formula::set\_key\_ ()** [protected, inherited]

Compute key\_ from dump\_.

Should be called once in each object, after dump\_ has been set.

### 12.3.5 Member Data Documentation

**12.3.5.1** map spot::ltl::atomic\_prop::instances [static, protected]

**12.3.5.2** std::string spot::ltl::atomic\_prop::name\_ [private]

**12.3.5.3** environment\* spot::ltl::atomic\_prop::env\_ [private]

**12.3.5.4** std::string spot::ltl::formula::dump\_ [protected, inherited]

The canonic representation of the [formula](#).

**12.3.5.5** size\_t spot::ltl::formula::hash\_key\_ [protected, inherited]

The hash key of this [formula](#).

Initialized by [set\\_key\\_\(\)](#).

The documentation for this class was generated from the following file:

- [ltlast/atomic\\_prop.hh](#)

## 12.4 spot::barand< gen > Class Template Reference

Compute pseudo-random integer value between 0 and  $n$  included, following a binomial distribution for probability  $p$ .

```
#include <misc/random.hh>
```

### Public Member Functions

- [barand](#) (int n, double p)
- int [rand](#) () const

### Protected Attributes

- const int [n\\_](#)
- const double [m\\_](#)
- const double [s\\_](#)

#### 12.4.1 Detailed Description

```
template<double(*)() gen> class spot::barand< gen >
```

Compute pseudo-random integer value between 0 and  $n$  included, following a binomial distribution for probability  $p$ .

*gen* must be a random function computing a pseudo-random double value following a standard normal distribution. Use [rand\(\)](#) or [bmrnd\(\)](#).

Usually approximating a binomial distribution using a normal distribution and is accurate only if  $n \cdot p$  and  $n \cdot (1-p)$  are greater than 5.

## 12.4.2 Constructor & Destructor Documentation

**12.4.2.1** `template<double(*)() gen> spot::barand< gen >::barand (int n, double p)` `[inline]`

## 12.4.3 Member Function Documentation

**12.4.3.1** `template<double(*)() gen> int spot::barand< gen >::rand () const` `[inline]`

## 12.4.4 Member Data Documentation

**12.4.4.1** `template<double(*)() gen> const int spot::barand< gen >::n_` `[protected]`

**12.4.4.2** `template<double(*)() gen> const double spot::barand< gen >::m_` `[protected]`

**12.4.4.3** `template<double(*)() gen> const double spot::barand< gen >::s_` `[protected]`

The documentation for this class was generated from the following file:

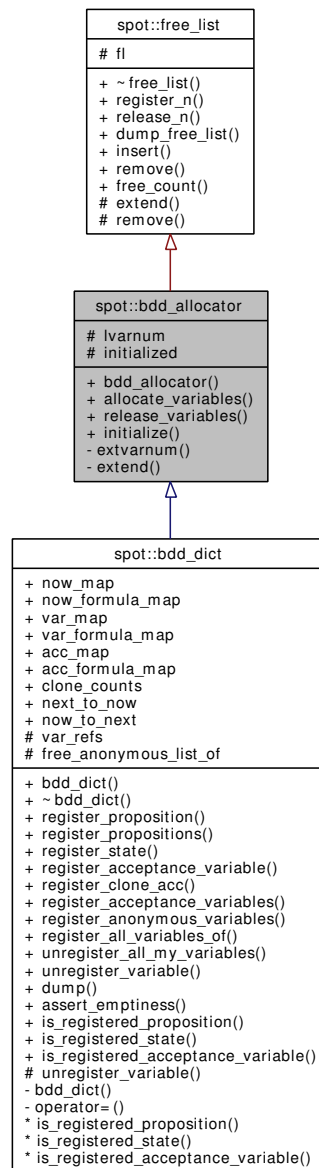
- [misc/random.hh](#)

## 12.5 spot::bdd\_allocator Class Reference

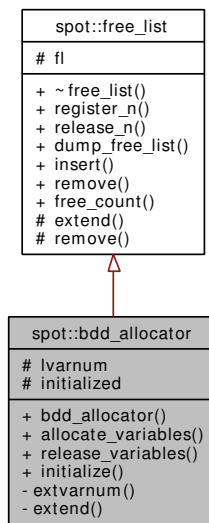
Manage ranges of variables.

```
#include <misc/bddalloc.hh>
```

Inheritance diagram for spot::bdd\_allocator:



Collaboration diagram for spot::bdd\_allocator:



### Public Member Functions

- [bdd\\_allocator\(\)](#)  
*Default constructor.*
- [int allocate\\_variables\(int n\)](#)  
*Allocate n BDD variables.*
- [void release\\_variables\(int base, int n\)](#)  
*Release n BDD variables starting at base.*

### Static Public Member Functions

- [static void initialize\(\)](#)  
*Initialize the BDD library.*

### Protected Attributes

- [int lvarnum](#)  
*number of variables in use in this allocator.*

### Static Protected Attributes

- [static bool initialized](#)  
*Whether the BDD library has been initialized.*

### Private Types

- typedef std::pair< int, int > [pos\\_lenght\\_pair](#)  
*Such pairs describe second free integer starting at first.*
- typedef std::list< [pos\\_lenght\\_pair](#) > [free\\_list\\_type](#)

### Private Member Functions

- void [extvarnum](#) (int more)  
*Require more variables.*
- virtual int [extend](#) (int n)
- int [register\\_n](#) (int n)  
*Find n consecutive integers.*
- void [release\\_n](#) (int base, int n)  
*Release n consecutive integers starting at base.*
- std::ostream & [dump\\_free\\_list](#) (std::ostream &os) const  
*Dump the list to os for debugging.*
- void [insert](#) (int base, int n)  
*Extend the list by inserting a new pos-lenght pair.*
- void [remove](#) (int base, int n=0)  
*Remove n consecutive entries from the list, starting at base.*
- void [remove](#) (free\_list\_type::iterator i, int base, int n)  
*Remove n consecutive entries from the list, starting at base.*
- int [free\\_count](#) () const  
*Return the number of free integers on the list.*

### Private Attributes

- [free\\_list\\_type](#) fl  
*Tracks unused BDD variables.*

#### 12.5.1 Detailed Description

Manage ranges of variables.

#### 12.5.2 Member Typedef Documentation

**12.5.2.1** typedef     std::pair<int,     int>     spot::free\_list::pos\_lenght\_pair   [protected, inherited]

Such pairs describe second free integer starting at first.

**12.5.2.2** `typedef std::list<pos_lenght_pair> spot::free_list::free_list_type` [protected, inherited]

### 12.5.3 Constructor & Destructor Documentation

#### 12.5.3.1 `spot::bdd_allocator::bdd_allocator ()`

Default constructor.

### 12.5.4 Member Function Documentation

#### 12.5.4.1 `static void spot::bdd_allocator::initialize ()` [static]

Initialize the BDD library.

#### 12.5.4.2 `int spot::bdd_allocator::allocate_variables (int n)`

Allocate *n* BDD variables.

#### 12.5.4.3 `void spot::bdd_allocator::release_variables (int base, int n)`

Release *n* BDD variables starting at *base*.

#### 12.5.4.4 `void spot::bdd_allocator::extvarnum (int more)` [private]

Require more variables.

#### 12.5.4.5 `virtual int spot::bdd_allocator::extend (int n)` [private, virtual]

Allocate *n* integer.

This function is called by [register\\_n\(\)](#) when the free list is empty or if *n* consecutive integers could not be found. It should allocate more integers, possibly changing the list, and return the first integer on a range of *n* consecutive integer requested by the user.

Implements [spot::free\\_list](#).

#### 12.5.4.6 `int spot::free_list::register_n (int n)` [inherited]

Find *n* consecutive integers.

Browse the list of free integers until *n* consecutive integers are found. Extend the list (using [extend\(\)](#)) otherwise.

#### Returns:

the first integer of the range

#### 12.5.4.7 `void spot::free_list::release_n (int base, int n)` [inherited]

Release *n* consecutive integers starting at *base*.

**12.5.4.8** `std::ostream& spot::free_list::dump_free_list (std::ostream & os) const` [inherited]

Dump the list to *os* for debugging.

**12.5.4.9** `void spot::free_list::insert (int base, int n)` [inherited]

Extend the list by inserting a new pos-length pair.

**12.5.4.10** `void spot::free_list::remove (int base, int n = 0)` [inherited]

Remove *n* consecutive entries from the list, starting at *base*.

**12.5.4.11** `void spot::free_list::remove (free_list_type::iterator i, int base, int n)` [protected, inherited]

Remove *n* consecutive entries from the list, starting at *base*.

**12.5.4.12** `int spot::free_list::free_count () const` [inherited]

Return the number of free integers on the list.

**12.5.5** Member Data Documentation**12.5.5.1** `bool spot::bdd_allocator::initialized` [static, protected]

Whether the BDD library has been initialized.

**12.5.5.2** `int spot::bdd_allocator::lvarnum` [protected]

number of variables in use in this allocator.

**12.5.5.3** `free_list_type spot::free_list::fl` [protected, inherited]

Tracks unused BDD variables.

The documentation for this class was generated from the following file:

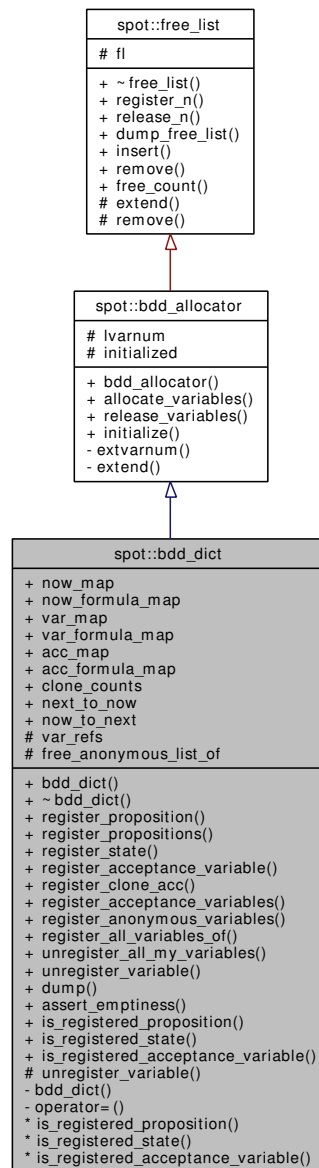
- [misc/bddalloc.hh](#)

**12.6** spot::bdd\_dict Class Reference

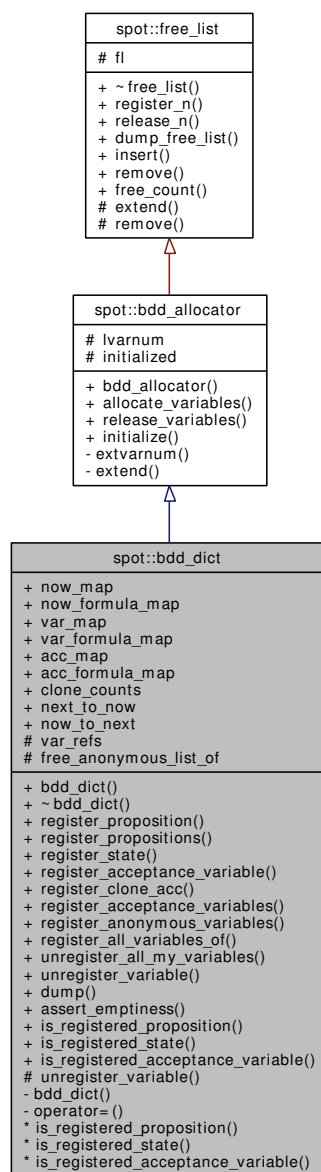
```
#include <tgba/bdddict.hh>
```



Inheritance diagram for spot::bdd\_dict:



Collaboration diagram for spot::bdd\_dict:



## Public Types

- typedef std::map< const [ltl::formula](#) \*, int > [fv\\_map](#)  
*Formula-to-BDD-variable maps.*
- typedef std::map< int, const [ltl::formula](#) \* > [vf\\_map](#)  
*BDD-variable-to-formula maps.*
- typedef std::map< int, int > [cc\\_map](#)  
*Clone counts.*

**Public Member Functions**

- [bdd\\_dict](#) ()
- [~bdd\\_dict](#) ()
- [int register\\_proposition](#) (const [ltl::formula](#) \*f, const void \*for\_me)  
*Register an atomic proposition.*
- [void register\\_propositions](#) (bdd f, const void \*for\_me)  
*Register BDD variables as atomic propositions.*
- [int register\\_state](#) (const [ltl::formula](#) \*f, const void \*for\_me)  
*Register a couple of Now/Next variables.*
- [int register\\_acceptance\\_variable](#) (const [ltl::formula](#) \*f, const void \*for\_me)  
*Register an atomic proposition.*
- [int register\\_clone\\_acc](#) (int var, const void \*for\_me)  
*Clone an acceptance variable VAR for FOR\_ME.*
- [void register\\_acceptance\\_variables](#) (bdd f, const void \*for\_me)  
*Register BDD variables as acceptance variables.*
- [int register\\_anonymous\\_variables](#) (int n, const void \*for\_me)  
*Register anonymous BDD variables.*
- [void register\\_all\\_variables\\_of](#) (const void \*from\_other, const void \*for\_me)  
*Duplicate the variable usage of another object.*
- [void unregister\\_all\\_my\\_variables](#) (const void \*me)  
*Release all variables used by an object.*
- [void unregister\\_variable](#) (int var, const void \*me)  
*Release a variable used by me.*
- [std::ostream & dump](#) (std::ostream &os) const  
*Dump all variables for debugging.*
- [void assert\\_emptyiness](#) () const  
*Make sure the dictionary is empty.*
- [int allocate\\_variables](#) (int n)  
*Allocate n BDD variables.*
- [void release\\_variables](#) (int base, int n)  
*Release n BDD variables starting at base.*
- [bool is\\_registered\\_proposition](#) (const [ltl::formula](#) \*f, const void \*by\_me)
- [bool is\\_registered\\_state](#) (const [ltl::formula](#) \*f, const void \*by\_me)
- [bool is\\_registered\\_acceptance\\_variable](#) (const [ltl::formula](#) \*f, const void \*by\_me)

### Static Public Member Functions

- static void [initialize](#) ()  
*Initialize the BDD library.*

### Public Attributes

- [fv\\_map](#) [now\\_map](#)  
*Maps formulae to "Now" BDD variables.*
- [vf\\_map](#) [now\\_formula\\_map](#)  
*Maps "Now" BDD variables to formulae.*
- [fv\\_map](#) [var\\_map](#)  
*Maps atomic propositions to BDD variables.*
- [vf\\_map](#) [var\\_formula\\_map](#)  
*Maps BDD variables to atomic propositions.*
- [fv\\_map](#) [acc\\_map](#)  
*Maps acceptance conditions to BDD variables.*
- [vf\\_map](#) [acc\\_formula\\_map](#)  
*Maps BDD variables to acceptance conditions.*
- [cc\\_map](#) [clone\\_counts](#)
- [bddPair](#) \* [next\\_to\\_now](#)  
*Map Next variables to Now variables.*
- [bddPair](#) \* [now\\_to\\_next](#)  
*Map Now variables to Next variables.*

### Protected Types

- typedef std::set< const void \* > [ref\\_set](#)  
*BDD-variable reference counts.*
- typedef std::map< int, [ref\\_set](#) > [vr\\_map](#)
- typedef std::map< const void \*, [anon\\_free\\_list](#) > [free\\_anonymous\\_list\\_of\\_type](#)  
*List of unused anonymous variable number for each automaton.*

### Protected Member Functions

- void [unregister\\_variable](#) (vr\_map::iterator &cur, const void \*me)

### Protected Attributes

- [vr\\_map](#) [var\\_refs](#)
- [free\\_anonymous\\_list\\_of\\_type](#) [free\\_anonymous\\_list\\_of](#)
- [int](#) [lvarnum](#)

*number of variables in use in this allocator.*

### Static Protected Attributes

- static [bool](#) [initialized](#)

*Whether the BDD library has been initialized.*

### Private Member Functions

- [bdd\\_dict](#) (const [bdd\\_dict](#) &other)
- [bdd\\_dict](#) & [operator=](#) (const [bdd\\_dict](#) &other)

### Classes

- class [anon\\_free\\_list](#)

#### 12.6.1 Detailed Description

Map BDD variables to formulae.

#### 12.6.2 Member Typedef Documentation

##### 12.6.2.1 `typedef std::map<const ltl::formula*, int> spot::bdd_dict::fv_map`

Formula-to-BDD-variable maps.

##### 12.6.2.2 `typedef std::map<int, const ltl::formula*> spot::bdd_dict::vf_map`

BDD-variable-to-formula maps.

##### 12.6.2.3 `typedef std::map<int, int> spot::bdd_dict::cc_map`

Clone counts.

##### 12.6.2.4 `typedef std::set<const void*> spot::bdd_dict::ref_set` [protected]

BDD-variable reference counts.

##### 12.6.2.5 `typedef std::map<int, ref_set> spot::bdd_dict::vr_map` [protected]

### 12.6.2.6 typedef std::map<const void\*, anon\_free\_list> spot::bdd\_dict::free\_anonymous\_list\_of\_type [protected]

List of unused anonymous variable number for each automaton.

## 12.6.3 Constructor & Destructor Documentation

### 12.6.3.1 spot::bdd\_dict::bdd\_dict ()

### 12.6.3.2 spot::bdd\_dict::~~bdd\_dict ()

### 12.6.3.3 spot::bdd\_dict::bdd\_dict (const bdd\_dict & other) [private]

## 12.6.4 Member Function Documentation

### 12.6.4.1 int spot::bdd\_dict::register\_proposition (const ltl::formula \*f, const void \*for\_me)

Register an atomic proposition.

Return (and maybe allocate) a BDD variable designating formula *f*. The *for\_me* argument should point to the object using this BDD variable, this is used for reference counting. It is perfectly safe to call this function several time with the same arguments.

#### Returns:

The variable number. Use `bdd_ithvar()` or `bdd_nithvar()` to convert this to a BDD.

### 12.6.4.2 void spot::bdd\_dict::register\_propositions (bdd f, const void \*for\_me)

Register BDD variables as atomic propositions.

Register all variables occurring in *f* as atomic propositions used by *for\_me*. This assumes that these atomic propositions are already known from the dictionary (i.e., they have already been registered by [register\\_proposition\(\)](#) for another automaton).

### 12.6.4.3 int spot::bdd\_dict::register\_state (const ltl::formula \*f, const void \*for\_me)

Register a couple of Now/Next variables.

Return (and maybe allocate) two BDD variables for a [state](#) associated to formula *f*. The *for\_me* argument should point to the object using this BDD variable, this is used for reference counting. It is perfectly safe to call this function several time with the same arguments.

#### Returns:

The first variable number. Add one to get the second variable. Use `bdd_ithvar()` or `bdd_nithvar()` to convert this to a BDD.

### 12.6.4.4 int spot::bdd\_dict::register\_acceptance\_variable (const ltl::formula \*f, const void \*for\_me)

Register an atomic proposition.

Return (and maybe allocate) a BDD variable designating an acceptance set associated to formula  $f$ . The *for\_me* argument should point to the object using this BDD variable, this is used for reference counting. It is perfectly safe to call this function several time with the same arguments.

#### Returns:

The variable number. Use `bdd_ithvar()` or `bdd_nithvar()` to convert this to a BDD.

#### 12.6.4.5 int spot::bdd\_dict::register\_clone\_acc (int var, const void \*for\_me)

Clone an acceptance variable VAR for FOR\_ME.

This is used in products TGBAs when both operands share the same acceptance variables but they need to be distinguished in the result.

#### 12.6.4.6 void spot::bdd\_dict::register\_acceptance\_variables (bdd f, const void \*for\_me)

Register BDD variables as acceptance variables.

Register all variables occurring in  $f$  as acceptance variables used by *for\_me*. This assumes that these acceptance variables are already known from the dictionary (i.e., they have already been registered by `register_acceptance_variable()` for another automaton).

#### 12.6.4.7 int spot::bdd\_dict::register\_anonymous\_variables (int n, const void \*for\_me)

Register anonymous BDD variables.

Return (and maybe allocate)  $n$  consecutive BDD variables which will be used only by *for\_me*.

#### Returns:

The variable number. Use `bdd_ithvar()` or `bdd_nithvar()` to convert this to a BDD.

#### 12.6.4.8 void spot::bdd\_dict::register\_all\_variables\_of (const void \*from\_other, const void \*for\_me)

Duplicate the variable usage of another object.

This tells this dictionary that the *for\_me* object will be using the same BDD variables as the *from\_other* objects. This ensure that the variables won't be freed when *from\_other* is deleted if *from\_other* is still alive.

#### 12.6.4.9 void spot::bdd\_dict::unregister\_all\_my\_variables (const void \*me)

Release all variables used by an object.

Usually called in the destructor if *me*.

#### 12.6.4.10 void spot::bdd\_dict::unregister\_variable (int var, const void \*me)

Release a variable used by *me*.

#### 12.6.4.11 bool spot::bdd\_dict::is\_registered\_proposition (const ltl::formula \*f, const void \*by\_me)

Check whether formula  $f$  has already been registered by *by\_me*.

**12.6.4.12** `bool spot::bdd_dict::is_registered_state (const ltl::formula *f, const void *by_me)`

**12.6.4.13** `bool spot::bdd_dict::is_registered_acceptance_variable (const ltl::formula *f, const void *by_me)`

**12.6.4.14** `std::ostream& spot::bdd_dict::dump (std::ostream &os) const`

Dump all variables for debugging.

**Parameters:**

*os* The output stream.

**12.6.4.15** `void spot::bdd_dict::assert_emptiness () const`

Make sure the dictionary is empty.

This will print diagnostics and abort if the dictionary is not empty. Use for debugging.

**12.6.4.16** `void spot::bdd_dict::unregister_variable (vr_map::iterator & cur, const void * me)`  
[protected]

**12.6.4.17** `bdd_dict& spot::bdd_dict::operator= (const bdd_dict & other)` [private]

**12.6.4.18** `static void spot::bdd_allocator::initialize ()` [static, inherited]

Initialize the BDD library.

**12.6.4.19** `int spot::bdd_allocator::allocate_variables (int n)` [inherited]

Allocate *n* BDD variables.

**12.6.4.20** `void spot::bdd_allocator::release_variables (int base, int n)` [inherited]

Release *n* BDD variables starting at *base*.

## 12.6.5 Member Data Documentation

**12.6.5.1** `fv_map spot::bdd_dict::now_map`

Maps formulae to "Now" BDD variables.

**12.6.5.2** `vf_map spot::bdd_dict::now_formula_map`

Maps "Now" BDD variables to formulae.

**12.6.5.3** `fv_map spot::bdd_dict::var_map`

Maps atomic propositions to BDD variables.



**12.6.5.4 `vf_map spot::bdd_dict::var_formula_map`**

Maps BDD variables to atomic propositions.

**12.6.5.5 `fv_map spot::bdd_dict::acc_map`**

Maps acceptance conditions to BDD variables.

**12.6.5.6 `vf_map spot::bdd_dict::acc_formula_map`**

Maps BDD variables to acceptance conditions.

**12.6.5.7 `cc_map spot::bdd_dict::clone_counts`****12.6.5.8 `bddPair* spot::bdd_dict::next_to_now`**

Map Next variables to Now variables.

Use with BuDDy's `bdd_replace()` function.

**12.6.5.9 `bddPair* spot::bdd_dict::now_to_next`**

Map Now variables to Next variables.

Use with BuDDy's `bdd_replace()` function.

**12.6.5.10 `vr_map spot::bdd_dict::var_refs` [protected]****12.6.5.11 `free_anonymous_list_of_type spot::bdd_dict::free_anonymous_list_of` [protected]****12.6.5.12 `bool spot::bdd_allocator::initialized` [static, protected, inherited]**

Whether the BDD library has been initialized.

**12.6.5.13 `int spot::bdd_allocator::lvarnum` [protected, inherited]**

number of variables in use in this allocator.

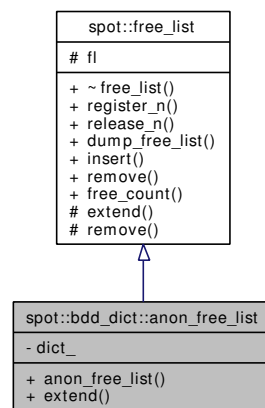
The documentation for this class was generated from the following file:

- [tgba/bdddict.hh](#)

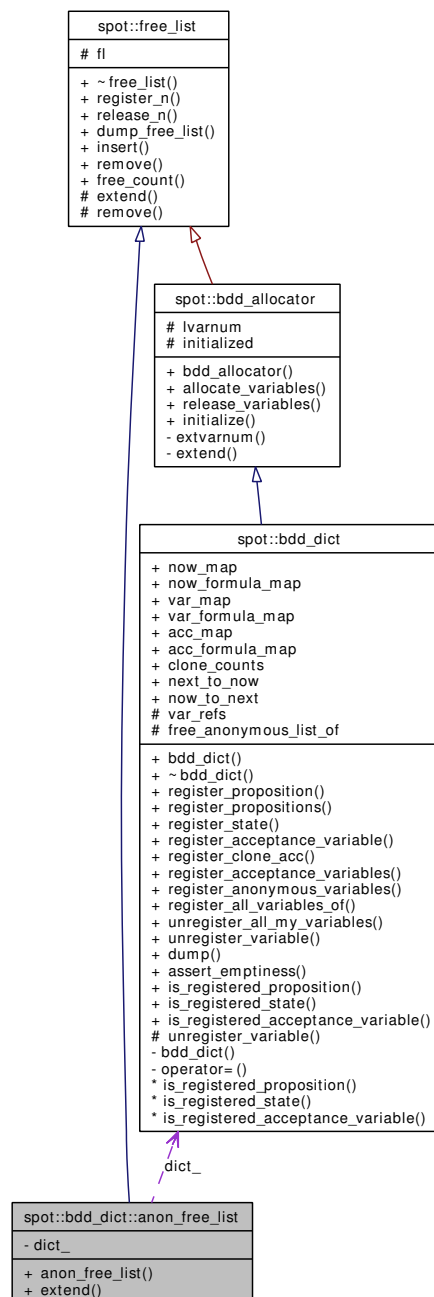
**12.7 `spot::bdd_dict::anon_free_list` Class Reference**

```
#include <tgba/bdddict.hh>
```

Inheritance diagram for spot::bdd\_dict::anon\_free\_list:



Collaboration diagram for spot::bdd\_dict::anon\_free\_list:



## Public Member Functions

- [anon\\_free\\_list](#) ([bdd\\_dict](#) \*d=0)
- virtual int [extend](#) (int n)
- int [register\\_n](#) (int n)  
*Find n consecutive integers.*
- void [release\\_n](#) (int base, int n)

*Release n consecutive integers starting at base.*

- `std::ostream & dump_free_list (std::ostream &os) const`  
*Dump the list to os for debugging.*
- `void insert (int base, int n)`  
*Extend the list by inserting a new pos-lenght pair.*
- `void remove (int base, int n=0)`  
*Remove n consecutive entries from the list, starting at base.*
- `int free_count () const`  
*Return the number of free integers on the list.*

### Protected Types

- `typedef std::pair< int, int > pos_lenght_pair`  
*Such pairs describe second free integer starting at first.*
- `typedef std::list< pos_lenght_pair > free_list_type`

### Protected Member Functions

- `void remove (free_list_type::iterator i, int base, int n)`  
*Remove n consecutive entries from the list, starting at base.*

### Protected Attributes

- `free_list_type fl`  
*Tracks unused BDD variables.*

### Private Attributes

- `bdd_dict * dict_`

## 12.7.1 Member Typedef Documentation

**12.7.1.1** `typedef std::pair<int, int> spot::free_list::pos_lenght_pair` [protected, inherited]

Such pairs describe second free integer starting at first.

**12.7.1.2** `typedef std::list<pos_lenght_pair> spot::free_list::free_list_type` [protected, inherited]

## 12.7.2 Constructor & Destructor Documentation

### 12.7.2.1 `spot::bdd_dict::anon_free_list::anon_free_list (bdd_dict * d = 0)`

## 12.7.3 Member Function Documentation

### 12.7.3.1 `virtual int spot::bdd_dict::anon_free_list::extend (int n)` `[virtual]`

Allocate *n* integer.

This function is called by [register\\_n\(\)](#) when the free list is empty or if *n* consecutive integers could not be found. It should allocate more integers, possibly changing the list, and return the first integer on a range of *n* consecutive integer requested by the user.

Implements [spot::free\\_list](#).

### 12.7.3.2 `int spot::free_list::register_n (int n)` `[inherited]`

Find *n* consecutive integers.

Browse the list of free integers until *n* consecutive integers are found. Extend the list (using [extend\(\)](#)) otherwise.

#### Returns:

the first integer of the range

### 12.7.3.3 `void spot::free_list::release_n (int base, int n)` `[inherited]`

Release *n* consecutive integers starting at *base*.

### 12.7.3.4 `std::ostream& spot::free_list::dump_free_list (std::ostream & os) const` `[inherited]`

Dump the list to *os* for debugging.

### 12.7.3.5 `void spot::free_list::insert (int base, int n)` `[inherited]`

Extend the list by inserting a new pos-length pair.

### 12.7.3.6 `void spot::free_list::remove (int base, int n = 0)` `[inherited]`

Remove *n* consecutive entries from the list, starting at *base*.

### 12.7.3.7 `void spot::free_list::remove (free_list_type::iterator i, int base, int n)` `[protected, inherited]`

Remove *n* consecutive entries from the list, starting at *base*.

### 12.7.3.8 `int spot::free_list::free_count () const` `[inherited]`

Return the number of free integers on the list.

## 12.7.4 Member Data Documentation

**12.7.4.1** `bdd_dict*` `spot::bdd_dict::anon_free_list::dict_` `[private]`

**12.7.4.2** `free_list_type` `spot::free_list::fl` `[protected, inherited]`

Tracks unused BDD variables.

The documentation for this class was generated from the following file:

- [tgba/bdddict.hh](#)

## 12.8 spot::bdd\_less\_than Struct Reference

Comparison functor for BDDs.

```
#include <misc/bddlt.hh>
```

### Public Member Functions

- `bool` `operator()` (`const bdd &left`, `const bdd &right`) `const`

### 12.8.1 Detailed Description

Comparison functor for BDDs.

### 12.8.2 Member Function Documentation

**12.8.2.1** `bool` `spot::bdd_less_than::operator()` (`const bdd & left`, `const bdd & right`) `const` `[inline]`

The documentation for this struct was generated from the following file:

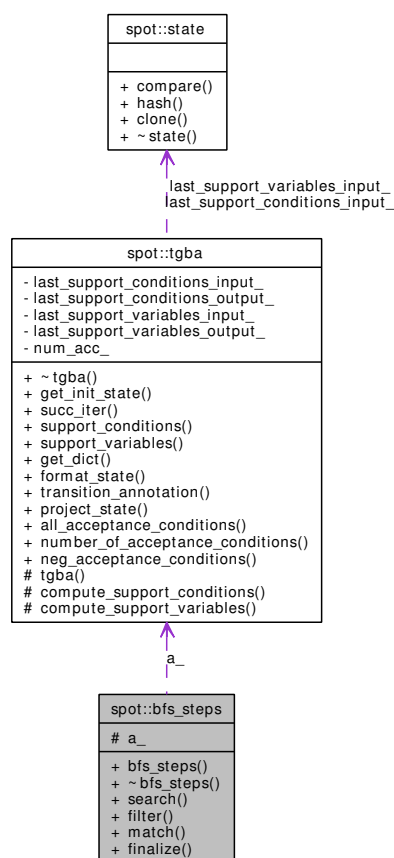
- [misc/bddlt.hh](#)

## 12.9 spot::bfs\_steps Class Reference

Make a BFS in a [spot::tgba](#) to compute a [tgba\\_run::steps](#).

```
#include <tgbaalgos/bfssteps.hh>
```

Collaboration diagram for spot::bfs\_steps:



## Public Member Functions

- [bfs\\_steps](#) (const [tgba](#) \*a)
- virtual [~bfs\\_steps](#) ()
- const [state](#) \* [search](#) (const [state](#) \*start, [tgba\\_run::steps](#) &l)  
*Start the search from start, and append the resulting path (if any) to l.*
- virtual const [state](#) \* [filter](#) (const [state](#) \*s)=0  
*Return a state\* that is unique for a.*
- virtual bool [match](#) ([tgba\\_run::step](#) &step, const [state](#) \*dest)=0  
*Whether a new transition completes a path.*
- virtual void [finalize](#) (const std::map< const [state](#) \*, [tgba\\_run::step](#), [state\\_ptr\\_less\\_than](#) > &father, const [tgba\\_run::step](#) &s, const [state](#) \*start, [tgba\\_run::steps](#) &l)  
*Append the resulting path to the resulting run.*

## Protected Attributes

- const [tgba](#) \* [a\\_](#)

The *spot::tgba* we are searching into.

### 12.9.1 Detailed Description

Make a BFS in a *spot::tgba* to compute a *tgba\_run::steps*.

This class should be used to compute the shortest path between a *state* of a *spot::tgba* and the first transition or *state* that matches some conditions.

These conditions should be specified by defining *bfs\_steps::match()* in a subclass. Also the search can be restricted to some set of states with a proper definition of *bfs\_steps::filter()*.

### 12.9.2 Constructor & Destructor Documentation

#### 12.9.2.1 spot::bfs\_steps::bfs\_steps (const tgba \* a)

#### 12.9.2.2 virtual spot::bfs\_steps::~~bfs\_steps () [virtual]

### 12.9.3 Member Function Documentation

#### 12.9.3.1 const state\* spot::bfs\_steps::search (const state \* start, tgba\_run::steps & l)

Start the search from *start*, and append the resulting path (if any) to *l*.

##### Returns:

the destination *state* of the last step (not included in *l*) if a matching path was found, or 0 otherwise.

#### 12.9.3.2 virtual const state\* spot::bfs\_steps::filter (const state \* s) [pure virtual]

Return a *state\** that is unique for *a*.

*bfs\_steps* does not do handle the memory for the states it generates, this is the job of *filter()*. Here *s* is a new *state\** that *search()* has just allocated (using *tgba\_succ\_iterator::current\_state()*), and the return of this function should be a *state\** that does not need to be freed by *search()*.

If you already have a map or a set which uses states as keys, you should probably arrange for *filter()* to return these keys, and delete *s*. Otherwise you will have to define such a set, just to be able to delete all the *state\** in a subclass.

This function can return 0 if the given *state* should not be explored.

#### 12.9.3.3 virtual bool spot::bfs\_steps::match (tgba\_run::step & step, const state \* dest) [pure virtual]

Whether a new transition completes a path.

This function is called immediately after each call to *filter()* that does not return 0.

##### Parameters:

*step* the source *state* (as returned by *filter()*), and the labels of the outgoing transition

*dest* the destination *state* (as returned by *filter()*)



**Returns:**

true iff a path that included this step should be accepted.

The [search\(\)](#) algorithm stops as soon as [match\(\)](#) returns true, and when this happens the list argument of [search\(\)](#) is augmented with the shortest path that ends with this transition.

**12.9.3.4 virtual void spot::bfs\_steps::finalize (const std::map< const state \*, tgba\_run::step, state\_ptr\_less\_than > & father, const tgba\_run::step & s, const state \* start, tgba\_run::steps & l)**  
[virtual]

Append the resulting path to the resulting run.

This is called after [match\(\)](#) has returned true, to append the resulting path to *l*. This seldom needs to be overridden, unless you do not want *l* to be updated (in which case an empty [finalize\(\)](#) will do).

**12.9.4 Member Data Documentation**

**12.9.4.1 const tgba\* spot::bfs\_steps::a\_** [protected]

The [spot::tgba](#) we are searching into.

The documentation for this class was generated from the following file:

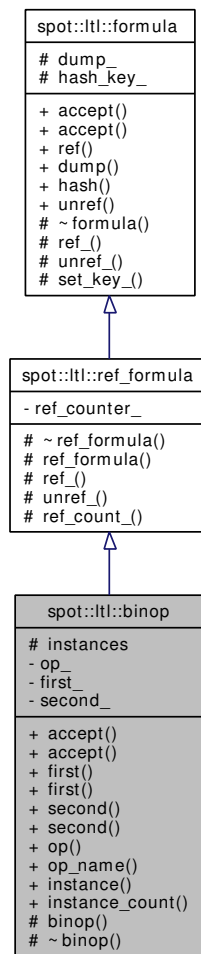
- [tgbaalgos/bfssteps.hh](#)

**12.10 spot::ltl::binop Class Reference**

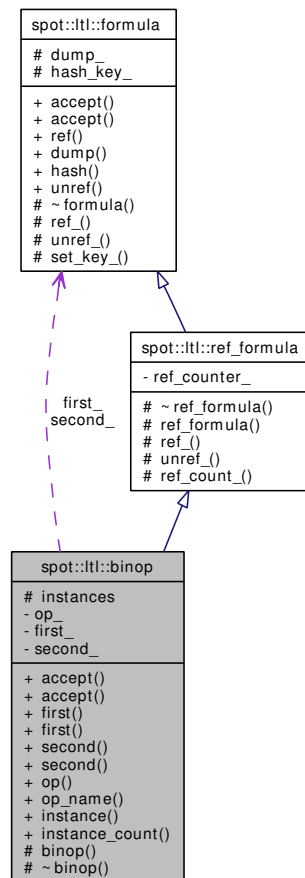
Binary operator.

```
#include <ltlast/binop.hh>
```

Inheritance diagram for spot::ltl::binop:



Collaboration diagram for spot::ltl::binop:



## Public Types

- enum `type` {  
`Xor`, `Implies`, `Equiv`, `U`,  
`R` }

## Public Member Functions

- virtual void `accept` (`visitor` &`v`)  
*Entry point for vspot::ltl::visitor instances.*
- virtual void `accept` (`const_visitor` &`v`) const  
*Entry point for vspot::ltl::const\_visitor instances.*
- const `formula` \* `first` () const  
*Get the first operand.*
- `formula` \* `first` ()  
*Get the first operand.*

- const formula \* second () const  
*Get the second operand.*
- formula \* second ()  
*Get the second operand.*
- type op () const  
*Get the type of this operator.*
- const char \* op\_name () const  
*Get the type of this operator, as a string.*
- formula \* ref ()  
*clone this node*
- const std::string & dump () const  
*Return a canonic representation of the formula.*
- const size\_t hash () const  
*Return a hash\_key for the formula.*

### Static Public Member Functions

- static binop \* instance (type op, formula \*first, formula \*second)
- static unsigned instance\_count ()  
*Number of instantiated binary operators. For debugging.*
- static void unref (formula \*f)  
*release this node*

### Protected Types

- typedef std::pair< formula \*, formula \* > pairf
- typedef std::pair< type, pairf > pair
- typedef std::map< pair, formula \* > map

### Protected Member Functions

- binop (type op, formula \*first, formula \*second)
- virtual ~binop ()
- void ref\_ ()  
*increment reference counter if any*
- bool unref\_ ()  
*decrement reference counter if any, return true when the instance must be deleted (usually when the counter hits 0).*

- unsigned `ref_count_()`  
*Number of references to this [formula](#).*
- void `set_key_()`  
*Compute `key_` from `dump_`.*

### Protected Attributes

- std::string `dump_`  
*The canonic representation of the [formula](#).*
- size\_t `hash_key_`  
*The hash key of this [formula](#).*

### Static Protected Attributes

- static `map instances`

### Private Attributes

- `type op_`
- `formula * first_`
- `formula * second_`

## 12.10.1 Detailed Description

Binary operator.

## 12.10.2 Member Typedef Documentation

**12.10.2.1** `typedef std::pair<formula*, formula*> spot::ltl::binop::pairf` [protected]

**12.10.2.2** `typedef std::pair<type, pairf> spot::ltl::binop::pair` [protected]

**12.10.2.3** `typedef std::map<pair, formula*> spot::ltl::binop::map` [protected]

## 12.10.3 Member Enumeration Documentation

### 12.10.3.1 enum spot::ltl::binop::type

Different kinds of binary opertaors

And and Or are not here. Because they are often nested we represent them as multops.

#### Enumerator:

*Xor*

*Implies**Equiv**U**R*

## 12.10.4 Constructor & Destructor Documentation

**12.10.4.1** `spot::ltl::binop::binop (type op, formula * first, formula * second)` [protected]

**12.10.4.2** `virtual spot::ltl::binop::~~binop ()` [protected, virtual]

## 12.10.5 Member Function Documentation

**12.10.5.1** `static binop* spot::ltl::binop::instance (type op, formula * first, formula * second)`  
[static]

Build an unary operator with operation *op* and children *first* and *second*.

**12.10.5.2** `virtual void spot::ltl::binop::accept (visitor & v)` [virtual]

Entry point for vspot::ltl::visitor instances.

Implements [spot::ltl::formula](#).

**12.10.5.3** `virtual void spot::ltl::binop::accept (const_visitor & v) const` [virtual]

Entry point for vspot::ltl::const\_visitor instances.

Implements [spot::ltl::formula](#).

**12.10.5.4** `const formula* spot::ltl::binop::first () const`

Get the first operand.

**12.10.5.5** `formula* spot::ltl::binop::first ()`

Get the first operand.

**12.10.5.6** `const formula* spot::ltl::binop::second () const`

Get the second operand.

**12.10.5.7** `formula* spot::ltl::binop::second ()`

Get the second operand.

**12.10.5.8** `type spot::ltl::binop::op () const`

Get the type of this operator.

**12.10.5.9** `const char* spot::ltl::binop::op_name () const`

Get the type of this operator, as a string.

**12.10.5.10** `static unsigned spot::ltl::binop::instance_count () [static]`

Number of instantiated binary operators. For debugging.

**12.10.5.11** `void spot::ltl::ref_formula::ref_ () [protected, virtual, inherited]`

increment reference counter if any

Reimplemented from [spot::ltl::formula](#).

**12.10.5.12** `bool spot::ltl::ref_formula::unref_ () [protected, virtual, inherited]`

decrement reference counter if any, return true when the instance must be deleted (usually when the counter hits 0).

Reimplemented from [spot::ltl::formula](#).

**12.10.5.13** `unsigned spot::ltl::ref_formula::ref_count_ () [protected, inherited]`

Number of references to this [formula](#).

**12.10.5.14** `formula* spot::ltl::formula::ref () [inherited]`

clone this node

This increments the reference counter of this node (if one is used). You should almost never use this method directly as it doesn't touch the children. If you want to clone a whole [formula](#), use [spot::ltl::clone\(\)](#) instead.

**12.10.5.15** `static void spot::ltl::formula::unref (formula *f) [static, inherited]`

release this node

This decrements the reference counter of this node (if one is used) and can free the object. You should almost never use this method directly as it doesn't touch the children. If you want to release a whole [formula](#), use [spot::ltl::destroy\(\)](#) instead.

**12.10.5.16** `const std::string& spot::ltl::formula::dump () const [inherited]`

Return a canonic representation of the [formula](#).

**12.10.5.17** `const size_t spot::ltl::formula::hash () const [inline, inherited]`

Return a hash\_key for the [formula](#).

**12.10.5.18** `void spot::ltl::formula::set_key_ () [protected, inherited]`

Compute key\_ from dump\_.

Should be called once in each object, after dump\_ has been set.

### 12.10.6 Member Data Documentation

**12.10.6.1** `map spot::ltl::binop::instances` `[static, protected]`

**12.10.6.2** `type spot::ltl::binop::op_` `[private]`

**12.10.6.3** `formula* spot::ltl::binop::first_` `[private]`

**12.10.6.4** `formula* spot::ltl::binop::second_` `[private]`

**12.10.6.5** `std::string spot::ltl::formula::dump_` `[protected, inherited]`

The canonic representation of the [formula](#).

**12.10.6.6** `size_t spot::ltl::formula::hash_key_` `[protected, inherited]`

The hash key of this [formula](#).

Initialized by [set\\_key\\_\(\)](#).

The documentation for this class was generated from the following file:

- [ltlast/binop.hh](#)

## 12.11 spot::char\_ptr\_less\_than Struct Reference

Strict Weak Ordering for `char*`.

```
#include <misc/ltstr.hh>
```

### Public Member Functions

- `bool operator()` (`const char *left`, `const char *right`) `const`

### 12.11.1 Detailed Description

Strict Weak Ordering for `char*`.

This is meant to be used as a comparison functor for STL map whose key are of type `const char*`.

For instance here is how one could declare a map of `const state*`.

```
std::map<const char*, int, spot::state_ptr_less_than> seen;
```

### 12.11.2 Member Function Documentation

**12.11.2.1** `bool spot::char_ptr_less_than::operator()` (`const char * left`, `const char * right`) `const` `[inline]`

The documentation for this struct was generated from the following file:

- [misc/ltstr.hh](#)

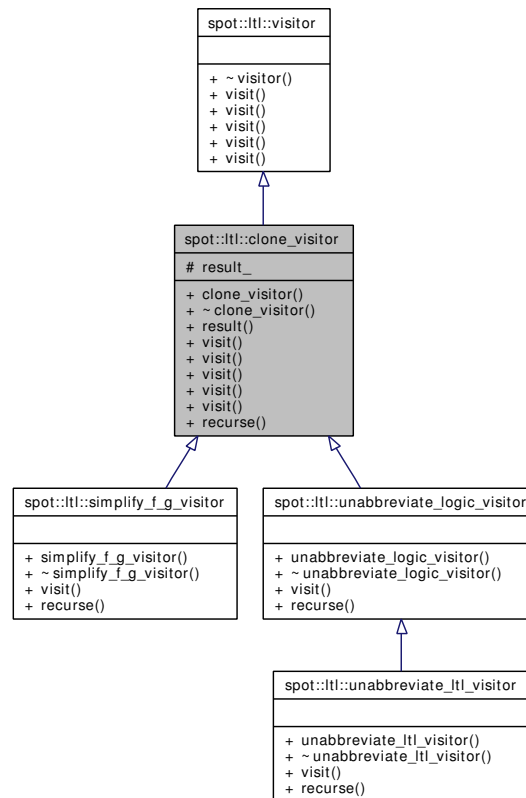


## 12.12 spot::ltl::clone\_visitor Class Reference

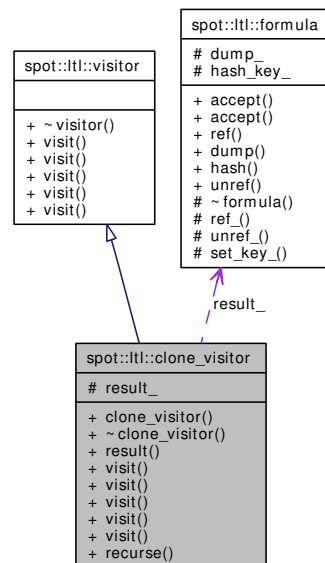
Clone a [formula](#).

```
#include <ltlvisit/clone.hh>
```

Inheritance diagram for spot::ltl::clone\_visitor:



Collaboration diagram for spot::ltl::clone\_visitor:



## Public Member Functions

- [clone\\_visitor\(\)](#)
- virtual [~clone\\_visitor\(\)](#)
- [formula \\* result\(\)](#) const
- void [visit\(atomic\\_prop \\*ap\)](#)
- void [visit\(unop \\*uo\)](#)
- void [visit\(binop \\*bo\)](#)
- void [visit\(multop \\*mo\)](#)
- void [visit\(constant \\*c\)](#)
- virtual [formula \\* recurse\(formula \\*f\)](#)

## Protected Attributes

- [formula \\* result\\_](#)

### 12.12.1 Detailed Description

Clone a [formula](#).

This [visitor](#) is public, because it's convenient to derive from it and override part of its methods. But if you just want the functionality, consider using [spot::ltl::clone](#) instead.

### 12.12.2 Constructor & Destructor Documentation

#### 12.12.2.1 spot::ltl::clone\_visitor::clone\_visitor()

#### 12.12.2.2 virtual spot::ltl::clone\_visitor::~~clone\_visitor() [virtual]

### 12.12.3 Member Function Documentation

**12.12.3.1** formula\* spot::ltl::clone\_visitor::result () const

**12.12.3.2** void spot::ltl::clone\_visitor::visit (atomic\_prop \* *ap*) [virtual]

Implements [spot::ltl::visitor](#).

**12.12.3.3** void spot::ltl::clone\_visitor::visit (unop \* *uo*) [virtual]

Implements [spot::ltl::visitor](#).

Reimplemented in [spot::ltl::unabbreviate\\_ltl\\_visitor](#).

**12.12.3.4** void spot::ltl::clone\_visitor::visit (binop \* *bo*) [virtual]

Implements [spot::ltl::visitor](#).

Reimplemented in [spot::ltl::unabbreviate\\_logic\\_visitor](#), and [spot::ltl::simplify\\_f\\_g\\_visitor](#).

**12.12.3.5** void spot::ltl::clone\_visitor::visit (multop \* *mo*) [virtual]

Implements [spot::ltl::visitor](#).

**12.12.3.6** void spot::ltl::clone\_visitor::visit (constant \* *c*) [virtual]

Implements [spot::ltl::visitor](#).

**12.12.3.7** virtual formula\* spot::ltl::clone\_visitor::recurse (formula \* *f*) [virtual]

Reimplemented in [spot::ltl::unabbreviate\\_logic\\_visitor](#), [spot::ltl::simplify\\_f\\_g\\_visitor](#), and [spot::ltl::unabbreviate\\_ltl\\_visitor](#).

### 12.12.4 Member Data Documentation

**12.12.4.1** formula\* spot::ltl::clone\_visitor::result\_ [protected]

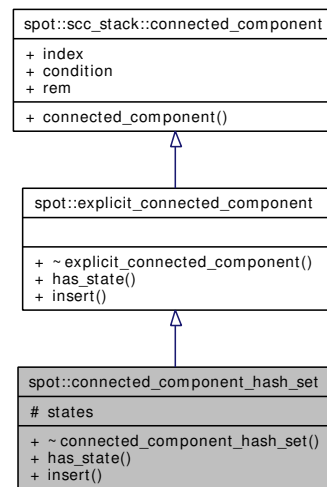
The documentation for this class was generated from the following file:

- [ltlvisit/clone.hh](#)

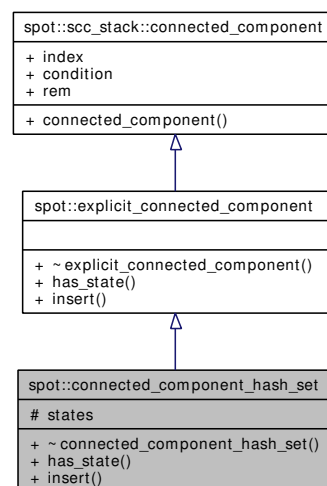
## 12.13 spot::connected\_component\_hash\_set Class Reference

```
#include <tgbaalgos/gtec/explscs.hh>
```

Inheritance diagram for spot::connected\_component\_hash\_set:



Collaboration diagram for spot::connected\_component\_hash\_set:



### Public Member Functions

- virtual `~connected_component_hash_set()`
- virtual const `state * has_state (const state *s)` const

*Check if the SCC contains states s.*

- virtual void `insert (const state *s)`

*Insert a new state in the SCC.*

### Public Attributes

- int `index`

*Index of the SCC.*

- bdd [condition](#)
- std::list< const [state](#) \* > [rem](#)

### Protected Types

- typedef Sgi::hash\_set< const [state](#) \*, [state\\_ptr\\_hash](#), [state\\_ptr\\_equal](#) > [set\\_type](#)

### Protected Attributes

- [set\\_type](#) [states](#)

### 12.13.1 Detailed Description

A straightforward implementation of [explicit\\_connected\\_component](#) using a hash.

### 12.13.2 Member Typedef Documentation

**12.13.2.1** typedef Sgi::hash\_set<const state\*, state\_ptr\_hash, state\_ptr\_equal> spot::connected\_component\_hash\_set::set\_type [protected]

### 12.13.3 Constructor & Destructor Documentation

**12.13.3.1** virtual spot::connected\_component\_hash\_set::~~connected\_component\_hash\_set () [inline, virtual]

### 12.13.4 Member Function Documentation

**12.13.4.1** virtual const state\* spot::connected\_component\_hash\_set::has\_state (const state \* s) const [virtual]

Check if the SCC contains states *s*.

Return the representative of *s* in the SCC, and delete *s* if it is different (acting like numbered\_state\_heap::filter), or 0 otherwise.

Implements [spot::explicit\\_connected\\_component](#).

**12.13.4.2** virtual void spot::connected\_component\_hash\_set::insert (const state \* s) [virtual]

Insert a new [state](#) in the SCC.

Implements [spot::explicit\\_connected\\_component](#).

### 12.13.5 Member Data Documentation

**12.13.5.1** set\_type spot::connected\_component\_hash\_set::states [protected]

**12.13.5.2** `int spot::scc_stack::connected_component::index` [inherited]

Index of the SCC.

**12.13.5.3** `bdd spot::scc_stack::connected_component::condition` [inherited]

The bdd condition is the union of all acceptance conditions of transitions which connect the states of the connected component.

**12.13.5.4** `std::list<const state*> spot::scc_stack::connected_component::rem` [inherited]

The documentation for this class was generated from the following file:

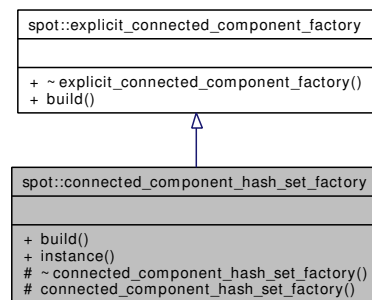
- [tgbaalgos/gtec/explscc.hh](#)

**12.14** `spot::connected_component_hash_set_factory` Class Reference

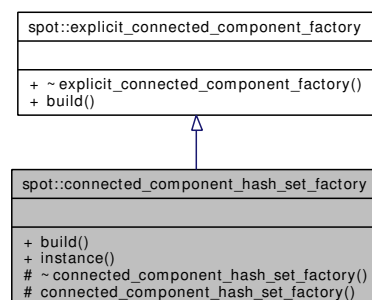
Factory for [connected\\_component\\_hash\\_set](#).

```
#include <tgbaalgos/gtec/explscc.hh>
```

Inheritance diagram for `spot::connected_component_hash_set_factory`:



Collaboration diagram for `spot::connected_component_hash_set_factory`:

**Public Member Functions**

- virtual [connected\\_component\\_hash\\_set](#) \* `build ()` const

*Create an [explicit\\_connected\\_component](#).*

### Static Public Member Functions

- static const [connected\\_component\\_hash\\_set\\_factory](#) \* [instance](#) ()

*Get the unique instance of this class.*

### Protected Member Functions

- virtual [~connected\\_component\\_hash\\_set\\_factory](#) ()
- [connected\\_component\\_hash\\_set\\_factory](#) ()

*Construction is forbidden.*

#### 12.14.1 Detailed Description

Factory for [connected\\_component\\_hash\\_set](#).

This class is a singleton. Retrieve the instance using [instance\(\)](#).

#### 12.14.2 Constructor & Destructor Documentation

**12.14.2.1 virtual spot::connected\_component\_hash\_set\_factory::~~connected\_component\_hash\_set\_factory ()** [inline, protected, virtual]

**12.14.2.2 spot::connected\_component\_hash\_set\_factory::connected\_component\_hash\_set\_factory ()** [protected]

Construction is forbidden.

#### 12.14.3 Member Function Documentation

**12.14.3.1 virtual connected\_component\_hash\_set\* spot::connected\_component\_hash\_set\_factory::build () const** [virtual]

Create an [explicit\\_connected\\_component](#).

Implements [spot::explicit\\_connected\\_component\\_factory](#).

**12.14.3.2 static const connected\_component\_hash\_set\_factory\* spot::connected\_component\_hash\_set\_factory::instance ()** [static]

Get the unique instance of this class.

The documentation for this class was generated from the following file:

- [tgbaalgos/gtec/explsc.hh](#)

## 12.15 spot::ltl::const\_visitor Struct Reference

Formula [visitor](#) that cannot modify the [formula](#).

```
#include <ltlast/visitor.hh>
```

## Public Member Functions

- virtual [~const\\_visitor](#) ()
- virtual void [visit](#) (const [atomic\\_prop](#) \*node)=0
- virtual void [visit](#) (const [constant](#) \*node)=0
- virtual void [visit](#) (const [binop](#) \*node)=0
- virtual void [visit](#) (const [unop](#) \*node)=0
- virtual void [visit](#) (const [multop](#) \*node)=0

### 12.15.1 Detailed Description

Formula [visitor](#) that cannot modify the [formula](#).

Writing visitors is the preferred way to traverse a [formula](#), since it doesn't involve any cast.

If you want to modify the visited [formula](#), inherit from [spot::ltl:visitor](#) instead.

### 12.15.2 Constructor & Destructor Documentation

**12.15.2.1** virtual [spot::ltl::const\\_visitor::~~const\\_visitor](#) () [inline, virtual]

### 12.15.3 Member Function Documentation

**12.15.3.1** virtual void [spot::ltl::const\\_visitor::visit](#) (const [atomic\\_prop](#) \* *node*) [pure virtual]

**12.15.3.2** virtual void [spot::ltl::const\\_visitor::visit](#) (const [constant](#) \* *node*) [pure virtual]

**12.15.3.3** virtual void [spot::ltl::const\\_visitor::visit](#) (const [binop](#) \* *node*) [pure virtual]

**12.15.3.4** virtual void [spot::ltl::const\\_visitor::visit](#) (const [unop](#) \* *node*) [pure virtual]

**12.15.3.5** virtual void [spot::ltl::const\\_visitor::visit](#) (const [multop](#) \* *node*) [pure virtual]

The documentation for this struct was generated from the following file:

- [ltlast/visitor.hh](#)

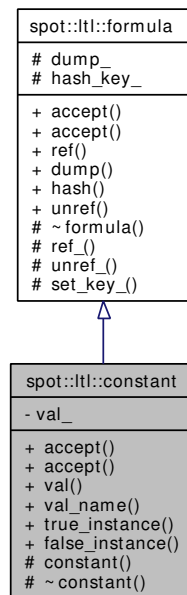
## 12.16 spot::ltl::constant Class Reference

A [constant](#) (True or False).

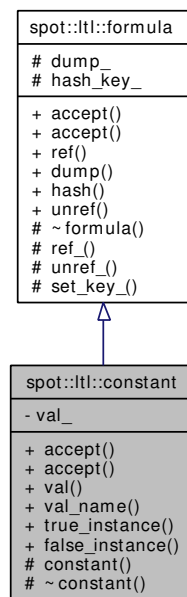
```
#include <ltlast/constant.hh>
```



Inheritance diagram for spot::ltl::constant:



Collaboration diagram for spot::ltl::constant:



## Public Types

- enum `type` { `False`, `True` }

## Public Member Functions

- virtual void `accept` (`visitor` &`v`)

*Entry point for vspot::ltl::visitor instances.*

- virtual void `accept (const_visitor &v)` const  
*Entry point for vspot::ltl::const\_visitor instances.*
- `type val ()` const  
*Return the value of the `constant`.*
- const char \* `val_name ()` const  
*Return the value of the `constant` as a string.*
- `formula * ref ()`  
*clone this node*
- const std::string & `dump ()` const  
*Return a canonic representation of the `formula`.*
- const size\_t `hash ()` const  
*Return a hash\_key for the `formula`.*

### Static Public Member Functions

- static `constant * true_instance ()`  
*Get the sole instance of spot::ltl::constant::constant(True).*
- static `constant * false_instance ()`  
*Get the sole instance of spot::ltl::constant::constant(False).*
- static void `unref (formula *f)`  
*release this node*

### Protected Member Functions

- `constant (type val)`
- virtual `~constant ()`
- virtual void `ref_ ()`  
*increment reference counter if any*
- virtual bool `unref_ ()`  
*decrement reference counter if any, return true when the instance must be deleted (usually when the counter hits 0).*
- void `set_key_ ()`  
*Compute key\_ from dump\_.*

### Protected Attributes

- std::string [dump\\_](#)  
*The canonic representation of the [formula](#).*
- size\_t [hash\\_key\\_](#)  
*The hash key of this [formula](#).*

### Private Attributes

- [type val\\_](#)

#### 12.16.1 Detailed Description

A [constant](#) (True or False).

#### 12.16.2 Member Enumeration Documentation

##### 12.16.2.1 enum spot::ltl::constant::type

Enumerator:

*False*  
*True*

#### 12.16.3 Constructor & Destructor Documentation

**12.16.3.1** spot::ltl::constant::constant (type *val*) [protected]

**12.16.3.2** virtual spot::ltl::constant::~~constant () [protected, virtual]

#### 12.16.4 Member Function Documentation

**12.16.4.1** virtual void spot::ltl::constant::accept (visitor & *v*) [virtual]

Entry point for vspot::ltl::visitor instances.

Implements [spot::ltl::formula](#).

**12.16.4.2** virtual void spot::ltl::constant::accept (const\_visitor & *v*) const [virtual]

Entry point for vspot::ltl::const\_visitor instances.

Implements [spot::ltl::formula](#).

**12.16.4.3** type spot::ltl::constant::val () const

Return the value of the [constant](#).

**12.16.4.4 const char\* spot::ltl::constant::val\_name () const**

Return the value of the [constant](#) as a string.

**12.16.4.5 static constant\* spot::ltl::constant::true\_instance () [static]**

Get the sole instance of `spot::ltl::constant::constant(True)`.

**12.16.4.6 static constant\* spot::ltl::constant::false\_instance () [static]**

Get the sole instance of `spot::ltl::constant::constant(False)`.

**12.16.4.7 formula\* spot::ltl::formula::ref () [inherited]**

clone this node

This increments the reference counter of this node (if one is used). You should almost never use this method directly as it doesn't touch the children. If you want to clone a whole [formula](#), use [spot::ltl::clone\(\)](#) instead.

**12.16.4.8 static void spot::ltl::formula::unref (formula \*f) [static, inherited]**

release this node

This decrements the reference counter of this node (if one is used) and can free the object. You should almost never use this method directly as it doesn't touch the children. If you want to release a whole [formula](#), use [spot::ltl::destroy\(\)](#) instead.

**12.16.4.9 const std::string& spot::ltl::formula::dump () const [inherited]**

Return a canonic representation of the [formula](#).

**12.16.4.10 const size\_t spot::ltl::formula::hash () const [inline, inherited]**

Return a hash\_key for the [formula](#).

**12.16.4.11 virtual void spot::ltl::formula::ref\_ () [protected, virtual, inherited]**

increment reference counter if any

Reimplemented in [spot::ltl::ref\\_formula](#).

**12.16.4.12 virtual bool spot::ltl::formula::unref\_ () [protected, virtual, inherited]**

decrement reference counter if any, return true when the instance must be deleted (usually when the counter hits 0).

Reimplemented in [spot::ltl::ref\\_formula](#).

**12.16.4.13 void spot::ltl::formula::set\_key\_ () [protected, inherited]**

Compute key\_ from dump\_.

Should be called once in each object, after dump\_ has been set.

### 12.16.5 Member Data Documentation

**12.16.5.1** type `spot::ltl::constant::val_` [private]

**12.16.5.2** `std::string spot::ltl::formula::dump_` [protected, inherited]

The canonic representation of the [formula](#).

**12.16.5.3** `size_t spot::ltl::formula::hash_key_` [protected, inherited]

The hash key of this [formula](#).

Initialized by [set\\_key\\_\(\)](#).

The documentation for this class was generated from the following file:

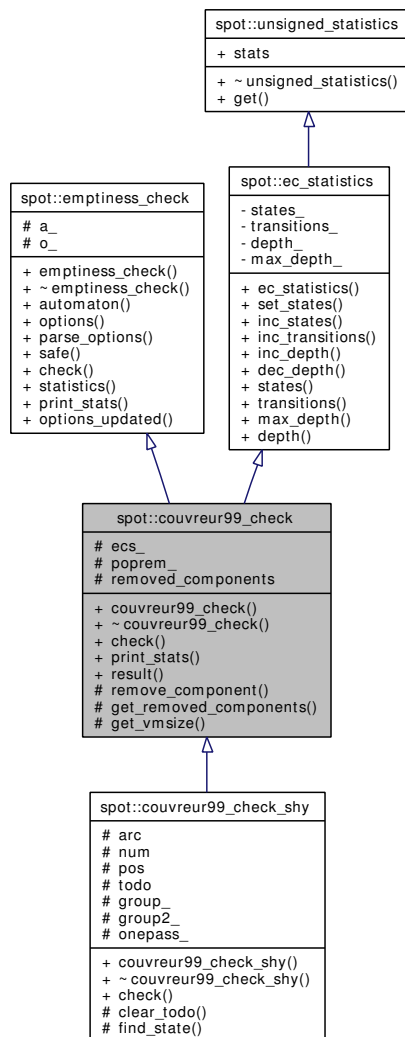
- [ltlast/constant.hh](#)

## 12.17 spot::couvreur99\_check Class Reference

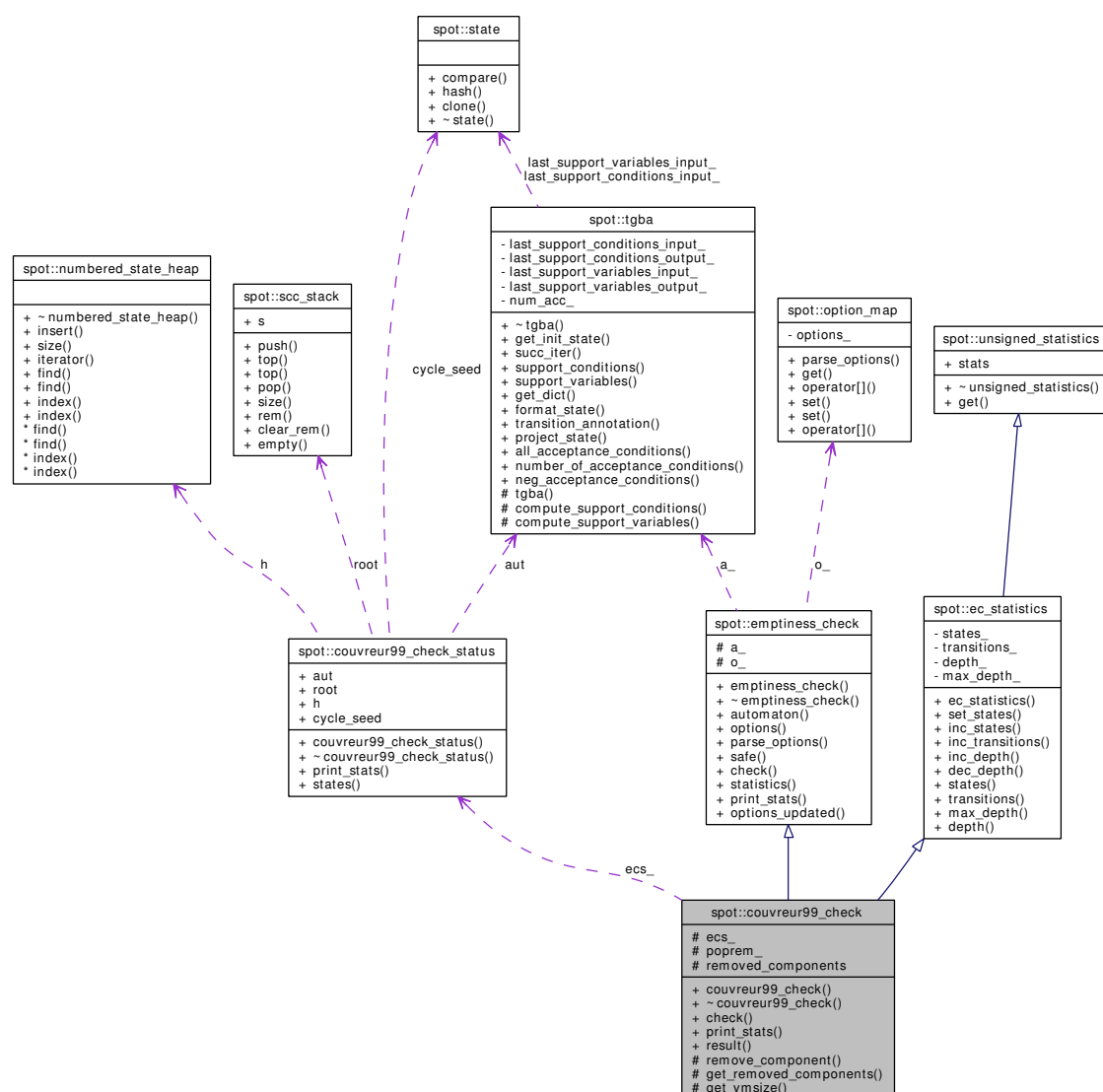
An implementation of the Couvreur99 emptiness-check algorithm.

```
#include <tgbaaalgos/gtec/gtec.hh>
```

Inheritance diagram for spot::couvreur99\_check:



Collaboration diagram for spot::couvreur99\_check:



## Public Types

- typedef unsigned(unsigned\_statistics::\*) [unsigned\\_fun](#) () const
- typedef std::map< const char \*, [unsigned\\_fun](#), char\_ptr\_less\_than > [stats\\_map](#)

## Public Member Functions

- [couvreur99\\_check](#) (const [tgba](#) \*a, [option\\_map](#) o=[option\\_map](#)(), const [numbered\\_state\\_heap\\_factory](#) \*nshf=[numbered\\_state\\_heap\\_hash\\_map\\_factory::instance\(\)](#))
- virtual [~couvreur99\\_check](#) ()
- virtual [emptiness\\_check\\_result](#) \* [check](#) ()  
*Check whether the automaton's language is empty.*
- virtual std::ostream & [print\\_stats](#) (std::ostream &os) const

*Print statistics, if any.*

- const [couvreur99\\_check\\_status](#) \* [result](#) () const  
*Return the status of the emptiness-check.*
- const [tgba](#) \* [automaton](#) () const  
*The automaton that this emptiness-check inspects.*
- const [option\\_map](#) & [options](#) () const  
*Return the options parametrizing how the emptiness check is realized.*
- const char \* [parse\\_options](#) (char \*options)  
*Modify the algorithm options.*
- virtual bool [safe](#) () const  
*Return false iff accepting\_run() can return 0 for non-empty automata.*
- virtual const [unsigned\\_statistics](#) \* [statistics](#) () const  
*Return statistics, if available.*
- virtual void [options\\_updated](#) (const [option\\_map](#) &old)  
*Notify option updates.*
- void [set\\_states](#) (unsigned n)
- void [inc\\_states](#) ()
- void [inc\\_transitions](#) ()
- void [inc\\_depth](#) (unsigned n=1)
- void [dec\\_depth](#) (unsigned n=1)
- unsigned [states](#) () const
- unsigned [transitions](#) () const
- unsigned [max\\_depth](#) () const
- unsigned [depth](#) () const
- unsigned [get](#) (const char \*str) const

### Public Attributes

- [stats\\_map](#) stats

### Protected Member Functions

- void [remove\\_component](#) (const [state](#) \*start\_delete)  
*Remove a strongly component from the hash.*
- unsigned [get\\_removed\\_components](#) () const
- unsigned [get\\_vmsize](#) () const



### Protected Attributes

- `couvreur99_check_status` \* `ecs_`
- `bool` `poprem_`  
*Whether to store the `state` to be removed.*
- `unsigned` `removed_components`  
*Number of dead SCC removed by the algorithm.*
- `const tgba` \* `a_`  
*The automaton.*
- `option_map` `o_`  
*The options.*

#### 12.17.1 Detailed Description

An implementation of the Couvreur99 emptiness-check algorithm.

See the documentation for `spot::couvreur99`.

#### 12.17.2 Member Typedef Documentation

**12.17.2.1** `typedef unsigned(unsigned_statistics::*) spot::unsigned_statistics::unsigned_fun() const` `[inherited]`

**12.17.2.2** `typedef std::map<const char*, unsigned_fun, char_ptr_less_than> spot::unsigned_statistics::stats_map` `[inherited]`

#### 12.17.3 Constructor & Destructor Documentation

**12.17.3.1** `spot::couvreur99_check::couvreur99_check (const tgba * a, option_map o = option_map(), const numbered_state_heap_factory * nshf = numbered_state_heap_hash_map_factory::instance())`

**12.17.3.2** `virtual spot::couvreur99_check::~~couvreur99_check ()` `[virtual]`

#### 12.17.4 Member Function Documentation

**12.17.4.1** `virtual emptiness_check_result* spot::couvreur99_check::check ()` `[virtual]`

Check whether the automaton's language is empty.

Implements `spot::emptiness_check`.

Reimplemented in `spot::couvreur99_check_shy`.

**12.17.4.2** `virtual std::ostream& spot::couvreur99_check::print_stats (std::ostream & os) const` `[virtual]`

Print statistics, if any.

Reimplemented from [spot::emptiness\\_check](#).

**12.17.4.3** `const couvreur99_check_status* spot::couvreur99_check::result () const`

Return the status of the emptiness-check.

When [check\(\)](#) succeed, the status should be passed along to `spot::counter_example`.

This status should not be deleted, it is a pointer to a member of this class that will be deleted when the `couvreur99` object is deleted.

**12.17.4.4** `void spot::couvreur99_check::remove_component (const state * start_delete)` `[protected]`

Remove a strongly component from the hash.

This function remove all accessible [state](#) from a given [state](#). In other words, it removes the strongly connected component that contains this [state](#).

**12.17.4.5** `unsigned spot::couvreur99_check::get_removed_components () const` `[protected]`

**12.17.4.6** `unsigned spot::couvreur99_check::get_vmsize () const` `[protected]`

**12.17.4.7** `const tgba* spot::emptiness_check::automaton () const` `[inline, inherited]`

The automaton that this emptiness-check inspects.

**12.17.4.8** `const option_map& spot::emptiness_check::options () const` `[inline, inherited]`

Return the options parametrizing how the emptiness check is realized.

**12.17.4.9** `const char* spot::emptiness_check::parse_options (char * options)` `[inherited]`

Modify the algorithm options.

**12.17.4.10** `virtual bool spot::emptiness_check::safe () const` `[virtual, inherited]`

Return false iff `accepting_run()` can return 0 for non-empty automata.

**12.17.4.11** `virtual const unsigned_statistics* spot::emptiness_check::statistics () const` `[virtual, inherited]`

Return statistics, if available.

**12.17.4.12** `virtual void spot::emptiness_check::options_updated (const option_map & old)` [virtual, inherited]

Notify option updates.

**12.17.4.13** `void spot::ec_statistics::set_states (unsigned n)` [inline, inherited]

**12.17.4.14** `void spot::ec_statistics::inc_states ()` [inline, inherited]

**12.17.4.15** `void spot::ec_statistics::inc_transitions ()` [inline, inherited]

**12.17.4.16** `void spot::ec_statistics::inc_depth (unsigned n = 1)` [inline, inherited]

**12.17.4.17** `void spot::ec_statistics::dec_depth (unsigned n = 1)` [inline, inherited]

**12.17.4.18** `unsigned spot::ec_statistics::states () const` [inline, inherited]

**12.17.4.19** `unsigned spot::ec_statistics::transitions () const` [inline, inherited]

**12.17.4.20** `unsigned spot::ec_statistics::max_depth () const` [inline, inherited]

**12.17.4.21** `unsigned spot::ec_statistics::depth () const` [inline, inherited]

**12.17.4.22** `unsigned spot::unsigned_statistics::get (const char * str) const` [inline, inherited]

## 12.17.5 Member Data Documentation

**12.17.5.1** `couvreur99_check_status* spot::couvreur99_check::ecs_` [protected]

**12.17.5.2** `bool spot::couvreur99_check::poprem_` [protected]

Whether to store the [state](#) to be removed.

**12.17.5.3** `unsigned spot::couvreur99_check::removed_components` [protected]

Number of dead SCC removed by the algorithm.

**12.17.5.4** `const tgba* spot::emptiness_check::a_` [protected, inherited]

The automaton.

**12.17.5.5** `option_map spot::emptiness_check::o_` [protected, inherited]

The options.

### 12.17.5.6 stats\_map spot::unsigned\_statistics::stats [inherited]

The documentation for this class was generated from the following file:

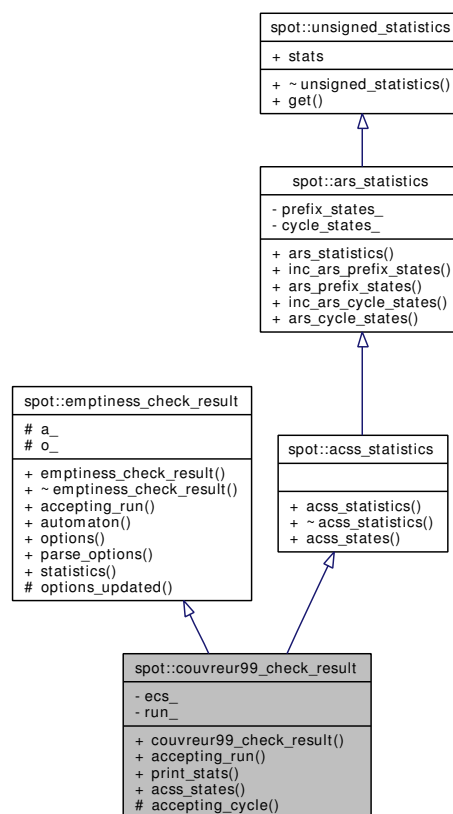
- [tgbaalgos/gtec/gtec.hh](#)

## 12.18 spot::couvreur99\_check\_result Class Reference

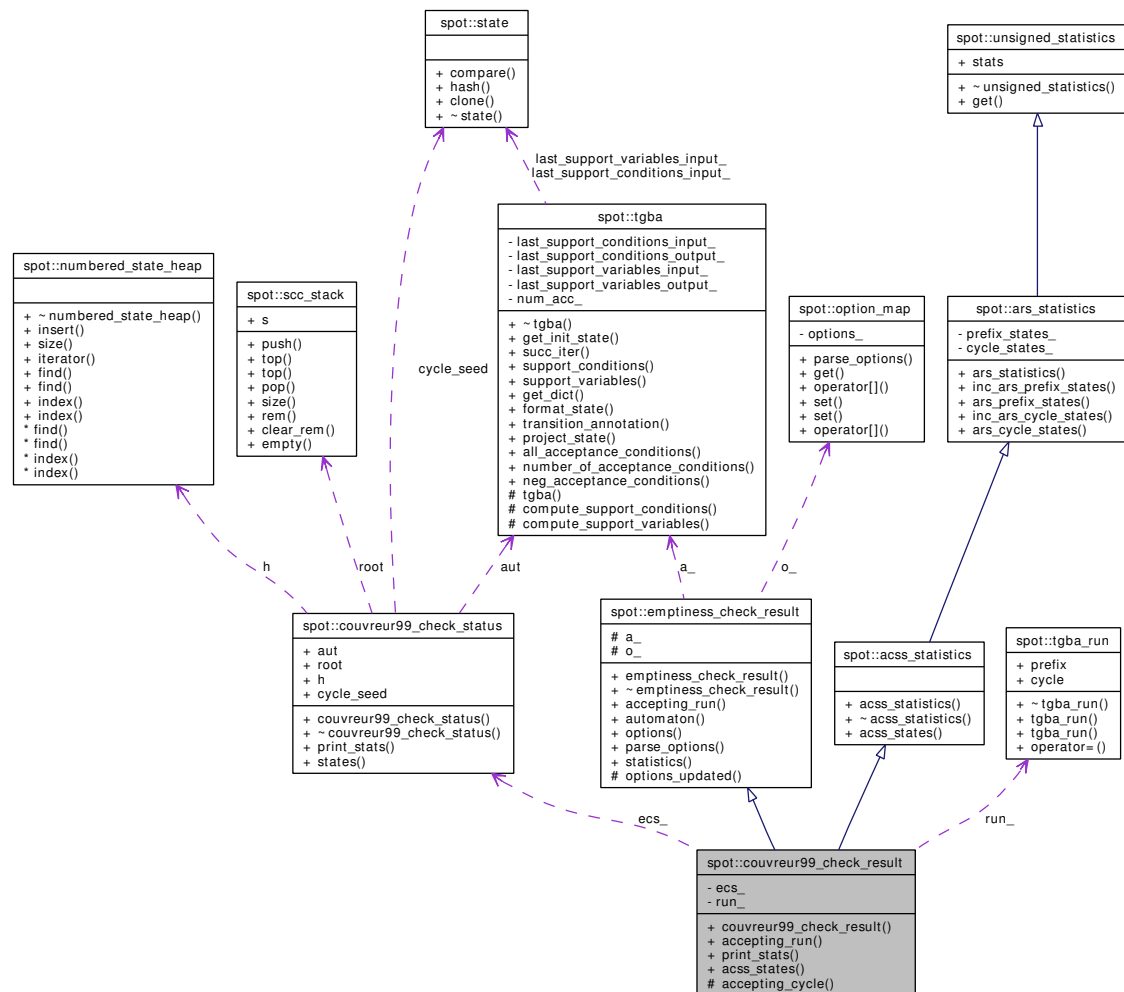
Compute a counter example from a [spot::couvreur99\\_check\\_status](#).

```
#include <tgbaalgos/gtec/ce.hh>
```

Inheritance diagram for spot::couvreur99\_check\_result:



spot::state



- typedef unsigned(unsigned\_statistics::\*) **unsigned\_fun** () const
- typedef std::map< const char \*, **unsigned\_fun**, char\_ptr\_less\_than > stats\_map

- `couvreur99_check_result` (const `couvreur99_check_status` \*ecs, `option_map` o=`option_map`())
- virtual `tgba_run` \* `accepting_run` ()

- void `print_stats` (std::ostream &os) const
- virtual unsigned `acss_states` () const  
*Number of states in the search space for the accepting cycle.*
- const `tgba * automaton` () const  
*The automaton on which an `accepting_run()` was found.*

- const [option\\_map](#) & [options](#) () const  
*Return the options parametrizing how the accepting run is computed.*
- const char \* [parse\\_options](#) (char \*options)  
*Modify the algorithm options.*
- virtual const [unsigned\\_statistics](#) \* [statistics](#) () const  
*Return statistics, if available.*
- void [inc\\_ars\\_prefix\\_states](#) ()
- unsigned [ars\\_prefix\\_states](#) () const
- void [inc\\_ars\\_cycle\\_states](#) ()
- unsigned [ars\\_cycle\\_states](#) () const
- unsigned [get](#) (const char \*str) const

### Public Attributes

- [stats\\_map](#) stats

### Protected Member Functions

- void [accepting\\_cycle](#) ()
- virtual void [options\\_updated](#) (const [option\\_map](#) &old)  
*Notify option updates.*

### Protected Attributes

- const [tgba](#) \* [a\\_](#)  
*The automaton.*
- [option\\_map](#) [o\\_](#)  
*The options.*

### Private Attributes

- const [couvreur99\\_check\\_status](#) \* [ecs\\_](#)
- [tgba\\_run](#) \* [run\\_](#)

## 12.18.1 Detailed Description

Compute a counter example from a [spot::couvreur99\\_check\\_status](#).

## 12.18.2 Member Typedef Documentation

**12.18.2.1** `typedef unsigned(unsigned_statistics::*) spot::unsigned_statistics::unsigned_fun() const`  
[inherited]

**12.18.2.2** `typedef std::map<const char*, unsigned_fun, char_ptr_less_than> spot::unsigned_statistics::stats_map` `[inherited]`

### 12.18.3 Constructor & Destructor Documentation

**12.18.3.1** `spot::couvreur99_check_result::couvreur99_check_result (const couvreur99_check_status * ecs, option_map o = option_map ())`

### 12.18.4 Member Function Documentation

**12.18.4.1** `virtual tgba_run* spot::couvreur99_check_result::accepting_run ()` `[virtual]`

Return a run accepted by the automata passed to the emptiness check.

This method might actually compute the acceptance run. (Not all emptiness check algorithms actually produce a counter-example as a side-effect of checking emptiness, some need some post-processing.)

This can also return 0 if the emptiness check algorithm cannot produce a counter example (that does not mean there is no counter-example; the mere existence of an instance of this class asserts the existence of a counter-example).

Reimplemented from [spot::emptiness\\_check\\_result](#).

**12.18.4.2** `void spot::couvreur99_check_result::print_stats (std::ostream & os) const`

**12.18.4.3** `virtual unsigned spot::couvreur99_check_result::acss_states () const` `[virtual]`

Number of states in the search space for the accepting cycle.

Implements [spot::acss\\_statistics](#).

**12.18.4.4** `void spot::couvreur99_check_result::accepting_cycle ()` `[protected]`

Called by [accepting\\_run\(\)](#) to find a cycle which traverses all acceptance conditions in the accepted SCC.

**12.18.4.5** `const tgba* spot::emptiness_check_result::automaton () const` `[inline, inherited]`

The automaton on which an [accepting\\_run\(\)](#) was found.

**12.18.4.6** `const option_map& spot::emptiness_check_result::options () const` `[inline, inherited]`

Return the options parametrizing how the accepting run is computed.

**12.18.4.7** `const char* spot::emptiness_check_result::parse_options (char * options)` `[inherited]`

Modify the algorithm options.

**12.18.4.8** `virtual const unsigned_statistics* spot::emptiness_check_result::statistics () const` `[virtual, inherited]`

Return statistics, if available.

**12.18.4.9** virtual void spot::emptiness\_check\_result::options\_updated (const option\_map & *old*)  
[protected, virtual, inherited]

Notify option updates.

**12.18.4.10** void spot::ars\_statistics::inc\_ars\_prefix\_states () [inline, inherited]

**12.18.4.11** unsigned spot::ars\_statistics::ars\_prefix\_states () const [inline, inherited]

**12.18.4.12** void spot::ars\_statistics::inc\_ars\_cycle\_states () [inline, inherited]

**12.18.4.13** unsigned spot::ars\_statistics::ars\_cycle\_states () const [inline, inherited]

**12.18.4.14** unsigned spot::unsigned\_statistics::get (const char \* *str*) const [inline, inherited]

## 12.18.5 Member Data Documentation

**12.18.5.1** const couvreur99\_check\_status\* spot::couvreur99\_check\_result::ecs\_ [private]

**12.18.5.2** tgba\_run\* spot::couvreur99\_check\_result::run\_ [private]

**12.18.5.3** const tgba\* spot::emptiness\_check\_result::a\_ [protected, inherited]

The automaton.

**12.18.5.4** option\_map spot::emptiness\_check\_result::o\_ [protected, inherited]

The options.

**12.18.5.5** stats\_map spot::unsigned\_statistics::stats [inherited]

The documentation for this class was generated from the following file:

- [tgbaalgos/gtec/ce.hh](#)

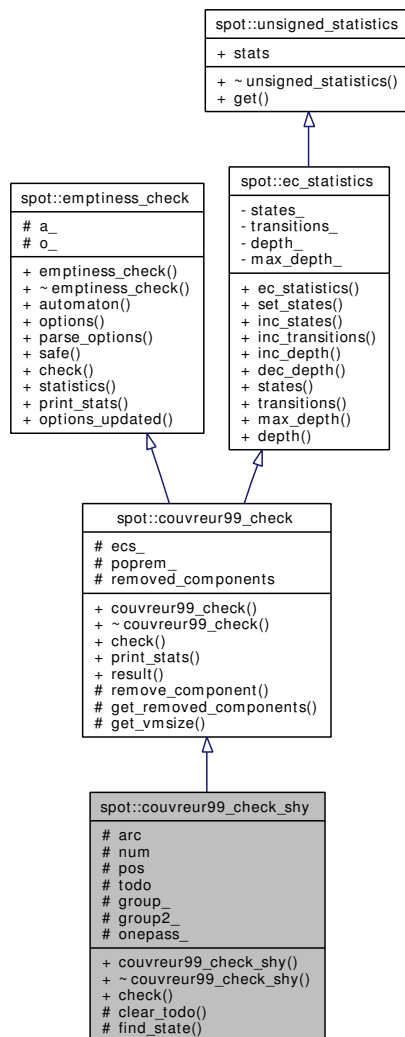
## 12.19 spot::couvreur99\_check\_shy Class Reference

A version of [spot::couvreur99\\_check](#) that tries to visit known states first.

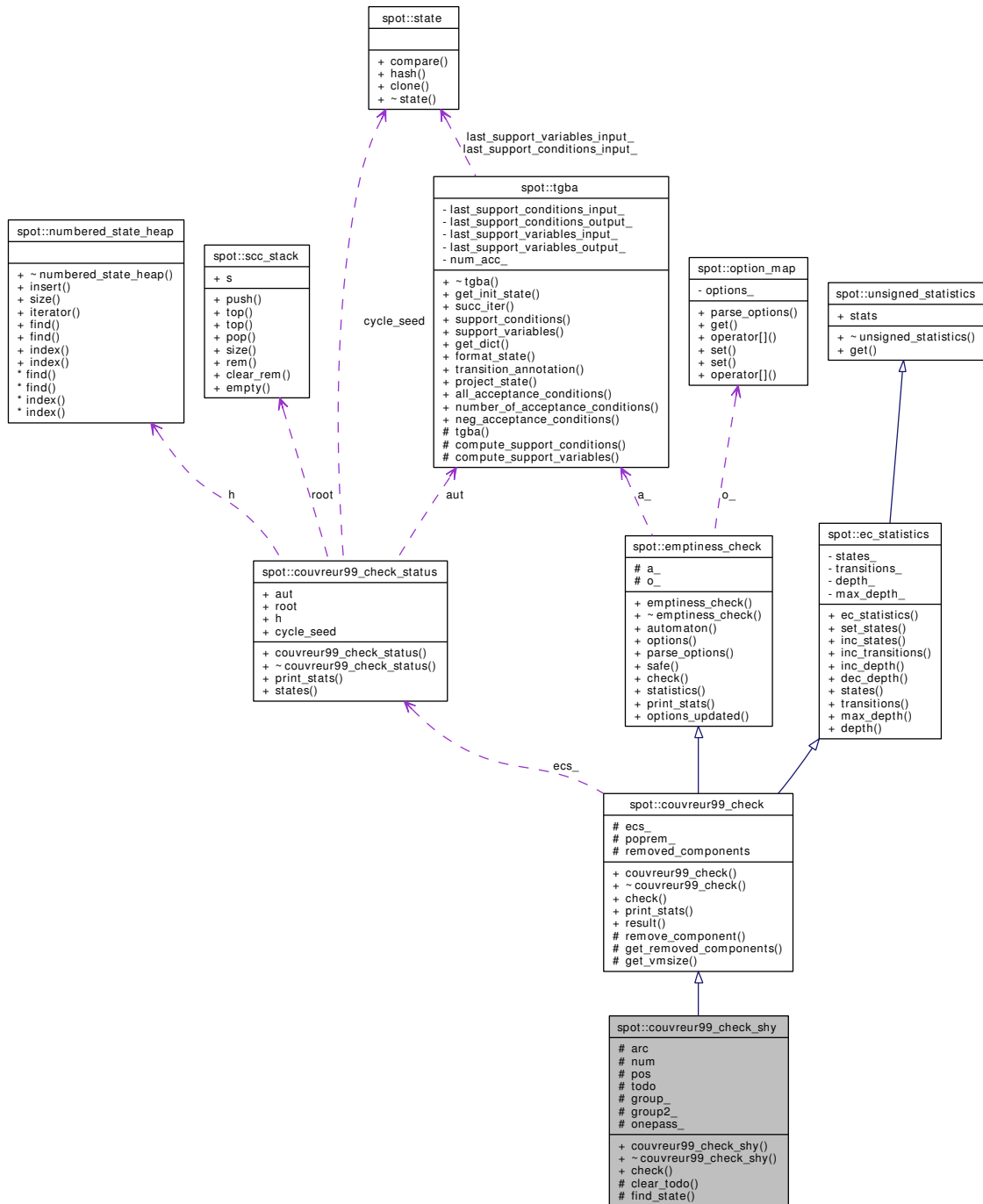
```
#include <tgbaalgos/gtec/gtec.hh>
```



Inheritance diagram for spot::couvreur99\_check\_shy:



Collaboration diagram for spot::couvreur99\_check\_shy:



## Public Types

- typedef unsigned(unsigned\_statistics::\*) [unsigned\\_fun](#) () const
- typedef std::map< const char \*, [unsigned\\_fun](#), [char\\_ptr\\_less\\_than](#) > [stats\\_map](#)

**Public Member Functions**

- `couvreur99_check_shy` (const `tgba` \*a, `option_map` o=`option_map`(), const `numbered_state_heap_factory` \*nshf=`numbered_state_heap_hash_map_factory::instance()`)
- virtual `~couvreur99_check_shy` ()
- virtual `emptiness_check_result` \* `check` ()  
*Check whether the automaton's language is empty.*
- virtual `std::ostream & print_stats` (`std::ostream &os`) const  
*Print statistics, if any.*
- const `couvreur99_check_status` \* `result` () const  
*Return the status of the emptiness-check.*
- const `tgba` \* `automaton` () const  
*The automaton that this emptiness-check inspects.*
- const `option_map` & `options` () const  
*Return the options parametrizing how the emptiness check is realized.*
- const char \* `parse_options` (char \*options)  
*Modify the algorithm options.*
- virtual bool `safe` () const  
*Return false iff accepting\_run() can return 0 for non-empty automata.*
- virtual const `unsigned_statistics` \* `statistics` () const  
*Return statistics, if available.*
- virtual void `options_updated` (const `option_map` &old)  
*Notify option updates.*
- void `set_states` (unsigned n)
- void `inc_states` ()
- void `inc_transitions` ()
- void `inc_depth` (unsigned n=1)
- void `dec_depth` (unsigned n=1)
- unsigned `states` () const
- unsigned `transitions` () const
- unsigned `max_depth` () const
- unsigned `depth` () const
- unsigned `get` (const char \*str) const

**Public Attributes**

- `stats_map` stats

**Protected Types**

- typedef `std::list`< `successor` > `succ_queue`
- typedef `std::list`< `todo_item` > `todo_list`

### Protected Member Functions

- void [clear\\_todo](#) ()
- virtual [numbered\\_state\\_heap::state\\_index\\_p find\\_state](#) (const [state](#) \*s)  
*find the SCC number of a unprocessed [state](#).*
- void [remove\\_component](#) (const [state](#) \*start\_delete)  
*Remove a strongly component from the hash.*
- unsigned [get\\_removed\\_components](#) () const
- unsigned [get\\_vmsize](#) () const

### Protected Attributes

- [std::stack< bdd > arc](#)
- int [num](#)
- [succ\\_queue::iterator pos](#)
- [todo\\_list todo](#)
- bool [group\\_](#)  
*Whether successors should be grouped for states in the same SCC.*
- bool [group2\\_](#)
- bool [onepass\\_](#)
- [couvreur99\\_check\\_status \\* ecs\\_](#)
- bool [poprem\\_](#)  
*Whether to store the [state](#) to be removed.*
- unsigned [removed\\_components](#)  
*Number of dead SCC removed by the algorithm.*
- const [tgba \\* a\\_](#)  
*The automaton.*
- [option\\_map o\\_](#)  
*The options.*

### Classes

- struct [successor](#)
- struct [todo\\_item](#)

#### 12.19.1 Detailed Description

A version of [spot::couvreur99\\_check](#) that tries to visit known states first.

See the documentation for [spot::couvreur99](#).

## 12.19.2 Member Typedef Documentation

**12.19.2.1** `typedef std::list<successor> spot::couvreur99_check_shy::succ_queue` [protected]

**12.19.2.2** `typedef std::list<todo_item> spot::couvreur99_check_shy::todo_list` [protected]

**12.19.2.3** `typedef unsigned(unsigned_statistics::*) spot::unsigned_statistics::unsigned_fun() const` [inherited]

**12.19.2.4** `typedef std::map<const char*, unsigned_fun, char_ptr_less_than> spot::unsigned_statistics::stats_map` [inherited]

## 12.19.3 Constructor & Destructor Documentation

**12.19.3.1** `spot::couvreur99_check_shy::couvreur99_check_shy (const tgba * a, option_map o = option_map(), const numbered_state_heap_factory * nshf = numbered_state_heap_hash_map_factory::instance())`

**12.19.3.2** `virtual spot::couvreur99_check_shy::~couvreur99_check_shy ()` [virtual]

## 12.19.4 Member Function Documentation

**12.19.4.1** `virtual emptiness_check_result* spot::couvreur99_check_shy::check ()` [virtual]

Check whether the automaton's language is empty.

Reimplemented from [spot::couvreur99\\_check](#).

**12.19.4.2** `void spot::couvreur99_check_shy::clear_todo ()` [protected]

**12.19.4.3** `virtual numbered_state_heap::state_index_p spot::couvreur99_check_shy::find_state (const state * s)` [protected, virtual]

find the SCC number of a unprocessed [state](#).

Sometimes we want to modify some of the above structures when looking up a new [state](#). This happens for instance when `find()` must perform inclusion checking and add new states to process to TODO during this step. (Because TODO must be known, sub-classing [spot::numbered\\_state\\_heap](#) is not enough.) Then overriding this method is the way to go.

**12.19.4.4** `virtual std::ostream& spot::couvreur99_check::print_stats (std::ostream & os) const` [virtual, inherited]

Print statistics, if any.

Reimplemented from [spot::emptiness\\_check](#).

**12.19.4.5** `const covreur99_check_status* spot::couvreur99_check::result () const` [inherited]

Return the status of the emptiness-check.

When `check()` succeed, the status should be passed along to `spot::counter_example`.

This status should not be deleted, it is a pointer to a member of this class that will be deleted when the `couvreur99` object is deleted.

**12.19.4.6** `void spot::couvreur99_check::remove_component (const state * start_delete)` [protected, inherited]

Remove a strongly component from the hash.

This function remove all accessible `state` from a given `state`. In other words, it removes the strongly connected component that contains this `state`.

**12.19.4.7** `unsigned spot::couvreur99_check::get_removed_components () const` [protected, inherited]

**12.19.4.8** `unsigned spot::couvreur99_check::get_vmsize () const` [protected, inherited]

**12.19.4.9** `const tgba* spot::emptiness_check::automaton () const` [inline, inherited]

The automaton that this emptiness-check inspects.

**12.19.4.10** `const option_map& spot::emptiness_check::options () const` [inline, inherited]

Return the options parametrizing how the emptiness check is realized.

**12.19.4.11** `const char* spot::emptiness_check::parse_options (char * options)` [inherited]

Modify the algorithm options.

**12.19.4.12** `virtual bool spot::emptiness_check::safe () const` [virtual, inherited]

Return false iff accepting\_run() can return 0 for non-empty automata.

**12.19.4.13** `virtual const unsigned_statistics* spot::emptiness_check::statistics () const` [virtual, inherited]

Return statistics, if available.

**12.19.4.14** `virtual void spot::emptiness_check::options_updated (const option_map & old)` [virtual, inherited]

Notify option updates.

**12.19.4.15** `void spot::ec_statistics::set_states (unsigned n)` [inline, inherited]

- 12.19.4.16** void spot::ec\_statistics::inc\_states () [inline, inherited]
- 12.19.4.17** void spot::ec\_statistics::inc\_transitions () [inline, inherited]
- 12.19.4.18** void spot::ec\_statistics::inc\_depth (unsigned *n* = 1) [inline, inherited]
- 12.19.4.19** void spot::ec\_statistics::dec\_depth (unsigned *n* = 1) [inline, inherited]
- 12.19.4.20** unsigned spot::ec\_statistics::states () const [inline, inherited]
- 12.19.4.21** unsigned spot::ec\_statistics::transitions () const [inline, inherited]
- 12.19.4.22** unsigned spot::ec\_statistics::max\_depth () const [inline, inherited]
- 12.19.4.23** unsigned spot::ec\_statistics::depth () const [inline, inherited]
- 12.19.4.24** unsigned spot::unsigned\_statistics::get (const char \* *str*) const [inline, inherited]
- 12.19.5 Member Data Documentation**
- 12.19.5.1** std::stack<bdd> spot::couvreur99\_check\_shy::arc [protected]
- 12.19.5.2** int spot::couvreur99\_check\_shy::num [protected]
- 12.19.5.3** succ\_queue::iterator spot::couvreur99\_check\_shy::pos [protected]
- 12.19.5.4** todo\_list spot::couvreur99\_check\_shy::todo [protected]
- 12.19.5.5** bool spot::couvreur99\_check\_shy::group\_ [protected]  
Whether successors should be grouped for states in the same SCC.
- 12.19.5.6** bool spot::couvreur99\_check\_shy::group2\_ [protected]
- 12.19.5.7** bool spot::couvreur99\_check\_shy::onepass\_ [protected]
- 12.19.5.8** couvreur99\_check\_status\* spot::couvreur99\_check::ecs\_ [protected, inherited]
- 12.19.5.9** bool spot::couvreur99\_check::poprem\_ [protected, inherited]  
Whether to store the [state](#) to be removed.

**12.19.5.10** unsigned spot::couvreur99\_check::removed\_components [protected, inherited]

Number of dead SCC removed by the algorithm.

**12.19.5.11** const tgba\* spot::emptiness\_check::a\_ [protected, inherited]

The automaton.

**12.19.5.12** option\_map spot::emptiness\_check::o\_ [protected, inherited]

The options.

**12.19.5.13** stats\_map spot::unsigned\_statistics::stats [inherited]

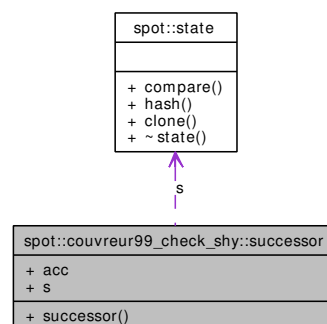
The documentation for this class was generated from the following file:

- [tgbaalgos/gtec/gtec.hh](#)

## 12.20 spot::couvreur99\_check\_shy::successor Struct Reference

```
#include <tgbaalgos/gtec/gtec.hh>
```

Collaboration diagram for spot::couvreur99\_check\_shy::successor:



### Public Member Functions

- [successor](#) (bdd [acc](#), const [spot::state](#) \*[s](#))

### Public Attributes

- bdd [acc](#)
- const [spot::state](#) \* [s](#)

## 12.20.1 Constructor & Destructor Documentation

**12.20.1.1** spot::couvreur99\_check\_shy::successor::successor (bdd *acc*, const spot::state \* *s*)  
[inline]



## 12.20.2 Member Data Documentation

### 12.20.2.1 bdd spot::couvreur99\_check\_shy::successor::acc

### 12.20.2.2 const spot::state\* spot::couvreur99\_check\_shy::successor::s

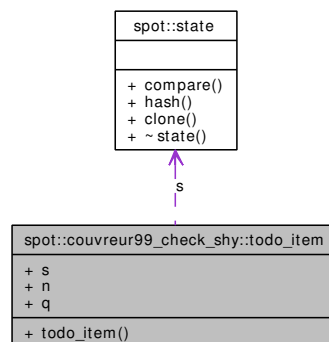
The documentation for this struct was generated from the following file:

- [tgbaalgos/gtec/gtec.hh](#)

## 12.21 spot::couvreur99\_check\_shy::todo\_item Struct Reference

```
#include <tgbaalgos/gtec/gtec.hh>
```

Collaboration diagram for spot::couvreur99\_check\_shy::todo\_item:



## Public Member Functions

- [todo\\_item](#) (const [state](#) \*[s](#), int [n](#), [couvreur99\\_check\\_shy](#) \*[shy](#))

## Public Attributes

- const [state](#) \* [s](#)
- int [n](#)
- [succ\\_queue](#) [q](#)

## 12.21.1 Constructor & Destructor Documentation

### 12.21.1.1 spot::couvreur99\_check\_shy::todo\_item::todo\_item (const state \* *s*, int *n*, couvreur99\_check\_shy \* *shy*)

## 12.21.2 Member Data Documentation

### 12.21.2.1 const state\* spot::couvreur99\_check\_shy::todo\_item::s

### 12.21.2.2 int spot::couvreur99\_check\_shy::todo\_item::n

## 12.21.2.3 succ\_queue spot::couvreur99\_check\_shy::todo\_item::q

The documentation for this struct was generated from the following file:

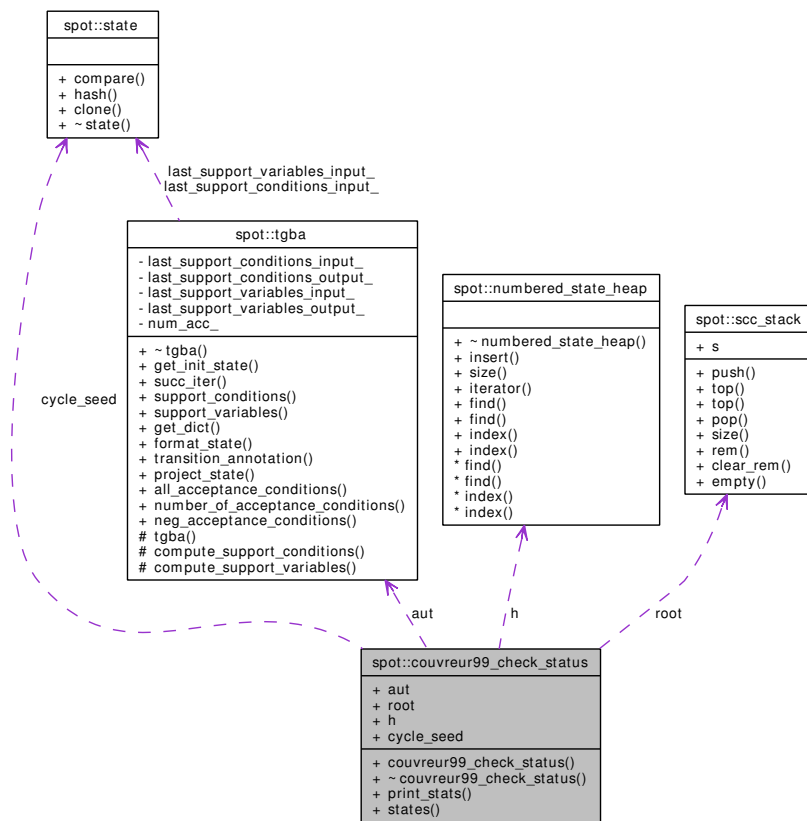
- [tgbaalgos/gtec/gtec.hh](#)

## 12.22 spot::couvreur99\_check\_status Class Reference

The status of the emptiness-check on success.

```
#include <tgbaalgos/gtec/status.hh>
```

Collaboration diagram for spot::couvreur99\_check\_status:



## Public Member Functions

- [couvreur99\\_check\\_status](#) (const [tgba](#) \*aut, const [numbered\\_state\\_heap\\_factory](#) \*nshf)
- [~couvreur99\\_check\\_status](#) ()
- void [print\\_stats](#) (std::ostream &os) const  
*Output statistics about this object.*
- int [states](#) () const  
*Return the number of states visited by the search.*

### Public Attributes

- const [tgba](#) \* [aut](#)
- [scc\\_stack](#) root
- [numbered\\_state\\_heap](#) \* [h](#)  
*Heap of visited states.*
- const [state](#) \* [cycle\\_seed](#)

#### 12.22.1 Detailed Description

The status of the emptiness-check on success.

This contains everything needed to construct a counter-example: the automata, the stack of SCCs traversed by the counter-example, and the heap of visited states with their indexes.

#### 12.22.2 Constructor & Destructor Documentation

**12.22.2.1** `spot::couvreur99_check_status::couvreur99_check_status (const tgba * aut, const numbered\_state\_heap\_factory * nshf)`

**12.22.2.2** `spot::couvreur99_check_status::~~couvreur99_check_status ()`

#### 12.22.3 Member Function Documentation

**12.22.3.1** `void spot::couvreur99_check_status::print_stats (std::ostream & os) const`

Output statistics about this object.

**12.22.3.2** `int spot::couvreur99_check_status::states () const`

Return the number of states visited by the search.

#### 12.22.4 Member Data Documentation

**12.22.4.1** `const tgba* spot::couvreur99_check_status::aut`

**12.22.4.2** `scc\_stack spot::couvreur99_check_status::root`

**12.22.4.3** `numbered\_state\_heap* spot::couvreur99_check_status::h`

Heap of visited states.

**12.22.4.4** `const state* spot::couvreur99_check_status::cycle_seed`

The documentation for this class was generated from the following file:

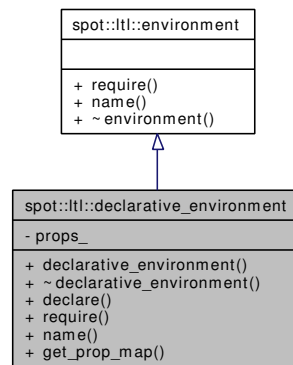
- [tgbaalgos/gtec/status.hh](#)

## 12.23 spot::ltl::declarative\_environment Class Reference

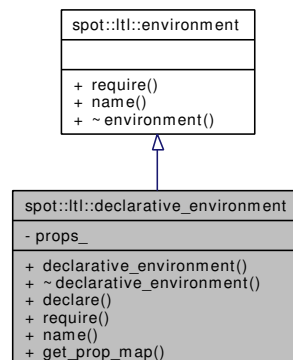
A declarative [environment](#).

```
#include <ltlenv/declenv.hh>
```

Inheritance diagram for spot::ltl::declarative\_environment:



Collaboration diagram for spot::ltl::declarative\_environment:



### Public Types

- typedef `std::map< const std::string, ltl::atomic\_prop * >` `prop_map`

### Public Member Functions

- `declarative_environment()`
- `~declarative_environment()`
- `bool declare(const std::string &prop_str)`
- `virtual ltl::formula * require(const std::string &prop_str)`  
*Obtain the [formula](#) associated to prop\_str.*
- `virtual const std::string & name()`  
*Get the name of the [environment](#).*
- `const prop\_map & get_prop_map() const`

*Get the map of atomic proposition known to this [environment](#).*

### Private Attributes

- `prop_map props_`

### 12.23.1 Detailed Description

A declarative [environment](#).

This [environment](#) recognizes all atomic propositions that have been previously declared. It will reject other.

### 12.23.2 Member Typedef Documentation

**12.23.2.1** `typedef std::map<const std::string, ltl::atomic_prop*> spot::ltl::declarative_environment::prop_map`

### 12.23.3 Constructor & Destructor Documentation

**12.23.3.1** `spot::ltl::declarative_environment::declarative_environment ()`

**12.23.3.2** `spot::ltl::declarative_environment::~~declarative_environment ()`

### 12.23.4 Member Function Documentation

**12.23.4.1** `bool spot::ltl::declarative_environment::declare (const std::string & prop_str)`

Declare an atomic proposition. Return false iff the proposition was already declared.

**12.23.4.2** `virtual ltl::formula* spot::ltl::declarative_environment::require (const std::string & prop_str) [virtual]`

Obtain the [formula](#) associated to `prop_str`.

Usually `prop_str`, is the name of an atomic proposition, and `spot::ltl::require` simply returns the associated [spot::ltl::atomic\\_prop](#).

Note this is not a `const` method. Some environments will "create" the atomic proposition when requested.

We return a [spot::ltl::formula](#) instead of an [spot::ltl::atomic\\_prop](#), because this will allow nifty tricks (e.g., we could name formulae in an [environment](#), and let the parser build a larger tree from these).

#### Returns:

0 iff `prop_str` is not part of the [environment](#), or the associated [spot::ltl::formula](#) otherwise.

Implements [spot::ltl::environment](#).

**12.23.4.3** `virtual const std::string& spot::ltl::declarative_environment::name () [virtual]`

Get the name of the [environment](#).

Implements [spot::ltl::environment](#).

#### 12.23.4.4 const prop\_map& spot::ltl::declarative\_environment::get\_prop\_map () const

Get the map of atomic proposition known to this [environment](#).

### 12.23.5 Member Data Documentation

#### 12.23.5.1 prop\_map spot::ltl::declarative\_environment::props\_ [private]

The documentation for this class was generated from the following file:

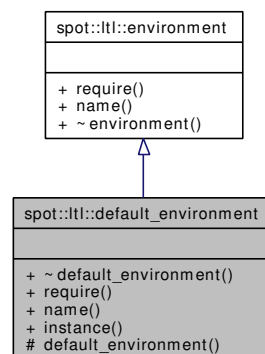
- ltlenv/[declenv.hh](#)

## 12.24 spot::ltl::default\_environment Class Reference

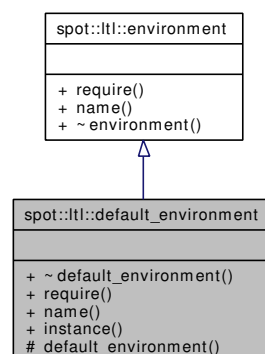
A laxist [environment](#).

```
#include <ltlenv/defaultenv.hh>
```

Inheritance diagram for spot::ltl::default\_environment:



Collaboration diagram for spot::ltl::default\_environment:



### Public Member Functions

- virtual `~default_environment ()`
- virtual `formula * require (const std::string &prop_str)`

*Obtain the [formula](#) associated to prop\_str.*

- virtual const std::string & `name` ()  
Get the name of the `environment`.

### Static Public Member Functions

- static `default_environment` & `instance` ()  
Get the sole instance of `spot::ltl::default_environment`.

### Protected Member Functions

- `default_environment` ()

#### 12.24.1 Detailed Description

A laxist `environment`.

This `environment` recognizes all atomic propositions.

This is a singleton. Use `default_environment::instance()` to obtain the instance.

#### 12.24.2 Constructor & Destructor Documentation

**12.24.2.1** virtual `spot::ltl::default_environment::~~default_environment` () [virtual]

**12.24.2.2** `spot::ltl::default_environment::default_environment` () [protected]

#### 12.24.3 Member Function Documentation

**12.24.3.1** virtual `formula*` `spot::ltl::default_environment::require` (const std::string & `prop_str`) [virtual]

Obtain the `formula` associated to `prop_str`.

Usually `prop_str`, is the name of an atomic proposition, and `spot::ltl::require` simply returns the associated `spot::ltl::atomic_prop`.

Note this is not a `const` method. Some environments will "create" the atomic proposition when requested.

We return a `spot::ltl::formula` instead of an `spot::ltl::atomic_prop`, because this will allow nifty tricks (e.g., we could name formulae in an `environment`, and let the parser build a larger tree from these).

#### Returns:

0 iff `prop_str` is not part of the `environment`, or the associated `spot::ltl::formula` otherwise.

Implements `spot::ltl::environment`.

**12.24.3.2 virtual const std::string& spot::ltl::default\_environment::name () [virtual]**

Get the name of the [environment](#).

Implements [spot::ltl::environment](#).

**12.24.3.3 static default\_environment& spot::ltl::default\_environment::instance () [static]**

Get the sole instance of [spot::ltl::default\\_environment](#).

The documentation for this class was generated from the following file:

- [ltlenv/defaultenv.hh](#)

**12.25 spot::delayed\_simulation\_relation Class Reference**

```
#include <tgba/tgbareduc.hh>
```

The documentation for this class was generated from the following file:

- [tgba/tgbareduc.hh](#)

**12.26 spot::direct\_simulation\_relation Class Reference**

```
#include <tgba/tgbareduc.hh>
```

The documentation for this class was generated from the following file:

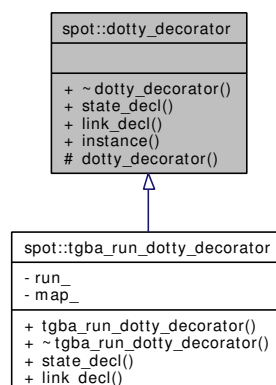
- [tgba/tgbareduc.hh](#)

**12.27 spot::dotty\_decorator Class Reference**

Choose [state](#) and link styles for [spot::dotty\\_reachable](#).

```
#include <tgbaalgos/dottydec.hh>
```

Inheritance diagram for [spot::dotty\\_decorator](#):





### Public Member Functions

- virtual `~dotty_decorator()`
- virtual `std::string state_decl (const tgba *a, const state *s, int n, tgba_succ_iterator *si, const std::string &label)`  
*Compute the style of a `state`.*
- virtual `std::string link_decl (const tgba *a, const state *in_s, int in, const state *out_s, int out, const tgba_succ_iterator *si, const std::string &label)`  
*Compute the style of a link.*

### Static Public Member Functions

- static `dotty_decorator * instance ()`  
*Get the unique instance of the default `dotty_decorator`.*

### Protected Member Functions

- `dotty_decorator()`

#### 12.27.1 Detailed Description

Choose `state` and link styles for `spot::dotty_reachable`.

#### 12.27.2 Constructor & Destructor Documentation

**12.27.2.1** virtual `spot::dotty_decorator::~~dotty_decorator()` [virtual]

**12.27.2.2** `spot::dotty_decorator::dotty_decorator()` [protected]

#### 12.27.3 Member Function Documentation

**12.27.3.1** virtual `std::string spot::dotty_decorator::state_decl (const tgba *a, const state *s, int n, tgba_succ_iterator *si, const std::string &label)` [virtual]

Compute the style of a `state`.

This function should output a string of the form `[label="foo", style=bar, ...]`. The default implementation will simply output `[label="LABEL"]` with LABEL replaced by the value of `label`.

##### Parameters:

- a* the automaton being drawn
- s* the `state` being drawn (owned by the caller)
- n* a unique number for this `state`
- si* an iterator over the successors of this `state` (owned by the caller, but can be freely iterated)
- label* the computed name of this `state`

Reimplemented in `spot::tgba_run_dotty_decorator`.

**12.27.3.2** `virtual std::string spot::dotty_decorator::link_decl (const tgba * a, const state * in_s, int in, const state * out_s, int out, const tgba_succ_iterator * si, const std::string & label)` [virtual]

Compute the style of a link.

This function should output a string of the form [`label="foo", style=bar, ...`]. The default implementation will simply output [`label="LABEL"`] with `LABEL` replaced by the value of *label*.

**Parameters:**

- a* the automaton being drawn
- in\_s* the source [state](#) of the transition being drawn (owned by the caller)
- in* the unique number associated to *in\_s*
- out\_s* the destination [state](#) of the transition being drawn (owned by the caller)
- out* the unique number associated to *out\_s*
- si* an iterator over the successors of *in\_s*, pointing to the current transition (owned by the caller and cannot be iterated)
- label* the computed name of this [state](#)

Reimplemented in [spot::tgba\\_run\\_dotty\\_decorator](#).

**12.27.3.3** `static dotty_decorator* spot::dotty_decorator::instance ()` [static]

Get the unique instance of the default [dotty\\_decorator](#).

The documentation for this class was generated from the following file:

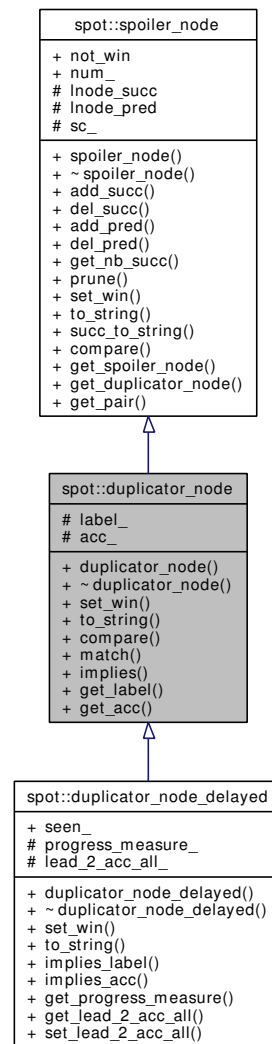
- [tgbaalgos/dottydec.hh](#)

## 12.28 spot::duplicator\_node Class Reference

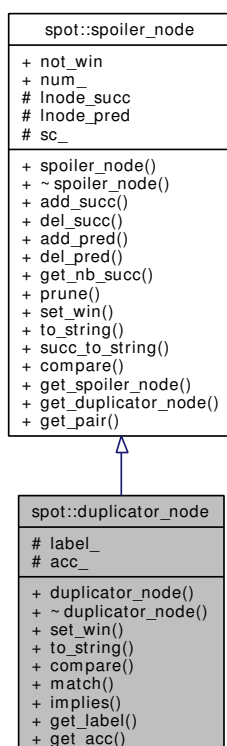
Duplicator node of parity game graph.

```
#include <tgbaalgos/reductgba_sim.hh>
```

Inheritance diagram for spot::duplicator\_node:



Collaboration diagram for spot::duplicator\_node:



## Public Member Functions

- `duplicator_node` (const `state` \*d\_node, const `state` \*s\_node, bdd l, bdd a, int num)
- virtual `~duplicator_node` ()
- virtual bool `set_win` ()
- virtual std::string `to_string` (const `tgba` \*a)
- virtual bool `compare` (`spoiler_node` \*n)
- bool `match` (bdd l, bdd a)
- bool `implies` (bdd l, bdd a)
- bdd `get_label` () const
- bdd `get_acc` () const
- bool `add_succ` (`spoiler_node` \*n)

*Add a successor. Return true if n wasn't yet in the list of successor, false otherwise.*

- void `del_succ` (`spoiler_node` \*n)
- virtual void `add_pred` (`spoiler_node` \*n)
- virtual void `del_pred` ()
- int `get_nb_succ` ()
- bool `prune` ()
- virtual std::string `succ_to_string` ()
- const `state` \* `get_spoiler_node` ()
- const `state` \* `get_duplicator_node` ()
- `state_couple` \* `get_pair` ()

### Public Attributes

- bool [not\\_win](#)
- int [num\\_](#)

### Protected Attributes

- bdd [label\\_](#)
- bdd [acc\\_](#)
- [sn\\_v](#) \* [lnode\\_succ](#)
- [sn\\_v](#) \* [lnode\\_pred](#)
- [state\\_couple](#) \* [sc\\_](#)

### 12.28.1 Detailed Description

Duplicator node of parity game graph.

### 12.28.2 Constructor & Destructor Documentation

**12.28.2.1** `spot::duplicator_node::duplicator_node (const state * d_node, const state * s_node, bdd l, bdd a, int num)`

**12.28.2.2** `virtual spot::duplicator_node::~~duplicator_node ()` [virtual]

### 12.28.3 Member Function Documentation

**12.28.3.1** `virtual bool spot::duplicator_node::set_win ()` [virtual]

Reimplemented from [spot::spoiler\\_node](#).

Reimplemented in [spot::duplicator\\_node\\_delayed](#).

**12.28.3.2** `virtual std::string spot::duplicator_node::to_string (const tgba * a)` [virtual]

Reimplemented from [spot::spoiler\\_node](#).

Reimplemented in [spot::duplicator\\_node\\_delayed](#).

**12.28.3.3** `virtual bool spot::duplicator_node::compare (spoiler_node * n)` [virtual]

Reimplemented from [spot::spoiler\\_node](#).

**12.28.3.4** `bool spot::duplicator_node::match (bdd l, bdd a)`

**12.28.3.5** `bool spot::duplicator_node::implies (bdd l, bdd a)`

**12.28.3.6** `bdd spot::duplicator_node::get_label () const`

**12.28.3.7** `bdd spot::duplicator_node::get_acc () const`

**12.28.3.8** `bool spot::spoiler_node::add_succ (spoiler_node * n)` [inherited]

Add a successor. Return true if *n* wasn't yet in the list of successor, false otherwise.

**12.28.3.9** `void spot::spoiler_node::del_succ (spoiler_node * n)` [inherited]

**12.28.3.10** `virtual void spot::spoiler_node::add_pred (spoiler_node * n)` [virtual, inherited]

**12.28.3.11** `virtual void spot::spoiler_node::del_pred ()` [virtual, inherited]

**12.28.3.12** `int spot::spoiler_node::get_nb_succ ()` [inherited]

**12.28.3.13** `bool spot::spoiler_node::prune ()` [inherited]

**12.28.3.14** `virtual std::string spot::spoiler_node::succ_to_string ()` [virtual, inherited]

**12.28.3.15** `const state* spot::spoiler_node::get_spoiler_node ()` [inherited]

**12.28.3.16** `const state* spot::spoiler_node::get_duplicator_node ()` [inherited]

**12.28.3.17** `state_couple* spot::spoiler_node::get_pair ()` [inherited]

## 12.28.4 Member Data Documentation

**12.28.4.1** `bdd spot::duplicator_node::label_` [protected]

**12.28.4.2** `bdd spot::duplicator_node::acc_` [protected]

**12.28.4.3** `bool spot::spoiler_node::not_win` [inherited]

**12.28.4.4** `int spot::spoiler_node::num_` [inherited]

**12.28.4.5** `sn_v* spot::spoiler_node::lnode_succ` [protected, inherited]

**12.28.4.6** `sn_v* spot::spoiler_node::lnode_pred` [protected, inherited]

**12.28.4.7** `state_couple* spot::spoiler_node::sc_` [protected, inherited]

The documentation for this class was generated from the following file:

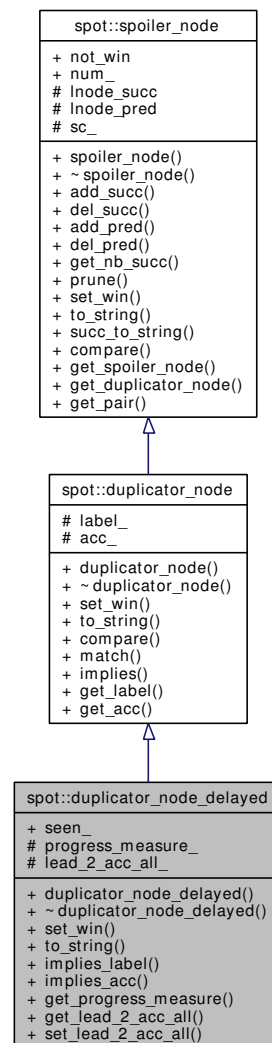
- [tgbalgorithms/reductgba\\_sim.hh](#)

## 12.29 spot::duplicator\_node\_delayed Class Reference

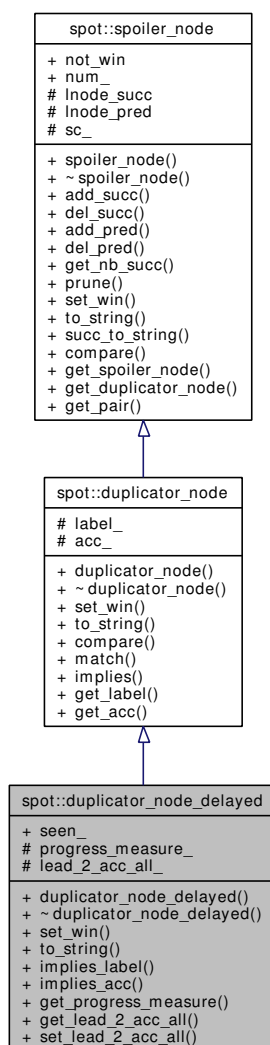
Duplicator node of parity game graph for delayed simulation.

```
#include <tgbaaalgos/reductgba_sim.hh>
```

Inheritance diagram for spot::duplicator\_node\_delayed:



Collaboration diagram for spot::duplicator\_node\_delayed:



## Public Member Functions

- `duplicator_node_delayed` (const `state` \*d\_node, const `state` \*s\_node, bdd l, bdd a, int num)
- `~duplicator_node_delayed` ()
- `bool set_win` ()  
*Return true if the progress\_measure has changed.*
- `virtual std::string to_string` (const `tgba` \*a)
- `bool implies_label` (bdd l)
- `bool implies_acc` (bdd a)
- `int get_progress_measure` ()
- `bool get_lead_2_acc_all` ()
- `bool set_lead_2_acc_all` (bdd acc=bddfalse)
- `virtual bool compare` (`spoiler_node` \*n)
- `bool match` (bdd l, bdd a)



- bool [implies](#) (bdd l, bdd a)
- bdd [get\\_label](#) () const
- bdd [get\\_acc](#) () const
- bool [add\\_succ](#) (spoiler\_node \*n)

*Add a successor. Return true if n wasn't yet in the list of successor, false otherwise.*

- void [del\\_succ](#) (spoiler\_node \*n)
- virtual void [add\\_pred](#) (spoiler\_node \*n)
- virtual void [del\\_pred](#) ()
- int [get\\_nb\\_succ](#) ()
- bool [prune](#) ()
- virtual std::string [succ\\_to\\_string](#) ()
- const state \* [get\\_spoiler\\_node](#) ()
- const state \* [get\\_duplicator\\_node](#) ()
- state\_couple \* [get\\_pair](#) ()

### Public Attributes

- bool [seen\\_](#)
- bool [not\\_win](#)
- int [num\\_](#)

### Protected Attributes

- int [progress\\_measure\\_](#)
- bool [lead\\_2\\_acc\\_all\\_](#)
- bdd [label\\_](#)
- bdd [acc\\_](#)
- sn\_v \* [lnode\\_succ](#)
- sn\_v \* [lnode\\_pred](#)
- state\_couple \* [sc\\_](#)

#### 12.29.1 Detailed Description

Duplicator node of parity game graph for delayed simulation.

#### 12.29.2 Constructor & Destructor Documentation

**12.29.2.1** spot::duplicator\_node\_delayed::duplicator\_node\_delayed (const state \* *d\_node*, const state \* *s\_node*, bdd *l*, bdd *a*, int *num*)

**12.29.2.2** spot::duplicator\_node\_delayed::~~duplicator\_node\_delayed ()

#### 12.29.3 Member Function Documentation

**12.29.3.1** bool spot::duplicator\_node\_delayed::set\_win () [virtual]

Return true if the progress\_measure has changed.

Reimplemented from [spot::duplicator\\_node](#).

**12.29.3.2** `virtual std::string spot::duplicator_node_delayed::to_string (const tgba * a)` [virtual]

Reimplemented from [spot::duplicator\\_node](#).

**12.29.3.3** `bool spot::duplicator_node_delayed::implies_label (bdd l)`

**12.29.3.4** `bool spot::duplicator_node_delayed::implies_acc (bdd a)`

**12.29.3.5** `int spot::duplicator_node_delayed::get_progress_measure ()`

**12.29.3.6** `bool spot::duplicator_node_delayed::get_lead_2_acc_all ()`

**12.29.3.7** `bool spot::duplicator_node_delayed::set_lead_2_acc_all (bdd acc = bddfalse)`

**12.29.3.8** `virtual bool spot::duplicator_node::compare (spoiler_node * n)` [virtual, inherited]

Reimplemented from [spot::spoiler\\_node](#).

**12.29.3.9** `bool spot::duplicator_node::match (bdd l, bdd a)` [inherited]

**12.29.3.10** `bool spot::duplicator_node::implies (bdd l, bdd a)` [inherited]

**12.29.3.11** `bdd spot::duplicator_node::get_label () const` [inherited]

**12.29.3.12** `bdd spot::duplicator_node::get_acc () const` [inherited]

**12.29.3.13** `bool spot::spoiler_node::add_succ (spoiler_node * n)` [inherited]

Add a successor. Return true if *n* wasn't yet in the list of successor, false otherwise.

**12.29.3.14** `void spot::spoiler_node::del_succ (spoiler_node * n)` [inherited]

**12.29.3.15** `virtual void spot::spoiler_node::add_pred (spoiler_node * n)` [virtual, inherited]

**12.29.3.16** `virtual void spot::spoiler_node::del_pred ()` [virtual, inherited]

**12.29.3.17** `int spot::spoiler_node::get_nb_succ ()` [inherited]

**12.29.3.18** `bool spot::spoiler_node::prune ()` [inherited]

**12.29.3.19** `virtual std::string spot::spoiler_node::succ_to_string ()` [virtual, inherited]

**12.29.3.20** `const state* spot::spoiler_node::get_spoiler_node ()` [inherited]

**12.29.3.21** `const state* spot::spoiler_node::get_duplicator_node ()` [inherited]

**12.29.3.22** `state_couple* spot::spoiler_node::get_pair ()` [inherited]

## 12.29.4 Member Data Documentation

**12.29.4.1** `bool spot::duplicator_node_delayed::seen_`

**12.29.4.2** `int spot::duplicator_node_delayed::progress_measure_` [protected]

**12.29.4.3** `bool spot::duplicator_node_delayed::lead_2_acc_all_` [protected]

**12.29.4.4** `bdd spot::duplicator_node::label_` [protected, inherited]

**12.29.4.5** `bdd spot::duplicator_node::acc_` [protected, inherited]

**12.29.4.6** `bool spot::spoiler_node::not_win` [inherited]

**12.29.4.7** `int spot::spoiler_node::num_` [inherited]

**12.29.4.8** `sn_v* spot::spoiler_node::lnode_succ` [protected, inherited]

**12.29.4.9** `sn_v* spot::spoiler_node::lnode_pred` [protected, inherited]

**12.29.4.10** `state_couple* spot::spoiler_node::sc_` [protected, inherited]

The documentation for this class was generated from the following file:

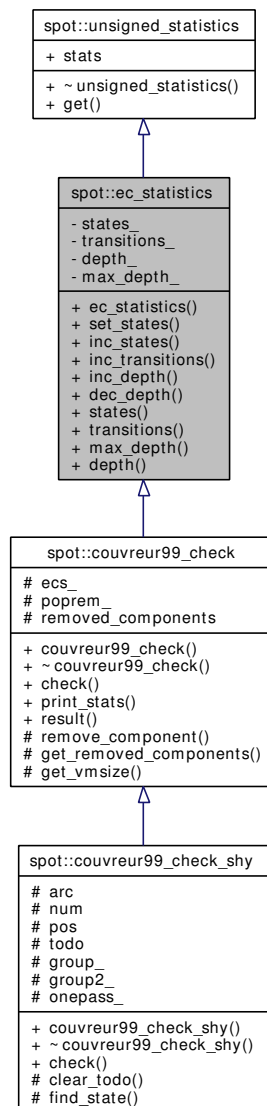
- [tgbaalgos/reductgba\\_sim.hh](#)

## 12.30 spot::ec\_statistics Class Reference

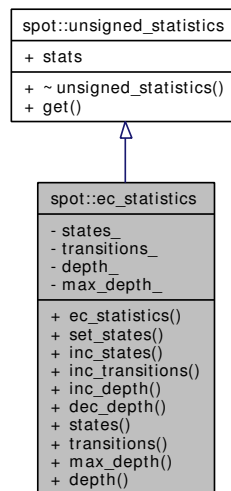
Emptiness-check statistics.

```
#include <tgbaalgos/emptiness_stats.hh>
```

Inheritance diagram for spot::ec\_statistics:



Collaboration diagram for spot::ec\_statistics:



## Public Types

- `typedef unsigned(unsigned_statistics::*) unsigned_fun () const`
- `typedef std::map< const char *, unsigned_fun, char_ptr_less_than > stats_map`

## Public Member Functions

- `ec_statistics ()`
- `void set_states (unsigned n)`
- `void inc_states ()`
- `void inc_transitions ()`
- `void inc_depth (unsigned n=1)`
- `void dec_depth (unsigned n=1)`
- `unsigned states () const`
- `unsigned transitions () const`
- `unsigned max_depth () const`
- `unsigned depth () const`
- `unsigned get (const char *str) const`

## Public Attributes

- `stats_map stats`

## Private Attributes

- `unsigned states_`
- `unsigned transitions_`  
*number of distinct visited states*
- `unsigned depth_`  
*number of visited transitions*

- unsigned [max\\_depth\\_](#)  
*maximal depth of the stack(s)*

### 12.30.1 Detailed Description

Emptiness-check statistics.

Implementations of [spot::emptiness\\_check](#) may also implement this interface. Try to dynamic\_cast the [spot::emptiness\\_check](#) pointer to know whether these statistics are available.

### 12.30.2 Member Typedef Documentation

**12.30.2.1** `typedef unsigned(unsigned_statistics::*) spot::unsigned_statistics::unsigned_fun() const` [\[inherited\]](#)

**12.30.2.2** `typedef std::map<const char*, unsigned_fun, char_ptr_less_than> spot::unsigned_statistics::stats_map` [\[inherited\]](#)

### 12.30.3 Constructor & Destructor Documentation

**12.30.3.1** `spot::ec_statistics::ec_statistics()` [\[inline\]](#)

### 12.30.4 Member Function Documentation

**12.30.4.1** `void spot::ec_statistics::set_states(unsigned n)` [\[inline\]](#)

**12.30.4.2** `void spot::ec_statistics::inc_states()` [\[inline\]](#)

**12.30.4.3** `void spot::ec_statistics::inc_transitions()` [\[inline\]](#)

**12.30.4.4** `void spot::ec_statistics::inc_depth(unsigned n = 1)` [\[inline\]](#)

**12.30.4.5** `void spot::ec_statistics::dec_depth(unsigned n = 1)` [\[inline\]](#)

**12.30.4.6** `unsigned spot::ec_statistics::states() const` [\[inline\]](#)

**12.30.4.7** `unsigned spot::ec_statistics::transitions() const` [\[inline\]](#)

**12.30.4.8** `unsigned spot::ec_statistics::max_depth() const` [\[inline\]](#)

**12.30.4.9** `unsigned spot::ec_statistics::depth() const` [\[inline\]](#)

**12.30.4.10** unsigned spot::unsigned\_statistics::get (const char \* str) const [inline, inherited]

## 12.30.5 Member Data Documentation

**12.30.5.1** unsigned spot::ec\_statistics::states\_ [private]

**12.30.5.2** unsigned spot::ec\_statistics::transitions\_ [private]  
number of distinct visited states

**12.30.5.3** unsigned spot::ec\_statistics::depth\_ [private]  
number of visited transitions

**12.30.5.4** unsigned spot::ec\_statistics::max\_depth\_ [private]  
maximal depth of the stack(s)

**12.30.5.5** stats\_map spot::unsigned\_statistics::stats [inherited]  
The documentation for this class was generated from the following file:

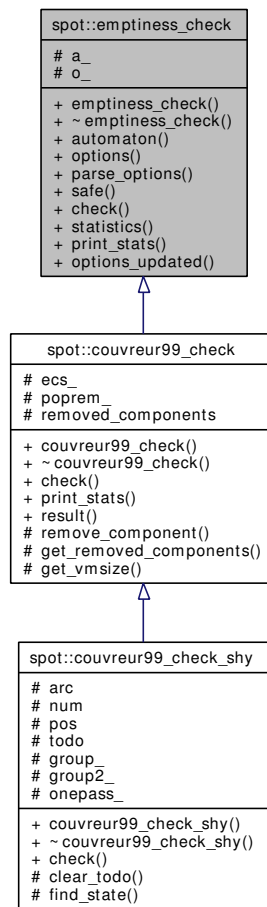
- tgbaalgos/[emptiness\\_stats.hh](#)

## 12.31 spot::emptiness\_check Class Reference

Common interface to emptiness check algorithms.

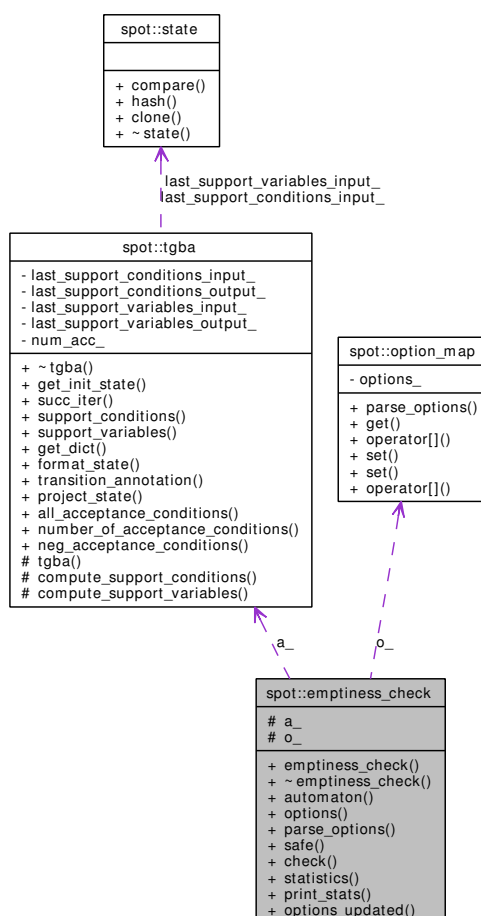
```
#include <tgbaalgos/emptiness.hh>
```

Inheritance diagram for spot::emptiness\_check:





Collaboration diagram for spot::emptiness\_check:



## Public Member Functions

- `emptiness_check` (const `tgba` \*a, `option_map` o=`option_map`())
- virtual `~emptiness_check` ()
- const `tgba` \* `automaton` () const

*The automaton that this emptiness-check inspects.*

- const `option_map` & `options` () const

*Return the options parametrizing how the emptiness check is realized.*

- const char \* `parse_options` (char \*options)

*Modify the algorithm options.*

- virtual bool `safe` () const

*Return false iff accepting\_run() can return 0 for non-empty automata.*

- virtual `emptiness_check_result` \* `check` ()=0

*Check whether the automaton contain an accepting run.*

- virtual const unsigned\_statistics \* statistics () const  
*Return statistics, if available.*
- virtual std::ostream & print\_stats (std::ostream &os) const  
*Print statistics, if any.*
- virtual void options\_updated (const option\_map &old)  
*Notify option updates.*

### Protected Attributes

- const tgba \* a\_  
*The automaton.*
- option\_map o\_  
*The options.*

#### 12.31.1 Detailed Description

Common interface to emptiness check algorithms.

#### 12.31.2 Constructor & Destructor Documentation

**12.31.2.1** spot::emptiness\_check::emptiness\_check (const tgba \* a, option\_map o = option\_map ())  
[inline]

**12.31.2.2** virtual spot::emptiness\_check::~~emptiness\_check () [virtual]

#### 12.31.3 Member Function Documentation

**12.31.3.1** const tgba\* spot::emptiness\_check::automaton () const [inline]

The automaton that this emptiness-check inspects.

**12.31.3.2** const option\_map& spot::emptiness\_check::options () const [inline]

Return the options parametrizing how the emptiness check is realized.

**12.31.3.3** const char\* spot::emptiness\_check::parse\_options (char \* options)

Modify the algorithm options.

**12.31.3.4** virtual bool spot::emptiness\_check::safe () const [virtual]

Return false iff accepting\_run() can return 0 for non-empty automata.

**12.31.3.5 virtual emptiness\_check\_result\* spot::emptiness\_check::check () [pure virtual]**

Check whether the automaton contain an accepting run.

Return 0 if the automaton accepts no run. Return an instance of [emptiness\\_check\\_result](#) otherwise. This instance might allow to obtain one sample acceptance run. The result has to be destroyed before the [emptiness\\_check](#) instance that generated it.

Some [emptiness\\_check](#) algorithms may allow [check\(\)](#) to be called several time, but generally you should not assume that.

Some [emptiness\\_check](#) algorithms, especially those using bit [state](#) hashing may return 0 even if the automaton is not empty.

See also:

[safe\(\)](#)

Implemented in [spot::couvreur99\\_check](#), and [spot::couvreur99\\_check\\_shy](#).

**12.31.3.6 virtual const unsigned\_statistics\* spot::emptiness\_check::statistics () const [virtual]**

Return statistics, if available.

**12.31.3.7 virtual std::ostream& spot::emptiness\_check::print\_stats (std::ostream & os) const [virtual]**

Print statistics, if any.

Reimplemented in [spot::couvreur99\\_check](#).

**12.31.3.8 virtual void spot::emptiness\_check::options\_updated (const option\_map & old) [virtual]**

Notify option updates.

**12.31.4 Member Data Documentation****12.31.4.1 const tgba\* spot::emptiness\_check::a\_ [protected]**

The automaton.

**12.31.4.2 option\_map spot::emptiness\_check::o\_ [protected]**

The options.

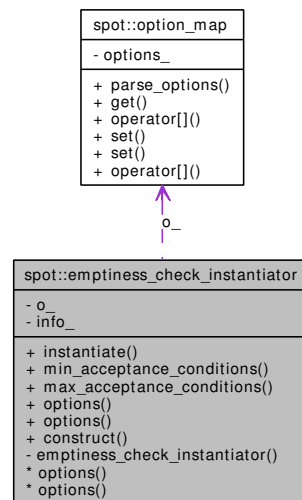
The documentation for this class was generated from the following file:

- [tgbaalgos/emptiness.hh](#)

**12.32 spot::emptiness\_check\_instantiator Class Reference**

```
#include <tgbaalgos/emptiness.hh>
```

Collaboration diagram for spot::emptiness\_check\_instantiator:



### Public Member Functions

- `emptiness_check * instantiate (const tgba *a) const`  
*Actually instantiate the emptiness check, for a.*
- `unsigned int min_acceptance_conditions () const`  
*Minimum number of acceptance conditions supported by the emptiness check.*
- `unsigned int max_acceptance_conditions () const`  
*Maximum number of acceptance conditions supported by the emptiness check.*
- `const option_map & options () const`
- `option_map & options ()`

### Static Public Member Functions

- `static emptiness_check_instantiator * construct (const char *name, const char **err)`  
*Create an emptiness-check instantiator, given the name of an emptiness check.*

### Private Member Functions

- `emptiness_check_instantiator (option_map o, void *i)`

### Private Attributes

- `option_map o_`
- `void * info_`

### 12.32.1 Constructor & Destructor Documentation

**12.32.1.1** `spot::emptiness_check_instantiator::emptiness_check_instantiator (option_map o, void *i) [private]`

### 12.32.2 Member Function Documentation

**12.32.2.1** `static emptiness_check_instantiator* spot::emptiness_check_instantiator::construct (const char * name, const char ** err) [static]`

Create an emptiness-check instantiator, given the name of an emptiness check.

*name* should have the form "name" or "name (options)".

On error, the function returns 0. If the name of the algorithm was unknown, *\*err* will be set to *name*. If some fragment of the options could not be parsed, *\*err* will point to that fragment.

**12.32.2.2** `emptiness_check* spot::emptiness_check_instantiator::instantiate (const tgba * a) const`

Actually instantiate the emptiness check, for *a*.

**12.32.2.3** `const option_map& spot::emptiness_check_instantiator::options () const [inline]`

Accessor to the options.

**12.32.2.4** `option_map& spot::emptiness_check_instantiator::options () [inline]`

**12.32.2.5** `unsigned int spot::emptiness_check_instantiator::min_acceptance_conditions () const`

Minimum number of acceptance conditions supported by the emptiness check.

**12.32.2.6** `unsigned int spot::emptiness_check_instantiator::max_acceptance_conditions () const`

Maximum number of acceptance conditions supported by the emptiness check.

#### Returns:

−1U if no upper bound exists.

### 12.32.3 Member Data Documentation

**12.32.3.1** `option_map spot::emptiness_check_instantiator::o_ [private]`

**12.32.3.2** `void* spot::emptiness_check_instantiator::info_ [private]`

The documentation for this class was generated from the following file:

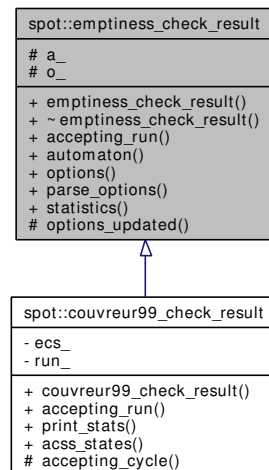
- [tgbaalgos/emptiness.hh](#)

## 12.33 spot::emptiness\_check\_result Class Reference

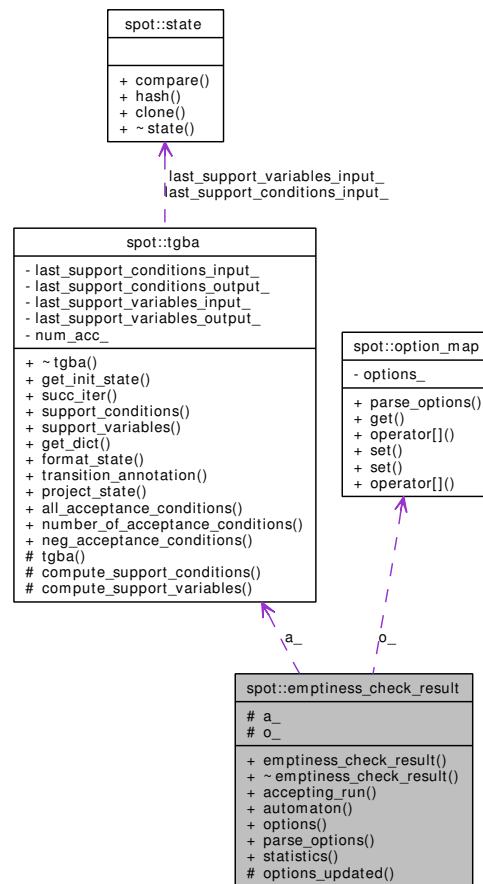
The result of an emptiness check.

```
#include <tgbaaalgos/emptiness.hh>
```

Inheritance diagram for spot::emptiness\_check\_result:



Collaboration diagram for spot::emptiness\_check\_result:



## Public Member Functions

- [emptiness\\_check\\_result](#) (const [tgba](#) \*a, [option\\_map](#) o=[option\\_map](#)())
- virtual [~emptiness\\_check\\_result](#) ()
- virtual [tgba\\_run](#) \* [accepting\\_run](#) ()

*Return a run accepted by the automata passed to the emptiness check.*

- const [tgba](#) \* [automaton](#) () const

*The automaton on which an [accepting\\_run\(\)](#) was found.*

- const [option\\_map](#) & [options](#) () const

*Return the options parametrizing how the accepting run is computed.*

- const char \* [parse\\_options](#) (char \*options)

*Modify the algorithm options.*

- virtual const [unsigned\\_statistics](#) \* [statistics](#) () const

*Return statistics, if available.*

### Protected Member Functions

- virtual void `options_updated` (const `option_map` &old)

*Notify option updates.*

### Protected Attributes

- const `tgba` \* `a_`  
*The automaton.*
- `option_map` `o_`  
*The options.*

#### 12.33.1 Detailed Description

The result of an emptiness check.

Instances of these class should not last longer than the instances of `emptiness_check` that produced them as they may reference data internal to the check.

#### 12.33.2 Constructor & Destructor Documentation

**12.33.2.1** `spot::emptiness_check_result::emptiness_check_result (const tgba * a, option_map o = option_map ())` `[inline]`

**12.33.2.2** `virtual spot::emptiness_check_result::~~emptiness_check_result ()` `[inline, virtual]`

#### 12.33.3 Member Function Documentation

**12.33.3.1** `virtual tgba_run* spot::emptiness_check_result::accepting_run ()` `[virtual]`

Return a run accepted by the automata passed to the emptiness check.

This method might actually compute the acceptance run. (Not all emptiness check algorithms actually produce a counter-example as a side-effect of checking emptiness, some need some post-processing.)

This can also return 0 if the emptiness check algorithm cannot produce a counter example (that does not mean there is no counter-example; the mere existence of an instance of this class asserts the existence of a counter-example).

Reimplemented in `spot::couvreur99_check_result`.

**12.33.3.2** `const tgba* spot::emptiness_check_result::automaton () const` `[inline]`

The automaton on which an `accepting_run()` was found.

**12.33.3.3** `const option_map& spot::emptiness_check_result::options () const` `[inline]`

Return the options parametrizing how the accepting run is computed.



**12.33.3.4** `const char* spot::emptiness_check_result::parse_options (char * options)`

Modify the algorithm options.

**12.33.3.5** `virtual const unsigned_statistics* spot::emptiness_check_result::statistics () const`  
[virtual]

Return statistics, if available.

**12.33.3.6** `virtual void spot::emptiness_check_result::options_updated (const option_map & old)`  
[protected, virtual]

Notify option updates.

**12.33.4 Member Data Documentation****12.33.4.1** `const tgba* spot::emptiness_check_result::a_` [protected]

The automaton.

**12.33.4.2** `option_map spot::emptiness_check_result::o_` [protected]

The options.

The documentation for this class was generated from the following file:

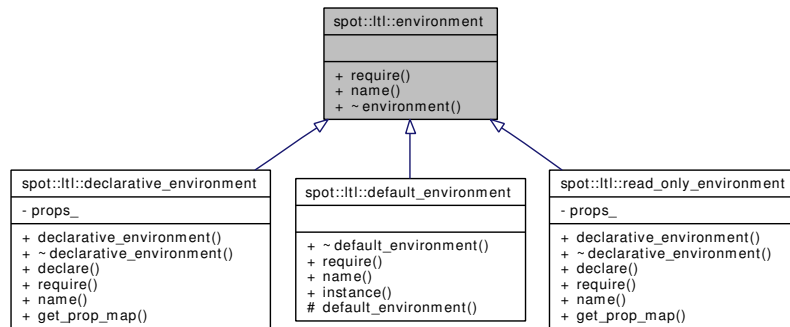
- [tgbaalgos/emptiness.hh](#)

**12.34 spot::ltl::environment Class Reference**

An [environment](#) that describes atomic propositions.

```
#include <ltlenv/environment.hh>
```

Inheritance diagram for `spot::ltl::environment`:

**Public Member Functions**

- virtual [formula](#) \* [require](#) (const std::string &prop\_str)=0

*Obtain the [formula](#) associated to prop\_str.*

- virtual const std::string & [name](#) ()=0  
*Get the name of the [environment](#).*
- virtual [~environment](#) ()

### 12.34.1 Detailed Description

An [environment](#) that describes atomic propositions.

### 12.34.2 Constructor & Destructor Documentation

#### 12.34.2.1 virtual spot::ltl::environment::~~environment () [inline, virtual]

### 12.34.3 Member Function Documentation

#### 12.34.3.1 virtual formula\* spot::ltl::environment::require (const std::string & *prop\_str*) [pure virtual]

Obtain the [formula](#) associated to *prop\_str*.

Usually *prop\_str*, is the name of an atomic proposition, and spot::ltl::require simply returns the associated [spot::ltl::atomic\\_prop](#).

Note this is not a const method. Some environments will "create" the atomic proposition when requested.

We return a [spot::ltl::formula](#) instead of an [spot::ltl::atomic\\_prop](#), because this will allow nifty tricks (e.g., we could name formulae in an [environment](#), and let the parser build a larger tree from these).

#### Returns:

0 iff *prop\_str* is not part of the [environment](#), or the associated [spot::ltl::formula](#) otherwise.

Implemented in [spot::ltl::declarative\\_environment](#), [spot::ltl::default\\_environment](#), and [spot::ltl::read\\_only\\_environment](#).

#### 12.34.3.2 virtual const std::string& spot::ltl::environment::name () [pure virtual]

Get the name of the [environment](#).

Implemented in [spot::ltl::declarative\\_environment](#), [spot::ltl::default\\_environment](#), and [spot::ltl::read\\_only\\_environment](#).

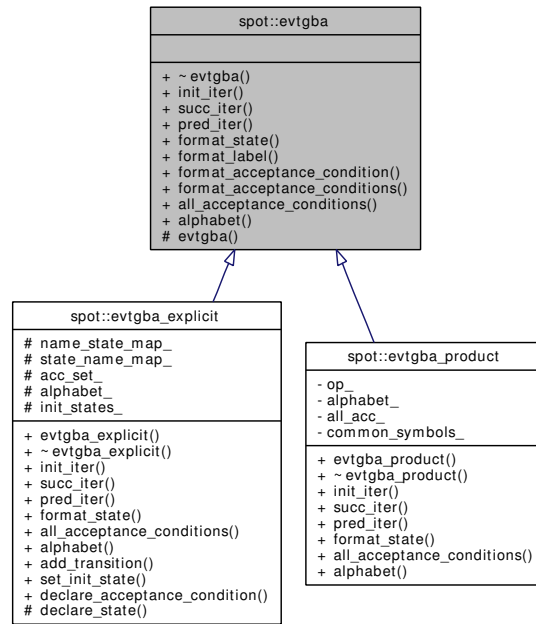
The documentation for this class was generated from the following file:

- [ltlenv/environment.hh](#)

## 12.35 spot::evtgba Class Reference

```
#include <evtgba/evtgba.hh>
```

Inheritance diagram for spot::evtgba:



### Public Member Functions

- virtual `~evtgba ()`
- virtual `evtgba_iterator * init_iter ()` const=0
- virtual `evtgba_iterator * succ_iter (const state *s)` const=0
- virtual `evtgba_iterator * pred_iter (const state *s)` const=0
- virtual `std::string format_state (const state *state)` const=0

*Format the `state` as a string for printing.*

- virtual `std::string format_label (const symbol *symbol)` const
- virtual `std::string format_acceptance_condition (const symbol *symbol)` const
- virtual `std::string format_acceptance_conditions (const symbol_set &symset)` const
- virtual `const symbol_set & all_acceptance_conditions ()` const=0

*Return the set of all acceptance conditions used by this automaton.*

- virtual `const symbol_set & alphabet ()` const=0

### Protected Member Functions

- `evtgba ()`

#### 12.35.1 Constructor & Destructor Documentation

##### 12.35.1.1 spot::evtgba::evtgba () [protected]

##### 12.35.1.2 virtual spot::evtgba::~~evtgba () [virtual]

## 12.35.2 Member Function Documentation

### 12.35.2.1 virtual evtgba\_iterator\* spot::evtgba::init\_iter () const [pure virtual]

Implemented in [spot::evtgba\\_explicit](#), and [spot::evtgba\\_product](#).

### 12.35.2.2 virtual evtgba\_iterator\* spot::evtgba::succ\_iter (const state \* s) const [pure virtual]

Implemented in [spot::evtgba\\_product](#).

### 12.35.2.3 virtual evtgba\_iterator\* spot::evtgba::pred\_iter (const state \* s) const [pure virtual]

Implemented in [spot::evtgba\\_product](#).

### 12.35.2.4 virtual std::string spot::evtgba::format\_state (const state \* state) const [pure virtual]

Format the [state](#) as a string for printing.

This formatting is the responsibility of the automata who owns the [state](#).

Implemented in [spot::evtgba\\_product](#).

### 12.35.2.5 virtual std::string spot::evtgba::format\_label (const symbol \* symbol) const [virtual]

### 12.35.2.6 virtual std::string spot::evtgba::format\_acceptance\_condition (const symbol \* symbol) const [virtual]

### 12.35.2.7 virtual std::string spot::evtgba::format\_acceptance\_conditions (const symbol\_set & sym-set) const [virtual]

### 12.35.2.8 virtual const symbol\_set& spot::evtgba::all\_acceptance\_conditions () const [pure virtual]

Return the set of all acceptance conditions used by this automaton.

The goal of the emptiness check is to ensure that a strongly connected component walks through each of these acceptance conditions. I.e., the union of the acceptance conditions of all transition in the SCC should be equal to the result of this function.

Implemented in [spot::evtgba\\_explicit](#), and [spot::evtgba\\_product](#).

### 12.35.2.9 virtual const symbol\_set& spot::evtgba::alphabet () const [pure virtual]

Implemented in [spot::evtgba\\_explicit](#), and [spot::evtgba\\_product](#).

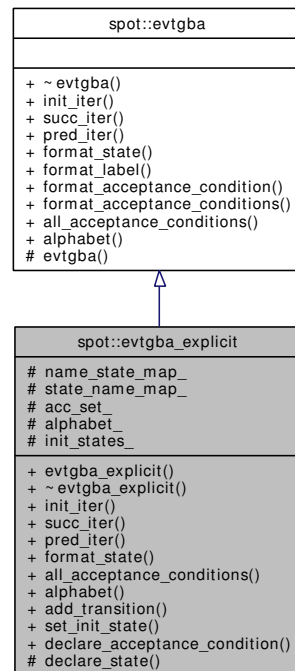
The documentation for this class was generated from the following file:

- [evtgba/evtgba.hh](#)

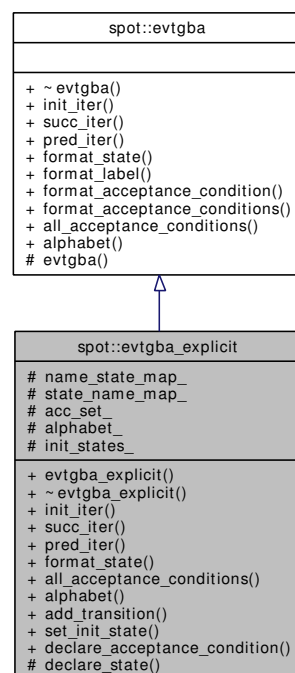
## 12.36 spot::evtgba\_explicit Class Reference

```
#include <evtgba/explicit.hh>
```

Inheritance diagram for spot::evtgba\_explicit:



Collaboration diagram for spot::evtgba\_explicit:



## Public Types

- typedef std::list< [transition](#) \* > [transition\\_list](#)

## Public Member Functions

- [evtgba\\_explicit](#) ()
- virtual [~evtgba\\_explicit](#) ()
- virtual [evtgba\\_iterator](#) \* [init\\_iter](#) () const
- virtual [evtgba\\_iterator](#) \* [succ\\_iter](#) (const [spot::state](#) \*s) const
- virtual [evtgba\\_iterator](#) \* [pred\\_iter](#) (const [spot::state](#) \*s) const
- virtual std::string [format\\_state](#) (const [spot::state](#) \*state) const
- virtual const [symbol\\_set](#) & [all\\_acceptance\\_conditions](#) () const  
*Return the set of all acceptance conditions used by this automaton.*
- virtual const [symbol\\_set](#) & [alphabet](#) () const
- [transition](#) \* [add\\_transition](#) (const std::string &source, const [rsymbol](#) &label, [rsymbol\\_set](#) acc, const std::string &dest)
- void [set\\_init\\_state](#) (const std::string &name)  
*Designate name as initial [state](#).*
- void [declare\\_acceptance\\_condition](#) (const [rsymbol](#) &acc)
- virtual [evtgba\\_iterator](#) \* [succ\\_iter](#) (const [state](#) \*s) const=0
- virtual [evtgba\\_iterator](#) \* [pred\\_iter](#) (const [state](#) \*s) const=0
- virtual std::string [format\\_state](#) (const [state](#) \*state) const=0  
*Format the [state](#) as a string for printing.*
- virtual std::string [format\\_label](#) (const [symbol](#) \*symbol) const
- virtual std::string [format\\_acceptance\\_condition](#) (const [symbol](#) \*symbol) const
- virtual std::string [format\\_acceptance\\_conditions](#) (const [symbol\\_set](#) &symset) const

## Protected Types

- typedef Sgi::hash\_map< const std::string, [evtgba\\_explicit::state](#) \*, [string\\_hash](#) > [ns\\_map](#)
- typedef Sgi::hash\_map< const [evtgba\\_explicit::state](#) \*, std::string, [ptr\\_hash](#)< [evtgba\\_explicit::state](#) > > [sn\\_map](#)

## Protected Member Functions

- [state](#) \* [declare\\_state](#) (const std::string &name)

## Protected Attributes

- [ns\\_map](#) [name\\_state\\_map\\_](#)
- [sn\\_map](#) [state\\_name\\_map\\_](#)
- [symbol\\_set](#) [acc\\_set\\_](#)
- [symbol\\_set](#) [alphabet\\_](#)
- [transition\\_list](#) [init\\_states\\_](#)

## Classes

- struct [state](#)
- struct [transition](#)

*Explicit transitions (used by [spot::evtgba\\_explicit](#)).*

## 12.36.1 Member Typedef Documentation

**12.36.1.1** `typedef std::list<transition*> spot::evtgba_explicit::transition_list`

**12.36.1.2** `typedef Sgi::hash_map<const std::string, evtgba_explicit::state*, string_hash> spot::evtgba_explicit::ns_map` [protected]

**12.36.1.3** `typedef Sgi::hash_map<const evtgba_explicit::state*, std::string, ptr_hash<evtgba_explicit::state>> spot::evtgba_explicit::sn_map` [protected]

## 12.36.2 Constructor &amp; Destructor Documentation

**12.36.2.1** `spot::evtgba_explicit::evtgba_explicit ()`

**12.36.2.2** `virtual spot::evtgba_explicit::~~evtgba_explicit ()` [virtual]

## 12.36.3 Member Function Documentation

**12.36.3.1** `virtual evtgba_iterator* spot::evtgba_explicit::init_iter () const` [virtual]

Implements [spot::evtgba](#).

**12.36.3.2** `virtual evtgba_iterator* spot::evtgba_explicit::succ_iter (const spot::state * s) const` [virtual]

**12.36.3.3** `virtual evtgba_iterator* spot::evtgba_explicit::pred_iter (const spot::state * s) const` [virtual]

**12.36.3.4** `virtual std::string spot::evtgba_explicit::format_state (const spot::state * state) const` [virtual]

**12.36.3.5** `virtual const symbol_set& spot::evtgba_explicit::all_acceptance_conditions () const` [virtual]

Return the set of all acceptance conditions used by this automaton.

The goal of the emptiness check is to ensure that a strongly connected component walks through each of these acceptance conditions. I.e., the union of the acceptance conditions of all [transition](#) in the SCC should be equal to the result of this function.

Implements [spot::evtgba](#).

**12.36.3.6** virtual const symbol\_set& spot::evtgba\_explicit::alphabet () const [virtual]

Implements [spot::evtgba](#).

**12.36.3.7** transition\* spot::evtgba\_explicit::add\_transition (const std::string & *source*, const rsymbol & *label*, rsymbol\_set *acc*, const std::string & *dest*)

**12.36.3.8** void spot::evtgba\_explicit::set\_init\_state (const std::string & *name*)

Designate *name* as initial [state](#).

Can be called multiple times in case there is several initial states.

**12.36.3.9** void spot::evtgba\_explicit::declare\_acceptance\_condition (const rsymbol & *acc*)

**12.36.3.10** state\* spot::evtgba\_explicit::declare\_state (const std::string & *name*) [protected]

**12.36.3.11** virtual evtgba\_iterator\* spot::evtgba::succ\_iter (const state \* *s*) const [pure virtual, inherited]

Implemented in [spot::evtgba\\_product](#).

**12.36.3.12** virtual evtgba\_iterator\* spot::evtgba::pred\_iter (const state \* *s*) const [pure virtual, inherited]

Implemented in [spot::evtgba\\_product](#).

**12.36.3.13** virtual std::string spot::evtgba::format\_state (const state \* *state*) const [pure virtual, inherited]

Format the [state](#) as a string for printing.

This formatting is the responsibility of the automata who owns the [state](#).

Implemented in [spot::evtgba\\_product](#).

**12.36.3.14** virtual std::string spot::evtgba::format\_label (const symbol \* *symbol*) const [virtual, inherited]

**12.36.3.15** virtual std::string spot::evtgba::format\_acceptance\_condition (const symbol \* *symbol*) const [virtual, inherited]

**12.36.3.16** virtual std::string spot::evtgba::format\_acceptance\_conditions (const symbol\_set & *symset*) const [virtual, inherited]

## 12.36.4 Member Data Documentation

**12.36.4.1** ns\_map spot::evtgba\_explicit::name\_state\_map\_ [protected]



**12.36.4.2** sn\_map spot::evtgba\_explicit::state\_name\_map\_ [protected]

**12.36.4.3** symbol\_set spot::evtgba\_explicit::acc\_set\_ [protected]

**12.36.4.4** symbol\_set spot::evtgba\_explicit::alphabet\_ [protected]

**12.36.4.5** transition\_list spot::evtgba\_explicit::init\_states\_ [protected]

The documentation for this class was generated from the following file:

- evtgba/[explicit.hh](#)

## 12.37 spot::evtgba\_explicit::state Struct Reference

```
#include <evtgba/explicit.hh>
```

### Public Attributes

- [transition\\_list](#) in
- [transition\\_list](#) out

### 12.37.1 Member Data Documentation

**12.37.1.1** transition\_list spot::evtgba\_explicit::state::in

**12.37.1.2** transition\_list spot::evtgba\_explicit::state::out

The documentation for this struct was generated from the following file:

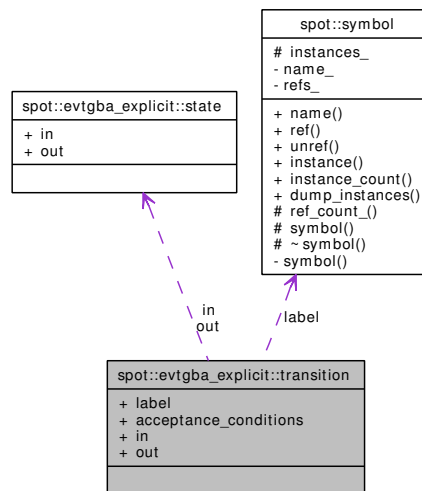
- evtgba/[explicit.hh](#)

## 12.38 spot::evtgba\_explicit::transition Struct Reference

Explicit transitions (used by [spot::evtgba\\_explicit](#)).

```
#include <evtgba/explicit.hh>
```

Collaboration diagram for spot::evtgba\_explicit::transition:



### Public Attributes

- const [symbol](#) \* [label](#)
- [symbol\\_set](#) [acceptance\\_conditions](#)
- [state](#) \* [in](#)
- [state](#) \* [out](#)

### 12.38.1 Detailed Description

Explicit transitions (used by [spot::evtgba\\_explicit](#)).

### 12.38.2 Member Data Documentation

**12.38.2.1** const [symbol](#)\* [spot::evtgba\\_explicit::transition::label](#)

**12.38.2.2** [symbol\\_set](#) [spot::evtgba\\_explicit::transition::acceptance\\_conditions](#)

**12.38.2.3** [state](#)\* [spot::evtgba\\_explicit::transition::in](#)

**12.38.2.4** [state](#)\* [spot::evtgba\\_explicit::transition::out](#)

The documentation for this struct was generated from the following file:

- [evtgba/explicit.hh](#)

## 12.39 spot::evtgba\_iterator Class Reference

```
#include <evtgba/evtgbaiter.hh>
```

### Public Member Functions

- virtual [~evtgba\\_iterator](#) ()
- virtual void [first](#) ()=0
- virtual void [next](#) ()=0
- virtual bool [done](#) () const=0
- virtual const [state](#) \* [current\\_state](#) () const=0
- virtual const [symbol](#) \* [current\\_label](#) () const=0
- virtual [symbol\\_set](#) [current\\_acceptance\\_conditions](#) () const=0

### 12.39.1 Constructor & Destructor Documentation

**12.39.1.1** virtual [spot::evtgba\\_iterator::~~evtgba\\_iterator](#) () [inline, virtual]

### 12.39.2 Member Function Documentation

**12.39.2.1** virtual void [spot::evtgba\\_iterator::first](#) () [pure virtual]

**12.39.2.2** virtual void [spot::evtgba\\_iterator::next](#) () [pure virtual]

**12.39.2.3** virtual bool [spot::evtgba\\_iterator::done](#) () const [pure virtual]

**12.39.2.4** virtual const [state](#)\* [spot::evtgba\\_iterator::current\\_state](#) () const [pure virtual]

**12.39.2.5** virtual const [symbol](#)\* [spot::evtgba\\_iterator::current\\_label](#) () const [pure virtual]

**12.39.2.6** virtual [symbol\\_set](#) [spot::evtgba\\_iterator::current\\_acceptance\\_conditions](#) () const [pure virtual]

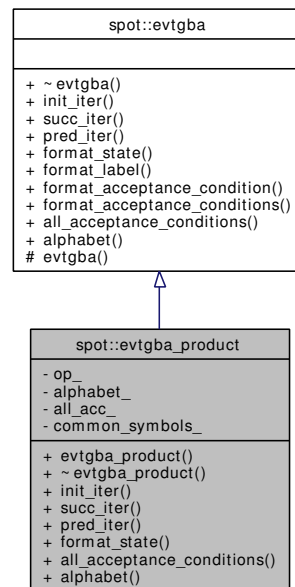
The documentation for this class was generated from the following file:

- [evtgba/evtgba\\_iter.hh](#)

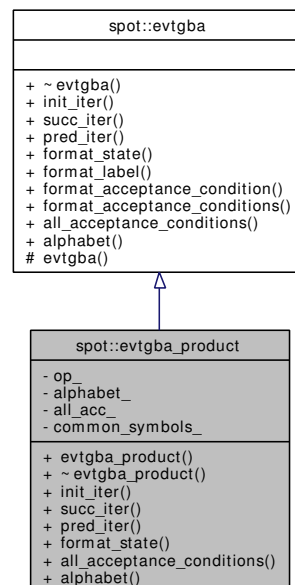
## 12.40 spot::evtgba\_product Class Reference

```
#include <evtgba/product.hh>
```

Inheritance diagram for spot::evtgba\_product:



Collaboration diagram for spot::evtgba\_product:



## Public Types

- typedef std::vector< const [evtgba](#) \* > [evtgba\\_product\\_operands](#)
- typedef std::map< const [symbol](#) \*, std::set< int > > [common\\_symbol\\_table](#)

## Public Member Functions

- [evtgba\\_product](#) (const [evtgba\\_product\\_operands](#) &op)

- virtual [~evtgba\\_product](#) ()
- virtual [evtgba\\_iterator](#) \* [init\\_iter](#) () const
- virtual [evtgba\\_iterator](#) \* [succ\\_iter](#) (const [state](#) \*s) const
- virtual [evtgba\\_iterator](#) \* [pred\\_iter](#) (const [state](#) \*s) const
- virtual std::string [format\\_state](#) (const [state](#) \*state) const  
*Format the [state](#) as a string for printing.*
- virtual const [symbol\\_set](#) & [all\\_acceptance\\_conditions](#) () const  
*Return the set of all acceptance conditions used by this automaton.*
- virtual const [symbol\\_set](#) & [alphabet](#) () const
- virtual std::string [format\\_label](#) (const [symbol](#) \*symbol) const
- virtual std::string [format\\_acceptance\\_condition](#) (const [symbol](#) \*symbol) const
- virtual std::string [format\\_acceptance\\_conditions](#) (const [symbol\\_set](#) &symset) const

### Private Attributes

- const [evtgba\\_product\\_operands](#) op\_
- [symbol\\_set](#) alphabet\_
- [symbol\\_set](#) all\_acc\_
- [common\\_symbol\\_table](#) common\_symbols\_

### 12.40.1 Member Typedef Documentation

**12.40.1.1** typedef std::vector<const [evtgba](#)\*> [spot::evtgba\\_product::evtgba\\_product\\_operands](#)

**12.40.1.2** typedef std::map<const [symbol](#)\*, std::set<int> > [spot::evtgba\\_product::common\\_symbol\\_table](#)

### 12.40.2 Constructor & Destructor Documentation

**12.40.2.1** [spot::evtgba\\_product::evtgba\\_product](#) (const [evtgba\\_product\\_operands](#) & op)

**12.40.2.2** virtual [spot::evtgba\\_product::~~evtgba\\_product](#) () [virtual]

### 12.40.3 Member Function Documentation

**12.40.3.1** virtual [evtgba\\_iterator](#)\* [spot::evtgba\\_product::init\\_iter](#) () const [virtual]

Implements [spot::evtgba](#).

**12.40.3.2** virtual [evtgba\\_iterator](#)\* [spot::evtgba\\_product::succ\\_iter](#) (const [state](#) \* s) const [virtual]

Implements [spot::evtgba](#).

**12.40.3.3** virtual [evtgba\\_iterator](#)\* [spot::evtgba\\_product::pred\\_iter](#) (const [state](#) \* s) const [virtual]

Implements [spot::evtgba](#).

**12.40.3.4** `virtual std::string spot::evtgba_product::format_state (const state * state) const` [virtual]

Format the [state](#) as a string for printing.

This formatting is the responsibility of the automata who owns the [state](#).

Implements [spot::evtgba](#).

**12.40.3.5** `virtual const symbol_set& spot::evtgba_product::all_acceptance_conditions () const` [virtual]

Return the set of all acceptance conditions used by this automaton.

The goal of the emptiness check is to ensure that a strongly connected component walks through each of these acceptance conditions. I.e., the union of the acceptance conditions of all transition in the SCC should be equal to the result of this function.

Implements [spot::evtgba](#).

**12.40.3.6** `virtual const symbol_set& spot::evtgba_product::alphabet () const` [virtual]

Implements [spot::evtgba](#).

**12.40.3.7** `virtual std::string spot::evtgba::format_label (const symbol * symbol) const` [virtual, inherited]

**12.40.3.8** `virtual std::string spot::evtgba::format_acceptance_condition (const symbol * symbol) const` [virtual, inherited]

**12.40.3.9** `virtual std::string spot::evtgba::format_acceptance_conditions (const symbol_set & sym-set) const` [virtual, inherited]

## 12.40.4 Member Data Documentation

**12.40.4.1** `const evtgba_product_operands spot::evtgba_product::op_` [private]

**12.40.4.2** `symbol_set spot::evtgba_product::alphabet_` [private]

**12.40.4.3** `symbol_set spot::evtgba_product::all_acc_` [private]

**12.40.4.4** `common_symbol_table spot::evtgba_product::common_symbols_` [private]

The documentation for this class was generated from the following file:

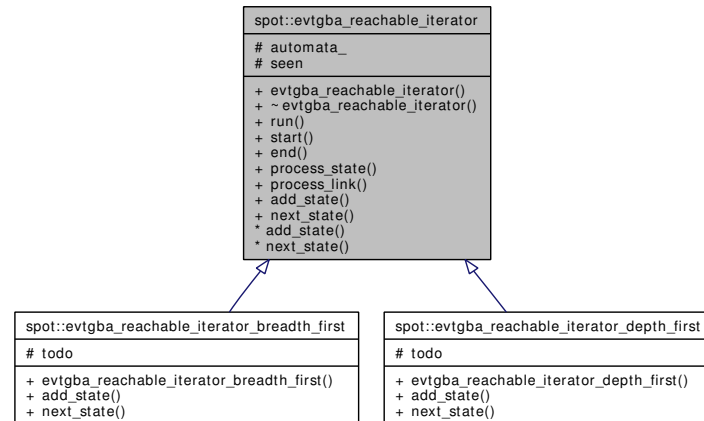
- [evtgba/product.hh](#)

## 12.41 spot::evtgba\_reachable\_iterator Class Reference

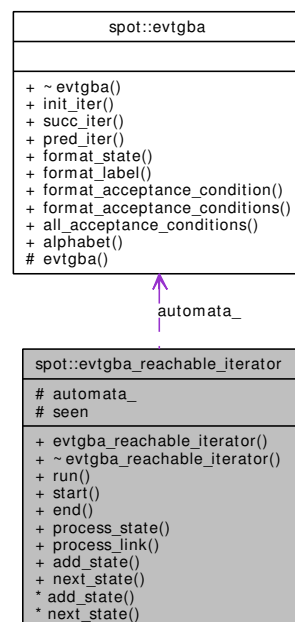
Iterate over all reachable states of a [spot::evtgba](#).

```
#include <evtgbaalgos/reachiter.hh>
```

Inheritance diagram for spot::evtgba\_reachable\_iterator:



Collaboration diagram for spot::evtgba\_reachable\_iterator:



## Public Member Functions

- [evtgba\\_reachable\\_iterator](#) (const [evtgba](#) \*a)
- virtual [~evtgba\\_reachable\\_iterator](#) ()
- void [run](#) ()  
*Iterate over all reachable states of a [spot::evtgba](#).*
- virtual void [start](#) (int n)  
*Called by [run\(\)](#) before starting its iteration.*

- virtual void `end()`  
*Called by `run()` once all states have been explored.*
- virtual void `process_state` (const `state` \*s, int n, `evtgba_iterator` \*si)
- virtual void `process_link` (int in, int out, const `evtgba_iterator` \*si)

**Todo list management.**

*Called by `run()` to register newly discovered states.*

*`spot::evtgba_reachable_iterator_depth_first` and `spot::evtgba_reachable_iterator_breadth_first` offer two precanned implementations for these functions.*

- virtual void `add_state` (const `state` \*s)=0
- virtual const `state` \* `next_state` ()=0  
*Called by `run()` to obtain the.*

**Protected Types**

- typedef Sgi::hash\_map< const `state` \*, int, `state_ptr_hash`, `state_ptr_equal` > `seen_map`

**Protected Attributes**

- const `evtgba` \* `automata_`  
*The `spot::evtgba` to explore.*
- `seen_map` `seen`  
*States already seen.*

**12.41.1 Detailed Description**

Iterate over all reachable states of a `spot::evtgba`.

**12.41.2 Member Typedef Documentation**

**12.41.2.1** typedef Sgi::hash\_map<const `state`\*, int, `state_ptr_hash`, `state_ptr_equal`>  
`spot::evtgba_reachable_iterator::seen_map` [protected]

**12.41.3 Constructor & Destructor Documentation**

**12.41.3.1** `spot::evtgba_reachable_iterator::evtgba_reachable_iterator` (const `evtgba` \* a)

**12.41.3.2** virtual `spot::evtgba_reachable_iterator::~~evtgba_reachable_iterator` () [virtual]



### 12.41.4 Member Function Documentation

#### 12.41.4.1 void spot::evtgba\_reachable\_iterator::run ()

Iterate over all reachable states of a [spot::evtgba](#).

This is a template method that will call [add\\_state\(\)](#), [next\\_state\(\)](#), [start\(\)](#), [end\(\)](#), [process\\_state\(\)](#), and [process\\_link\(\)](#), while it iterate over [state](#).

#### 12.41.4.2 virtual void spot::evtgba\_reachable\_iterator::add\_state (const state \* s) [pure virtual]

Implemented in [spot::evtgba\\_reachable\\_iterator\\_depth\\_first](#), and [spot::evtgba\\_reachable\\_iterator\\_breadth\\_first](#).

#### 12.41.4.3 virtual const state\* spot::evtgba\_reachable\_iterator::next\_state () [pure virtual]

Called by [run\(\)](#) to obtain the.

Implemented in [spot::evtgba\\_reachable\\_iterator\\_depth\\_first](#), and [spot::evtgba\\_reachable\\_iterator\\_breadth\\_first](#).

#### 12.41.4.4 virtual void spot::evtgba\_reachable\_iterator::start (int n) [virtual]

Called by [run\(\)](#) before starting its iteration.

##### Parameters:

*n* The number of initial states.

#### 12.41.4.5 virtual void spot::evtgba\_reachable\_iterator::end () [virtual]

Called by [run\(\)](#) once all states have been explored.

#### 12.41.4.6 virtual void spot::evtgba\_reachable\_iterator::process\_state (const state \* s, int n, evtgba\_iterator \* si) [virtual]

Called by [run\(\)](#) to process a [state](#).

##### Parameters:

*s* The current [state](#).

*n* An unique number assigned to *s*.

*si* The [spot::evtgba\\_iterator](#) for *s*.

#### 12.41.4.7 virtual void spot::evtgba\_reachable\_iterator::process\_link (int in, int out, const evtgba\_iterator \* si) [virtual]

Called by [run\(\)](#) to process a transition.

##### Parameters:

*in* The source [state](#) number.

*out* The destination [state](#) number.

*si* The [spot::evtgba\\_iterator](#) positionned on the current transition.

### 12.41.5 Member Data Documentation

#### 12.41.5.1 const evtgba\* spot::evtgba\_reachable\_iterator::automata\_ [protected]

The [spot::evtgba](#) to explore.

#### 12.41.5.2 seen\_map spot::evtgba\_reachable\_iterator::seen [protected]

States already seen.

The documentation for this class was generated from the following file:

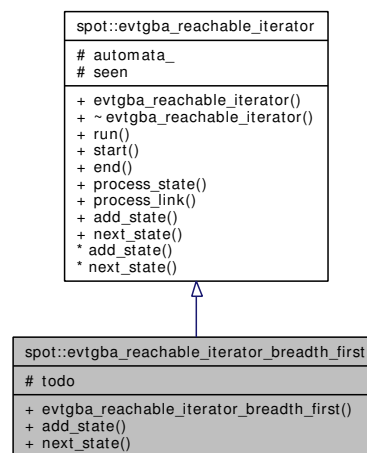
- evtgbaalgos/[reachiter.hh](#)

## 12.42 spot::evtgba\_reachable\_iterator\_breadth\_first Class Reference

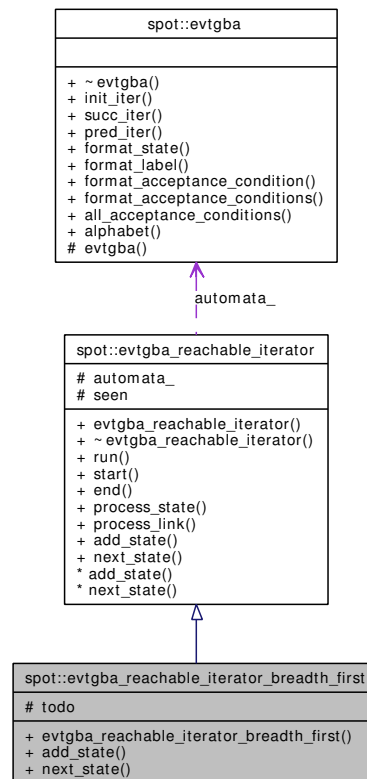
An implementation of [spot::evtgba\\_reachable\\_iterator](#) that browses states breadth first.

```
#include <evtgbaalgos/reachiter.hh>
```

Inheritance diagram for spot::evtgba\_reachable\_iterator\_breadth\_first:



Collaboration diagram for spot::evtgba\_reachable\_iterator\_breadth\_first:



## Public Member Functions

- `evtgba_reachable_iterator_breadth_first` (const `evtgba` \*a)
- virtual void `add_state` (const `state` \*s)
- virtual const `state` \* `next_state` ()  
*Called by `run()` to obtain the.*
- void `run` ()  
*Iterate over all reachable states of a `spot::evtgba`.*
- virtual void `start` (int n)  
*Called by `run()` before starting its iteration.*
- virtual void `end` ()  
*Called by `run()` once all states have been explored.*
- virtual void `process_state` (const `state` \*s, int n, `evtgba_iterator` \*si)
- virtual void `process_link` (int in, int out, const `evtgba_iterator` \*si)

## Protected Types

- typedef `Sgi::hash_map`< const `state` \*, int, `state_ptr_hash`, `state_ptr_equal` > `seen_map`

## Protected Attributes

- `std::deque< const state * >` `todo`  
*A queue of states yet to explore.*
- `const evtgba * automata_`  
*The [spot::evtgba](#) to explore.*
- `seen\_map seen`  
*States already seen.*

### 12.42.1 Detailed Description

An implementation of [spot::evtgba\\_reachable\\_iterator](#) that browses states breadth first.

### 12.42.2 Member Typedef Documentation

**12.42.2.1** `typedef Sgi::hash_map<const state*, int, state_ptr_hash, state_ptr_equal>`  
`spot::evtgba_reachable_iterator::seen_map` [protected, inherited]

### 12.42.3 Constructor & Destructor Documentation

**12.42.3.1** `spot::evtgba_reachable_iterator_breadth_first::evtgba_reachable_iterator_breadth_first (const evtgba * a)`

### 12.42.4 Member Function Documentation

**12.42.4.1** `virtual void spot::evtgba_reachable_iterator_breadth_first::add_state (const state * s)`  
[virtual]

Implements [spot::evtgba\\_reachable\\_iterator](#).

**12.42.4.2** `virtual const state* spot::evtgba_reachable_iterator_breadth_first::next_state ()`  
[virtual]

Called by [run\(\)](#) to obtain the.

Implements [spot::evtgba\\_reachable\\_iterator](#).

**12.42.4.3** `void spot::evtgba_reachable_iterator::run ()` [inherited]

Iterate over all reachable states of a [spot::evtgba](#).

This is a template method that will call [add\\_state\(\)](#), [next\\_state\(\)](#), [start\(\)](#), [end\(\)](#), [process\\_state\(\)](#), and [process\\_link\(\)](#), while it iterate over [state](#).

**12.42.4.4** `virtual void spot::evtgba_reachable_iterator::start (int n)` [virtual, inherited]

Called by [run\(\)](#) before starting its iteration.

**Parameters:**

*n* The number of initial states.

**12.42.4.5 virtual void spot::evtgba\_reachable\_iterator::end ()** [virtual, inherited]

Called by [run\(\)](#) once all states have been explored.

**12.42.4.6 virtual void spot::evtgba\_reachable\_iterator::process\_state (const state \* s, int n, evtgba\_iterator \* si)** [virtual, inherited]

Called by [run\(\)](#) to process a [state](#).

**Parameters:**

*s* The current [state](#).

*n* An unique number assigned to *s*.

*si* The [spot::evtgba\\_iterator](#) for *s*.

**12.42.4.7 virtual void spot::evtgba\_reachable\_iterator::process\_link (int in, int out, const evtgba\_iterator \* si)** [virtual, inherited]

Called by [run\(\)](#) to process a transition.

**Parameters:**

*in* The source [state](#) number.

*out* The destination [state](#) number.

*si* The [spot::evtgba\\_iterator](#) positionned on the current transition.

**12.42.5 Member Data Documentation****12.42.5.1 std::deque<const state\*> spot::evtgba\_reachable\_iterator\_breadth\_first::todo** [protected]

A queue of states yet to explore.

**12.42.5.2 const evtgba\* spot::evtgba\_reachable\_iterator::automata\_** [protected, inherited]

The [spot::evtgba](#) to explore.

**12.42.5.3 seen\_map spot::evtgba\_reachable\_iterator::seen** [protected, inherited]

States already seen.

The documentation for this class was generated from the following file:

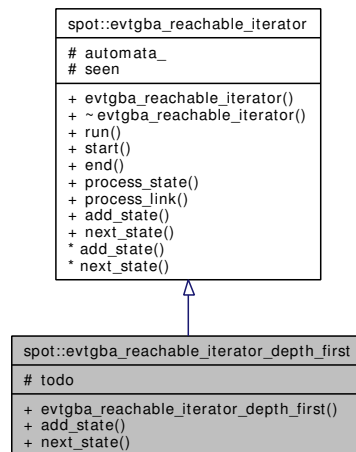
- [evtgbaalgorithms/reachiter.hh](#)

## 12.43 spot::evtgba\_reachable\_iterator\_depth\_first Class Reference

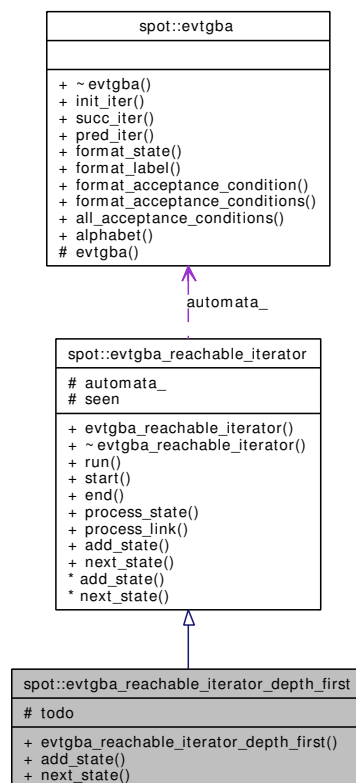
An implementation of [spot::evtgba\\_reachable\\_iterator](#) that browses states depth first.

```
#include <evtgbaalgs/reachiter.hh>
```

Inheritance diagram for spot::evtgba\_reachable\_iterator\_depth\_first:



Collaboration diagram for spot::evtgba\_reachable\_iterator\_depth\_first:



### Public Member Functions

- [evtgba\\_reachable\\_iterator\\_depth\\_first](#) (const [evtgba](#) \*a)
- virtual void [add\\_state](#) (const [state](#) \*s)
- virtual const [state](#) \* [next\\_state](#) ()  
*Called by [run\(\)](#) to obtain the.*
- void [run](#) ()  
*Iterate over all reachable states of a [spot::evtgba](#).*
- virtual void [start](#) (int n)  
*Called by [run\(\)](#) before starting its iteration.*
- virtual void [end](#) ()  
*Called by [run\(\)](#) once all states have been explored.*
- virtual void [process\\_state](#) (const [state](#) \*s, int n, [evtgba\\_iterator](#) \*si)
- virtual void [process\\_link](#) (int in, int out, const [evtgba\\_iterator](#) \*si)

### Protected Types

- typedef Sgi::hash\_map< const [state](#) \*, int, [state\\_ptr\\_hash](#), [state\\_ptr\\_equal](#) > [seen\\_map](#)

### Protected Attributes

- std::stack< const [state](#) \* > [todo](#)  
*A stack of states yet to explore.*
- const [evtgba](#) \* [automata\\_](#)  
*The [spot::evtgba](#) to explore.*
- [seen\\_map](#) [seen](#)  
*States already seen.*

#### 12.43.1 Detailed Description

An implementation of [spot::evtgba\\_reachable\\_iterator](#) that browses states depth first.

#### 12.43.2 Member Typedef Documentation

**12.43.2.1** typedef Sgi::hash\_map<const state\*, int, state\_ptr\_hash, state\_ptr\_equal>  
spot::evtgba\_reachable\_iterator::seen\_map [protected, inherited]

#### 12.43.3 Constructor & Destructor Documentation

**12.43.3.1** spot::evtgba\_reachable\_iterator\_depth\_first::evtgba\_reachable\_iterator\_depth\_first  
(const [evtgba](#) \* a)

### 12.43.4 Member Function Documentation

**12.43.4.1** `virtual void spot::evtgba_reachable_iterator_depth_first::add_state (const state * s)` [virtual]

Implements [spot::evtgba\\_reachable\\_iterator](#).

**12.43.4.2** `virtual const state* spot::evtgba_reachable_iterator_depth_first::next_state ()` [virtual]

Called by [run\(\)](#) to obtain the.

Implements [spot::evtgba\\_reachable\\_iterator](#).

**12.43.4.3** `void spot::evtgba_reachable_iterator::run ()` [inherited]

Iterate over all reachable states of a [spot::evtgba](#).

This is a template method that will call [add\\_state\(\)](#), [next\\_state\(\)](#), [start\(\)](#), [end\(\)](#), [process\\_state\(\)](#), and [process\\_link\(\)](#), while it iterate over [state](#).

**12.43.4.4** `virtual void spot::evtgba_reachable_iterator::start (int n)` [virtual, inherited]

Called by [run\(\)](#) before starting its iteration.

#### Parameters:

*n* The number of initial states.

**12.43.4.5** `virtual void spot::evtgba_reachable_iterator::end ()` [virtual, inherited]

Called by [run\(\)](#) once all states have been explored.

**12.43.4.6** `virtual void spot::evtgba_reachable_iterator::process_state (const state * s, int n, evtgba_iterator * si)` [virtual, inherited]

Called by [run\(\)](#) to process a [state](#).

#### Parameters:

*s* The current [state](#).

*n* An unique number assigned to *s*.

*si* The [spot::evtgba\\_iterator](#) for *s*.

**12.43.4.7** `virtual void spot::evtgba_reachable_iterator::process_link (int in, int out, const evtgba_iterator * si)` [virtual, inherited]

Called by [run\(\)](#) to process a transition.

#### Parameters:

*in* The source [state](#) number.

*out* The destination [state](#) number.

*si* The [spot::evtgba\\_iterator](#) positionned on the current transition.



### 12.43.5 Member Data Documentation

**12.43.5.1** `std::stack<const state*>` `spot::evtgba_reachable_iterator_depth_first::todo` [protected]

A stack of states yet to explore.

**12.43.5.2** `const evtgba*` `spot::evtgba_reachable_iterator::automata_` [protected, inherited]

The [spot::evtgba](#) to explore.

**12.43.5.3** `seen_map` `spot::evtgba_reachable_iterator::seen` [protected, inherited]

States already seen.

The documentation for this class was generated from the following file:

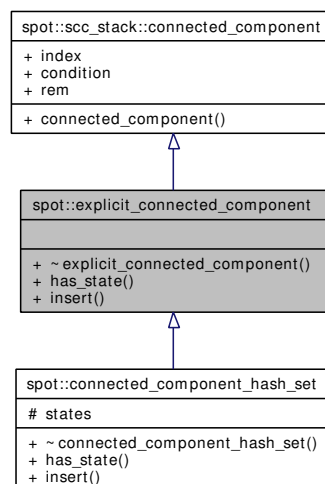
- [evtgbaalgs/reachiter.hh](#)

## 12.44 spot::explicit\_connected\_component Class Reference

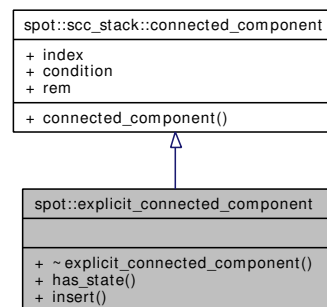
An SCC storing all its states explicitly.

```
#include <tgbaalgs/gtec/explsccl.hh>
```

Inheritance diagram for `spot::explicit_connected_component`:



Collaboration diagram for spot::explicit\_connected\_component:



### Public Member Functions

- virtual [~explicit\\_connected\\_component](#) ()
- virtual const [state](#) \* [has\\_state](#) (const [state](#) \*s) const=0  
*Check if the SCC contains states s.*
- virtual void [insert](#) (const [state](#) \*s)=0  
*Insert a new [state](#) in the SCC.*

### Public Attributes

- int [index](#)  
*Index of the SCC.*
- bdd [condition](#)
- std::list< const [state](#) \* > [rem](#)

#### 12.44.1 Detailed Description

An SCC storing all its states explicitly.

#### 12.44.2 Constructor & Destructor Documentation

**12.44.2.1** virtual `spot::explicit_connected_component::~~explicit_connected_component` ()  
[inline, virtual]

#### 12.44.3 Member Function Documentation

**12.44.3.1** virtual const [state](#)\* `spot::explicit_connected_component::has_state` (const [state](#) \* s) const  
[pure virtual]

Check if the SCC contains states *s*.

Return the representative of *s* in the SCC, and delete *s* if it is different (acting like `numbered_state_-heap::filter`), or 0 otherwise.

Implemented in [spot::connected\\_component\\_hash\\_set](#).

**12.44.3.2** `virtual void spot::explicit_connected_component::insert (const state * s) [pure virtual]`

Insert a new [state](#) in the SCC.

Implemented in [spot::connected\\_component\\_hash\\_set](#).

#### 12.44.4 Member Data Documentation

**12.44.4.1** `int spot::scc_stack::connected_component::index [inherited]`

Index of the SCC.

**12.44.4.2** `bdd spot::scc_stack::connected_component::condition [inherited]`

The bdd condition is the union of all acceptance conditions of transitions which connect the states of the connected component.

**12.44.4.3** `std::list<const state*> spot::scc_stack::connected_component::rem [inherited]`

The documentation for this class was generated from the following file:

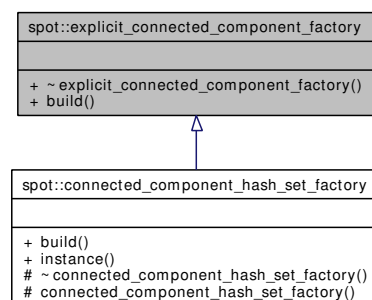
- [tgbaalgos/gtec/explscg.hh](#)

## 12.45 spot::explicit\_connected\_component\_factory Class Reference

Abstract factory for [explicit\\_connected\\_component](#).

```
#include <tgbaalgos/gtec/explscg.hh>
```

Inheritance diagram for `spot::explicit_connected_component_factory`:



#### Public Member Functions

- virtual [~explicit\\_connected\\_component\\_factory](#) ()
- virtual [explicit\\_connected\\_component](#) \* [build](#) () const=0  
Create an [explicit\\_connected\\_component](#).

#### 12.45.1 Detailed Description

Abstract factory for [explicit\\_connected\\_component](#).

## 12.45.2 Constructor & Destructor Documentation

**12.45.2.1** virtual `spot::explicit_connected_component_factory::~explicit_connected_component_factory()` [`inline`, `virtual`]

## 12.45.3 Member Function Documentation

**12.45.3.1** virtual `explicit_connected_component*` `spot::explicit_connected_component_factory::build()` `const` [`pure virtual`]

Create an [explicit\\_connected\\_component](#).

Implemented in [spot::connected\\_component\\_hash\\_set\\_factory](#).

The documentation for this class was generated from the following file:

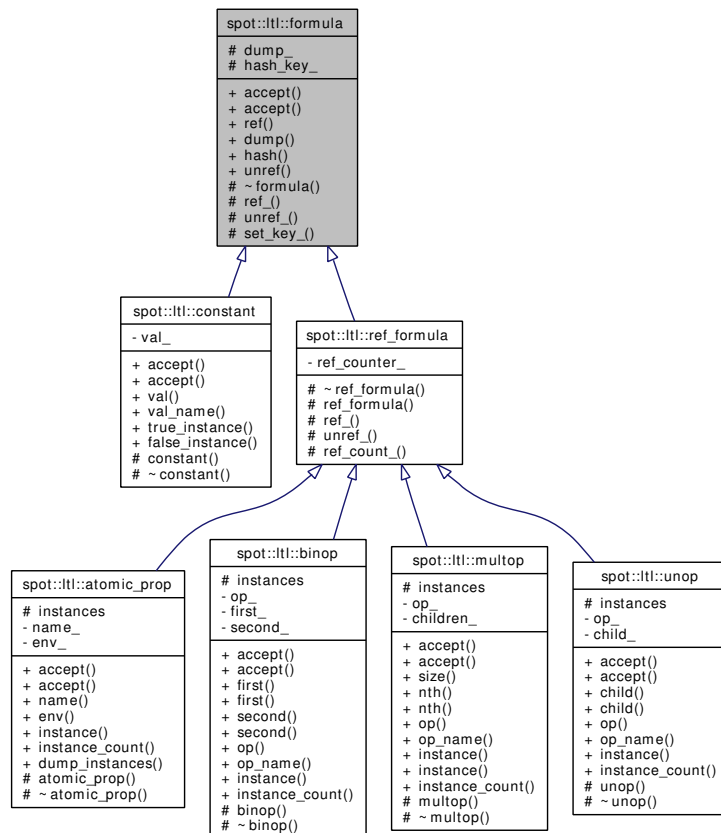
- [tgbaalgorithms/gtec/explsc.h](#)

## 12.46 spot::ltl::formula Class Reference

An LTL [formula](#).

```
#include <ltlast/formula.hh>
```

Inheritance diagram for `spot::ltl::formula`:



### Public Member Functions

- virtual void `accept (visitor &v)=0`  
*Entry point for vspot::ltl::visitor instances.*
- virtual void `accept (const_visitor &v) const=0`  
*Entry point for vspot::ltl::const\_visitor instances.*
- `formula * ref ()`  
*clone this node*
- const std::string & `dump () const`  
*Return a canonic representation of the *formula*.*
- const size\_t `hash () const`  
*Return a hash\_key for the *formula*.*

### Static Public Member Functions

- static void `unref (formula *f)`  
*release this node*

### Protected Member Functions

- virtual `~formula ()`
- virtual void `ref_ ()`  
*increment reference counter if any*
- virtual bool `unref_ ()`  
*decrement reference counter if any, return true when the instance must be deleted (usually when the counter hits 0).*
- void `set_key_ ()`  
*Compute key\_ from dump\_.*

### Protected Attributes

- std::string `dump_`  
*The canonic representation of the *formula*.*
- size\_t `hash_key_`  
*The hash key of this *formula*.*

### 12.46.1 Detailed Description

An LTL [formula](#).

The only way you can work with a [formula](#) is to build a [spot::ltl::visitor](#) or [spot::ltl::const\\_visitor](#).

### 12.46.2 Constructor & Destructor Documentation

**12.46.2.1** `virtual spot::ltl::formula::~~formula ()` `[protected, virtual]`

### 12.46.3 Member Function Documentation

**12.46.3.1** `virtual void spot::ltl::formula::accept (visitor & v)` `[pure virtual]`

Entry point for `vspot::ltl::visitor` instances.

Implemented in [spot::ltl::atomic\\_prop](#), [spot::ltl::binop](#), [spot::ltl::constant](#), [spot::ltl::multop](#), and [spot::ltl::unop](#).

**12.46.3.2** `virtual void spot::ltl::formula::accept (const_visitor & v) const` `[pure virtual]`

Entry point for `vspot::ltl::const_visitor` instances.

Implemented in [spot::ltl::atomic\\_prop](#), [spot::ltl::binop](#), [spot::ltl::constant](#), [spot::ltl::multop](#), and [spot::ltl::unop](#).

**12.46.3.3** `formula* spot::ltl::formula::ref ()`

clone this node

This increments the reference counter of this node (if one is used). You should almost never use this method directly as it doesn't touch the children. If you want to clone a whole [formula](#), use [spot::ltl::clone\(\)](#) instead.

**12.46.3.4** `static void spot::ltl::formula::unref (formula * f)` `[static]`

release this node

This decrements the reference counter of this node (if one is used) and can free the object. You should almost never use this method directly as it doesn't touch the children. If you want to release a whole [formula](#), use [spot::ltl::destroy\(\)](#) instead.

**12.46.3.5** `const std::string& spot::ltl::formula::dump () const`

Return a canonic representation of the [formula](#).

**12.46.3.6** `const size_t spot::ltl::formula::hash () const` `[inline]`

Return a hash\_key for the [formula](#).

**12.46.3.7** `virtual void spot::ltl::formula::ref_ ()` `[protected, virtual]`

increment reference counter if any

Reimplemented in [spot::ltl::ref\\_formula](#).

**12.46.3.8** `virtual bool spot::ltl::formula::unref_()` `[protected, virtual]`

decrement reference counter if any, return true when the instance must be deleted (usually when the counter hits 0).

Reimplemented in [spot::ltl::ref\\_formula](#).

**12.46.3.9** `void spot::ltl::formula::set_key_()` `[protected]`

Compute `key_` from `dump_`.

Should be called once in each object, after `dump_` has been set.

**12.46.4** Member Data Documentation**12.46.4.1** `std::string spot::ltl::formula::dump_` `[protected]`

The canonic representation of the [formula](#).

**12.46.4.2** `size_t spot::ltl::formula::hash_key_` `[protected]`

The hash key of this [formula](#).

Initialized by [set\\_key\\_\(\)](#).

The documentation for this class was generated from the following file:

- [ltlast/formula.hh](#)

**12.47** `spot::ltl::formula_ptr_hash` Struct Reference

Hash Function for `const formula*`.

```
#include <ltlast/formula.hh>
```

**Public Member Functions**

- `size_t operator()` (`const formula *that`) `const`

**12.47.1** Detailed Description

Hash Function for `const formula*`.

This is meant to be used as a hash functor for Sgi's `hash_map` whose key are of type `const formula*`.

For instance here is how one could declare a map of `const :: formula*`.

```
// Remember how many times each formula has been seen.
Sgi::hash_map<const spot::ltl::formula*, int,
              const spot::ltl::formula_ptr_hash> seen;
```

**12.47.2** Member Function Documentation**12.47.2.1** `size_t spot::ltl::formula_ptr_hash::operator()` (`const formula * that`) `const` `[inline]`

The documentation for this struct was generated from the following file:

- [ltlast/formula.hh](#)

## 12.48 `spot::ltl::formula_ptr_less_than` Struct Reference

Strict Weak Ordering for `const formula*`.

```
#include <ltlast/formula.hh>
```

### Public Member Functions

- `bool operator()` (`const formula *left`, `const formula *right`) `const`

#### 12.48.1 Detailed Description

Strict Weak Ordering for `const formula*`.

This is meant to be used as a comparison functor for STL map whose key are of type `const formula*`.

For instance here is how one could declare a map of `const :: formula*`.

```
// Remember how many times each formula has been seen.
std::map<const spot::ltl::formula*, int,
        spot::formula_ptr_less_than> seen;
```

#### 12.48.2 Member Function Documentation

**12.48.2.1** `bool spot::ltl::formula_ptr_less_than::operator()` (`const formula * left`, `const formula * right`) `const` [`inline`]

The documentation for this struct was generated from the following file:

- [ltlast/formula.hh](#)

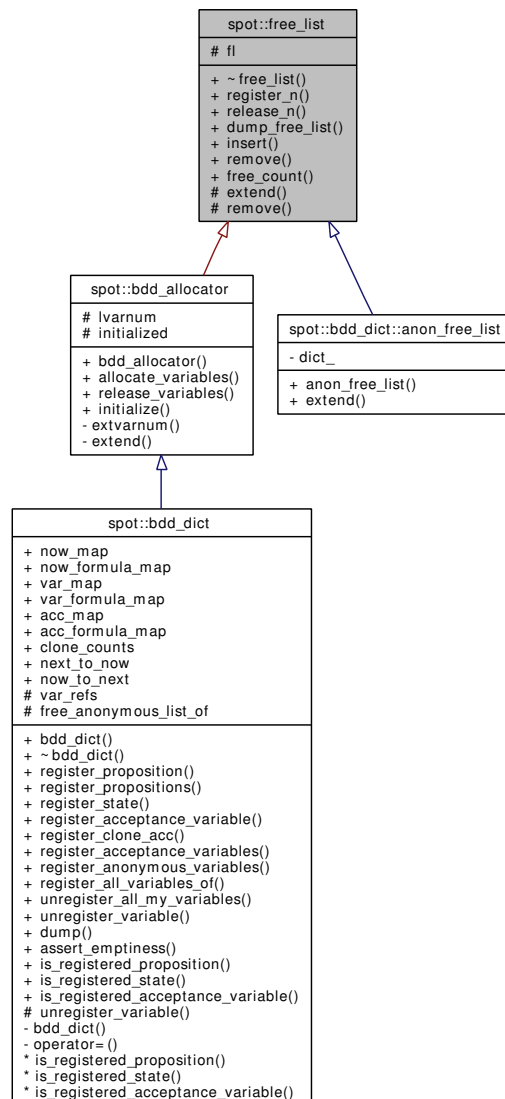
## 12.49 `spot::free_list` Class Reference

Manage list of free integers.

```
#include <misc/freelist.hh>
```



Inheritance diagram for spot::free\_list:



## Public Member Functions

- virtual [~free\\_list](#) ()
- int [register\\_n](#) (int n)  
*Find n consecutive integers.*
- void [release\\_n](#) (int base, int n)  
*Release n consecutive integers starting at base.*
- std::ostream & [dump\\_free\\_list](#) (std::ostream &os) const  
*Dump the list to os for debugging.*
- void [insert](#) (int base, int n)  
*Extend the list by inserting a new pos-length pair.*

- void `remove` (int base, int n=0)  
*Remove n consecutive entries from the list, starting at base.*
- int `free_count` () const  
*Return the number of free integers on the list.*

### Protected Types

- typedef std::pair< int, int > `pos_lenght_pair`  
*Such pairs describe second free integer starting at first.*
- typedef std::list< `pos_lenght_pair` > `free_list_type`

### Protected Member Functions

- virtual int `extend` (int n)=0
- void `remove` (free\_list\_type::iterator i, int base, int n)  
*Remove n consecutive entries from the list, starting at base.*

### Protected Attributes

- `free_list_type` fl  
*Tracks unused BDD variables.*

## 12.49.1 Detailed Description

Manage list of free integers.

## 12.49.2 Member Typedef Documentation

### 12.49.2.1 typedef std::pair<int, int> spot::free\_list::pos\_lenght\_pair [protected]

Such pairs describe second free integer starting at first.

### 12.49.2.2 typedef std::list<pos\_lenght\_pair> spot::free\_list::free\_list\_type [protected]

## 12.49.3 Constructor & Destructor Documentation

### 12.49.3.1 virtual spot::free\_list::~~free\_list () [virtual]

## 12.49.4 Member Function Documentation

### 12.49.4.1 `int spot::free_list::register_n (int n)`

Find *n* consecutive integers.

Browse the list of free integers until *n* consecutive integers are found. Extend the list (using [extend\(\)](#)) otherwise.

#### Returns:

the first integer of the range

### 12.49.4.2 `void spot::free_list::release_n (int base, int n)`

Release *n* consecutive integers starting at *base*.

### 12.49.4.3 `std::ostream& spot::free_list::dump_free_list (std::ostream & os) const`

Dump the list to *os* for debugging.

### 12.49.4.4 `void spot::free_list::insert (int base, int n)`

Extend the list by inserting a new pos-length pair.

### 12.49.4.5 `void spot::free_list::remove (int base, int n = 0)`

Remove *n* consecutive entries from the list, starting at *base*.

### 12.49.4.6 `int spot::free_list::free_count () const`

Return the number of free integers on the list.

### 12.49.4.7 `virtual int spot::free_list::extend (int n)` `[protected, pure virtual]`

Allocate *n* integer.

This function is called by [register\\_n\(\)](#) when the free list is empty or if *n* consecutive integers could not be found. It should allocate more integers, possibly changing the list, and return the first integer on a range of *n* consecutive integer requested by the user.

Implemented in [spot::bdd\\_allocator](#), and [spot::bdd\\_dict::anon\\_free\\_list](#).

### 12.49.4.8 `void spot::free_list::remove (free_list_type::iterator i, int base, int n)` `[protected]`

Remove *n* consecutive entries from the list, starting at *base*.

## 12.49.5 Member Data Documentation

### 12.49.5.1 `free_list_type spot::free_list::fl` `[protected]`

Tracks unused BDD variables.

The documentation for this class was generated from the following file:

- [misc/freelist.hh](#)

## 12.50 `spot::gspn_exception` Class Reference

An exception used to forward GSPN errors.

```
#include <gspn/common.hh>
```

### Public Member Functions

- [gspn\\_exception](#) (const std::string &where, int err)
- int [get\\_err](#) () const
- std::string [get\\_where](#) () const

### Private Attributes

- int [err\\_](#)
- std::string [where\\_](#)

#### 12.50.1 Detailed Description

An exception used to forward GSPN errors.

#### 12.50.2 Constructor & Destructor Documentation

**12.50.2.1** `spot::gspn_exception::gspn_exception (const std::string & where, int err)` [inline]

#### 12.50.3 Member Function Documentation

**12.50.3.1** `int spot::gspn_exception::get_err () const` [inline]

**12.50.3.2** `std::string spot::gspn_exception::get_where () const` [inline]

#### 12.50.4 Member Data Documentation

**12.50.4.1** `int spot::gspn_exception::err_` [private]

**12.50.4.2** `std::string spot::gspn_exception::where_` [private]

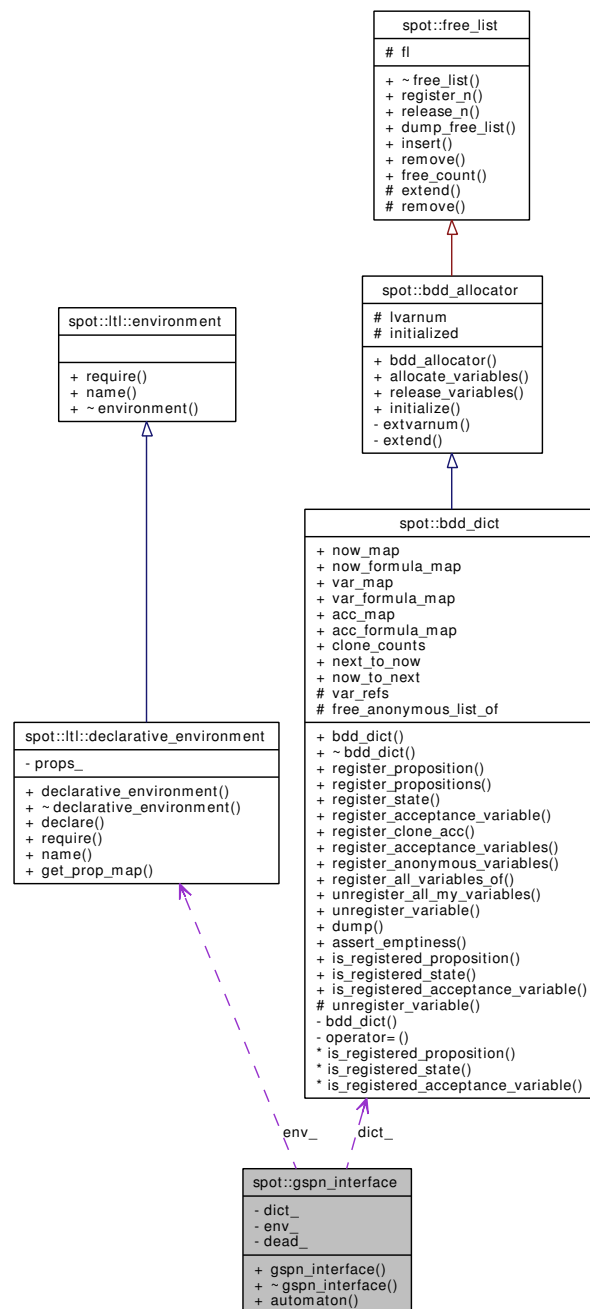
The documentation for this class was generated from the following file:

- [gspn/common.hh](#)

## 12.51 `spot::gspn_interface` Class Reference

```
#include <gspn/gspn.hh>
```

Collaboration diagram for spot::gspn\_interface:



## Public Member Functions

- [gspn\\_interface](#) (int argc, char \*\*argv, [bdd\\_dict](#) \*dict, [ltl::declarative\\_environment](#) &env, const std::string &dead="true")
- [~gspn\\_interface](#) ()
- [tgba](#) \* [automaton](#) () const

### Private Attributes

- [bdd\\_dict](#) \* [dict\\_](#)
- [ltl::declarative\\_environment](#) & [env\\_](#)
- const std::string [dead\\_](#)

### 12.51.1 Constructor & Destructor Documentation

**12.51.1.1** `spot::gspn_interface::gspn_interface (int argc, char ** argv, bdd_dict * dict, ltl::declarative_environment & env, const std::string & dead = "true")`

**12.51.1.2** `spot::gspn_interface::~~gspn_interface ()`

### 12.51.2 Member Function Documentation

**12.51.2.1** `tgba* spot::gspn_interface::automaton () const`

### 12.51.3 Member Data Documentation

**12.51.3.1** `bdd_dict* spot::gspn_interface::dict_ [private]`

**12.51.3.2** `ltl::declarative_environment& spot::gspn_interface::env_ [private]`

**12.51.3.3** `const std::string spot::gspn_interface::dead_ [private]`

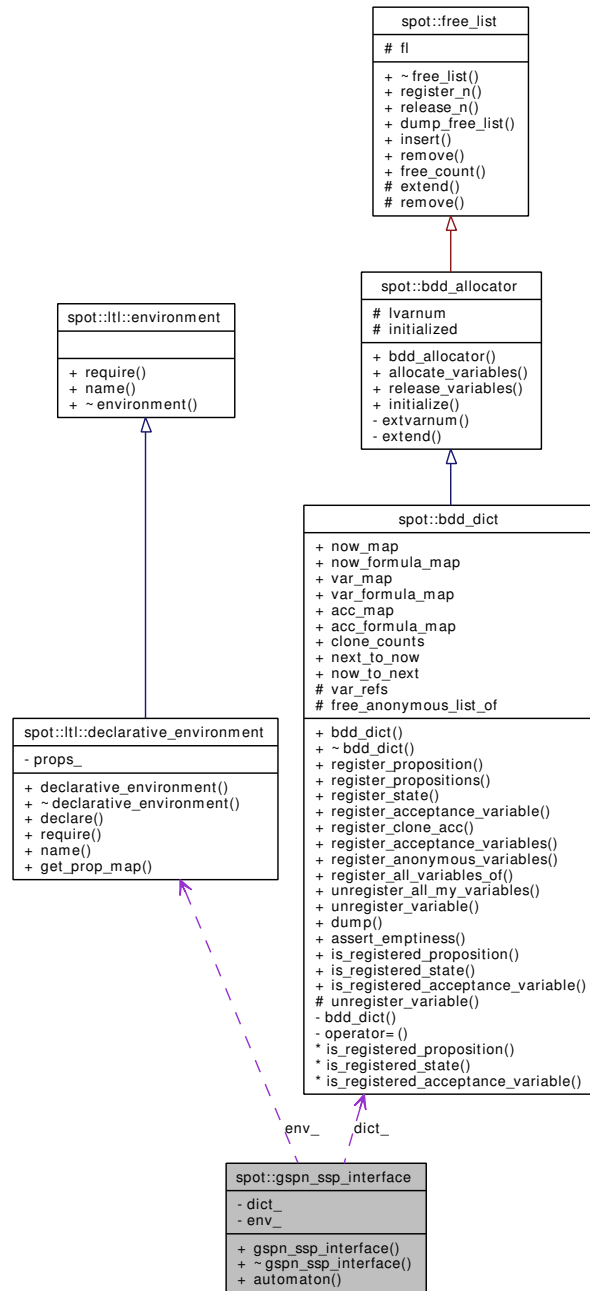
The documentation for this class was generated from the following file:

- [gspn/gspn.hh](#)

## 12.52 spot::gspn\_ssp\_interface Class Reference

```
#include <gspn/ssp.hh>
```

Collaboration diagram for spot::gspn\_ssp\_interface:



### Public Member Functions

- `gspn_ssp_interface` (int argc, char \*\*argv, `bdd_dict` \*dict, const `ltl::declarative_environment` &env, bool inclusion=false, bool doublehash=true)
- `~gspn_ssp_interface` ()
- `tgba * automaton` (const `tgba` \*operand) const

### Private Attributes

- [bdd\\_dict](#) \* [dict\\_](#)
- const [ltl::declarative\\_environment](#) & [env\\_](#)

### 12.52.1 Constructor & Destructor Documentation

**12.52.1.1** `spot::gspn_ssp_interface::gspn_ssp_interface (int argc, char ** argv, bdd\_dict * dict, const ltl::declarative\_environment & env, bool inclusion = false, bool doublehash = true)`

**12.52.1.2** `spot::gspn_ssp_interface::~~gspn_ssp_interface ()`

### 12.52.2 Member Function Documentation

**12.52.2.1** `tgba* spot::gspn_ssp_interface::automaton (const tgba * operand) const`

### 12.52.3 Member Data Documentation

**12.52.3.1** `bdd\_dict* spot::gspn_ssp_interface::dict_ [private]`

**12.52.3.2** `const ltl::declarative\_environment& spot::gspn_ssp_interface::env_ [private]`

The documentation for this class was generated from the following file:

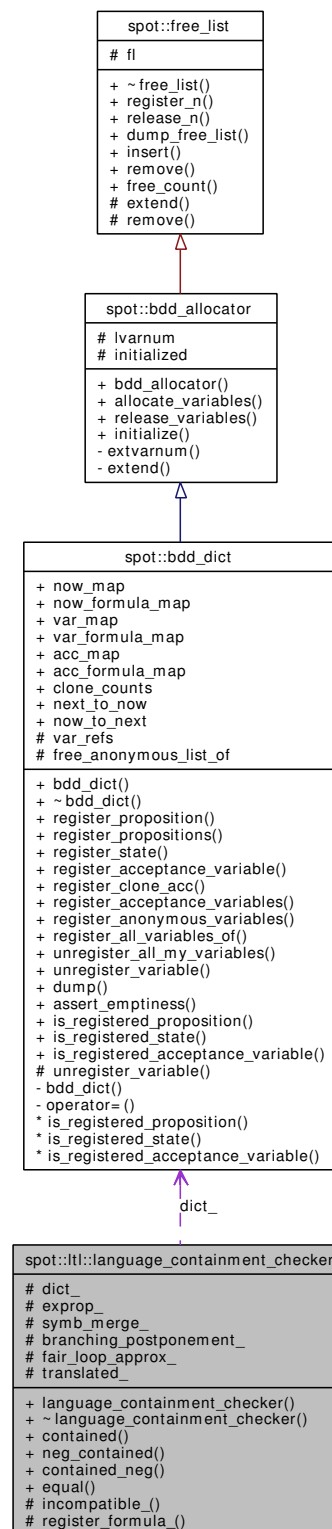
- [gspn/ssp.hh](#)

## 12.53 spot::ltl::language\_containment\_checker Class Reference

```
#include <ltlvisit/contain.hh>
```



Collaboration diagram for spot::ltl::language\_containment\_checker:



### Public Member Functions

- [language\\_containment\\_checker](#) ([bdd\\_dict](#) \*dict, bool exprop, bool symb\_merge, bool branching\_postponement, bool fair\_loop\_approx)
- [~language\\_containment\\_checker](#) ()
- bool [contained](#) (const [formula](#) \*l, const [formula](#) \*g)  
*Check whether  $L(l)$  is a subset of  $L(g)$ .*
- bool [neg\\_contained](#) (const [formula](#) \*l, const [formula](#) \*g)  
*Check whether  $L(!l)$  is a subset of  $L(g)$ .*
- bool [contained\\_neg](#) (const [formula](#) \*l, const [formula](#) \*g)  
*Check whether  $L(l)$  is a subset of  $L(!g)$ .*
- bool [equal](#) (const [formula](#) \*l, const [formula](#) \*g)  
*Check whether  $L(l) = L(g)$ .*

### Protected Member Functions

- bool [incompatible\\_](#) ([record\\_](#) \*l, [record\\_](#) \*g)
- [record\\_](#) \* [register\\_formula\\_](#) (const [formula](#) \*f)

### Protected Attributes

- [bdd\\_dict](#) \* [dict\\_](#)
- bool [exprop\\_](#)
- bool [symb\\_merge\\_](#)
- bool [branching\\_postponement\\_](#)
- bool [fair\\_loop\\_approx\\_](#)
- [trans\\_map](#) [translated\\_](#)

### Private Types

- typedef Sgi::hash\_map< const [formula](#) \*, [record\\_](#), [formula\\_ptr\\_hash](#) > [trans\\_map](#)

### Classes

- struct [record\\_](#)

#### 12.53.1 Member Typedef Documentation

**12.53.1.1** typedef Sgi::hash\_map<const [formula](#)\*, [record\\_](#), [formula\\_ptr\\_hash](#)>  
spot::ltl::language\_containment\_checker::trans\_map [private]

### 12.53.2 Constructor & Destructor Documentation

**12.53.2.1** `spot::ltl::language_containment_checker::language_containment_checker (bdd_dict * dict, bool exprop, bool symb_merge, bool branching_postponement, bool fair_loop_approx)`

This class uses [spot::ltl\\_to\\_tgba\\_fm](#) to translate LTL formulae. See that class for the meaning of these options.

**12.53.2.2** `spot::ltl::language_containment_checker::~~language_containment_checker ()`

### 12.53.3 Member Function Documentation

**12.53.3.1** `bool spot::ltl::language_containment_checker::contained (const formula * l, const formula * g)`

Check whether  $L(l)$  is a subset of  $L(g)$ .

**12.53.3.2** `bool spot::ltl::language_containment_checker::neg_contained (const formula * l, const formula * g)`

Check whether  $L(!l)$  is a subset of  $L(g)$ .

**12.53.3.3** `bool spot::ltl::language_containment_checker::contained_neg (const formula * l, const formula * g)`

Check whether  $L(l)$  is a subset of  $L(!g)$ .

**12.53.3.4** `bool spot::ltl::language_containment_checker::equal (const formula * l, const formula * g)`

Check whether  $L(l) = L(g)$ .

**12.53.3.5** `bool spot::ltl::language_containment_checker::incompatible_ (record_ * l, record_ * g) [protected]`

**12.53.3.6** `record_ * spot::ltl::language_containment_checker::register_formula_ (const formula * f) [protected]`

### 12.53.4 Member Data Documentation

**12.53.4.1** `bdd_dict* spot::ltl::language_containment_checker::dict_ [protected]`

**12.53.4.2** `bool spot::ltl::language_containment_checker::exprop_ [protected]`

**12.53.4.3** `bool spot::ltl::language_containment_checker::symb_merge_ [protected]`

**12.53.4.4** `bool spot::ltl::language_containment_checker::branching_postponement_ [protected]`

12.53.4.5 bool spot::ltl::language\_containment\_checker::fair\_loop\_approx\_ [protected]

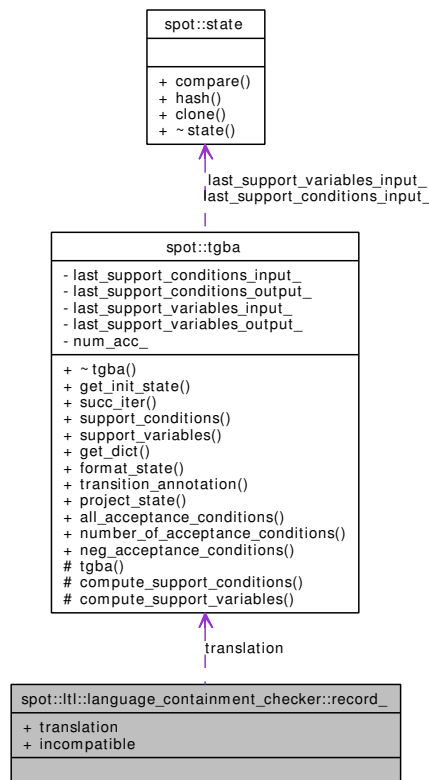
12.53.4.6 trans\_map spot::ltl::language\_containment\_checker::translated\_ [protected]

The documentation for this class was generated from the following file:

- ltlvisit/contain.hh

## 12.54 spot::ltl::language\_containment\_checker::record\_Struct Reference

Collaboration diagram for spot::ltl::language\_containment\_checker::record\_:



### Public Types

- typedef std::map< const [record\\_\\*](#), bool > [incomp\\_map](#)

### Public Attributes

- const [tgba\\*](#) [translation](#)
- [incomp\\_map](#) [incompatible](#)

### 12.54.1 Member Typedef Documentation

12.54.1.1 typedef std::map<const record\_\*, bool> spot::ltl::language\_containment\_checker::record\_::incomp\_map

## 12.54.2 Member Data Documentation

### 12.54.2.1 const tgba\* spot::ltl::language\_containment\_checker::record\_::translation

### 12.54.2.2 incomp\_map spot::ltl::language\_containment\_checker::record\_::incompatible

The documentation for this struct was generated from the following file:

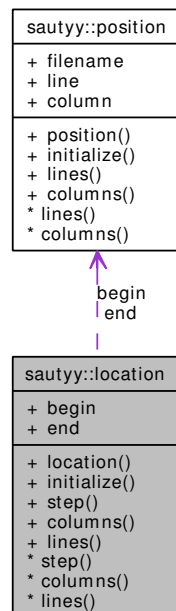
- ltlvisit/[contain.hh](#)

## 12.55 sautyy::location Class Reference

Abstract a [location](#).

```
#include <sautyparse/location.hh>
```

Collaboration diagram for sautyy::location:



## Public Member Functions

- [location](#) ()  
*Construct a [location](#).*
- void [initialize](#) (std::string \*fn)  
*Initialization.*

## Line and Column related manipulators

- void [step](#) ()  
*Reset initial [location](#) to final [location](#).*

- void [columns](#) (unsigned int count=1)  
*Extend the current [location](#) to the COUNT next columns.*
- void [lines](#) (unsigned int count=1)  
*Extend the current [location](#) to the COUNT next lines.*

### Public Attributes

- [position begin](#)  
*Beginning of the located region.*
- [position end](#)  
*End of the located region.*

### 12.55.1 Detailed Description

Abstract a [location](#).

### 12.55.2 Constructor & Destructor Documentation

#### 12.55.2.1 sautyy::location::location () [inline]

Construct a [location](#).

### 12.55.3 Member Function Documentation

#### 12.55.3.1 void sautyy::location::initialize (std::string \*fn) [inline]

Initialization.

#### 12.55.3.2 void sautyy::location::step () [inline]

Reset initial [location](#) to final [location](#).

#### 12.55.3.3 void sautyy::location::columns (unsigned int count = 1) [inline]

Extend the current [location](#) to the COUNT next columns.

#### 12.55.3.4 void sautyy::location::lines (unsigned int count = 1) [inline]

Extend the current [location](#) to the COUNT next lines.

### 12.55.4 Member Data Documentation

#### 12.55.4.1 position sautyy::location::begin

Beginning of the located region.

### 12.55.4.2 position sautty::location::end

End of the located region.

The documentation for this class was generated from the following file:

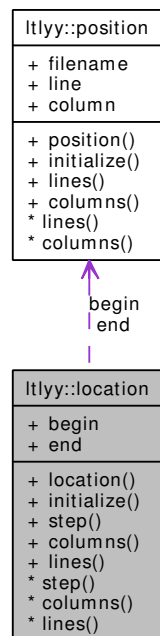
- sautparse/[location.hh](#)

## 12.56 Itlly::location Class Reference

Abstract a [location](#).

```
#include <ltlparse/location.hh>
```

Collaboration diagram for Itlly::location:



### Public Member Functions

- [location](#) ()  
*Construct a [location](#).*
- void [initialize](#) (std::string \*fn)  
*Initialization.*

### Line and Column related manipulators

- void [step](#) ()  
*Reset initial [location](#) to final [location](#).*
- void [columns](#) (unsigned int count=1)  
*Extend the current [location](#) to the COUNT next columns.*

- void [lines](#) (unsigned int count=1)  
*Extend the current [location](#) to the COUNT next lines.*

### Public Attributes

- [position begin](#)  
*Beginning of the located region.*
- [position end](#)  
*End of the located region.*

## 12.56.1 Detailed Description

Abstract a [location](#).

## 12.56.2 Constructor & Destructor Documentation

### 12.56.2.1 ltlyy::location::location () [inline]

Construct a [location](#).

## 12.56.3 Member Function Documentation

### 12.56.3.1 void ltlyy::location::initialize (std::string \*fn) [inline]

Initialization.

### 12.56.3.2 void ltlyy::location::step () [inline]

Reset initial [location](#) to final [location](#).

### 12.56.3.3 void ltlyy::location::columns (unsigned int count = 1) [inline]

Extend the current [location](#) to the COUNT next columns.

### 12.56.3.4 void ltlyy::location::lines (unsigned int count = 1) [inline]

Extend the current [location](#) to the COUNT next lines.

## 12.56.4 Member Data Documentation

### 12.56.4.1 position ltlyy::location::begin

Beginning of the located region.



### 12.56.4.2 position ltlyy::location::end

End of the located region.

The documentation for this class was generated from the following file:

- [ltlparse/location.hh](#)

## 12.57 spot::loopless\_modular\_mixed\_radix\_gray\_code Class Reference

Loopless modular mixed radix Gray code iteration.

```
#include <misc/modgray.hh>
```

### Public Member Functions

- [loopless\\_modular\\_mixed\\_radix\\_gray\\_code](#) (int n)
- virtual [~loopless\\_modular\\_mixed\\_radix\\_gray\\_code](#) ()

#### iteration over an element in a tuple

*The class does not know how to modify the elements of the tuple (Knuth's  $a_j$ 's). These changes are therefore abstracted using the `a_first()`, `a_next()`, and `a_last()` abstract functions. These need to be implemented in subclasses as appropriate.*

- virtual void [a\\_first](#) (int j)=0  
*Reset  $a_j$  to its initial value.*
- virtual void [a\\_next](#) (int j)=0  
*Advance  $a_j$  to its next value.*
- virtual bool [a\\_last](#) (int j) const=0  
*Whether  $a_j$  is on its last value.*

#### iteration over all the tuples

- void [first](#) ()  
*Reset the iteration to the first tuple.*
- bool [last](#) () const  
*Whether this the last tuple.*
- bool [done](#) () const  
*Whether all tuple have been explored.*
- int [next](#) ()  
*Update one item of the tuple and return its position.*

### Protected Attributes

- int [n\\_](#)
- bool [done\\_](#)

- int \* [a\\_](#)
- int \* [f\\_](#)
- int \* [m\\_](#)
- int \* [s\\_](#)
- int \* [non\\_one\\_radixes\\_](#)

### 12.57.1 Detailed Description

Loopless modular mixed radix Gray code iteration.

This class is based on the loopless modular mixed radix gray code algorithm described in exercise 77 of "The Art of Computer Programming", Pre-Fascicle 2A (Draft of section 7.2.1.1: generating all n-tuples) by Donald E. Knuth.

The idea is to enumerate the set of all n-tuples  $(a_0, a_1, \dots, a_{n-1})$  where each  $a_j$  range over a distinct set (this is the *mixed radix* part), so that only one  $a_j$  changes between two successive tuples of the iteration (that is the *Gray code* part), and that this changes occurs always in the same direction, cycling over the set  $a_j$  must cover (i.e., *modular*). The algorithm is *loopless* in that computing the next tuple done without any loop, i.e., in constant time.

This class does not need to know the type of the  $a_j$ , it will handle them indirectly through three methods: [a\\_first\(\)](#), [a\\_next\(\)](#), and [a\\_last\(\)](#). These methods need to be implemented in a subclass for the particular type of  $a_j$  at hand.

The class itself offers four functions to control the iteration over the set of all the  $(a_0, a_1, \dots, a_{n-1})$  tuples: [first\(\)](#), [next\(\)](#), [last\(\)](#), and [done\(\)](#). These functions are usually used as follows:

```
for (g.first(); !g.done(); g.next())
    use the tuple
```

How to use the tuple of course depends on the way it as been stored in the subclass.

Finally, let's mention two differences between this algorithm and the one in Knuth's book. This version of the algorithm does not need to know the radixes (i.e., the size of set of each  $a_j$ ) beforehand: it will discover them on-the-fly when [a\\_last\(j\)](#) first return true. It will also work with  $a_j$  that cannot be changed. (This is achieved by reindexing the elements through [non\\_one\\_radixes\\_](#), to consider only the elements with a non-singleton range.)

### 12.57.2 Constructor & Destructor Documentation

#### 12.57.2.1 spot::loopless\_modular\_mixed\_radix\_gray\_code::loopless\_modular\_mixed\_radix\_gray\_code(int n)

Constructor.

##### Parameters:

- n** The size of the tuples to enumerate.

#### 12.57.2.2 virtual spot::loopless\_modular\_mixed\_radix\_gray\_code::~~loopless\_modular\_mixed\_radix\_gray\_code() [virtual]

### 12.57.3 Member Function Documentation

**12.57.3.1** `virtual void spot::loopless_modular_mixed_radix_gray_code::a_first (int j) [pure virtual]`

Reset  $a_j$  to its initial value.

**12.57.3.2** `virtual void spot::loopless_modular_mixed_radix_gray_code::a_next (int j) [pure virtual]`

Advance  $a_j$  to its next value.

This will never be called if `a_last(j)` is true.

**12.57.3.3** `virtual bool spot::loopless_modular_mixed_radix_gray_code::a_last (int j) const [pure virtual]`

Whether  $a_j$  is on its last value.

**12.57.3.4** `void spot::loopless_modular_mixed_radix_gray_code::first ()`

Reset the iteration to the first tuple.

This must be called before calling any of `next()`, `last()`, or `done()`.

**12.57.3.5** `bool spot::loopless_modular_mixed_radix_gray_code::last () const [inline]`

Whether this the last tuple.

At this point it is still OK to call `next()`, and then `done()` will become true.

**12.57.3.6** `bool spot::loopless_modular_mixed_radix_gray_code::done () const [inline]`

Whether all tuple have been explored.

**12.57.3.7** `int spot::loopless_modular_mixed_radix_gray_code::next ()`

Update one item of the tuple and return its position.

`next()` should never be called if `done()` is true. If it is called on the last tuple (i.e., `last()` is true), it will return -1. Otherwise it will update one  $a_j$  of the tuple through one the  $a_j$  handling functions, and return  $j$ .

### 12.57.4 Member Data Documentation

**12.57.4.1** `int spot::loopless_modular_mixed_radix_gray_code::n_ [protected]`

**12.57.4.2** `bool spot::loopless_modular_mixed_radix_gray_code::done_ [protected]`

**12.57.4.3** `int* spot::loopless_modular_mixed_radix_gray_code::a_ [protected]`

**12.57.4.4** `int* spot::loopless_modular_mixed_radix_gray_code::f_ [protected]`

12.57.4.5 `int* spot::loopless_modular_mixed_radix_gray_code::m_` [protected]

12.57.4.6 `int* spot::loopless_modular_mixed_radix_gray_code::s_` [protected]

12.57.4.7 `int* spot::loopless_modular_mixed_radix_gray_code::non_one_radixes_` [protected]

The documentation for this class was generated from the following file:

- [misc/modgray.hh](#)

## 12.58 spot::minato\_isop Class Reference

Generate an irredundant sum-of-products (ISOP) form of a BDD function.

```
#include <misc/minato.hh>
```

### Public Member Functions

- [minato\\_isop](#) (bdd input)  
*Constructor.*
- [minato\\_isop](#) (bdd input, bdd vars)  
*Constructor.*
- `bdd next ()`  
*Compute the next sum term of the ISOP form. Return bddfalses when all terms have been output.*

### Private Attributes

- `std::stack< local_vars > todo_`
- `std::stack< bdd > cube_`
- `bdd ret_`

### Classes

- `struct local_vars`  
*Internal variables for [minato\\_isop](#).*

### 12.58.1 Detailed Description

Generate an irredundant sum-of-products (ISOP) form of a BDD function.

This algorithm implements a derecursived version the Minato-Morreale algorithm presented in the following paper.

```

/// @InProceedings{ minato.92.sasimi,
///   author      = {Shin-ichi Minato},
///   title       = {Fast Generation of Irredundant Sum-of-Products Forms
///                 from Binary Decision Diagrams},
///   booktitle    = {Proceedings of the third Synthesis and Simulation
///                 and Meeting International Interchange workshop
///                 (SASIMI'92)},
///   pages       = {64--73},
///   year        = {1992},
///   address     = {Kobe, Japan},
///   month       = {April}
/// }
///

```

## 12.58.2 Constructor & Destructor Documentation

### 12.58.2.1 spot::minato\_isop::minato\_isop (bdd *input*)

Constructor.

- *input* The BDD function to translate in ISOP.

### 12.58.2.2 spot::minato\_isop::minato\_isop (bdd *input*, bdd *vars*)

Constructor.

- *input* The BDD function to translate in ISOP.
- *vars* The set of BDD variables to factorize in *input*.

## 12.58.3 Member Function Documentation

### 12.58.3.1 bdd spot::minato\_isop::next ()

Compute the next sum term of the ISOP form. Return `bddfalse` when all terms have been output.

## 12.58.4 Member Data Documentation

### 12.58.4.1 std::stack<local\_vars> spot::minato\_isop::todo\_ [private]

### 12.58.4.2 std::stack<bdd> spot::minato\_isop::cube\_ [private]

### 12.58.4.3 bdd spot::minato\_isop::ret\_ [private]

The documentation for this class was generated from the following file:

- [misc/minato.hh](#)

## 12.59 spot::minato\_isop::local\_vars Struct Reference

Internal variables for [minato\\_isop](#).

## Public Types

- enum { [FirstStep](#), [SecondStep](#), [ThirdStep](#), [FourthStep](#) }

## Public Member Functions

- [local\\_vars](#) (bdd [f\\_min](#), bdd [f\\_max](#), bdd [vars](#))

## Public Attributes

- bdd [f\\_min](#)
- bdd [f\\_max](#)
- enum spot::minato\_isop::local\_vars:: { ... } [step](#)
- bdd [vars](#)
- bdd [v1](#)
- bdd [f0\\_min](#)
- bdd [f0\\_max](#)
- bdd [f1\\_min](#)
- bdd [f1\\_max](#)
- bdd [g0](#)
- bdd [g1](#)

### 12.59.1 Detailed Description

Internal variables for [minato\\_isop](#).

### 12.59.2 Member Enumeration Documentation

#### 12.59.2.1 anonymous enum

Enumerator:

*FirstStep*  
*SecondStep*  
*ThirdStep*  
*FourthStep*

### 12.59.3 Constructor & Destructor Documentation

#### 12.59.3.1 spot::minato\_isop::local\_vars::local\_vars (bdd *f\_min*, bdd *f\_max*, bdd *vars*) [inline]

### 12.59.4 Member Data Documentation

#### 12.59.4.1 bdd spot::minato\_isop::local\_vars::f\_min

#### 12.59.4.2 bdd spot::minato\_isop::local\_vars::f\_max

**12.59.4.3** enum { ... } spot::minato\_isop::local\_vars::step

**12.59.4.4** bdd spot::minato\_isop::local\_vars::vars

**12.59.4.5** bdd spot::minato\_isop::local\_vars::v1

**12.59.4.6** bdd spot::minato\_isop::local\_vars::f0\_min

**12.59.4.7** bdd spot::minato\_isop::local\_vars::f0\_max

**12.59.4.8** bdd spot::minato\_isop::local\_vars::f1\_min

**12.59.4.9** bdd spot::minato\_isop::local\_vars::f1\_max

**12.59.4.10** bdd spot::minato\_isop::local\_vars::g0

**12.59.4.11** bdd spot::minato\_isop::local\_vars::g1

The documentation for this struct was generated from the following file:

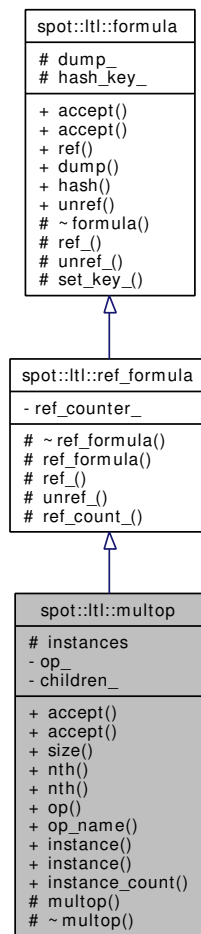
- misc/[minato.hh](#)

## 12.60 spot::lfl::multop Class Reference

Multi-operand operators.

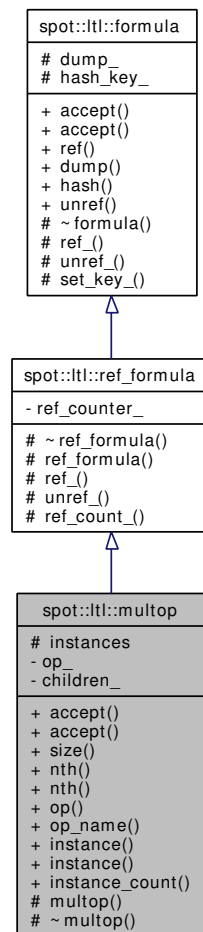
```
#include <ltlast/multop.hh>
```

Inheritance diagram for spot::ltl::multop:





Collaboration diagram for spot::ltl::multop:



## Public Types

- enum `type` { `Or`, `And` }
- typedef `std::vector< formula * >` `vec`

*List of formulae.*

## Public Member Functions

- virtual void `accept (visitor &v)`  
*Entry point for `vspot::ltl::visitor` instances.*
- virtual void `accept (const_visitor &v) const`  
*Entry point for `vspot::ltl::const_visitor` instances.*
- unsigned `size () const`  
*Get the number of children.*

- `const formula * nth (unsigned n) const`  
*Get the  $n$ th children.*
- `formula * nth (unsigned n)`  
*Get the  $n$ th children.*
- `type op () const`  
*Get the type of this operator.*
- `const char * op_name () const`  
*Get the type of this operator, as a string.*
- `formula * ref ()`  
*clone this node*
- `const std::string & dump () const`  
*Return a canonic representation of the `formula`.*
- `const size_t hash () const`  
*Return a `hash_key` for the `formula`.*

### Static Public Member Functions

- `static formula * instance (type op, formula *first, formula *second)`  
*Build a `spot::ltl::multop` with two children.*
- `static formula * instance (type op, vec *v)`  
*Build a `spot::ltl::multop` with many children.*
- `static unsigned instance_count ()`  
*Number of instantiated multi-operand operators. For debugging.*
- `static void unref (formula *f)`  
*release this node*

### Protected Types

- `typedef std::pair< type, vec * > pair`
- `typedef std::map< pair, formula *, paircmp > map`

### Protected Member Functions

- `multop (type op, vec *v)`
- `virtual ~multop ()`
- `void ref_ ()`  
*increment reference counter if any*

- bool `unref_()`  
*decrement reference counter if any, return true when the instance must be deleted (usually when the counter hits 0).*
- unsigned `ref_count_()`  
*Number of references to this *formula*.*
- void `set_key_()`  
*Compute key\_ from dump\_.*

### Protected Attributes

- std::string `dump_`  
*The canonic representation of the *formula*.*
- size\_t `hash_key_`  
*The hash key of this *formula*.*

### Static Protected Attributes

- static `map instances`

### Private Attributes

- `type op_`
- `vec * children_`

### Classes

- struct `paircmp`  
*Comparison functor used internally by *ltl::multop*.*

## 12.60.1 Detailed Description

Multi-operand operators.

These operators are considered commutative and associative.

## 12.60.2 Member Typedef Documentation

### 12.60.2.1 `typedef std::vector<formula*> spot::ltl::multop::vec`

List of formulae.

### 12.60.2.2 `typedef std::pair<type, vec*> spot::ltl::multop::pair` [protected]

**12.60.2.3** `typedef std::map<pair, formula*, pairemp> spot::ltl::multop::map` `[protected]`

### 12.60.3 Member Enumeration Documentation

**12.60.3.1** `enum spot::ltl::multop::type`

Enumerator:

*Or*

*And*

### 12.60.4 Constructor & Destructor Documentation

**12.60.4.1** `spot::ltl::multop::multop (type op, vec * v)` `[protected]`

**12.60.4.2** `virtual spot::ltl::multop::~~multop ()` `[protected, virtual]`

### 12.60.5 Member Function Documentation

**12.60.5.1** `static formula* spot::ltl::multop::instance (type op, formula * first, formula * second)`  
`[static]`

Build a `spot::ltl::multop` with two children.

If one of the children itself is a `spot::ltl::multop` with the same type, it will be merged. I.e., children if that child will be added, and that child itself will be destroyed. This allows incremental building of n-ary `ltl::multop`.

This functions can perform slight optimizations and may not return an `ltl::multop` objects. For instance if `first` and `second` are equal, that `formula` is returned as-is.

**12.60.5.2** `static formula* spot::ltl::multop::instance (type op, vec * v)` `[static]`

Build a `spot::ltl::multop` with many children.

Same as the other `instance()` function, but take a vector of `formula` in argument. This vector is acquired by the `spot::ltl::multop` class, the caller should allocate it with `new`, but not use it (especially not destroy it) after it has been passed to `spot::ltl::multop`.

This functions can perform slight optimizations and may not return an `ltl::multop` objects. For instance if the vector contain only one unique element, this `formula` will be returned as-is.

**12.60.5.3** `virtual void spot::ltl::multop::accept (visitor & v)` `[virtual]`

Entry point for `vspot::ltl::visitor` instances.

Implements `spot::ltl::formula`.

**12.60.5.4** `virtual void spot::ltl::multop::accept (const_visitor & v) const` `[virtual]`

Entry point for `vspot::ltl::const_visitor` instances.

Implements `spot::ltl::formula`.

**12.60.5.5** `unsigned spot::ltl::multop::size () const`

Get the number of children.

**12.60.5.6** `const formula* spot::ltl::multop::nth (unsigned n) const`

Get the *n*th children.

Starting with *n* = 0.

**12.60.5.7** `formula* spot::ltl::multop::nth (unsigned n)`

Get the *n*th children.

Starting with *n* = 0.

**12.60.5.8** `type spot::ltl::multop::op () const`

Get the type of this operator.

**12.60.5.9** `const char* spot::ltl::multop::op_name () const`

Get the type of this operator, as a string.

**12.60.5.10** `static unsigned spot::ltl::multop::instance_count ()` [static]

Number of instantiated multi-operand operators. For debugging.

**12.60.5.11** `void spot::ltl::ref_formula::ref_ ()` [protected, virtual, inherited]

increment reference counter if any

Reimplemented from [spot::ltl::formula](#).

**12.60.5.12** `bool spot::ltl::ref_formula::unref_ ()` [protected, virtual, inherited]

decrement reference counter if any, return true when the instance must be deleted (usually when the counter hits 0).

Reimplemented from [spot::ltl::formula](#).

**12.60.5.13** `unsigned spot::ltl::ref_formula::ref_count_ ()` [protected, inherited]

Number of references to this [formula](#).

**12.60.5.14** `formula* spot::ltl::formula::ref ()` [inherited]

clone this node

This increments the reference counter of this node (if one is used). You should almost never use this method directly as it doesn't touch the children. If you want to clone a whole [formula](#), use [spot::ltl::clone\(\)](#) instead.

**12.60.5.15** static void spot::ltl::formula::unref (formula \*f) [static, inherited]

release this node

This decrements the reference counter of this node (if one is used) and can free the object. You should almost never use this method directly as it doesn't touch the children. If you want to release a whole [formula](#), use [spot::ltl::destroy\(\)](#) instead.

**12.60.5.16** const std::string& spot::ltl::formula::dump () const [inherited]

Return a canonic representation of the [formula](#).

**12.60.5.17** const size\_t spot::ltl::formula::hash () const [inline, inherited]

Return a hash\_key for the [formula](#).

**12.60.5.18** void spot::ltl::formula::set\_key\_ () [protected, inherited]

Compute key\_ from dump\_.

Should be called once in each object, after dump\_ has been set.

## 12.60.6 Member Data Documentation

**12.60.6.1** map spot::ltl::multop::instances [static, protected]

**12.60.6.2** type spot::ltl::multop::op\_ [private]

**12.60.6.3** vec\* spot::ltl::multop::children\_ [private]

**12.60.6.4** std::string spot::ltl::formula::dump\_ [protected, inherited]

The canonic representation of the [formula](#).

**12.60.6.5** size\_t spot::ltl::formula::hash\_key\_ [protected, inherited]

The hash key of this [formula](#).

Initialized by [set\\_key\\_\(\)](#).

The documentation for this class was generated from the following file:

- [ltlast/multop.hh](#)

## 12.61 spot::ltl::multop::pairemp Struct Reference

Comparison functor used internally by [ltl::multop](#).

```
#include <ltlast/multop.hh>
```

### Public Member Functions

- bool [operator\(\)](#) (const [pair](#) &p1, const [pair](#) &p2) const

### 12.61.1 Detailed Description

Comparison functor used internally by [ltl::multop](#).

### 12.61.2 Member Function Documentation

#### 12.61.2.1 bool spot::ltl::multop::paircmp::operator() (const pair & p1, const pair & p2) const [inline]

The documentation for this struct was generated from the following file:

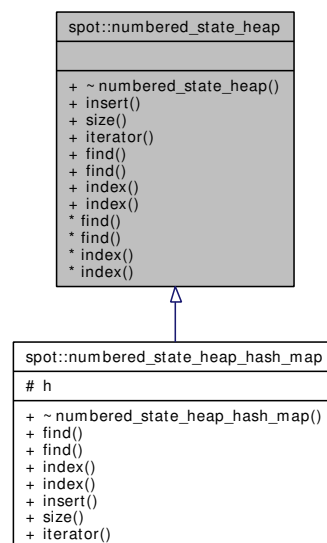
- ltl/multop.hh

## 12.62 spot::numbered\_state\_heap Class Reference

Keep track of a large quantity of indexed states.

```
#include <tgbalgorithms/gtec/nsheap.hh>
```

Inheritance diagram for spot::numbered\_state\_heap:



### Public Types

- typedef std::pair< const [state](#) \*, int \* > [state\\_index\\_p](#)
- typedef std::pair< const [state](#) \*, int > [state\\_index](#)

### Public Member Functions

- virtual [~numbered\\_state\\_heap](#) ()
- virtual void [insert](#) (const [state](#) \*s, int index)=0  
Add a new [state](#) s with index index.
- virtual int [size](#) () const=0

*The number of stored states.*

- virtual [numbered\\_state\\_heap\\_const\\_iterator](#) \* [iterator](#) () const=0

*Return an iterator on the states/indexes pairs.*

- virtual [state\\_index](#) find (const [state](#) \*s) const=0  
*Is [state](#) in the heap?*
- virtual [state\\_index\\_p](#) find (const [state](#) \*s)=0
- virtual [state\\_index](#) index (const [state](#) \*s) const=0  
*Return the index of an existing [state](#).*
- virtual [state\\_index\\_p](#) index (const [state](#) \*s)=0

### 12.62.1 Detailed Description

Keep track of a large quantity of indexed states.

### 12.62.2 Member Typedef Documentation

#### 12.62.2.1 typedef std::pair<const state\*, int\*> spot::numbered\_state\_heap::state\_index\_p

#### 12.62.2.2 typedef std::pair<const state\*, int> spot::numbered\_state\_heap::state\_index

### 12.62.3 Constructor & Destructor Documentation

#### 12.62.3.1 virtual spot::numbered\_state\_heap::~~numbered\_state\_heap () [inline, virtual]

### 12.62.4 Member Function Documentation

#### 12.62.4.1 virtual state\_index spot::numbered\_state\_heap::find (const state \* s) const [pure virtual]

Is [state](#) in the heap?

Returns a pair (0,0) if *s* is not in the heap. or a pair (p, i) if there is a clone *p* of *s* *i* in the heap with index. If *s* is in the heap and is different from *p* it will be freed.

These functions are called by the algorithm to check whether a successor is a new [state](#) to explore or an already visited [state](#).

These functions can be redefined to search for more than an equal match. For example we could redefine it to check [state](#) inclusion.

Implemented in [spot::numbered\\_state\\_heap\\_hash\\_map](#).

#### 12.62.4.2 virtual state\_index\_p spot::numbered\_state\_heap::find (const state \* s) [pure virtual]

Implemented in [spot::numbered\\_state\\_heap\\_hash\\_map](#).



**12.62.4.3** `virtual state_index spot::numbered_state_heap::index (const state * s) const` [pure virtual]

Return the index of an existing [state](#).

This is mostly similar to `find()`, except it will be called for [state](#) which we know are already in the heap, or for [state](#) which may not be in the heap but for which it is always OK to do equality checks.

Implemented in [spot::numbered\\_state\\_heap\\_hash\\_map](#).

**12.62.4.4** `virtual state_index_p spot::numbered_state_heap::index (const state * s)` [pure virtual]

Implemented in [spot::numbered\\_state\\_heap\\_hash\\_map](#).

**12.62.4.5** `virtual void spot::numbered_state_heap::insert (const state * s, int index)` [pure virtual]

Add a new [state](#) *s* with index *index*.

Implemented in [spot::numbered\\_state\\_heap\\_hash\\_map](#).

**12.62.4.6** `virtual int spot::numbered_state_heap::size () const` [pure virtual]

The number of stored states.

Implemented in [spot::numbered\\_state\\_heap\\_hash\\_map](#).

**12.62.4.7** `virtual numbered_state_heap_const_iterator* spot::numbered_state_heap::iterator () const` [pure virtual]

Return an iterator on the states/indexes pairs.

Implemented in [spot::numbered\\_state\\_heap\\_hash\\_map](#).

The documentation for this class was generated from the following file:

- [tgbaalgos/gtec/nsheap.hh](#)

## 12.63 `spot::numbered_state_heap_const_iterator` Class Reference

Iterator on [numbered\\_state\\_heap](#) objects.

```
#include <tgbaalgos/gtec/nsheap.hh>
```

### Public Member Functions

- `virtual ~numbered_state_heap_const_iterator ()`
- `virtual void first ()=0`  
*Iteration.*
- `virtual void next ()=0`
- `virtual bool done () const=0`
- `virtual const state * get_state () const=0`

*Inspection.*

- virtual int [get\\_index](#) () const=0

### 12.63.1 Detailed Description

Iterator on [numbered\\_state\\_heap](#) objects.

### 12.63.2 Constructor & Destructor Documentation

**12.63.2.1** virtual `spot::numbered_state_heap_const_iterator::~~numbered_state_heap_const_iterator ()` `[inline, virtual]`

### 12.63.3 Member Function Documentation

**12.63.3.1** virtual void `spot::numbered_state_heap_const_iterator::first ()` `[pure virtual]`

Iteration.

**12.63.3.2** virtual void `spot::numbered_state_heap_const_iterator::next ()` `[pure virtual]`

**12.63.3.3** virtual `bool spot::numbered_state_heap_const_iterator::done () const` `[pure virtual]`

**12.63.3.4** virtual `const state* spot::numbered_state_heap_const_iterator::get_state () const` `[pure virtual]`

Inspection.

**12.63.3.5** virtual `int spot::numbered_state_heap_const_iterator::get_index () const` `[pure virtual]`

The documentation for this class was generated from the following file:

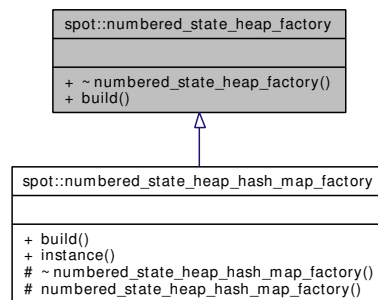
- `tgbaalgos/gtec/nsheap.hh`

## 12.64 spot::numbered\_state\_heap\_factory Class Reference

Abstract factory for [numbered\\_state\\_heap](#).

```
#include <tgbaalgos/gtec/nsheap.hh>
```

Inheritance diagram for spot::numbered\_state\_heap\_factory:



### Public Member Functions

- virtual [~numbered\\_state\\_heap\\_factory](#) ()
- virtual [numbered\\_state\\_heap](#) \* [build](#) () const=0

### 12.64.1 Detailed Description

Abstract factory for [numbered\\_state\\_heap](#).

### 12.64.2 Constructor & Destructor Documentation

**12.64.2.1** virtual `spot::numbered_state_heap_factory::~~numbered_state_heap_factory` ()  
 [inline, virtual]

### 12.64.3 Member Function Documentation

**12.64.3.1** virtual `numbered_state_heap*` `spot::numbered_state_heap_factory::build` () const  
 [pure virtual]

Implemented in [spot::numbered\\_state\\_heap\\_hash\\_map\\_factory](#).

The documentation for this class was generated from the following file:

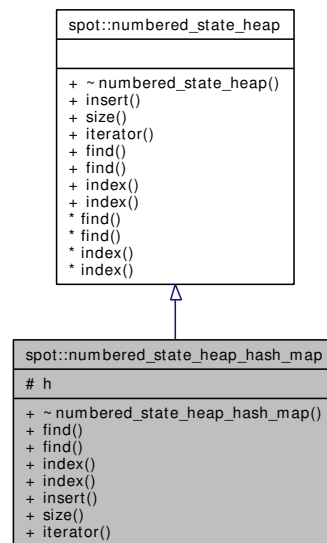
- `tgbaalgos/gtec/nsheap.hh`

## 12.65 spot::numbered\_state\_heap\_hash\_map Class Reference

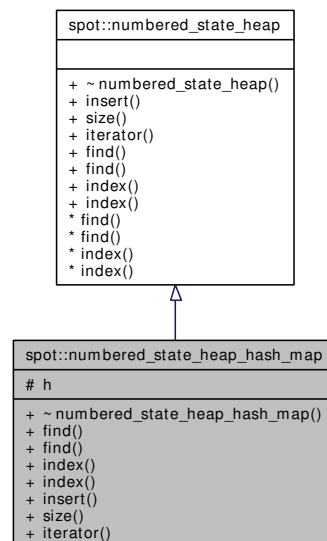
A straightforward implementation of [numbered\\_state\\_heap](#) with a hash map.

```
#include <tgbaalgos/gtec/nsheap.hh>
```

Inheritance diagram for spot::numbered\_state\_heap\_hash\_map:



Collaboration diagram for spot::numbered\_state\_heap\_hash\_map:



## Public Types

- `typedef Sgi::hash_map< const state *, int, state\_ptr\_hash, state\_ptr\_equal > hash\_type`
- `typedef std::pair< const state *, int * > state\_index\_p`
- `typedef std::pair< const state *, int > state\_index`

## Public Member Functions

- virtual `~numbered\_state\_heap\_hash\_map ()`
- virtual `state\_index find (const state *s) const`

Is *state* in the heap?

- virtual `state_index_p find` (const `state` \*s)
- virtual `state_index index` (const `state` \*s) const  
Return the index of an existing *state*.
- virtual `state_index_p index` (const `state` \*s)
- virtual void `insert` (const `state` \*s, int index)  
Add a new *state* s with index index.
- virtual int `size` () const  
The number of stored states.
- virtual `numbered_state_heap_const_iterator` \* `iterator` () const  
Return an iterator on the states/indexes pairs.

### Protected Attributes

- `hash_type h`  
Map of visited states.

### 12.65.1 Detailed Description

A straightforward implementation of `numbered_state_heap` with a hash map.

### 12.65.2 Member Typedef Documentation

**12.65.2.1** `typedef Sgi::hash_map<const state*, int, state_ptr_hash, state_ptr_equal> spot::numbered_state_heap_hash_map::hash_type`

**12.65.2.2** `typedef std::pair<const state*, int*> spot::numbered_state_heap::state_index_p`  
[inherited]

**12.65.2.3** `typedef std::pair<const state*, int> spot::numbered_state_heap::state_index`  
[inherited]

### 12.65.3 Constructor & Destructor Documentation

**12.65.3.1** `virtual spot::numbered_state_heap_hash_map::~~numbered_state_heap_hash_map ()`  
[virtual]

### 12.65.4 Member Function Documentation

**12.65.4.1** `virtual state_index spot::numbered_state_heap_hash_map::find (const state * s) const`  
[virtual]

Is [state](#) in the heap?

Returns a pair (0,0) if *s* is not in the heap. or a pair (p, i) if there is a clone *p* of *s* *i* in the heap with index. If *s* is in the heap and is different from *p* it will be freed.

These functions are called by the algorithm to check whether a successor is a new [state](#) to explore or an already visited [state](#).

These functions can be redefined to search for more than an equal match. For example we could redefine it to check [state](#) inclusion.

Implements [spot::numbered\\_state\\_heap](#).

**12.65.4.2 virtual state\_index\_p spot::numbered\_state\_heap\_hash\_map::find (const state \* s)**  
[virtual]

Implements [spot::numbered\\_state\\_heap](#).

**12.65.4.3 virtual state\_index spot::numbered\_state\_heap\_hash\_map::index (const state \* s) const**  
[virtual]

Return the index of an existing [state](#).

This is mostly similar to [find\(\)](#), except it will be called for [state](#) which we know are already in the heap, or for [state](#) which may not be in the heap but for which it is always OK to do equality checks.

Implements [spot::numbered\\_state\\_heap](#).

**12.65.4.4 virtual state\_index\_p spot::numbered\_state\_heap\_hash\_map::index (const state \* s)**  
[virtual]

Implements [spot::numbered\\_state\\_heap](#).

**12.65.4.5 virtual void spot::numbered\_state\_heap\_hash\_map::insert (const state \* s, int index)**  
[virtual]

Add a new [state](#) *s* with index *index*.

Implements [spot::numbered\\_state\\_heap](#).

**12.65.4.6 virtual int spot::numbered\_state\_heap\_hash\_map::size () const** [virtual]

The number of stored states.

Implements [spot::numbered\\_state\\_heap](#).

**12.65.4.7 virtual numbered\_state\_heap\_const\_iterator\* spot::numbered\_state\_heap\_hash\_map::iterator () const** [virtual]

Return an iterator on the states/indexes pairs.

Implements [spot::numbered\\_state\\_heap](#).

## 12.65.5 Member Data Documentation

**12.65.5.1 hash\_type spot::numbered\_state\_heap\_hash\_map::h** [protected]

Map of visited states.

The documentation for this class was generated from the following file:

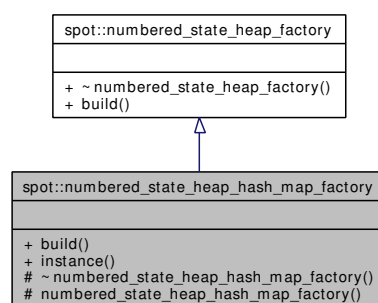
- [tgbalgorithms/gtec/nsheap.hh](#)

## 12.66 spot::numbered\_state\_heap\_hash\_map\_factory Class Reference

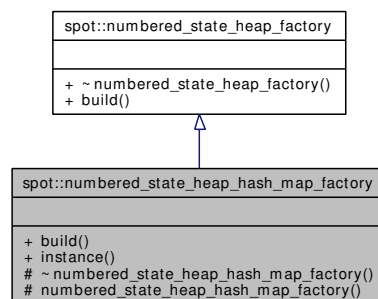
Factory for [numbered\\_state\\_heap\\_hash\\_map](#).

```
#include <tgbalgorithms/gtec/nsheap.hh>
```

Inheritance diagram for spot::numbered\_state\_heap\_hash\_map\_factory:



Collaboration diagram for spot::numbered\_state\_heap\_hash\_map\_factory:



### Public Member Functions

- virtual [numbered\\_state\\_heap\\_hash\\_map](#) \* [build](#) () const

### Static Public Member Functions

- static const [numbered\\_state\\_heap\\_hash\\_map\\_factory](#) \* [instance](#) ()  
*Get the unique instance of this class.*

### Protected Member Functions

- virtual [~numbered\\_state\\_heap\\_hash\\_map\\_factory](#) ()
- [numbered\\_state\\_heap\\_hash\\_map\\_factory](#) ()

### 12.66.1 Detailed Description

Factory for [numbered\\_state\\_heap\\_hash\\_map](#).

This class is a singleton. Retrieve the instance using [instance\(\)](#).

### 12.66.2 Constructor & Destructor Documentation

**12.66.2.1** `virtual spot::numbered_state_heap_hash_map_factory::~~numbered_state_heap_hash_map_factory () [inline, protected, virtual]`

**12.66.2.2** `spot::numbered_state_heap_hash_map_factory::numbered_state_heap_hash_map_factory () [protected]`

### 12.66.3 Member Function Documentation

**12.66.3.1** `virtual numbered_state_heap_hash_map* spot::numbered_state_heap_hash_map_factory::build () const [virtual]`

Implements [spot::numbered\\_state\\_heap\\_factory](#).

**12.66.3.2** `static const numbered_state_heap_hash_map_factory* spot::numbered_state_heap_hash_map_factory::instance () [static]`

Get the unique instance of this class.

The documentation for this class was generated from the following file:

- [tgbalgos/gtec/nsheap.hh](#)

## 12.67 spot::option\_map Class Reference

Manage a map of options.

```
#include <misc/optionmap.hh>
```

### Public Member Functions

- `const char * parse\_options (const char *options)`  
*Add the parsed options to the map.*
- `int get (const char *option, int def=0) const`  
*Get the value of option.*
- `int operator\[\] (const char *option) const`  
*Get the value of option.*
- `int set (const char *option, int val, int def=0)`  
*Set the value of option to val.*
- `void set (const option\_map &o)`



*Acquire all the settings of o.*

- `int & operator[] (const char *option)`  
*Get a reference to the current value of option.*

### Private Attributes

- `std::map< std::string, int > options_`

### Friends

- `std::ostream & operator<< (std::ostream &os, const option_map &m)`  
*Print the `option_map` m.*

## 12.67.1 Detailed Description

Manage a map of options.

Each option is defined by a string and is associated to an integer value.

## 12.67.2 Member Function Documentation

### 12.67.2.1 `const char* spot::option_map::parse_options (const char * options)`

Add the parsed options to the map.

*options* are separated by a space, comma, semicolon or tabulation and can be optionally followed by an integer value (preceded by an equal sign). If not specified, the default value is 1.

The following three lines are equivalent.

```
/// optA !optB optC=4194304
/// optA=1, optB=0, optC=4096K
/// optC = 4M; optA !optB
///
```

### Returns:

A non-null pointer to the option for which an expected integer value cannot be parsed.

### 12.67.2.2 `int spot::option_map::get (const char * option, int def = 0) const`

Get the value of *option*.

### Returns:

The value associated to *option* if it exists, *def* otherwise.

### See also:

[operator\[\]\(\)](#)

**12.67.2.3 int spot::option\_map::operator[] (const char \* *option*) const**

Get the value of *option*.

**Returns:**

The value associated to *option* if it exists, 0 otherwise.

**See also:**

[get\(\)](#)

**12.67.2.4 int spot::option\_map::set (const char \* *option*, int *val*, int *def* = 0)**

Set the value of *option* to *val*.

**Returns:**

The previous value associated to *option* if declared, or *def* otherwise.

**12.67.2.5 void spot::option\_map::set (const option\_map & *o*)**

Acquire all the settings of *o*.

**12.67.2.6 int& spot::option\_map::operator[] (const char \* *option*)**

Get a reference to the current value of *option*.

**12.67.3 Friends And Related Function Documentation****12.67.3.1 std::ostream& operator<< (std::ostream & *os*, const option\_map & *m*)** [friend]

Print the [option\\_map](#) *m*.

**12.67.4 Member Data Documentation****12.67.4.1 std::map<std::string, int> spot::option\_map::options\_** [private]

The documentation for this class was generated from the following file:

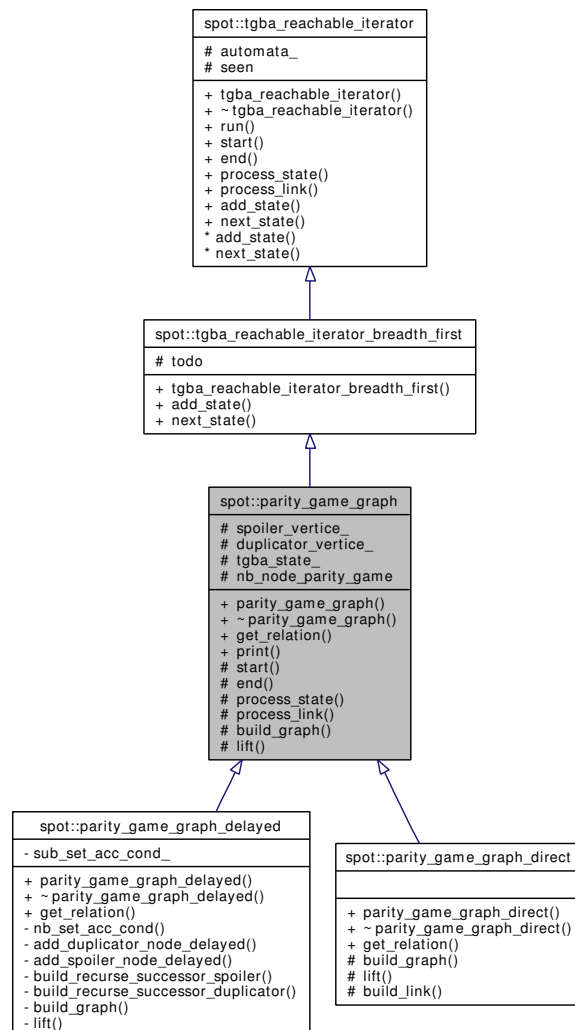
- [misc/optionmap.hh](#)

**12.68 spot::parity\_game\_graph Class Reference**

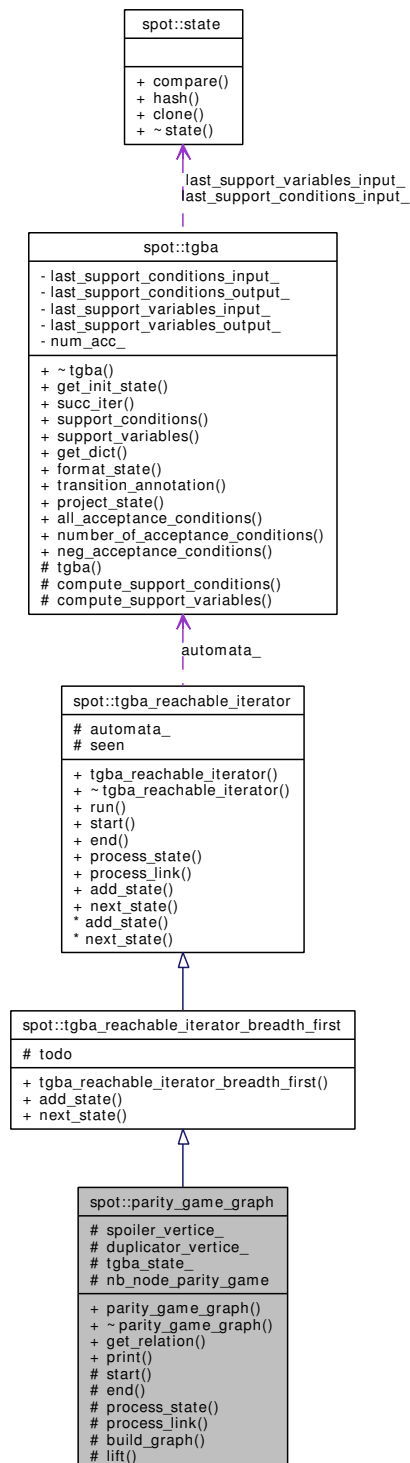
Parity game graph which compute a simulation relation.

```
#include <tgbaalgorithms/reductgba_sim.hh>
```

Inheritance diagram for spot::parity\_game\_graph:



Collaboration diagram for spot::parity\_game\_graph:



## Public Member Functions

- `parity_game_graph` (const `tgba` \*a)

- virtual `~parity_game_graph()`
- virtual `simulation_relation * get_relation()=0`
- void `print` (std::ostream &os)
- virtual void `add_state` (const `state` \*s)
- virtual const `state` \* `next_state` ()  
*Called by `run()` to obtain the.*
- void `run` ()  
*Iterate over all reachable states of a `spot::tgba`.*
- virtual void `process_link` (const `state` \*in\_s, int in, const `state` \*out\_s, int out, const `tgba_succ_iterator` \*si)

### Protected Types

- typedef Sgi::hash\_map< const `state` \*, int, `state_ptr_hash`, `state_ptr_equal` > `seen_map`

### Protected Member Functions

- void `start` ()  
*Called by `run()` before starting its iteration.*
- void `end` ()  
*Called by `run()` once all states have been explored.*
- void `process_state` (const `state` \*s, int n, `tgba_succ_iterator` \*si)
- void `process_link` (int in, int out, const `tgba_succ_iterator` \*si)
- virtual void `build_graph` ()=0  
*Compute each node of the graph.*
- virtual void `lift` ()=0  
*Remove edge from spoiler to duplicator that make duplicator loose. Spoiler node whose still have some link, reveal a direct simulation relation.*

### Protected Attributes

- `sn_v spoiler_vertice_`
- `dn_v duplicator_vertice_`
- `s_v tgba_state_`
- int `nb_node_parity_game`
- std::deque< const `state` \* > `todo`  
*A queue of states yet to explore.*
- const `tgba` \* `automata_`  
*The `spot::tgba` to explore.*
- `seen_map` `seen`  
*States already seen.*

### 12.68.1 Detailed Description

Parity game graph which compute a simulation relation.

### 12.68.2 Member Typedef Documentation

**12.68.2.1** `typedef Sgi::hash_map<const state*, int, state_ptr_hash, state_ptr_equal> spot::tgba_reachable_iterator::seen_map` [protected, inherited]

Reimplemented in [spot::tgba\\_reduc](#).

### 12.68.3 Constructor & Destructor Documentation

**12.68.3.1** `spot::parity_game_graph::parity_game_graph (const tgba * a)`

**12.68.3.2** `virtual spot::parity_game_graph::~~parity_game_graph ()` [virtual]

### 12.68.4 Member Function Documentation

**12.68.4.1** `virtual simulation_relation* spot::parity_game_graph::get_relation ()` [pure virtual]

Implemented in [spot::parity\\_game\\_graph\\_direct](#), and [spot::parity\\_game\\_graph\\_delayed](#).

**12.68.4.2** `void spot::parity_game_graph::print (std::ostream & os)`

**12.68.4.3** `void spot::parity_game_graph::start ()` [protected, virtual]

Called by [run\(\)](#) before starting its iteration.

Reimplemented from [spot::tgba\\_reachable\\_iterator](#).

**12.68.4.4** `void spot::parity_game_graph::end ()` [protected, virtual]

Called by [run\(\)](#) once all states have been explored.

Reimplemented from [spot::tgba\\_reachable\\_iterator](#).

**12.68.4.5** `void spot::parity_game_graph::process_state (const state * s, int n, tgba_succ_iterator * si)` [protected, virtual]

Called by [run\(\)](#) to process a [state](#).

#### Parameters:

- s* The current [state](#).
- n* A unique number assigned to *s*.
- si* The [spot::tgba\\_succ\\_iterator](#) for *s*.

Reimplemented from [spot::tgba\\_reachable\\_iterator](#).

**12.68.4.6** `void spot::parity_game_graph::process_link (int in, int out, const tgba_succ_iterator * si)` [protected]

**12.68.4.7** `virtual void spot::parity_game_graph::build_graph ()` [protected, pure virtual]

Compute each node of the graph.

Implemented in [spot::parity\\_game\\_graph\\_direct](#), and [spot::parity\\_game\\_graph\\_delayed](#).

**12.68.4.8** `virtual void spot::parity_game_graph::lift ()` [protected, pure virtual]

Remove edge from spoiler to duplicator that make duplicator loose. Spoiler node whose still have some link, reveal a direct simulation relation.

Implemented in [spot::parity\\_game\\_graph\\_direct](#), and [spot::parity\\_game\\_graph\\_delayed](#).

**12.68.4.9** `virtual void spot::tgba_reachable_iterator_breadth_first::add_state (const state * s)` [virtual, inherited]

Implements [spot::tgba\\_reachable\\_iterator](#).

**12.68.4.10** `virtual const state* spot::tgba_reachable_iterator_breadth_first::next_state ()` [virtual, inherited]

Called by [run\(\)](#) to obtain the.

Implements [spot::tgba\\_reachable\\_iterator](#).

**12.68.4.11** `void spot::tgba_reachable_iterator::run ()` [inherited]

Iterate over all reachable states of a [spot::tgba](#).

This is a template method that will call [add\\_state\(\)](#), [next\\_state\(\)](#), [start\(\)](#), [end\(\)](#), [process\\_state\(\)](#), and [process\\_link\(\)](#), while it iterate over [state](#).

**12.68.4.12** `virtual void spot::tgba_reachable_iterator::process_link (const state * in_s, int in, const state * out_s, int out, const tgba_succ_iterator * si)` [virtual, inherited]

Called by [run\(\)](#) to process a transition.

#### Parameters:

*in\_s* The source [state](#)

*in* The source [state](#) number.

*out\_s* The destination [state](#)

*out* The destination [state](#) number.

*si* The [spot::tgba\\_succ\\_iterator](#) positionned on the current transition.

The *in\_s* and *out\_s* states are owned by the [spot::tgba\\_reachable\\_iterator](#) instance and destroyed when the instance is destroyed.

## 12.68.5 Member Data Documentation

**12.68.5.1** `sn_v spot::parity_game_graph::spoiler_vertice_` [protected]

**12.68.5.2** `dn_v spot::parity_game_graph::duplicator_vertice_` [protected]

**12.68.5.3** `s_v spot::parity_game_graph::tgba_state_` [protected]

**12.68.5.4** `int spot::parity_game_graph::nb_node_parity_game` [protected]

**12.68.5.5** `std::deque<const state*> spot::tgba_reachable_iterator_breadth_first::todo`  
[protected, inherited]

A queue of states yet to explore.

**12.68.5.6** `const tgba* spot::tgba_reachable_iterator::automata_` [protected, inherited]

The [spot::tgba](#) to explore.

**12.68.5.7** `seen_map spot::tgba_reachable_iterator::seen` [protected, inherited]

States already seen.

The documentation for this class was generated from the following file:

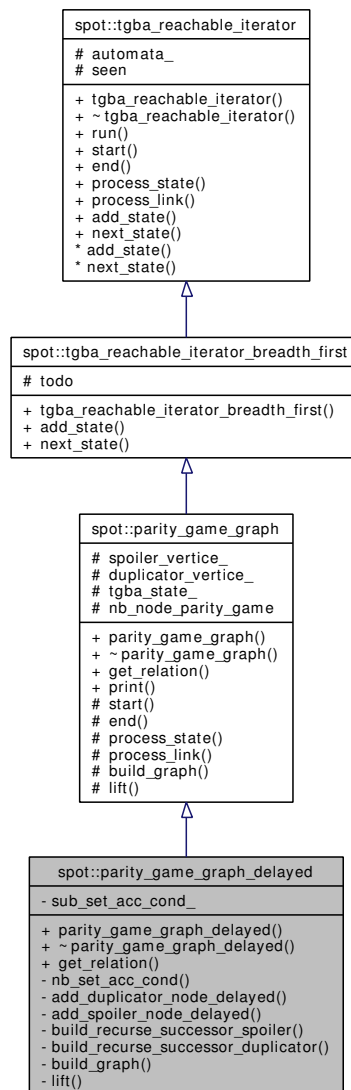
- [tgbaalgos/reductgba\\_sim.hh](#)

## 12.69 spot::parity\_game\_graph\_delayed Class Reference

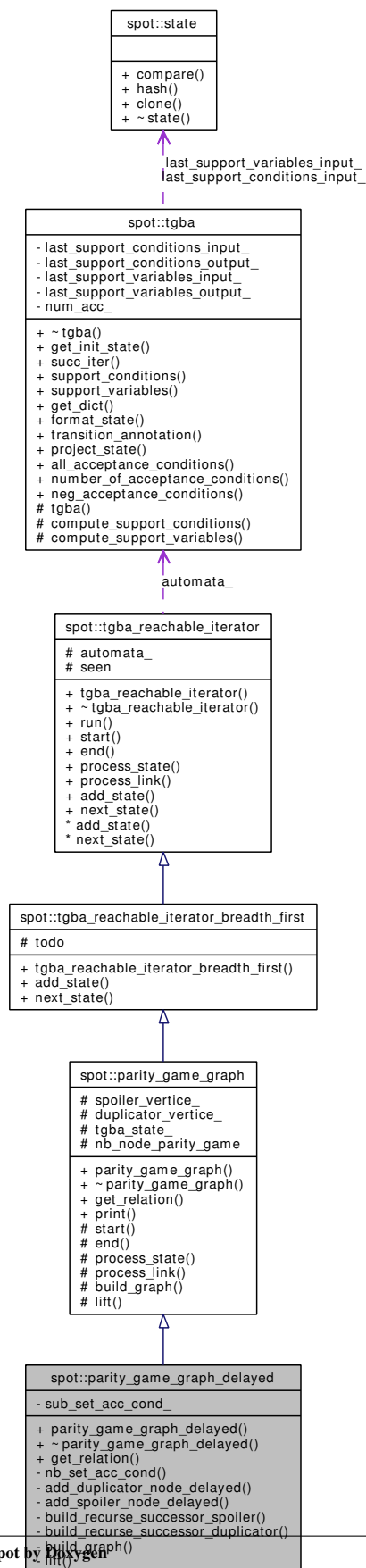
```
#include <tgbaalgos/reductgba_sim.hh>
```



Inheritance diagram for spot::parity\_game\_graph\_delayed:



Collaboration diagram for spot::parity\_game\_graph\_delayed:



### Public Member Functions

- [parity\\_game\\_graph\\_delayed](#) (const [tgba](#) \*a)
- [~parity\\_game\\_graph\\_delayed](#) ()
- virtual [delayed\\_simulation\\_relation](#) \* [get\\_relation](#) ()
- void [print](#) (std::ostream &os)
- virtual void [process\\_link](#) (const [state](#) \*in\_s, int in, const [state](#) \*out\_s, int out, const [tgba\\_succ\\_iterator](#) \*si)
- virtual void [add\\_state](#) (const [state](#) \*s)
- virtual const [state](#) \* [next\\_state](#) ()  
*Called by [run\(\)](#) to obtain the.*
- void [run](#) ()  
*Iterate over all reachable states of a [spot::tgba](#).*

### Protected Types

- typedef Sgi::hash\_map< const [state](#) \*, int, [state\\_ptr\\_hash](#), [state\\_ptr\\_equal](#) > [seen\\_map](#)

### Protected Member Functions

- void [start](#) ()  
*Called by [run\(\)](#) before starting its iteration.*
- void [end](#) ()  
*Called by [run\(\)](#) once all states have been explored.*
- void [process\\_state](#) (const [state](#) \*s, int n, [tgba\\_succ\\_iterator](#) \*si)
- void [process\\_link](#) (int in, int out, const [tgba\\_succ\\_iterator](#) \*si)

### Protected Attributes

- [sn\\_v](#) [spoiler\\_vertice\\_](#)
- [dn\\_v](#) [duplicator\\_vertice\\_](#)
- [s\\_v](#) [tgba\\_state\\_](#)
- int [nb\\_node\\_parity\\_game](#)
- std::deque< const [state](#) \* > [todo](#)  
*A queue of states yet to explore.*
- const [tgba](#) \* [automata\\_](#)  
*The [spot::tgba](#) to explore.*
- [seen\\_map](#) [seen](#)  
*States already seen.*

### Private Types

- typedef std::vector< bdd > [bdd\\_v](#)

### Private Member Functions

- `int nb_set_acc_cond ()`  
*Return the number of acceptance condition.*
- `duplicator_node_delayed * add_duplicator_node_delayed (const spot::state *sn, const spot::state *dn, bdd acc, bdd label, int nb)`
- `spoiler_node_delayed * add_spoiler_node_delayed (const spot::state *sn, const spot::state *dn, bdd acc, int nb)`
- `void build_recurse_successor_spoiler (spoiler_node *sn, std::ostream &os)`
- `void build_recurse_successor_duplicator (duplicator_node *dn, spoiler_node *sn, std::ostream &os)`
- `virtual void build_graph ()`  
*Compute the couple as for direct simulation,.*
- `virtual void lift ()`  
*The Jurdzinski's lifting algorithm.*

### Private Attributes

- `bdd_v sub_set_acc_cond_`

## 12.69.1 Detailed Description

Parity game graph which computes the delayed simulation relation as explained in

```
/// @InProceedings{etessami.01.alp,
///   author = {Kousha Etessami and Thomas Wilke and Rebecca A. Schuller},
///   title = {Fair Simulation Relations, Parity Games, and State Space
///     Reduction for Buchi Automata},
///   booktitle = {Proceedings of the 28th international colloquium on
///     Automata, Languages and Programming},
///   pages = {694--707},
///   year = {2001},
///   editor = {Fernando Orejas and Paul G. Spirakis and Jan van Leeuwen},
///   volume = {2076},
///   series = {Lecture Notes in Computer Science},
///   address = {Crete, Greece},
///   month = {July},
///   publisher = {Springer-Verlag}
/// }
```

## 12.69.2 Member Typedef Documentation

### 12.69.2.1 `typedef std::vector<bdd> spot::parity_game_graph_delayed::bdd_v` [private]

Vector which contain all the sub-set of the set of acceptance condition.

### 12.69.2.2 `typedef Sgi::hash_map<const state*, int, state_ptr_hash, state_ptr_equal> spot::tgba_reachable_iterator::seen_map` [protected, inherited]

Reimplemented in `spot::tgba_reduc`.

### 12.69.3 Constructor & Destructor Documentation

**12.69.3.1** spot::parity\_game\_graph\_delayed::parity\_game\_graph\_delayed (const tgba \* a)

**12.69.3.2** spot::parity\_game\_graph\_delayed::~~parity\_game\_graph\_delayed ()

### 12.69.4 Member Function Documentation

**12.69.4.1** virtual delayed\_simulation\_relation\* spot::parity\_game\_graph\_delayed::get\_relation ()  
[virtual]

Implements [spot::parity\\_game\\_graph](#).

**12.69.4.2** int spot::parity\_game\_graph\_delayed::nb\_set\_acc\_cond () [private]

Return the number of acceptance condition.

**12.69.4.3** duplicator\_node\_delayed\* spot::parity\_game\_graph\_delayed::add\_duplicator\_node\_delayed (const spot::state \* sn, const spot::state \* dn, bdd acc, bdd label, int nb) [private]

**12.69.4.4** spoiler\_node\_delayed\* spot::parity\_game\_graph\_delayed::add\_spoiler\_node\_delayed (const spot::state \* sn, const spot::state \* dn, bdd acc, int nb) [private]

**12.69.4.5** void spot::parity\_game\_graph\_delayed::build\_recurse\_successor\_spoiler (spoiler\_node \* sn, std::ostream & os) [private]

**12.69.4.6** void spot::parity\_game\_graph\_delayed::build\_recurse\_successor\_duplicator (duplicator\_node \* dn, spoiler\_node \* sn, std::ostream & os) [private]

**12.69.4.7** virtual void spot::parity\_game\_graph\_delayed::build\_graph () [private, virtual]

Compute the couple as for direct simulation,.

Implements [spot::parity\\_game\\_graph](#).

**12.69.4.8** virtual void spot::parity\_game\_graph\_delayed::lift () [private, virtual]

The Jurdzinski's lifting algorithm.

Implements [spot::parity\\_game\\_graph](#).

**12.69.4.9** void spot::parity\_game\_graph::print (std::ostream & os) [inherited]

**12.69.4.10** void spot::parity\_game\_graph::start () [protected, virtual, inherited]

Called by [run\(\)](#) before starting its iteration.

Reimplemented from [spot::tgba\\_reachable\\_iterator](#).

**12.69.4.11 void spot::parity\_game\_graph::end ()** [protected, virtual, inherited]

Called by [run\(\)](#) once all states have been explored.

Reimplemented from [spot::tgba\\_reachable\\_iterator](#).

**12.69.4.12 void spot::parity\_game\_graph::process\_state (const state \* *s*, int *n*, tgba\_succ\_iterator \* *si*)** [protected, virtual, inherited]

Called by [run\(\)](#) to process a [state](#).

**Parameters:**

- s* The current [state](#).
- n* A unique number assigned to *s*.
- si* The [spot::tgba\\_succ\\_iterator](#) for *s*.

Reimplemented from [spot::tgba\\_reachable\\_iterator](#).

**12.69.4.13 void spot::parity\_game\_graph::process\_link (int *in*, int *out*, const tgba\_succ\_iterator \* *si*)** [protected, inherited]**12.69.4.14 virtual void spot::tgba\_reachable\_iterator::process\_link (const state \* *in\_s*, int *in*, const state \* *out\_s*, int *out*, const tgba\_succ\_iterator \* *si*)** [virtual, inherited]

Called by [run\(\)](#) to process a transition.

**Parameters:**

- in\_s* The source [state](#)
- in* The source [state](#) number.
- out\_s* The destination [state](#)
- out* The destination [state](#) number.
- si* The [spot::tgba\\_succ\\_iterator](#) positionned on the current transition.

The *in\_s* and *out\_s* states are owned by the [spot::tgba\\_reachable\\_iterator](#) instance and destroyed when the instance is destroyed.

**12.69.4.15 virtual void spot::tgba\_reachable\_iterator\_breadth\_first::add\_state (const state \* *s*)** [virtual, inherited]

Implements [spot::tgba\\_reachable\\_iterator](#).

**12.69.4.16 virtual const state\* spot::tgba\_reachable\_iterator\_breadth\_first::next\_state ()** [virtual, inherited]

Called by [run\(\)](#) to obtain the.

Implements [spot::tgba\\_reachable\\_iterator](#).

**12.69.4.17 void spot::tgba\_reachable\_iterator::run () [inherited]**

Iterate over all reachable states of a [spot::tgba](#).

This is a template method that will call [add\\_state\(\)](#), [next\\_state\(\)](#), [start\(\)](#), [end\(\)](#), [process\\_state\(\)](#), and [process\\_link\(\)](#), while it iterate over [state](#).

**12.69.5 Member Data Documentation****12.69.5.1 bdd\_v spot::parity\_game\_graph\_delayed::sub\_set\_acc\_cond\_ [private]****12.69.5.2 sn\_v spot::parity\_game\_graph::spoiler\_vertice\_ [protected, inherited]****12.69.5.3 dn\_v spot::parity\_game\_graph::duplicator\_vertice\_ [protected, inherited]****12.69.5.4 s\_v spot::parity\_game\_graph::tgba\_state\_ [protected, inherited]****12.69.5.5 int spot::parity\_game\_graph::nb\_node\_parity\_game [protected, inherited]****12.69.5.6 std::deque<const state\*> spot::tgba\_reachable\_iterator\_breadth\_first::todo [protected, inherited]**

A queue of states yet to explore.

**12.69.5.7 const tgba\* spot::tgba\_reachable\_iterator::automata\_ [protected, inherited]**

The [spot::tgba](#) to explore.

**12.69.5.8 seen\_map spot::tgba\_reachable\_iterator::seen [protected, inherited]**

States already seen.

The documentation for this class was generated from the following file:

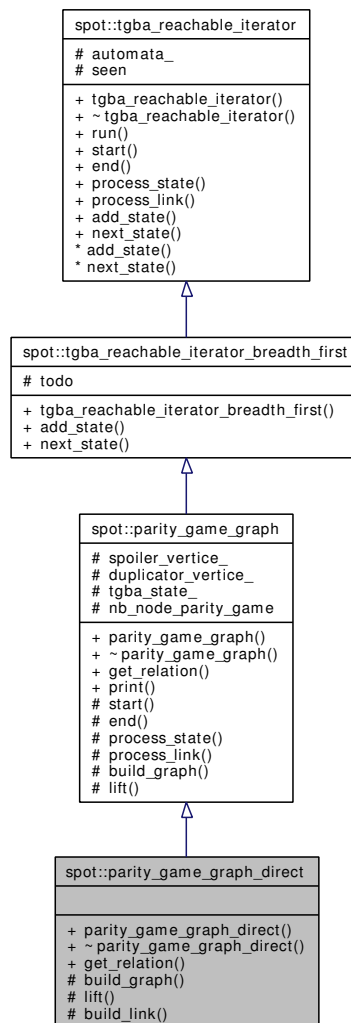
- [tgbaalgos/reductgba\\_sim.hh](#)

**12.70 spot::parity\_game\_graph\_direct Class Reference**

Parity game graph which compute the direct simulation relation.

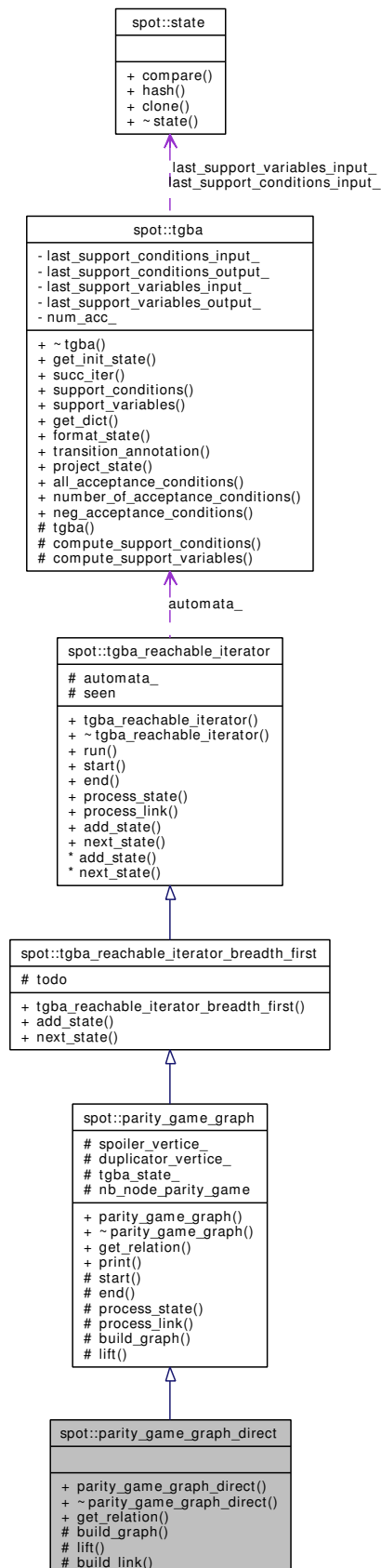
```
#include <tgbaalgos/reductgba_sim.hh>
```

Inheritance diagram for spot::parity\_game\_graph\_direct:





Collaboration diagram for spot::parity\_game\_graph\_direct:



### Public Member Functions

- [parity\\_game\\_graph\\_direct](#) (const [tgba](#) \*a)
- [~parity\\_game\\_graph\\_direct](#) ()
- virtual [direct\\_simulation\\_relation](#) \* [get\\_relation](#) ()
- void [print](#) (std::ostream &os)
- virtual void [process\\_link](#) (const [state](#) \*in\_s, int in, const [state](#) \*out\_s, int out, const [tgba\\_succ\\_iterator](#) \*si)
- virtual void [add\\_state](#) (const [state](#) \*s)
- virtual const [state](#) \* [next\\_state](#) ()  
*Called by [run\(\)](#) to obtain the.*
- void [run](#) ()  
*Iterate over all reachable states of a [spot::tgba](#).*

### Protected Types

- typedef Sgi::hash\_map< const [state](#) \*, int, [state\\_ptr\\_hash](#), [state\\_ptr\\_equal](#) > [seen\\_map](#)

### Protected Member Functions

- virtual void [build\\_graph](#) ()  
*Compute each node of the graph.*
- virtual void [lift](#) ()  
*Remove edge from spoiler to duplicator that make duplicator loose. Spoiler node whose still have some link, reveal a direct simulation relation.*
- void [build\\_link](#) ()
- void [start](#) ()  
*Called by [run\(\)](#) before starting its iteration.*
- void [end](#) ()  
*Called by [run\(\)](#) once all states have been explored.*
- void [process\\_state](#) (const [state](#) \*s, int n, [tgba\\_succ\\_iterator](#) \*si)
- void [process\\_link](#) (int in, int out, const [tgba\\_succ\\_iterator](#) \*si)

### Protected Attributes

- [sn\\_v spoiler\\_vertice\\_](#)
- [dn\\_v duplicator\\_vertice\\_](#)
- [s\\_v tgba\\_state\\_](#)
- int [nb\\_node\\_parity\\_game](#)
- std::deque< const [state](#) \* > [todo](#)  
*A queue of states yet to explore.*
- const [tgba](#) \* [automata\\_](#)

The `spot::tgba` to explore.

- `seen_map` seen

States already seen.

### 12.70.1 Detailed Description

Parity game graph which compute the direct simulation relation.

### 12.70.2 Member Typedef Documentation

**12.70.2.1** `typedef Sgi::hash_map<const state*, int, state_ptr_hash, state_ptr_equal> spot::tgba_reachable_iterator::seen_map` [protected, inherited]

Reimplemented in `spot::tgba_reduc`.

### 12.70.3 Constructor & Destructor Documentation

**12.70.3.1** `spot::parity_game_graph_direct::parity_game_graph_direct (const tgba * a)`

**12.70.3.2** `spot::parity_game_graph_direct::~~parity_game_graph_direct ()`

### 12.70.4 Member Function Documentation

**12.70.4.1** `virtual direct_simulation_relation* spot::parity_game_graph_direct::get_relation ()` [virtual]

Implements `spot::parity_game_graph`.

**12.70.4.2** `virtual void spot::parity_game_graph_direct::build_graph ()` [protected, virtual]

Compute each node of the graph.

Implements `spot::parity_game_graph`.

**12.70.4.3** `virtual void spot::parity_game_graph_direct::lift ()` [protected, virtual]

Remove edge from spoiler to duplicator that make duplicator loose. Spoiler node whose still have some link, reveal a direct simulation relation.

Implements `spot::parity_game_graph`.

**12.70.4.4** `void spot::parity_game_graph_direct::build_link ()` [protected]

**12.70.4.5** `void spot::parity_game_graph::print (std::ostream & os)` [inherited]

**12.70.4.6** `void spot::parity_game_graph::start ()` [protected, virtual, inherited]

Called by `run()` before starting its iteration.

Reimplemented from `spot::tgba_reachable_iterator`.

**12.70.4.7** `void spot::parity_game_graph::end ()` [protected, virtual, inherited]

Called by `run()` once all states have been explored.

Reimplemented from `spot::tgba_reachable_iterator`.

**12.70.4.8** `void spot::parity_game_graph::process_state (const state * s, int n, tgba_succ_iterator * si)` [protected, virtual, inherited]

Called by `run()` to process a `state`.

**Parameters:**

- `s` The current `state`.
- `n` A unique number assigned to `s`.
- `si` The `spot::tgba_succ_iterator` for `s`.

Reimplemented from `spot::tgba_reachable_iterator`.

**12.70.4.9** `void spot::parity_game_graph::process_link (int in, int out, const tgba_succ_iterator * si)` [protected, inherited]**12.70.4.10** `virtual void spot::tgba_reachable_iterator::process_link (const state * in_s, int in, const state * out_s, int out, const tgba_succ_iterator * si)` [virtual, inherited]

Called by `run()` to process a transition.

**Parameters:**

- `in_s` The source `state`
- `in` The source `state` number.
- `out_s` The destination `state`
- `out` The destination `state` number.
- `si` The `spot::tgba_succ_iterator` positionned on the current transition.

The `in_s` and `out_s` states are owned by the `spot::tgba_reachable_iterator` instance and destroyed when the instance is destroyed.

**12.70.4.11** `virtual void spot::tgba_reachable_iterator_breadth_first::add_state (const state * s)` [virtual, inherited]

Implements `spot::tgba_reachable_iterator`.

**12.70.4.12** `virtual const state* spot::tgba_reachable_iterator_breadth_first::next_state ()` [virtual, inherited]

Called by `run()` to obtain the.

Implements `spot::tgba_reachable_iterator`.

**12.70.4.13 void spot::tgba\_reachable\_iterator::run ()** [inherited]

Iterate over all reachable states of a [spot::tgba](#).

This is a template method that will call [add\\_state\(\)](#), [next\\_state\(\)](#), [start\(\)](#), [end\(\)](#), [process\\_state\(\)](#), and [process\\_link\(\)](#), while it iterate over [state](#).

**12.70.5 Member Data Documentation****12.70.5.1 sn\_v spot::parity\_game\_graph::spoiler\_vertice\_** [protected, inherited]**12.70.5.2 dn\_v spot::parity\_game\_graph::duplicator\_vertice\_** [protected, inherited]**12.70.5.3 s\_v spot::parity\_game\_graph::tgba\_state\_** [protected, inherited]**12.70.5.4 int spot::parity\_game\_graph::nb\_node\_parity\_game** [protected, inherited]**12.70.5.5 std::deque<const state\*> spot::tgba\_reachable\_iterator\_breadth\_first::todo** [protected, inherited]

A queue of states yet to explore.

**12.70.5.6 const tgba\* spot::tgba\_reachable\_iterator::automata\_** [protected, inherited]

The [spot::tgba](#) to explore.

**12.70.5.7 seen\_map spot::tgba\_reachable\_iterator::seen** [protected, inherited]

States already seen.

The documentation for this class was generated from the following file:

- [tgbaalgorithms/reductgba\\_sim.hh](#)

**12.71 Itlly::position Class Reference**

Abstract a [position](#).

```
#include <ltlparse/position.hh>
```

**Public Member Functions**

- [position \(\)](#)  
*Construct a [position](#).*
- void [initialize](#) (std::string \*fn)  
*Initialization.*

### Line and Column related manipulators

- void [lines](#) (int count=1)  
(line related) Advance to the *COUNT* next lines.
- void [columns](#) (int count=1)  
(column related) Advance to the *COUNT* next columns.

### Public Attributes

- std::string \* [filename](#)  
File name to which this *position* refers.
- unsigned int [line](#)  
Current line number.
- unsigned int [column](#)  
Current column number.

#### 12.71.1 Detailed Description

Abstract a [position](#).

#### 12.71.2 Constructor & Destructor Documentation

##### 12.71.2.1 Itlly::position::position () [inline]

Construct a [position](#).

#### 12.71.3 Member Function Documentation

##### 12.71.3.1 void Itlly::position::initialize (std::string \*fn) [inline]

Initialization.

##### 12.71.3.2 void Itlly::position::lines (int count = 1) [inline]

(line related) Advance to the *COUNT* next lines.

##### 12.71.3.3 void Itlly::position::columns (int count = 1) [inline]

(column related) Advance to the *COUNT* next columns.

#### 12.71.4 Member Data Documentation

##### 12.71.4.1 std::string\* Itlly::position::filename

File name to which this [position](#) refers.

#### 12.71.4.2 unsigned int ltly::position::line

Current line number.

#### 12.71.4.3 unsigned int ltly::position::column

Current column number.

The documentation for this class was generated from the following file:

- Itlparse/[position.hh](#)

## 12.72 sautyy::position Class Reference

Abstract a [position](#).

```
#include <sautparse/position.hh>
```

### Public Member Functions

- [position](#) ()  
*Construct a [position](#).*
- void [initialize](#) (std::string \*fn)  
*Initialization.*

### Line and Column related manipulators

- void [lines](#) (int count=1)  
*(line related) Advance to the COUNT next lines.*
- void [columns](#) (int count=1)  
*(column related) Advance to the COUNT next columns.*

### Public Attributes

- std::string \* [filename](#)  
*File name to which this [position](#) refers.*
- unsigned int [line](#)  
*Current line number.*
- unsigned int [column](#)  
*Current column number.*

### 12.72.1 Detailed Description

Abstract a [position](#).

## 12.72.2 Constructor & Destructor Documentation

### 12.72.2.1 `sauty::position::position ()` [inline]

Construct a [position](#).

## 12.72.3 Member Function Documentation

### 12.72.3.1 `void sauty::position::initialize (std::string *fn)` [inline]

Initialization.

### 12.72.3.2 `void sauty::position::lines (int count = 1)` [inline]

(line related) Advance to the COUNT next lines.

### 12.72.3.3 `void sauty::position::columns (int count = 1)` [inline]

(column related) Advance to the COUNT next columns.

## 12.72.4 Member Data Documentation

### 12.72.4.1 `std::string* sauty::position::filename`

File name to which this [position](#) refers.

### 12.72.4.2 `unsigned int sauty::position::line`

Current line number.

### 12.72.4.3 `unsigned int sauty::position::column`

Current column number.

The documentation for this class was generated from the following file:

- [sautparse/position.hh](#)

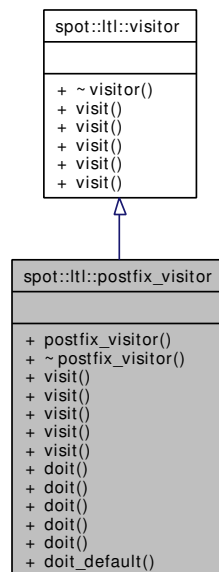
## 12.73 `spot::ltl::postfix_visitor` Class Reference

Apply an algorithm on each node of an AST, during a postfix traversal.

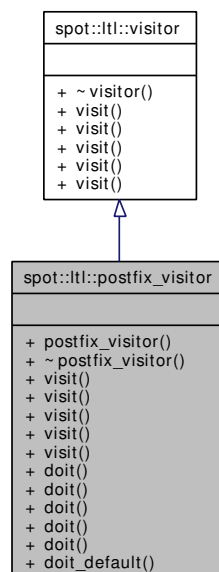
```
#include <ltlvisit/postfix.hh>
```



Inheritance diagram for spot::ltl::postfix\_visitor:



Collaboration diagram for spot::ltl::postfix\_visitor:



## Public Member Functions

- [postfix\\_visitor\(\)](#)
- [virtual ~postfix\\_visitor\(\)](#)
- [void visit \(atomic\\_prop \\*ap\)](#)
- [void visit \(unop \\*uo\)](#)
- [void visit \(binop \\*bo\)](#)
- [void visit \(multop \\*mo\)](#)

- void [visit](#) ([constant](#) \*c)
- virtual void [doit](#) ([atomic\\_prop](#) \*ap)
- virtual void [doit](#) ([unop](#) \*uo)
- virtual void [doit](#) ([binop](#) \*bo)
- virtual void [doit](#) ([multop](#) \*mo)
- virtual void [doit](#) ([constant](#) \*c)
- virtual void [doit\\_default](#) ([formula](#) \*f)

### 12.73.1 Detailed Description

Apply an algorithm on each node of an AST, during a postfix traversal.

Override one or more of the postfix\_visitor::doit methods with the algorithm to apply.

### 12.73.2 Constructor & Destructor Documentation

#### 12.73.2.1 spot::ltl::postfix\_visitor::postfix\_visitor ()

#### 12.73.2.2 virtual spot::ltl::postfix\_visitor::~~postfix\_visitor () [virtual]

### 12.73.3 Member Function Documentation

#### 12.73.3.1 void spot::ltl::postfix\_visitor::visit ([atomic\\_prop](#) \* *ap*) [virtual]

Implements [spot::ltl::visitor](#).

#### 12.73.3.2 void spot::ltl::postfix\_visitor::visit ([unop](#) \* *uo*) [virtual]

Implements [spot::ltl::visitor](#).

#### 12.73.3.3 void spot::ltl::postfix\_visitor::visit ([binop](#) \* *bo*) [virtual]

Implements [spot::ltl::visitor](#).

#### 12.73.3.4 void spot::ltl::postfix\_visitor::visit ([multop](#) \* *mo*) [virtual]

Implements [spot::ltl::visitor](#).

#### 12.73.3.5 void spot::ltl::postfix\_visitor::visit ([constant](#) \* *c*) [virtual]

Implements [spot::ltl::visitor](#).

#### 12.73.3.6 virtual void spot::ltl::postfix\_visitor::doit ([atomic\\_prop](#) \* *ap*) [virtual]

#### 12.73.3.7 virtual void spot::ltl::postfix\_visitor::doit ([unop](#) \* *uo*) [virtual]

#### 12.73.3.8 virtual void spot::ltl::postfix\_visitor::doit ([binop](#) \* *bo*) [virtual]

**12.73.3.9** virtual void spot::ltl::postfix\_visitor::doit (multop \* *mo*) [virtual]

**12.73.3.10** virtual void spot::ltl::postfix\_visitor::doit (constant \* *c*) [virtual]

**12.73.3.11** virtual void spot::ltl::postfix\_visitor::doit\_default (formula \* *f*) [virtual]

The documentation for this class was generated from the following file:

- ltlvisit/[postfix.hh](#)

## 12.74 spot::ptr\_hash< T > Struct Template Reference

A hash function for pointers.

```
#include <misc/hash.hh>
```

### Public Member Functions

- size\_t [operator\(\)](#) (const T \*p) const

### 12.74.1 Detailed Description

```
template<class T> struct spot::ptr_hash< T >
```

A hash function for pointers.

### 12.74.2 Member Function Documentation

**12.74.2.1** template<class T> size\_t spot::ptr\_hash< T >::operator() (const T \* *p*) const [inline]

The documentation for this struct was generated from the following file:

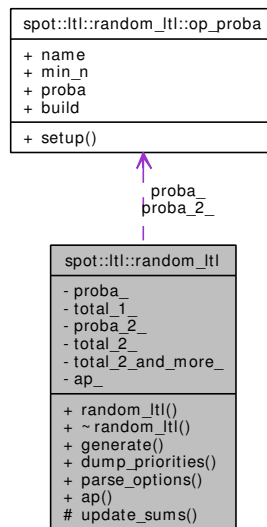
- misc/[hash.hh](#)

## 12.75 spot::ltl::random\_ltl Class Reference

Generate random LTL formulae.

```
#include <ltlvisit/randomltl.hh>
```

Collaboration diagram for spot::ltl::random\_ltl:



### Public Member Functions

- [random\\_ltl](#) (const [atomic\\_prop\\_set](#) \*ap)  
*Create a random LTL generator using atomic propositions from ap.*
- [~random\\_ltl](#) ()
- [formula](#) \* [generate](#) (int n) const  
*Generate a [formula](#) of size n.*
- std::ostream & [dump\\_priorities](#) (std::ostream &os) const  
*Print the priorities of each operator, constants, and atomic propositions.*
- const char \* [parse\\_options](#) (char \*options)  
*Update the priorities used to generate LTL formulae.*
- const [atomic\\_prop\\_set](#) \* [ap](#) () const  
*Return the set of atomic proposition used to build formulae.*

### Protected Member Functions

- void [update\\_sums](#) ()

### Private Attributes

- [op\\_proba](#) \* [proba\\_](#)
- double [total\\_1\\_](#)
- [op\\_proba](#) \* [proba\\_2\\_](#)
- double [total\\_2\\_](#)
- double [total\\_2\\_and\\_more\\_](#)
- const [atomic\\_prop\\_set](#) \* [ap\\_](#)

## Classes

- struct `op_proba`

## 12.75.1 Detailed Description

Generate random LTL formulae.

This class recursively construct LTL formulae of a given size. The formulae will use the use atomic propositions from the set of proposition passed to the constructor, in addition to the `constant` and all LTL operators supported by Spot.

By default each operator has equal chance to be selected. Also, each atomic proposition has as much chance as each `constant` (i.e., true and false) to be picked. This can be tuned using `parse_options()`.

## 12.75.2 Constructor &amp; Destructor Documentation

12.75.2.1 `spot::ltl::random_ltl::random_ltl (const atomic_prop_set * ap)`

Create a random LTL generator using atomic propositions from *ap*.

12.75.2.2 `spot::ltl::random_ltl::~~random_ltl ()`

## 12.75.3 Member Function Documentation

12.75.3.1 `formula* spot::ltl::random_ltl::generate (int n) const`

Generate a `formula` of size *n*.

It is possible to obtain formulae that are smaller than *n*, because some simple simplifications are performed by the AST. (For instance the `formula` `a | a` is automatically reduced to `a` by `spot::ltl::multop`.)

Furthermore, for the purpose of this generator, `a | b | c` has length 5, while it has length 4 for `spot::ltl::length()`.

12.75.3.2 `std::ostream& spot::ltl::random_ltl::dump_priorities (std::ostream & os) const`

Print the priorities of each operator, constants, and atomic propositions.

12.75.3.3 `const char* spot::ltl::random_ltl::parse_options (char * options)`

Update the priorities used to generate LTL formulae.

The initial priorities are as follows.

```

/// ap      n
/// false   1
/// true     1
/// not      1
/// F        1
/// G        1
/// X        1
/// equiv    1
/// implies  1
/// xor      1
/// R        1

```

```

/// U      1
/// and    1
/// or     1
///

```

Where  $n$  is the number of atomic propositions in the set passed to the constructor.

This means that each operator has equal chance to be selected. Also, each atomic proposition has as much chance as each [constant](#) (i.e., true and false) to be picked. This can be

These priorities can be altered using this function. *options* should be comma-separated list of KEY=VALUE assignments, using keys from the above list. For instance "xor=0, F=3" will prevent xor from being used, and will raise the relative probability of occurrences of the F operator.

#### 12.75.3.4 const atomic\_prop\_set\* spot::ltl::random\_ltl::ap () const [inline]

Return the set of atomic proposition used to build formulae.

#### 12.75.3.5 void spot::ltl::random\_ltl::update\_sums () [protected]

### 12.75.4 Member Data Documentation

#### 12.75.4.1 op\_proba\* spot::ltl::random\_ltl::proba\_ [private]

#### 12.75.4.2 double spot::ltl::random\_ltl::total\_1\_ [private]

#### 12.75.4.3 op\_proba\* spot::ltl::random\_ltl::proba\_2\_ [private]

#### 12.75.4.4 double spot::ltl::random\_ltl::total\_2\_ [private]

#### 12.75.4.5 double spot::ltl::random\_ltl::total\_2\_and\_more\_ [private]

#### 12.75.4.6 const atomic\_prop\_set\* spot::ltl::random\_ltl::ap\_ [private]

The documentation for this class was generated from the following file:

- [ltlvisit/randomltl.hh](#)

## 12.76 spot::ltl::random\_ltl::op\_proba Struct Reference

### Public Types

- typedef [formula](#) (\*)([builder](#) (const [random\\_ltl](#) \*rl, int n)

### Public Member Functions

- void [setup](#) (const char \*name, int min\_n, [builder](#) build)

### Public Attributes

- const char \* [name](#)
- int [min\\_n](#)
- double [proba](#)
- [builder](#) [build](#)

### 12.76.1 Member Typedef Documentation

**12.76.1.1** typedef formula\*(\*) spot::ltl::random\_ltl::op\_proba::builder(const random\_ltl \*rl, int n)

### 12.76.2 Member Function Documentation

**12.76.2.1** void spot::ltl::random\_ltl::op\_proba::setup (const char \* *name*, int *min\_n*, builder *build*)

### 12.76.3 Member Data Documentation

**12.76.3.1** const char\* spot::ltl::random\_ltl::op\_proba::name

**12.76.3.2** int spot::ltl::random\_ltl::op\_proba::min\_n

**12.76.3.3** double spot::ltl::random\_ltl::op\_proba::proba

**12.76.3.4** builder spot::ltl::random\_ltl::op\_proba::build

The documentation for this struct was generated from the following file:

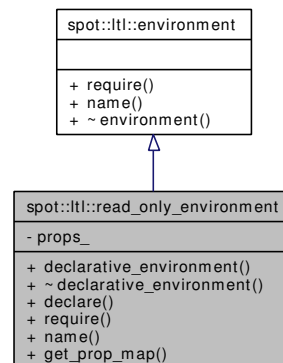
- ltlvisit/[randomltl.hh](#)

## 12.77 spot::ltl::read\_only\_environment Class Reference

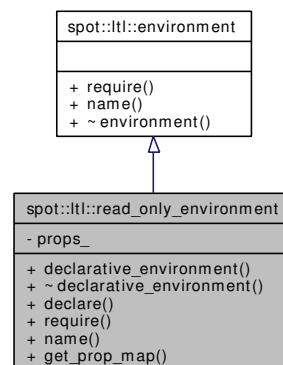
A read only [environment](#).

```
#include <ltlenv/rodeco.hh>
```

Inheritance diagram for spot::ltl::read\_only\_environment:



Collaboration diagram for spot::ltl::read\_only\_environment:



## Public Types

- `typedef std::map< const std::string, ltl::atomic\_prop * > prop\_map`

## Public Member Functions

- `declarative\_environment ()`
- `~declarative\_environment ()`
- `bool declare (const std::string &prop_str)`
- `virtual ltl::formula * require (const std::string &prop_str)`  
*Obtain the [formula](#) associated to prop\_str.*
- `virtual const std::string & name ()`  
*Get the name of the [environment](#).*
- `const prop\_map & get\_prop\_map () const`  
*Get the map of atomic proposition known to this [environment](#).*



### Private Attributes

- `prop_map` `props_`

### 12.77.1 Detailed Description

A read only `environment`.

This `environment` decorate another `environment`, and allow only atomic propositions that have been declared therein.

### 12.77.2 Member Typedef Documentation

**12.77.2.1** `typedef std::map<const std::string, ltl::atomic_prop*> spot::ltl::read_only_environment::prop_map`

### 12.77.3 Constructor & Destructor Documentation

**12.77.3.1** `spot::ltl::read_only_environment::~~declarative_environment ()`

### 12.77.4 Member Function Documentation

**12.77.4.1** `spot::ltl::read_only_environment::declarative_environment ()`

**12.77.4.2** `bool spot::ltl::read_only_environment::declare (const std::string & prop_str)`

Declare an atomic proposition. Return false iff the proposition was already declared.

**12.77.4.3** `virtual ltl::formula* spot::ltl::read_only_environment::require (const std::string & prop_str) [virtual]`

Obtain the `formula` associated to `prop_str`.

Usually `prop_str`, is the name of an atomic proposition, and `spot::ltl::require` simply returns the associated `spot::ltl::atomic_prop`.

Note this is not a `const` method. Some environments will "create" the atomic proposition when requested.

We return a `spot::ltl::formula` instead of an `spot::ltl::atomic_prop`, because this will allow nifty tricks (e.g., we could name formulae in an `environment`, and let the parser build a larger tree from these).

### Returns:

0 iff `prop_str` is not part of the `environment`, or the associated `spot::ltl::formula` otherwise.

Implements `spot::ltl::environment`.

**12.77.4.4** `virtual const std::string& spot::ltl::read_only_environment::name () [virtual]`

Get the name of the `environment`.

Implements `spot::ltl::environment`.

#### 12.77.4.5 const prop\_map& spot::ltl::read\_only\_environment::get\_prop\_map () const

Get the map of atomic proposition known to this [environment](#).

### 12.77.5 Member Data Documentation

#### 12.77.5.1 prop\_map spot::ltl::read\_only\_environment::props\_ [private]

The documentation for this class was generated from the following file:

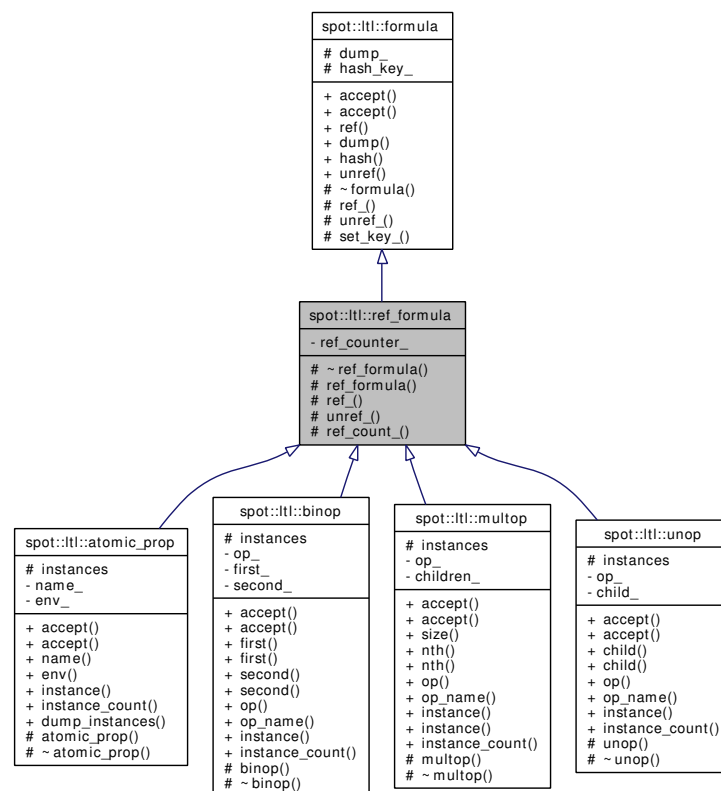
- [ltlenv/rodeco.hh](#)

## 12.78 spot::ltl::ref\_formula Class Reference

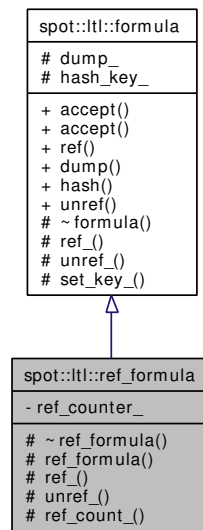
A reference-counted LTL [formula](#).

```
#include <ltlast/refformula.hh>
```

Inheritance diagram for spot::ltl::ref\_formula:



Collaboration diagram for spot::ltl::ref\_formula:



### Public Member Functions

- virtual void `accept (visitor &v)=0`  
*Entry point for vspot::ltl::visitor instances.*
- virtual void `accept (const_visitor &v) const=0`  
*Entry point for vspot::ltl::const\_visitor instances.*
- `formula * ref ()`  
*clone this node*
- const std::string & `dump () const`  
*Return a canonic representation of the *formula*.*
- const size\_t `hash () const`  
*Return a hash\_key for the *formula*.*

### Static Public Member Functions

- static void `unref (formula *f)`  
*release this node*

### Protected Member Functions

- virtual `~ref_formula ()`
- `ref_formula ()`
- void `ref_ ()`  
*increment reference counter if any*

- `bool unref_()`  
*decrement reference counter if any, return true when the instance must be deleted (usually when the counter hits 0).*
- `unsigned ref_count_()`  
*Number of references to this [formula](#).*
- `void set_key_()`  
*Compute key\_ from dump\_.*

### Protected Attributes

- `std::string dump_`  
*The canonic representation of the [formula](#).*
- `size_t hash_key_`  
*The hash key of this [formula](#).*

### Private Attributes

- `unsigned ref_counter_`

## 12.78.1 Detailed Description

A reference-counted LTL [formula](#).

## 12.78.2 Constructor & Destructor Documentation

**12.78.2.1** `virtual spot::ltl::ref_formula::~~ref_formula()` `[protected, virtual]`

**12.78.2.2** `spot::ltl::ref_formula::ref_formula()` `[protected]`

## 12.78.3 Member Function Documentation

**12.78.3.1** `void spot::ltl::ref_formula::ref_()` `[protected, virtual]`

increment reference counter if any

Reimplemented from [spot::ltl::formula](#).

**12.78.3.2** `bool spot::ltl::ref_formula::unref_()` `[protected, virtual]`

decrement reference counter if any, return true when the instance must be deleted (usually when the counter hits 0).

Reimplemented from [spot::ltl::formula](#).

**12.78.3.3** `unsigned spot::ltl::ref_formula::ref_count_()` [protected]

Number of references to this [formula](#).

**12.78.3.4** `virtual void spot::ltl::formula::accept (visitor & v)` [pure virtual, inherited]

Entry point for `vspot::ltl::visitor` instances.

Implemented in [spot::ltl::atomic\\_prop](#), [spot::ltl::binop](#), [spot::ltl::constant](#), [spot::ltl::multop](#), and [spot::ltl::unop](#).

**12.78.3.5** `virtual void spot::ltl::formula::accept (const_visitor & v) const` [pure virtual, inherited]

Entry point for `vspot::ltl::const_visitor` instances.

Implemented in [spot::ltl::atomic\\_prop](#), [spot::ltl::binop](#), [spot::ltl::constant](#), [spot::ltl::multop](#), and [spot::ltl::unop](#).

**12.78.3.6** `formula* spot::ltl::formula::ref ()` [inherited]

clone this node

This increments the reference counter of this node (if one is used). You should almost never use this method directly as it doesn't touch the children. If you want to clone a whole [formula](#), use [spot::ltl::clone\(\)](#) instead.

**12.78.3.7** `static void spot::ltl::formula::unref (formula *f)` [static, inherited]

release this node

This decrements the reference counter of this node (if one is used) and can free the object. You should almost never use this method directly as it doesn't touch the children. If you want to release a whole [formula](#), use [spot::ltl::destroy\(\)](#) instead.

**12.78.3.8** `const std::string& spot::ltl::formula::dump () const` [inherited]

Return a canonic representation of the [formula](#).

**12.78.3.9** `const size_t spot::ltl::formula::hash () const` [inline, inherited]

Return a `hash_key` for the [formula](#).

**12.78.3.10** `void spot::ltl::formula::set_key_()` [protected, inherited]

Compute `key_` from `dump_`.

Should be called once in each object, after `dump_` has been set.

**12.78.4** Member Data Documentation**12.78.4.1** `unsigned spot::ltl::ref_formula::ref_counter_` [private]

**12.78.4.2** `std::string spot::ltl::formula::dump_` [protected, inherited]

The canonic representation of the [formula](#).

**12.78.4.3** `size_t spot::ltl::formula::hash_key_` [protected, inherited]

The hash key of this [formula](#).

Initialized by [set\\_key\\_\(\)](#).

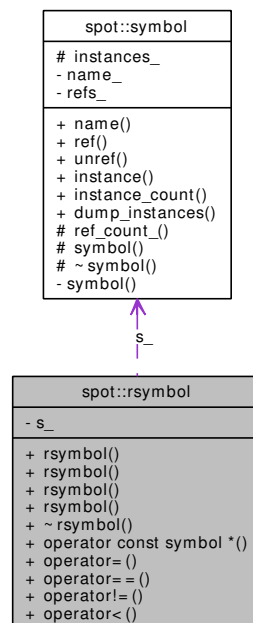
The documentation for this class was generated from the following file:

- [ltlast/refformula.hh](#)

**12.79** `spot::rsymbol` Class Reference

```
#include <evtgba/symbol.hh>
```

Collaboration diagram for `spot::rsymbol`:

**Public Member Functions**

- [rsymbol](#) (const [symbol](#) \*s)
- [rsymbol](#) (const std::string &s)
- [rsymbol](#) (const char \*s)
- [rsymbol](#) (const [rsymbol](#) &rs)
- [~rsymbol](#) ()
- [operator const symbol \\* \(\)](#) const
- const [rsymbol](#) & [operator=](#) (const [rsymbol](#) &rs)
- bool [operator==](#) (const [rsymbol](#) &rs) const
- bool [operator!=](#) (const [rsymbol](#) &rs) const
- bool [operator<](#) (const [rsymbol](#) &rs) const

### Private Attributes

- const [symbol](#) \* [s\\_](#)

### 12.79.1 Constructor & Destructor Documentation

**12.79.1.1** `spot::rsymbol::rsymbol (const symbol * s)` `[inline]`

**12.79.1.2** `spot::rsymbol::rsymbol (const std::string & s)` `[inline]`

**12.79.1.3** `spot::rsymbol::rsymbol (const char * s)` `[inline]`

**12.79.1.4** `spot::rsymbol::rsymbol (const rsymbol & rs)` `[inline]`

**12.79.1.5** `spot::rsymbol::~rsymbol ()` `[inline]`

### 12.79.2 Member Function Documentation

**12.79.2.1** `spot::rsymbol::operator const symbol * () const` `[inline]`

**12.79.2.2** `const rsymbol& spot::rsymbol::operator= (const rsymbol & rs)` `[inline]`

**12.79.2.3** `bool spot::rsymbol::operator== (const rsymbol & rs) const` `[inline]`

**12.79.2.4** `bool spot::rsymbol::operator!= (const rsymbol & rs) const` `[inline]`

**12.79.2.5** `bool spot::rsymbol::operator< (const rsymbol & rs) const` `[inline]`

### 12.79.3 Member Data Documentation

**12.79.3.1** `const symbol* spot::rsymbol::s_` `[private]`

The documentation for this class was generated from the following file:

- [evtgba/symbol.hh](#)

## 12.80 spot::scc\_stack Class Reference

```
#include <tgbaalgos/gtec/sccstack.hh>
```

### Public Types

- typedef std::list< [connected\\_component](#) > [stack\\_type](#)

## Public Member Functions

- void [push](#) (int *index*)  
*Stack a new SCC with index index.*
- [connected\\_component](#) & [top](#) ()  
*Access the top SCC.*
- const [connected\\_component](#) & [top](#) () const  
*Access the top SCC.*
- void [pop](#) ()  
*Pop the top SCC.*
- size\_t [size](#) () const  
*How many SCC are in stack.*
- std::list< const [state](#) \* > & [rem](#) ()  
*The `rem` member of the top SCC.*
- unsigned [clear\\_rem](#) ()
- bool [empty](#) () const  
*Is the stack empty?*

## Public Attributes

- [stack\\_type](#) *s*

## Classes

- struct [connected\\_component](#)

### 12.80.1 Member Typedef Documentation

#### 12.80.1.1 typedef std::list<connected\_component> spot::scc\_stack::stack\_type

### 12.80.2 Member Function Documentation

#### 12.80.2.1 void spot::scc\_stack::push (int *index*)

Stack a new SCC with index *index*.

#### 12.80.2.2 connected\_component& spot::scc\_stack::top ()

Access the top SCC.

#### 12.80.2.3 const connected\_component& spot::scc\_stack::top () const

Access the top SCC.



**12.80.2.4 void spot::scc\_stack::pop ()**

Pop the top SCC.

**12.80.2.5 size\_t spot::scc\_stack::size () const**

How many SCC are in stack.

**12.80.2.6 std::list<const state\*>& spot::scc\_stack::rem ()**

The `rem` member of the top SCC.

**12.80.2.7 unsigned spot::scc\_stack::clear\_rem ()**

Purge all `rem` members.

**Returns:**

the number of elements cleared.

**12.80.2.8 bool spot::scc\_stack::empty () const**

Is the stack empty?

**12.80.3 Member Data Documentation****12.80.3.1 stack\_type spot::scc\_stack::s**

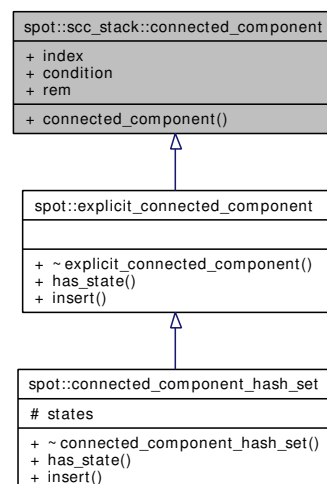
The documentation for this class was generated from the following file:

- [tgbaalgos/gtec/sccstack.hh](#)

**12.81 spot::scc\_stack::connected\_component Struct Reference**

```
#include <tgbaalgos/gtec/sccstack.hh>
```

Inheritance diagram for `spot::scc_stack::connected_component`:



**Public Member Functions**

- [connected\\_component](#) (int *index*=-1)

**Public Attributes**

- int [index](#)  
*Index of the SCC.*
- bdd [condition](#)
- std::list< const [state](#) \* > [rem](#)

**12.81.1 Constructor & Destructor Documentation****12.81.1.1 spot::scc\_stack::connected\_component::connected\_component (int *index* = -1)****12.81.2 Member Data Documentation****12.81.2.1 int spot::scc\_stack::connected\_component::index**

Index of the SCC.

**12.81.2.2 bdd spot::scc\_stack::connected\_component::condition**

The bdd condition is the union of all acceptance conditions of transitions which connect the states of the connected component.

**12.81.2.3 std::list<const state\*> spot::scc\_stack::connected\_component::rem**

The documentation for this struct was generated from the following file:

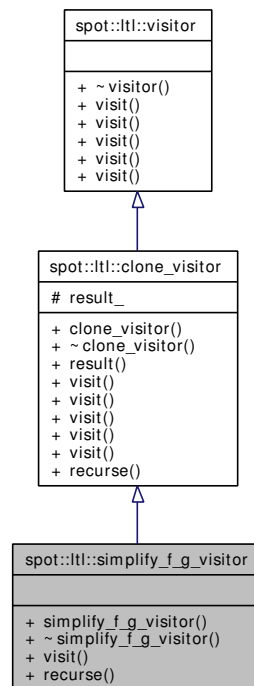
- [tgbaalgos/gtec/sccstack.hh](#)

**12.82 spot::ltl::simplify\_f\_g\_visitor Class Reference**

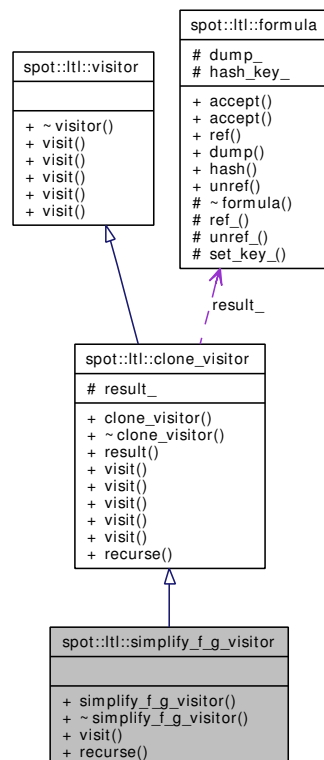
Replace true U f and false R g by F f and G g.

```
#include <ltlvisit/simpfg.hh>
```

Inheritance diagram for spot::ltl::simplify\_f\_g\_visitor:



Collaboration diagram for spot::ltl::simplify\_f\_g\_visitor:



**Public Member Functions**

- [simplify\\_f\\_g\\_visitor \(\)](#)
- virtual [~simplify\\_f\\_g\\_visitor \(\)](#)
- void [visit \(binop \\*bo\)](#)
- virtual [formula \\* recurse \(formula \\*f\)](#)
- [formula \\* result \(\)](#) const
- void [visit \(atomic\\_prop \\*ap\)](#)
- void [visit \(unop \\*uo\)](#)
- void [visit \(multop \\*mo\)](#)
- void [visit \(constant \\*c\)](#)

**Protected Attributes**

- [formula \\* result\\_](#)

**Private Types**

- typedef [clone\\_visitor](#) super

**12.82.1 Detailed Description**

Replace true U f and false R g by F f and G g.

**12.82.2 Member Typedef Documentation**

**12.82.2.1** typedef clone\_visitor spot::ltl::simplify\_f\_g\_visitor::super [private]

**12.82.3 Constructor & Destructor Documentation**

**12.82.3.1** spot::ltl::simplify\_f\_g\_visitor::simplify\_f\_g\_visitor ()

**12.82.3.2** virtual spot::ltl::simplify\_f\_g\_visitor::~~simplify\_f\_g\_visitor () [virtual]

**12.82.4 Member Function Documentation**

**12.82.4.1** void spot::ltl::simplify\_f\_g\_visitor::visit (binop \* *bo*) [virtual]

Reimplemented from [spot::ltl::clone\\_visitor](#).

**12.82.4.2** virtual formula\* spot::ltl::simplify\_f\_g\_visitor::recurse (formula \**f*) [virtual]

Reimplemented from [spot::ltl::clone\\_visitor](#).

**12.82.4.3** formula\* spot::ltl::clone\_visitor::result () const [inherited]

**12.82.4.4** void spot::ltl::clone\_visitor::visit (atomic\_prop \**ap*) [virtual, inherited]

Implements [spot::ltl::visitor](#).

**12.82.4.5** void spot::ltl::clone\_visitor::visit (unop \* *uo*) [virtual, inherited]

Implements [spot::ltl::visitor](#).

Reimplemented in [spot::ltl::unabbreviate\\_ltl\\_visitor](#).

**12.82.4.6** void spot::ltl::clone\_visitor::visit (multop \* *mo*) [virtual, inherited]

Implements [spot::ltl::visitor](#).

**12.82.4.7** void spot::ltl::clone\_visitor::visit (constant \* *c*) [virtual, inherited]

Implements [spot::ltl::visitor](#).

## 12.82.5 Member Data Documentation

**12.82.5.1** formula\* spot::ltl::clone\_visitor::result\_ [protected, inherited]

The documentation for this class was generated from the following file:

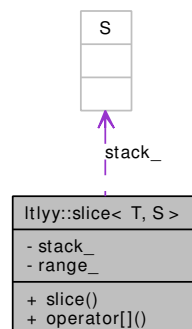
- ltlvisit/[simpfg.hh](#)

## 12.83 Itlly::slice< T, S > Class Template Reference

Present a [slice](#) of the top of a [stack](#).

```
#include <ltlparse/stack.hh>
```

Collaboration diagram for Itlly::slice< T, S >:



### Public Member Functions

- [slice](#) (const S &[stack](#), unsigned int range)
- const T & [operator\[\]](#) (unsigned int i) const

### Private Attributes

- const S & [stack\\_](#)
- unsigned int [range\\_](#)

### 12.83.1 Detailed Description

`template<class T, class S = stack<T>> class ltlyy::slice< T, S >`

Present a [slice](#) of the top of a [stack](#).

### 12.83.2 Constructor & Destructor Documentation

**12.83.2.1** `template<class T, class S = stack<T>> ltlyy::slice< T, S >::slice (const S & stack, unsigned int range)` `[inline]`

### 12.83.3 Member Function Documentation

**12.83.3.1** `template<class T, class S = stack<T>> const T& ltlyy::slice< T, S >::operator[] (unsigned int i) const` `[inline]`

### 12.83.4 Member Data Documentation

**12.83.4.1** `template<class T, class S = stack<T>> const S& ltlyy::slice< T, S >::stack_` `[private]`

**12.83.4.2** `template<class T, class S = stack<T>> unsigned int ltlyy::slice< T, S >::range_` `[private]`

The documentation for this class was generated from the following file:

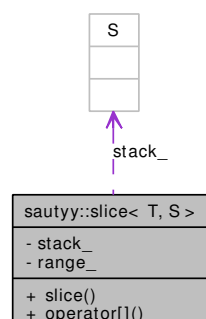
- [ltlparse/stack.hh](#)

## 12.84 sautyy::slice< T, S > Class Template Reference

Present a [slice](#) of the top of a [stack](#).

```
#include <sautparse/stack.hh>
```

Collaboration diagram for sautyy::slice< T, S >:



### Public Member Functions

- [slice](#) (const S &[stack](#), unsigned int range)

- const T & [operator\[\]](#) (unsigned int i) const

#### Private Attributes

- const S & [stack\\_](#)
- unsigned int [range\\_](#)

#### 12.84.1 Detailed Description

**template<class T, class S = stack<T>> class sautyy::slice< T, S >**

Present a [slice](#) of the top of a [stack](#).

#### 12.84.2 Constructor & Destructor Documentation

**12.84.2.1 template<class T, class S = stack<T>> sautyy::slice< T, S >::slice (const S & *stack*, unsigned int *range*) [inline]**

#### 12.84.3 Member Function Documentation

**12.84.3.1 template<class T, class S = stack<T>> const T& sautyy::slice< T, S >::operator[] (unsigned int *i*) const [inline]**

#### 12.84.4 Member Data Documentation

**12.84.4.1 template<class T, class S = stack<T>> const S& sautyy::slice< T, S >::stack\_ [private]**

**12.84.4.2 template<class T, class S = stack<T>> unsigned int sautyy::slice< T, S >::range\_ [private]**

The documentation for this class was generated from the following file:

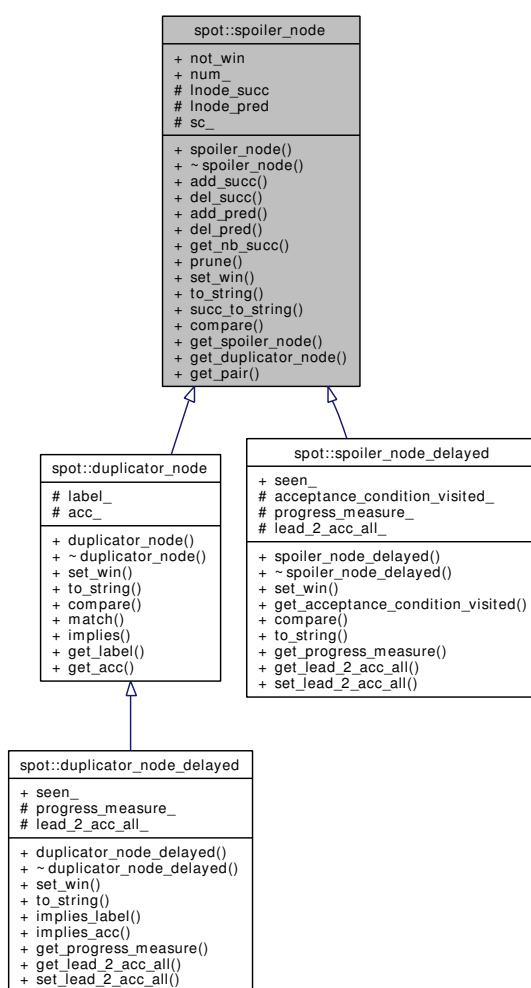
- sautparse/[stack.hh](#)

## 12.85 spot::spoiler\_node Class Reference

Spoiler node of parity game graph.

```
#include <tgbalgorithms/reductgba_sim.hh>
```

Inheritance diagram for spot::spoiler\_node:



## Public Member Functions

- `spoiler_node` (const `state` \*d\_node, const `state` \*s\_node, int num)
- virtual `~spoiler_node` ()
- bool `add_succ` (spoiler\_node \*n)  
Add a successor. Return true if n wasn't yet in the list of successor, false otherwise.
- void `del_succ` (spoiler\_node \*n)
- virtual void `add_pred` (spoiler\_node \*n)
- virtual void `del_pred` ()
- int `get_nb_succ` ()
- bool `prune` ()
- virtual bool `set_win` ()
- virtual std::string `to_string` (const `tgba` \*a)
- virtual std::string `succ_to_string` ()
- virtual bool `compare` (spoiler\_node \*n)
- const `state` \* `get_spoiler_node` ()



- `const state * get_duplicator_node ()`
- `state_couple * get_pair ()`

### Public Attributes

- `bool not_win`
- `int num_`

### Protected Attributes

- `sn_v * lnode_succ`
- `sn_v * lnode_pred`
- `state_couple * sc_`

## 12.85.1 Detailed Description

Spoiler node of parity game graph.

## 12.85.2 Constructor & Destructor Documentation

**12.85.2.1** `spot::spoiler_node::spoiler_node (const state * d_node, const state * s_node, int num)`

**12.85.2.2** `virtual spot::spoiler_node::~~spoiler_node ()` [virtual]

## 12.85.3 Member Function Documentation

**12.85.3.1** `bool spot::spoiler_node::add_succ (spoiler_node * n)`

Add a successor. Return true if *n* wasn't yet in the list of successor, false otherwise.

**12.85.3.2** `void spot::spoiler_node::del_succ (spoiler_node * n)`

**12.85.3.3** `virtual void spot::spoiler_node::add_pred (spoiler_node * n)` [virtual]

**12.85.3.4** `virtual void spot::spoiler_node::del_pred ()` [virtual]

**12.85.3.5** `int spot::spoiler_node::get_nb_succ ()`

**12.85.3.6** `bool spot::spoiler_node::prune ()`

**12.85.3.7** `virtual bool spot::spoiler_node::set_win ()` [virtual]

Reimplemented in [spot::duplicator\\_node](#), [spot::spoiler\\_node\\_delayed](#), and [spot::duplicator\\_node\\_delayed](#).

**12.85.3.8** virtual std::string spot::spoiler\_node::to\_string (const tgba \* *a*) [virtual]

Reimplemented in [spot::duplicator\\_node](#), [spot::spoiler\\_node\\_delayed](#), and [spot::duplicator\\_node\\_delayed](#).

**12.85.3.9** virtual std::string spot::spoiler\_node::succ\_to\_string () [virtual]

**12.85.3.10** virtual bool spot::spoiler\_node::compare (spoiler\_node \* *n*) [virtual]

Reimplemented in [spot::duplicator\\_node](#), and [spot::spoiler\\_node\\_delayed](#).

**12.85.3.11** const state\* spot::spoiler\_node::get\_spoiler\_node ()

**12.85.3.12** const state\* spot::spoiler\_node::get\_duplicator\_node ()

**12.85.3.13** state\_couple\* spot::spoiler\_node::get\_pair ()

## 12.85.4 Member Data Documentation

**12.85.4.1** bool spot::spoiler\_node::not\_win

**12.85.4.2** int spot::spoiler\_node::num\_

**12.85.4.3** sn\_v\* spot::spoiler\_node::lnode\_succ [protected]

**12.85.4.4** sn\_v\* spot::spoiler\_node::lnode\_pred [protected]

**12.85.4.5** state\_couple\* spot::spoiler\_node::sc\_ [protected]

The documentation for this class was generated from the following file:

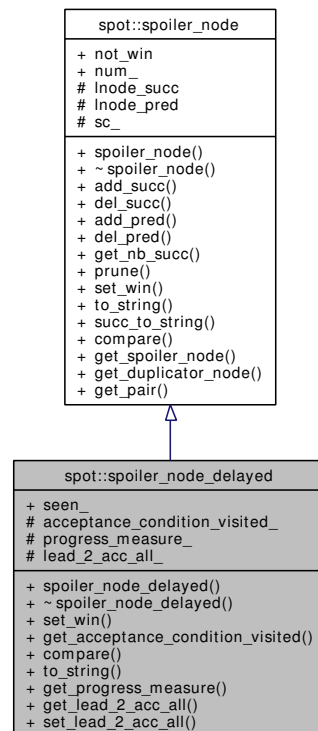
- tgbaalgos/[reductgba\\_sim.hh](#)

## 12.86 spot::spoiler\_node\_delayed Class Reference

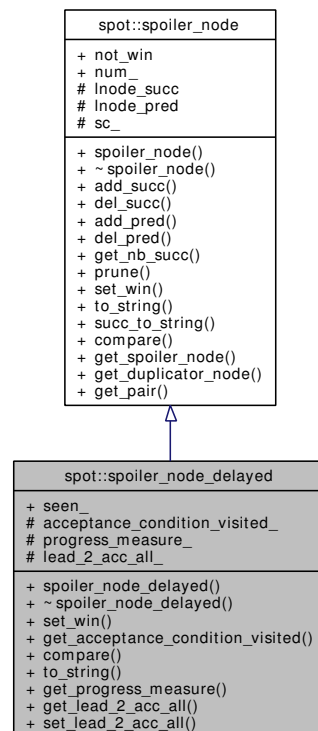
Spoiler node of parity game graph for delayed simulation.

```
#include <tgbaalgos/reductgba_sim.hh>
```

Inheritance diagram for spot::spoiler\_node\_delayed:



Collaboration diagram for spot::spoiler\_node\_delayed:



### Public Member Functions

- [spoiler\\_node\\_delayed](#) (const [state](#) \*d\_node, const [state](#) \*s\_node, bdd a, int num)
- [~spoiler\\_node\\_delayed](#) ()
- bool [set\\_win](#) ()

*Return true if the progress\_measure has changed.*

- bdd [get\\_acceptance\\_condition\\_visited](#) () const
- virtual bool [compare](#) ([spoiler\\_node](#) \*n)
- virtual std::string [to\\_string](#) (const [tgba](#) \*a)
- int [get\\_progress\\_measure](#) () const
- bool [get\\_lead\\_2\\_acc\\_all](#) ()
- bool [set\\_lead\\_2\\_acc\\_all](#) (bdd acc=bddfalse)
- bool [add\\_succ](#) ([spoiler\\_node](#) \*n)

*Add a successor. Return true if n wasn't yet in the list of successor, false otherwise.*

- void [del\\_succ](#) ([spoiler\\_node](#) \*n)
- virtual void [add\\_pred](#) ([spoiler\\_node](#) \*n)
- virtual void [del\\_pred](#) ()
- int [get\\_nb\\_succ](#) ()
- bool [prune](#) ()
- virtual std::string [succ\\_to\\_string](#) ()
- const [state](#) \* [get\\_spoiler\\_node](#) ()
- const [state](#) \* [get\\_duplicator\\_node](#) ()
- [state\\_couple](#) \* [get\\_pair](#) ()

### Public Attributes

- bool [seen\\_](#)
- bool [not\\_win](#)
- int [num\\_](#)

### Protected Attributes

- bdd [acceptance\\_condition\\_visited\\_](#)
- int [progress\\_measure\\_](#)
- bool [lead\\_2\\_acc\\_all\\_](#)
- [sn\\_v](#) \* [lnode\\_succ](#)
- [sn\\_v](#) \* [lnode\\_pred](#)
- [state\\_couple](#) \* [sc\\_](#)

#### 12.86.1 Detailed Description

Spoiler node of parity game graph for delayed simulation.

#### 12.86.2 Constructor & Destructor Documentation

**12.86.2.1** [spot::spoiler\\_node\\_delayed::spoiler\\_node\\_delayed](#) (const [state](#) \* *d\_node*, const [state](#) \* *s\_node*, bdd *a*, int *num*)

### 12.86.2.2 `spot::spoiler_node_delayed::~~spoiler_node_delayed ()`

## 12.86.3 Member Function Documentation

### 12.86.3.1 `bool spot::spoiler_node_delayed::set_win ()` [virtual]

Return true if the `progress_measure` has changed.

Reimplemented from [spot::spoiler\\_node](#).

### 12.86.3.2 `bdd spot::spoiler_node_delayed::get_acceptance_condition_visited () const`

### 12.86.3.3 `virtual bool spot::spoiler_node_delayed::compare (spoiler_node * n)` [virtual]

Reimplemented from [spot::spoiler\\_node](#).

### 12.86.3.4 `virtual std::string spot::spoiler_node_delayed::to_string (const tgba * a)` [virtual]

Reimplemented from [spot::spoiler\\_node](#).

### 12.86.3.5 `int spot::spoiler_node_delayed::get_progress_measure () const`

### 12.86.3.6 `bool spot::spoiler_node_delayed::get_lead_2_acc_all ()`

### 12.86.3.7 `bool spot::spoiler_node_delayed::set_lead_2_acc_all (bdd acc = bddfalse)`

### 12.86.3.8 `bool spot::spoiler_node::add_succ (spoiler_node * n)` [inherited]

Add a successor. Return true if *n* wasn't yet in the list of successor, false otherwise.

### 12.86.3.9 `void spot::spoiler_node::del_succ (spoiler_node * n)` [inherited]

### 12.86.3.10 `virtual void spot::spoiler_node::add_pred (spoiler_node * n)` [virtual, inherited]

### 12.86.3.11 `virtual void spot::spoiler_node::del_pred ()` [virtual, inherited]

### 12.86.3.12 `int spot::spoiler_node::get_nb_succ ()` [inherited]

### 12.86.3.13 `bool spot::spoiler_node::prune ()` [inherited]

### 12.86.3.14 `virtual std::string spot::spoiler_node::succ_to_string ()` [virtual, inherited]

### 12.86.3.15 `const state* spot::spoiler_node::get_spoiler_node ()` [inherited]

**12.86.3.16** `const state* spot::spoiler_node::get_duplicator_node ()` [inherited]

**12.86.3.17** `state_couple* spot::spoiler_node::get_pair ()` [inherited]

## 12.86.4 Member Data Documentation

**12.86.4.1** `bool spot::spoiler_node_delayed::seen_`

**12.86.4.2** `bdd spot::spoiler_node_delayed::acceptance_condition_visited_` [protected]

a Bdd for retain all the acceptance condition that a node has visited.

**12.86.4.3** `int spot::spoiler_node_delayed::progress_measure_` [protected]

**12.86.4.4** `bool spot::spoiler_node_delayed::lead_2_acc_all_` [protected]

**12.86.4.5** `bool spot::spoiler_node::not_win` [inherited]

**12.86.4.6** `int spot::spoiler_node::num_` [inherited]

**12.86.4.7** `sn_v* spot::spoiler_node::lnode_succ` [protected, inherited]

**12.86.4.8** `sn_v* spot::spoiler_node::lnode_pred` [protected, inherited]

**12.86.4.9** `state_couple* spot::spoiler_node::sc_` [protected, inherited]

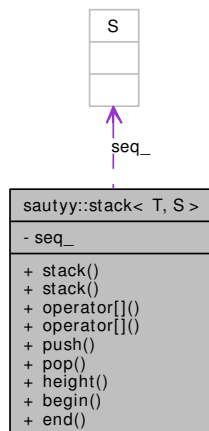
The documentation for this class was generated from the following file:

- `tgbaalgorithms/reductgba_sim.hh`

## 12.87 sautyy::stack< T, S > Class Template Reference

```
#include <sautyparse/stack.hh>
```

Collaboration diagram for sautyy::stack< T, S >:



### Public Types

- typedef S::reverse\_iterator [iterator](#)
- typedef S::const\_reverse\_iterator [const\\_iterator](#)

### Public Member Functions

- [stack](#) ()
- [stack](#) (unsigned int n)
- T & [operator](#)[ ] (unsigned int i)
- const T & [operator](#)[ ] (unsigned int i) const
- void [push](#) (const T &t)
- void [pop](#) (unsigned int n=1)
- unsigned int [height](#) () const
- [const\\_iterator](#) [begin](#) () const
- [const\\_iterator](#) [end](#) () const

### Private Attributes

- S [seq\\_](#)

```
template<class T, class S = std::deque<T>> class sautyy::stack< T, S >
```

#### 12.87.1 Member Typedef Documentation

**12.87.1.1** template<class T, class S = std::deque<T>> typedef S::reverse\_iterator sautyy::stack< T, S >::iterator

**12.87.1.2** template<class T, class S = std::deque<T>> typedef S::const\_reverse\_iterator sautyy::stack< T, S >::const\_iterator

## 12.87.2 Constructor & Destructor Documentation

**12.87.2.1** `template<class T, class S = std::deque<T>> sautyy::stack< T, S >::stack ()` `[inline]`

**12.87.2.2** `template<class T, class S = std::deque<T>> sautyy::stack< T, S >::stack (unsigned int n)` `[inline]`

## 12.87.3 Member Function Documentation

**12.87.3.1** `template<class T, class S = std::deque<T>> T& sautyy::stack< T, S >::operator[] (unsigned int i)` `[inline]`

**12.87.3.2** `template<class T, class S = std::deque<T>> const T& sautyy::stack< T, S >::operator[] (unsigned int i) const` `[inline]`

**12.87.3.3** `template<class T, class S = std::deque<T>> void sautyy::stack< T, S >::push (const T & t)` `[inline]`

**12.87.3.4** `template<class T, class S = std::deque<T>> void sautyy::stack< T, S >::pop (unsigned int n = 1)` `[inline]`

**12.87.3.5** `template<class T, class S = std::deque<T>> unsigned int sautyy::stack< T, S >::height () const` `[inline]`

**12.87.3.6** `template<class T, class S = std::deque<T>> const_iterator sautyy::stack< T, S >::begin () const` `[inline]`

**12.87.3.7** `template<class T, class S = std::deque<T>> const_iterator sautyy::stack< T, S >::end () const` `[inline]`

## 12.87.4 Member Data Documentation

**12.87.4.1** `template<class T, class S = std::deque<T>> S sautyy::stack< T, S >::seq_` `[private]`

The documentation for this class was generated from the following file:

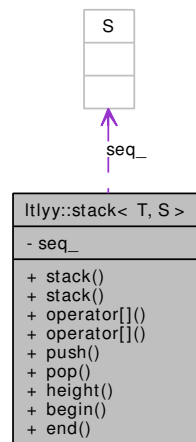
- sautparse/[stack.hh](#)

## 12.88 Itlly::stack< T, S > Class Template Reference

```
#include <ltlparse/stack.hh>
```



Collaboration diagram for ltlyy::stack< T, S >:



## Public Types

- typedef S::reverse\_iterator [iterator](#)
- typedef S::const\_reverse\_iterator [const\\_iterator](#)

## Public Member Functions

- [stack](#) ()
- [stack](#) (unsigned int n)
- T & [operator\[\]](#) (unsigned int i)
- const T & [operator\[\]](#) (unsigned int i) const
- void [push](#) (const T &t)
- void [pop](#) (unsigned int n=1)
- unsigned int [height](#) () const
- [const\\_iterator](#) [begin](#) () const
- [const\\_iterator](#) [end](#) () const

## Private Attributes

- S [seq\\_](#)

```
template<class T, class S = std::deque<T>> class ltlyy::stack< T, S >
```

### 12.88.1 Member Typedef Documentation

**12.88.1.1** template<class T, class S = std::deque<T>> typedef S::reverse\_iterator ltlyy::stack< T, S >::iterator

**12.88.1.2** template<class T, class S = std::deque<T>> typedef S::const\_reverse\_iterator ltlyy::stack< T, S >::const\_iterator

## 12.88.2 Constructor & Destructor Documentation

**12.88.2.1** `template<class T, class S = std::deque<T>> ltlyy::stack< T, S >::stack ()` `[inline]`

**12.88.2.2** `template<class T, class S = std::deque<T>> ltlyy::stack< T, S >::stack (unsigned int n)` `[inline]`

## 12.88.3 Member Function Documentation

**12.88.3.1** `template<class T, class S = std::deque<T>> T& ltlyy::stack< T, S >::operator[] (unsigned int i)` `[inline]`

**12.88.3.2** `template<class T, class S = std::deque<T>> const T& ltlyy::stack< T, S >::operator[] (unsigned int i) const` `[inline]`

**12.88.3.3** `template<class T, class S = std::deque<T>> void ltlyy::stack< T, S >::push (const T & t)` `[inline]`

**12.88.3.4** `template<class T, class S = std::deque<T>> void ltlyy::stack< T, S >::pop (unsigned int n = 1)` `[inline]`

**12.88.3.5** `template<class T, class S = std::deque<T>> unsigned int ltlyy::stack< T, S >::height () const` `[inline]`

**12.88.3.6** `template<class T, class S = std::deque<T>> const_iterator ltlyy::stack< T, S >::begin () const` `[inline]`

**12.88.3.7** `template<class T, class S = std::deque<T>> const_iterator ltlyy::stack< T, S >::end () const` `[inline]`

## 12.88.4 Member Data Documentation

**12.88.4.1** `template<class T, class S = std::deque<T>> S ltlyy::stack< T, S >::seq_` `[private]`

The documentation for this class was generated from the following file:

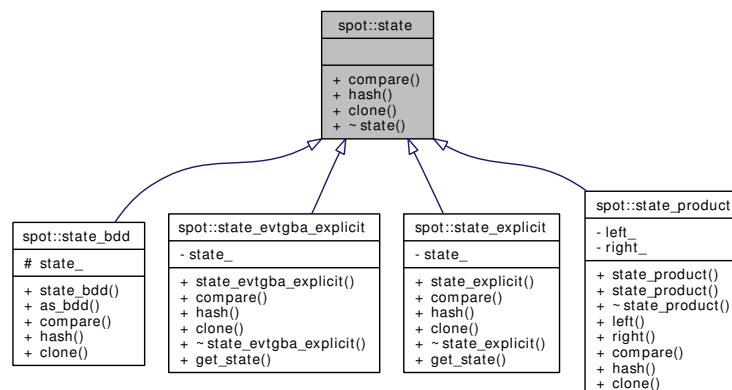
- [ltlparse/stack.hh](#)

## 12.89 spot::state Class Reference

Abstract class for states.

```
#include <tgba/state.hh>
```

Inheritance diagram for spot::state:



## Public Member Functions

- virtual int `compare` (const `state` \*other) const=0  
*Compares two states (that come from the same automaton).*
- virtual size\_t `hash` () const=0  
*Hash a `state`.*
- virtual `state` \* `clone` () const=0  
*Duplicate a `state`.*
- virtual `~state` ()

### 12.89.1 Detailed Description

Abstract class for states.

### 12.89.2 Constructor & Destructor Documentation

#### 12.89.2.1 virtual `spot::state::~~state` () [inline, virtual]

### 12.89.3 Member Function Documentation

#### 12.89.3.1 virtual int `spot::state::compare` (const `state` \*other) const [pure virtual]

Compares two states (that come from the same automaton).

This method returns an integer less than, equal to, or greater than zero if *this* is found, respectively, to be less than, equal to, or greater than *other* according to some implicit total order.

This method should not be called to compare states from different automata.

See also:

[spot::state\\_ptr\\_less\\_than](#)

Implemented in `spot::state_evtgba_explicit`, `spot::state_bdd`, `spot::state_explicit`, and `spot::state_product`.

**12.89.3.2 virtual size\_t spot::state::hash () const** [pure virtual]

Hash a [state](#).

This method returns an integer that can be used as a hash value for this [state](#).

Note that the hash value is guaranteed to be unique for all equal states (in [compare\(\)](#)'s sense) for only has long has one of these states exists. So it's OK to use a [spot::state](#) as a key in a `hash_map` because the mere use of the [state](#) as a key in the hash will ensure the [state](#) continues to exist.

However if you create the [state](#), get its hash key, delete the [state](#), recreate the same [state](#), and get its hash key, you may obtain two different hash keys if the same [state](#) were not already used elsewhere. In practice this weird situation can occur only when the [state](#) is BDD-encoded, because BDD numbers (used to build the hash value) can be reused for other formulas. That probably doesn't matter, since the hash value is meant to be used in a `hash_map`, but it had to be noted.

Implemented in [spot::state\\_evtgba\\_explicit](#), [spot::state\\_bdd](#), [spot::state\\_explicit](#), and [spot::state\\_product](#).

**12.89.3.3 virtual state\* spot::state::clone () const** [pure virtual]

Duplicate a [state](#).

Implemented in [spot::state\\_evtgba\\_explicit](#), [spot::state\\_bdd](#), [spot::state\\_explicit](#), and [spot::state\\_product](#).

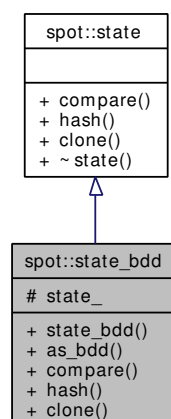
The documentation for this class was generated from the following file:

- [tgba/state.hh](#)

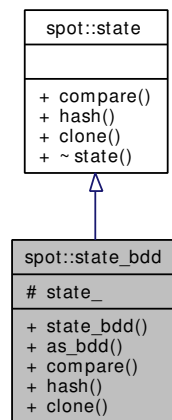
**12.90 spot::state\_bdd Class Reference**

```
#include <tgba/statebdd.hh>
```

Inheritance diagram for `spot::state_bdd`:



Collaboration diagram for spot::state\_bdd:



### Public Member Functions

- `state_bdd` (bdd s)
- virtual bdd `as_bdd` () const  
*Return the BDD part of the `state`.*
- virtual int `compare` (const `state` \*other) const  
*Compares two states (that come from the same automaton).*
- virtual size\_t `hash` () const  
*Hash a `state`.*
- virtual `state_bdd` \* `clone` () const  
*Duplicate a `state`.*

### Protected Attributes

- bdd `state_`  
*BDD representation of the `state`.*

#### 12.90.1 Detailed Description

A `state` whose representation is a BDD.

#### 12.90.2 Constructor & Destructor Documentation

##### 12.90.2.1 spot::state\_bdd::state\_bdd (bdd s) [inline]

### 12.90.3 Member Function Documentation

#### 12.90.3.1 virtual bdd spot::state\_bdd::as\_bdd () const [inline, virtual]

Return the BDD part of the [state](#).

#### 12.90.3.2 virtual int spot::state\_bdd::compare (const state \* *other*) const [virtual]

Compares two states (that come from the same automaton).

This method returns an integer less than, equal to, or greater than zero if *this* is found, respectively, to be less than, equal to, or greater than *other* according to some implicit total order.

This method should not be called to compare states from different automata.

See also:

[spot::state\\_ptr\\_less\\_than](#)

Implements [spot::state](#).

#### 12.90.3.3 virtual size\_t spot::state\_bdd::hash () const [virtual]

Hash a [state](#).

This method returns an integer that can be used as a hash value for this [state](#).

Note that the hash value is guaranteed to be unique for all equal states (in [compare\(\)](#)'s sense) for only has long has one of these states exists. So it's OK to use a [spot::state](#) as a key in a `hash_map` because the mere use of the [state](#) as a key in the hash will ensure the [state](#) continues to exist.

However if you create the [state](#), get its hash key, delete the [state](#), recreate the same [state](#), and get its hash key, you may obtain two different hash keys if the same [state](#) were not already used elsewhere. In practice this weird situation can occur only when the [state](#) is BDD-encoded, because BDD numbers (used to build the hash value) can be reused for other formulas. That probably doesn't matter, since the hash value is meant to be used in a `hash_map`, but it had to be noted.

Implements [spot::state](#).

#### 12.90.3.4 virtual state\_bdd\* spot::state\_bdd::clone () const [virtual]

Duplicate a [state](#).

Implements [spot::state](#).

### 12.90.4 Member Data Documentation

#### 12.90.4.1 bdd spot::state\_bdd::state\_ [protected]

BDD representation of the [state](#).

The documentation for this class was generated from the following file:

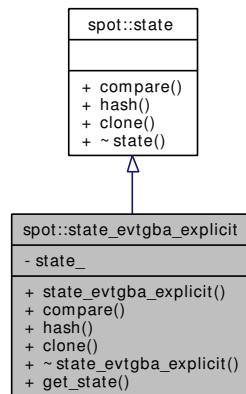
- [tgba/statebdd.hh](#)

## 12.91 spot::state\_evtgba\_explicit Class Reference

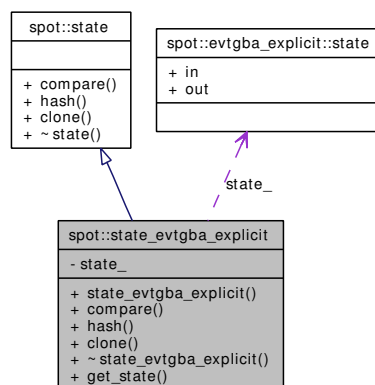
States used by `spot::tgba_evtgba_explicit`.

```
#include <evtgba/explicit.hh>
```

Inheritance diagram for spot::state\_evtgba\_explicit:



Collaboration diagram for spot::state\_evtgba\_explicit:



## Public Member Functions

- [state\\_evtgba\\_explicit](#) (const [evtgba\\_explicit::state](#) \*s)
- virtual int [compare](#) (const [spot::state](#) \*other) const  
*Compares two states (that come from the same automaton).*
- virtual size\_t [hash](#) () const  
*Hash a [state](#).*
- virtual [state\\_evtgba\\_explicit](#) \* [clone](#) () const  
*Duplicate a [state](#).*
- virtual [~state\\_evtgba\\_explicit](#) ()
- const [evtgba\\_explicit::state](#) \* [get\\_state](#) () const

## Private Attributes

- const [evtgba\\_explicit::state](#) \* [state\\_](#)

### 12.91.1 Detailed Description

States used by `spot::tgba_evtgba_explicit`.

### 12.91.2 Constructor & Destructor Documentation

**12.91.2.1** `spot::state_evtgba_explicit::state_evtgba_explicit (const evtgba_explicit::state * s)`  
[inline]

**12.91.2.2** `virtual spot::state_evtgba_explicit::~state_evtgba_explicit ()` [inline, virtual]

### 12.91.3 Member Function Documentation

**12.91.3.1** `virtual int spot::state_evtgba_explicit::compare (const spot::state * other) const`  
[virtual]

Compares two states (that come from the same automaton).

This method returns an integer less than, equal to, or greater than zero if *this* is found, respectively, to be less than, equal to, or greater than *other* according to some implicit total order.

This method should not be called to compare states from different automata.

See also:

[spot::state\\_ptr\\_less\\_than](#)

Implements [spot::state](#).

**12.91.3.2** `virtual size_t spot::state_evtgba_explicit::hash () const` [virtual]

Hash a [state](#).

This method returns an integer that can be used as a hash value for this [state](#).

Note that the hash value is guaranteed to be unique for all equal states (in [compare\(\)](#)'s sense) for only has long has one of these states exists. So it's OK to use a [spot::state](#) as a key in a `hash_map` because the mere use of the [state](#) as a key in the hash will ensure the [state](#) continues to exist.

However if you create the [state](#), get its hash key, delete the [state](#), recreate the same [state](#), and get its hash key, you may obtain two different hash keys if the same [state](#) were not already used elsewhere. In practice this weird situation can occur only when the [state](#) is BDD-encoded, because BDD numbers (used to build the hash value) can be reused for other formulas. That probably doesn't matter, since the hash value is meant to be used in a `hash_map`, but it had to be noted.

Implements [spot::state](#).

**12.91.3.3** `virtual state_evtgba_explicit* spot::state_evtgba_explicit::clone () const` [virtual]

Duplicate a [state](#).

Implements [spot::state](#).

**12.91.3.4** `const evtgba_explicit::state* spot::state_evtgba_explicit::get_state () const`



### 12.91.4 Member Data Documentation

#### 12.91.4.1 const evtgba\_explicit::state\* spot::state\_evtgba\_explicit::state\_ [private]

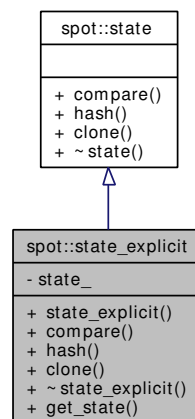
The documentation for this class was generated from the following file:

- [evtgba/explicit.hh](#)

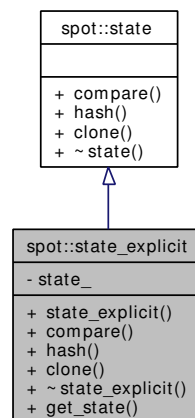
## 12.92 spot::state\_explicit Class Reference

```
#include <tgba/tgbaexplicit.hh>
```

Inheritance diagram for spot::state\_explicit:



Collaboration diagram for spot::state\_explicit:



### Public Member Functions

- `state_explicit` (const `tgba_explicit::state` \*s)
- virtual int `compare` (const `spot::state` \*other) const  
*Compares two states (that come from the same automaton).*

- virtual `size_t hash () const`  
*Hash a [state](#).*
- virtual `state_explicit * clone () const`  
*Duplicate a [state](#).*
- virtual `~state_explicit ()`
- const `tgba_explicit::state * get_state () const`

### Private Attributes

- const `tgba_explicit::state * state_`

#### 12.92.1 Detailed Description

States used by `spot::tgba_explicit`.

#### 12.92.2 Constructor & Destructor Documentation

**12.92.2.1** `spot::state_explicit::state_explicit (const tgba_explicit::state * s) [inline]`

**12.92.2.2** `virtual spot::state_explicit::~state_explicit () [inline, virtual]`

#### 12.92.3 Member Function Documentation

**12.92.3.1** `virtual int spot::state_explicit::compare (const spot::state * other) const [virtual]`

Compares two states (that come from the same automaton).

This method returns an integer less than, equal to, or greater than zero if *this* is found, respectively, to be less than, equal to, or greater than *other* according to some implicit total order.

This method should not be called to compare states from different automata.

See also:

[spot::state\\_ptr\\_less\\_than](#)

Implements [spot::state](#).

**12.92.3.2** `virtual size_t spot::state_explicit::hash () const [virtual]`

Hash a [state](#).

This method returns an integer that can be used as a hash value for this [state](#).

Note that the hash value is guaranteed to be unique for all equal states (in `compare()`'s sense) for only has long has one of these states exists. So it's OK to use a [spot::state](#) as a key in a `hash_map` because the mere use of the [state](#) as a key in the hash will ensure the [state](#) continues to exist.

However if you create the [state](#), get its hash key, delete the [state](#), recreate the same [state](#), and get its hash key, you may obtain two different hash keys if the same [state](#) were not already used elsewhere. In practice this weird situation can occur only when the [state](#) is BDD-encoded, because BDD numbers (used to build

the hash value) can be reused for other formulas. That probably doesn't matter, since the hash value is meant to be used in a `hash_map`, but it had to be noted.

Implements [spot::state](#).

### 12.92.3.3 virtual state\_explicit\* spot::state\_explicit::clone() const [virtual]

Duplicate a [state](#).

Implements [spot::state](#).

### 12.92.3.4 const tgba\_explicit::state\* spot::state\_explicit::get\_state() const

## 12.92.4 Member Data Documentation

### 12.92.4.1 const tgba\_explicit::state\* spot::state\_explicit::state\_ [private]

The documentation for this class was generated from the following file:

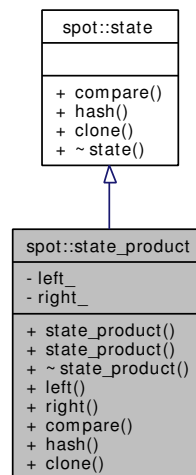
- [tgba/tgbaexplicit.hh](#)

## 12.93 spot::state\_product Class Reference

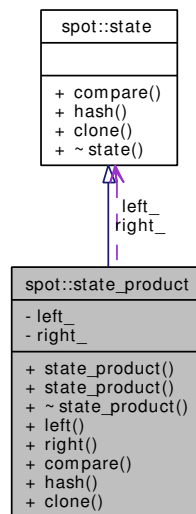
A [state](#) for [spot::tgba\\_product](#).

```
#include <tgba/tgbaproduct.hh>
```

Inheritance diagram for `spot::state_product`:



Collaboration diagram for spot::state\_product:



## Public Member Functions

- `state_product (state *left, state *right)`  
*Constructor.*
- `state_product (const state_product &o)`  
*Copy constructor.*
- `virtual ~state_product ()`
- `state * left () const`
- `state * right () const`
- `virtual int compare (const state *other) const`  
*Compares two states (that come from the same automaton).*
- `virtual size_t hash () const`  
*Hash a [state](#).*
- `virtual state_product * clone () const`  
*Duplicate a [state](#).*

## Private Attributes

- `state * left_`  
*State from the left automaton.*
- `state * right_`  
*State from the right automaton.*

### 12.93.1 Detailed Description

A [state](#) for [spot::tgba\\_product](#).

This [state](#) is in fact a pair of [state](#): the [state](#) from the left automaton and that of the right.

### 12.93.2 Constructor & Destructor Documentation

#### 12.93.2.1 spot::state\_product::state\_product (state \* *left*, state \* *right*) [inline]

Constructor.

##### Parameters:

*left* The [state](#) from the left automaton.

*right* The [state](#) from the right automaton. These states are acquired by [spot::state\\_product](#), and will be deleted on destruction.

#### 12.93.2.2 spot::state\_product::state\_product (const state\_product & *o*)

Copy constructor.

#### 12.93.2.3 virtual spot::state\_product::~~state\_product () [virtual]

### 12.93.3 Member Function Documentation

#### 12.93.3.1 state\* spot::state\_product::left () const [inline]

#### 12.93.3.2 state\* spot::state\_product::right () const [inline]

#### 12.93.3.3 virtual int spot::state\_product::compare (const state \* *other*) const [virtual]

Compares two states (that come from the same automaton).

This method returns an integer less than, equal to, or greater than zero if *this* is found, respectively, to be less than, equal to, or greater than *other* according to some implicit total order.

This method should not be called to compare states from different automata.

##### See also:

[spot::state\\_ptr\\_less\\_than](#)

Implements [spot::state](#).

#### 12.93.3.4 virtual size\_t spot::state\_product::hash () const [virtual]

Hash a [state](#).

This method returns an integer that can be used as a hash value for this [state](#).

Note that the hash value is guaranteed to be unique for all equal states (in [compare\(\)](#)'s sense) for only has long has one of these states exists. So it's OK to use a [spot::state](#) as a key in a `hash_map` because the mere use of the [state](#) as a key in the hash will ensure the [state](#) continues to exist.

However if you create the [state](#), get its hash key, delete the [state](#), recreate the same [state](#), and get its hash key, you may obtain two different hash keys if the same [state](#) were not already used elsewhere. In practice this weird situation can occur only when the [state](#) is BDD-encoded, because BDD numbers (used to build the hash value) can be reused for other formulas. That probably doesn't matter, since the hash value is meant to be used in a `hash_map`, but it had to be noted.

Implements [spot::state](#).

#### 12.93.3.5 virtual state\_product\* spot::state\_product::clone () const [virtual]

Duplicate a [state](#).

Implements [spot::state](#).

### 12.93.4 Member Data Documentation

#### 12.93.4.1 state\* spot::state\_product::left\_ [private]

State from the left automaton.

#### 12.93.4.2 state\* spot::state\_product::right\_ [private]

State from the right automaton.

The documentation for this class was generated from the following file:

- [tgba/tgbaproduct.hh](#)

## 12.94 spot::state\_ptr\_equal Struct Reference

An Equivalence Relation for `state*`.

```
#include <tgba/state.hh>
```

### Public Member Functions

- `bool operator() (const state *left, const state *right) const`

#### 12.94.1 Detailed Description

An Equivalence Relation for `state*`.

This is meant to be used as a comparison functor for `Sgi hash_map` whose key are of type `state*`.

For instance here is how one could declare a map of `state*`.

```
// Remember how many times each state has been visited.
Sgi::hash_map<spot::state*, int, spot::state_ptr_hash,
              spot::state_ptr_equal> seen;
```

#### 12.94.2 Member Function Documentation

##### 12.94.2.1 bool spot::state\_ptr\_equal::operator() (const state \* left, const state \* right) const [inline]

The documentation for this struct was generated from the following file:

- [tgba/state.hh](#)

## 12.95 spot::state\_ptr\_hash Struct Reference

Hash Function for state\*.

```
#include <tgba/state.hh>
```

### Public Member Functions

- `size_t operator() (const state *that) const`

#### 12.95.1 Detailed Description

Hash Function for state\*.

This is meant to be used as a hash functor for Sgi's hash\_map whose key are of type state\*.

For instance here is how one could declare a map of state\*.

```
// Remember how many times each state has been visited.
Sgi::hash_map<spot::state*, int, spot::state_ptr_hash,
              spot::state_ptr_equal> seen;
```

#### 12.95.2 Member Function Documentation

##### 12.95.2.1 size\_t spot::state\_ptr\_hash::operator() (const state \* *that*) const [inline]

The documentation for this struct was generated from the following file:

- [tgba/state.hh](#)

## 12.96 spot::state\_ptr\_less\_than Struct Reference

Strict Weak Ordering for state\*.

```
#include <tgba/state.hh>
```

### Public Member Functions

- `bool operator() (const state *left, const state *right) const`

#### 12.96.1 Detailed Description

Strict Weak Ordering for state\*.

This is meant to be used as a comparison functor for STL map whose key are of type state\*.

For instance here is how one could declare a map of state\*.

```
// Remember how many times each state has been visited.
std::map<spot::state*, int, spot::state_ptr_less_than> seen;
```

## 12.96.2 Member Function Documentation

**12.96.2.1** `bool spot::state_ptr_less_than::operator() (const state * left, const state * right) const` `[inline]`

The documentation for this struct was generated from the following file:

- [tgba/state.hh](#)

## 12.97 spot::string\_hash Struct Reference

A hash function for strings.

```
#include <misc/hash.hh>
```

### Public Member Functions

- `size_t operator() (const std::string &s) const`

### 12.97.1 Detailed Description

A hash function for strings.

## 12.97.2 Member Function Documentation

**12.97.2.1** `size_t spot::string_hash::operator() (const std::string &s) const` `[inline]`

The documentation for this struct was generated from the following file:

- [misc/hash.hh](#)

## 12.98 spot::symbol Class Reference

```
#include <evtgba/symbol.hh>
```

### Public Member Functions

- `const std::string & name () const`
- `void ref () const`
- `void unref () const`

### Static Public Member Functions

- `static const symbol * instance (const std::string &name)`
- `static unsigned instance_count ()`  
*Number of instantiated atomic propositions. For debugging.*
- `static std::ostream & dump_instances (std::ostream &os)`  
*List all instances of atomic propositions. For debugging.*



### Protected Types

- typedef std::map< const std::string, const symbol \* > map

### Protected Member Functions

- int ref\_count\_ () const
- symbol (const std::string \*name)
- ~symbol ()

### Static Protected Attributes

- static map instances\_

### Private Member Functions

- symbol (const symbol &)

### Private Attributes

- const std::string \* name\_  
*Undefined.*
- int refs\_

## 12.98.1 Member Typedef Documentation

**12.98.1.1** typedef std::map<const std::string, const symbol\*> spot::symbol::map  
[protected]

## 12.98.2 Constructor & Destructor Documentation

**12.98.2.1** spot::symbol::symbol (const std::string \* name) [protected]

**12.98.2.2** spot::symbol::~symbol () [protected]

**12.98.2.3** spot::symbol::symbol (const symbol &) [private]

## 12.98.3 Member Function Documentation

**12.98.3.1** static const symbol\* spot::symbol::instance (const std::string & name) [static]

**12.98.3.2** const std::string& spot::symbol::name () const

**12.98.3.3** static unsigned spot::symbol::instance\_count () [static]

Number of instantiated atomic propositions. For debugging.

**12.98.3.4 static std::ostream& spot::symbol::dump\_instances (std::ostream & os) [static]**

List all instances of atomic propositions. For debugging.

**12.98.3.5 void spot::symbol::ref () const****12.98.3.6 void spot::symbol::unref () const****12.98.3.7 int spot::symbol::ref\_count\_ () const [protected]****12.98.4 Member Data Documentation****12.98.4.1 map spot::symbol::instances\_ [static, protected]****12.98.4.2 const std::string\* spot::symbol::name\_ [private]**

Undefined.

**12.98.4.3 int spot::symbol::refs\_ [mutable, private]**

The documentation for this class was generated from the following file:

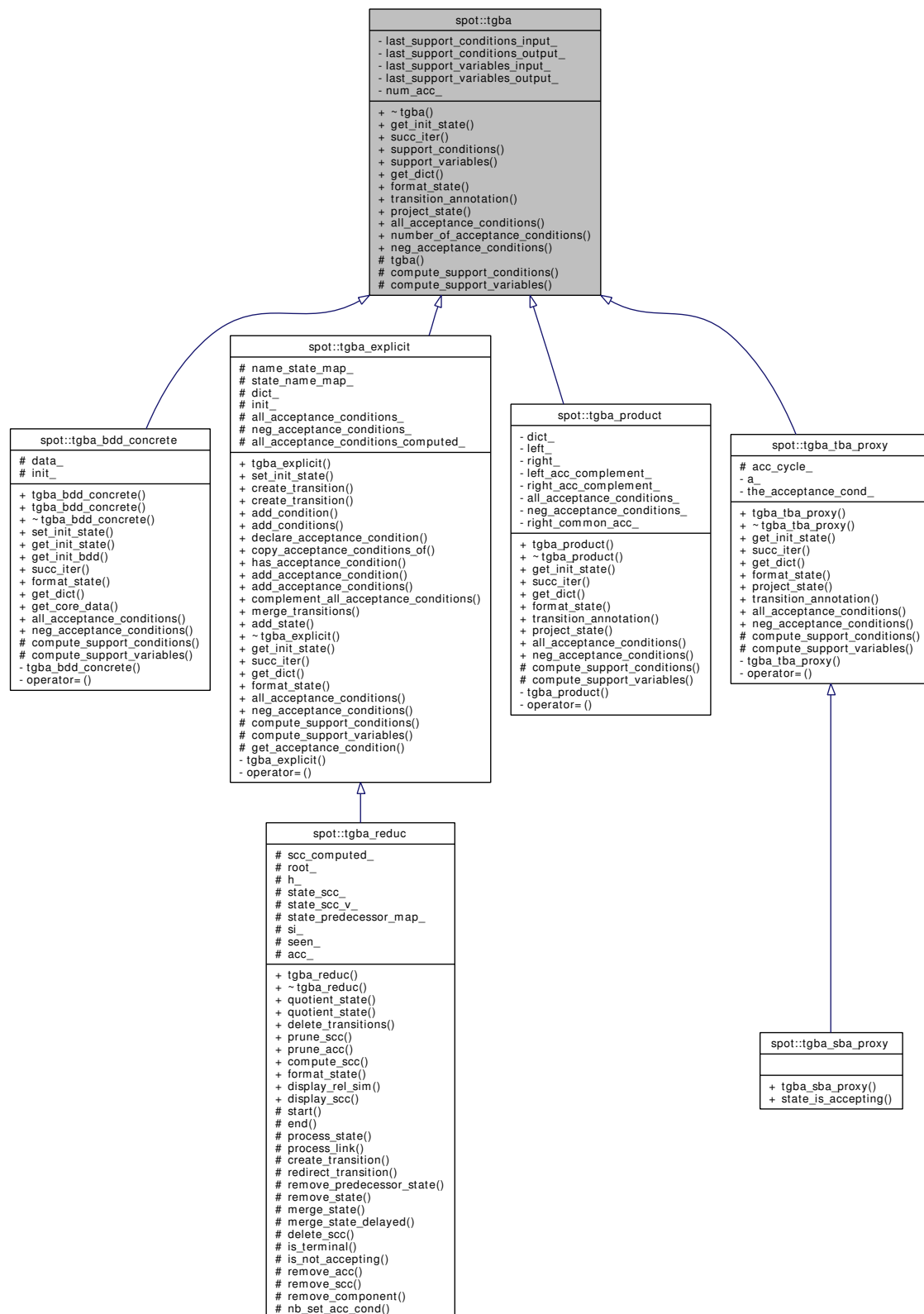
- [evtgba/symbol.hh](#)

**12.99 spot::tgba Class Reference**

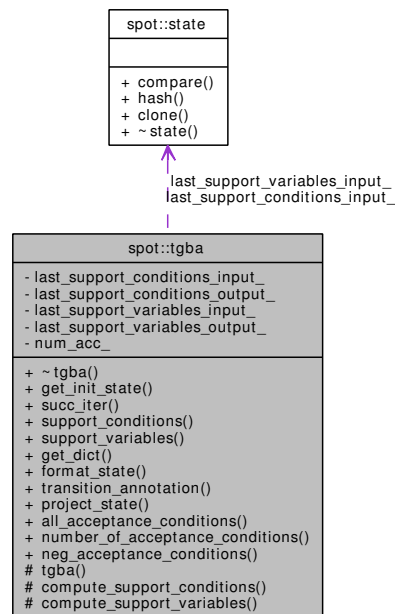
A Transition-based Generalized Büchi Automaton.

```
#include <tgba/tgba.hh>
```

Inheritance diagram for spot::tgba:



Collaboration diagram for spot::tgba:



## Public Member Functions

- virtual `~tgba()`
- virtual `state * get_init_state()` const=0  
*Get the initial [state](#) of the automaton.*
- virtual `tgba_succ_iterator * succ_iter` (const `state *local_state`, const `state *global_state=0`, const `tgba *global_automaton=0`) const=0  
*Get an iterator over the successors of `local_state`.*
- bdd `support_conditions` (const `state *state`) const  
*Get a formula that must hold whatever successor is taken.*
- bdd `support_variables` (const `state *state`) const  
*Get the conjunctions of variables tested by the outgoing transitions of [state](#).*
- virtual bdd\_dict \* `get_dict()` const=0  
*Get the dictionary associated to the automaton.*
- virtual std::string `format_state` (const `state *state`) const=0  
*Format the [state](#) as a string for printing.*
- virtual std::string `transition_annotation` (const `tgba_succ_iterator *t`) const  
*Return a possible annotation for the transition pointed to by the iterator.*
- virtual `state * project_state` (const `state *s`, const `tgba *t`) const  
*Project a [state](#) on an automaton.*

- virtual bdd [all\\_acceptance\\_conditions](#) () const=0  
*Return the set of all acceptance conditions used by this automaton.*
- virtual unsigned int [number\\_of\\_acceptance\\_conditions](#) () const  
*The number of acceptance conditions.*
- virtual bdd [neg\\_acceptance\\_conditions](#) () const=0  
*Return the conjunction of all negated acceptance variables.*

### Protected Member Functions

- [tgba](#) ()
- virtual bdd [compute\\_support\\_conditions](#) (const [state](#) \*state) const=0  
*Do the actual computation of [tgba::support\\_conditions\(\)](#).*
- virtual bdd [compute\\_support\\_variables](#) (const [state](#) \*state) const=0  
*Do the actual computation of [tgba::support\\_variables\(\)](#).*

### Private Attributes

- const [state](#) \* [last\\_support\\_conditions\\_input\\_](#)
- bdd [last\\_support\\_conditions\\_output\\_](#)
- const [state](#) \* [last\\_support\\_variables\\_input\\_](#)
- bdd [last\\_support\\_variables\\_output\\_](#)
- int [num\\_acc\\_](#)

#### 12.99.1 Detailed Description

A Transition-based Generalized Büchi Automaton.

The acronym TGBA (Transition-based Generalized Büchi Automaton) was coined by Dimitra Gianakopoulou and Flavio Lerda in "From States to Transitions: Improving Translation of LTL Formulae to Büchi Automata". (FORTE'02)

TGBAs are transition-based, meanings their labels are put on arcs, not on nodes. They use Generalized Büchi acceptance conditions: there are several acceptance sets (of transitions), and a path can be accepted only if it traverse at least one transition of each set infinitely often.

Browsing such automaton can be achieved using two functions. `get_init_state`, and `succ_iter`. The former returns the initial [state](#) while the latter allows to explore the successor states of any [state](#).

Note that although this is a transition-based automata, we never represent transitions! Transition informations are obtained by querying the iterator over the successors of a [state](#).

#### 12.99.2 Constructor & Destructor Documentation

##### 12.99.2.1 spot::tgba::tgba () [protected]

##### 12.99.2.2 virtual spot::tgba::~~tgba () [virtual]

### 12.99.3 Member Function Documentation

#### 12.99.3.1 virtual state\* spot::tgba::get\_init\_state () const [pure virtual]

Get the initial [state](#) of the automaton.

The [state](#) has been allocated with `new`. It is the responsibility of the caller to `delete` it when no longer needed.

Implemented in [spot::tgba\\_bdd\\_concrete](#), [spot::tgba\\_explicit](#), [spot::tgba\\_product](#), and [spot::tgba\\_tba\\_proxy](#).

#### 12.99.3.2 virtual tgba\_succ\_iterator\* spot::tgba::succ\_iter (const state \* local\_state, const state \* global\_state = 0, const tgba \* global\_automaton = 0) const [pure virtual]

Get an iterator over the successors of *local\_state*.

The iterator has been allocated with `new`. It is the responsibility of the caller to `delete` it when no longer needed.

During synchornized products, additional informations are passed about the entire product and its [state](#). Recall that products can be nested, forming a tree of [spot::tgba](#) where most values are computed on demand. *global\_automaton* designate the root [spot::tgba](#), and *global\_state* its [state](#). This two objects can be used by [succ\\_iter\(\)](#) to restrict the set of successors to compute.

#### Parameters:

*local\_state* The [state](#) whose successors are to be explored. This pointer is not adopted in any way by [succ\\_iter](#), and it is still the caller's responsibility to delete it when appropriate (this can be done during the lifetime of the iterator).

*global\_state* In a product, the [state](#) of the global product automaton. Otherwise, 0. Like *locale\_state*, *global\_state* is not adopted by [succ\\_iter](#).

*global\_automaton* In a product, the global product automaton. Otherwise, 0.

Implemented in [spot::tgba\\_bdd\\_concrete](#), [spot::tgba\\_explicit](#), [spot::tgba\\_product](#), and [spot::tgba\\_tba\\_proxy](#).

#### 12.99.3.3 bdd spot::tgba::support\_conditions (const state \* state) const

Get a formula that must hold whatever successor is taken.

#### Returns:

A formula which must be verified for all successors of [state](#).

This can be as simple as `bddtrue`, or more completely the disjunction of the condition of all successors. This is used as an hint by [succ\\_iter\(\)](#) to reduce the number of successor to compute in a product.

Sub classes should implement [compute\\_support\\_conditions\(\)](#), this function is just a wrapper that will cache the last return value for efficiency.

#### 12.99.3.4 bdd spot::tgba::support\_variables (const state \* state) const

Get the conjunctions of variables tested by the outgoing transitions of [state](#).

All variables tested by outgoing transitions must be returned. This is mandatory.

This is used as an hint by some `succ_iter()` to reduce the number of successor to compute in a product. Sub classes should implement `compute_support_variables()`, this function is just a wrapper that will cache the last return value for efficiency.

### 12.99.3.5 virtual bdd\_dict\* spot::tgba::get\_dict () const [pure virtual]

Get the dictionary associated to the automaton.

State are represented as BDDs. The dictionary allows to map BDD variables back to formulae, and vice versa. This is useful when dealing with several automata (which may use the same BDD variable for different formula), or simply when printing.

Implemented in `spot::tgba_bdd_concrete`, `spot::tgba_explicit`, `spot::tgba_product`, and `spot::tgba_tba_proxy`.

### 12.99.3.6 virtual std::string spot::tgba::format\_state (const state \* state) const [pure virtual]

Format the `state` as a string for printing.

This formatting is the responsibility of the automata who owns the `state`.

Implemented in `spot::tgba_bdd_concrete`, `spot::tgba_explicit`, `spot::tgba_product`, `spot::tgba_reduc`, and `spot::tgba_tba_proxy`.

### 12.99.3.7 virtual std::string spot::tgba::transition\_annotation (const tgba\_succ\_iterator \* t) const [virtual]

Return a possible annotation for the transition pointed to by the iterator.

Implementing this function is optional; the default annotation is the empty string.

#### Parameters:

`t` a non-done `tgba_succ_iterator` for this automata

Reimplemented in `spot::tgba_product`, and `spot::tgba_tba_proxy`.

### 12.99.3.8 virtual state\* spot::tgba::project\_state (const state \* s, const tgba \* t) const [virtual]

Project a `state` on an automaton.

This converts `s`, into that corresponding `spot::state` for `t`. This is useful when you have the `state` of a product, and want restrict this `state` to a specific automata occurring in the product.

It goes without saying that `s` and `t` should be compatible (i.e., `s` is a `state` of `t`).

#### Returns:

0 if the projection fails (`s` is unrelated to `t`), or a new `state*` (the projected `state`) that must be deleted by the caller.

Reimplemented in `spot::tgba_product`, and `spot::tgba_tba_proxy`.

**12.99.3.9 virtual bdd spot::tgba::all\_acceptance\_conditions () const** [pure virtual]

Return the set of all acceptance conditions used by this automaton.

The goal of the emptiness check is to ensure that a strongly connected component walks through each of these accepting conditions. I.e., the union of the accepting conditions of all transition in the SCC should be equal to the result of this function.

Implemented in [spot::tgba\\_bdd\\_concrete](#), [spot::tgba\\_explicit](#), [spot::tgba\\_product](#), and [spot::tgba\\_tba\\_proxy](#).

**12.99.3.10 virtual unsigned int spot::tgba::number\_of\_acceptance\_conditions () const** [virtual]

The number of acceptance conditions.

**12.99.3.11 virtual bdd spot::tgba::neg\_acceptance\_conditions () const** [pure virtual]

Return the conjunction of all negated acceptance variables.

For instance if the automaton uses variables `Acc[a]`, `Acc[b]` and `Acc[c]` to describe acceptance sets, this function should return `!Acc[a] & !Acc[b] & !Acc[c]`.

This is useful when making products: each operand's condition set should be augmented with the [neg\\_acceptance\\_conditions\(\)](#) of the other operand.

Implemented in [spot::tgba\\_bdd\\_concrete](#), [spot::tgba\\_explicit](#), [spot::tgba\\_product](#), and [spot::tgba\\_tba\\_proxy](#).

**12.99.3.12 virtual bdd spot::tgba::compute\_support\_conditions (const state \* state) const** [protected, pure virtual]

Do the actual computation of [tgba::support\\_conditions\(\)](#).

Implemented in [spot::tgba\\_bdd\\_concrete](#), [spot::tgba\\_explicit](#), [spot::tgba\\_product](#), and [spot::tgba\\_tba\\_proxy](#).

**12.99.3.13 virtual bdd spot::tgba::compute\_support\_variables (const state \* state) const** [protected, pure virtual]

Do the actual computation of [tgba::support\\_variables\(\)](#).

Implemented in [spot::tgba\\_bdd\\_concrete](#), [spot::tgba\\_explicit](#), [spot::tgba\\_product](#), and [spot::tgba\\_tba\\_proxy](#).

**12.99.4 Member Data Documentation****12.99.4.1 const state\* spot::tgba::last\_support\_conditions\_input\_** [mutable, private]**12.99.4.2 bdd spot::tgba::last\_support\_conditions\_output\_** [mutable, private]**12.99.4.3 const state\* spot::tgba::last\_support\_variables\_input\_** [mutable, private]**12.99.4.4 bdd spot::tgba::last\_support\_variables\_output\_** [mutable, private]



#### 12.99.4.5 int spot::tgba::num\_acc\_ [mutable, private]

The documentation for this class was generated from the following file:

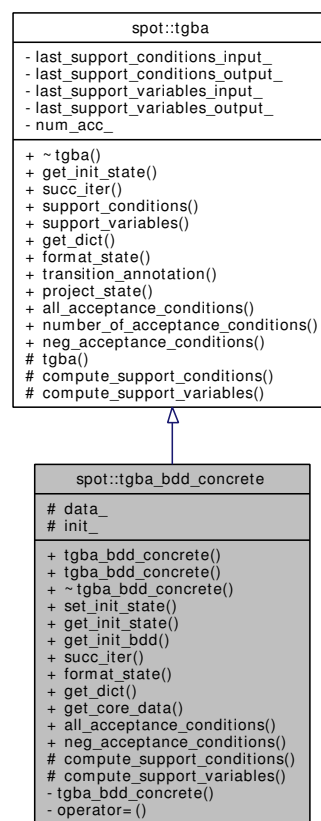
- [tgba/tgba.hh](#)

### 12.100 spot::tgba\_bdd\_concrete Class Reference

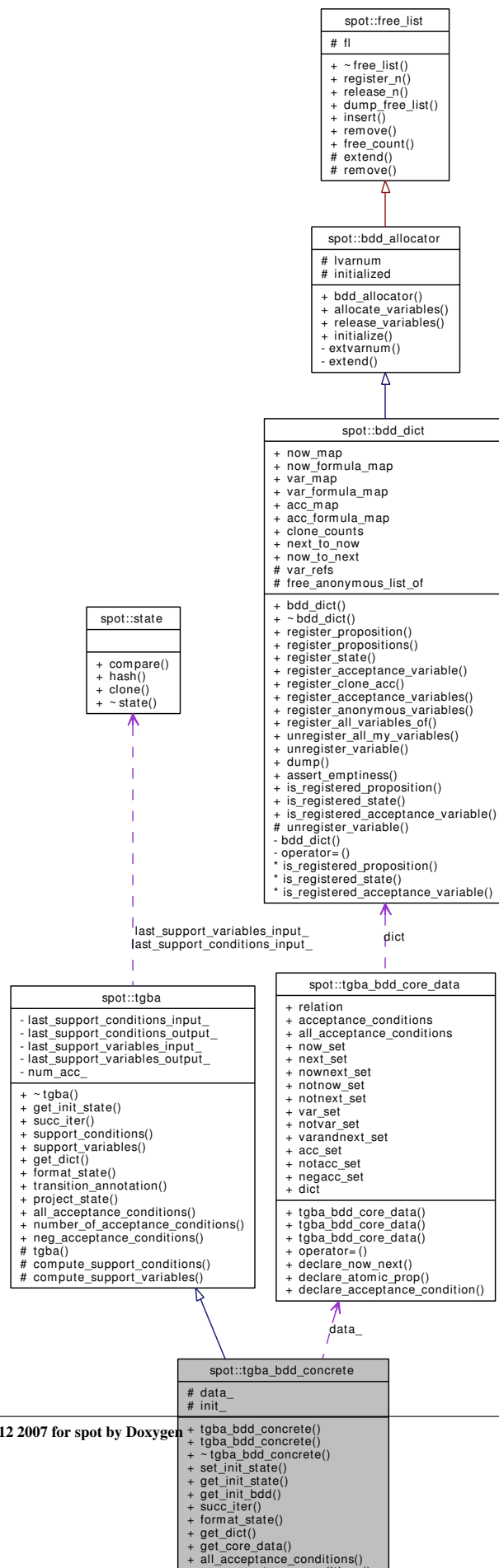
A concrete [spot::tgba](#) implemented using BDDs.

```
#include <tgba/tgbabddconcrete.hh>
```

Inheritance diagram for spot::tgba\_bdd\_concrete:



Collaboration diagram for spot::tgba\_bdd\_concrete:



## Public Member Functions

- [tgba\\_bdd\\_concrete](#) (const [tgba\\_bdd\\_factory](#) &fact)  
*Construct a [tgba\\_bdd\\_concrete](#) with unknown initial [state](#).*
- [tgba\\_bdd\\_concrete](#) (const [tgba\\_bdd\\_factory](#) &fact, bdd init)  
*Construct a [tgba\\_bdd\\_concrete](#) with known initial [state](#).*
- virtual [~tgba\\_bdd\\_concrete](#) ()
- virtual void [set\\_init\\_state](#) (bdd s)  
*Set the initial [state](#).*
- virtual [state\\_bdd](#) \* [get\\_init\\_state](#) () const  
*Get the initial [state](#) of the automaton.*
- bdd [get\\_init\\_bdd](#) () const  
*Get the initial [state](#) directly as a BDD.*
- virtual [tgba\\_succ\\_iterator\\_concrete](#) \* [succ\\_iter](#) (const [state](#) \*local\_state, const [state](#) \*global\_state=0, const [tgba](#) \*global\_automaton=0) const  
*Get an iterator over the successors of [local\\_state](#).*
- virtual std::string [format\\_state](#) (const [state](#) \*state) const  
*Format the [state](#) as a string for printing.*
- virtual [bdd\\_dict](#) \* [get\\_dict](#) () const  
*Get the dictionary associated to the automaton.*
- const [tgba\\_bdd\\_core\\_data](#) & [get\\_core\\_data](#) () const  
*Get the core data associated to this automaton.*
- virtual bdd [all\\_acceptance\\_conditions](#) () const  
*Return the set of all acceptance conditions used by this automaton.*
- virtual bdd [neg\\_acceptance\\_conditions](#) () const  
*Return the conjunction of all negated acceptance variables.*
- bdd [support\\_conditions](#) (const [state](#) \*state) const  
*Get a formula that must hold whatever successor is taken.*
- bdd [support\\_variables](#) (const [state](#) \*state) const  
*Get the conjunctions of variables tested by the outgoing transitions of [state](#).*
- virtual std::string [transition\\_annotation](#) (const [tgba\\_succ\\_iterator](#) \*t) const  
*Return a possible annotation for the transition pointed to by the iterator.*
- virtual [state](#) \* [project\\_state](#) (const [state](#) \*s, const [tgba](#) \*t) const  
*Project a [state](#) on an automaton.*
- virtual unsigned int [number\\_of\\_acceptance\\_conditions](#) () const  
*The number of acceptance conditions.*

### Protected Member Functions

- virtual bdd [compute\\_support\\_conditions](#) (const [state](#) \*state) const  
*Do the actual computation of [tgba::support\\_conditions\(\)](#).*
- virtual bdd [compute\\_support\\_variables](#) (const [state](#) \*state) const  
*Do the actual computation of [tgba::support\\_variables\(\)](#).*

### Protected Attributes

- [tgba\\_bdd\\_core\\_data](#) data\_  
*Core data associated to the automaton.*
- bdd [init](#)\_  
*Initial [state](#).*

### Private Member Functions

- [tgba\\_bdd\\_concrete](#) (const [tgba\\_bdd\\_concrete](#) &)
- [tgba\\_bdd\\_concrete](#) & operator= (const [tgba\\_bdd\\_concrete](#) &)

#### 12.100.1 Detailed Description

A concrete [spot::tgba](#) implemented using BDDs.

#### 12.100.2 Constructor & Destructor Documentation

##### 12.100.2.1 spot::tgba\_bdd\_concrete::tgba\_bdd\_concrete (const tgba\_bdd\_factory & fact)

Construct a [tgba\\_bdd\\_concrete](#) with unknown initial [state](#).

[set\\_init\\_state\(\)](#) should be called later.

##### 12.100.2.2 spot::tgba\_bdd\_concrete::tgba\_bdd\_concrete (const tgba\_bdd\_factory & fact, bdd init)

Construct a [tgba\\_bdd\\_concrete](#) with known initial [state](#).

##### 12.100.2.3 virtual spot::tgba\_bdd\_concrete::~~tgba\_bdd\_concrete () [virtual]

##### 12.100.2.4 spot::tgba\_bdd\_concrete::tgba\_bdd\_concrete (const tgba\_bdd\_concrete &) [private]

#### 12.100.3 Member Function Documentation

##### 12.100.3.1 virtual void spot::tgba\_bdd\_concrete::set\_init\_state (bdd s) [virtual]

Set the initial [state](#).

**12.100.3.2 virtual state\_bdd\* spot::tgba\_bdd\_concrete::get\_init\_state () const** [virtual]

Get the initial [state](#) of the automaton.

The [state](#) has been allocated with `new`. It is the responsibility of the caller to `delete` it when no longer needed.

Implements [spot::tgba](#).

**12.100.3.3 bdd spot::tgba\_bdd\_concrete::get\_init\_bdd () const**

Get the initial [state](#) directly as a BDD.

The sole point of this method is to prevent writing horrors such as

```
state_bdd* s = automata.get_init_state();
some_class some_instance(s->as_bdd());
delete s;
```

**12.100.3.4 virtual tgba\_succ\_iterator\_concrete\* spot::tgba\_bdd\_concrete::succ\_iter (const state \* local\_state, const state \* global\_state = 0, const tgba \* global\_automaton = 0) const** [virtual]

Get an iterator over the successors of *local\_state*.

The iterator has been allocated with `new`. It is the responsibility of the caller to `delete` it when no longer needed.

During synchornized products, additional informations are passed about the entire product and its [state](#). Recall that products can be nested, forming a tree of [spot::tgba](#) where most values are computed on demand. *global\_automaton* designate the root [spot::tgba](#), and *global\_state* its [state](#). This two objects can be used by [succ\\_iter\(\)](#) to restrict the set of successors to compute.

**Parameters:**

*local\_state* The [state](#) whose successors are to be explored. This pointer is not adopted in any way by [succ\\_iter](#), and it is still the caller's responsibility to delete it when appropriate (this can be done during the lifetime of the iterator).

*global\_state* In a product, the [state](#) of the global product automaton. Otherwise, 0. Like *locale\_state*, *global\_state* is not adopted by [succ\\_iter](#).

*global\_automaton* In a product, the global product automaton. Otherwise, 0.

Implements [spot::tgba](#).

**12.100.3.5 virtual std::string spot::tgba\_bdd\_concrete::format\_state (const state \* state) const** [virtual]

Format the [state](#) as a string for printing.

This formatting is the responsibility of the automata who owns the [state](#).

Implements [spot::tgba](#).

**12.100.3.6 virtual bdd\_dict\* spot::tgba\_bdd\_concrete::get\_dict () const** [virtual]

Get the dictionary associated to the automaton.

State are represented as BDDs. The dictionary allows to map BDD variables back to formulae, and vice versa. This is useful when dealing with several automata (which may use the same BDD variable for different formula), or simply when printing.

Implements [spot::tgba](#).

#### 12.100.3.7 const tgba\_bdd\_core\_data& spot::tgba\_bdd\_concrete::get\_core\_data () const

Get the core data associated to this automaton.

These data includes the various BDD used to represent the relation, encode variable sets, Next-to-Now rewrite rules, etc.

#### 12.100.3.8 virtual bdd spot::tgba\_bdd\_concrete::all\_acceptance\_conditions () const [virtual]

Return the set of all acceptance conditions used by this automaton.

The goal of the emptiness check is to ensure that a strongly connected component walks through each of these accepting conditions. I.e., the union of the accepting conditions of all transition in the SCC should be equal to the result of this function.

Implements [spot::tgba](#).

#### 12.100.3.9 virtual bdd spot::tgba\_bdd\_concrete::neg\_acceptance\_conditions () const [virtual]

Return the conjunction of all negated acceptance variables.

For instance if the automaton uses variables  $Acc[a]$ ,  $Acc[b]$  and  $Acc[c]$  to describe acceptance sets, this function should return  $!Acc[a] \& !Acc[b] \& !Acc[c]$ .

This is useful when making products: each operand's condition set should be augmented with the [neg\\_acceptance\\_conditions\(\)](#) of the other operand.

Implements [spot::tgba](#).

#### 12.100.3.10 virtual bdd spot::tgba\_bdd\_concrete::compute\_support\_conditions (const state \* state) const [protected, virtual]

Do the actual computation of [tgba::support\\_conditions\(\)](#).

Implements [spot::tgba](#).

#### 12.100.3.11 virtual bdd spot::tgba\_bdd\_concrete::compute\_support\_variables (const state \* state) const [protected, virtual]

Do the actual computation of [tgba::support\\_variables\(\)](#).

Implements [spot::tgba](#).

#### 12.100.3.12 tgba\_bdd\_concrete& spot::tgba\_bdd\_concrete::operator= (const tgba\_bdd\_concrete &) [private]

#### 12.100.3.13 bdd spot::tgba::support\_conditions (const state \* state) const [inherited]

Get a formula that must hold whatever successor is taken.

**Returns:**

A formula which must be verified for all successors of *state*.

This can be as simple as `bddtrue`, or more completely the disjunction of the condition of all successors. This is used as an hint by `succ_iter()` to reduce the number of successor to compute in a product.

Sub classes should implement `compute_support_conditions()`, this function is just a wrapper that will cache the last return value for efficiency.

**12.100.3.14 bdd spot::tgba::support\_variables (const state \* state) const [inherited]**

Get the conjunctions of variables tested by the outgoing transitions of *state*.

All variables tested by outgoing transitions must be returned. This is mandatory.

This is used as an hint by some `succ_iter()` to reduce the number of successor to compute in a product.

Sub classes should implement `compute_support_variables()`, this function is just a wrapper that will cache the last return value for efficiency.

**12.100.3.15 virtual std::string spot::tgba::transition\_annotation (const tgba\_succ\_iterator \* t) const [virtual, inherited]**

Return a possible annotation for the transition pointed to by the iterator.

Implementing this function is optional; the default annotation is the empty string.

**Parameters:**

*t* a non-done `tgba_succ_iterator` for this automata

Reimplemented in `spot::tgba_product`, and `spot::tgba_tba_proxy`.

**12.100.3.16 virtual state\* spot::tgba::project\_state (const state \* s, const tgba \* t) const [virtual, inherited]**

Project a *state* on an automaton.

This converts *s*, into that corresponding `spot::state` for *t*. This is useful when you have the *state* of a product, and want restrict this *state* to a specific automata occurring in the product.

It goes without saying that *s* and *t* should be compatible (i.e., *s* is a *state* of *t*).

**Returns:**

0 if the projection fails (*s* is unrelated to *t*), or a new `state*` (the projected *state*) that must be deleted by the caller.

Reimplemented in `spot::tgba_product`, and `spot::tgba_tba_proxy`.

**12.100.3.17 virtual unsigned int spot::tgba::number\_of\_acceptance\_conditions () const [virtual, inherited]**

The number of acceptance conditions.

### 12.100.4 Member Data Documentation

#### 12.100.4.1 tgba\_bdd\_core\_data spot::tgba\_bdd\_concrete::data\_ [protected]

Core data associated to the automaton.

#### 12.100.4.2 bdd spot::tgba\_bdd\_concrete::init\_ [protected]

Initial [state](#).

The documentation for this class was generated from the following file:

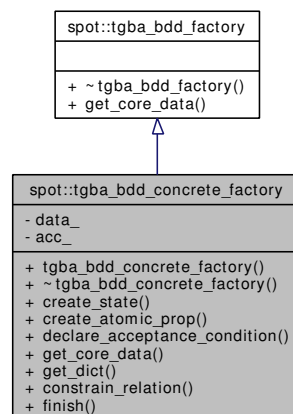
- [tgba/tgababddconcrete.hh](#)

## 12.101 spot::tgba\_bdd\_concrete\_factory Class Reference

Helper class to build a [spot::tgba\\_bdd\\_concrete](#) object.

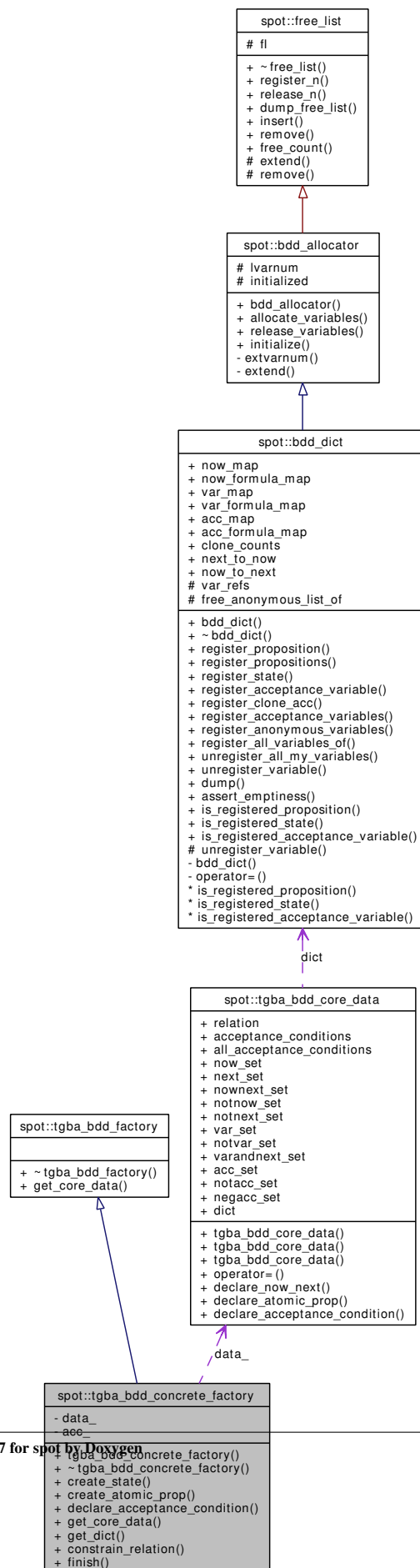
```
#include <tgba/tgababddconcretefactory.hh>
```

Inheritance diagram for spot::tgba\_bdd\_concrete\_factory:





Collaboration diagram for spot::tgba\_bdd\_concrete\_factory:



## Public Member Functions

- [tgba\\_bdd\\_concrete\\_factory](#) (bdd\_dict \*dict)
- virtual [~tgba\\_bdd\\_concrete\\_factory](#) ()
- int [create\\_state](#) (const ltl::formula \*f)
- int [create\\_atomic\\_prop](#) (const ltl::formula \*f)
- void [declare\\_acceptance\\_condition](#) (bdd b, const ltl::formula \*a)
- const [tgba\\_bdd\\_core\\_data](#) & [get\\_core\\_data](#) () const

*Get the core data for the new automata.*

- bdd\_dict \* [get\\_dict](#) () const
- void [constrain\\_relation](#) (bdd new\_rel)

*Add a new constraint to the relation.*

- void [finish](#) ()

*Perform final computations before the relation can be used.*

## Private Types

- typedef Sgi::hash\_map< const ltl::formula \*, bdd, ltl::formula\_ptr\_hash > [acc\\_map\\_](#)

## Private Attributes

- [tgba\\_bdd\\_core\\_data](#) [data\\_](#)  
*Core data for the new automata.*
- [acc\\_map\\_](#) [acc\\_](#)  
*BDD associated to each acceptance condition.*

### 12.101.1 Detailed Description

Helper class to build a [spot::tgba\\_bdd\\_concrete](#) object.

### 12.101.2 Member Typedef Documentation

**12.101.2.1** typedef Sgi::hash\_map<const ltl::formula\*, bdd, ltl::formula\_ptr\_hash> spot::tgba\_bdd\_concrete\_factory::acc\_map\_ [private]

### 12.101.3 Constructor & Destructor Documentation

**12.101.3.1** spot::tgba\_bdd\_concrete\_factory::tgba\_bdd\_concrete\_factory (bdd\_dict \* dict)

**12.101.3.2** virtual spot::tgba\_bdd\_concrete\_factory::~tgba\_bdd\_concrete\_factory ()  
[virtual]

### 12.101.4 Member Function Documentation

#### 12.101.4.1 int spot::tgba\_bdd\_concrete\_factory::create\_state (const ltl::formula \* *f*)

Create a [state](#) variable for formula *f*.

##### Parameters:

*f* The formula to create a [state](#) for.

##### Returns:

The variable number for this [state](#).

The [state](#) is not created if it already exists. Instead its existing variable number is returned. Variable numbers can be turned into BDD using `ithvar()`.

#### 12.101.4.2 int spot::tgba\_bdd\_concrete\_factory::create\_atomic\_prop (const ltl::formula \* *f*)

Create an atomic proposition variable for formula *f*.

##### Parameters:

*f* The formula to create an atomic proposition for.

##### Returns:

The variable number for this [state](#).

The atomic proposition is not created if it already exists. Instead its existing variable number is returned. Variable numbers can be turned into BDD using `ithvar()`.

#### 12.101.4.3 void spot::tgba\_bdd\_concrete\_factory::declare\_acceptance\_condition (bdd *b*, const ltl::formula \* *a*)

Declare an acceptance condition.

Formula such as '`f U g`' or '`F g`' make the promise that '`g`' will be fulfilled eventually. So once one of this formula has been translated into a BDD, we use [declare\\_acceptance\\_condition\(\)](#) to associate all other states to the acceptance set of '`g`'.

##### Parameters:

*b* a BDD indicating which variables are in the acceptance set

*a* the formula associated

#### 12.101.4.4 const tgba\_bdd\_core\_data& spot::tgba\_bdd\_concrete\_factory::get\_core\_data () const [virtual]

Get the core data for the new automata.

Implements [spot::tgba\\_bdd\\_factory](#).

#### 12.101.4.5 bdd\_dict\* spot::tgba\_bdd\_concrete\_factory::get\_dict () const

**12.101.4.6 void spot::tgba\_bdd\_concrete\_factory::constrain\_relation (bdd *new\_rel*)**

Add a new constraint to the relation.

**12.101.4.7 void spot::tgba\_bdd\_concrete\_factory::finish ()**

Perform final computations before the relation can be used.

This function should be called after all propositions, [state](#), acceptance conditions, and constraints have been declared, and before calling [get\\_code\\_data\(\)](#) or [get\\_dict\(\)](#).

**12.101.5 Member Data Documentation****12.101.5.1 tgba\_bdd\_core\_data spot::tgba\_bdd\_concrete\_factory::data\_ [private]**

Core data for the new automata.

**12.101.5.2 acc\_map\_ spot::tgba\_bdd\_concrete\_factory::acc\_ [private]**

BDD associated to each acceptance condition.

The documentation for this class was generated from the following file:

- [tgba/tgabddconcretefactory.hh](#)

**12.102 spot::tgba\_bdd\_core\_data Struct Reference**

Core data for a TGBA encoded using BDDs.

```
#include <tgba/tgabddcoredata.hh>
```

Collaboration diagram for spot::tgba\_bdd\_core\_data:



## Public Member Functions

- [tgba\\_bdd\\_core\\_data](#) ([bdd\\_dict](#) \*dict)  
*Default constructor.*
- [tgba\\_bdd\\_core\\_data](#) (const [tgba\\_bdd\\_core\\_data](#) &copy)  
*Copy constructor.*
- [tgba\\_bdd\\_core\\_data](#) (const [tgba\\_bdd\\_core\\_data](#) &left, const [tgba\\_bdd\\_core\\_data](#) &right)  
*Merge two [tgba\\_bdd\\_core\\_data](#).*
- const [tgba\\_bdd\\_core\\_data](#) & operator= (const [tgba\\_bdd\\_core\\_data](#) &copy)
- void [declare\\_now\\_next](#) (bdd now, bdd next)  
*Update the variable sets to take a new pair of variables into account.*
- void [declare\\_atomic\\_prop](#) (bdd var)  
*Update the variable sets to take a new atomic proposition into account.*
- void [declare\\_acceptance\\_condition](#) (bdd prom)  
*Update the variable sets to take a new acceptance condition into account.*

## Public Attributes

- bdd [relation](#)  
*encodes the transition relation of the TGBA.*
- bdd [acceptance\\_conditions](#)  
*encodes the acceptance conditions*
- bdd [all\\_acceptance\\_conditions](#)  
*The set of all acceptance conditions used by the Automaton.*
- bdd [now\\_set](#)  
*The conjunction of all Now variables, in their positive form.*
- bdd [next\\_set](#)  
*The conjunction of all Next variables, in their positive form.*
- bdd [nownext\\_set](#)  
*The conjunction of all Now and Next variables, in their positive form.*
- bdd [notnow\\_set](#)  
*The (positive) conjunction of all variables which are not Now variables.*
- bdd [notnext\\_set](#)  
*The (positive) conjunction of all variables which are not Next variables.*
- bdd [var\\_set](#)  
*The (positive) conjunction of all variables which are atomic propositions.*

- [bdd\\_notvar\\_set](#)  
*The (positive) conjunction of all variables which are not atomic propositions.*
- [bdd\\_varandnext\\_set](#)  
*The (positive) conjunction of all Next variables and atomic propositions.*
- [bdd\\_acc\\_set](#)  
*The (positive) conjunction of all variables which are acceptance conditions.*
- [bdd\\_notacc\\_set](#)  
*The (positive) conjunction of all variables which are not acceptance conditions.*
- [bdd\\_negacc\\_set](#)  
*The negative conjunction of all variables which are acceptance conditions.*
- [bdd\\_dict](#) \* [dict](#)  
*The dictionary used by the automata.*

### 12.102.1 Detailed Description

Core data for a TGBA encoded using BDDs.

### 12.102.2 Constructor & Destructor Documentation

#### 12.102.2.1 spot::tgba\_bdd\_core\_data::tgba\_bdd\_core\_data (bdd\_dict \* dict)

Default constructor.

Initially all variable set are empty and the `relation` is true.

#### 12.102.2.2 spot::tgba\_bdd\_core\_data::tgba\_bdd\_core\_data (const tgba\_bdd\_core\_data & copy)

Copy constructor.

#### 12.102.2.3 spot::tgba\_bdd\_core\_data::tgba\_bdd\_core\_data (const tgba\_bdd\_core\_data & left, const tgba\_bdd\_core\_data & right)

Merge two [tgba\\_bdd\\_core\\_data](#).

This is used when building a product of two automata.

### 12.102.3 Member Function Documentation

#### 12.102.3.1 const tgba\_bdd\_core\_data& spot::tgba\_bdd\_core\_data::operator= (const tgba\_bdd\_core\_data & copy)

#### 12.102.3.2 void spot::tgba\_bdd\_core\_data::declare\_now\_next (bdd now, bdd next)

Update the variable sets to take a new pair of variables into account.

**12.102.3.3 void spot::tgba\_bdd\_core\_data::declare\_atomic\_prop (bdd var)**

Update the variable sets to take a new atomic proposition into account.

**12.102.3.4 void spot::tgba\_bdd\_core\_data::declare\_acceptance\_condition (bdd prom)**

Update the variable sets to take a new acceptance condition into account.

**12.102.4 Member Data Documentation****12.102.4.1 bdd spot::tgba\_bdd\_core\_data::relation**

encodes the transition relation of the TGBA.

relation uses three kinds of variables:

- "Now" variables, that encode the current [state](#)
- "Next" variables, that encode the destination [state](#)
- atomic propositions, which are things to verify before going on to the next [state](#)

**12.102.4.2 bdd spot::tgba\_bdd\_core\_data::acceptance\_conditions**

encodes the acceptance conditions

$a \cup b$ , or  $F b$ , both imply that  $b$  should be verified eventually. We encode this with generalized Büchi accepting conditions. An acceptance set, called  $Acc[b]$ , hold all the [state](#) that do not promise to verify  $b$  eventually. (I.e., all the states that contain  $b$ , or do not contain  $a \cup b$ , or  $F b$ .)

The `spot::succ_iter::current_acceptance_conditions()` method will return the  $Acc[x]$  variables of the acceptance sets in which a transition is. Actually we never return  $Acc[x]$  alone, but  $Acc[x]$  and all other acceptance variables negated.

So if there is three acceptance set  $a$ ,  $b$ , and  $c$ , and a transition is in set  $a$ , we'll return  $Acc[a] \& !Acc[b] \& !Acc[c]$ . If the transition is in both  $a$  and  $b$ , we'll return  $(Acc[a] \& !Acc[b] \& !Acc[c]) \mid (!Acc[a] \& Acc[b] \& !Acc[c])$ .

Acceptance conditions are attributed to transitions and are only concerned by atomic propositions (which label the transitions) and Next variables (the destination). Typically, a transition should bear the variable  $Acc[b]$  if it doesn't check for 'b' and have a destination of the form  $a \cup b$ , or  $F b$ .

To summarize, `acceptance_conditions` contains three kinds of variables:

- "Next" variables, that encode the destination [state](#),
- atomic propositions, which are things to verify before going on to the next [state](#),
- "Acc" variables.

**12.102.4.3 bdd spot::tgba\_bdd\_core\_data::all\_acceptance\_conditions**

The set of all acceptance conditions used by the Automaton.

The goal of the emptiness check is to ensure that a strongly connected component walks through each of these accepting conditions. I.e., the union of the accepting conditions of all transition in the SCC should be equal to the result of this function.



**12.102.4.4 bdd spot::tgba\_bdd\_core\_data::now\_set**

The conjunction of all Now variables, in their positive form.

**12.102.4.5 bdd spot::tgba\_bdd\_core\_data::next\_set**

The conjunction of all Next variables, in their positive form.

**12.102.4.6 bdd spot::tgba\_bdd\_core\_data::nownext\_set**

The conjunction of all Now and Next variables, in their positive form.

**12.102.4.7 bdd spot::tgba\_bdd\_core\_data::notnow\_set**

The (positive) conjunction of all variables which are not Now variables.

**12.102.4.8 bdd spot::tgba\_bdd\_core\_data::notnext\_set**

The (positive) conjunction of all variables which are not Next variables.

**12.102.4.9 bdd spot::tgba\_bdd\_core\_data::var\_set**

The (positive) conjunction of all variables which are atomic propositions.

**12.102.4.10 bdd spot::tgba\_bdd\_core\_data::notvar\_set**

The (positive) conjunction of all variables which are not atomic propositions.

**12.102.4.11 bdd spot::tgba\_bdd\_core\_data::varandnext\_set**

The (positive) conjunction of all Next variables and atomic propositions.

**12.102.4.12 bdd spot::tgba\_bdd\_core\_data::acc\_set**

The (positive) conjunction of all variables which are acceptance conditions.

**12.102.4.13 bdd spot::tgba\_bdd\_core\_data::notacc\_set**

The (positive) conjunction of all variables which are not acceptance conditions.

**12.102.4.14 bdd spot::tgba\_bdd\_core\_data::negacc\_set**

The negative conjunction of all variables which are acceptance conditions.

**12.102.4.15 bdd\_dict\* spot::tgba\_bdd\_core\_data::dict**

The dictionary used by the automata.

The documentation for this struct was generated from the following file:

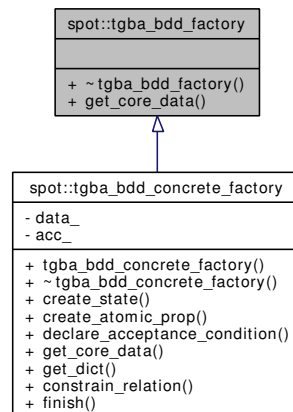
- [tgba/tgabddcoredata.hh](#)

## 12.103 spot::tgba\_bdd\_factory Class Reference

Abstract class for [spot::tgba\\_bdd\\_concrete](#) factories.

```
#include <tgba/tgbabddfactory.hh>
```

Inheritance diagram for spot::tgba\_bdd\_factory:



### Public Member Functions

- virtual `~tgba_bdd_factory()`
- virtual const `tgba_bdd_core_data & get_core_data()` const=0

*Get the core data for the new automata.*

### 12.103.1 Detailed Description

Abstract class for [spot::tgba\\_bdd\\_concrete](#) factories.

A [spot::tgba\\_bdd\\_concrete](#) can be constructed from anything that supplies core data and their associated dictionary.

### 12.103.2 Constructor & Destructor Documentation

**12.103.2.1** virtual `spot::tgba_bdd_factory::~~tgba_bdd_factory()` [inline, virtual]

### 12.103.3 Member Function Documentation

**12.103.3.1** virtual const `tgba_bdd_core_data& spot::tgba_bdd_factory::get_core_data()` const [pure virtual]

Get the core data for the new automata.

Implemented in [spot::tgba\\_bdd\\_concrete\\_factory](#).

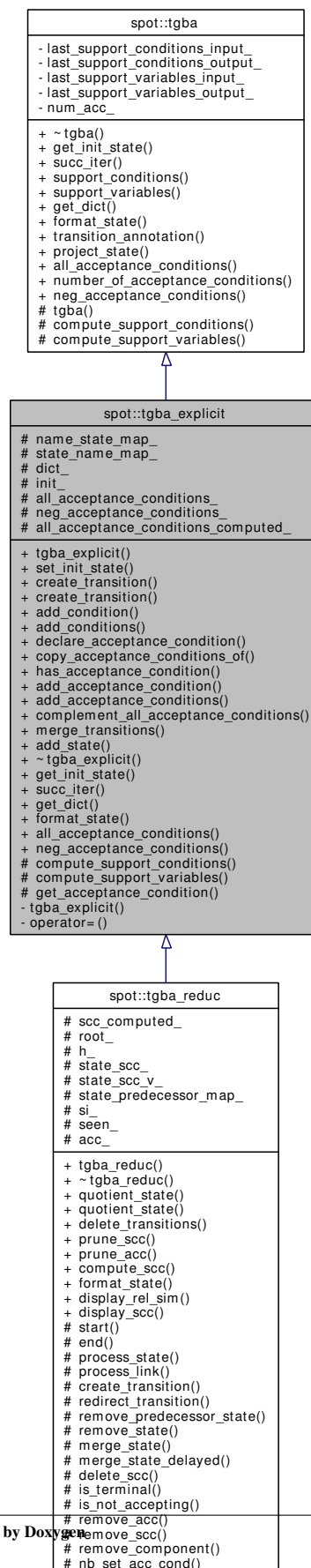
The documentation for this class was generated from the following file:

- [tgba/tgbabddfactory.hh](#)

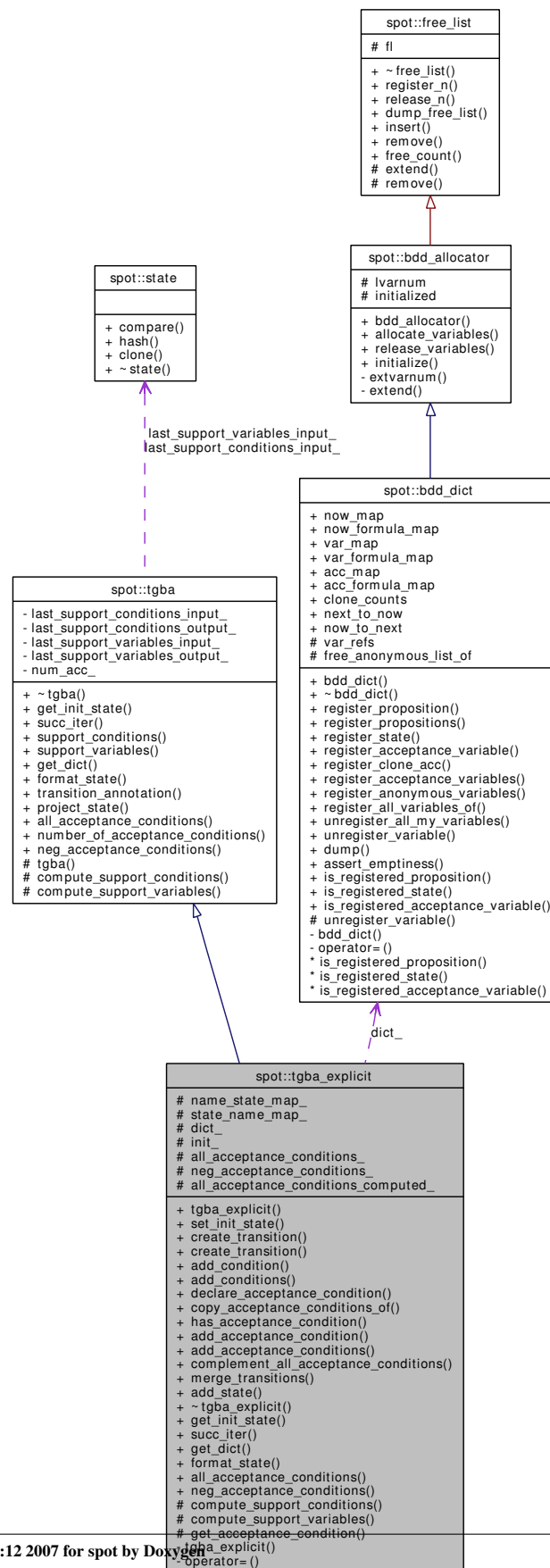
## 12.104 spot::tgba\_explicit Class Reference

```
#include <tgba/tgbaexplicit.hh>
```

Inheritance diagram for spot::tgba\_explicit:



Collaboration diagram for spot::tgba\_explicit:



## Public Types

- typedef std::list< [transition](#) \* > [state](#)

## Public Member Functions

- [tgba\\_explicit](#) (bdd\_dict \*dict)
- [state](#) \* [set\\_init\\_state](#) (const std::string &state)
- [transition](#) \* [create\\_transition](#) (const std::string &source, const std::string &dest)
- [transition](#) \* [create\\_transition](#) ([state](#) \*source, const [state](#) \*dest)
- void [add\\_condition](#) ([transition](#) \*t, const [ltl::formula](#) \*f)
- void [add\\_conditions](#) ([transition](#) \*t, bdd f)  
*This assumes that all variables in f are known from dict.*
- void [declare\\_acceptance\\_condition](#) (const [ltl::formula](#) \*f)
- void [copy\\_acceptance\\_conditions\\_of](#) (const [tgba](#) \*a)  
*Copy the acceptance conditions of a tgba.*
- bool [has\\_acceptance\\_condition](#) (const [ltl::formula](#) \*f) const
- void [add\\_acceptance\\_condition](#) ([transition](#) \*t, const [ltl::formula](#) \*f)
- void [add\\_acceptance\\_conditions](#) ([transition](#) \*t, bdd f)  
*This assumes that all acceptance conditions in f are known from dict.*
- void [complement\\_all\\_acceptance\\_conditions](#) ()
- void [merge\\_transitions](#) ()
- [state](#) \* [add\\_state](#) (const std::string &name)
- virtual ~[tgba\\_explicit](#) ()
- virtual [spot::state](#) \* [get\\_init\\_state](#) () const  
*Get the initial state of the automaton.*
- virtual [tgba\\_succ\\_iterator](#) \* [succ\\_iter](#) (const [spot::state](#) \*local\_state, const [spot::state](#) \*global\_state=0, const [tgba](#) \*global\_automaton=0) const  
*Get an iterator over the successors of local\_state.*
- virtual bdd\_dict \* [get\\_dict](#) () const  
*Get the dictionary associated to the automaton.*
- virtual std::string [format\\_state](#) (const [spot::state](#) \*state) const  
*Format the state as a string for printing.*
- virtual bdd [all\\_acceptance\\_conditions](#) () const  
*Return the set of all acceptance conditions used by this automaton.*
- virtual bdd [neg\\_acceptance\\_conditions](#) () const  
*Return the conjunction of all negated acceptance variables.*
- bdd [support\\_conditions](#) (const [state](#) \*state) const  
*Get a formula that must hold whatever successor is taken.*
- bdd [support\\_variables](#) (const [state](#) \*state) const

*Get the conjunctions of variables tested by the outgoing transitions of [state](#).*

- virtual std::string [transition\\_annotation](#) (const [tgba\\_succ\\_iterator](#) \*t) const  
*Return a possible annotation for the transition pointed to by the iterator.*
- virtual [state](#) \* [project\\_state](#) (const [state](#) \*s, const [tgba](#) \*t) const  
*Project a [state](#) on an automaton.*
- virtual unsigned int [number\\_of\\_acceptance\\_conditions](#) () const  
*The number of acceptance conditions.*

## Protected Types

- typedef Sgi::hash\_map< const std::string, [tgba\\_explicit::state](#) \*, string\_hash > [ns\\_map](#)
- typedef Sgi::hash\_map< const [tgba\\_explicit::state](#) \*, std::string, ptr\_hash< [tgba\\_explicit::state](#) > > [sn\\_map](#)

## Protected Member Functions

- virtual bdd [compute\\_support\\_conditions](#) (const [spot::state](#) \*state) const  
*Do the actual computation of [tgba::support\\_conditions\(\)](#).*
- virtual bdd [compute\\_support\\_variables](#) (const [spot::state](#) \*state) const  
*Do the actual computation of [tgba::support\\_variables\(\)](#).*
- bdd [get\\_acceptance\\_condition](#) (const ltl::formula \*f)

## Protected Attributes

- [ns\\_map](#) [name\\_state\\_map\\_](#)
- [sn\\_map](#) [state\\_name\\_map\\_](#)
- bdd\_dict \* [dict\\_](#)
- [tgba\\_explicit::state](#) \* [init\\_](#)
- bdd [all\\_acceptance\\_conditions\\_](#)
- bdd [neg\\_acceptance\\_conditions\\_](#)
- bool [all\\_acceptance\\_conditions\\_computed\\_](#)

## Private Member Functions

- [tgba\\_explicit](#) (const [tgba\\_explicit](#) &other)
- [tgba\\_explicit](#) & [operator=](#) (const [tgba\\_explicit](#) &other)

## Classes

- struct [transition](#)  
*Explicit transitions (used by [spot::tgba\\_explicit](#)).*

### 12.104.1 Detailed Description

Explicit representation of a [spot::tgba](#).

### 12.104.2 Member Typedef Documentation

**12.104.2.1** `typedef std::list<transition*> spot::tgba_explicit::state`

**12.104.2.2** `typedef Sgi::hash_map<const std::string, tgba_explicit::state*, string_hash> spot::tgba_explicit::ns_map [protected]`

**12.104.2.3** `typedef Sgi::hash_map<const tgba_explicit::state*, std::string, ptr_hash<tgba_explicit::state>> spot::tgba_explicit::sn_map [protected]`

### 12.104.3 Constructor & Destructor Documentation

**12.104.3.1** `spot::tgba_explicit::tgba_explicit (bdd_dict * dict)`

**12.104.3.2** `virtual spot::tgba_explicit::~~tgba_explicit () [virtual]`

**12.104.3.3** `spot::tgba_explicit::tgba_explicit (const tgba_explicit & other) [private]`

### 12.104.4 Member Function Documentation

**12.104.4.1** `state* spot::tgba_explicit::set_init_state (const std::string & state)`

**12.104.4.2** `transition* spot::tgba_explicit::create_transition (const std::string & source, const std::string & dest)`

**12.104.4.3** `transition* spot::tgba_explicit::create_transition (state * source, const state * dest)`

**12.104.4.4** `void spot::tgba_explicit::add_condition (transition * t, const ltl::formula * f)`

**12.104.4.5** `void spot::tgba_explicit::add_conditions (transition * t, bdd f)`

This assumes that all variables in *f* are known from dict.

**12.104.4.6** `void spot::tgba_explicit::declare_acceptance_condition (const ltl::formula * f)`

**12.104.4.7** `void spot::tgba_explicit::copy_acceptance_conditions_of (const tgba * a)`

Copy the acceptance conditions of a [tgba](#).

If used, this function should be called before creating any [transition](#).

**12.104.4.8** `bool spot::tgba_explicit::has_acceptance_condition (const ltl::formula * f) const`



**12.104.4.9** void spot::tgba\_explicit::add\_acceptance\_condition (transition \* *t*, const ltl::formula \* *f*)

**12.104.4.10** void spot::tgba\_explicit::add\_acceptance\_conditions (transition \* *t*, bdd *f*)

This assumes that all acceptance conditions in *f* are known from dict.

**12.104.4.11** void spot::tgba\_explicit::complement\_all\_acceptance\_conditions ()

**12.104.4.12** void spot::tgba\_explicit::merge\_transitions ()

**12.104.4.13** state\* spot::tgba\_explicit::add\_state (const std::string & *name*)

Return the tgba\_explicit::state for *name*, creating the state if it does not exist.

**12.104.4.14** virtual spot::state\* spot::tgba\_explicit::get\_init\_state () const [virtual]

Get the initial state of the automaton.

The state has been allocated with new. It is the responsibility of the caller to delete it when no longer needed.

Implements spot::tgba.

**12.104.4.15** virtual tgba\_succ\_iterator\* spot::tgba\_explicit::succ\_iter (const spot::state \* *local\_state*, const spot::state \* *global\_state* = 0, const tgba \* *global\_automaton* = 0) const [virtual]

Get an iterator over the successors of *local\_state*.

The iterator has been allocated with new. It is the responsibility of the caller to delete it when no longer needed.

During synchornized products, additional informations are passed about the entire product and its state. Recall that products can be nested, forming a tree of spot::tgba where most values are computed on demand. *global\_automaton* designate the root spot::tgba, and *global\_state* its state. This two objects can be used by succ\_iter() to restrict the set of successors to compute.

#### Parameters:

*local\_state* The state whose successors are to be explored. This pointer is not adopted in any way by succ\_iter, and it is still the caller's responsibility to delete it when appropriate (this can be done during the lifetime of the iterator).

*global\_state* In a product, the state of the global product automaton. Otherwise, 0. Like *local\_state*, *global\_state* is not adopted by succ\_iter.

*global\_automaton* In a product, the global product automaton. Otherwise, 0.

Implements spot::tgba.

**12.104.4.16** virtual bdd\_dict\* spot::tgba\_explicit::get\_dict () const [virtual]

Get the dictionary associated to the automaton.

State are represented as BDDs. The dictionary allows to map BDD variables back to formulae, and vice versa. This is useful when dealing with several automata (which may use the same BDD variable for different formula), or simply when printing.

Implements [spot::tgba](#).

**12.104.4.17** `virtual std::string spot::tgba_explicit::format_state (const spot::state * state) const` [virtual]

Format the [state](#) as a string for printing.

This formatting is the responsibility of the automata who owns the [state](#).

Implements [spot::tgba](#).

Reimplemented in [spot::tgba\\_reduc](#).

**12.104.4.18** `virtual bdd spot::tgba_explicit::all_acceptance_conditions () const` [virtual]

Return the set of all acceptance conditions used by this automaton.

The goal of the emptiness check is to ensure that a strongly connected component walks through each of these acceptance conditions. I.e., the union of the acceptance conditions of all [transition](#) in the SCC should be equal to the result of this function.

Implements [spot::tgba](#).

**12.104.4.19** `virtual bdd spot::tgba_explicit::neg_acceptance_conditions () const` [virtual]

Return the conjunction of all negated acceptance variables.

For instance if the automaton uses variables `Acc[a]`, `Acc[b]` and `Acc[c]` to describe acceptance sets, this function should return `!Acc[a] & !Acc[b] & !Acc[c]`.

This is useful when making products: each operand's condition set should be augmented with the [neg\\_acceptance\\_conditions\(\)](#) of the other operand.

Implements [spot::tgba](#).

**12.104.4.20** `virtual bdd spot::tgba_explicit::compute_support_conditions (const spot::state * state) const` [protected, virtual]

Do the actual computation of [tgba::support\\_conditions\(\)](#).

Implements [spot::tgba](#).

**12.104.4.21** `virtual bdd spot::tgba_explicit::compute_support_variables (const spot::state * state) const` [protected, virtual]

Do the actual computation of [tgba::support\\_variables\(\)](#).

Implements [spot::tgba](#).

**12.104.4.22** `bdd spot::tgba_explicit::get_acceptance_condition (const ltl::formula * f)` [protected]

**12.104.4.23** `tgba_explicit& spot::tgba_explicit::operator= (const tgba_explicit & other)` [private]

**12.104.4.24 bdd spot::tgba::support\_conditions (const state \* state) const** [inherited]

Get a formula that must hold whatever successor is taken.

**Returns:**

A formula which must be verified for all successors of *state*.

This can be as simple as `bddtrue`, or more completely the disjunction of the condition of all successors. This is used as an hint by `succ_iter()` to reduce the number of successor to compute in a product.

Sub classes should implement `compute_support_conditions()`, this function is just a wrapper that will cache the last return value for efficiency.

**12.104.4.25 bdd spot::tgba::support\_variables (const state \* state) const** [inherited]

Get the conjunctions of variables tested by the outgoing transitions of *state*.

All variables tested by outgoing transitions must be returned. This is mandatory.

This is used as an hint by some `succ_iter()` to reduce the number of successor to compute in a product.

Sub classes should implement `compute_support_variables()`, this function is just a wrapper that will cache the last return value for efficiency.

**12.104.4.26 virtual std::string spot::tgba::transition\_annotation (const tgba\_succ\_iterator \* t) const** [virtual, inherited]

Return a possible annotation for the transition pointed to by the iterator.

Implementing this function is optional; the default annotation is the empty string.

**Parameters:**

*t* a non-done `tgba_succ_iterator` for this automata

Reimplemented in `spot::tgba_product`, and `spot::tgba_tba_proxy`.

**12.104.4.27 virtual state\* spot::tgba::project\_state (const state \* s, const tgba \* t) const** [virtual, inherited]

Project a *state* on an automaton.

This converts *s*, into that corresponding `spot::state` for *t*. This is useful when you have the *state* of a product, and want restrict this *state* to a specific automata occurring in the product.

It goes without saying that *s* and *t* should be compatible (i.e., *s* is a *state* of *t*).

**Returns:**

0 if the projection fails (*s* is unrelated to *t*), or a new `state*` (the projected *state*) that must be deleted by the caller.

Reimplemented in `spot::tgba_product`, and `spot::tgba_tba_proxy`.

**12.104.4.28 virtual unsigned int spot::tgba::number\_of\_acceptance\_conditions () const** [virtual, inherited]

The number of acceptance conditions.

### 12.104.5 Member Data Documentation

**12.104.5.1** ns\_map spot::tgba\_explicit::name\_state\_map\_ [protected]

**12.104.5.2** sn\_map spot::tgba\_explicit::state\_name\_map\_ [protected]

**12.104.5.3** bdd\_dict\* spot::tgba\_explicit::dict\_ [protected]

**12.104.5.4** tgba\_explicit::state\* spot::tgba\_explicit::init\_ [protected]

**12.104.5.5** bdd spot::tgba\_explicit::all\_acceptance\_conditions\_ [mutable, protected]

**12.104.5.6** bdd spot::tgba\_explicit::neg\_acceptance\_conditions\_ [protected]

**12.104.5.7** bool spot::tgba\_explicit::all\_acceptance\_conditions\_computed\_ [mutable, protected]

The documentation for this class was generated from the following file:

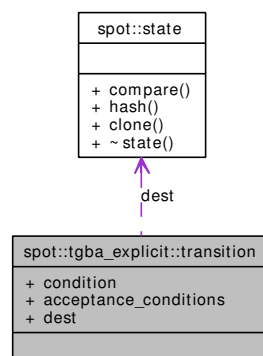
- [tgba/tgbaexplicit.hh](#)

## 12.105 spot::tgba\_explicit::transition Struct Reference

Explicit transitions (used by [spot::tgba\\_explicit](#)).

```
#include <tgba/tgbaexplicit.hh>
```

Collaboration diagram for spot::tgba\_explicit::transition:



### Public Attributes

- bdd [condition](#)
- bdd [acceptance\\_conditions](#)
- const [state](#) \* [dest](#)

### 12.105.1 Detailed Description

Explicit transitions (used by [spot::tgba\\_explicit](#)).

### 12.105.2 Member Data Documentation

#### 12.105.2.1 bdd spot::tgba\_explicit::transition::condition

#### 12.105.2.2 bdd spot::tgba\_explicit::transition::acceptance\_conditions

#### 12.105.2.3 const state\* spot::tgba\_explicit::transition::dest

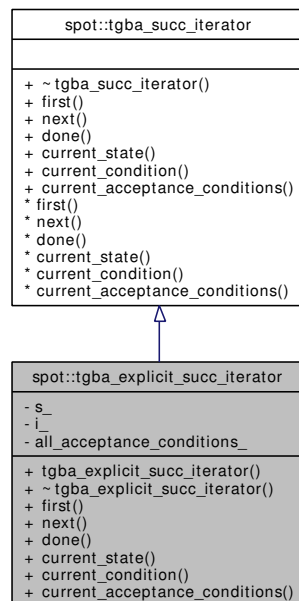
The documentation for this struct was generated from the following file:

- [tgba/tgbaexplicit.hh](#)

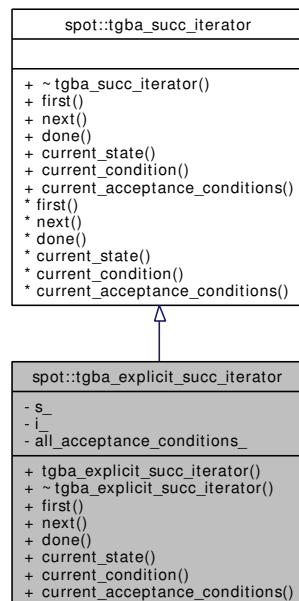
## 12.106 spot::tgba\_explicit\_succ\_iterator Class Reference

```
#include <tgba/tgbaexplicit.hh>
```

Inheritance diagram for spot::tgba\_explicit\_succ\_iterator:



Collaboration diagram for spot::tgba\_explicit\_succ\_iterator:



## Public Member Functions

- [tgba\\_explicit\\_succ\\_iterator](#) (const [tgba\\_explicit::state](#) \*s, bdd all\_acc)
- virtual [~tgba\\_explicit\\_succ\\_iterator](#) ()
- virtual void [first](#) ()

*Position the iterator on the first successor (if any).*

- virtual void [next](#) ()

*Jump to the next successor (if any).*

- virtual bool [done](#) () const

*Check whether the iteration is finished.*

- virtual [state\\_explicit](#) \* [current\\_state](#) () const

*Get the [state](#) of the current successor.*

- virtual bdd [current\\_condition](#) () const

*Get the condition on the transition leading to this successor.*

- virtual bdd [current\\_acceptance\\_conditions](#) () const

*Get the acceptance conditions on the transition leading to this successor.*

## Private Attributes

- const [tgba\\_explicit::state](#) \* [s\\_](#)
- [tgba\\_explicit::state::const\\_iterator](#) [i\\_](#)
- bdd [all\\_acceptance\\_conditions\\_](#)

### 12.106.1 Detailed Description

Successor iterators used by [spot::tgba\\_explicit](#).

### 12.106.2 Constructor & Destructor Documentation

**12.106.2.1** `spot::tgba_explicit_succ_iterator::tgba_explicit_succ_iterator (const tgba_explicit::state * s, bdd all_acc)`

**12.106.2.2** `virtual spot::tgba_explicit_succ_iterator::~~tgba_explicit_succ_iterator ()` `[inline, virtual]`

### 12.106.3 Member Function Documentation

**12.106.3.1** `virtual void spot::tgba_explicit_succ_iterator::first ()` `[virtual]`

Position the iterator on the first successor (if any).

This method can be called several times to make multiple passes over successors.

#### Warning:

One should always call `done ()` to ensure there is a successor, even after `first ()`. A common trap is to assume that there is at least one successor: this is wrong.

Implements [spot::tgba\\_succ\\_iterator](#).

**12.106.3.2** `virtual void spot::tgba_explicit_succ_iterator::next ()` `[virtual]`

Jump to the next successor (if any).

#### Warning:

Again, one should always call `done ()` to ensure there is a successor.

Implements [spot::tgba\\_succ\\_iterator](#).

**12.106.3.3** `virtual bool spot::tgba_explicit_succ_iterator::done () const` `[virtual]`

Check whether the iteration is finished.

This function should be called after any call to `first ()` or `next ()` and before any enquiry about the current `state`.

The usual way to do this is with a `for` loop.

```
for (s->first(); !s->done(); s->next())
    ...
```

Implements [spot::tgba\\_succ\\_iterator](#).

**12.106.3.4** `virtual state_explicit* spot::tgba_explicit_succ_iterator::current_state () const` [virtual]

Get the [state](#) of the current successor.

Note that the same [state](#) may occur at different points in the iteration. These actually correspond to the same destination. It just means there were several transitions, with different conditions, leading to the same [state](#).

Implements [spot::tgba\\_succ\\_iterator](#).

**12.106.3.5** `virtual bdd spot::tgba_explicit_succ_iterator::current_condition () const` [virtual]

Get the condition on the transition leading to this successor.

This is a boolean function of atomic propositions.

Implements [spot::tgba\\_succ\\_iterator](#).

**12.106.3.6** `virtual bdd spot::tgba_explicit_succ_iterator::current_acceptance_conditions () const` [virtual]

Get the acceptance conditions on the transition leading to this successor.

Implements [spot::tgba\\_succ\\_iterator](#).

## 12.106.4 Member Data Documentation

**12.106.4.1** `const tgba_explicit::state* spot::tgba_explicit_succ_iterator::s_` [private]

**12.106.4.2** `tgba_explicit::state::const_iterator spot::tgba_explicit_succ_iterator::i_` [private]

**12.106.4.3** `bdd spot::tgba_explicit_succ_iterator::all_acceptance_conditions_` [private]

The documentation for this class was generated from the following file:

- [tgba/tgbaexplicit.hh](#)

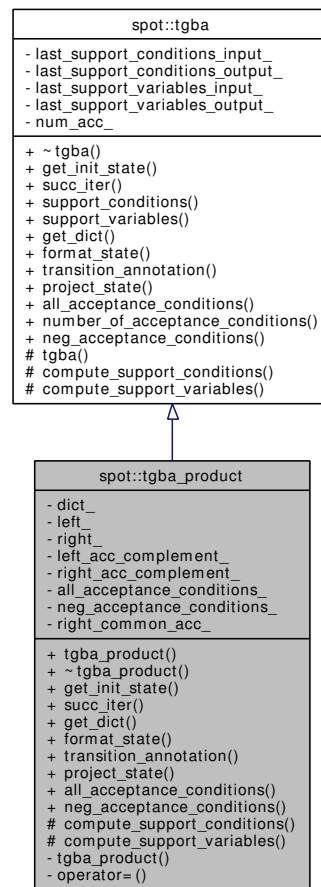
## 12.107 spot::tgba\_product Class Reference

A lazy product. (States are computed on the fly.).

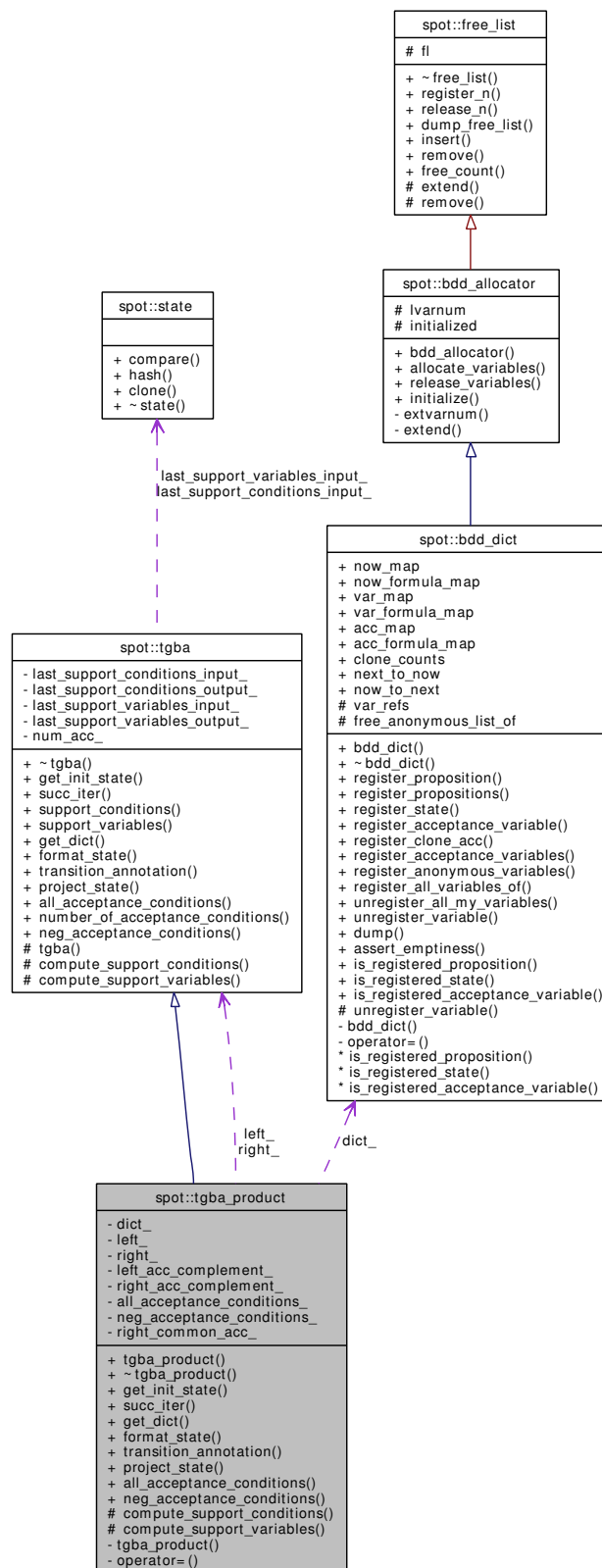
```
#include <tgba/tgbaproduct.hh>
```



Inheritance diagram for spot::tgba\_product:



Collaboration diagram for spot::tgba\_product:



## Public Member Functions

- [tgba\\_product](#) (const [tgba](#) \*left, const [tgba](#) \*right)  
*Constructor.*
- virtual [~tgba\\_product](#) ()
- virtual [state](#) \* [get\\_init\\_state](#) () const  
*Get the initial [state](#) of the automaton.*
- virtual [tgba\\_succ\\_iterator\\_product](#) \* [succ\\_iter](#) (const [state](#) \*local\_state, const [state](#) \*global\_state=0, const [tgba](#) \*global\_automaton=0) const  
*Get an iterator over the successors of local\_state.*
- virtual [bdd\\_dict](#) \* [get\\_dict](#) () const  
*Get the dictionary associated to the automaton.*
- virtual std::string [format\\_state](#) (const [state](#) \*state) const  
*Format the [state](#) as a string for printing.*
- virtual std::string [transition\\_annotation](#) (const [tgba\\_succ\\_iterator](#) \*t) const  
*Return a possible annotation for the transition pointed to by the iterator.*
- virtual [state](#) \* [project\\_state](#) (const [state](#) \*s, const [tgba](#) \*t) const  
*Project a [state](#) on an automaton.*
- virtual bdd [all\\_acceptance\\_conditions](#) () const  
*Return the set of all acceptance conditions used by this automaton.*
- virtual bdd [neg\\_acceptance\\_conditions](#) () const  
*Return the conjunction of all negated acceptance variables.*
- bdd [support\\_conditions](#) (const [state](#) \*state) const  
*Get a formula that must hold whatever successor is taken.*
- bdd [support\\_variables](#) (const [state](#) \*state) const  
*Get the conjunctions of variables tested by the outgoing transitions of [state](#).*
- virtual unsigned int [number\\_of\\_acceptance\\_conditions](#) () const  
*The number of acceptance conditions.*

## Protected Member Functions

- virtual bdd [compute\\_support\\_conditions](#) (const [state](#) \*state) const  
*Do the actual computation of [tgba::support\\_conditions\(\)](#).*
- virtual bdd [compute\\_support\\_variables](#) (const [state](#) \*state) const  
*Do the actual computation of [tgba::support\\_variables\(\)](#).*

## Private Member Functions

- [tgba\\_product](#) (const [tgba\\_product](#) &)
- [tgba\\_product](#) & [operator=](#) (const [tgba\\_product](#) &)

## Private Attributes

- [bdd\\_dict](#) \* [dict\\_](#)
- const [tgba](#) \* [left\\_](#)
- const [tgba](#) \* [right\\_](#)
- [bdd](#) [left\\_acc\\_complement\\_](#)
- [bdd](#) [right\\_acc\\_complement\\_](#)
- [bdd](#) [all\\_acceptance\\_conditions\\_](#)
- [bdd](#) [neg\\_acceptance\\_conditions\\_](#)
- [bddPair](#) \* [right\\_common\\_acc\\_](#)

### 12.107.1 Detailed Description

A lazy product. (States are computed on the fly.).

### 12.107.2 Constructor & Destructor Documentation

#### 12.107.2.1 spot::tgba\_product::tgba\_product (const [tgba](#) \* *left*, const [tgba](#) \* *right*)

Constructor.

#### Parameters:

*left* The left automata in the product.

*right* The right automata in the product. Do not be fooled by these arguments: a product is commutative.

#### 12.107.2.2 virtual spot::tgba\_product::~~tgba\_product () [virtual]

#### 12.107.2.3 spot::tgba\_product::tgba\_product (const [tgba\\_product](#) &) [private]

### 12.107.3 Member Function Documentation

#### 12.107.3.1 virtual [state](#)\* spot::tgba\_product::get\_init\_state () const [virtual]

Get the initial [state](#) of the automaton.

The [state](#) has been allocated with `new`. It is the responsibility of the caller to `delete` it when no longer needed.

Implements [spot::tgba](#).

**12.107.3.2 virtual tgba\_succ\_iterator\_product\* spot::tgba\_product::succ\_iter (const state \* *local\_state*, const state \* *global\_state* = 0, const tgba \* *global\_automaton* = 0) const** [virtual]

Get an iterator over the successors of *local\_state*.

The iterator has been allocated with `new`. It is the responsibility of the caller to `delete` it when no longer needed.

During synchornized products, additional informations are passed about the entire product and its *state*. Recall that products can be nested, forming a tree of `spot::tgba` where most values are computed on demand. *global\_automaton* designate the root `spot::tgba`, and *global\_state* its *state*. This two objects can be used by `succ_iter()` to restrict the set of successors to compute.

#### Parameters:

*local\_state* The *state* whose successors are to be explored. This pointer is not adopted in any way by `succ_iter`, and it is still the caller's responsibility to delete it when appropriate (this can be done during the lifetime of the iterator).

*global\_state* In a product, the *state* of the global product automaton. Otherwise, 0. Like *locale\_state*, *global\_state* is not adopted by `succ_iter`.

*global\_automaton* In a product, the global product automaton. Otherwise, 0.

Implements `spot::tgba`.

**12.107.3.3 virtual bdd\_dict\* spot::tgba\_product::get\_dict () const** [virtual]

Get the dictionary associated to the automaton.

State are represented as BDDs. The dictionary allows to map BDD variables back to formulae, and vice versa. This is useful when dealing with several automata (which may use the same BDD variable for different formula), or simply when printing.

Implements `spot::tgba`.

**12.107.3.4 virtual std::string spot::tgba\_product::format\_state (const state \* *state*) const** [virtual]

Format the *state* as a string for printing.

This formatting is the responsibility of the automata who owns the *state*.

Implements `spot::tgba`.

**12.107.3.5 virtual std::string spot::tgba\_product::transition\_annotation (const tgba\_succ\_iterator \* *t*) const** [virtual]

Return a possible annotation for the transition pointed to by the iterator.

Implementing this function is optional; the default annotation it the empty string.

#### Parameters:

*t* a non-done `tgba_succ_iterator` for this automata

Reimplemented from `spot::tgba`.

**12.107.3.6** `virtual state* spot::tgba_product::project_state (const state * s, const tgba * t) const` [virtual]

Project a [state](#) on an automaton.

This converts *s*, into that corresponding [spot::state](#) for *t*. This is useful when you have the [state](#) of a product, and want restrict this [state](#) to a specific automata occuring in the product.

It goes without saying that *s* and *t* should be compatible (i.e., *s* is a [state](#) of *t*).

**Returns:**

0 if the projection fails (*s* is unrelated to *t*), or a new `state*` (the projected [state](#)) that must be deleted by the caller.

Reimplemented from [spot::tgba](#).

**12.107.3.7** `virtual bdd spot::tgba_product::all_acceptance_conditions () const` [virtual]

Return the set of all acceptance conditions used by this automaton.

The goal of the emptiness check is to ensure that a strongly connected component walks through each of these acceptance conditions. I.e., the union of the acceptance conditions of all transition in the SCC should be equal to the result of this function.

Implements [spot::tgba](#).

**12.107.3.8** `virtual bdd spot::tgba_product::neg_acceptance_conditions () const` [virtual]

Return the conjunction of all negated acceptance variables.

For instance if the automaton uses variables `Acc[a]`, `Acc[b]` and `Acc[c]` to describe acceptance sets, this function should return `!Acc[a] & !Acc[b] & !Acc[c]`.

This is useful when making products: each operand's condition set should be augmented with the [neg\\_acceptance\\_conditions\(\)](#) of the other operand.

Implements [spot::tgba](#).

**12.107.3.9** `virtual bdd spot::tgba_product::compute_support_conditions (const state * state) const` [protected, virtual]

Do the actual computation of [tgba::support\\_conditions\(\)](#).

Implements [spot::tgba](#).

**12.107.3.10** `virtual bdd spot::tgba_product::compute_support_variables (const state * state) const` [protected, virtual]

Do the actual computation of [tgba::support\\_variables\(\)](#).

Implements [spot::tgba](#).

**12.107.3.11** `tgba_product& spot::tgba_product::operator= (const tgba_product &) [private]`

**12.107.3.12 bdd spot::tgba::support\_conditions (const state \* state) const** [inherited]

Get a formula that must hold whatever successor is taken.

**Returns:**

A formula which must be verified for all successors of *state*.

This can be as simple as `bddtrue`, or more completely the disjunction of the condition of all successors. This is used as an hint by `succ_iter()` to reduce the number of successor to compute in a product.

Sub classes should implement `compute_support_conditions()`, this function is just a wrapper that will cache the last return value for efficiency.

**12.107.3.13 bdd spot::tgba::support\_variables (const state \* state) const** [inherited]

Get the conjunctions of variables tested by the outgoing transitions of *state*.

All variables tested by outgoing transitions must be returned. This is mandatory.

This is used as an hint by some `succ_iter()` to reduce the number of successor to compute in a product.

Sub classes should implement `compute_support_variables()`, this function is just a wrapper that will cache the last return value for efficiency.

**12.107.3.14 virtual unsigned int spot::tgba::number\_of\_acceptance\_conditions () const** [virtual, inherited]

The number of acceptance conditions.

**12.107.4 Member Data Documentation****12.107.4.1 bdd\_dict\* spot::tgba\_product::dict\_** [private]**12.107.4.2 const tgba\* spot::tgba\_product::left\_** [private]**12.107.4.3 const tgba\* spot::tgba\_product::right\_** [private]**12.107.4.4 bdd spot::tgba\_product::left\_acc\_complement\_** [private]**12.107.4.5 bdd spot::tgba\_product::right\_acc\_complement\_** [private]**12.107.4.6 bdd spot::tgba\_product::all\_acceptance\_conditions\_** [private]**12.107.4.7 bdd spot::tgba\_product::neg\_acceptance\_conditions\_** [private]**12.107.4.8 bddPair\* spot::tgba\_product::right\_common\_acc\_** [private]

The documentation for this class was generated from the following file:

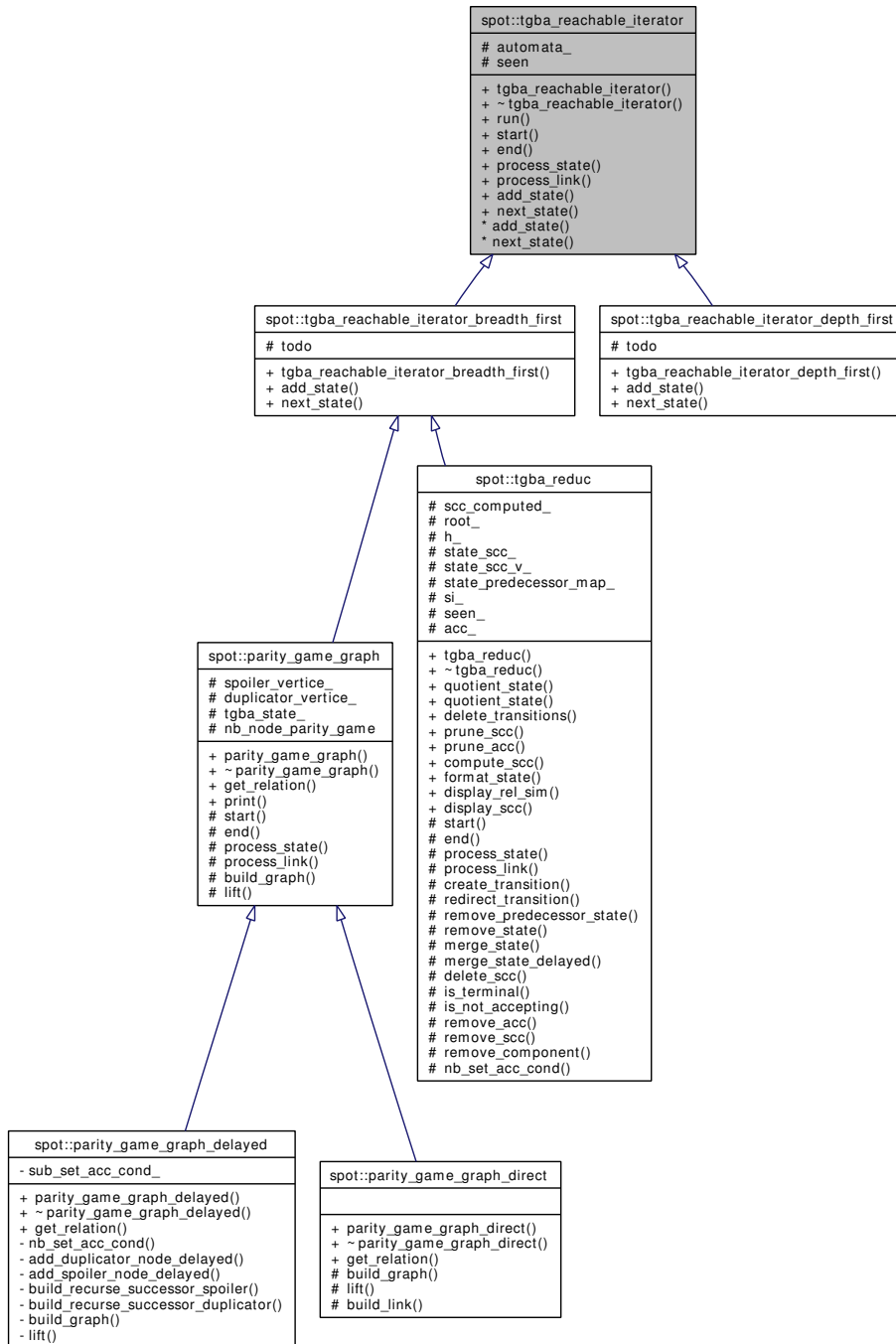
- [tgba/tgbaproduct.hh](#)

## 12.108 spot::tgba\_reachable\_iterator Class Reference

Iterate over all reachable states of a [spot::tgba](#).

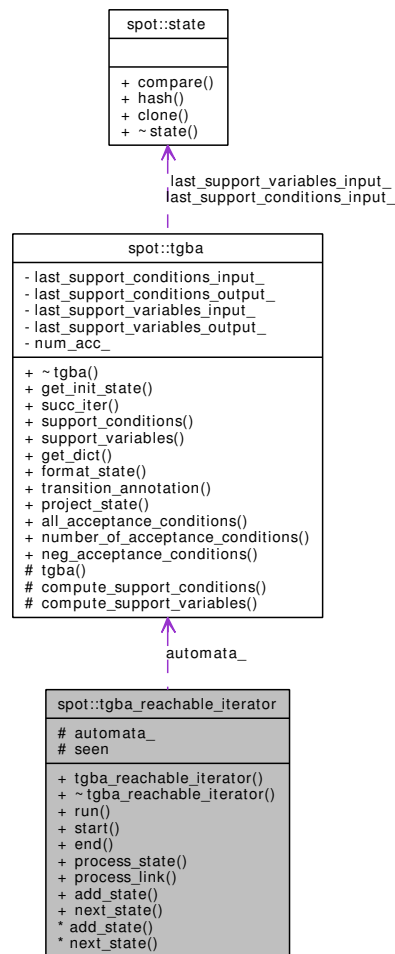
```
#include <tgbaalgos/reachiter.hh>
```

Inheritance diagram for spot::tgba\_reachable\_iterator:





Collaboration diagram for spot::tgba\_reachable\_iterator:



## Public Member Functions

- [tgba\\_reachable\\_iterator](#) (const [tgba](#) \*a)
- virtual [~tgba\\_reachable\\_iterator](#) ()
- void [run](#) ()  
*Iterate over all reachable states of a [spot::tgba](#).*
- virtual void [start](#) ()  
*Called by [run\(\)](#) before starting its iteration.*
- virtual void [end](#) ()  
*Called by [run\(\)](#) once all states have been explored.*
- virtual void [process\\_state](#) (const [state](#) \*s, int n, [tgba\\_succ\\_iterator](#) \*si)
- virtual void [process\\_link](#) (const [state](#) \*in\_s, int in, const [state](#) \*out\_s, int out, const [tgba\\_succ\\_iterator](#) \*si)

## Todo list management.

Called by `run()` to register newly discovered states.

`spot::tgba_reachable_iterator_depth_first` and `spot::tgba_reachable_iterator_breadth_first` offer two precanned implementations for these functions.

- virtual void `add_state` (const `state` \*s)=0
- virtual const `state` \* `next_state` ()=0

Called by `run()` to obtain the.

## Protected Types

- typedef `Sgi::hash_map`< const `state` \*, int, `state_ptr_hash`, `state_ptr_equal` > `seen_map`

## Protected Attributes

- const `tgba` \* `automata_`  
The `spot::tgba` to explore.
- `seen_map` seen  
States already seen.

### 12.108.1 Detailed Description

Iterate over all reachable states of a `spot::tgba`.

### 12.108.2 Member Typedef Documentation

**12.108.2.1** `typedef Sgi::hash_map<const state*, int, state_ptr_hash, state_ptr_equal> spot::tgba_reachable_iterator::seen_map` [protected]

Reimplemented in `spot::tgba_reduc`.

### 12.108.3 Constructor & Destructor Documentation

**12.108.3.1** `spot::tgba_reachable_iterator::tgba_reachable_iterator (const tgba * a)`

**12.108.3.2** `virtual spot::tgba_reachable_iterator::~~tgba_reachable_iterator ()` [virtual]

### 12.108.4 Member Function Documentation

**12.108.4.1** `void spot::tgba_reachable_iterator::run ()`

Iterate over all reachable states of a `spot::tgba`.

This is a template method that will call `add_state()`, `next_state()`, `start()`, `end()`, `process_state()`, and `process_link()`, while it iterate over `state`.

**12.108.4.2** `virtual void spot::tgba_reachable_iterator::add_state (const state * s) [pure virtual]`

Implemented in [spot::tgba\\_reachable\\_iterator\\_depth\\_first](#), and [spot::tgba\\_reachable\\_iterator\\_breadth\\_first](#).

**12.108.4.3** `virtual const state* spot::tgba_reachable_iterator::next_state () [pure virtual]`

Called by [run\(\)](#) to obtain the.

Implemented in [spot::tgba\\_reachable\\_iterator\\_depth\\_first](#), and [spot::tgba\\_reachable\\_iterator\\_breadth\\_first](#).

**12.108.4.4** `virtual void spot::tgba_reachable_iterator::start () [virtual]`

Called by [run\(\)](#) before starting its iteration.

Reimplemented in [spot::tgba\\_reduc](#), and [spot::parity\\_game\\_graph](#).

**12.108.4.5** `virtual void spot::tgba_reachable_iterator::end () [virtual]`

Called by [run\(\)](#) once all states have been explored.

Reimplemented in [spot::tgba\\_reduc](#), and [spot::parity\\_game\\_graph](#).

**12.108.4.6** `virtual void spot::tgba_reachable_iterator::process_state (const state * s, int n, tgba_succ_iterator * si) [virtual]`

Called by [run\(\)](#) to process a [state](#).

#### Parameters:

- s* The current [state](#).
- n* A unique number assigned to *s*.
- si* The [spot::tgba\\_succ\\_iterator](#) for *s*.

Reimplemented in [spot::tgba\\_reduc](#), and [spot::parity\\_game\\_graph](#).

**12.108.4.7** `virtual void spot::tgba_reachable_iterator::process_link (const state * in_s, int in, const state * out_s, int out, const tgba_succ_iterator * si) [virtual]`

Called by [run\(\)](#) to process a transition.

#### Parameters:

- in\_s* The source [state](#)
- in* The source [state](#) number.
- out\_s* The destination [state](#)
- out* The destination [state](#) number.
- si* The [spot::tgba\\_succ\\_iterator](#) positionned on the current transition.

The *in\_s* and *out\_s* states are owned by the [spot::tgba\\_reachable\\_iterator](#) instance and destroyed when the instance is destroyed.

### 12.108.5 Member Data Documentation

#### 12.108.5.1 `const tgba* spot::tgba_reachable_iterator::automata_` [protected]

The [spot::tgba](#) to explore.

#### 12.108.5.2 `seen_map spot::tgba_reachable_iterator::seen` [protected]

States already seen.

The documentation for this class was generated from the following file:

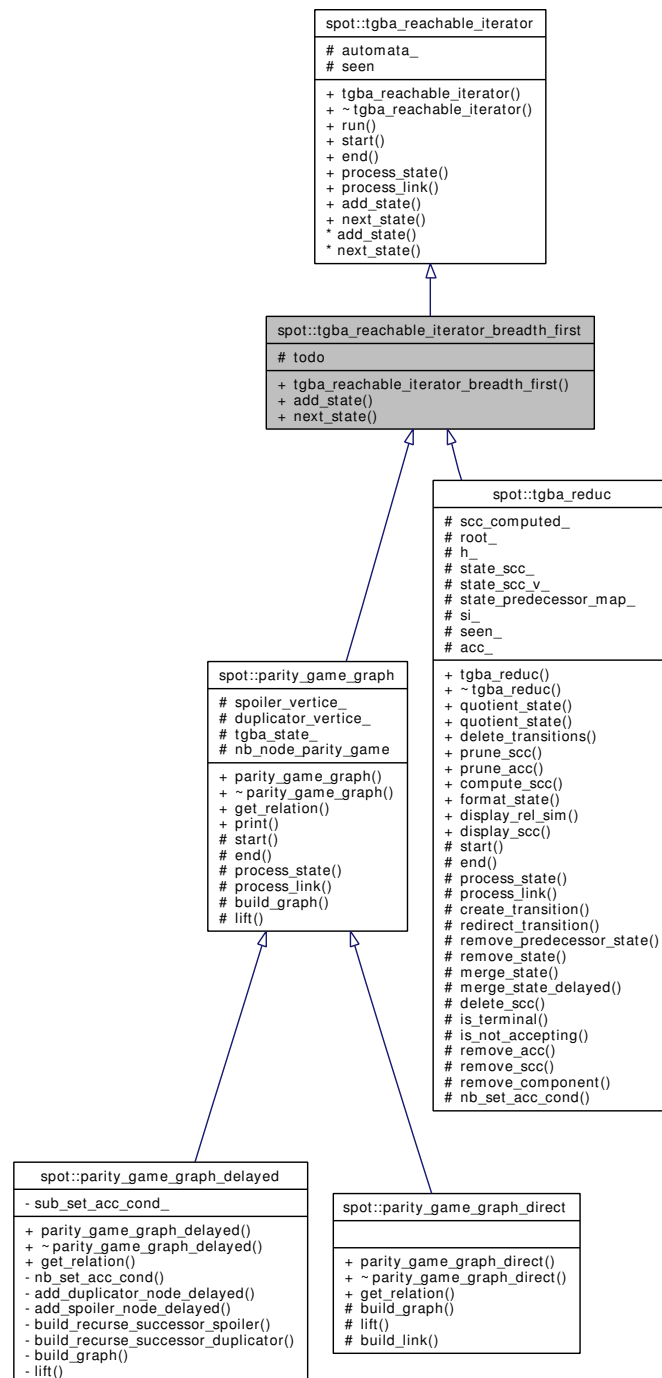
- `tgbaalgos/reachiter.hh`

## 12.109 `spot::tgba_reachable_iterator_breadth_first` Class Reference

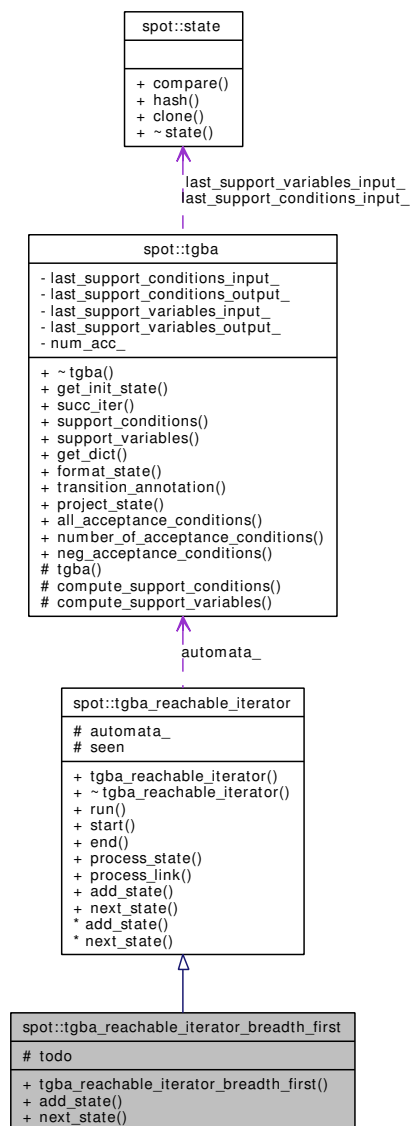
An implementation of [spot::tgba\\_reachable\\_iterator](#) that browses states breadth first.

```
#include <tgbaalgos/reachiter.hh>
```

Inheritance diagram for spot::tgba\_reachable\_iterator\_breadth\_first:



Collaboration diagram for spot::tgba\_reachable\_iterator\_breadth\_first:



## Public Member Functions

- [tgba\\_reachable\\_iterator\\_breadth\\_first](#) (const [tgba](#) \*a)
- virtual void [add\\_state](#) (const [state](#) \*s)
- virtual const [state](#) \* [next\\_state](#) ()

*Called by [run\(\)](#) to obtain the.*

- void [run](#) ()

*Iterate over all reachable states of a [spot::tgba](#).*

- virtual void [start](#) ()

*Called by [run\(\)](#) before starting its iteration.*

- virtual void `end()`  
*Called by `run()` once all states have been explored.*
- virtual void `process_state` (const `state` \*s, int n, `tgba_succ_iterator` \*si)
- virtual void `process_link` (const `state` \*in\_s, int in, const `state` \*out\_s, int out, const `tgba_succ_iterator` \*si)

### Protected Types

- typedef `Sgi::hash_map< const state *, int, state_ptr_hash, state_ptr_equal >` `seen_map`

### Protected Attributes

- `std::deque< const state * >` `todo`  
*A queue of states yet to explore.*
- const `tgba` \* `automata_`  
*The `spot::tgba` to explore.*
- `seen_map` `seen`  
*States already seen.*

## 12.109.1 Detailed Description

An implementation of `spot::tgba_reachable_iterator` that browses states breadth first.

## 12.109.2 Member Typedef Documentation

**12.109.2.1** `typedef Sgi::hash_map<const state*, int, state_ptr_hash, state_ptr_equal> spot::tgba_reachable_iterator::seen_map` [protected, inherited]

Reimplemented in `spot::tgba_reduc`.

## 12.109.3 Constructor & Destructor Documentation

**12.109.3.1** `spot::tgba_reachable_iterator_breadth_first::tgba_reachable_iterator_breadth_first (const tgba * a)`

## 12.109.4 Member Function Documentation

**12.109.4.1** `virtual void spot::tgba_reachable_iterator_breadth_first::add_state (const state * s)` [virtual]

Implements `spot::tgba_reachable_iterator`.

**12.109.4.2** `virtual const state* spot::tgba_reachable_iterator_breadth_first::next_state ()` [virtual]

Called by [run\(\)](#) to obtain the.

Implements [spot::tgba\\_reachable\\_iterator](#).

**12.109.4.3** `void spot::tgba_reachable_iterator::run ()` [inherited]

Iterate over all reachable states of a [spot::tgba](#).

This is a template method that will call [add\\_state\(\)](#), [next\\_state\(\)](#), [start\(\)](#), [end\(\)](#), [process\\_state\(\)](#), and [process\\_link\(\)](#), while it iterate over [state](#).

**12.109.4.4** `virtual void spot::tgba_reachable_iterator::start ()` [virtual, inherited]

Called by [run\(\)](#) before starting its iteration.

Reimplemented in [spot::tgba\\_reduc](#), and [spot::parity\\_game\\_graph](#).

**12.109.4.5** `virtual void spot::tgba_reachable_iterator::end ()` [virtual, inherited]

Called by [run\(\)](#) once all states have been explored.

Reimplemented in [spot::tgba\\_reduc](#), and [spot::parity\\_game\\_graph](#).

**12.109.4.6** `virtual void spot::tgba_reachable_iterator::process_state (const state * s, int n, tgba_succ_iterator * si)` [virtual, inherited]

Called by [run\(\)](#) to process a [state](#).

#### Parameters:

- s* The current [state](#).
- n* A unique number assigned to *s*.
- si* The [spot::tgba\\_succ\\_iterator](#) for *s*.

Reimplemented in [spot::tgba\\_reduc](#), and [spot::parity\\_game\\_graph](#).

**12.109.4.7** `virtual void spot::tgba_reachable_iterator::process_link (const state * in_s, int in, const state * out_s, int out, const tgba_succ_iterator * si)` [virtual, inherited]

Called by [run\(\)](#) to process a transition.

#### Parameters:

- in\_s* The source [state](#)
- in* The source [state](#) number.
- out\_s* The destination [state](#)
- out* The destination [state](#) number.
- si* The [spot::tgba\\_succ\\_iterator](#) positionned on the current transition.

The *in\_s* and *out\_s* states are owned by the [spot::tgba\\_reachable\\_iterator](#) instance and destroyed when the instance is destroyed.



### 12.109.5 Member Data Documentation

#### 12.109.5.1 std::deque<const state\*> spot::tgba\_reachable\_iterator\_breadth\_first::todo [protected]

A queue of states yet to explore.

#### 12.109.5.2 const tgba\* spot::tgba\_reachable\_iterator::automata\_ [protected, inherited]

The [spot::tgba](#) to explore.

#### 12.109.5.3 seen\_map spot::tgba\_reachable\_iterator::seen [protected, inherited]

States already seen.

The documentation for this class was generated from the following file:

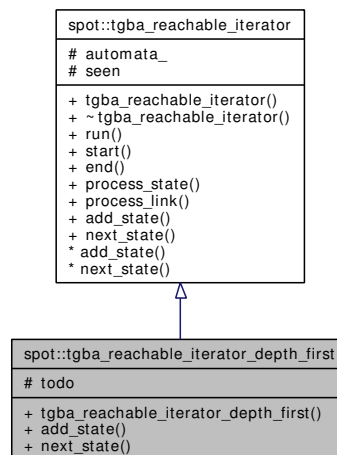
- tgbaalgos/[reachiter.hh](#)

## 12.110 spot::tgba\_reachable\_iterator\_depth\_first Class Reference

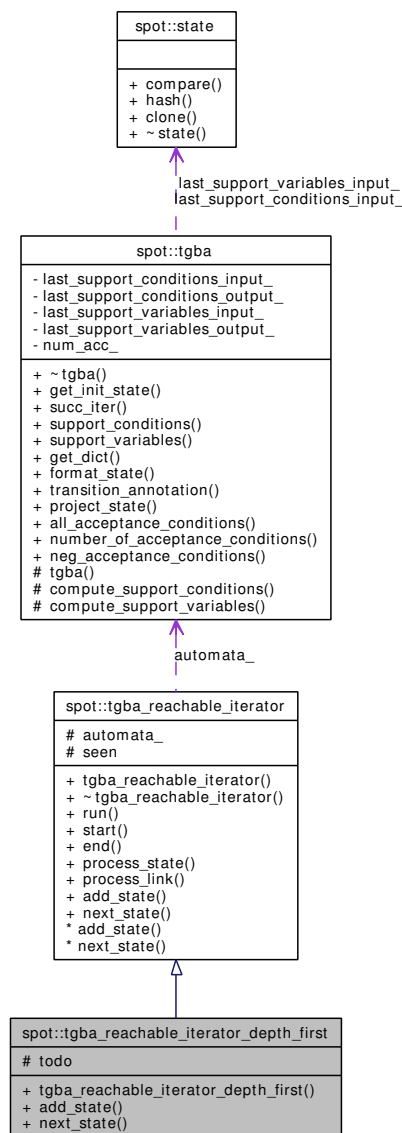
An implementation of [spot::tgba\\_reachable\\_iterator](#) that browses states depth first.

```
#include <tgbaalgos/reachiter.hh>
```

Inheritance diagram for spot::tgba\_reachable\_iterator\_depth\_first:



Collaboration diagram for spot::tgba\_reachable\_iterator\_depth\_first:



## Public Member Functions

- [tgba\\_reachable\\_iterator\\_depth\\_first](#) (const [tgba](#) \*a)
- virtual void [add\\_state](#) (const [state](#) \*s)
- virtual const [state](#) \* [next\\_state](#) ()

*Called by [run\(\)](#) to obtain the.*

- void [run](#) ()

*Iterate over all reachable states of a [spot::tgba](#).*

- virtual void [start](#) ()

*Called by [run\(\)](#) before starting its iteration.*

- virtual void `end()`  
*Called by `run()` once all states have been explored.*
- virtual void `process_state` (const `state` \*s, int n, `tgba_succ_iterator` \*si)
- virtual void `process_link` (const `state` \*in\_s, int in, const `state` \*out\_s, int out, const `tgba_succ_iterator` \*si)

### Protected Types

- typedef `Sgi::hash_map< const state *, int, state_ptr_hash, state_ptr_equal > seen_map`

### Protected Attributes

- `std::stack< const state * > todo`  
*A stack of states yet to explore.*
- const `tgba` \* `automata_`  
*The `spot::tgba` to explore.*
- `seen_map` `seen`  
*States already seen.*

## 12.110.1 Detailed Description

An implementation of `spot::tgba_reachable_iterator` that browses states depth first.

## 12.110.2 Member Typedef Documentation

**12.110.2.1** `typedef Sgi::hash_map<const state*, int, state_ptr_hash, state_ptr_equal> spot::tgba_reachable_iterator::seen_map` [protected, inherited]

Reimplemented in `spot::tgba_reduc`.

## 12.110.3 Constructor & Destructor Documentation

**12.110.3.1** `spot::tgba_reachable_iterator_depth_first::tgba_reachable_iterator_depth_first (const tgba * a)`

## 12.110.4 Member Function Documentation

**12.110.4.1** `virtual void spot::tgba_reachable_iterator_depth_first::add_state (const state * s)` [virtual]

Implements `spot::tgba_reachable_iterator`.

**12.110.4.2** `virtual const state* spot::tgba_reachable_iterator_depth_first::next_state ()` [virtual]

Called by [run\(\)](#) to obtain the.

Implements [spot::tgba\\_reachable\\_iterator](#).

**12.110.4.3** `void spot::tgba_reachable_iterator::run ()` [inherited]

Iterate over all reachable states of a [spot::tgba](#).

This is a template method that will call [add\\_state\(\)](#), [next\\_state\(\)](#), [start\(\)](#), [end\(\)](#), [process\\_state\(\)](#), and [process\\_link\(\)](#), while it iterate over [state](#).

**12.110.4.4** `virtual void spot::tgba_reachable_iterator::start ()` [virtual, inherited]

Called by [run\(\)](#) before starting its iteration.

Reimplemented in [spot::tgba\\_reduc](#), and [spot::parity\\_game\\_graph](#).

**12.110.4.5** `virtual void spot::tgba_reachable_iterator::end ()` [virtual, inherited]

Called by [run\(\)](#) once all states have been explored.

Reimplemented in [spot::tgba\\_reduc](#), and [spot::parity\\_game\\_graph](#).

**12.110.4.6** `virtual void spot::tgba_reachable_iterator::process_state (const state * s, int n, tgba_succ_iterator * si)` [virtual, inherited]

Called by [run\(\)](#) to process a [state](#).

#### Parameters:

- s* The current [state](#).
- n* A unique number assigned to *s*.
- si* The [spot::tgba\\_succ\\_iterator](#) for *s*.

Reimplemented in [spot::tgba\\_reduc](#), and [spot::parity\\_game\\_graph](#).

**12.110.4.7** `virtual void spot::tgba_reachable_iterator::process_link (const state * in_s, int in, const state * out_s, int out, const tgba_succ_iterator * si)` [virtual, inherited]

Called by [run\(\)](#) to process a transition.

#### Parameters:

- in\_s* The source [state](#)
- in* The source [state](#) number.
- out\_s* The destination [state](#)
- out* The destination [state](#) number.
- si* The [spot::tgba\\_succ\\_iterator](#) positionned on the current transition.

The *in\_s* and *out\_s* states are owned by the [spot::tgba\\_reachable\\_iterator](#) instance and destroyed when the instance is destroyed.

### 12.110.5 Member Data Documentation

**12.110.5.1** `std::stack<const state*>` `spot::tgba_reachable_iterator_depth_first::todo`  
[protected]

A stack of states yet to explore.

**12.110.5.2** `const tgba* spot::tgba_reachable_iterator::automata_` [protected, inherited]

The [spot::tgba](#) to explore.

**12.110.5.3** `seen_map spot::tgba_reachable_iterator::seen` [protected, inherited]

States already seen.

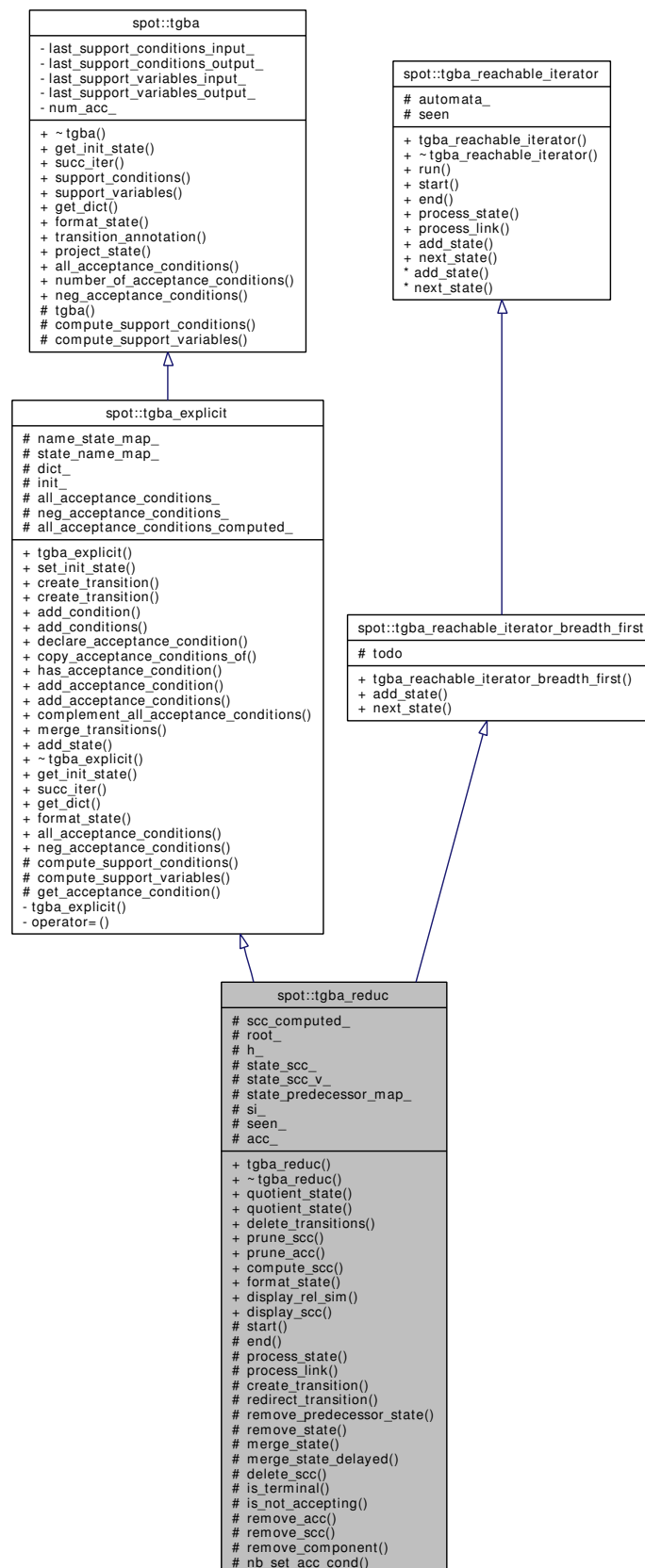
The documentation for this class was generated from the following file:

- `tgbaalgos/`[reachiter.hh](#)

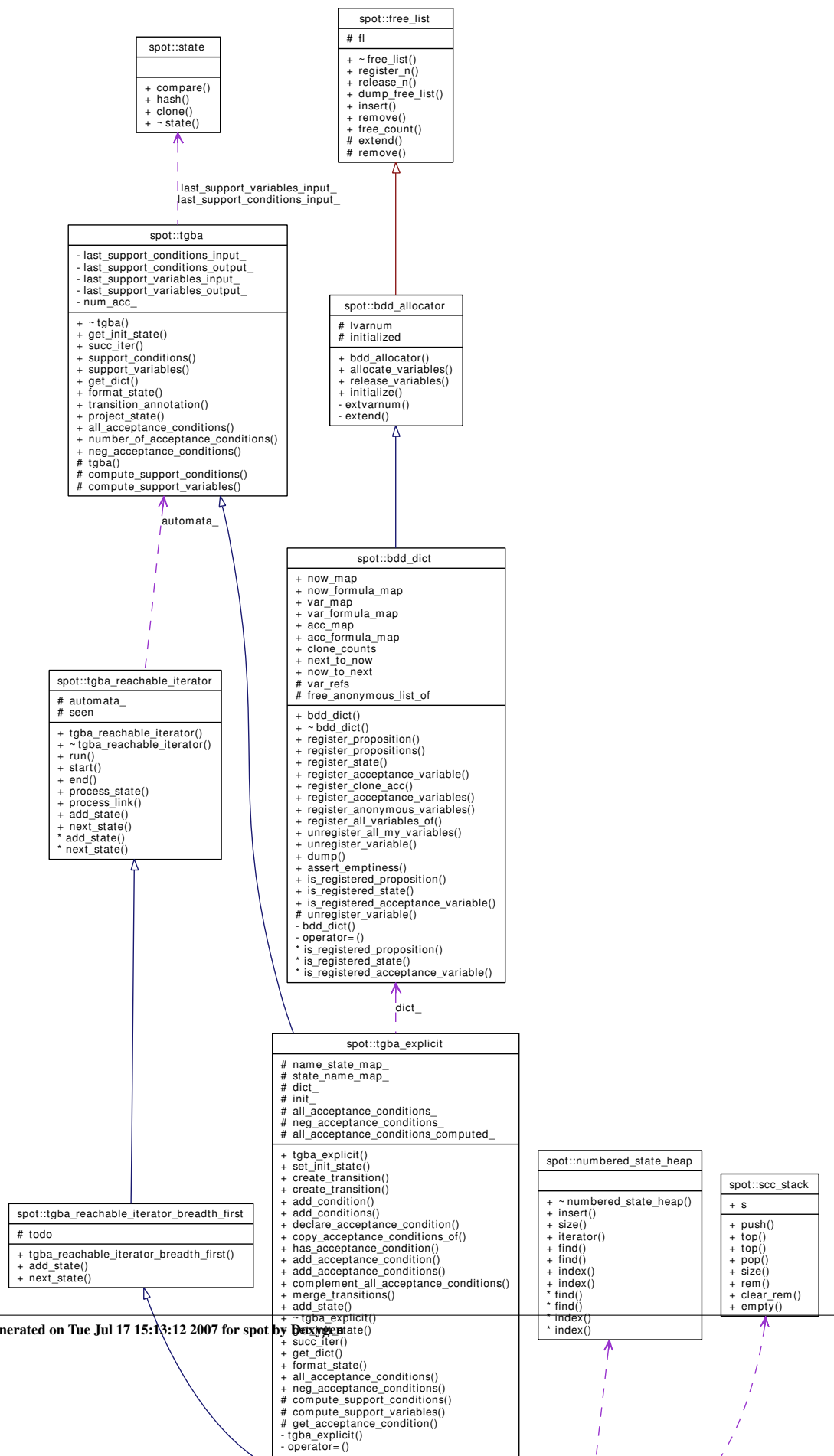
## 12.111 `spot::tgba_reduc` Class Reference

```
#include <tgba/tgbareduc.hh>
```

Inheritance diagram for spot::tgba\_reduc:



Collaboration diagram for spot::tgba\_reduc:



## Public Types

- typedef std::list< transition \* > [state](#)

## Public Member Functions

- [tgba\\_reduc](#) (const [tgba](#) \*a, const [numbered\\_state\\_heap\\_factory](#) \*nshf=numbered\_state\_heap\_hash\_map\_factory::instance())
- [~tgba\\_reduc](#) ()
- void [quotient\\_state](#) (direct\_simulation\_relation \*rel)
- void [quotient\\_state](#) (delayed\_simulation\_relation \*rel)
- void [delete\\_transitions](#) (simulation\_relation \*rel)  
*Delete some transitions with help of a simulation relation.*
- void [prune\\_scc](#) ()  
*Remove all [state](#) which not lead to an accepting cycle.*
- void [prune\\_acc](#) ()  
*Remove some useless acceptance condition.*
- void [compute\\_scc](#) ()  
*Compute the maximal SCC of the automata.*
- virtual std::string [format\\_state](#) (const [spot::state](#) \*state) const  
*Add the SCC index to the display of the [state](#) state.*
- void [display\\_rel\\_sim](#) (simulation\_relation \*rel, std::ostream &os)
- void [display\\_scc](#) (std::ostream &os)
- [state](#) \* [set\\_init\\_state](#) (const std::string &state)
- transition \* [create\\_transition](#) (const std::string &source, const std::string &dest)
- transition \* [create\\_transition](#) (state \*source, const [state](#) \*dest)
- void [add\\_condition](#) (transition \*t, const [ltl::formula](#) \*f)
- void [add\\_conditions](#) (transition \*t, bdd f)  
*This assumes that all variables in f are known from dict.*
- void [declare\\_acceptance\\_condition](#) (const [ltl::formula](#) \*f)
- void [copy\\_acceptance\\_conditions\\_of](#) (const [tgba](#) \*a)  
*Copy the acceptance conditions of a [tgba](#).*
- bool [has\\_acceptance\\_condition](#) (const [ltl::formula](#) \*f) const
- void [add\\_acceptance\\_condition](#) (transition \*t, const [ltl::formula](#) \*f)
- void [add\\_acceptance\\_conditions](#) (transition \*t, bdd f)  
*This assumes that all acceptance conditions in f are known from dict.*
- void [complement\\_all\\_acceptance\\_conditions](#) ()
- void [merge\\_transitions](#) ()
- [state](#) \* [add\\_state](#) (const std::string &name)
- virtual [spot::state](#) \* [get\\_init\\_state](#) () const  
*Get the initial [state](#) of the automaton.*



- virtual `tgba_succ_iterator` \* `succ_iter` (const `spot::state` \*`local_state`, const `spot::state` \*`global_state`=0, const `tgba` \*`global_automaton`=0) const  
*Get an iterator over the successors of `local_state`.*
- virtual `bdd_dict` \* `get_dict` () const  
*Get the dictionary associated to the automaton.*
- virtual `bdd` `all_acceptance_conditions` () const  
*Return the set of all acceptance conditions used by this automaton.*
- virtual `bdd` `neg_acceptance_conditions` () const  
*Return the conjunction of all negated acceptance variables.*
- `bdd` `support_conditions` (const `state` \*`state`) const  
*Get a formula that must hold whatever successor is taken.*
- `bdd` `support_variables` (const `state` \*`state`) const  
*Get the conjunctions of variables tested by the outgoing transitions of `state`.*
- virtual `std::string` `transition_annotation` (const `tgba_succ_iterator` \*`t`) const  
*Return a possible annotation for the transition pointed to by the iterator.*
- virtual `state` \* `project_state` (const `state` \*`s`, const `tgba` \*`t`) const  
*Project a `state` on an automaton.*
- virtual unsigned int `number_of_acceptance_conditions` () const  
*The number of acceptance conditions.*
- virtual void `add_state` (const `state` \*`s`)
- virtual const `state` \* `next_state` ()  
*Called by `run()` to obtain the.*
- void `run` ()  
*Iterate over all reachable states of a `spot::tgba`.*
- virtual void `process_link` (const `state` \*`in_s`, int `in`, const `state` \*`out_s`, int `out`, const `tgba_succ_iterator` \*`si`)

## Protected Types

- typedef `Sgi::hash_map`< const `tgba_explicit::state` \*, `std::list`< `state` \* > \*, `ptr_hash`< `tgba_explicit::state` > > `sp_map`
- typedef `Sgi::hash_map`< const `spot::state` \*, int, `state_ptr_hash`, `state_ptr_equal` > `seen_map`
- typedef `Sgi::hash_map`< const `std::string`, `tgba_explicit::state` \*, `string_hash` > `ns_map`
- typedef `Sgi::hash_map`< const `tgba_explicit::state` \*, `std::string`, `ptr_hash`< `tgba_explicit::state` > > `sn_map`

## Protected Member Functions

- void [start](#) ()  
*Called by [run\(\)](#) before starting its iteration.*
- void [end](#) ()  
*Called by [run\(\)](#) once all states have been explored.*
- void [process\\_state](#) (const [spot::state](#) \*s, int n, [tgba\\_succ\\_iterator](#) \*si)
- void [process\\_link](#) (int in, int out, const [tgba\\_succ\\_iterator](#) \*si)
- transition \* [create\\_transition](#) (const [spot::state](#) \*source, const [spot::state](#) \*dest)  
*Create a transition using two [state](#) of a TGBA.*
- void [redirect\\_transition](#) (const [spot::state](#) \*s, const [spot::state](#) \*simul)
- void [remove\\_predecessor\\_state](#) (const [state](#) \*s, const [state](#) \*p)  
*Remove p of the predecessor of s.*
- void [remove\\_state](#) (const [spot::state](#) \*s)
- void [merge\\_state](#) (const [spot::state](#) \*s1, const [spot::state](#) \*s2)
- void [merge\\_state\\_delayed](#) (const [spot::state](#) \*s1, const [spot::state](#) \*s2)
- void [delete\\_scc](#) ()
- bool [is\\_terminal](#) (const [spot::state](#) \*s, int n=-1)
- bool [is\\_not\\_accepting](#) (const [spot::state](#) \*s, int n=-1)
- void [remove\\_acc](#) (const [spot::state](#) \*s)
- void [remove\\_scc](#) ([spot::state](#) \*s)  
*Remove all the [state](#) which belong to the same scc that s.*
- void [remove\\_component](#) (const [spot::state](#) \*from)  
*For [compute\\_scc](#).*
- int [nb\\_set\\_acc\\_cond](#) () const
- virtual bdd [compute\\_support\\_conditions](#) (const [spot::state](#) \*state) const  
*Do the actual computation of [tgba::support\\_conditions\(\)](#).*
- virtual bdd [compute\\_support\\_variables](#) (const [spot::state](#) \*state) const  
*Do the actual computation of [tgba::support\\_variables\(\)](#).*
- bdd [get\\_acceptance\\_condition](#) (const [ltl::formula](#) \*f)

## Protected Attributes

- bool [scc\\_computed\\_](#)
- [scc\\_stack](#) root\_
- [numbered\\_state\\_heap](#) \* h\_
- std::stack< const [spot::state](#) \* > [state\\_scc\\_](#)
- Sgi::hash\_map< int, const [spot::state](#) \* > [state\\_scc\\_v\\_](#)
- [sp\\_map](#) [state\\_predecessor\\_map\\_](#)
- [seen\\_map](#) si\_
- [seen\\_map](#) \* seen\_
- bdd [acc\\_](#)

- [ns\\_map](#) [name\\_state\\_map\\_](#)
- [sn\\_map](#) [state\\_name\\_map\\_](#)
- [bdd\\_dict](#) \* [dict\\_](#)
- [tgba\\_explicit::state](#) \* [init\\_](#)
- [bdd](#) [all\\_acceptance\\_conditions\\_](#)
- [bdd](#) [neg\\_acceptance\\_conditions\\_](#)
- [bool](#) [all\\_acceptance\\_conditions\\_computed\\_](#)
- [std::deque](#)< [const state](#) \* > [todo](#)  
*A queue of states yet to explore.*
- [const tgba](#) \* [automata\\_](#)  
*The [spot::tgba](#) to explore.*
- [seen\\_map](#) [seen](#)  
*States already seen.*

### 12.111.1 Detailed Description

Explicit automata used in reductions.

### 12.111.2 Member Typedef Documentation

**12.111.2.1** `typedef Sgi::hash_map<const tgba_explicit::state*, std::list<state*>*, ptr_hash<tgba_explicit::state> > spot::tgba_reduc::sp_map` [protected]

**12.111.2.2** `typedef Sgi::hash_map<const spot::state*, int, state_ptr_hash, state_ptr_equal> spot::tgba_reduc::seen_map` [protected]

Reimplemented from [spot::tgba\\_reachable\\_iterator](#).

**12.111.2.3** `typedef std::list<transition*> spot::tgba_explicit::state` [inherited]

**12.111.2.4** `typedef Sgi::hash_map<const std::string, tgba_explicit::state*, string_hash> spot::tgba_explicit::ns_map` [protected, inherited]

**12.111.2.5** `typedef Sgi::hash_map<const tgba_explicit::state*, std::string, ptr_hash<tgba_explicit::state> > spot::tgba_explicit::sn_map` [protected, inherited]

### 12.111.3 Constructor & Destructor Documentation

**12.111.3.1** `spot::tgba_reduc::tgba_reduc (const tgba * a, const numbered_state_heap_factory * nshf = numbered_state_heap_hash_map_factory::instance())`

**12.111.3.2** `spot::tgba_reduc::~~tgba_reduc ()`

#### 12.111.4 Member Function Documentation

##### 12.111.4.1 void spot::tgba\_reduc::quotient\_state (direct\_simulation\_relation \* *rel*)

Reduce the automata using a relation simulation Do not call this method with a delayed simulation relation.

##### 12.111.4.2 void spot::tgba\_reduc::quotient\_state (delayed\_simulation\_relation \* *rel*)

Build the quotient automata. Call this method when use to a delayed simulation relation.

##### 12.111.4.3 void spot::tgba\_reduc::delete\_transitions (simulation\_relation \* *rel*)

Delete some transitions with help of a simulation relation.

##### 12.111.4.4 void spot::tgba\_reduc::prune\_scc ()

Remove all [state](#) which not lead to an accepting cycle.

##### 12.111.4.5 void spot::tgba\_reduc::prune\_acc ()

Remove some useless acceptance condition.

##### 12.111.4.6 void spot::tgba\_reduc::compute\_scc ()

Compute the maximal SCC of the automata.

##### 12.111.4.7 virtual std::string spot::tgba\_reduc::format\_state (const spot::state \* *state*) const [virtual]

Add the SCC index to the display of the [state state](#).

Reimplemented from [spot::tgba\\_explicit](#).

##### 12.111.4.8 void spot::tgba\_reduc::display\_rel\_sim (simulation\_relation \* *rel*, std::ostream & *os*)

##### 12.111.4.9 void spot::tgba\_reduc::display\_scc (std::ostream & *os*)

##### 12.111.4.10 void spot::tgba\_reduc::start () [protected, virtual]

Called by [run\(\)](#) before starting its iteration.

Reimplemented from [spot::tgba\\_reachable\\_iterator](#).

##### 12.111.4.11 void spot::tgba\_reduc::end () [protected, virtual]

Called by [run\(\)](#) once all states have been explored.

Reimplemented from [spot::tgba\\_reachable\\_iterator](#).

**12.111.4.12** `void spot::tgba_reduc::process_state (const spot::state * s, int n, tgba_succ_iterator * si)` [protected, virtual]

Called by [run\(\)](#) to process a [state](#).

**Parameters:**

- s* The current [state](#).
- n* A unique number assigned to *s*.
- si* The [spot::tgba\\_succ\\_iterator](#) for *s*.

Reimplemented from [spot::tgba\\_reachable\\_iterator](#).

**12.111.4.13** `void spot::tgba_reduc::process_link (int in, int out, const tgba_succ_iterator * si)` [protected]

**12.111.4.14** `transition* spot::tgba_reduc::create_transition (const spot::state * source, const spot::state * dest)` [protected]

Create a transition using two [state](#) of a TGBA.

**12.111.4.15** `void spot::tgba_reduc::redirect_transition (const spot::state * s, const spot::state * simul)` [protected]

Remove all the transition from the [state](#) *q*, predecessor of both *s* and *simul*, which can be removed.

**12.111.4.16** `void spot::tgba_reduc::remove_predecessor_state (const state * s, const state * p)` [protected]

Remove *p* of the predecessor of *s*.

**12.111.4.17** `void spot::tgba_reduc::remove_state (const spot::state * s)` [protected]

Remove all the transition leading to *s*. *s* is then unreachable and can be consider as remove.

**12.111.4.18** `void spot::tgba_reduc::merge_state (const spot::state * s1, const spot::state * s2)` [protected]

Redirect all transition leading to *s1* to *s2*. Note that we can do the reverse because *s1* and *s2* belong to a co-simulate relation.

**12.111.4.19** `void spot::tgba_reduc::merge_state_delayed (const spot::state * s1, const spot::state * s2)` [protected]

Redirect all transition leading to *s1* to *s2*. Note that we can do the reverse because *s1* and *s2* belong to a co-simulate relation.

**12.111.4.20** `void spot::tgba_reduc::delete_scc ()` [protected]

Remove all the scc which are terminal and doesn't contains all the acceptance conditions.

**12.111.4.21 bool spot::tgba\_reduc::is\_terminal (const spot::state \* *s*, int *n* = -1)** [protected]

Return true if the scc which contains *s* is an fixed-formula alpha-ball. this is explain in

```

/// @InProceedings{ etessami.00.concur,
/// author = {Kousha Etessami and Gerard J. Holzmann},
/// title = {Optimizing {B}\u}chi Automata},
/// booktitle = {Proceedings of the 11th International Conference on
/// Concurrency Theory (Concur'2000)},
/// pages = {153--167},
/// year = {2000},
/// editor = {C. Palamidessi},
/// volume = {1877},
/// series = {Lecture Notes in Computer Science},
/// publisher = {Springer-Verlag}
/// }
///

```

**12.111.4.22 bool spot::tgba\_reduc::is\_not\_accepting (const spot::state \* *s*, int *n* = -1)** [protected]**12.111.4.23 void spot::tgba\_reduc::remove\_acc (const spot::state \* *s*)** [protected]

If a scc maximal do not contains all the acceptance conditions we can remove all the acceptance conditions in this scc.

**12.111.4.24 void spot::tgba\_reduc::remove\_scc (spot::state \* *s*)** [protected]

Remove all the [state](#) which belong to the same scc that *s*.

**12.111.4.25 void spot::tgba\_reduc::remove\_component (const spot::state \* *from*)** [protected]

For compute\_scc.

**12.111.4.26 int spot::tgba\_reduc::nb\_set\_acc\_cond () const** [protected]**12.111.4.27 state\* spot::tgba\_explicit::set\_init\_state (const std::string & *state*)** [inherited]**12.111.4.28 transition\* spot::tgba\_explicit::create\_transition (const std::string & *source*, const std::string & *dest*)** [inherited]**12.111.4.29 transition\* spot::tgba\_explicit::create\_transition (state \* *source*, const state \* *dest*)** [inherited]**12.111.4.30 void spot::tgba\_explicit::add\_condition (transition \* *t*, const ltl::formula \* *f*)** [inherited]**12.111.4.31 void spot::tgba\_explicit::add\_conditions (transition \* *t*, bdd *f*)** [inherited]

This assumes that all variables in *f* are known from dict.

**12.111.4.32** void spot::tgba\_explicit::declare\_acceptance\_condition (const ltl::formula \* *f*) [inherited]

**12.111.4.33** void spot::tgba\_explicit::copy\_acceptance\_conditions\_of (const tgba \* *a*) [inherited]

Copy the acceptance conditions of a [tgba](#).

If used, this function should be called before creating any [transition](#).

**12.111.4.34** bool spot::tgba\_explicit::has\_acceptance\_condition (const ltl::formula \* *f*) const [inherited]

**12.111.4.35** void spot::tgba\_explicit::add\_acceptance\_condition (transition \* *t*, const ltl::formula \* *f*) [inherited]

**12.111.4.36** void spot::tgba\_explicit::add\_acceptance\_conditions (transition \* *t*, bdd *f*) [inherited]

This assumes that all acceptance conditions in *f* are known from dict.

**12.111.4.37** void spot::tgba\_explicit::complement\_all\_acceptance\_conditions () [inherited]

**12.111.4.38** void spot::tgba\_explicit::merge\_transitions () [inherited]

**12.111.4.39** state\* spot::tgba\_explicit::add\_state (const std::string & *name*) [inherited]

Return the [tgba\\_explicit::state](#) for *name*, creating the [state](#) if it does not exist.

**12.111.4.40** virtual spot::state\* spot::tgba\_explicit::get\_init\_state () const [virtual, inherited]

Get the initial [state](#) of the automaton.

The [state](#) has been allocated with `new`. It is the responsibility of the caller to `delete` it when no longer needed.

Implements [spot::tgba](#).

**12.111.4.41** virtual tgba\_succ\_iterator\* spot::tgba\_explicit::succ\_iter (const spot::state \* *local\_state*, const spot::state \* *global\_state* = 0, const tgba \* *global\_automaton* = 0) const [virtual, inherited]

Get an iterator over the successors of *local\_state*.

The iterator has been allocated with `new`. It is the responsibility of the caller to `delete` it when no longer needed.

During synchornized products, additional informations are passed about the entire product and its [state](#). Recall that products can be nested, forming a tree of [spot::tgba](#) where most values are computed on demand. *global\_automaton* designate the root [spot::tgba](#), and *global\_state* its [state](#). This two objects can be used by [succ\\_iter\(\)](#) to restrict the set of successors to compute.

**Parameters:**

*local\_state* The [state](#) whose successors are to be explored. This pointer is not adopted in any way by `succ_iter`, and it is still the caller's responsibility to delete it when appropriate (this can be done during the lifetime of the iterator).

*global\_state* In a product, the [state](#) of the global product automaton. Otherwise, 0. Like *local\_state*, *global\_state* is not adopted by `succ_iter`.

*global\_automaton* In a product, the global product automaton. Otherwise, 0.

Implements [spot::tgba](#).

**12.111.4.42 virtual bdd\_dict\* spot::tgba\_explicit::get\_dict () const** [virtual, inherited]

Get the dictionary associated to the automaton.

State are represented as BDDs. The dictionary allows to map BDD variables back to formulae, and vice versa. This is useful when dealing with several automata (which may use the same BDD variable for different formula), or simply when printing.

Implements [spot::tgba](#).

**12.111.4.43 virtual bdd spot::tgba\_explicit::all\_acceptance\_conditions () const** [virtual, inherited]

Return the set of all acceptance conditions used by this automaton.

The goal of the emptiness check is to ensure that a strongly connected component walks through each of these acceptance conditions. I.e., the union of the acceptance conditions of all [transition](#) in the SCC should be equal to the result of this function.

Implements [spot::tgba](#).

**12.111.4.44 virtual bdd spot::tgba\_explicit::neg\_acceptance\_conditions () const** [virtual, inherited]

Return the conjunction of all negated acceptance variables.

For instance if the automaton uses variables `Acc[a]`, `Acc[b]` and `Acc[c]` to describe acceptance sets, this function should return `!Acc[a] & !Acc[b] & !Acc[c]`.

This is useful when making products: each operand's condition set should be augmented with the [neg\\_acceptance\\_conditions\(\)](#) of the other operand.

Implements [spot::tgba](#).

**12.111.4.45 virtual bdd spot::tgba\_explicit::compute\_support\_conditions (const spot::state \* state) const** [protected, virtual, inherited]

Do the actual computation of [tgba::support\\_conditions\(\)](#).

Implements [spot::tgba](#).

**12.111.4.46 virtual bdd spot::tgba\_explicit::compute\_support\_variables (const spot::state \* state) const** [protected, virtual, inherited]

Do the actual computation of [tgba::support\\_variables\(\)](#).



Implements [spot::tgba](#).

**12.111.4.47 bdd spot::tgba\_explicit::get\_acceptance\_condition (const ltl::formula \* f)**  
[protected, inherited]

**12.111.4.48 bdd spot::tgba::support\_conditions (const state \* state) const** [inherited]

Get a formula that must hold whatever successor is taken.

**Returns:**

A formula which must be verified for all successors of [state](#).

This can be as simple as `bddtrue`, or more completely the disjunction of the condition of all successors. This is used as an hint by `succ_iter()` to reduce the number of successor to compute in a product.

Sub classes should implement `compute_support_conditions()`, this function is just a wrapper that will cache the last return value for efficiency.

**12.111.4.49 bdd spot::tgba::support\_variables (const state \* state) const** [inherited]

Get the conjunctions of variables tested by the outgoing transitions of [state](#).

All variables tested by outgoing transitions must be returned. This is mandatory.

This is used as an hint by some `succ_iter()` to reduce the number of successor to compute in a product.

Sub classes should implement `compute_support_variables()`, this function is just a wrapper that will cache the last return value for efficiency.

**12.111.4.50 virtual std::string spot::tgba::transition\_annotation (const tgba\_succ\_iterator \* t) const** [virtual, inherited]

Return a possible annotation for the transition pointed to by the iterator.

Implementing this function is optional; the default annotation is the empty string.

**Parameters:**

*t* a non-done [tgba\\_succ\\_iterator](#) for this automata

Reimplemented in [spot::tgba\\_product](#), and [spot::tgba\\_tba\\_proxy](#).

**12.111.4.51 virtual state\* spot::tgba::project\_state (const state \* s, const tgba \* t) const**  
[virtual, inherited]

Project a [state](#) on an automaton.

This converts *s*, into that corresponding [spot::state](#) for *t*. This is useful when you have the [state](#) of a product, and want restrict this [state](#) to a specific automata occurring in the product.

It goes without saying that *s* and *t* should be compatible (i.e., *s* is a [state](#) of *t*).

**Returns:**

0 if the projection fails (*s* is unrelated to *t*), or a new `state*` (the projected [state](#)) that must be deleted by the caller.

Reimplemented in [spot::tgba\\_product](#), and [spot::tgba\\_tba\\_proxy](#).

**12.111.4.52** `virtual unsigned int spot::tgba::number_of_acceptance_conditions () const`  
[virtual, inherited]

The number of acceptance conditions.

**12.111.4.53** `virtual void spot::tgba_reachable_iterator_breadth_first::add_state (const state * s)`  
[virtual, inherited]

Implements [spot::tgba\\_reachable\\_iterator](#).

**12.111.4.54** `virtual const state* spot::tgba_reachable_iterator_breadth_first::next_state ()`  
[virtual, inherited]

Called by [run\(\)](#) to obtain the.

Implements [spot::tgba\\_reachable\\_iterator](#).

**12.111.4.55** `void spot::tgba_reachable_iterator::run ()` [inherited]

Iterate over all reachable states of a [spot::tgba](#).

This is a template method that will call [add\\_state\(\)](#), [next\\_state\(\)](#), [start\(\)](#), [end\(\)](#), [process\\_state\(\)](#), and [process\\_link\(\)](#), while it iterate over [state](#).

**12.111.4.56** `virtual void spot::tgba_reachable_iterator::process_link (const state * in_s, int in, const state * out_s, int out, const tgba_succ_iterator * si)` [virtual, inherited]

Called by [run\(\)](#) to process a transition.

#### Parameters:

*in\_s* The source [state](#)

*in* The source [state](#) number.

*out\_s* The destination [state](#)

*out* The destination [state](#) number.

*si* The [spot::tgba\\_succ\\_iterator](#) positionned on the current transition.

The *in\_s* and *out\_s* states are owned by the [spot::tgba\\_reachable\\_iterator](#) instance and destroyed when the instance is destroyed.

### 12.111.5 Member Data Documentation

**12.111.5.1** `bool spot::tgba_reduc::scc_computed_` [protected]

**12.111.5.2** `scc_stack spot::tgba_reduc::root_` [protected]

**12.111.5.3** `numbered_state_heap* spot::tgba_reduc::h_` [protected]

**12.111.5.4** `std::stack<const spot::state*> spot::tgba_reduc::state_scc_` [protected]

**12.111.5.5** Sgi::hash\_map<int, const spot::state\*> spot::tgba\_reduc::state\_scc\_v\_ [protected]

**12.111.5.6** sp\_map spot::tgba\_reduc::state\_predecessor\_map\_ [protected]

**12.111.5.7** seen\_map spot::tgba\_reduc::si\_ [protected]

**12.111.5.8** seen\_map\* spot::tgba\_reduc::seen\_ [protected]

**12.111.5.9** bdd spot::tgba\_reduc::acc\_ [protected]

**12.111.5.10** ns\_map spot::tgba\_explicit::name\_state\_map\_ [protected, inherited]

**12.111.5.11** sn\_map spot::tgba\_explicit::state\_name\_map\_ [protected, inherited]

**12.111.5.12** bdd\_dict\* spot::tgba\_explicit::dict\_ [protected, inherited]

**12.111.5.13** tgba\_explicit::state\* spot::tgba\_explicit::init\_ [protected, inherited]

**12.111.5.14** bdd spot::tgba\_explicit::all\_acceptance\_conditions\_ [mutable, protected, inherited]

**12.111.5.15** bdd spot::tgba\_explicit::neg\_acceptance\_conditions\_ [protected, inherited]

**12.111.5.16** bool spot::tgba\_explicit::all\_acceptance\_conditions\_computed\_ [mutable, protected, inherited]

**12.111.5.17** std::deque<const state\*> spot::tgba\_reachable\_iterator\_breadth\_first::todo [protected, inherited]

A queue of states yet to explore.

**12.111.5.18** const tgba\* spot::tgba\_reachable\_iterator::automata\_ [protected, inherited]

The [spot::tgba](#) to explore.

**12.111.5.19** seen\_map spot::tgba\_reachable\_iterator::seen [protected, inherited]

States already seen.

The documentation for this class was generated from the following file:

- [tgba/tgbareduc.hh](#)

## 12.112 spot::tgba\_run Struct Reference

An accepted run, for a [tgba](#).

```
#include <tgbaalgos/emptiness.hh>
```

### Public Types

- typedef std::list< [step](#) > [steps](#)

### Public Member Functions

- [~tgba\\_run](#) ()
- [tgba\\_run](#) ()
- [tgba\\_run](#) (const [tgba\\_run](#) &run)
- [tgba\\_run](#) & [operator=](#) (const [tgba\\_run](#) &run)

### Public Attributes

- [steps](#) prefix
- [steps](#) cycle

### Classes

- struct [step](#)

#### 12.112.1 Detailed Description

An accepted run, for a [tgba](#).

#### 12.112.2 Member Typedef Documentation

##### 12.112.2.1 typedef std::list<step> spot::tgba\_run::steps

#### 12.112.3 Constructor & Destructor Documentation

##### 12.112.3.1 spot::tgba\_run::~~tgba\_run ()

##### 12.112.3.2 spot::tgba\_run::tgba\_run () [inline]

##### 12.112.3.3 spot::tgba\_run::tgba\_run (const tgba\_run & run)

#### 12.112.4 Member Function Documentation

##### 12.112.4.1 tgba\_run& spot::tgba\_run::operator= (const tgba\_run & run)

### 12.112.5 Member Data Documentation

#### 12.112.5.1 steps spot::tgba\_run::prefix

#### 12.112.5.2 steps spot::tgba\_run::cycle

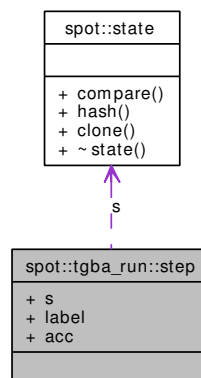
The documentation for this struct was generated from the following file:

- [tgbalgorithms/emptiness.hh](#)

## 12.113 spot::tgba\_run::step Struct Reference

```
#include <tgbalgorithms/emptiness.hh>
```

Collaboration diagram for spot::tgba\_run::step:



### Public Attributes

- const [state](#) \* [s](#)
- bdd [label](#)
- bdd [acc](#)

### 12.113.1 Member Data Documentation

#### 12.113.1.1 const state\* spot::tgba\_run::step::s

#### 12.113.1.2 bdd spot::tgba\_run::step::label

#### 12.113.1.3 bdd spot::tgba\_run::step::acc

The documentation for this struct was generated from the following file:

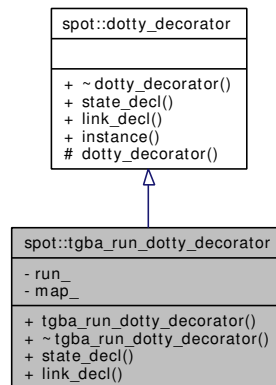
- [tgbalgorithms/emptiness.hh](#)

## 12.114 spot::tgba\_run\_dotty\_decorator Class Reference

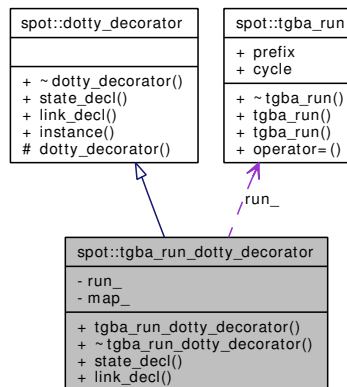
Highlight a [spot::tgba\\_run](#) on a [spot::tgba](#).

```
#include <tgbaalgos/rundotdec.hh>
```

Inheritance diagram for `spot::tgba_run_dotty_decorator`:



Collaboration diagram for `spot::tgba_run_dotty_decorator`:



### Public Member Functions

- `tgba_run_dotty_decorator` (const [tgba\\_run](#) \*run)
- virtual `~tgba_run_dotty_decorator` ()
- virtual std::string `state_decl` (const [tgba](#) \*a, const [state](#) \*s, int n, [tgba\\_succ\\_iterator](#) \*si, const std::string &label)

*Compute the style of a [state](#).*

- virtual std::string `link_decl` (const [tgba](#) \*a, const [state](#) \*in\_s, int in, const [state](#) \*out\_s, int out, const [tgba\\_succ\\_iterator](#) \*si, const std::string &label)

*Compute the style of a [link](#).*

## Static Public Member Functions

- static [dotty\\_decorator](#) \* [instance](#) ()  
*Get the unique instance of the default [dotty\\_decorator](#).*

## Private Types

- typedef std::pair< tgba\_run::steps::const\_iterator, int > [step\\_num](#)
- typedef std::list< [step\\_num](#) > [step\\_set](#)
- typedef std::map< const [state](#) \*, std::pair< [step\\_set](#), [step\\_set](#) >, spot::state\_ptr\_less\_than > [step\\_map](#)

## Private Attributes

- const [tgba\\_run](#) \* [run\\_](#)
- [step\\_map](#) [map\\_](#)

### 12.114.1 Detailed Description

Highlight a [spot::tgba\\_run](#) on a [spot::tgba](#).

An instance of this class can be passed to [spot::dotty\\_reachable](#).

### 12.114.2 Member Typedef Documentation

**12.114.2.1** typedef std::pair<tgba\_run::steps::const\_iterator, int> spot::tgba\_run\_dotty\_decorator::step\_num [private]

**12.114.2.2** typedef std::list<step\_num> spot::tgba\_run\_dotty\_decorator::step\_set [private]

**12.114.2.3** typedef std::map<const state\*, std::pair<step\_set, step\_set>, spot::state\_ptr\_less\_than> spot::tgba\_run\_dotty\_decorator::step\_map [private]

### 12.114.3 Constructor & Destructor Documentation

**12.114.3.1** spot::tgba\_run\_dotty\_decorator::tgba\_run\_dotty\_decorator (const tgba\_run \* [run](#))

**12.114.3.2** virtual spot::tgba\_run\_dotty\_decorator::~~tgba\_run\_dotty\_decorator () [virtual]

### 12.114.4 Member Function Documentation

**12.114.4.1** virtual std::string spot::tgba\_run\_dotty\_decorator::state\_decl (const tgba \* [a](#), const state \* [s](#), int [n](#), tgba\_succ\_iterator \* [si](#), const std::string & [label](#)) [virtual]

Compute the style of a [state](#).

This function should output a string of the form `[label="foo", style=bar, ...]`. The default implementation will simply output `[label="LABEL"]` with `LABEL` replaced by the value of *label*.

**Parameters:**

- a* the automaton being drawn
- s* the [state](#) being drawn (owned by the caller)
- n* a unique number for this [state](#)
- si* an iterator over the successors of this [state](#) (owned by the caller, but can be freely iterated)
- label* the computed name of this [state](#)

Reimplemented from [spot::dotty\\_decorator](#).

**12.114.4.2** `virtual std::string spot::tgba_run_dotty_decorator::link_decl (const tgba * a, const state * in_s, int in, const state * out_s, int out, const tgba_succ_iterator * si, const std::string & label)` `[virtual]`

Compute the style of a link.

This function should output a string of the form `[label="foo", style=bar, ...]`. The default implementation will simply output `[label="LABEL"]` with `LABEL` replaced by the value of *label*.

**Parameters:**

- a* the automaton being drawn
- in\_s* the source [state](#) of the transition being drawn (owned by the caller)
- in* the unique number associated to *in\_s*
- out\_s* the destination [state](#) of the transition being drawn (owned by the caller)
- out* the unique number associated to *out\_s*
- si* an iterator over the successors of *in\_s*, pointing to the current transition (owned by the caller and cannot be iterated)
- label* the computed name of this [state](#)

Reimplemented from [spot::dotty\\_decorator](#).

**12.114.4.3** `static dotty_decorator* spot::dotty_decorator::instance ()` `[static, inherited]`

Get the unique instance of the default [dotty\\_decorator](#).

## 12.114.5 Member Data Documentation

**12.114.5.1** `const tgba_run* spot::tgba_run_dotty_decorator::run_` `[private]`

**12.114.5.2** `step_map spot::tgba_run_dotty_decorator::map_` `[private]`

The documentation for this class was generated from the following file:

- [tgbaalgos/rundotdec.hh](#)

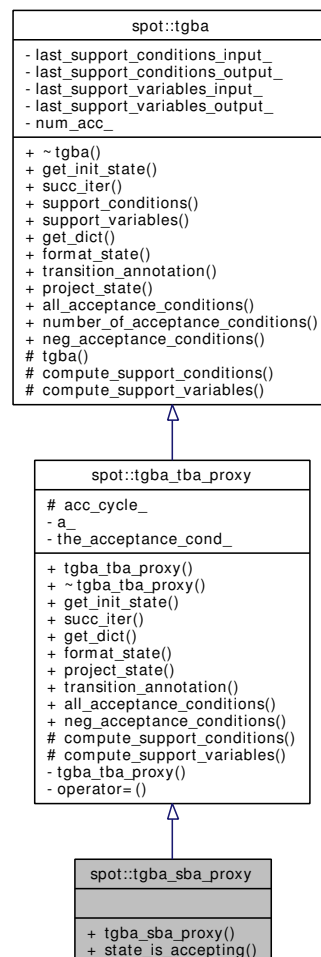


## 12.115 spot::tgba\_sba\_proxy Class Reference

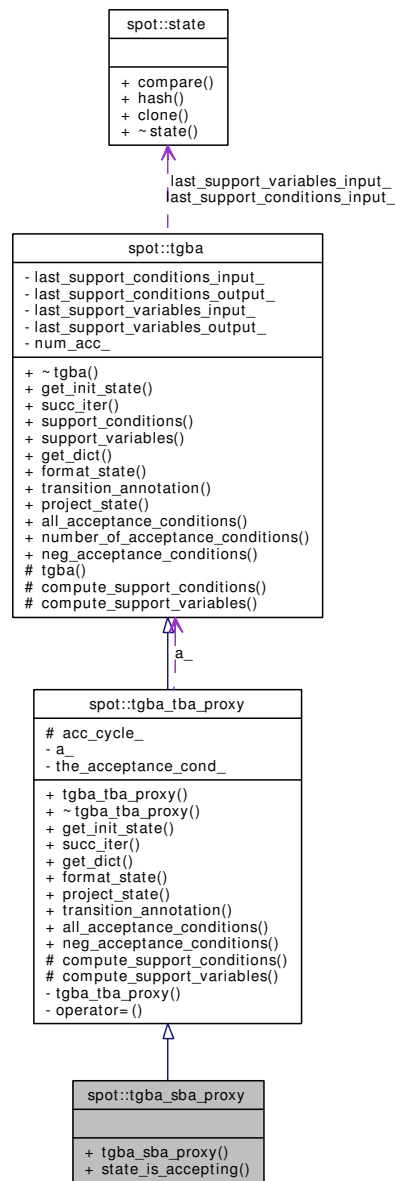
Degeneralize a [spot::tgba](#) on the fly, producing an SBA.

```
#include <tgba/tgbatba.hh>
```

Inheritance diagram for spot::tgba\_sba\_proxy:



Collaboration diagram for spot::tgba\_sba\_proxy:



## Public Types

- typedef std::list< bdd > [cycle\\_list](#)

## Public Member Functions

- [tgba\\_sba\\_proxy](#) (const [tgba](#) \*a)
- bool [state\\_is\\_accepting](#) (const [state](#) \*state) const  
Whether the [state](#) is accepting.
- virtual [state](#) \* [get\\_init\\_state](#) () const

Get the initial *state* of the automaton.

- virtual `tgba_succ_iterator * succ_iter` (const `state *local_state`, const `state *global_state=0`, const `tgba *global_automaton=0`) const

Get an iterator over the successors of `local_state`.

- virtual `bdd_dict * get_dict` () const

Get the dictionary associated to the automaton.

- virtual `std::string format_state` (const `state *state`) const

Format the *state* as a string for printing.

- virtual `state * project_state` (const `state *s`, const `tgba *t`) const

Project a *state* on an automaton.

- virtual `std::string transition_annotation` (const `tgba_succ_iterator *t`) const

Return a possible annotation for the transition pointed to by the iterator.

- virtual `bdd all_acceptance_conditions` () const

Return the set of all acceptance conditions used by this automaton.

- virtual `bdd neg_acceptance_conditions` () const

Return the conjunction of all negated acceptance variables.

- `bdd support_conditions` (const `state *state`) const

Get a formula that must hold whatever successor is taken.

- `bdd support_variables` (const `state *state`) const

Get the conjunctions of variables tested by the outgoing transitions of *state*.

- virtual `unsigned int number_of_acceptance_conditions` () const

The number of acceptance conditions.

### Protected Member Functions

- virtual `bdd compute_support_conditions` (const `state *state`) const

Do the actual computation of `tgba::support_conditions()`.

- virtual `bdd compute_support_variables` (const `state *state`) const

Do the actual computation of `tgba::support_variables()`.

### Protected Attributes

- `cycle_list acc_cycle_`

### 12.115.1 Detailed Description

Degeneralize a `spot::tgba` on the fly, producing an SBA.

This class acts as a proxy in front of a `spot::tgba`, that should be degeneralized on the fly.

This is similar to `tgba_tba_proxy`, except that automata produced with this algorithms can also be seen as State-based Büchi Automata (SBA). See `tgba_sba_proxy::state_is_accepting()`. (An SBA is a TBA, and a TBA is a TGBA.)

This extra property has a small cost in size: if the input automaton uses  $N$  acceptance conditions, the output automaton can have at most  $\max(N,1)+1$  times more states and transitions. (This is only  $\max(N,1)$  for `tgba_tba_proxy`.)

### 12.115.2 Member Typedef Documentation

**12.115.2.1** `typedef std::list<bdd> spot::tgba_tba_proxy::cycle_list` [inherited]

### 12.115.3 Constructor & Destructor Documentation

**12.115.3.1** `spot::tgba_sba_proxy::tgba_sba_proxy (const tgba * a)`

### 12.115.4 Member Function Documentation

**12.115.4.1** `bool spot::tgba_sba_proxy::state_is_accepting (const state * state) const`

Whether the `state` is accepting.

A particularity of a `spot::tgba_sba_proxy` automaton is that when a `state` has an outgoing accepting arc, all its outgoing arcs are accepting. The `state` itself can therefore be considered accepting. This is useful in algorithms working on degeneralized automata with `state` acceptance conditions.

**12.115.4.2** `virtual state* spot::tgba_tba_proxy::get_init_state () const` [virtual, inherited]

Get the initial `state` of the automaton.

The `state` has been allocated with `new`. It is the responsibility of the caller to `delete` it when no longer needed.

Implements `spot::tgba`.

**12.115.4.3** `virtual tgba_succ_iterator* spot::tgba_tba_proxy::succ_iter (const state * local_state, const state * global_state = 0, const tgba * global_automaton = 0) const` [virtual, inherited]

Get an iterator over the successors of `local_state`.

The iterator has been allocated with `new`. It is the responsibility of the caller to `delete` it when no longer needed.

During synchronized products, additional informations are passed about the entire product and its `state`. Recall that products can be nested, forming a tree of `spot::tgba` where most values are computed on demand. `global_automaton` designate the root `spot::tgba`, and `global_state` its `state`. This two objects can be used by `succ_iter()` to restrict the set of successors to compute.

**Parameters:**

*local\_state* The [state](#) whose successors are to be explored. This pointer is not adopted in any way by `succ_iter`, and it is still the caller's responsibility to delete it when appropriate (this can be done during the lifetime of the iterator).

*global\_state* In a product, the [state](#) of the global product automaton. Otherwise, 0. Like *local\_state*, *global\_state* is not adopted by `succ_iter`.

*global\_automaton* In a product, the global product automaton. Otherwise, 0.

Implements [spot::tgba](#).

**12.115.4.4 virtual bdd\_dict\* spot::tgba\_tba\_proxy::get\_dict () const** [virtual, inherited]

Get the dictionary associated to the automaton.

State are represented as BDDs. The dictionary allows to map BDD variables back to formulae, and vice versa. This is useful when dealing with several automata (which may use the same BDD variable for different formula), or simply when printing.

Implements [spot::tgba](#).

**12.115.4.5 virtual std::string spot::tgba\_tba\_proxy::format\_state (const state \* state) const** [virtual, inherited]

Format the [state](#) as a string for printing.

This formatting is the responsibility of the automata who owns the [state](#).

Implements [spot::tgba](#).

**12.115.4.6 virtual state\* spot::tgba\_tba\_proxy::project\_state (const state \* s, const tgba \* t) const** [virtual, inherited]

Project a [state](#) on an automaton.

This converts *s*, into that corresponding [spot::state](#) for *t*. This is useful when you have the [state](#) of a product, and want restrict this [state](#) to a specific automata occurring in the product.

It goes without saying that *s* and *t* should be compatible (i.e., *s* is a [state](#) of *t*).

**Returns:**

0 if the projection fails (*s* is unrelated to *t*), or a new `state*` (the projected [state](#)) that must be deleted by the caller.

Reimplemented from [spot::tgba](#).

**12.115.4.7 virtual std::string spot::tgba\_tba\_proxy::transition\_annotation (const tgba\_succ\_iterator \* t) const** [virtual, inherited]

Return a possible annotation for the transition pointed to by the iterator.

Implementing this function is optional; the default annotation is the empty string.

**Parameters:**

*t* a non-done [tgba\\_succ\\_iterator](#) for this automata

Reimplemented from [spot::tgba](#).

**12.115.4.8** `virtual bdd spot::tgba_tba_proxy::all_acceptance_conditions () const` [virtual, inherited]

Return the set of all acceptance conditions used by this automaton.

The goal of the emptiness check is to ensure that a strongly connected component walks through each of these acceptance conditions. I.e., the union of the acceptance conditions of all transition in the SCC should be equal to the result of this function.

Implements [spot::tgba](#).

**12.115.4.9** `virtual bdd spot::tgba_tba_proxy::neg_acceptance_conditions () const` [virtual, inherited]

Return the conjunction of all negated acceptance variables.

For instance if the automaton uses variables `Acc[a]`, `Acc[b]` and `Acc[c]` to describe acceptance sets, this function should return `!Acc[a] & !Acc[b] & !Acc[c]`.

This is useful when making products: each operand's condition set should be augmented with the [neg\\_acceptance\\_conditions\(\)](#) of the other operand.

Implements [spot::tgba](#).

**12.115.4.10** `virtual bdd spot::tgba_tba_proxy::compute_support_conditions (const state * state) const` [protected, virtual, inherited]

Do the actual computation of [tgba::support\\_conditions\(\)](#).

Implements [spot::tgba](#).

**12.115.4.11** `virtual bdd spot::tgba_tba_proxy::compute_support_variables (const state * state) const` [protected, virtual, inherited]

Do the actual computation of [tgba::support\\_variables\(\)](#).

Implements [spot::tgba](#).

**12.115.4.12** `bdd spot::tgba::support_conditions (const state * state) const` [inherited]

Get a formula that must hold whatever successor is taken.

#### Returns:

A formula which must be verified for all successors of [state](#).

This can be as simple as `bddtrue`, or more completely the disjunction of the condition of all successors. This is used as an hint by [succ\\_iter\(\)](#) to reduce the number of successor to compute in a product.

Sub classes should implement [compute\\_support\\_conditions\(\)](#), this function is just a wrapper that will cache the last return value for efficiency.

**12.115.4.13** `bdd spot::tgba::support_variables (const state * state) const` [inherited]

Get the conjunctions of variables tested by the outgoing transitions of [state](#).

All variables tested by outgoing transitions must be returned. This is mandatory.

This is used as an hint by some `succ_iter()` to reduce the number of successor to compute in a product.

Sub classes should implement `compute_support_variables()`, this function is just a wrapper that will cache the last return value for efficiency.

**12.115.4.14** `virtual unsigned int spot::tgba::number_of_acceptance_conditions () const`  
[virtual, inherited]

The number of acceptance conditions.

### 12.115.5 Member Data Documentation

**12.115.5.1** `cycle_list spot::tgba_tba_proxy::acc_cycle_` [protected, inherited]

The documentation for this class was generated from the following file:

- [tgba/tgbatba.hh](#)

## 12.116 `spot::tgba_statistics` Struct Reference

```
#include <tgbaalgos/stats.hh>
```

### Public Attributes

- unsigned [transitions](#)
- unsigned [states](#)

### 12.116.1 Member Data Documentation

**12.116.1.1** `unsigned spot::tgba_statistics::transitions`

**12.116.1.2** `unsigned spot::tgba_statistics::states`

The documentation for this struct was generated from the following file:

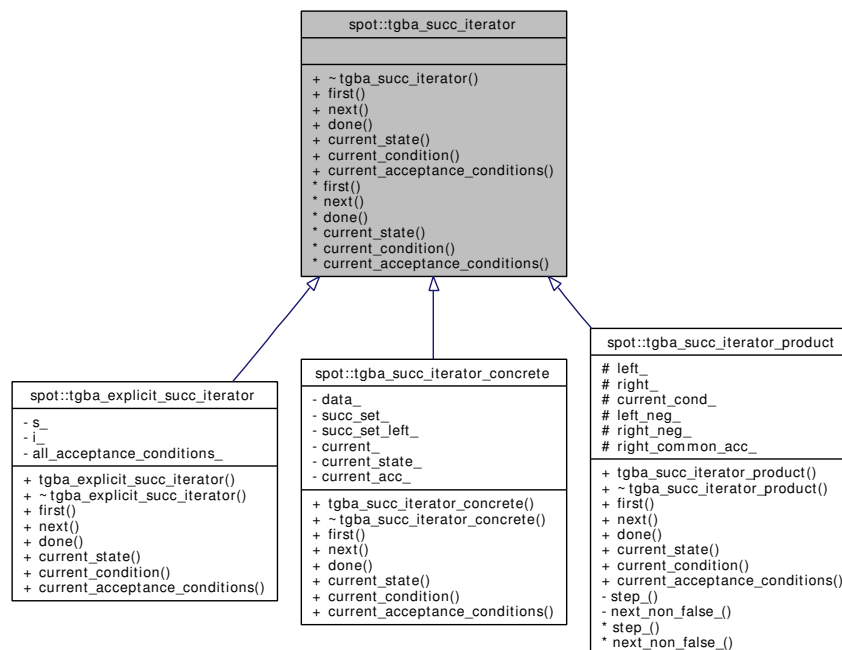
- [tgbaalgos/stats.hh](#)

## 12.117 `spot::tgba_succ_iterator` Class Reference

Iterate over the successors of a [state](#).

```
#include <tgba/succiter.hh>
```

Inheritance diagram for spot::tgba\_succ\_iterator:



## Public Member Functions

- virtual `~tgba_succ_iterator()`

## Iteration

- virtual void `first()`=0  
*Position the iterator on the first successor (if any).*
- virtual void `next()`=0  
*Jump to the next successor (if any).*
- virtual bool `done()` const=0  
*Check whether the iteration is finished.*

## Inspection

- virtual `state * current_state()` const=0  
*Get the `state` of the current successor.*
- virtual bdd `current_condition()` const=0  
*Get the condition on the transition leading to this successor.*
- virtual bdd `current_acceptance_conditions()` const=0  
*Get the acceptance conditions on the transition leading to this successor.*



### 12.117.1 Detailed Description

Iterate over the successors of a [state](#).

This class provides the basic functionalities required to iterate over the successors of a [state](#), as well as querying transition labels. Because transitions are never explicitly encoded, labels (conditions and acceptance conditions) can only be queried while iterating over the successors.

### 12.117.2 Constructor & Destructor Documentation

**12.117.2.1** `virtual spot::tgba_succ_iterator::~~tgba_succ_iterator () [inline, virtual]`

### 12.117.3 Member Function Documentation

**12.117.3.1** `virtual void spot::tgba_succ_iterator::first () [pure virtual]`

Position the iterator on the first successor (if any).

This method can be called several times to make multiple passes over successors.

#### Warning:

One should always call `done ()` to ensure there is a successor, even after `first ()`. A common trap is to assume that there is at least one successor: this is wrong.

Implemented in `spot::tgba_succ_iterator_concrete`, `spot::tgba_explicit_succ_iterator`, and `spot::tgba_succ_iterator_product`.

**12.117.3.2** `virtual void spot::tgba_succ_iterator::next () [pure virtual]`

Jump to the next successor (if any).

#### Warning:

Again, one should always call `done ()` to ensure there is a successor.

Implemented in `spot::tgba_succ_iterator_concrete`, `spot::tgba_explicit_succ_iterator`, and `spot::tgba_succ_iterator_product`.

**12.117.3.3** `virtual bool spot::tgba_succ_iterator::done () const [pure virtual]`

Check whether the iteration is finished.

This function should be called after any call to `first ()` or `next ()` and before any enquiry about the current [state](#).

The usual way to do this is with a `for` loop.

```
for (s->first(); !s->done(); s->next())
    ...
```

Implemented in `spot::tgba_succ_iterator_concrete`, `spot::tgba_explicit_succ_iterator`, and `spot::tgba_succ_iterator_product`.

**12.117.3.4** `virtual state* spot::tgba_succ_iterator::current_state () const` [pure virtual]

Get the [state](#) of the current successor.

Note that the same [state](#) may occur at different points in the iteration. These actually correspond to the same destination. It just means there were several transitions, with different conditions, leading to the same [state](#).

Implemented in [spot::tgba\\_succ\\_iterator\\_concrete](#), [spot::tgba\\_explicit\\_succ\\_iterator](#), and [spot::tgba\\_succ\\_iterator\\_product](#).

**12.117.3.5** `virtual bdd spot::tgba_succ_iterator::current_condition () const` [pure virtual]

Get the condition on the transition leading to this successor.

This is a boolean function of atomic propositions.

Implemented in [spot::tgba\\_succ\\_iterator\\_concrete](#), [spot::tgba\\_explicit\\_succ\\_iterator](#), and [spot::tgba\\_succ\\_iterator\\_product](#).

**12.117.3.6** `virtual bdd spot::tgba_succ_iterator::current_acceptance_conditions () const` [pure virtual]

Get the acceptance conditions on the transition leading to this successor.

Implemented in [spot::tgba\\_succ\\_iterator\\_concrete](#), [spot::tgba\\_explicit\\_succ\\_iterator](#), and [spot::tgba\\_succ\\_iterator\\_product](#).

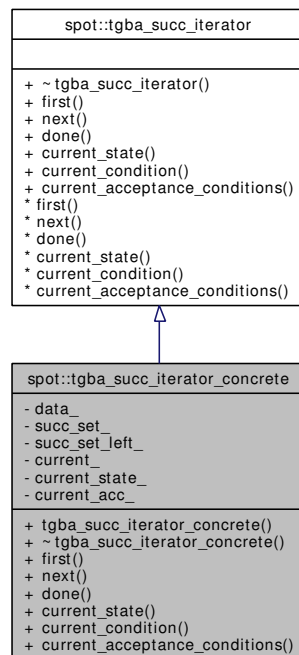
The documentation for this class was generated from the following file:

- [tgba/succiter.hh](#)

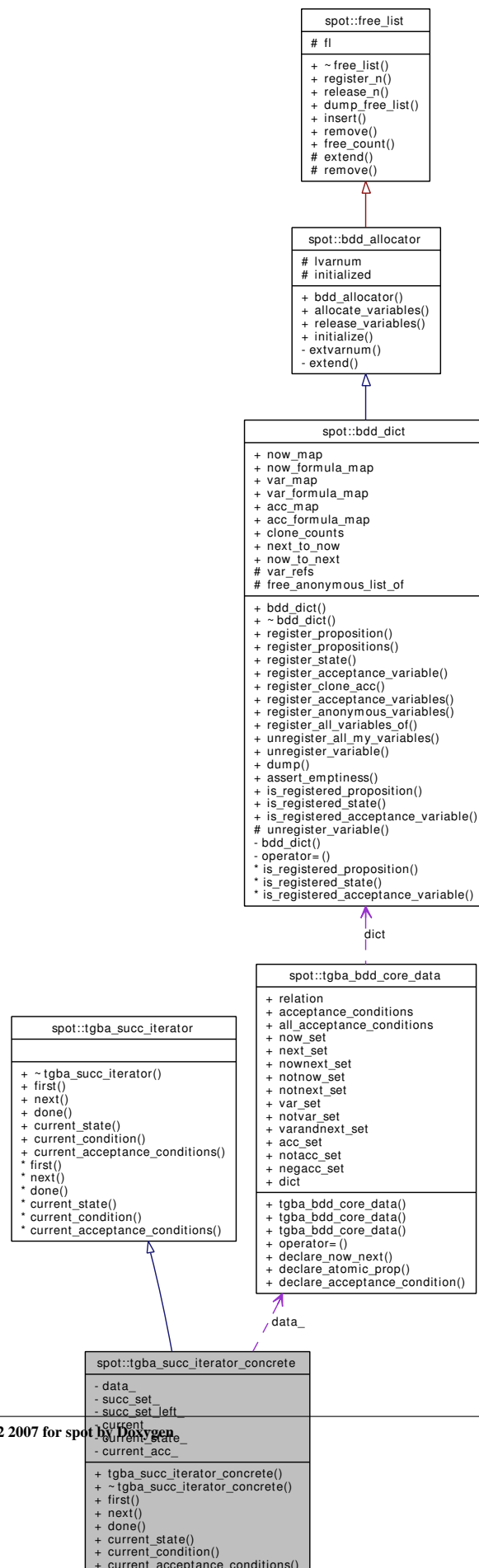
**12.118** `spot::tgba_succ_iterator_concrete` Class Reference

```
#include <tgba/succiterconcrete.hh>
```

Inheritance diagram for spot::tgba\_succ\_iterator\_concrete:



Collaboration diagram for spot::tgba\_succ\_iterator\_concrete:



## Public Member Functions

- [tgba\\_succ\\_iterator\\_concrete](#) (const [tgba\\_bdd\\_core\\_data](#) &d, bdd successors)  
*Build a [spot::tgba\\_succ\\_iterator\\_concrete](#).*
- virtual [~tgba\\_succ\\_iterator\\_concrete](#) ()
- void [first](#) ()  
*Position the iterator on the first successor (if any).*
- void [next](#) ()  
*Jump to the next successor (if any).*
- bool [done](#) () const  
*Check whether the iteration is finished.*
- [state\\_bdd](#) \* [current\\_state](#) () const  
*Get the [state](#) of the current successor.*
- bdd [current\\_condition](#) () const  
*Get the condition on the transition leading to this successor.*
- bdd [current\\_acceptance\\_conditions](#) () const  
*Get the acceptance conditions on the transition leading to this successor.*

## Private Attributes

- const [tgba\\_bdd\\_core\\_data](#) & [data\\_](#)  
*Core data of the automaton.*
- bdd [succ\\_set\\_](#)  
*The set of successors.*
- bdd [succ\\_set\\_left\\_](#)  
*Unexplored successors (including [current\\_](#)).*
- bdd [current\\_](#)  
*Current successor, as a conjunction of atomic proposition and Next variables.*
- bdd [current\\_state\\_](#)  
*Current successor, as a conjunction of Now variables.*
- bdd [current\\_acc\\_](#)  
*Acceptance conditions for the current transition.*

### 12.118.1 Detailed Description

A concrete iterator over successors of a TGBA [state](#).

## 12.118.2 Constructor & Destructor Documentation

### 12.118.2.1 spot::tgba\_succ\_iterator\_concrete::tgba\_succ\_iterator\_concrete (const tgba\_bdd\_core\_data & d, bdd successors)

Build a [spot::tgba\\_succ\\_iterator\\_concrete](#).

#### Parameters:

*successors* The set of successors with ingoing conditions and acceptance conditions, represented as a BDD. The job of this iterator will be to enumerate the satisfactions of that BDD and split them into destination states and conditions, and compute acceptance conditions.

*d* The core data of the automata. These contains sets of variables useful to split a BDD, and compute acceptance conditions.

### 12.118.2.2 virtual spot::tgba\_succ\_iterator\_concrete::~tgba\_succ\_iterator\_concrete () [virtual]

## 12.118.3 Member Function Documentation

### 12.118.3.1 void spot::tgba\_succ\_iterator\_concrete::first () [virtual]

Position the iterator on the first successor (if any).

This method can be called several times to make multiple passes over successors.

#### Warning:

One should always call [done \(\)](#) to ensure there is a successor, even after [first \(\)](#). A common trap is to assume that there is at least one successor: this is wrong.

Implements [spot::tgba\\_succ\\_iterator](#).

### 12.118.3.2 void spot::tgba\_succ\_iterator\_concrete::next () [virtual]

Jump to the next successor (if any).

#### Warning:

Again, one should always call [done \(\)](#) to ensure there is a successor.

Implements [spot::tgba\\_succ\\_iterator](#).

### 12.118.3.3 bool spot::tgba\_succ\_iterator\_concrete::done () const [virtual]

Check whether the iteration is finished.

This function should be called after any call to [first \(\)](#) or [next \(\)](#) and before any enquiry about the current [state](#).

The usual way to do this is with a `for` loop.

```
for (s->first(); !s->done(); s->next())
    ...
```

Implements [spot::tgba\\_succ\\_iterator](#).

**12.118.3.4 state\_bdd\* spot::tgba\_succ\_iterator\_concrete::current\_state () const** [virtual]

Get the [state](#) of the current successor.

Note that the same [state](#) may occur at different points in the iteration. These actually correspond to the same destination. It just means there were several transitions, with different conditions, leading to the same [state](#).

Implements [spot::tgba\\_succ\\_iterator](#).

**12.118.3.5 bdd spot::tgba\_succ\_iterator\_concrete::current\_condition () const** [virtual]

Get the condition on the transition leading to this successor.

This is a boolean function of atomic propositions.

Implements [spot::tgba\\_succ\\_iterator](#).

**12.118.3.6 bdd spot::tgba\_succ\_iterator\_concrete::current\_acceptance\_conditions () const** [virtual]

Get the acceptance conditions on the transition leading to this successor.

Implements [spot::tgba\\_succ\\_iterator](#).

**12.118.4 Member Data Documentation****12.118.4.1 const tgba\_bdd\_core\_data& spot::tgba\_succ\_iterator\_concrete::data\_** [private]

Core data of the automaton.

**12.118.4.2 bdd spot::tgba\_succ\_iterator\_concrete::succ\_set\_** [private]

The set of successors.

**12.118.4.3 bdd spot::tgba\_succ\_iterator\_concrete::succ\_set\_left\_** [private]

Unexplored successors (including current\_).

**12.118.4.4 bdd spot::tgba\_succ\_iterator\_concrete::current\_** [private]

Current successor, as a conjunction of atomic proposition and Next variables.

**12.118.4.5 bdd spot::tgba\_succ\_iterator\_concrete::current\_state\_** [private]

Current successor, as a conjunction of Now variables.

**12.118.4.6 bdd spot::tgba\_succ\_iterator\_concrete::current\_acc\_** [private]

Acceptance conditions for the current transition.

The documentation for this class was generated from the following file:

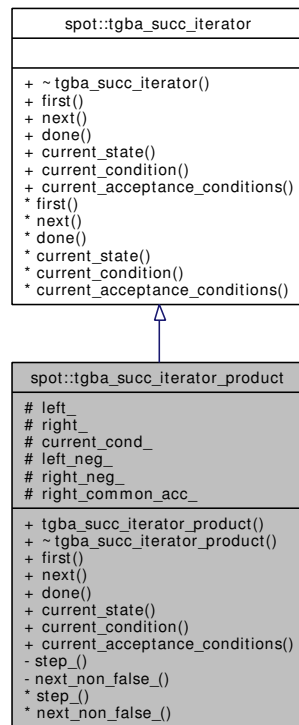
- [tgba/succiterconcrete.hh](#)

## 12.119 spot::tgba\_succ\_iterator\_product Class Reference

Iterate over the successors of a product computed on the fly.

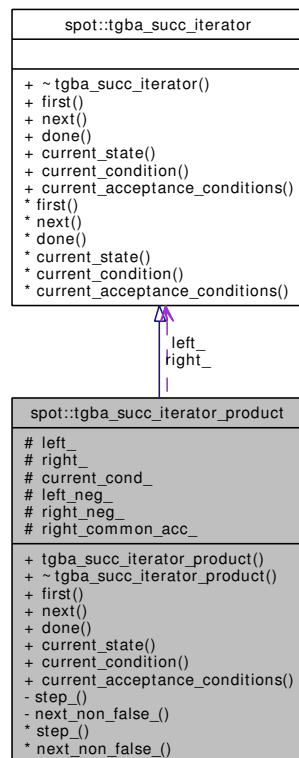
```
#include <tgba/tgbaproduct.hh>
```

Inheritance diagram for spot::tgba\_succ\_iterator\_product:





Collaboration diagram for spot::tgba\_succ\_iterator\_product:



## Public Member Functions

- [tgba\\_succ\\_iterator\\_product](#) ([tgba\\_succ\\_iterator](#) \*left, [tgba\\_succ\\_iterator](#) \*right, bdd left\_neg, bdd right\_neg, bddPair \*right\_common\_acc)
- virtual [~tgba\\_succ\\_iterator\\_product](#) ()
- void [first](#) ()  
*Position the iterator on the first successor (if any).*
- void [next](#) ()  
*Jump to the next successor (if any).*
- bool [done](#) () const  
*Check whether the iteration is finished.*
- [state\\_product](#) \* [current\\_state](#) () const  
*Get the [state](#) of the current successor.*
- bdd [current\\_condition](#) () const  
*Get the condition on the transition leading to this successor.*
- bdd [current\\_acceptance\\_conditions](#) () const  
*Get the acceptance conditions on the transition leading to this successor.*

### Protected Attributes

- [tgba\\_succ\\_iterator](#) \* [left\\_](#)
- [tgba\\_succ\\_iterator](#) \* [right\\_](#)
- bdd [current\\_cond\\_](#)
- bdd [left\\_neg\\_](#)
- bdd [right\\_neg\\_](#)
- bddPair \* [right\\_common\\_acc\\_](#)

### Private Member Functions

- void [step\\_](#) ()  
*Internal routines to advance to the next successor.*
- void [next\\_non\\_false\\_](#) ()

### Friends

- class [tgba\\_product](#)

#### 12.119.1 Detailed Description

Iterate over the successors of a product computed on the fly.

#### 12.119.2 Constructor & Destructor Documentation

**12.119.2.1** `spot::tgba_succ_iterator_product::tgba_succ_iterator_product (tgba_succ_iterator * left, tgba_succ_iterator * right, bdd left_neg, bdd right_neg, bddPair * right_common_acc)`

**12.119.2.2** `virtual spot::tgba_succ_iterator_product::~~tgba_succ_iterator_product ()`  
[virtual]

#### 12.119.3 Member Function Documentation

**12.119.3.1** `void spot::tgba_succ_iterator_product::first ()` [virtual]

Position the iterator on the first successor (if any).

This method can be called several times to make multiple passes over successors.

#### Warning:

One should always call [done \(\)](#) to ensure there is a successor, even after [first \(\)](#). A common trap is to assume that there is at least one successor: this is wrong.

Implements [spot::tgba\\_succ\\_iterator](#).

**12.119.3.2 void spot::tgba\_succ\_iterator\_product::next () [virtual]**

Jump to the next successor (if any).

**Warning:**

Again, one should always call [done \(\)](#) to ensure there is a successor.

Implements [spot::tgba\\_succ\\_iterator](#).

**12.119.3.3 bool spot::tgba\_succ\_iterator\_product::done () const [virtual]**

Check whether the iteration is finished.

This function should be called after any call to [first \(\)](#) or [next \(\)](#) and before any enquiry about the current [state](#).

The usual way to do this is with a `for` loop.

```
for (s->first(); !s->done(); s->next())
    ...
```

Implements [spot::tgba\\_succ\\_iterator](#).

**12.119.3.4 state\_product\* spot::tgba\_succ\_iterator\_product::current\_state () const [virtual]**

Get the [state](#) of the current successor.

Note that the same [state](#) may occur at different points in the iteration. These actually correspond to the same destination. It just means there were several transitions, with different conditions, leading to the same [state](#).

Implements [spot::tgba\\_succ\\_iterator](#).

**12.119.3.5 bdd spot::tgba\_succ\_iterator\_product::current\_condition () const [virtual]**

Get the condition on the transition leading to this successor.

This is a boolean function of atomic propositions.

Implements [spot::tgba\\_succ\\_iterator](#).

**12.119.3.6 bdd spot::tgba\_succ\_iterator\_product::current\_acceptance\_conditions () const [virtual]**

Get the acceptance conditions on the transition leading to this successor.

Implements [spot::tgba\\_succ\\_iterator](#).

**12.119.3.7 void spot::tgba\_succ\_iterator\_product::step\_ () [private]**

Internal routines to advance to the next successor.

**12.119.3.8 void spot::tgba\_succ\_iterator\_product::next\_non\_false\_ () [private]**

#### 12.119.4 Friends And Related Function Documentation

12.119.4.1 friend class `tgba_product` [`friend`]

#### 12.119.5 Member Data Documentation

12.119.5.1 `tgba_succ_iterator*` `spot::tgba_succ_iterator_product::left_` [`protected`]

12.119.5.2 `tgba_succ_iterator*` `spot::tgba_succ_iterator_product::right_` [`protected`]

12.119.5.3 `bdd` `spot::tgba_succ_iterator_product::current_cond_` [`protected`]

12.119.5.4 `bdd` `spot::tgba_succ_iterator_product::left_neg_` [`protected`]

12.119.5.5 `bdd` `spot::tgba_succ_iterator_product::right_neg_` [`protected`]

12.119.5.6 `bddPair*` `spot::tgba_succ_iterator_product::right_common_acc_` [`protected`]

The documentation for this class was generated from the following file:

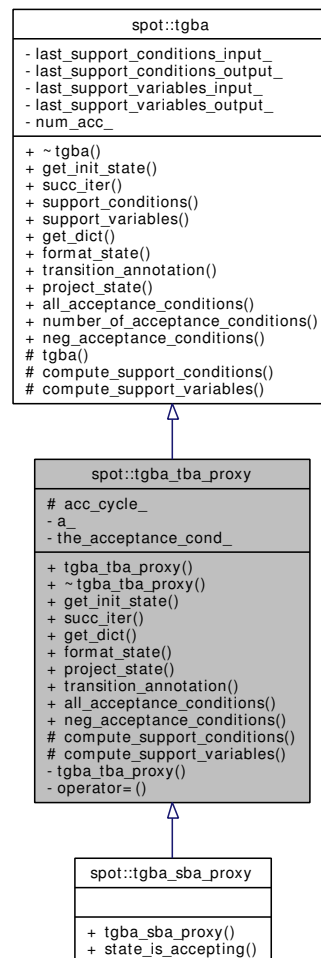
- [tgba/tgbaproduct.hh](#)

### 12.120 `spot::tgba_tba_proxy` Class Reference

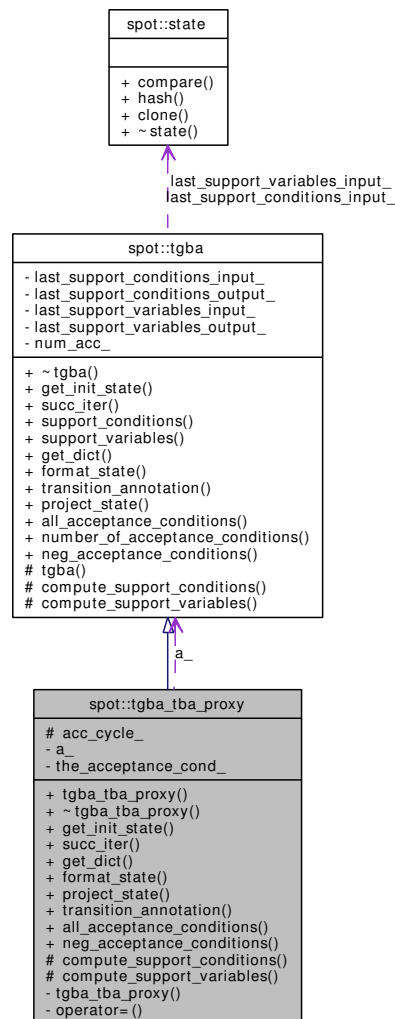
Degeneralize a [spot::tgba](#) on the fly, producing a TBA.

```
#include <tgba/tgbatba.hh>
```

Inheritance diagram for spot::tgba\_tba\_proxy:



Collaboration diagram for spot::tgba\_tba\_proxy:



## Public Types

- typedef std::list< bdd > [cycle\\_list](#)

## Public Member Functions

- [tgba\\_tba\\_proxy](#) (const [tgba](#) \*a)
- virtual [~tgba\\_tba\\_proxy](#) ()
- virtual [state](#) \* [get\\_init\\_state](#) () const  
Get the initial [state](#) of the automaton.
- virtual [tgba\\_succ\\_iterator](#) \* [succ\\_iter](#) (const [state](#) \*local\_state, const [state](#) \*global\_state=0, const [tgba](#) \*global\_automaton=0) const  
Get an iterator over the successors of local\_state.
- virtual [bdd\\_dict](#) \* [get\\_dict](#) () const

*Get the dictionary associated to the automaton.*

- virtual std::string [format\\_state](#) (const [state](#) \*state) const  
*Format the [state](#) as a string for printing.*
- virtual [state](#) \* [project\\_state](#) (const [state](#) \*s, const [tgba](#) \*t) const  
*Project a [state](#) on an automaton.*
- virtual std::string [transition\\_annotation](#) (const [tgba\\_succ\\_iterator](#) \*t) const  
*Return a possible annotation for the transition pointed to by the iterator.*
- virtual bdd [all\\_acceptance\\_conditions](#) () const  
*Return the set of all acceptance conditions used by this automaton.*
- virtual bdd [neg\\_acceptance\\_conditions](#) () const  
*Return the conjunction of all negated acceptance variables.*
- bdd [support\\_conditions](#) (const [state](#) \*state) const  
*Get a formula that must hold whatever successor is taken.*
- bdd [support\\_variables](#) (const [state](#) \*state) const  
*Get the conjunctions of variables tested by the outgoing transitions of [state](#).*
- virtual unsigned int [number\\_of\\_acceptance\\_conditions](#) () const  
*The number of acceptance conditions.*

### Protected Member Functions

- virtual bdd [compute\\_support\\_conditions](#) (const [state](#) \*state) const  
*Do the actual computation of [tgba::support\\_conditions\(\)](#).*
- virtual bdd [compute\\_support\\_variables](#) (const [state](#) \*state) const  
*Do the actual computation of [tgba::support\\_variables\(\)](#).*

### Protected Attributes

- [cycle\\_list](#) [acc\\_cycle\\_](#)

### Private Member Functions

- [tgba\\_tba\\_proxy](#) (const [tgba\\_tba\\_proxy](#) &)
- [tgba\\_tba\\_proxy](#) & operator= (const [tgba\\_tba\\_proxy](#) &)

### Private Attributes

- const [tgba](#) \* [a\\_](#)
- bdd [the\\_acceptance\\_cond\\_](#)

### 12.120.1 Detailed Description

Degeneralize a `spot::tgba` on the fly, producing a TBA.

This class acts as a proxy in front of a `spot::tgba`, that should be degeneralized on the fly. The result is still a `spot::tgba`, but it will always have exactly one acceptance condition so it could be called TBA (without the G).

The degeneralization is done by synchronizing the input automaton with a "counter" automaton such as the one shown in "On-the-fly Verification of Linear Temporal Logic" (Jean-Michel Couvreur, FME99).

If the input automaton uses  $N$  acceptance conditions, the output automaton can have at most  $\max(N,1)$  times more states and transitions.

See also:

[tgba\\_sba\\_proxy](#)

### 12.120.2 Member Typedef Documentation

**12.120.2.1** `typedef std::list<bdd> spot::tgba_tba_proxy::cycle_list`

### 12.120.3 Constructor & Destructor Documentation

**12.120.3.1** `spot::tgba_tba_proxy::tgba_tba_proxy (const tgba * a)`

**12.120.3.2** `virtual spot::tgba_tba_proxy::~~tgba_tba_proxy ()` `[virtual]`

**12.120.3.3** `spot::tgba_tba_proxy::tgba_tba_proxy (const tgba_tba_proxy &)` `[private]`

### 12.120.4 Member Function Documentation

**12.120.4.1** `virtual state* spot::tgba_tba_proxy::get_init_state () const` `[virtual]`

Get the initial [state](#) of the automaton.

The [state](#) has been allocated with `new`. It is the responsibility of the caller to `delete` it when no longer needed.

Implements [spot::tgba](#).

**12.120.4.2** `virtual tgba_succ_iterator* spot::tgba_tba_proxy::succ_iter (const state * local_state, const state * global_state = 0, const tgba * global_automaton = 0) const` `[virtual]`

Get an iterator over the successors of *local\_state*.

The iterator has been allocated with `new`. It is the responsibility of the caller to `delete` it when no longer needed.

During synchronizing products, additional informations are passed about the entire product and its [state](#). Recall that products can be nested, forming a tree of `spot::tgba` where most values are computed on demand. *global\_automaton* designate the root `spot::tgba`, and *global\_state* its [state](#). This two objects can be used by [succ\\_iter\(\)](#) to restrict the set of successors to compute.



**Parameters:**

*local\_state* The [state](#) whose successors are to be explored. This pointer is not adopted in any way by `succ_iter`, and it is still the caller's responsibility to delete it when appropriate (this can be done during the lifetime of the iterator).

*global\_state* In a product, the [state](#) of the global product automaton. Otherwise, 0. Like *local\_state*, *global\_state* is not adopted by `succ_iter`.

*global\_automaton* In a product, the global product automaton. Otherwise, 0.

Implements [spot::tgba](#).

**12.120.4.3 virtual bdd\_dict\* spot::tgba\_tba\_proxy::get\_dict () const [virtual]**

Get the dictionary associated to the automaton.

State are represented as BDDs. The dictionary allows to map BDD variables back to formulae, and vice versa. This is useful when dealing with several automata (which may use the same BDD variable for different formula), or simply when printing.

Implements [spot::tgba](#).

**12.120.4.4 virtual std::string spot::tgba\_tba\_proxy::format\_state (const state \* state) const [virtual]**

Format the [state](#) as a string for printing.

This formatting is the responsibility of the automata who owns the [state](#).

Implements [spot::tgba](#).

**12.120.4.5 virtual state\* spot::tgba\_tba\_proxy::project\_state (const state \* s, const tgba \* t) const [virtual]**

Project a [state](#) on an automaton.

This converts *s*, into that corresponding [spot::state](#) for *t*. This is useful when you have the [state](#) of a product, and want restrict this [state](#) to a specific automata occurring in the product.

It goes without saying that *s* and *t* should be compatible (i.e., *s* is a [state](#) of *t*).

**Returns:**

0 if the projection fails (*s* is unrelated to *t*), or a new `state*` (the projected [state](#)) that must be deleted by the caller.

Reimplemented from [spot::tgba](#).

**12.120.4.6 virtual std::string spot::tgba\_tba\_proxy::transition\_annotation (const tgba\_succ\_iterator \* t) const [virtual]**

Return a possible annotation for the transition pointed to by the iterator.

Implementing this function is optional; the default annotation is the empty string.

**Parameters:**

*t* a non-done [tgba\\_succ\\_iterator](#) for this automata

Reimplemented from [spot::tgba](#).

**12.120.4.7 virtual bdd spot::tgba\_tba\_proxy::all\_acceptance\_conditions () const** [virtual]

Return the set of all acceptance conditions used by this automaton.

The goal of the emptiness check is to ensure that a strongly connected component walks through each of these accepting conditions. I.e., the union of the accepting conditions of all transition in the SCC should be equal to the result of this function.

Implements [spot::tgba](#).

**12.120.4.8 virtual bdd spot::tgba\_tba\_proxy::neg\_acceptance\_conditions () const** [virtual]

Return the conjunction of all negated acceptance variables.

For instance if the automaton uses variables `Acc[a]`, `Acc[b]` and `Acc[c]` to describe acceptance sets, this function should return `!Acc[a] & !Acc[b] & !Acc[c]`.

This is useful when making products: each operand's condition set should be augmented with the [neg\\_acceptance\\_conditions\(\)](#) of the other operand.

Implements [spot::tgba](#).

**12.120.4.9 virtual bdd spot::tgba\_tba\_proxy::compute\_support\_conditions (const state \* state) const** [protected, virtual]

Do the actual computation of [tgba::support\\_conditions\(\)](#).

Implements [spot::tgba](#).

**12.120.4.10 virtual bdd spot::tgba\_tba\_proxy::compute\_support\_variables (const state \* state) const** [protected, virtual]

Do the actual computation of [tgba::support\\_variables\(\)](#).

Implements [spot::tgba](#).

**12.120.4.11 tgba\_tba\_proxy& spot::tgba\_tba\_proxy::operator= (const tgba\_tba\_proxy &)** [private]**12.120.4.12 bdd spot::tgba::support\_conditions (const state \* state) const** [inherited]

Get a formula that must hold whatever successor is taken.

**Returns:**

A formula which must be verified for all successors of [state](#).

This can be as simple as `bddtrue`, or more completely the disjunction of the condition of all successors. This is used as an hint by [succ\\_iter\(\)](#) to reduce the number of successor to compute in a product.

Sub classes should implement [compute\\_support\\_conditions\(\)](#), this function is just a wrapper that will cache the last return value for efficiency.

**12.120.4.13 bdd spot::tgba::support\_variables (const state \* state) const** [inherited]

Get the conjunctions of variables tested by the outgoing transitions of [state](#).

All variables tested by outgoing transitions must be returned. This is mandatory.

This is used as an hint by some `succ_iter()` to reduce the number of successor to compute in a product. Sub classes should implement `compute_support_variables()`, this function is just a wrapper that will cache the last return value for efficiency.

**12.120.4.14** `virtual unsigned int spot::tgba::number_of_acceptance_conditions () const` `[virtual, inherited]`

The number of acceptance conditions.

## 12.120.5 Member Data Documentation

**12.120.5.1** `cycle_list spot::tgba_tba_proxy::acc_cycle_` `[protected]`

**12.120.5.2** `const tgba* spot::tgba_tba_proxy::a_` `[private]`

**12.120.5.3** `bdd spot::tgba_tba_proxy::the_acceptance_cond_` `[private]`

The documentation for this class was generated from the following file:

- [tgba/tgbatba.hh](#)

## 12.121 `spot::time_info` Struct Reference

A structure to record elapsed time in clock ticks.

```
#include <misc/timer.hh>
```

### Public Member Functions

- [time\\_info\(\)](#)

### Public Attributes

- `clock_t` [utime](#)
- `clock_t` [stime](#)

### 12.121.1 Detailed Description

A structure to record elapsed time in clock ticks.

### 12.121.2 Constructor & Destructor Documentation

**12.121.2.1** `spot::time_info::time_info ()` `[inline]`

### 12.121.3 Member Data Documentation

**12.121.3.1** `clock_t spot::time_info::utime`

### 12.121.3.2 clock\_t spot::time\_info::stime

The documentation for this struct was generated from the following file:

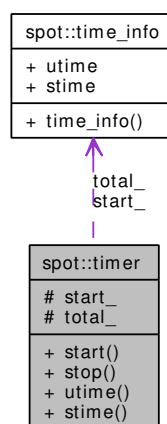
- [misc/timer.hh](#)

## 12.122 spot::timer Class Reference

A timekeeper that accumulate interval of time.

```
#include <misc/timer.hh>
```

Collaboration diagram for spot::timer:



### Public Member Functions

- void [start](#) ()  
*Start a time interval.*
- void [stop](#) ()  
*Stop a time interval and update the sum of all intervals.*
- clock\_t [utime](#) () const  
*Return the user time of all accumulated interval.*
- clock\_t [stime](#) () const  
*Return the system time of all accumulated interval.*

### Protected Attributes

- [time\\_info start\\_](#)
- [time\\_info total\\_](#)

### 12.122.1 Detailed Description

A timekeeper that accumulate interval of time.

### 12.122.2 Member Function Documentation

#### 12.122.2.1 `void spot::timer::start ()` [inline]

Start a time interval.

#### 12.122.2.2 `void spot::timer::stop ()` [inline]

Stop a time interval and update the sum of all intervals.

#### 12.122.2.3 `clock_t spot::timer::utime () const` [inline]

Return the user time of all accumulated interval.

Any time interval that has been [start\(\)](#)ed but not [stop\(\)](#)ed will not be accounted for.

#### 12.122.2.4 `clock_t spot::timer::stime () const` [inline]

Return the system time of all accumulated interval.

Any time interval that has been [start\(\)](#)ed but not [stop\(\)](#)ed will not be accounted for.

### 12.122.3 Member Data Documentation

#### 12.122.3.1 `time_info spot::timer::start_` [protected]

#### 12.122.3.2 `time_info spot::timer::total_` [protected]

The documentation for this class was generated from the following file:

- [misc/timer.hh](#)

## 12.123 `spot::timer_map` Class Reference

A map of [timer](#), where each [timer](#) has a name.

```
#include <misc/timer.hh>
```

### Public Member Functions

- `void start (const std::string &name)`  
*Start a [timer](#) with name name.*
- `void stop (const std::string &name)`  
*Stop [timer](#) name.*
- `void cancel (const std::string &name)`  
*Cancel [timer](#) name.*
- `const spot::timer & timer (const std::string &name) const`  
*Return the [timer](#) name.*

- `spot::timer & timer` (const std::string &name)  
*Return the `timer` name.*
- `bool empty ()` const  
*Whether there is no `timer` in the map.*
- `std::ostream & print` (std::ostream &os) const  
*Format information about all timers in a table.*

### Protected Types

- `typedef std::pair< spot::timer, int > item_type`
- `typedef std::map< std::string, item_type > tm_type`

### Protected Attributes

- `tm_type tm`

#### 12.123.1 Detailed Description

A map of `timer`, where each `timer` has a name.

Timer\_map also keeps track of the number of measures each `timer` has performed.

#### 12.123.2 Member Typedef Documentation

**12.123.2.1** `typedef std::pair<spot::timer, int> spot::timer_map::item_type` [protected]

**12.123.2.2** `typedef std::map<std::string, item_type> spot::timer_map::tm_type` [protected]

#### 12.123.3 Member Function Documentation

**12.123.3.1** `void spot::timer_map::start (const std::string & name)` [inline]

Start a `timer` with name *name*.

The `timer` is created if it did not exist already. Once started, a `timer` should be either `stop()`ed or `cancel()`ed.

**12.123.3.2** `void spot::timer_map::stop (const std::string & name)` [inline]

Stop `timer` *name*.

The `timer` must have been previously started with `start()`.

**12.123.3.3 void spot::timer\_map::cancel (const std::string & name) [inline]**

Cancel [timer](#) *name*.

The [timer](#) must have been previously started with [start\(\)](#).

This cancel only the current measure. (Previous measures recorded by the [timer](#) are preserved.) When a [timer](#) that has not done any measure is canceled, it is removed from the map.

**12.123.3.4 const spot::timer& spot::timer\_map::timer (const std::string & name) const [inline]**

Return the [timer](#) *name*.

**12.123.3.5 spot::timer& spot::timer\_map::timer (const std::string & name) [inline]**

Return the [timer](#) *name*.

**12.123.3.6 bool spot::timer\_map::empty () const [inline]**

Whether there is no [timer](#) in the map.

If [empty\(\)](#) return true, then either no [timer](#) where ever started, or all started timers were canceled without completing any measure.

**12.123.3.7 std::ostream& spot::timer\_map::print (std::ostream & os) const**

Format information about all timers in a table.

**12.123.4 Member Data Documentation****12.123.4.1 tm\_type spot::timer\_map::tm [protected]**

The documentation for this class was generated from the following file:

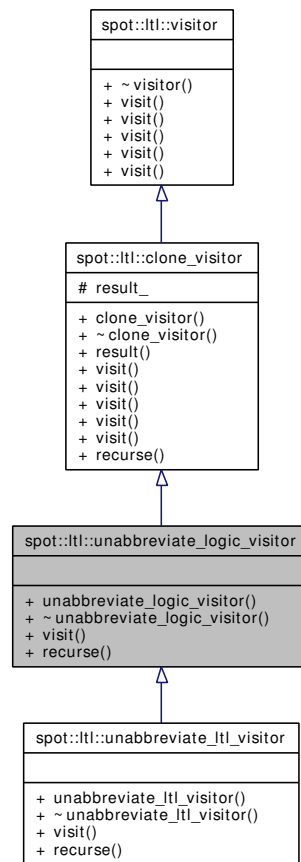
- [misc/timer.hh](#)

**12.124 spot::ltl::unabbreviate\_logic\_visitor Class Reference**

Clone and rewrite a [formula](#) to remove most of the abbreviated logical operators.

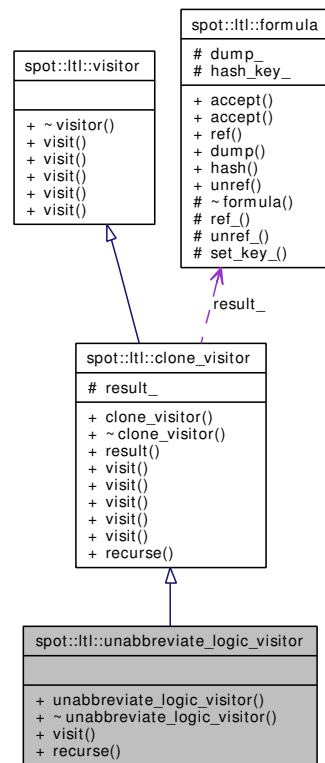
```
#include <ltlvisit/ltlunabbrev.hh>
```

Inheritance diagram for spot::ltl::unabbreviate\_logic\_visitor:





Collaboration diagram for spot::ltl::unabbreviate\_logic\_visitor:



## Public Member Functions

- [unabbreviate\\_logic\\_visitor](#) ()
- virtual [~unabbreviate\\_logic\\_visitor](#) ()
- void [visit](#) ([binop](#) \*bo)
- virtual [formula](#) \* [recurse](#) ([formula](#) \*f)
- [formula](#) \* [result](#) () const
- void [visit](#) ([atomic\\_prop](#) \*ap)
- void [visit](#) ([unop](#) \*uo)
- void [visit](#) ([multop](#) \*mo)
- void [visit](#) ([constant](#) \*c)

## Protected Attributes

- [formula](#) \* [result\\_](#)

## Private Types

- typedef [clone\\_visitor](#) [super](#)

### 12.124.1 Detailed Description

Clone and rewrite a [formula](#) to remove most of the abbreviated logical operators.

This will rewrite binary operators such as [binop::Implies](#), [binop::Equals](#), and [binop::Xor](#), using only [unop::Not](#), [multop::Or](#), and [multop::And](#).

This [visitor](#) is public, because it's convenient to derive from it and override some of its methods. But if you just want the functionality, consider using [spot::ltl::unabbreviate\\_logic](#) instead.

### 12.124.2 Member Typedef Documentation

**12.124.2.1** `typedef clone_visitor spot::ltl::unabbreviate_logic_visitor::super` `[private]`

Reimplemented in [spot::ltl::unabbreviate\\_ltl\\_visitor](#).

### 12.124.3 Constructor & Destructor Documentation

**12.124.3.1** `spot::ltl::unabbreviate_logic_visitor::unabbreviate_logic_visitor ()`

**12.124.3.2** `virtual spot::ltl::unabbreviate_logic_visitor::~~unabbreviate_logic_visitor ()`  
`[virtual]`

### 12.124.4 Member Function Documentation

**12.124.4.1** `void spot::ltl::unabbreviate_logic_visitor::visit (binop * bo)` `[virtual]`

Reimplemented from [spot::ltl::clone\\_visitor](#).

**12.124.4.2** `virtual formula* spot::ltl::unabbreviate_logic_visitor::recurse (formula * f)`  
`[virtual]`

Reimplemented from [spot::ltl::clone\\_visitor](#).

Reimplemented in [spot::ltl::unabbreviate\\_ltl\\_visitor](#).

**12.124.4.3** `formula* spot::ltl::clone_visitor::result () const` `[inherited]`

**12.124.4.4** `void spot::ltl::clone_visitor::visit (atomic_prop * ap)` `[virtual, inherited]`

Implements [spot::ltl::visitor](#).

**12.124.4.5** `void spot::ltl::clone_visitor::visit (unop * uo)` `[virtual, inherited]`

Implements [spot::ltl::visitor](#).

Reimplemented in [spot::ltl::unabbreviate\\_ltl\\_visitor](#).

**12.124.4.6** `void spot::ltl::clone_visitor::visit (multop * mo)` `[virtual, inherited]`

Implements [spot::ltl::visitor](#).

**12.124.4.7** void spot::ltl::clone\_visitor::visit (constant \* c) [virtual, inherited]

Implements [spot::ltl::visitor](#).

## 12.124.5 Member Data Documentation

**12.124.5.1** formula\* spot::ltl::clone\_visitor::result\_ [protected, inherited]

The documentation for this class was generated from the following file:

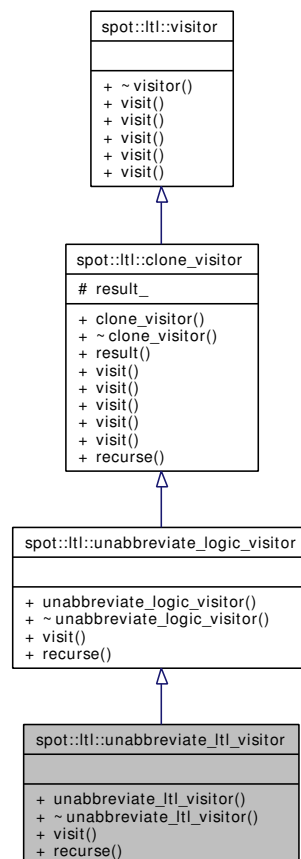
- ltlvisit/[lunabbrev.hh](#)

## 12.125 spot::ltl::unabbreviate\_ltl\_visitor Class Reference

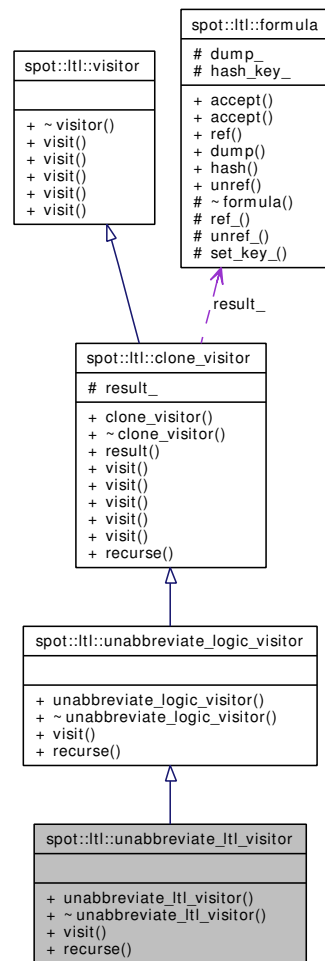
Clone and rewrite a [formula](#) to remove most of the abbreviated LTL and logical operators.

```
#include <ltlvisit/tunabbrev.hh>
```

Inheritance diagram for spot::ltl::unabbreviate\_ltl\_visitor:



Collaboration diagram for spot::ltl::unabbreviate\_ltl\_visitor:



## Public Member Functions

- [unabbreviate\\_ltl\\_visitor\(\)](#)
- [virtual ~unabbreviate\\_ltl\\_visitor\(\)](#)
- [void visit\(unop \\*uo\)](#)
- [formula \\* recurse\(formula \\*f\)](#)
- [void visit\(binop \\*bo\)](#)
- [void visit\(atomic\\_prop \\*ap\)](#)
- [void visit\(multop \\*mo\)](#)
- [void visit\(constant \\*c\)](#)
- [formula \\* result\(\)](#) const

## Protected Attributes

- [formula \\* result\\_](#)

## Private Types

- typedef `unabbreviate_logic_visitor` `super`

### 12.125.1 Detailed Description

Clone and rewrite a [formula](#) to remove most of the abbreviated LTL and logical operators.

The rewriting performed on logical operator is the same as the one done by [spot::ltl::unabbreviate\\_logic\\_visitor](#).

This will also rewrite unary operators such as `unop::F`, and `unop::G`, using only `binop::U`, and `binop::R`.

This [visitor](#) is public, because it's convenient to derive from it and override some of its methods. But if you just want the functionality, consider using [spot::ltl::unabbreviate\\_ltl](#) instead.

### 12.125.2 Member Typedef Documentation

**12.125.2.1**    typedef            `unabbreviate_logic_visitor`            `spot::ltl::unabbreviate_ltl_visitor::super`  
[private]

Reimplemented from [spot::ltl::unabbreviate\\_logic\\_visitor](#).

### 12.125.3 Constructor & Destructor Documentation

**12.125.3.1**    `spot::ltl::unabbreviate_ltl_visitor::unabbreviate_ltl_visitor ()`

**12.125.3.2**    virtual `spot::ltl::unabbreviate_ltl_visitor::~~unabbreviate_ltl_visitor ()` [virtual]

### 12.125.4 Member Function Documentation

**12.125.4.1**    void `spot::ltl::unabbreviate_ltl_visitor::visit (unop * uo)` [virtual]

Reimplemented from [spot::ltl::clone\\_visitor](#).

**12.125.4.2**    formula\* `spot::ltl::unabbreviate_ltl_visitor::recurse (formula * f)` [virtual]

Reimplemented from [spot::ltl::unabbreviate\\_logic\\_visitor](#).

**12.125.4.3**    void        `spot::ltl::unabbreviate_logic_visitor::visit (binop * bo)` [virtual, inherited]

Reimplemented from [spot::ltl::clone\\_visitor](#).

**12.125.4.4**    void `spot::ltl::clone_visitor::visit (atomic_prop * ap)` [virtual, inherited]

Implements [spot::ltl::visitor](#).

**12.125.4.5**    void `spot::ltl::clone_visitor::visit (multop * mo)` [virtual, inherited]

Implements [spot::ltl::visitor](#).

**12.125.4.6** void spot::ltl::clone\_visitor::visit (constant \* c) [virtual, inherited]

Implements [spot::ltl::visitor](#).

**12.125.4.7** formula\* spot::ltl::clone\_visitor::result () const [inherited]

## 12.125.5 Member Data Documentation

**12.125.5.1** formula\* spot::ltl::clone\_visitor::result\_ [protected, inherited]

The documentation for this class was generated from the following file:

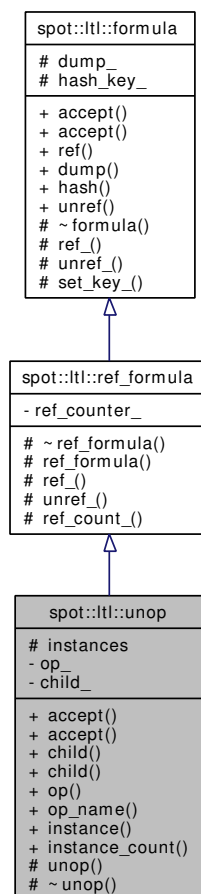
- ltlvisit/[tunabbrev.hh](#)

## 12.126 spot::ltl::unop Class Reference

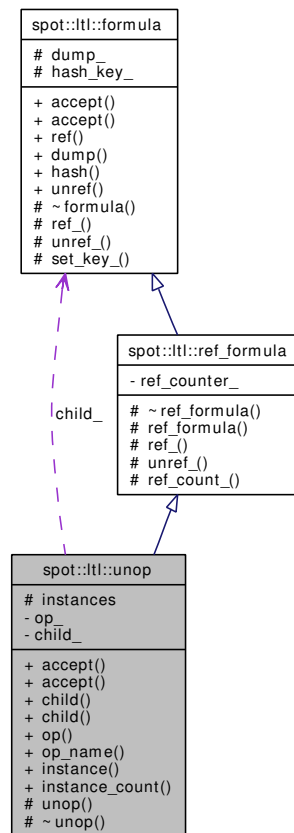
Unary operators.

```
#include <ltlast/unop.hh>
```

Inheritance diagram for spot::ltl::unop:



Collaboration diagram for spot::ltl::unop:



## Public Types

- enum type { Not, X, F, G }

## Public Member Functions

- virtual void `accept` (visitor &v)  
*Entry point for vspot::ltl::visitor instances.*
- virtual void `accept` (const\_visitor &v) const  
*Entry point for vspot::ltl::const\_visitor instances.*
- const formula \* `child` () const  
*Get the sole operand of this operator.*
- formula \* `child` ()  
*Get the sole operand of this operator.*
- type op () const  
*Get the type of this operator.*

- const char \* [op\\_name](#) () const  
*Get the type of this operator, as a string.*
- [formula](#) \* [ref](#) ()  
*clone this node*
- const std::string & [dump](#) () const  
*Return a canonic representation of the [formula](#).*
- const size\_t [hash](#) () const  
*Return a hash\_key for the [formula](#).*

### Static Public Member Functions

- static [unop](#) \* [instance](#) (type op, [formula](#) \*child)
- static unsigned [instance\\_count](#) ()  
*Number of instantiated unary operators. For debugging.*
- static void [unref](#) ([formula](#) \*f)  
*release this node*

### Protected Types

- typedef std::pair< [type](#), [formula](#) \* > [pair](#)
- typedef std::map< [pair](#), [formula](#) \* > [map](#)

### Protected Member Functions

- [unop](#) (type op, [formula](#) \*child)
- virtual [~unop](#) ()
- void [ref\\_](#) ()  
*increment reference counter if any*
- bool [unref\\_](#) ()  
*decrement reference counter if any, return true when the instance must be deleted (usually when the counter hits 0).*
- unsigned [ref\\_count\\_](#) ()  
*Number of references to this [formula](#).*
- void [set\\_key\\_](#) ()  
*Compute key\_ from dump\_.*



**Protected Attributes**

- std::string [dump\\_](#)  
*The canonic representation of the [formula](#).*
- size\_t [hash\\_key\\_](#)  
*The hash key of this [formula](#).*

**Static Protected Attributes**

- static [map instances](#)

**Private Attributes**

- [type op\\_](#)
- [formula \\* child\\_](#)

**12.126.1 Detailed Description**

Unary operators.

**12.126.2 Member Typedef Documentation**

**12.126.2.1** `typedef std::pair<type, formula*> spot::ltl::unop::pair` [protected]

**12.126.2.2** `typedef std::map<pair, formula*> spot::ltl::unop::map` [protected]

**12.126.3 Member Enumeration Documentation**

**12.126.3.1** `enum spot::ltl::unop::type`

Enumerator:

*Not*

*X*

*F*

*G*

**12.126.4 Constructor & Destructor Documentation**

**12.126.4.1** `spot::ltl::unop::unop (type op, formula * child)` [protected]

**12.126.4.2** `virtual spot::ltl::unop::~~unop ()` [protected, virtual]

### 12.126.5 Member Function Documentation

#### 12.126.5.1 `static unop* spot::ltl::unop::instance (type op, formula * child)` [static]

Build an unary operator with operation *op* and child *child*.

#### 12.126.5.2 `virtual void spot::ltl::unop::accept (visitor & v)` [virtual]

Entry point for `vspot::ltl::visitor` instances.

Implements [spot::ltl::formula](#).

#### 12.126.5.3 `virtual void spot::ltl::unop::accept (const_visitor & v) const` [virtual]

Entry point for `vspot::ltl::const_visitor` instances.

Implements [spot::ltl::formula](#).

#### 12.126.5.4 `const formula* spot::ltl::unop::child () const`

Get the sole operand of this operator.

#### 12.126.5.5 `formula* spot::ltl::unop::child ()`

Get the sole operand of this operator.

#### 12.126.5.6 `type spot::ltl::unop::op () const`

Get the type of this operator.

#### 12.126.5.7 `const char* spot::ltl::unop::op_name () const`

Get the type of this operator, as a string.

#### 12.126.5.8 `static unsigned spot::ltl::unop::instance_count ()` [static]

Number of instantiated unary operators. For debugging.

#### 12.126.5.9 `void spot::ltl::ref_formula::ref_ ()` [protected, virtual, inherited]

increment reference counter if any

Reimplemented from [spot::ltl::formula](#).

#### 12.126.5.10 `bool spot::ltl::ref_formula::unref_ ()` [protected, virtual, inherited]

decrement reference counter if any, return true when the instance must be deleted (usually when the counter hits 0).

Reimplemented from [spot::ltl::formula](#).

#### 12.126.5.11 `unsigned spot::ltl::ref_formula::ref_count_ ()` [protected, inherited]

Number of references to this [formula](#).

**12.126.5.12** formula\* spot::ltl::formula::ref () [inherited]

clone this node

This increments the reference counter of this node (if one is used). You should almost never use this method directly as it doesn't touch the children. If you want to clone a whole [formula](#), use [spot::ltl::clone\(\)](#) instead.

**12.126.5.13** static void spot::ltl::formula::unref (formula \*f) [static, inherited]

release this node

This decrements the reference counter of this node (if one is used) and can free the object. You should almost never use this method directly as it doesn't touch the children. If you want to release a whole [formula](#), use [spot::ltl::destroy\(\)](#) instead.

**12.126.5.14** const std::string& spot::ltl::formula::dump () const [inherited]

Return a canonic representation of the [formula](#).

**12.126.5.15** const size\_t spot::ltl::formula::hash () const [inline, inherited]

Return a hash\_key for the [formula](#).

**12.126.5.16** void spot::ltl::formula::set\_key\_ () [protected, inherited]

Compute key\_ from dump\_.

Should be called once in each object, after dump\_ has been set.

**12.126.6** Member Data Documentation**12.126.6.1** map spot::ltl::unop::instances [static, protected]**12.126.6.2** type spot::ltl::unop::op\_ [private]**12.126.6.3** formula\* spot::ltl::unop::child\_ [private]**12.126.6.4** std::string spot::ltl::formula::dump\_ [protected, inherited]

The canonic representation of the [formula](#).

**12.126.6.5** size\_t spot::ltl::formula::hash\_key\_ [protected, inherited]

The hash key of this [formula](#).

Initialized by [set\\_key\\_\(\)](#).

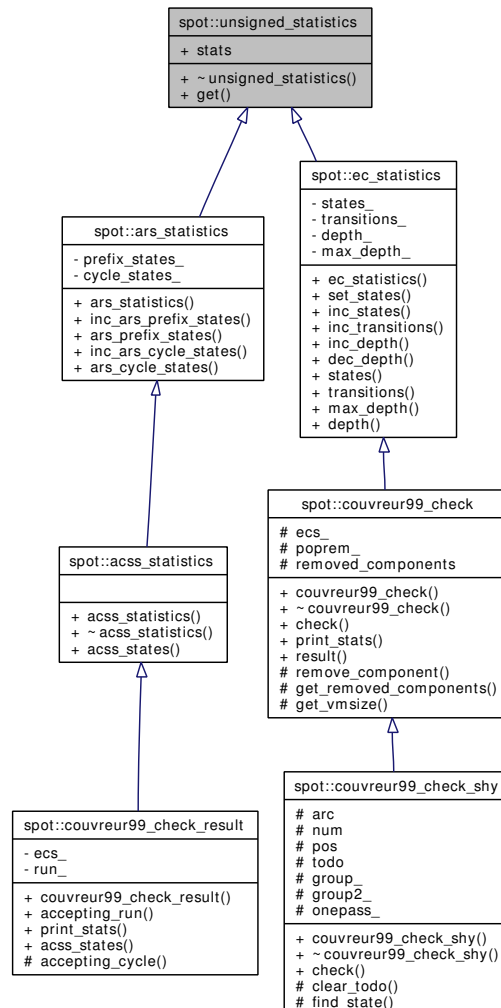
The documentation for this class was generated from the following file:

- [ltlast/unop.hh](#)

## 12.127 spot::unsigned\_statistics Struct Reference

```
#include <tgbaaalgos/emptiness_stats.hh>
```

Inheritance diagram for spot::unsigned\_statistics:



### Public Types

- typedef unsigned(unsigned\_statistics::\*) [unsigned\\_fun](#) () const
- typedef std::map< const char \*, [unsigned\\_fun](#), [char\\_ptr\\_less\\_than](#) > [stats\\_map](#)

### Public Member Functions

- virtual [~unsigned\\_statistics](#) ()
- unsigned [get](#) (const char \*str) const

### Public Attributes

- [stats\\_map](#) stats

### 12.127.1 Member Typedef Documentation

**12.127.1.1** `typedef unsigned(unsigned_statistics::*) spot::unsigned_statistics::unsigned_fun() const`

**12.127.1.2** `typedef std::map<const char*, unsigned_fun, char_ptr_less_than> spot::unsigned_statistics::stats_map`

### 12.127.2 Constructor & Destructor Documentation

**12.127.2.1** `virtual spot::unsigned_statistics::~~unsigned_statistics() [inline, virtual]`

### 12.127.3 Member Function Documentation

**12.127.3.1** `unsigned spot::unsigned_statistics::get(const char * str) const [inline]`

### 12.127.4 Member Data Documentation

**12.127.4.1** `stats_map spot::unsigned_statistics::stats`

The documentation for this struct was generated from the following file:

- [tgbaalgos/emptiness\\_stats.hh](#)

## 12.128 spot::unsigned\_statistics\_copy Class Reference

comparable statistics

```
#include <tgbaalgos/emptiness_stats.hh>
```

### Public Types

- `typedef std::map< const char *, unsigned, char\_ptr\_less\_than > stats_map`

### Public Member Functions

- [unsigned\\_statistics\\_copy](#) ()
- [unsigned\\_statistics\\_copy](#) (const [unsigned\\_statistics](#) &o)
- bool [seteq](#) (const [unsigned\\_statistics](#) &o)
- bool [operator==](#) (const [unsigned\\_statistics\\_copy](#) &o) const
- bool [operator!=](#) (const [unsigned\\_statistics\\_copy](#) &o) const

### Public Attributes

- [stats\\_map](#) [stats](#)
- bool [set](#)

### 12.128.1 Detailed Description

comparable statistics

This must be built from a [spot::unsigned\\_statistics](#). But unlike [spot::unsigned\\_statistics](#), it supports equality and inequality tests. (It's the only operations it supports, BTW.)

### 12.128.2 Member Typedef Documentation

**12.128.2.1** typedef std::map<const char\*, unsigned, char\_ptr\_less\_than> spot::unsigned\_statistics\_copy::stats\_map

### 12.128.3 Constructor & Destructor Documentation

**12.128.3.1** spot::unsigned\_statistics\_copy::unsigned\_statistics\_copy () [inline]

**12.128.3.2** spot::unsigned\_statistics\_copy::unsigned\_statistics\_copy (const unsigned\_statistics & o) [inline]

### 12.128.4 Member Function Documentation

**12.128.4.1** bool spot::unsigned\_statistics\_copy::seteq (const unsigned\_statistics & o) [inline]

**12.128.4.2** bool spot::unsigned\_statistics\_copy::operator== (const unsigned\_statistics\_copy & o) const [inline]

**12.128.4.3** bool spot::unsigned\_statistics\_copy::operator!= (const unsigned\_statistics\_copy & o) const [inline]

### 12.128.5 Member Data Documentation

**12.128.5.1** stats\_map spot::unsigned\_statistics\_copy::stats

**12.128.5.2** bool spot::unsigned\_statistics\_copy::set

The documentation for this class was generated from the following file:

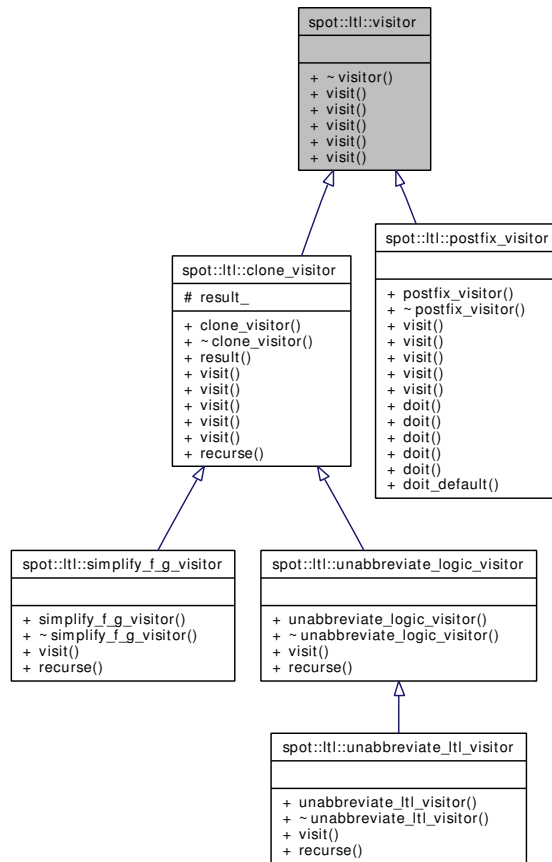
- [tgbaalgos/emptiness\\_stats.hh](#)

## 12.129 spot::ltl::visitor Struct Reference

Formula [visitor](#) that can modify the [formula](#).

```
#include <ltlast/visitor.hh>
```

Inheritance diagram for spot::ltl::visitor:



## Public Member Functions

- virtual [~visitor](#) ()
- virtual void [visit](#) ([atomic\\_prop](#) \*node)=0
- virtual void [visit](#) ([constant](#) \*node)=0
- virtual void [visit](#) ([binop](#) \*node)=0
- virtual void [visit](#) ([unop](#) \*node)=0
- virtual void [visit](#) ([multop](#) \*node)=0

### 12.129.1 Detailed Description

Formula [visitor](#) that can modify the [formula](#).

Writing visitors is the preferred way to traverse a [formula](#), since it doesn't involve any cast.

If you do not need to modify the visited [formula](#), inherit from [spot::ltl::const\\_visitor](#) instead.

### 12.129.2 Constructor & Destructor Documentation

#### 12.129.2.1 virtual [spot::ltl::visitor::~visitor](#) () [inline, virtual]

### 12.129.3 Member Function Documentation

#### 12.129.3.1 `virtual void spot::ltl::visitor::visit (atomic_prop * node)` [pure virtual]

Implemented in [spot::ltl::clone\\_visitor](#), and [spot::ltl::postfix\\_visitor](#).

#### 12.129.3.2 `virtual void spot::ltl::visitor::visit (constant * node)` [pure virtual]

Implemented in [spot::ltl::clone\\_visitor](#), and [spot::ltl::postfix\\_visitor](#).

#### 12.129.3.3 `virtual void spot::ltl::visitor::visit (binop * node)` [pure virtual]

Implemented in [spot::ltl::clone\\_visitor](#), [spot::ltl::unabbreviate\\_logic\\_visitor](#), [spot::ltl::postfix\\_visitor](#), and [spot::ltl::simplify\\_f\\_g\\_visitor](#).

#### 12.129.3.4 `virtual void spot::ltl::visitor::visit (unop * node)` [pure virtual]

Implemented in [spot::ltl::clone\\_visitor](#), [spot::ltl::postfix\\_visitor](#), and [spot::ltl::unabbreviate\\_ltl\\_visitor](#).

#### 12.129.3.5 `virtual void spot::ltl::visitor::visit (multop * node)` [pure virtual]

Implemented in [spot::ltl::clone\\_visitor](#), and [spot::ltl::postfix\\_visitor](#).

The documentation for this struct was generated from the following file:

- [ltlast/visitor.hh](#)

## 12.130 `spot::weight` Class Reference

Manage for a given automaton a vector of counter indexed by its acceptance condition.

```
#include <tgbaaalgos/weight.hh>
```

### Public Member Functions

- [weight](#) (const bdd &neg\_all\_cond)
- [weight](#) & [operator+=](#) (const bdd &acc)  
*Increment by one the counters of each acceptance condition in acc.*
- [weight](#) & [operator-=](#) (const bdd &acc)  
*Decrement by one the counters of each acceptance condition in acc.*
- bdd [operator-](#) (const [weight](#) &w) const

### Private Types

- typedef std::map< int, int > [weight\\_vector](#)

### Static Private Member Functions

- static void [inc\\_weight\\_handler](#) (char \*varset, int size)
- static void [dec\\_weight\\_handler](#) (char \*varset, int size)



**Private Attributes**

- `weight_vector` `m`
- bdd `neg_all_acc`

**Static Private Attributes**

- static `weight_vector` \* `pm`

**Friends**

- `std::ostream` & `operator<<` (`std::ostream` &`os`, const `weight` &`w`)

**12.130.1 Detailed Description**

Manage for a given automaton a vector of counter indexed by its acceptance condition.

**12.130.2 Member Typedef Documentation**

**12.130.2.1** `typedef std::map<int, int> spot::weight::weight_vector` `[private]`

**12.130.3 Constructor & Destructor Documentation**

**12.130.3.1** `spot::weight::weight (const bdd &neg_all_cond)`

Construct a empty vector (all counters set to zero).

**Parameters:**

*neg\_all\_cond* : negation of all the acceptance conditions of the automaton (the bdd returned by `tgba::neg_acceptance_conditions()`).

**12.130.4 Member Function Documentation**

**12.130.4.1** `weight& spot::weight::operator+= (const bdd &acc)`

Increment by one the counters of each acceptance condition in *acc*.

**12.130.4.2** `weight& spot::weight::operator-= (const bdd &acc)`

Decrement by one the counters of each acceptance condition in *acc*.

**12.130.4.3** `bdd spot::weight::operator- (const weight &w) const`

Return the set of each acceptance condition such that its counter is strictly greatest than the corresponding counter in *w*.

**Precondition:**

For each acceptance condition, its counter is greatest or equal to the corresponding counter in *w*.

**12.130.4.4** `static void spot::weight::inc_weight_handler (char * varset, int size)` [static, private]

**12.130.4.5** `static void spot::weight::dec_weight_handler (char * varset, int size)` [static, private]

### 12.130.5 Friends And Related Function Documentation

**12.130.5.1** `std::ostream& operator<< (std::ostream & os, const weight & w)` [friend]

### 12.130.6 Member Data Documentation

**12.130.6.1** `weight_vector spot::weight::m` [private]

**12.130.6.2** `bdd spot::weight::neg_all_acc` [private]

**12.130.6.3** `weight_vector* spot::weight::pm` [static, private]

The documentation for this class was generated from the following file:

- [tgbaalgos/weight.hh](#)

## 13 spot File Documentation

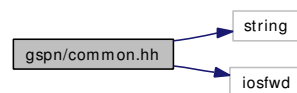
### 13.1 mainpage.dox File Reference

### 13.2 gspn/common.hh File Reference

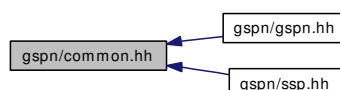
```
#include <string>
```

```
#include <iosfwd>
```

Include dependency graph for common.hh:



This graph shows which files directly or indirectly include this file:



### Namespaces

- namespace [spot](#)

## Classes

- class [spot::gspn\\_exception](#)  
An exception used to forward GSPN errors.

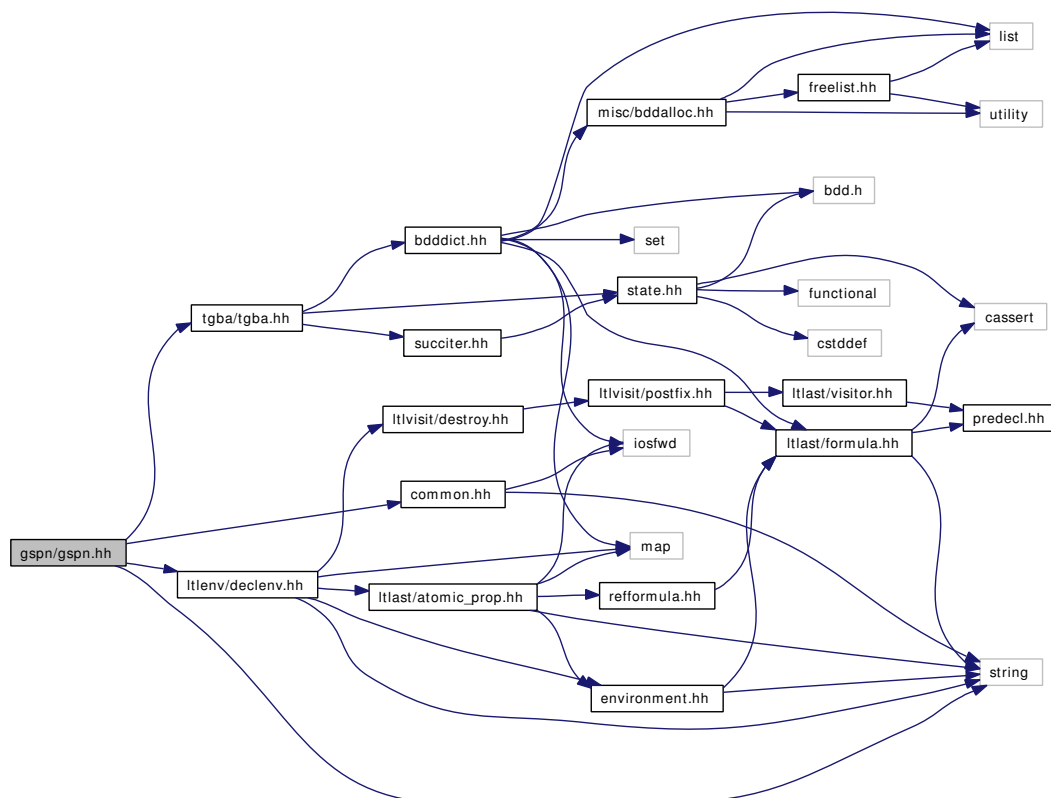
## Functions

- std::ostream & [spot::operator<<](#) (std::ostream &os, const gspn\_exception &e)

## 13.3 gspn/gspn.hh File Reference

```
#include <string>
#include "tgba/tgba.hh"
#include "common.hh"
#include "ltlenv/declenv.hh"
```

Include dependency graph for gspn.hh:



## Namespaces

- namespace [spot](#)

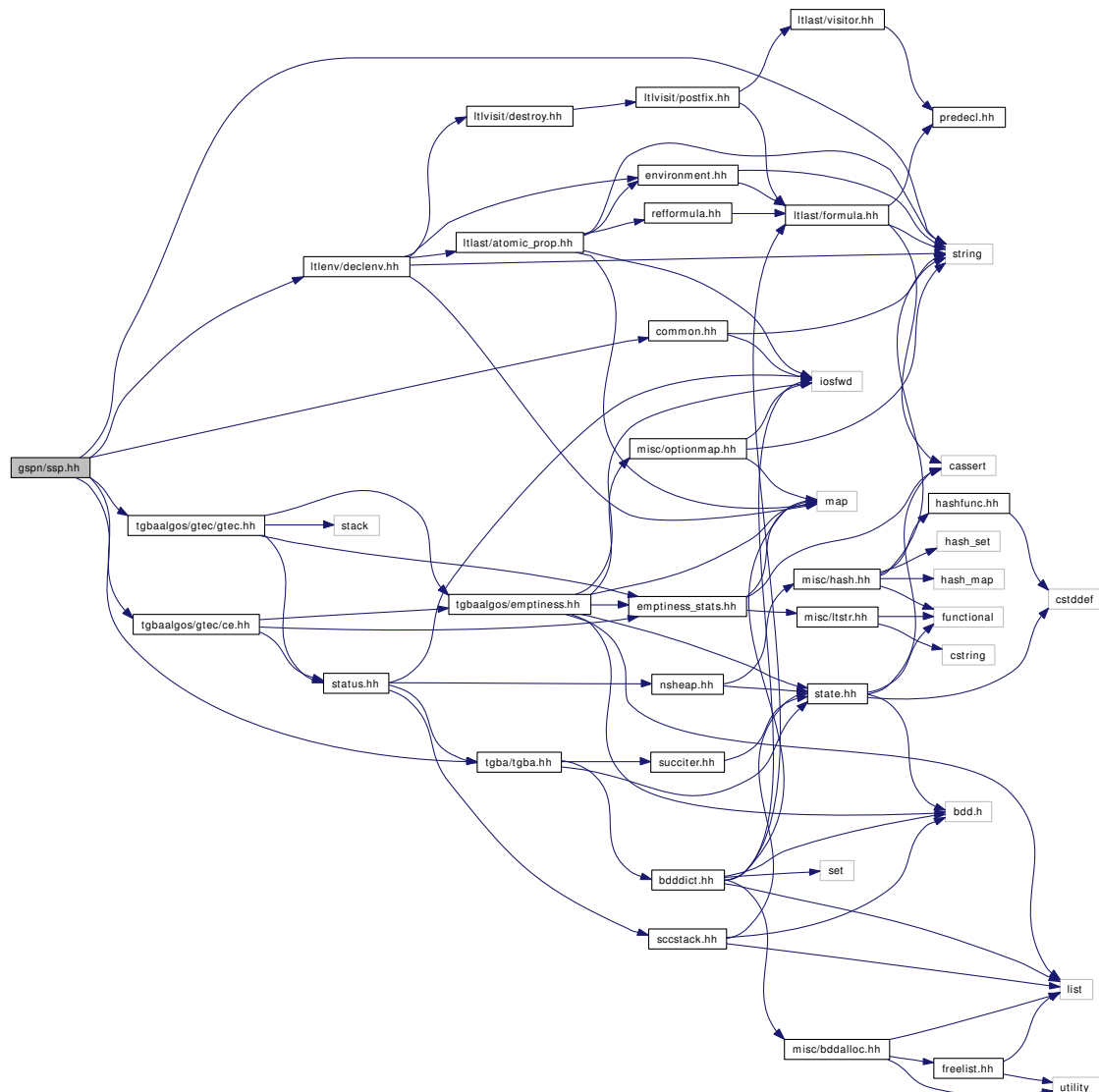
## Classes

- class `spot::gspn_interface`

## 13.4 gspn/ssp.hh File Reference

```
#include <string>
#include "tgba/tgba.hh"
#include "common.hh"
#include "tgbaalgos/gtec/gtec.hh"
#include "tgbaalgos/gtec/ce.hh"
#include "ltenv/declenv.hh"
```

Include dependency graph for ssp.hh:



## Namespaces

- namespace [spot](#)

## Classes

- class [spot::gspn\\_ssp\\_interface](#)

## Functions

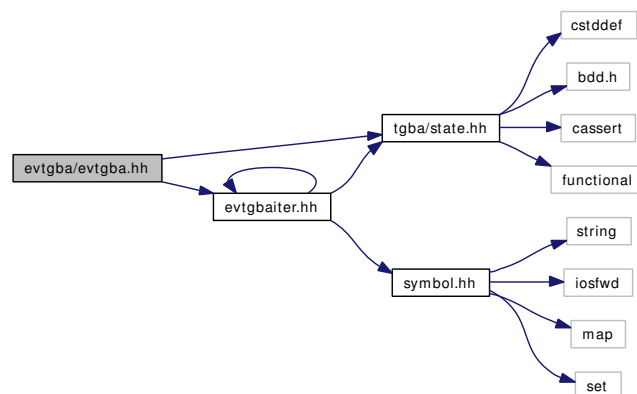
- couvreur99\_check \* [spot::couvreur99\\_check\\_ssp\\_semi](#) (const tgba \*ssp\_automata)
- couvreur99\_check \* [spot::couvreur99\\_check\\_ssp\\_shy\\_semi](#) (const tgba \*ssp\_automata)
- couvreur99\_check \* [spot::couvreur99\\_check\\_ssp\\_shy](#) (const tgba \*ssp\_automata, bool stack\_inclusion=true)

## 13.5 evtgba/evtgba.hh File Reference

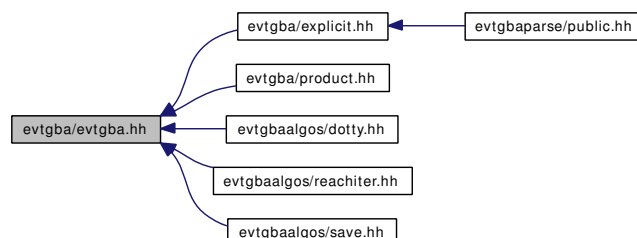
```
#include "tgba/state.hh"
```

```
#include "evtgbaiter.hh"
```

Include dependency graph for evtgba.hh:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [spot](#)

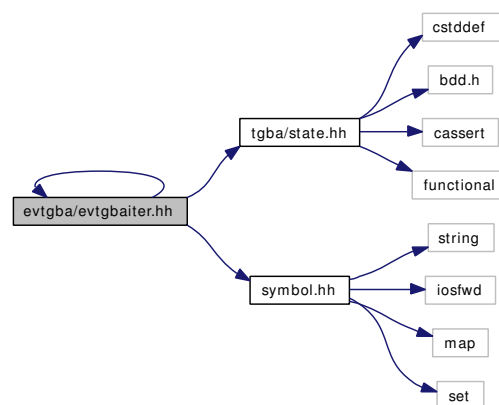
## Classes

- class [spot::evtgba](#)

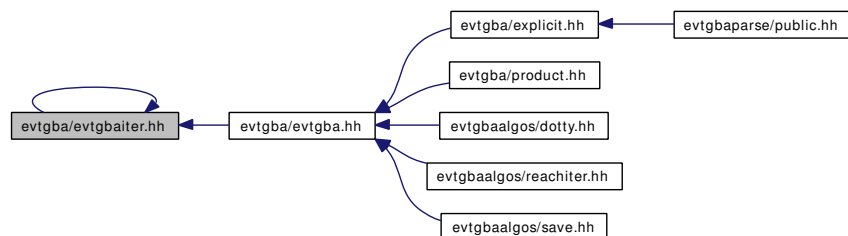
## 13.6 evtgba/evtgbaiter.hh File Reference

```
#include "tgba/state.hh"
#include "symbol.hh"
#include "evtgbaiter.hh"
```

Include dependency graph for evtgbaiter.hh:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [spot](#)

## Classes

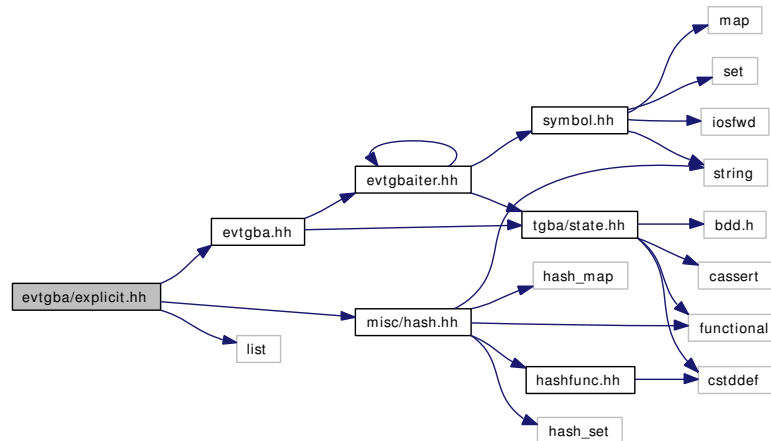
- class [spot::evtgba\\_iterator](#)

## 13.7 evtgba/explicit.hh File Reference

```
#include "evtgba.hh"
#include <list>
```

```
#include "misc/hash.hh"
```

Include dependency graph for explicit.hh:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [spot](#)

## Classes

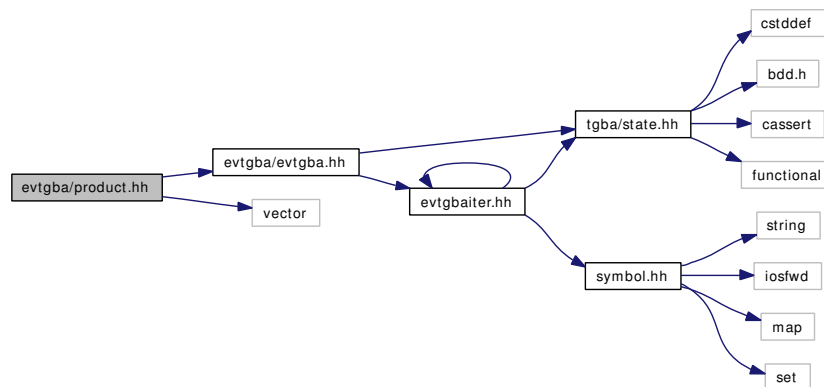
- class [spot::evtgba\\_explicit](#)
- struct [spot::evtgba\\_explicit::state](#)
- struct [spot::evtgba\\_explicit::transition](#)  
*Explicit transitions (used by [spot::evtgba\\_explicit](#)).*
- class [spot::state\\_evtgba\\_explicit](#)  
*States used by [spot::tgba\\_evtgba\\_explicit](#).*

## 13.8 evtgba/product.hh File Reference

```
#include "evtgba/evtgba.hh"
```

```
#include <vector>
```

Include dependency graph for product.hh:



## Namespaces

- namespace `spot`

## Classes

- class `spot::evtgba_product`

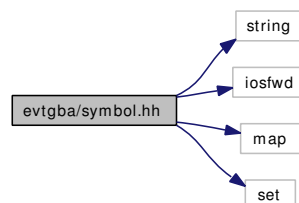
## 13.9 evtgba/symbol.hh File Reference

```

#include <string>
#include <iosfwd>
#include <map>
#include <set>

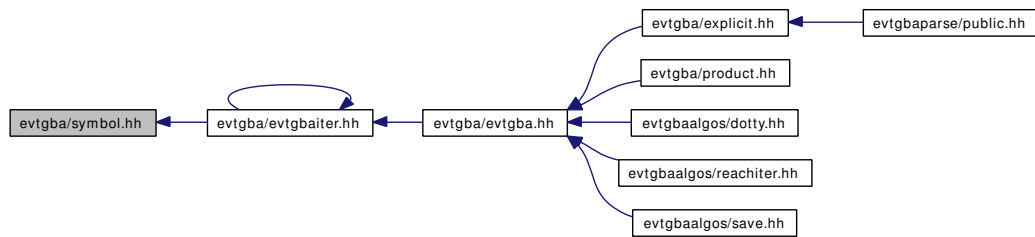
```

Include dependency graph for symbol.hh:





This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [spot](#)

## Classes

- class [spot::symbol](#)
- class [spot::rsymbol](#)

## Typedefs

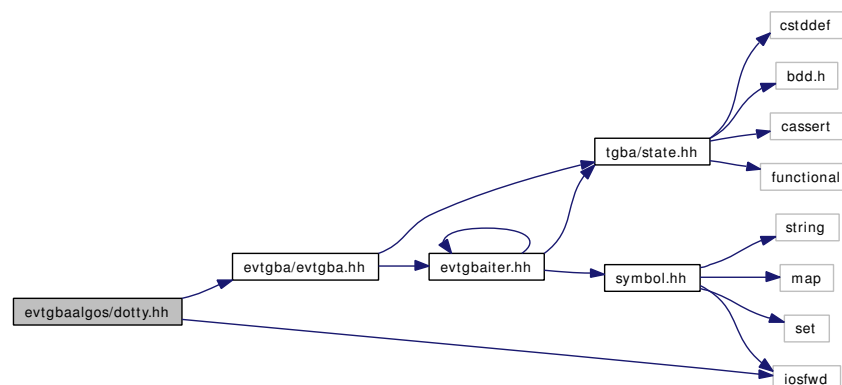
- typedef std::set< const symbol \* > [spot::symbol\\_set](#)
- typedef std::set< rsymbol > [spot::rsymbol\\_set](#)

## 13.10 evtgbaalgos/dotty.hh File Reference

```
#include "evtgba/evtgba.hh"
```

```
#include <iosfwd>
```

Include dependency graph for dotty.hh:



## Namespaces

- namespace [spot](#)

**Functions**

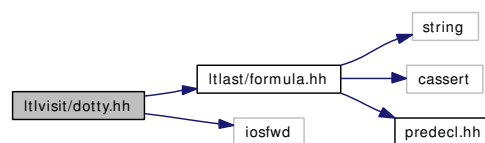
- `std::ostream & spot::dotty_reachable` (`std::ostream &os, const evtgba *g`)  
*Print reachable states in dot format.*

**13.11 Itlvisit/dotty.hh File Reference**

```
#include <ltlast/formula.hh>
```

```
#include <iosfwd>
```

Include dependency graph for dotty.hh:

**Namespaces**

- namespace `spot`
- namespace `spot::ltl`

**Functions**

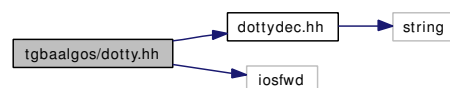
- `std::ostream & spot::ltl::dotty` (`std::ostream &os, const formula *f`)  
*Write a [formula](#) tree using dot's syntax.*

**13.12 tgbaalgorithms/dotty.hh File Reference**

```
#include "dottydec.hh"
```

```
#include <iosfwd>
```

Include dependency graph for dotty.hh:

**Namespaces**

- namespace `spot`

## Functions

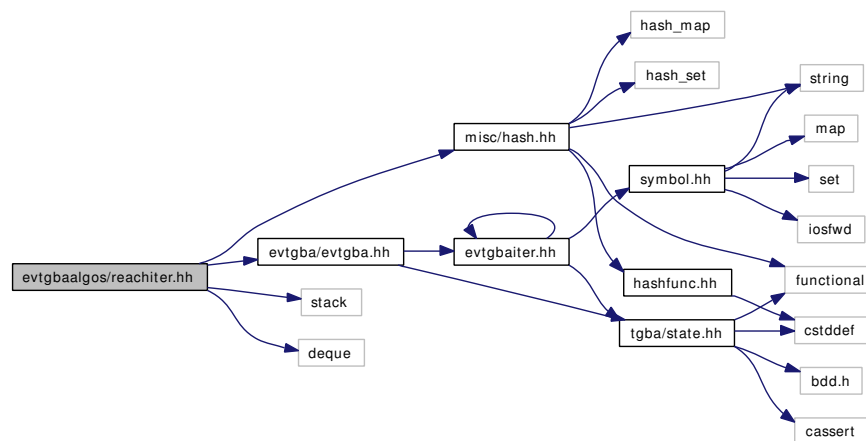
- `std::ostream & spot::dotty_reachable` (`std::ostream &os`, `const tgba *g`, `dotty_decorator *dd=dotty_decorator::instance()`)

*Print reachable states in dot format.*

## 13.13 evtgbaalgos/reachiter.hh File Reference

```
#include "misc/hash.hh"
#include "evtgba/evtgba.hh"
#include <stack>
#include <deque>
```

Include dependency graph for reachiter.hh:



## Namespaces

- namespace `spot`

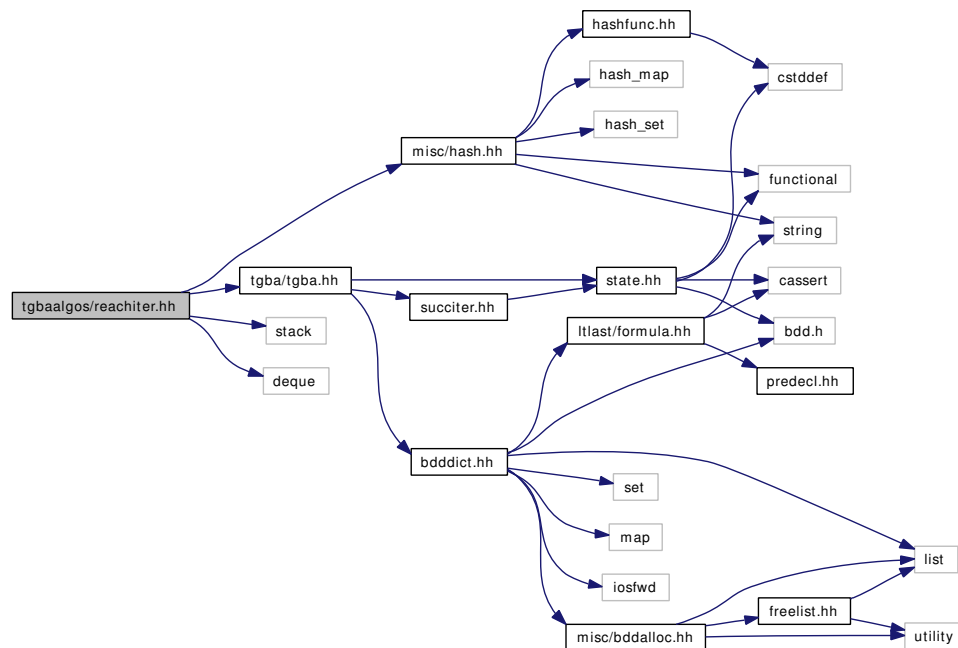
## Classes

- class `spot::evtgba_reachable_iterator`  
*Iterate over all reachable states of a `spot::evtgba`.*
- class `spot::evtgba_reachable_iterator_depth_first`  
*An implementation of `spot::evtgba_reachable_iterator` that browses states depth first.*
- class `spot::evtgba_reachable_iterator_breadth_first`  
*An implementation of `spot::evtgba_reachable_iterator` that browses states breadth first.*

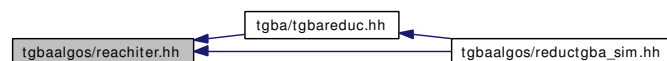
## 13.14 tgbaalgos/reachiter.hh File Reference

```
#include "misc/hash.hh"
#include "tgba/tgba.hh"
#include <stack>
#include <deque>
```

Include dependency graph for reachiter.hh:



This graph shows which files directly or indirectly include this file:



### Namespaces

- namespace [spot](#)

### Classes

- class [spot::tgba\\_reachable\\_iterator](#)  
*Iterate over all reachable states of a [spot::tgba](#).*
- class [spot::tgba\\_reachable\\_iterator\\_depth\\_first](#)  
*An implementation of [spot::tgba\\_reachable\\_iterator](#) that browses states depth first.*

- class [spot::tgba\\_reachable\\_iterator\\_breadth\\_first](#)

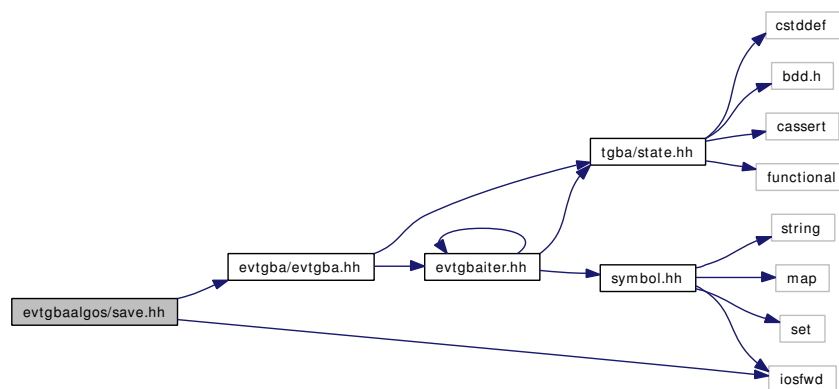
*An implementation of [spot::tgba\\_reachable\\_iterator](#) that browses states breadth first.*

## 13.15 evtgbaalgos/save.hh File Reference

```
#include "evtgba/evtgba.hh"
```

```
#include <iosfwd>
```

Include dependency graph for save.hh:



## Namespaces

- namespace [spot](#)

## Functions

- `std::ostream & spot::evtgba\_save\_reachable (std::ostream &os, const evtgba *g)`

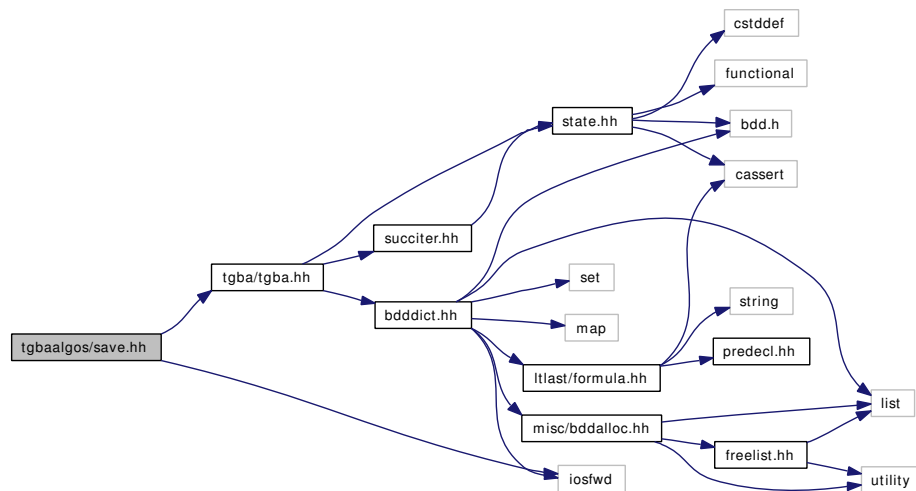
*Save reachable states in text format.*

## 13.16 tgbaalgos/save.hh File Reference

```
#include "tgba/tgba.hh"
```

```
#include <iosfwd>
```

Include dependency graph for save.hh:



## Namespaces

- namespace [spot](#)

## Functions

- `std::ostream & spot::tgba\_save\_reachable (std::ostream &os, const tgba *g)`  
Save reachable states in text format.

## 13.17 evtgbaalgorithms/tgba2evtgba.hh File Reference

### Namespaces

- namespace [spot](#)

### Functions

- `evtgba_explicit * spot::tgba\_to\_evtgba (const tgba *a)`  
Convert a [tgba](#) into an [evtgba](#).

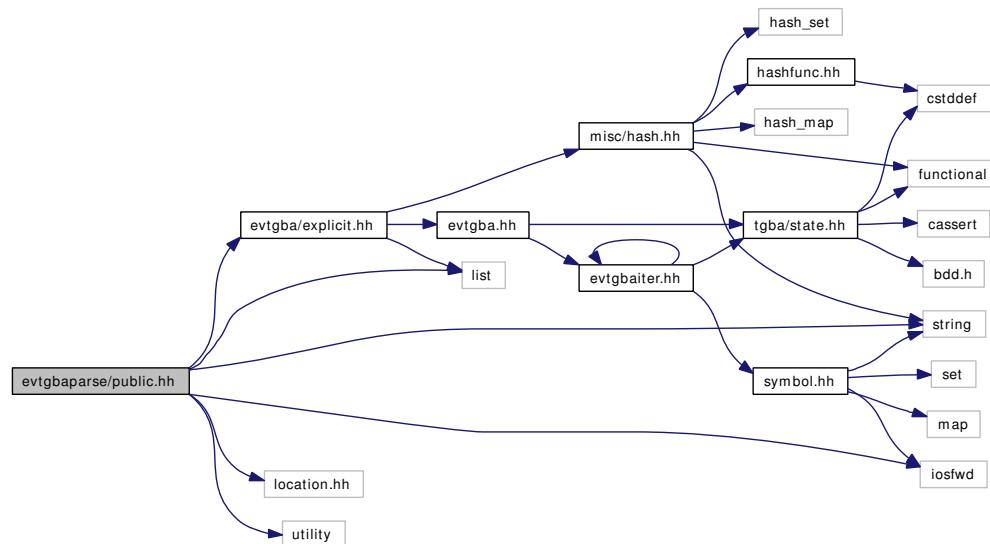
## 13.18 evtgbaparse/public.hh File Reference

```
#include "evtgba/explicit.hh"
#include "location.hh"
#include <string>
#include <list>
```

```
#include <utility>
```

```
#include <iosfwd>
```

Include dependency graph for public.hh:



## Namespaces

- namespace **spot**

## Typedefs

- `typedef std::pair< evtgbayy::location, std::string > spot::evtgba_parse_error`  
*A parse diagnostic with its location.*
- `typedef std::list< evtgba_parse_error > spot::evtgba_parse_error_list`  
*A list of parser diagnostics, as filled by parse.*

## Functions

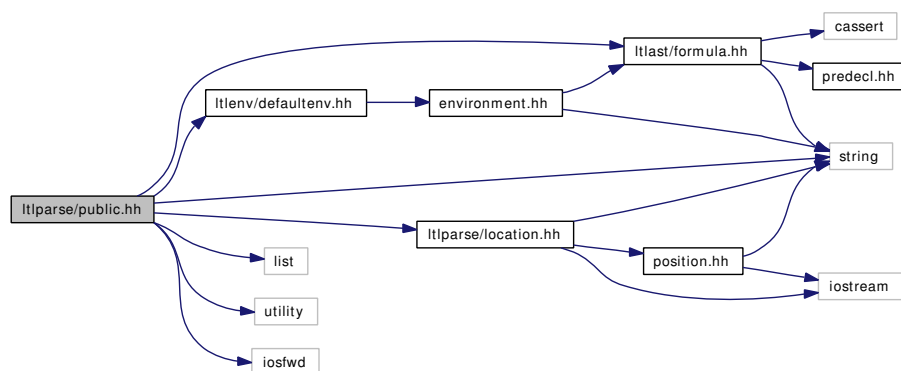
- `evtgba_explicit * spot::evtgba_parse` (const std::string &filename, evtgba\_parse\_error\_list &error\_list, bool debug=false)  
*Build a `spot::evtgba_explicit` from a text file.*
- `bool spot::format_evtgba_parse_errors` (std::ostream &os, const std::string &filename, evtgba\_parse\_error\_list &error\_list)  
*Format diagnostics produced by `spot::evtgba_parse`.*

### 13.19 Itlparse/public.hh File Reference

```
#include "ltlast/formula.hh"
```

```
#include "ltlparse/location.hh"
#include "ltlenv/defaultenv.hh"
#include <string>
#include <list>
#include <utility>
#include <iosfwd>
```

Include dependency graph for public.hh:



## Namespaces

- namespace [spot](#)
- namespace [spot::ltl](#)

## Typedefs

- typedef std::pair< [ltl::location](#), std::string > [spot::ltl::parse\\_error](#)  
*A parse diagnostic with its location.*
- typedef std::list< [parse\\_error](#) > [spot::ltl::parse\\_error\\_list](#)  
*A list of parser diagnostics, as filled by parse.*

## Functions

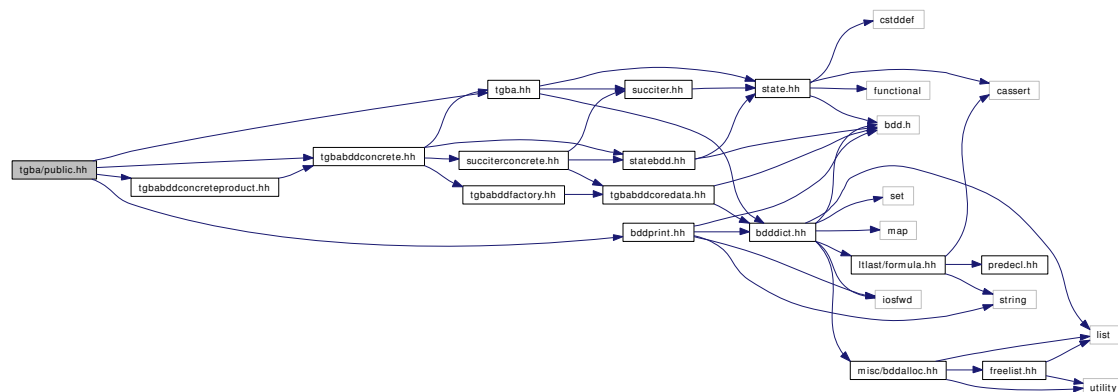
- formula \* [spot::ltl::parse](#) (const std::string &ltl\_string, parse\_error\_list &error\_list, environment &env=default\_environment::instance(), bool debug=false)  
*Build a [formula](#) from an LTL string.*
- bool [spot::ltl::format\\_parse\\_errors](#) (std::ostream &os, const std::string &ltl\_string, parse\_error\_list &error\_list)  
*Format diagnostics produced by [spot::ltl::parse](#).*



## 13.20 tgba/public.hh File Reference

```
#include "tgba.hh"
#include "tgbabddconcrete.hh"
#include "tgbabddconcreteproduct.hh"
#include "bddprint.hh"
```

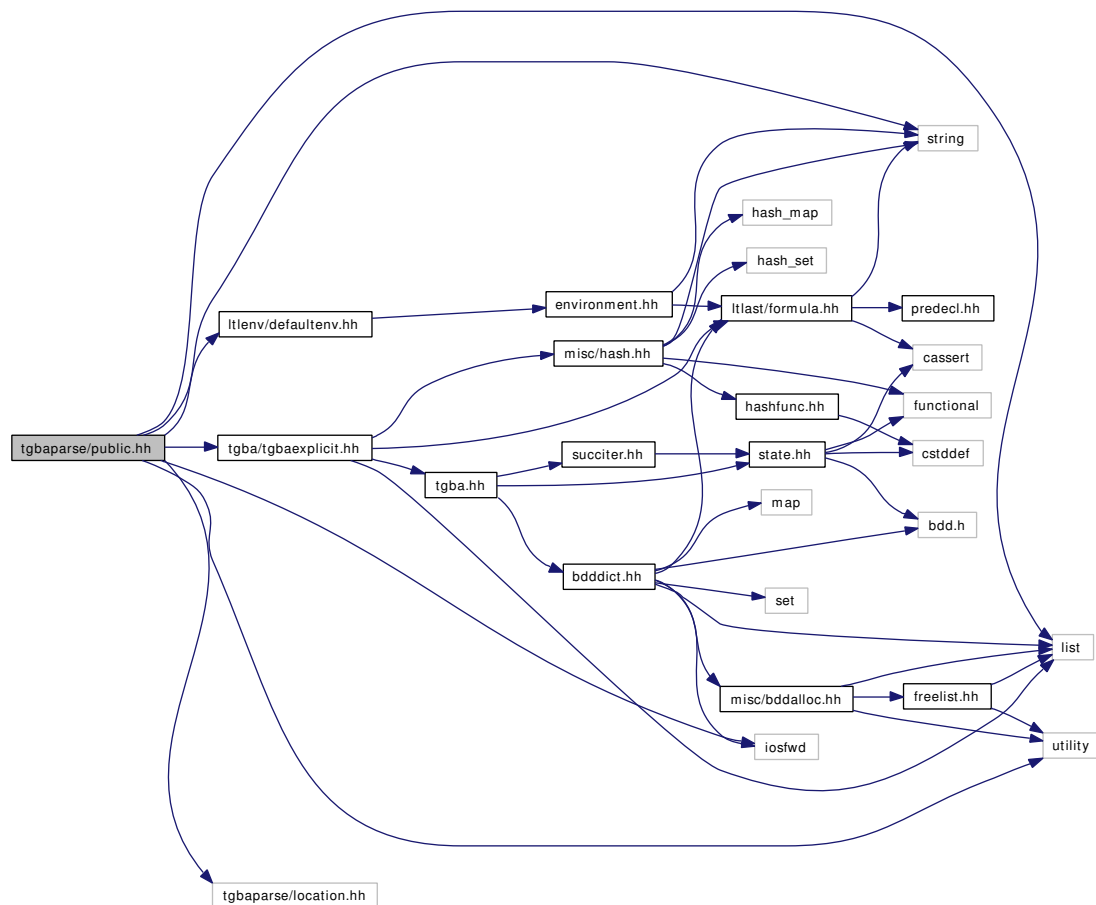
Include dependency graph for public.hh:



## 13.21 tgbaparse/public.hh File Reference

```
#include "tgba/tgbaexplicit.hh"
#include "tgbaparse/location.hh"
#include "ltlenv/defaultenv.hh"
#include <string>
#include <list>
#include <utility>
#include <iosfwd>
```

Include dependency graph for public.hh:



## Namespaces

- namespace [spot](#)

## Typedefs

- typedef std::pair< tgbayy::location, std::string > [spot::tgba\\_parse\\_error](#)  
A parse diagnostic with its location.
- typedef std::list< tgba\_parse\_error > [spot::tgba\\_parse\\_error\\_list](#)  
A list of parser diagnostics, as filled by parse.

## Functions

- tgba\_explicit \* [spot::tgba\\_parse](#) (const std::string &filename, tgba\_parse\_error\_list &error\_list, bdd\_dict \*dict, ltl::environment &env=ltl::default\_environment::instance(), ltl::environment &envacc=ltl::default\_environment::instance(), bool debug=false)  
Build a [spot::tgba\\_explicit](#) from a text file.

- bool [spot::format\\_tgba\\_parse\\_errors](#) (std::ostream &os, const std::string &filename, tgba\_parse\_error\_list &error\_list)

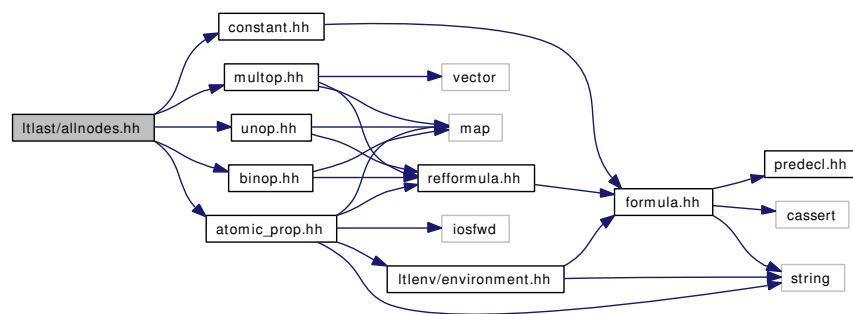
*Format diagnostics produced by [spot::tgba\\_parse](#).*

## 13.22 Itlast/allnodes.hh File Reference

Define all LTL node types.

```
#include "binop.hh"
#include "unop.hh"
#include "multop.hh"
#include "atomic_prop.hh"
#include "constant.hh"
```

Include dependency graph for allnodes.hh:



### 13.22.1 Detailed Description

Define all LTL node types.

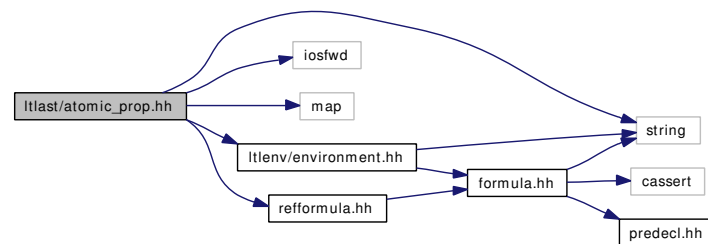
This file is usually needed when **defining** a visitor. Prefer [ltlast/predecl.hh](#) when only **declaring** methods and functions over LTL nodes.

## 13.23 Itlast/atomic\_prop.hh File Reference

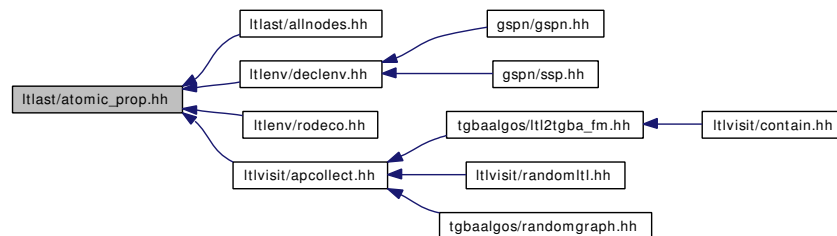
LTL atomic propositions.

```
#include <string>
#include <iosfwd>
#include <map>
#include "reformula.hh"
#include "ltlenv/environment.hh"
```

Include dependency graph for atomic\_prop.hh:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace `spot`
- namespace `spot::ltl`

## Classes

- class `spot::ltl::atomic_prop`  
*Atomic propositions.*

### 13.23.1 Detailed Description

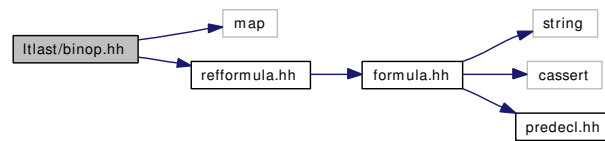
LTL atomic propositions.

## 13.24 Itlast/binop.hh File Reference

LTL binary operators.

```
#include <map>
#include "reformula.hh"
```

Include dependency graph for binop.hh:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [spot](#)
- namespace [spot::ltl](#)

## Classes

- class [spot::ltl::binop](#)  
*Binary operator.*

### 13.24.1 Detailed Description

LTL binary operators.

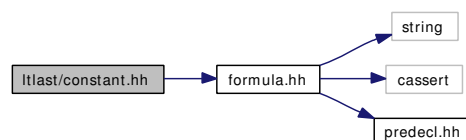
This does not include AND and OR operators. These are considered to be multi-operand operators (see [spot::ltl::multop](#)).

## 13.25 Itlast/constant.hh File Reference

LTL constants.

```
#include "formula.hh"
```

Include dependency graph for constant.hh:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [spot](#)
- namespace [spot::ltl](#)

## Classes

- class [spot::ltl::constant](#)  
A *constant* (*True or False*).

### 13.25.1 Detailed Description

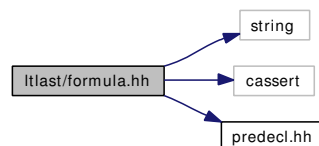
LTL constants.

## 13.26 Itlast/formula.hh File Reference

LTL formula interface.

```
#include <string>
#include <cassert>
#include "predecl.hh"
```

Include dependency graph for formula.hh:





- struct [spot::ltl::formula\\_ptr\\_less\\_than](#)  
*Strict Weak Ordering for const formula\*.*
- struct [spot::ltl::formula\\_ptr\\_hash](#)  
*Hash Function for const formula\*.*

### 13.26.1 Detailed Description

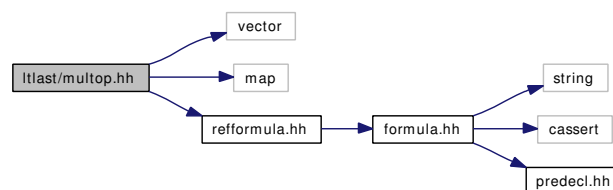
LTL formula interface.

## 13.27 Itlast/multop.hh File Reference

LTL multi-operand operators.

```
#include <vector>
#include <map>
#include "reformula.hh"
```

Include dependency graph for multop.hh:



This graph shows which files directly or indirectly include this file:



### Namespaces

- namespace [spot](#)
- namespace [spot::ltl](#)

### Classes

- class [spot::ltl::multop](#)  
*Multi-operand operators.*
- struct [spot::ltl::multop::paircmp](#)  
*Comparison functor used internally by `ltl::multop`.*



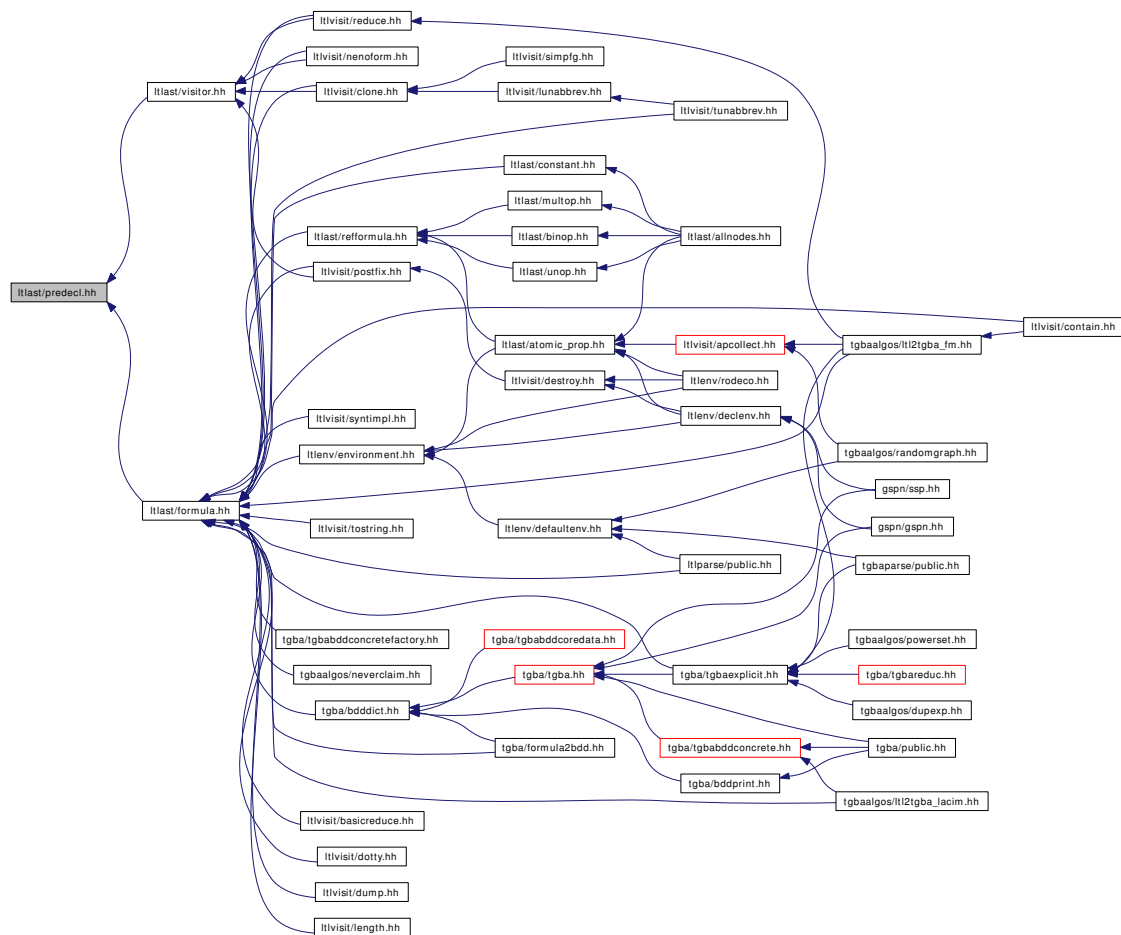
### 13.27.1 Detailed Description

LTl multi-operand operators.

## 13.28 `ltlast/predecl.hh` File Reference

Predeclare all LTL node types.

This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace `spot`
- namespace `spot::ltn`

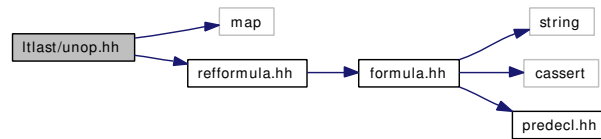
### 13.28.1 Detailed Description

Predeclare all LTL node types.

This file is usually used when **declaring** methods and functions over LTL nodes. Use [ltlast/allnodes.hh](#) or an individual header when the definition of the node is actually needed.



Include dependency graph for unop.hh:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace `spot`
- namespace `spot::ltn`

## Namespaces

- namespace `spot`
- namespace `spot::ltl`

## Classes

- struct `spot::ltl::visitor`  
Formula *visitor* that can modify the *formula*.
- struct `spot::ltl::const_visitor`  
Formula *visitor* that cannot modify the *formula*.

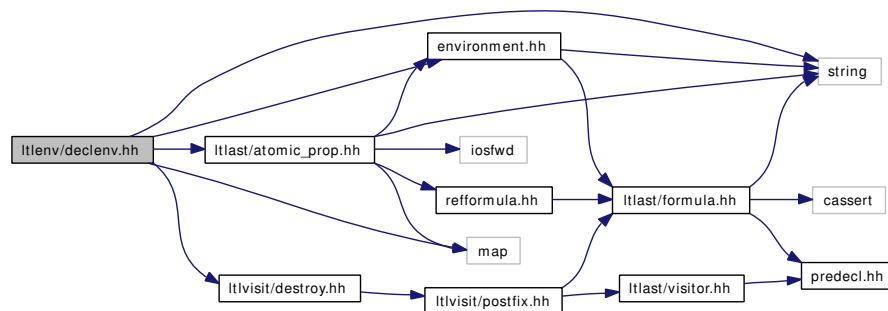
### 13.31.1 Detailed Description

LTL visitor interface.

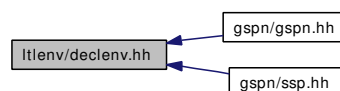
## 13.32 Itlenv/declenv.hh File Reference

```
#include "environment.hh"
#include <string>
#include <map>
#include "ltlvisit/destroy.hh"
#include "ltlast/atomic_prop.hh"
```

Include dependency graph for declenv.hh:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [spot](#)
- namespace [spot::ltl](#)

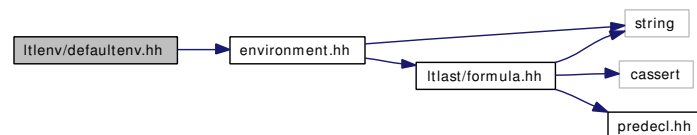
## Classes

- class [spot::ltl::declarative\\_environment](#)  
A declarative [environment](#).

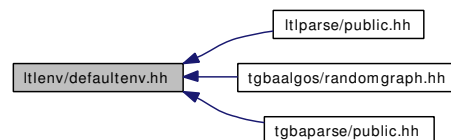
## 13.33 Itlenv/defaultenv.hh File Reference

```
#include "environment.hh"
```

Include dependency graph for defaultenv.hh:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [spot](#)
- namespace [spot::ltl](#)

## Classes

- class [spot::ltl::default\\_environment](#)  
A laxist [environment](#).

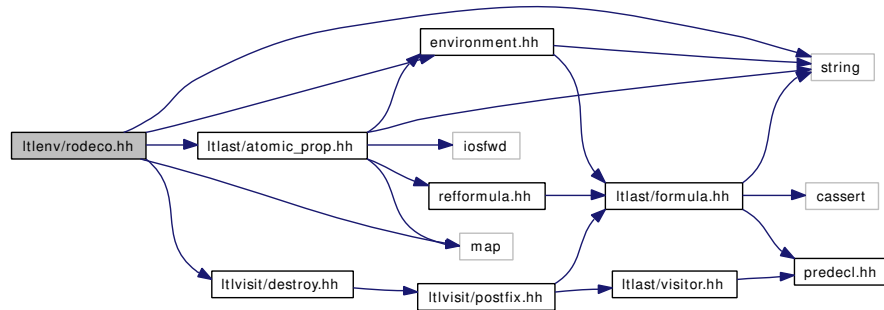
## 13.34 Itlenv/environment.hh File Reference

```
#include "ltlast/formula.hh"
```

```
#include <string>
```



Include dependency graph for rodeco.hh:



### Namespaces

- namespace [spot](#)
- namespace [spot::ltl](#)

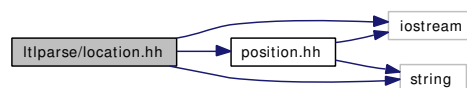
### Classes

- class [spot::ltl::read\\_only\\_environment](#)  
A read only *environment*.

## 13.36 Itlparse/location.hh File Reference

```
#include <iostream>
#include <string>
#include "position.hh"
```

Include dependency graph for location.hh:



This graph shows which files directly or indirectly include this file:



### Namespaces

- namespace [ltltyy](#)

## Classes

- class [ltlyy::location](#)

*Abstract a [location](#).*

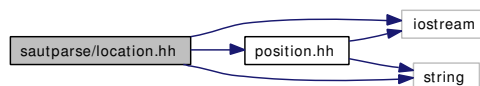
## Functions

- const location [ltlyy::operator+](#) (const location &begin, const location &end)  
*Join two [location](#) objects to create a [location](#).*
- const location [ltlyy::operator+](#) (const location &begin, unsigned int width)  
*Add two [location](#) objects.*
- location & [ltlyy::operator+=](#) (location &res, unsigned int width)  
*Add and assign a [location](#).*
- std::ostream & [ltlyy::operator<<](#) (std::ostream &ostr, const location &loc)  
*Intercept output stream redirection.*

## 13.37 sautparse/location.hh File Reference

```
#include <iostream>
#include <string>
#include "position.hh"
```

Include dependency graph for location.hh:



## Namespaces

- namespace [sautyy](#)

## Classes

- class [sautyy::location](#)

*Abstract a [location](#).*



**Functions**

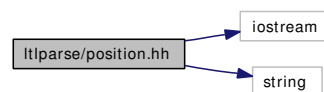
- const location [sauty::operator+](#) (const location &begin, const location &end)  
*Join two [location](#) objects to create a [location](#).*
- const location [sauty::operator+](#) (const location &begin, unsigned int width)  
*Add two [location](#) objects.*
- location & [sauty::operator+=](#) (location &res, unsigned int width)  
*Add and assign a [location](#).*
- std::ostream & [sauty::operator<<](#) (std::ostream &ostr, const location &loc)  
*Intercept output stream redirection.*

**13.38 Itlparse/position.hh File Reference**

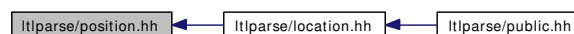
```
#include <iostream>
```

```
#include <string>
```

Include dependency graph for position.hh:



This graph shows which files directly or indirectly include this file:

**Namespaces**

- namespace [ltlyy](#)

**Classes**

- class [ltlyy::position](#)  
*Abstract a [position](#).*

**Functions**

- const position & [ltlyy::operator+=](#) (position &res, const int width)  
*Add and assign a [position](#).*
- const position [ltlyy::operator+](#) (const position &begin, const int width)  
*Add two [position](#) objects.*

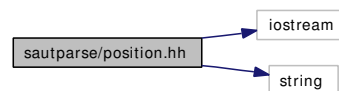
- const position & [ltlyy::operator=](#) (position &res, const int width)  
*Add and assign a [position](#).*
- const position [ltlyy::operator-](#) (const position &begin, const int width)  
*Add two [position](#) objects.*
- std::ostream & [ltlyy::operator<<](#) (std::ostream &ostr, const position &pos)  
*Intercept output stream redirection.*

## 13.39 sautparse/position.hh File Reference

```
#include <iostream>
```

```
#include <string>
```

Include dependency graph for position.hh:



This graph shows which files directly or indirectly include this file:



### Namespaces

- namespace [sauty](#)

### Classes

- class [sauty::position](#)  
*Abstract a [position](#).*

### Functions

- const position & [sauty::operator+=](#) (position &res, const int width)  
*Add and assign a [position](#).*
- const position [sauty::operator+](#) (const position &begin, const int width)  
*Add two [position](#) objects.*
- const position & [sauty::operator=](#) (position &res, const int width)  
*Add and assign a [position](#).*
- const position [sauty::operator-](#) (const position &begin, const int width)

Add two *position* objects.

- `std::ostream & sauty::operator<<` (`std::ostream &ostr, const position &pos`)

Intercept output stream redirection.

## 13.40 Itlparse/stack.hh File Reference

```
#include <deque>
```

Include dependency graph for stack.hh:



### Namespaces

- namespace `ltlyy`

### Classes

- class `ltlyy::stack< T, S >`
- class `ltlyy::slice< T, S >`

Present a *slice* of the top of a *stack*.

## 13.41 sautparse/stack.hh File Reference

```
#include <deque>
```

Include dependency graph for stack.hh:



### Namespaces

- namespace `sauty`

### Classes

- class `sauty::stack< T, S >`
- class `sauty::slice< T, S >`

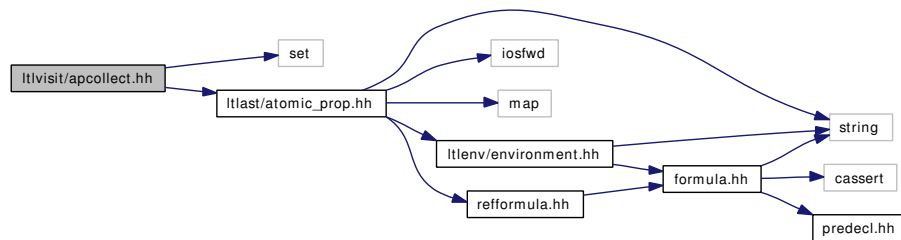
Present a *slice* of the top of a *stack*.

## 13.42 Itlvisit/apcollect.hh File Reference

```
#include <set>
```

```
#include "ltlast/atomic_prop.hh"
```

Include dependency graph for apcollect.hh:



This graph shows which files directly or indirectly include this file:



### Namespaces

- namespace `spot`
- namespace `spot::ltl`

### Typedefs

- typedef `std::set< atomic_prop *, formula_ptr_less_than >` `spot::ltl::atomic_prop_set`  
*Set of atomic propositions.*

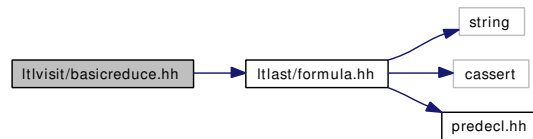
### Functions

- `atomic_prop_set *` `spot::ltl::atomic_prop_collect` (`const formula *f`, `atomic_prop_set *s=0`)  
*Return the set of atomic propositions occurring in a `formula`.*

## 13.43 Itlvisit/basicreduce.hh File Reference

```
#include "ltlast/formula.hh"
```

Include dependency graph for basicreduce.hh:



## Namespaces

- namespace `spot`
- namespace `spot::ltl`

## Functions

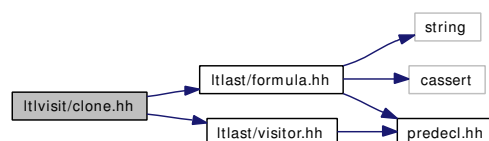
- formula \* `spot::ltl::basic_reduce` (const formula \*f)  
*Basic rewritings.*
- bool `spot::ltl::is_GF` (const formula \*f)  
*Whether a *formula* starts with GF.*
- bool `spot::ltl::is_FG` (const formula \*f)  
*Whether a *formula* starts with FG.*

## 13.44 Itlvisit/clone.hh File Reference

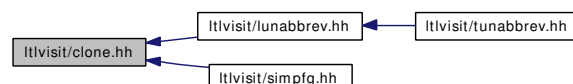
```
#include "ltlast/formula.hh"
```

```
#include "ltlast/visitor.hh"
```

Include dependency graph for clone.hh:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace `spot`
- namespace `spot::ltl`

## Classes

- class `spot::ltl::clone_visitor`  
*Clone a formula.*

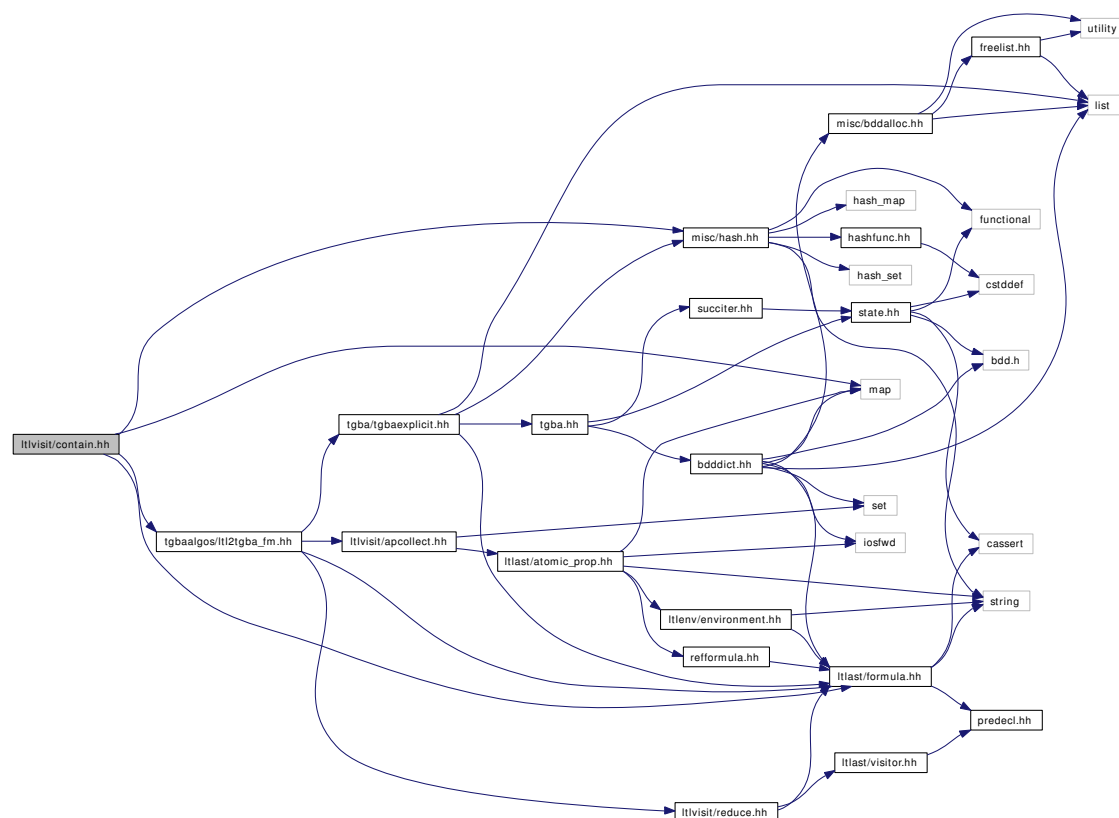
## Functions

- `formula * spot::ltl::clone` (const formula \*f)  
*Clone a `formula`.*

### 13.45 ltlvisit/contain.hh File Reference

```
#include "ltlast/formula.hh"
#include "tgbaalgos/ltl2tgba_fm.hh"
#include "misc/hash.hh"
#include <map>
```

Include dependency graph for contain.hh:



## Namespaces

- namespace [spot](#)
- namespace [spot::ltl](#)

## Classes

- class [spot::ltl::language\\_containment\\_checker](#)
- struct [spot::ltl::language\\_containment\\_checker::record\\_](#)

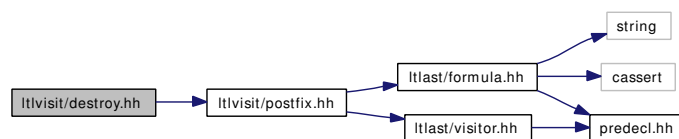
## Functions

- formula \* [spot::ltl::reduce\\_tau03](#) (const formula \*f, bool stronger=true)  
*Reduce a [formula](#) using language containment relationships.*

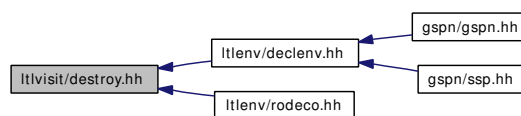
## 13.46 Itlvisit/destroy.hh File Reference

```
#include "ltlvisit/postfix.hh"
```

Include dependency graph for destroy.hh:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [spot](#)
- namespace [spot::ltl](#)

## Functions

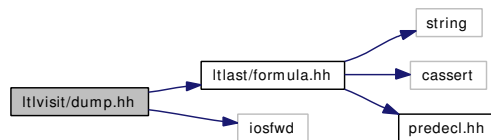
- void [spot::ltl::destroy](#) (const formula \*f)  
*Destroys a [formula](#).*

## 13.47 Itlvisit/dump.hh File Reference

```
#include "ltlast/formula.hh"
```

```
#include <iosfwd>
```

Include dependency graph for dump.hh:



### Namespaces

- namespace `spot`
- namespace `spot::ltl`

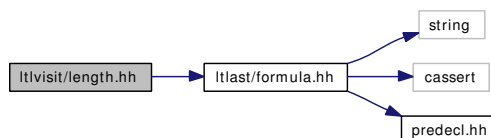
### Functions

- `std::ostream & spot::ltl::dump` (`std::ostream &os, const formula *f`)  
*Dump a `formula` tree.*

## 13.48 Itlvisit/length.hh File Reference

```
#include "ltlast/formula.hh"
```

Include dependency graph for length.hh:



### Namespaces

- namespace `spot`
- namespace `spot::ltl`

### Functions

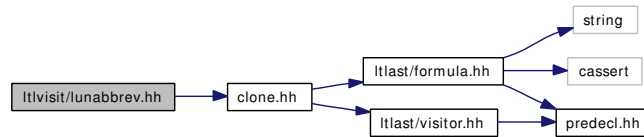
- `int spot::ltl::length` (`const formula *f`)  
*Compute the length of a `formula`.*



## 13.49 Itlvisit/lunabbrev.hh File Reference

```
#include "clone.hh"
```

Include dependency graph for lunabbrev.hh:



This graph shows which files directly or indirectly include this file:



### Namespaces

- namespace [spot](#)
- namespace [spot::ltl](#)

### Classes

- class [spot::ltl::unabbreviate\\_logic\\_visitor](#)  
Clone and rewrite a [formula](#) to remove most of the abbreviated logical operators.

### Functions

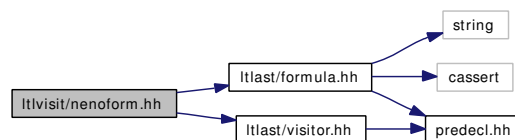
- formula \* [spot::ltl::unabbreviate\\_logic](#) (const formula \*f)  
Clone and rewrite a [formula](#) to remove most of the abbreviated logical operators.

## 13.50 Itlvisit/nenoform.hh File Reference

```
#include "ltlast/formula.hh"
```

```
#include "ltlast/visitor.hh"
```

Include dependency graph for nenoform.hh:



### Namespaces

- namespace [spot](#)
- namespace [spot::ltl](#)

## Functions

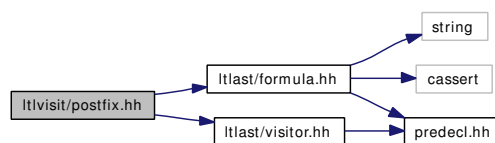
- formula \* [spot::ltl::negative\\_normal\\_form](#) (const formula \*f, bool negated=false)  
*Build the negative normal form of f.*

## 13.51 Itlvisit/postfix.hh File Reference

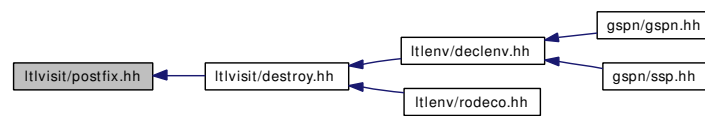
```
#include "ltlast/formula.hh"
```

```
#include "ltlast/visitor.hh"
```

Include dependency graph for postfix.hh:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [spot](#)
- namespace [spot::ltl](#)

## Classes

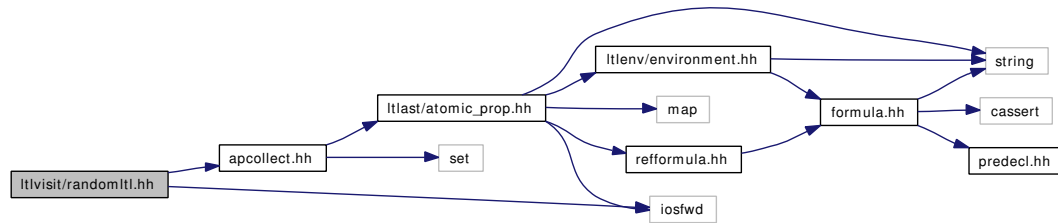
- class [spot::ltl::postfix\\_visitor](#)  
*Apply an algorithm on each node of an AST, during a postfix traversal.*

## 13.52 Itlvisit/randomltl.hh File Reference

```
#include "apcollect.hh"
```

```
#include <iosfwd>
```

Include dependency graph for randomltl.hh:



## Namespaces

- namespace [spot](#)
- namespace [spot::ltl](#)

## Classes

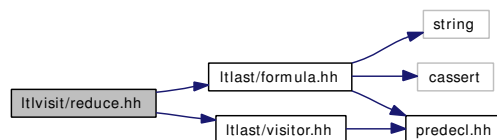
- class [spot::ltl::random\\_ltl](#)  
*Generate random LTL formulae.*
- struct [spot::ltl::random\\_ltl::op\\_proba](#)

## 13.53 Itlvisit/reduce.hh File Reference

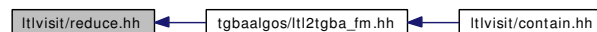
```
#include "ltlast/formula.hh"
```

```
#include "ltlast/visitor.hh"
```

Include dependency graph for reduce.hh:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [spot](#)
- namespace [spot::ltl](#)

## Enumerations

- enum `spot::ltl::reduce_options` {  
`spot::ltl::Reduce_None` = 0, `spot::ltl::Reduce_Basics` = 1, `spot::ltl::Reduce_Syntactic_Implications` = 2, `spot::ltl::Reduce_Eventuality_And_Universality` = 4,  
`spot::ltl::Reduce_Containment_Checks` = 8, `spot::ltl::Reduce_Containment_Checks_Stronger` = 16,  
`spot::ltl::Reduce_All` = -1U }  
*Options for `spot::ltl::reduce`.*

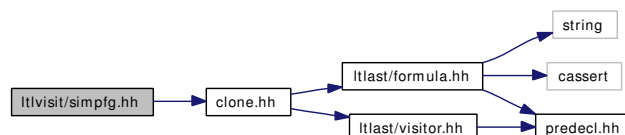
## Functions

- formula \* `spot::ltl::reduce` (const formula \*f, int opt=Reduce\_All)  
*Reduce a `formula` f.*
- bool `spot::ltl::is_eventual` (const formula \*f)  
*Check whether a `formula` is a pure eventuality.*
- bool `spot::ltl::is_universal` (const formula \*f)  
*Check whether a `formula` is purely universal.*

## 13.54 Itlvisit/simpfg.hh File Reference

```
#include "clone.hh"
```

Include dependency graph for simpfg.hh:



## Namespaces

- namespace `spot`
- namespace `spot::ltl`

## Classes

- class `spot::ltl::simplify_f_g_visitor`  
*Replace true U f and false R g by F f and G g.*

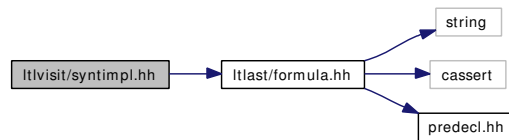
## Functions

- formula \* `spot::ltl::simplify_f_g` (const formula \*f)  
*Replace true U f and false R g by F f and G g.*

## 13.55 Itlvisit/syntimpl.hh File Reference

```
#include "ltlast/formula.hh"
```

Include dependency graph for syntimpl.hh:



### Namespaces

- namespace [spot](#)
- namespace [spot::ltl](#)

### Functions

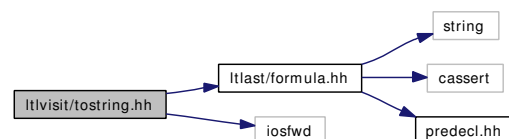
- bool [spot::ltl::syntactic\\_implication](#) (const formula \*f1, const formula \*f2)  
*Syntactic implication.*
- bool [spot::ltl::syntactic\\_implication\\_neg](#) (const formula \*f1, const formula \*f2, bool right)  
*Syntactic implication.*

## 13.56 Itlvisit/tostring.hh File Reference

```
#include <ltlast/formula.hh>
```

```
#include <iosfwd>
```

Include dependency graph for tostring.hh:



### Namespaces

- namespace [spot](#)
- namespace [spot::ltl](#)

### Functions

- std::ostream & [spot::ltl::to\\_string](#) (const formula \*f, std::ostream &os)  
*Output a [formula](#) as a (parsable) string.*
- std::string [spot::ltl::to\\_string](#) (const formula \*f)

Convert a *formula* into a (parsable) string.

- `std::ostream & spot::ltl::to_spin_string` (const formula \*f, std::ostream &os)

Output a *formula* as a (parsable by Spin) string.

- `std::string spot::ltl::to_spin_string` (const formula \*f)

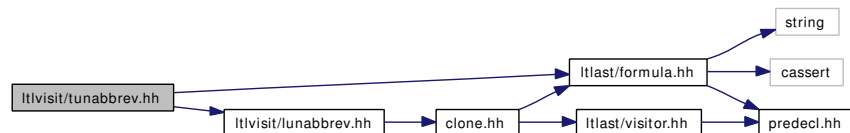
Convert a *formula* into a (parsable by Spin) string.

## 13.57 ltlvisit/tunabbrev.hh File Reference

```
#include "ltlast/formula.hh"
```

```
#include "ltlvisit/lunabbrev.hh"
```

Include dependency graph for tunabbrev.hh:



### Namespaces

- namespace `spot`
- namespace `spot::ltl`

### Classes

- class `spot::ltl::unabbreviate_ltl_visitor`

Clone and rewrite a *formula* to remove most of the abbreviated LTL and logical operators.

### Functions

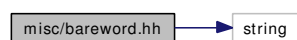
- formula \* `spot::ltl::unabbreviate_ltl` (const formula \*f)

Clone and rewrite a *formula* to remove most of the abbreviated LTL and logical operators.

## 13.58 misc/bareword.hh File Reference

```
#include <string>
```

Include dependency graph for bareword.hh:



## Namespaces

- namespace [spot](#)

## Functions

- bool [spot::is\\_bare\\_word](#) (const char \*str)
- std::string [spot::quote\\_unless\\_bare\\_word](#) (const std::string &str)

*Double-quote words that are not bare.*

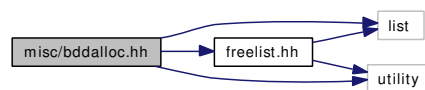
## 13.59 misc/bddalloc.hh File Reference

```
#include "freelist.hh"
```

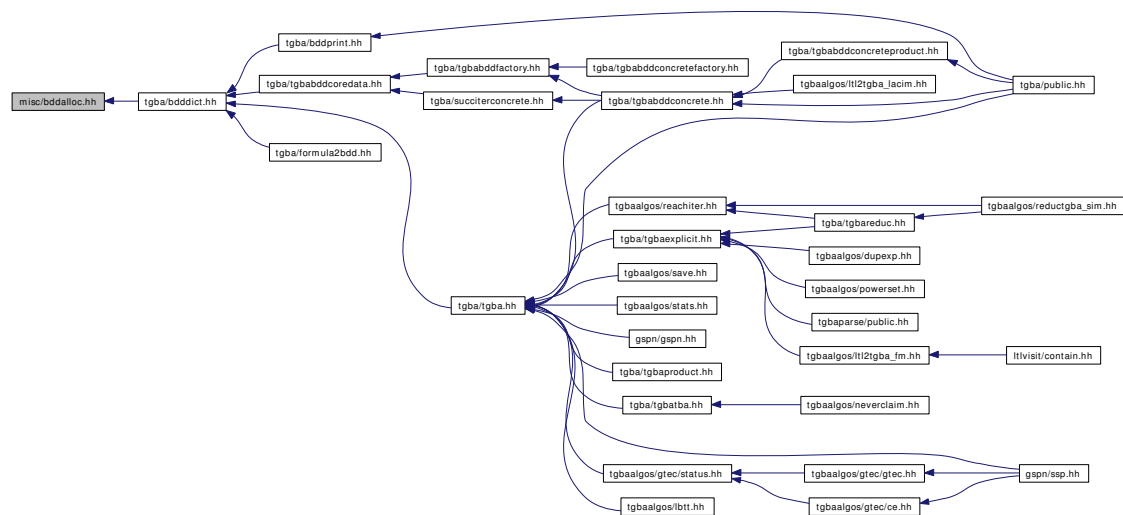
```
#include <list>
```

```
#include <utility>
```

Include dependency graph for bddalloc.hh:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [spot](#)

## Classes

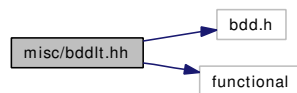
- class [spot::bdd\\_allocator](#)  
*Manage ranges of variables.*

## 13.60 misc/bddlt.hh File Reference

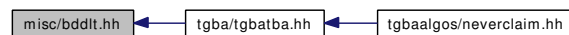
```
#include <bdd.h>
```

```
#include <functional>
```

Include dependency graph for bddlt.hh:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [spot](#)

## Classes

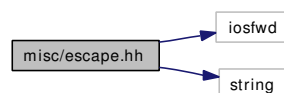
- struct [spot::bdd\\_less\\_than](#)  
*Comparison functor for BDDs.*

## 13.61 misc/escape.hh File Reference

```
#include <iosfwd>
```

```
#include <string>
```

Include dependency graph for escape.hh:



## Namespaces

- namespace [spot](#)



## Functions

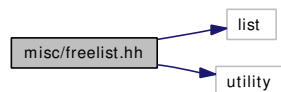
- `std::ostream & spot::escape\_str (std::ostream &os, const std::string &str)`  
*Escape " and \ characters in str.*
- `std::string spot::escape\_str (const std::string &str)`  
*Escape " and \ characters in str.*

### 13.62 misc/freelist.hh File Reference

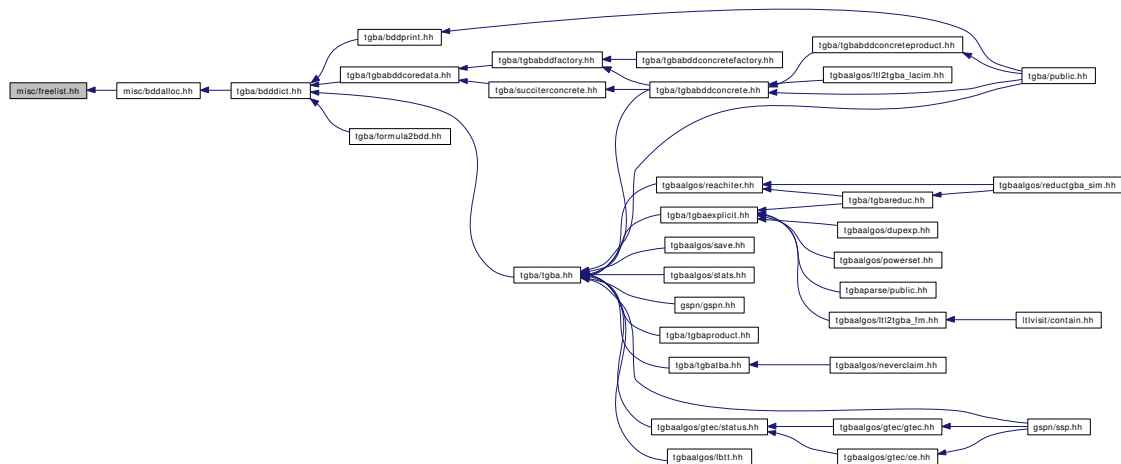
```
#include <list>
```

```
#include <utility>
```

Include dependency graph for freelist.hh:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace **spot**

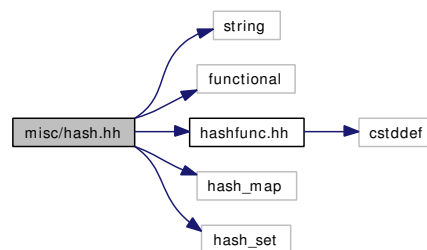
## Classes

- class `spot::free_list`  
*Manage list of free integers.*

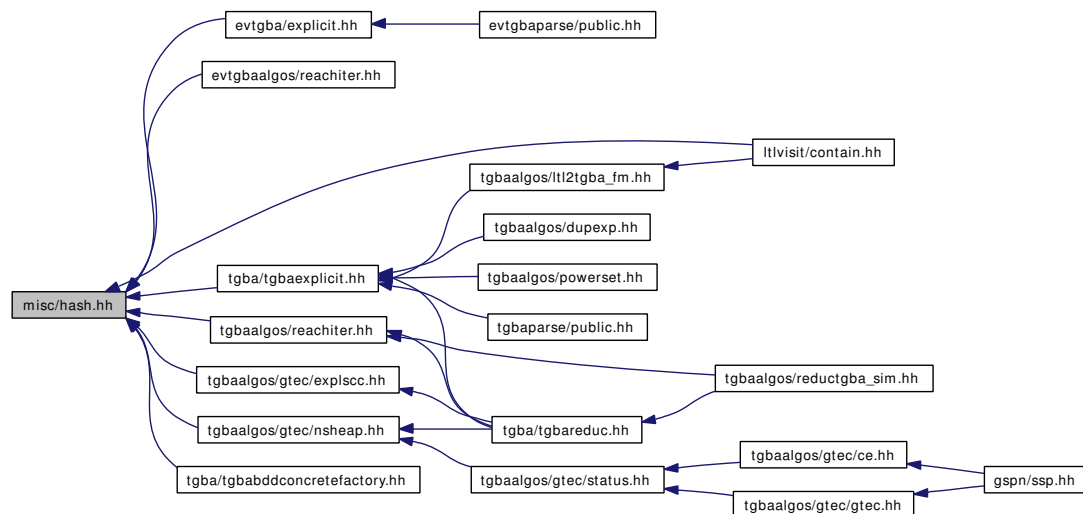
### 13.63 misc/hash.hh File Reference

```
#include <string>
#include <functional>
#include "hashfunc.hh"
#include <hash_map>
#include <hash_set>
```

Include dependency graph for hash.hh:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace **spot**

## Classes

- struct `spot::ptr_hash< T >`  
*A hash function for pointers.*
- struct `spot::string_hash`  
*A hash function for strings.*

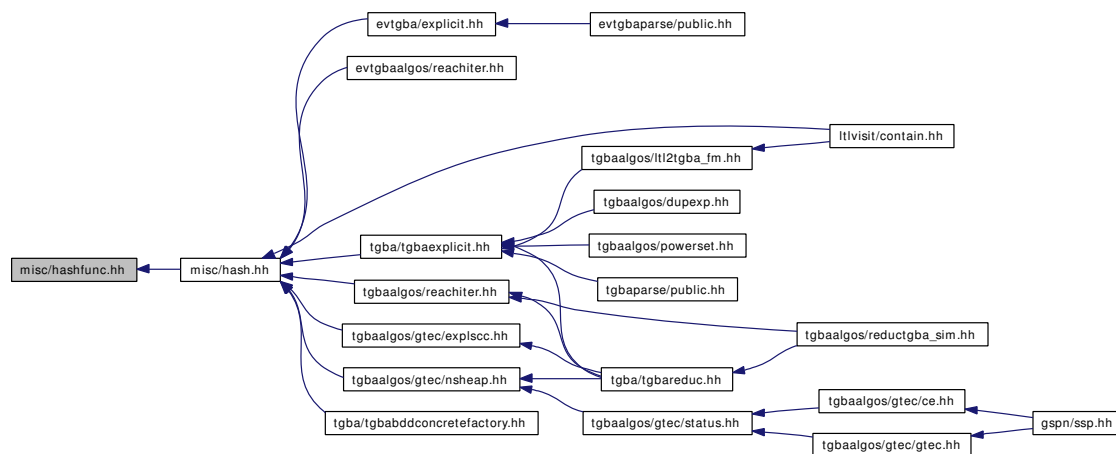
## 13.64 misc/hashfunc.hh File Reference

```
#include <cstdint>
```

Include dependency graph for hashfunc.hh:



This graph shows which files directly or indirectly include this file:



### Namespaces

- namespace [spot](#)

### Functions

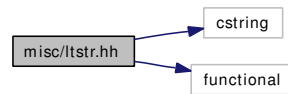
- `size_t spot::wang32\_hash (size_t key)`  
*Thomas Wang's 32 bit hash function.*
- `size_t spot::knuth32\_hash (size_t key)`  
*Knuth's Multiplicative hash function.*

## 13.65 misc/ltstr.hh File Reference

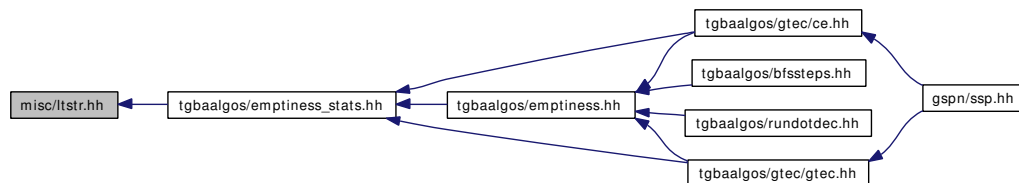
```
#include <cstring>
```

```
#include <functional>
```

Include dependency graph for ltstr.hh:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [spot](#)

## Classes

- struct [spot::char\\_ptr\\_less\\_than](#)  
*Strict Weak Ordering for char\*.*

## 13.66 misc/memusage.hh File Reference

### Namespaces

- namespace [spot](#)

### Functions

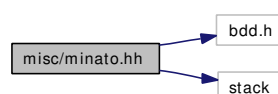
- int [spot::memusage](#) ()  
*Total number of pages in use by the program.*

## 13.67 misc/minato.hh File Reference

```
#include <bdd.h>
```

```
#include <stack>
```

Include dependency graph for minato.hh:



### Namespaces

- namespace [spot](#)

### Classes

- class [spot::minato\\_isop](#)  
*Generate an irredundant sum-of-products (ISOP) form of a BDD function.*
- struct [spot::minato\\_isop::local\\_vars](#)  
*Internal variables for [minato\\_isop](#).*

## 13.68 misc/modgray.hh File Reference

### Namespaces

- namespace [spot](#)

### Classes

- class [spot::loopless\\_modular\\_mixed\\_radix\\_gray\\_code](#)  
*Loopless modular mixed radix Gray code iteration.*

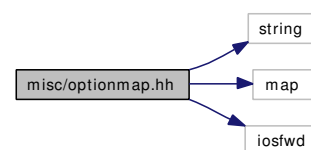
## 13.69 misc/optionmap.hh File Reference

```
#include <string>
```

```
#include <map>
```

```
#include <iosfwd>
```

Include dependency graph for optionmap.hh:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [spot](#)

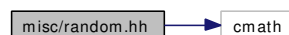
## Classes

- class [spot::option\\_map](#)  
*Manage a map of options.*

## 13.70 misc/random.hh File Reference

```
#include <cmath>
```

Include dependency graph for random.hh:



## Namespaces

- namespace [spot](#)

## Classes

- class [spot::barand< gen >](#)  
*Compute pseudo-random integer value between 0 and n included, following a binomial distribution for probability p.*

## Functions

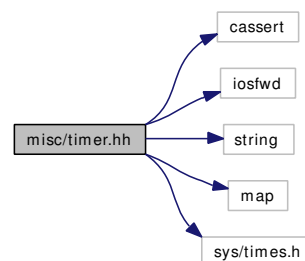
- void [spot::srand](#) (unsigned int seed)  
*Reset the seed of the pseudo-random number generator.*

- int [spot::rrand](#) (int min, int max)  
*Compute a pseudo-random integer value between min and max included.*
- int [spot::mrnd](#) (int max)  
*Compute a pseudo-random integer value between 0 and max-1 included.*
- double [spot::drand](#) ()  
*Compute a pseudo-random double value between 0.0 and 1.0 (1.0 excluded).*
- double [spot::nrnd](#) ()  
*Compute a pseudo-random double value following a standard normal distribution. (Odeh & Evans).*
- double [spot::bmrand](#) ()  
*Compute a pseudo-random double value following a standard normal distribution. (Box-Muller).*
- int [spot::prand](#) (double p)  
*Return a pseudo-random positive integer value following a Poisson distribution with parameter p.*

## 13.71 misc/timer.hh File Reference

```
#include <cassert>
#include <iosfwd>
#include <string>
#include <map>
#include <sys/times.h>
```

Include dependency graph for timer.hh:



### Namespaces

- namespace [spot](#)

### Classes

- struct [spot::time\\_info](#)  
*A structure to record elapsed time in clock ticks.*
- class [spot::timer](#)

*A timekeeper that accumulate interval of time.*

- class `spot::timer_map`

*A map of `timer`, where each `timer` has a name.*

## 13.72 misc/version.hh File Reference

### Namespaces

- namespace `spot`

### Functions

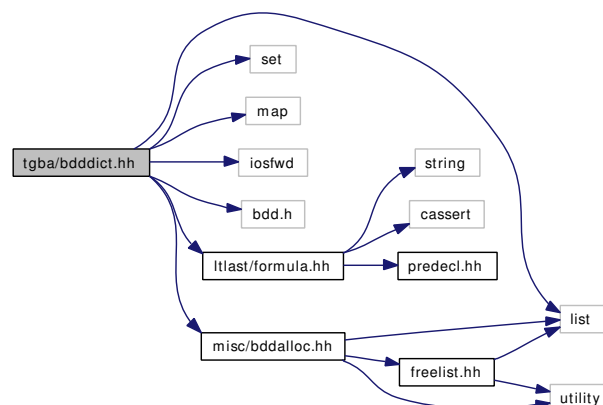
- `const char * spot::version ()`

*Return Spot's version.*

## 13.73 tgba/bdddict.hh File Reference

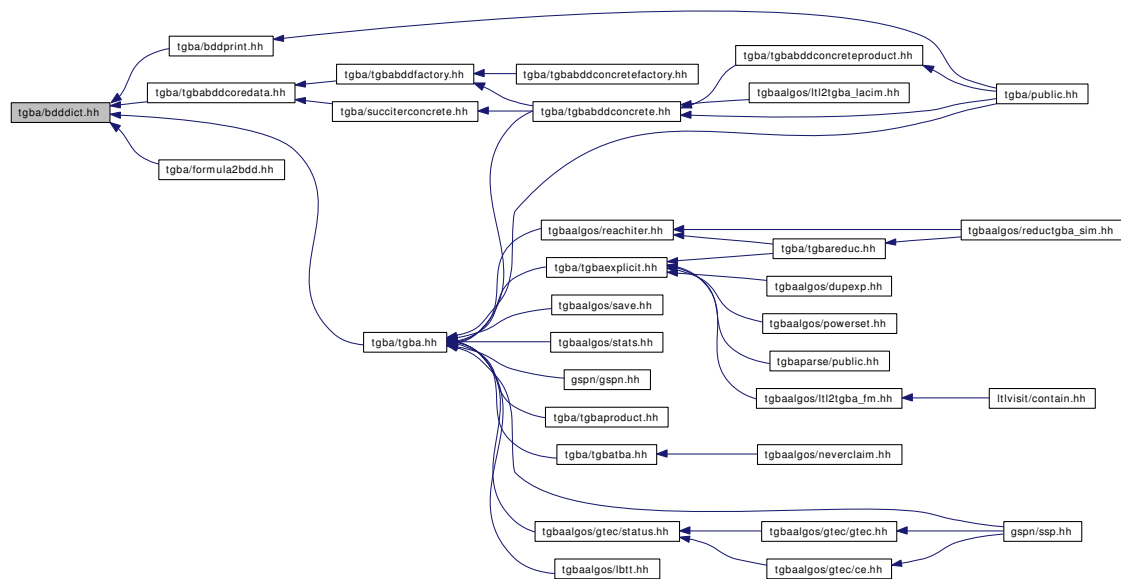
```
#include <list>
#include <set>
#include <map>
#include <iosfwd>
#include <bdd.h>
#include "ltlast/formula.hh"
#include "misc/bddalloc.hh"
```

Include dependency graph for bdddict.hh:





This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace **spot**

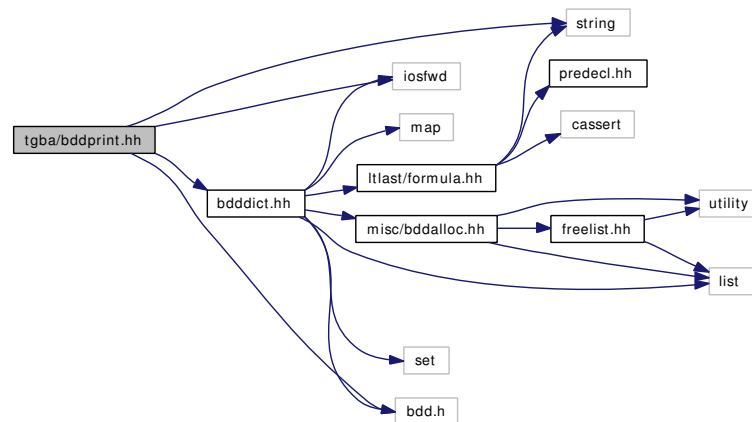
## Classes

- class `spot::bdd_dict`
- class `spot::bdd_dict::anon_free_list`

### 13.74 tgba/bddprint.hh File Reference

```
#include <string>
#include <iosfwd>
#include "bdddict.hh"
#include <bdd.h>
```

Include dependency graph for bddprint.hh:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [spot](#)

## Functions

- `std::ostream & spot::bdd\_print\_sat (std::ostream &os, const bdd_dict *dict, bdd b)`  
*Print a BDD as a list of literals.*
- `std::string spot::bdd\_format\_sat (const bdd_dict *dict, bdd b)`  
*Format a BDD as a list of literals.*
- `std::ostream & spot::bdd\_print\_acc (std::ostream &os, const bdd_dict *dict, bdd b)`  
*Print a BDD as a list of acceptance conditions.*
- `std::ostream & spot::bdd\_print\_accset (std::ostream &os, const bdd_dict *dict, bdd b)`  
*Print a BDD as a set of acceptance conditions.*
- `std::string spot::bdd\_format\_accset (const bdd_dict *dict, bdd b)`  
*Format a BDD as a set of acceptance conditions.*
- `std::ostream & spot::bdd\_print\_set (std::ostream &os, const bdd_dict *dict, bdd b)`  
*Print a BDD as a set.*
- `std::string spot::bdd\_format\_set (const bdd_dict *dict, bdd b)`  
*Format a BDD as a set.*
- `std::ostream & spot::bdd\_print\_formula (std::ostream &os, const bdd_dict *dict, bdd b)`

*Print a BDD as a formula.*

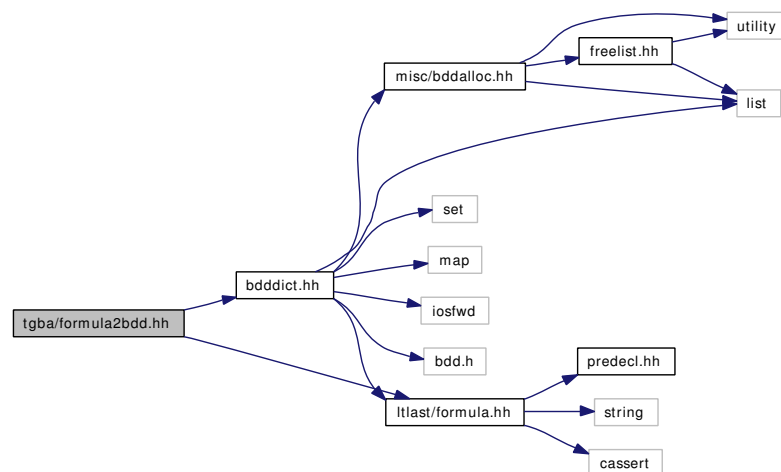
- std::string [spot::bdd\\_format\\_formula](#) (const bdd\_dict \*dict, bdd b)  
*Format a BDD as a formula.*
- std::ostream & [spot::bdd\\_print\\_dot](#) (std::ostream &os, const bdd\_dict \*dict, bdd b)  
*Print a BDD as a diagram in doty format.*
- std::ostream & [spot::bdd\\_print\\_table](#) (std::ostream &os, const bdd\_dict \*dict, bdd b)  
*Print a BDD as a table.*

## 13.75 tgba/formula2bdd.hh File Reference

```
#include "bdddict.hh"
```

```
#include "ltlast/formula.hh"
```

Include dependency graph for formula2bdd.hh:



### Namespaces

- namespace [spot](#)

### Functions

- bdd [spot::formula\\_to\\_bdd](#) (const ltl::formula \*f, bdd\_dict \*d, void \*for\_me)
- const ltl::formula \* [spot::bdd\\_to\\_formula](#) (bdd f, const bdd\_dict \*d)

## 13.76 tgba/state.hh File Reference

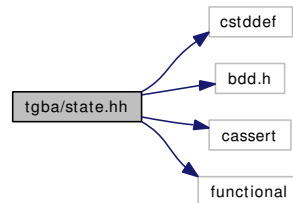
```
#include <cstdint>
```

```
#include <bdd.h>
```

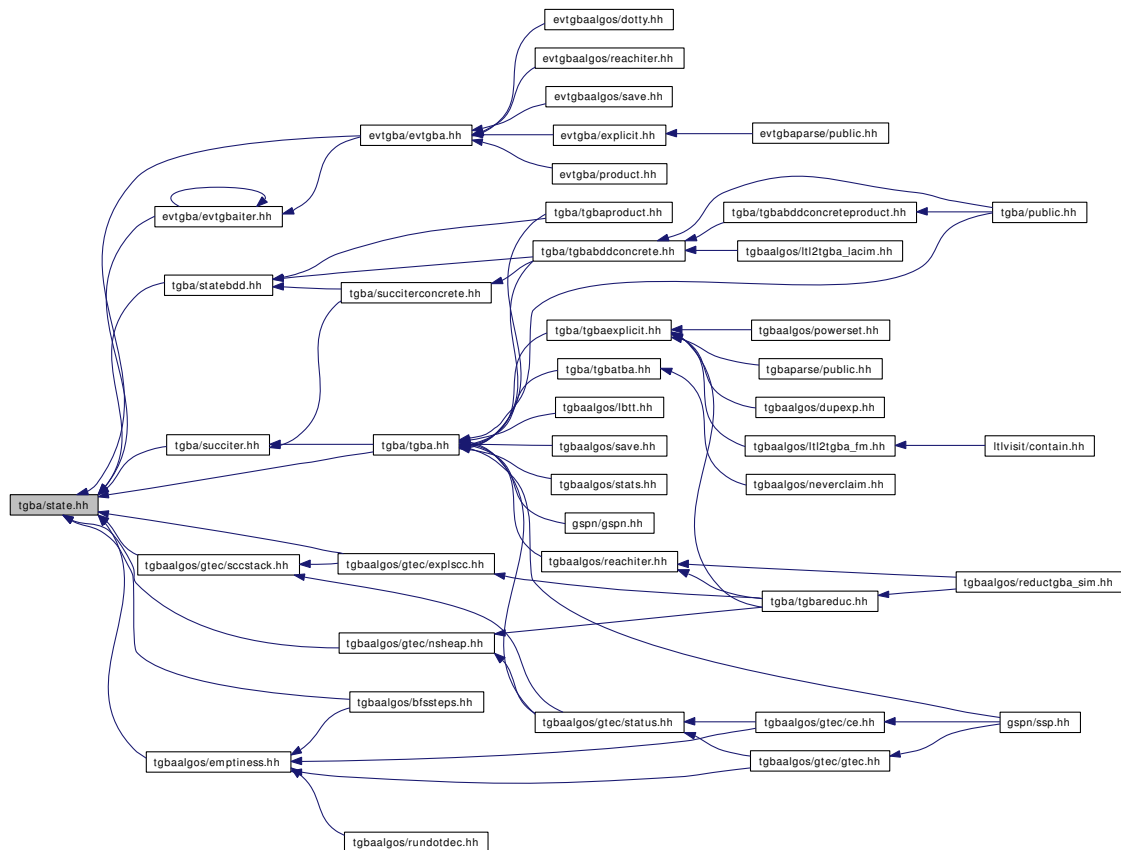
```
#include <cassert>
```

```
#include <functional>
```

Include dependency graph for state.hh:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace **spot**

## Classes

- class `spot::state`  
*Abstract class for states.*

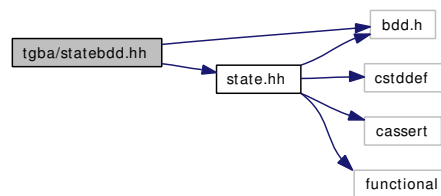
- struct [spot::state\\_ptr\\_less\\_than](#)  
*Strict Weak Ordering for state\*.*
- struct [spot::state\\_ptr\\_equal](#)  
*An Equivalence Relation for state\*.*
- struct [spot::state\\_ptr\\_hash](#)  
*Hash Function for state\*.*

## 13.77 tgba/statebdd.hh File Reference

```
#include <bdd.h>
```

```
#include "state.hh"
```

Include dependency graph for statebdd.hh:



This graph shows which files directly or indirectly include this file:



### Namespaces

- namespace [spot](#)

### Classes

- class [spot::state\\_bdd](#)

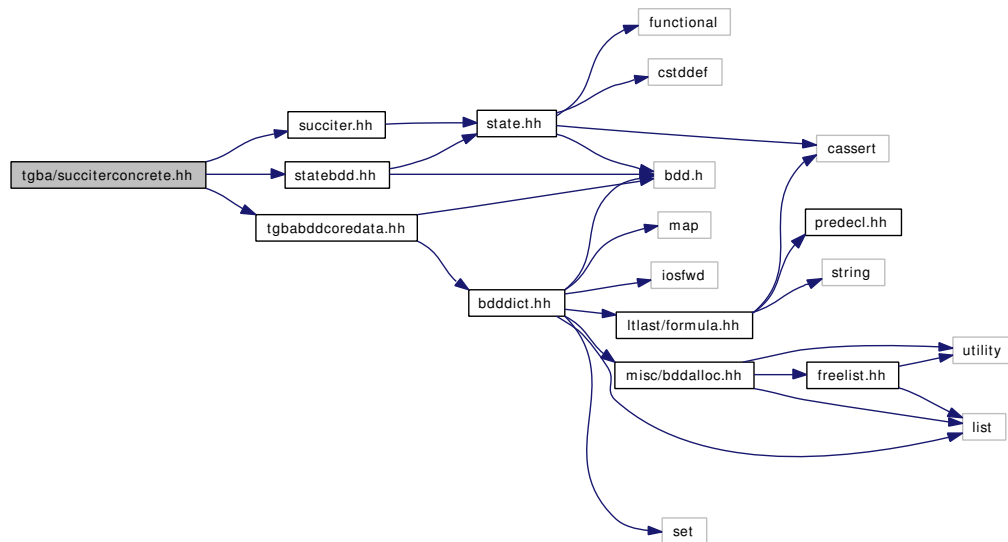
## 13.78 tgba/succiter.hh File Reference

```
#include "state.hh"
```

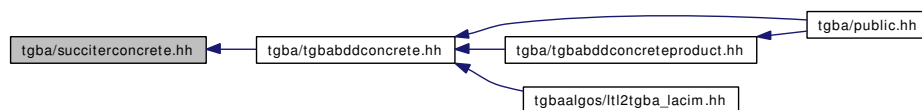


```
#include "succiter.hh"
#include "tgbabddcoredata.hh"
```

Include dependency graph for succiterconcrete.hh:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [spot](#)

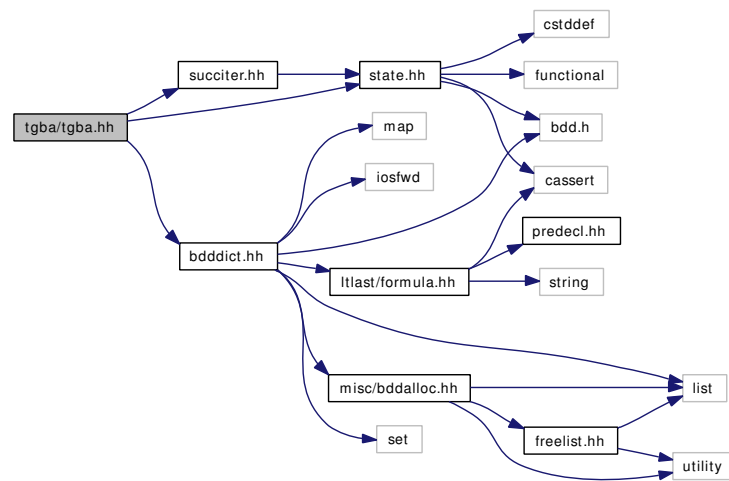
## Classes

- class [spot::tgba\\_succ\\_iterator\\_concrete](#)

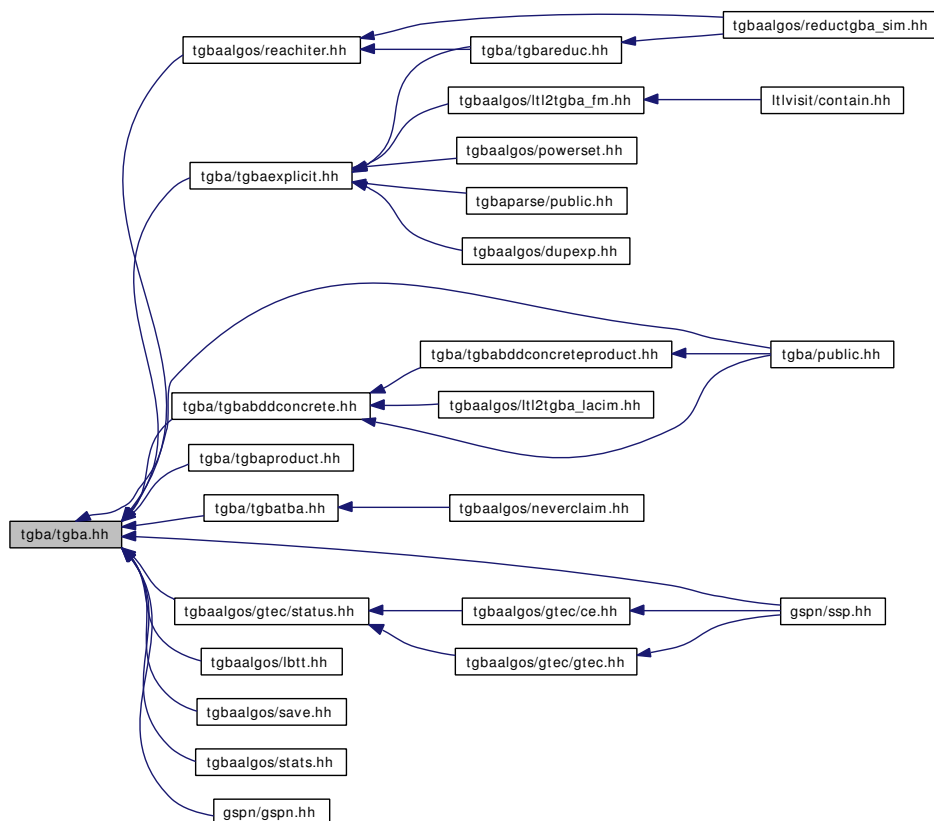
## 13.80 tgba/tgba.hh File Reference

```
#include "state.hh"
#include "succiter.hh"
#include "bdddict.hh"
```

Include dependency graph for tgba.hh:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace **spot**



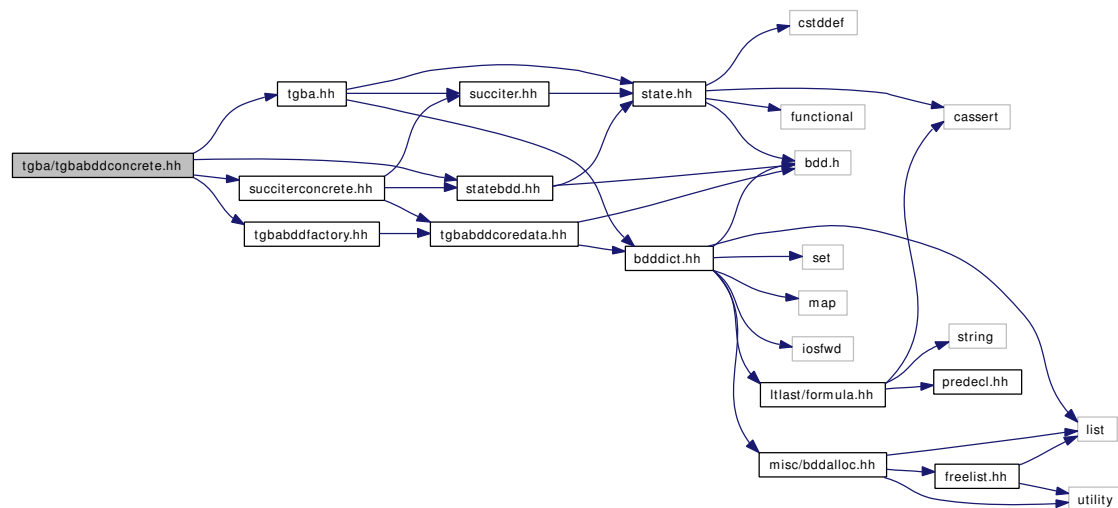
## Classes

- class [spot::tgba](#)  
*A Transition-based Generalized Büchi Automaton.*

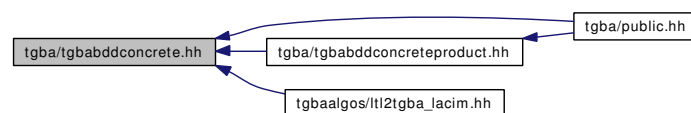
## 13.81 tgba/tgbabddconcrete.hh File Reference

```
#include "tgba.hh"
#include "statebdd.hh"
#include "tgbabddfactory.hh"
#include "succiterconcrete.hh"
```

Include dependency graph for tgbabddconcrete.hh:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [spot](#)

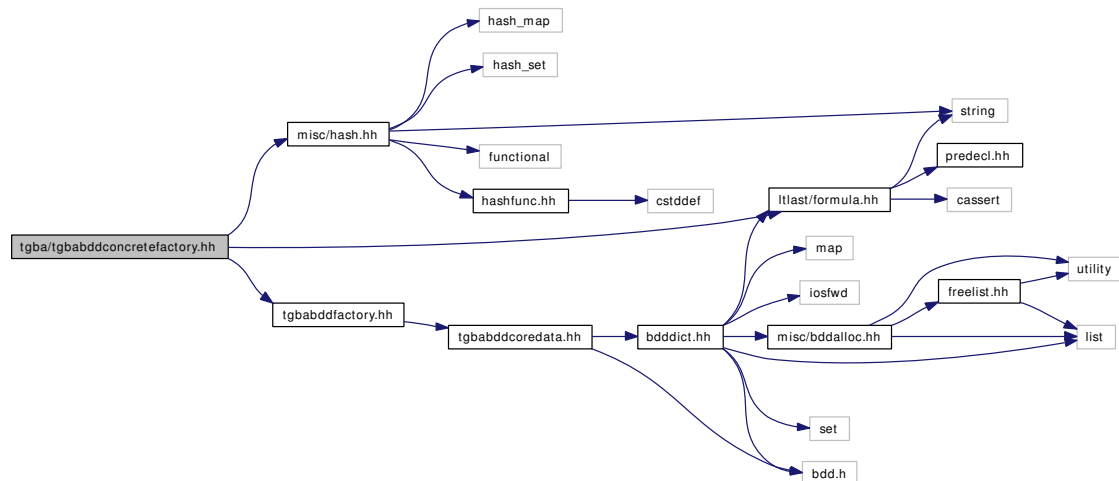
## Classes

- class [spot::tgba\\_bdd\\_concrete](#)  
*A concrete [spot::tgba](#) implemented using BDDs.*

## 13.82 tgba/tgbabddconcretefactory.hh File Reference

```
#include "misc/hash.hh"
#include "ltlast/formula.hh"
#include "tgbabddfactory.hh"
```

Include dependency graph for tgbabddconcretefactory.hh:



### Namespaces

- namespace [spot](#)

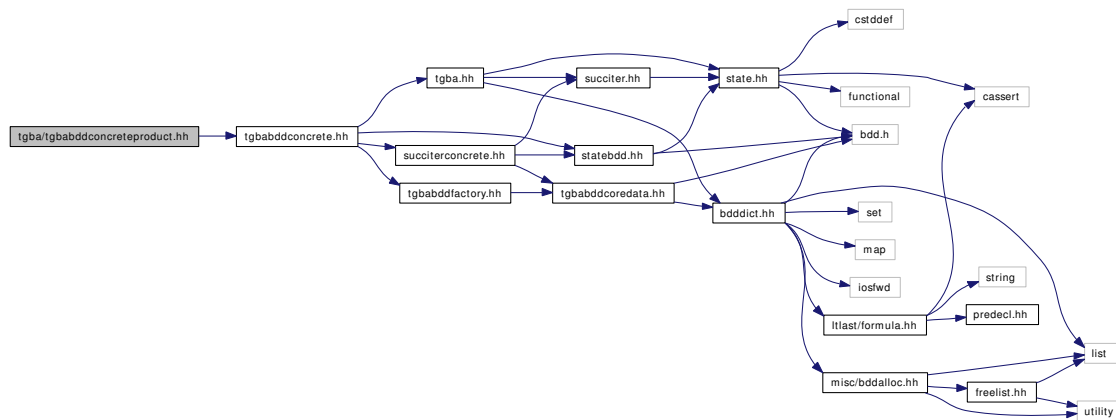
### Classes

- class [spot::tgba\\_bdd\\_concrete\\_factory](#)  
Helper class to build a [spot::tgba\\_bdd\\_concrete](#) object.

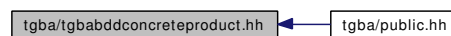
## 13.83 tgba/tgbabddconcreteproduct.hh File Reference

```
#include "tgbabddconcrete.hh"
```

Include dependency graph for tgbabddconcreteproduct.hh:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [spot](#)

## Functions

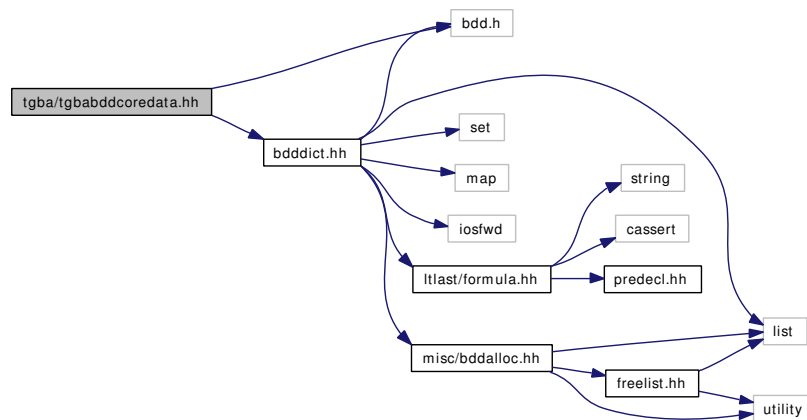
- `tgba_bdd_concrete * spot::product (const tgba_bdd_concrete *left, const tgba_bdd_concrete *right)`

*Multiplies two [spot::tgba\\_bdd\\_concrete](#) automata.*

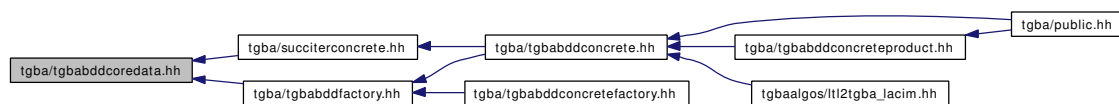
## 13.84 tgba/tgbabddcoredata.hh File Reference

```
#include <bdd.h>
#include "bdddict.hh"
```

Include dependency graph for tgbabddcoredata.hh:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [spot](#)

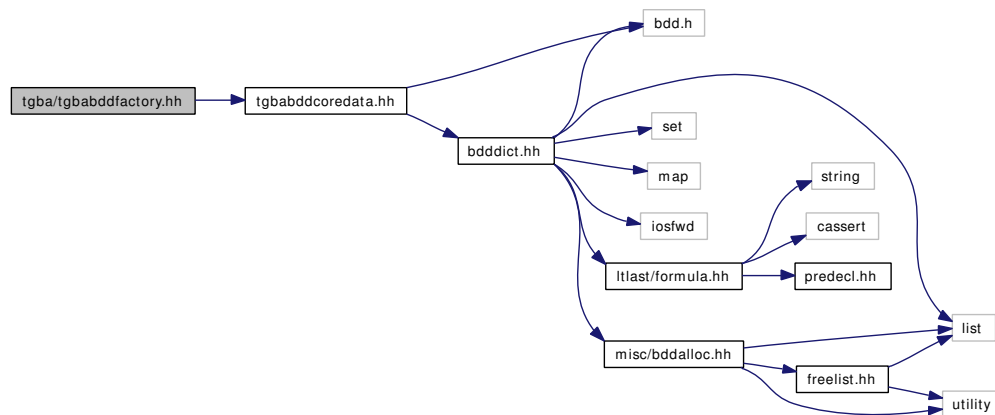
## Classes

- struct [spot::tgba\\_bdd\\_core\\_data](#)  
*Core data for a TGBA encoded using BDDs.*

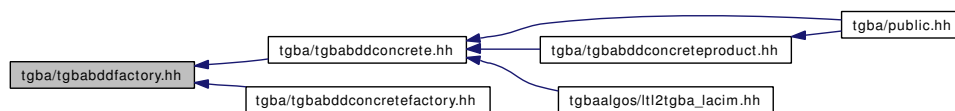
## 13.85 tgba/tgbabddfactory.hh File Reference

```
#include "tgbabddcoredata.hh"
```

Include dependency graph for tgbabddfactory.hh:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [spot](#)

## Classes

- class [spot::tgba\\_bdd\\_factory](#)  
*Abstract class for [spot::tgba\\_bdd\\_concrete](#) factories.*

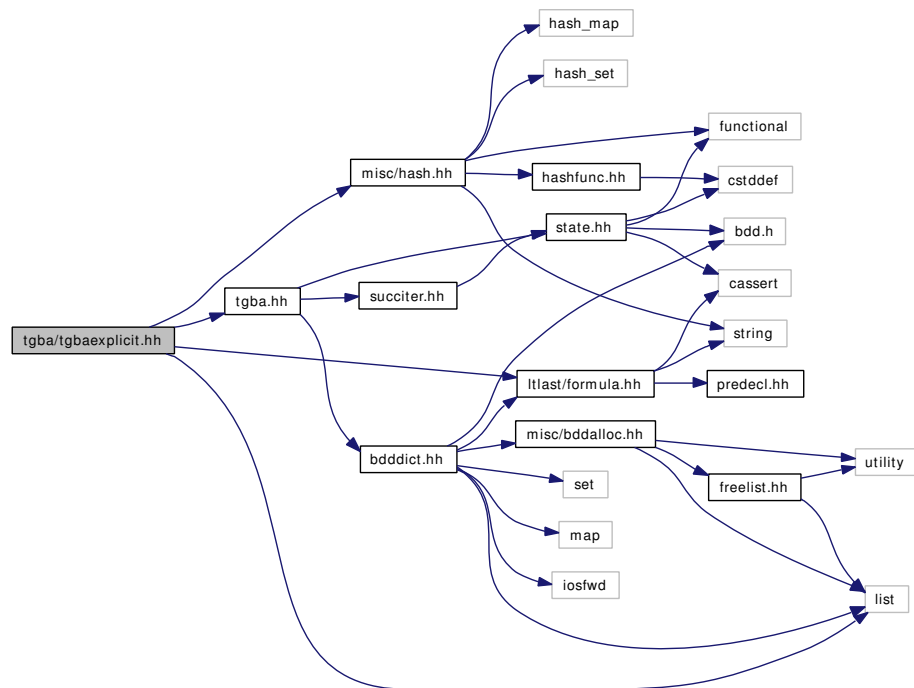
## 13.86 tgba/tgbaexplicit.hh File Reference

```

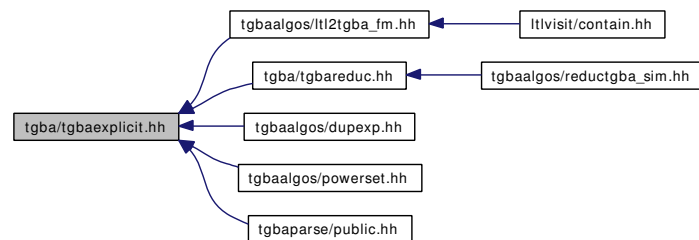
#include "misc/hash.hh"
#include <list>
#include "tgba.hh"
#include "ltlast/formula.hh"

```

Include dependency graph for tgbaexplicit.hh:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [spot](#)

## Classes

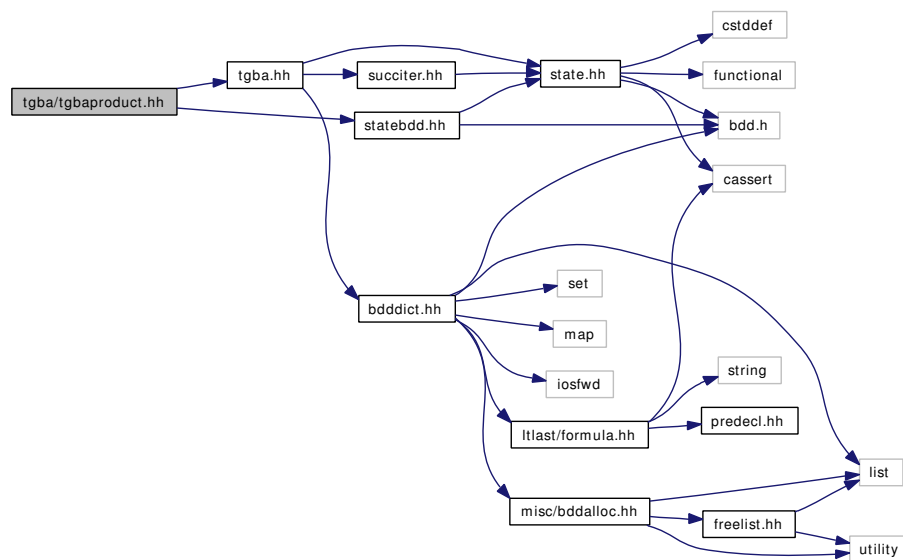
- class [spot::tgba\\_explicit](#)
- struct [spot::tgba\\_explicit::transition](#)  
*Explicit transitions (used by [spot::tgba\\_explicit](#)).*
- class [spot::state\\_explicit](#)
- class [spot::tgba\\_explicit\\_succ\\_iterator](#)

## 13.87 tgba/tgbaproduct.hh File Reference

```
#include "tgba.hh"
```

```
#include "statebdd.hh"
```

Include dependency graph for tgbaproduct.hh:



### Namespaces

- namespace [spot](#)

### Classes

- class [spot::state\\_product](#)  
*A state for [spot::tgba\\_product](#).*
- class [spot::tgba\\_succ\\_iterator\\_product](#)  
*Iterate over the successors of a product computed on the fly.*
- class [spot::tgba\\_product](#)  
*A lazy product. (States are computed on the fly.).*

## 13.88 tgba/tgbareduc.hh File Reference

```
#include "tgbaexplicit.hh"
```

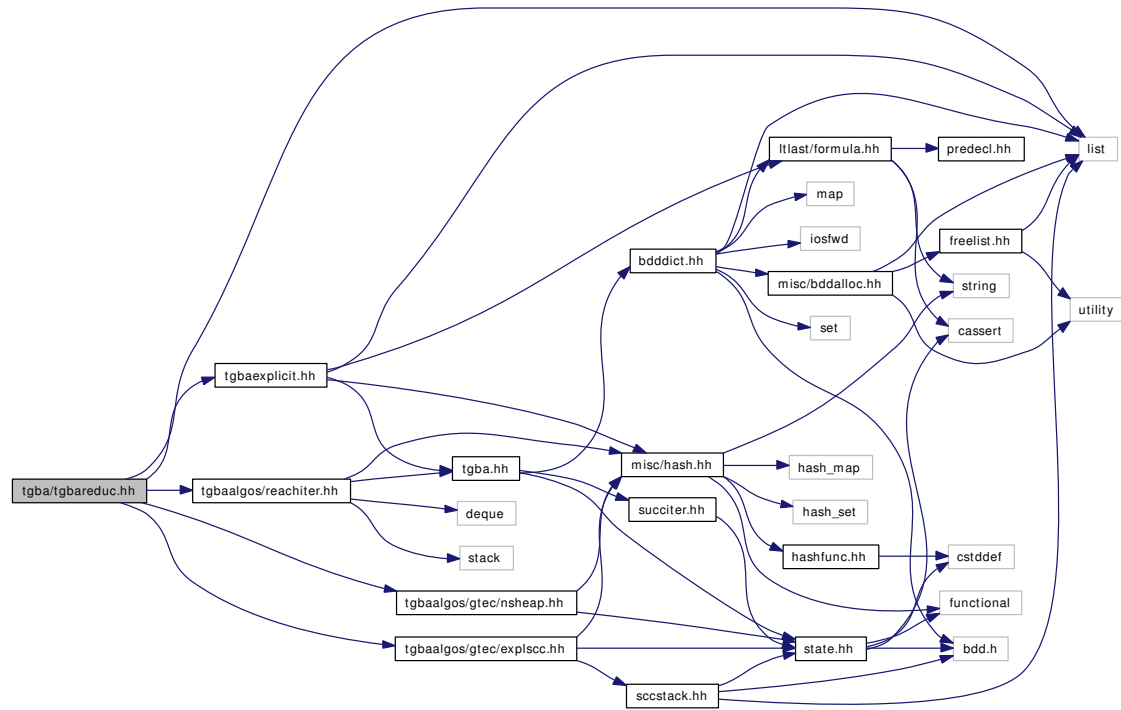
```
#include "tgbaalgos/reachiter.hh"
```

```
#include "tgbaalgos/gtec/explsccl.hh"
```

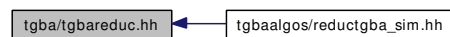
```
#include "tgbaalgos/gtec/nsheap.hh"
```

```
#include <list>
```

Include dependency graph for tgba/tgbareduc.hh:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [spot](#)

## Classes

- class [spot::direct\\_simulation\\_relation](#)
- class [spot::delayed\\_simulation\\_relation](#)
- class [spot::tgba\\_reduc](#)

## Typedefs

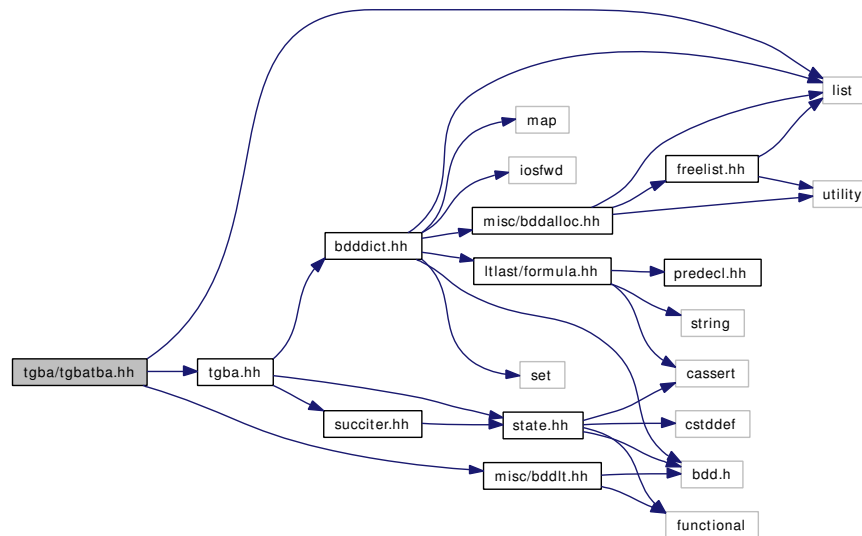
- typedef std::pair< const [spot::state](#) \*, const [spot::state](#) \* > [spot::state\\_couple](#)
- typedef std::vector< state\_couple \* > [spot::simulation\\_relation](#)



## 13.89 tgba/tgbatba.hh File Reference

```
#include <list>
#include "tgba.hh"
#include "misc/bddlt.hh"
```

Include dependency graph for tgbatba.hh:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [spot](#)

## Classes

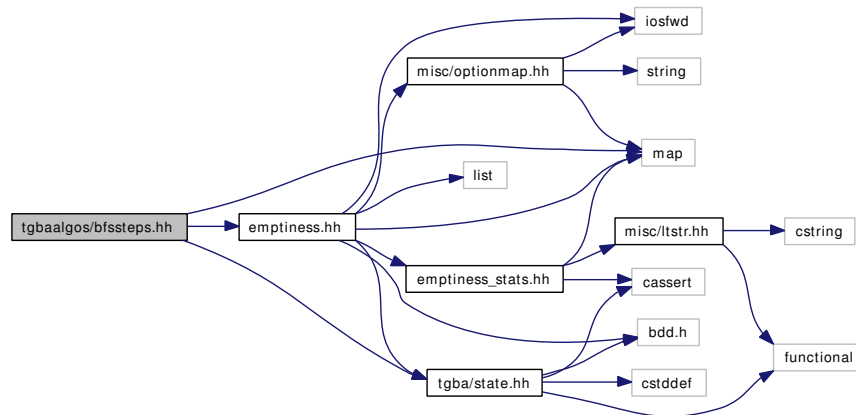
- class [spot::tgba\\_tba\\_proxy](#)  
*Degeneralize a [spot::tgba](#) on the fly, producing a TBA.*
- class [spot::tgba\\_sba\\_proxy](#)  
*Degeneralize a [spot::tgba](#) on the fly, producing an SBA.*

## 13.90 tgbaalgos/bfssteps.hh File Reference

```
#include <map>
#include "tgba/state.hh"
```

```
#include "emptiness.hh"
```

Include dependency graph for bfssteps.hh:



## Namespaces

- namespace [spot](#)

## Classes

- class [spot::bfs\\_steps](#)

*Make a BFS in a [spot::tgba](#) to compute a [tgba\\_run::steps](#).*

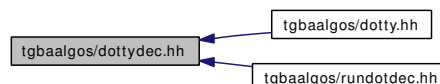
## 13.91 tgbaalgos/dottydec.hh File Reference

```
#include <string>
```

Include dependency graph for dottydec.hh:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [spot](#)

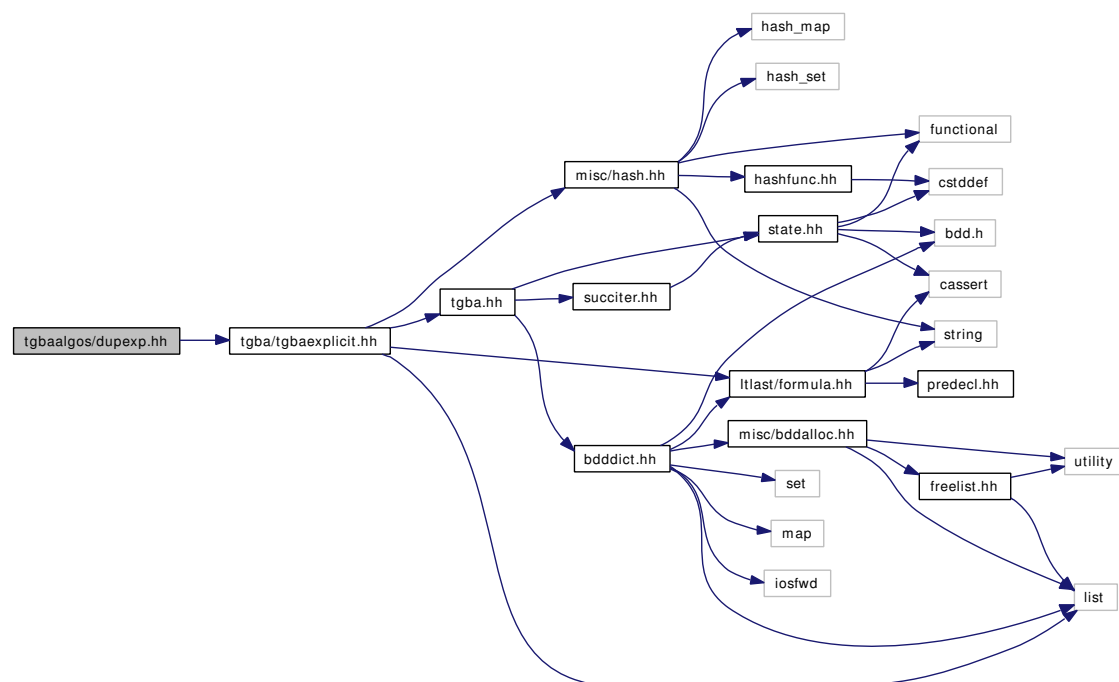
## Classes

- class [spot::dotty\\_decorator](#)  
Choose [state](#) and link styles for [spot::dotty\\_reachable](#).

## 13.92 tgbaalgorithms/dupep.h File Reference

```
#include "tgba/tgbaexplicit.hh"
```

Include dependency graph for dupep.h:



## Namespaces

- namespace [spot](#)

## Functions

- tgba\_explicit \* [spot::tgba\\_dupexp\\_bfs](#) (const tgba \*aut)  
Build an explicit automata from all states of aut, numbering states in bread first order as they are processed.
- tgba\_explicit \* [spot::tgba\\_dupexp\\_dfs](#) (const tgba \*aut)  
Build an explicit automata from all states of aut, numbering states in depth first order as they are processed.

## 13.93 tgbaalgorithms/emptiness.h File Reference

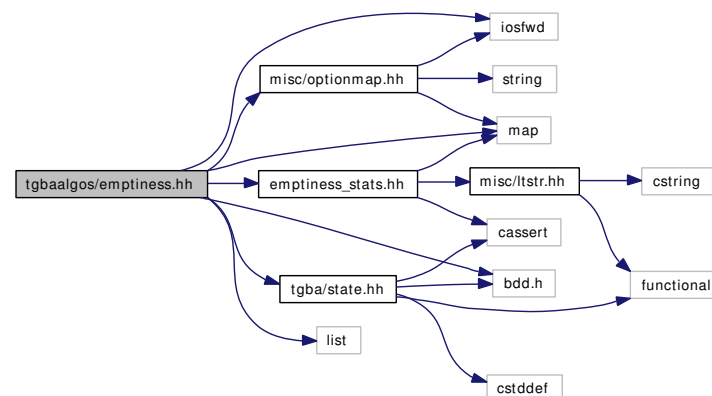
```
#include <map>
```

```

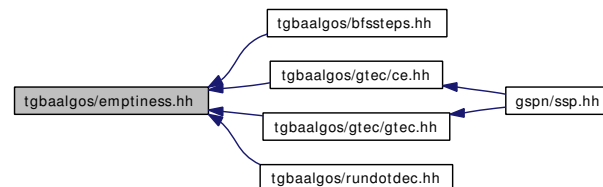
#include <list>
#include <iosfwd>
#include <bdd.h>
#include "misc/optionmap.hh"
#include "tgba/state.hh"
#include "emptiness_stats.hh"

```

Include dependency graph for emptiness.hh:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [spot](#)

## Classes

- class [spot::emptiness\\_check\\_result](#)  
*The result of an emptiness check.*
- class [spot::emptiness\\_check](#)  
*Common interface to emptiness check algorithms.*
- class [spot::emptiness\\_check\\_instantiator](#)
- struct [spot::tgba\\_run](#)  
*An accepted run, for a [tgba](#).*
- struct [spot::tgba\\_run::step](#)

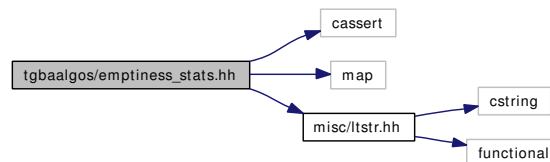
## Functions

- `std::ostream & spot::print_tgba_run` (`std::ostream &os`, `const tgba *a`, `const tgba_run *run`)  
*Display a `tgba_run`.*
- `tgba * spot::tgba_run_to_tgba` (`const tgba *a`, `const tgba_run *run`)  
*Return an `explicit_tgba` corresponding to run (i.e. comparable states are merged).*

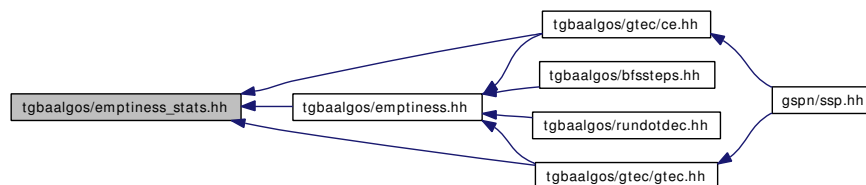
## 13.94 tgbaalgos/emptiness\_stats.hh File Reference

```
#include <cassert>
#include <map>
#include "misc/ltstr.hh"
```

Include dependency graph for `emptiness_stats.hh`:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace `spot`

## Classes

- struct `spot::unsigned_statistics`
- class `spot::unsigned_statistics_copy`  
*comparable statistics*
- class `spot::ec_statistics`  
*Emptiness-check statistics.*
- class `spot::ars_statistics`  
*Accepting Run Search statistics.*

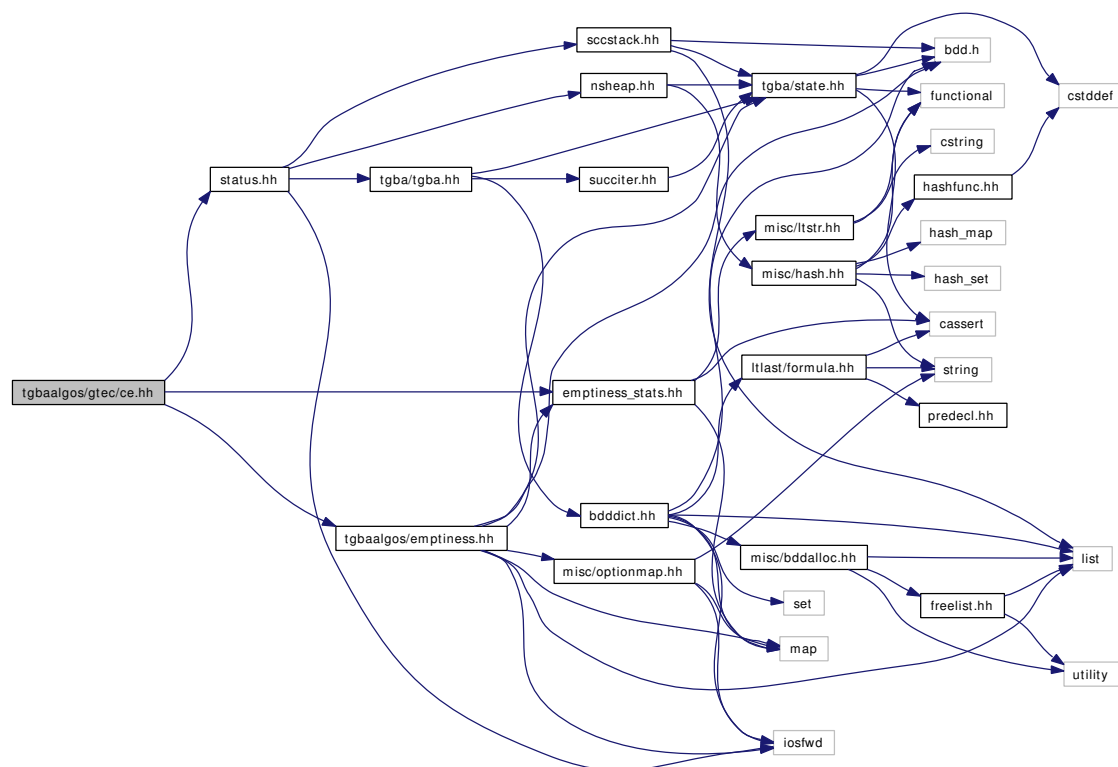
- class [spot::acss\\_statistics](#)

*Accepting Cycle Search Space statistics.*

## 13.95 tgbaalgos/gtec/ce.hh File Reference

```
#include "status.hh"
#include "tgbaalgos/emptiness.hh"
#include "tgbaalgos/emptiness_stats.hh"
```

Include dependency graph for ce.hh:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [spot](#)

## Classes

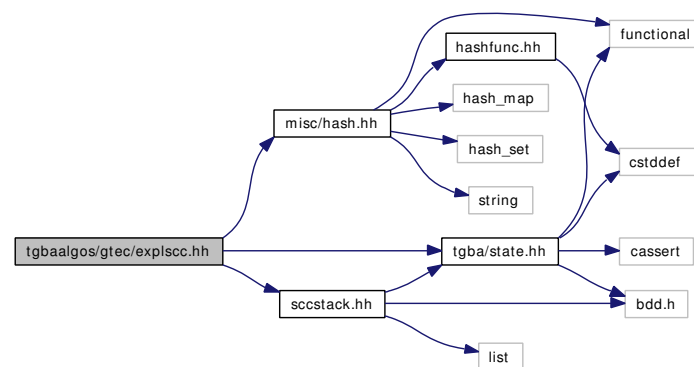
- class [spot::couvreur99\\_check\\_result](#)

Compute a counter example from a [spot::couvreur99\\_check\\_status](#).

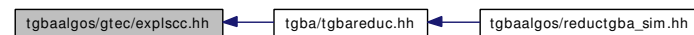
## 13.96 tgbaalgos/gtec/explsc.h File Reference

```
#include "misc/hash.hh"
#include "tgba/state.hh"
#include "sccstack.hh"
```

Include dependency graph for explsc.h:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [spot](#)

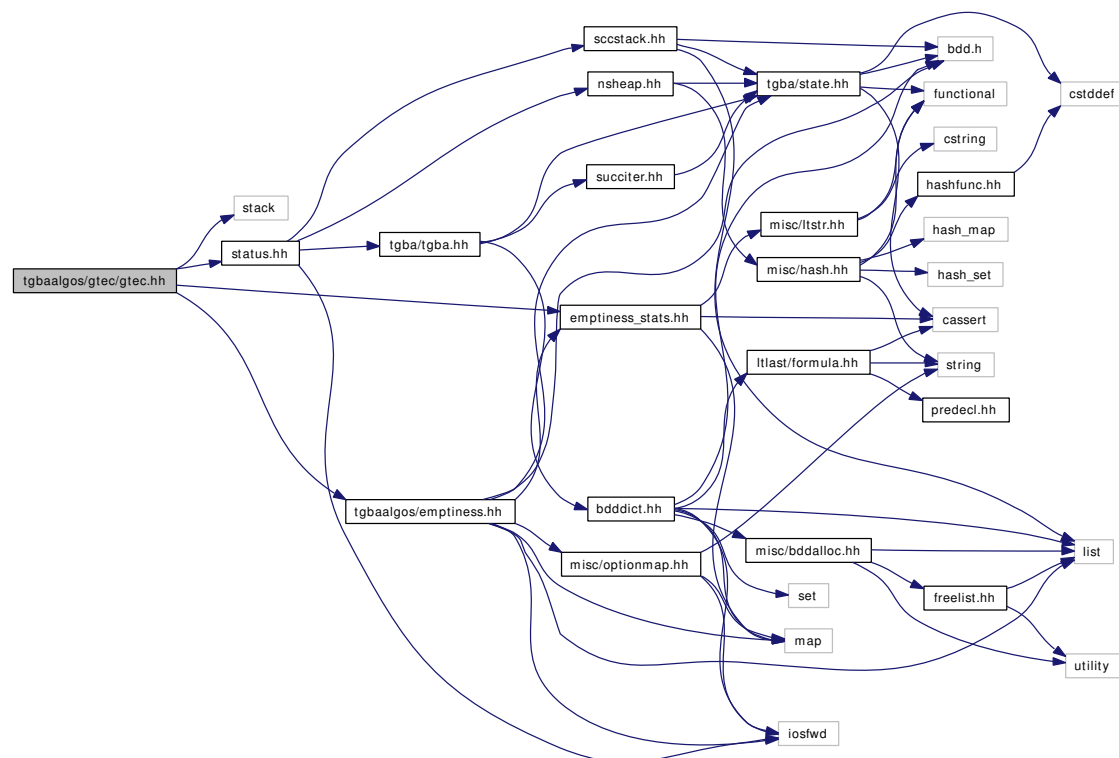
## Classes

- class [spot::explicit\\_connected\\_component](#)  
An SCC storing all its states explicitly.
- class [spot::connected\\_component\\_hash\\_set](#)
- class [spot::explicit\\_connected\\_component\\_factory](#)  
Abstract factory for [explicit\\_connected\\_component](#).
- class [spot::connected\\_component\\_hash\\_set\\_factory](#)  
Factory for [connected\\_component\\_hash\\_set](#).

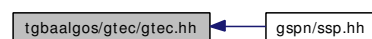
## 13.97 tgbaalgos/gtec/gtec.hh File Reference

```
#include <stack>
#include "status.hh"
#include "tgbaalgos/emptiness.hh"
#include "tgbaalgos/emptiness_stats.hh"
```

Include dependency graph for gtec.hh:



This graph shows which files directly or indirectly include this file:



### Namespaces

- namespace [spot](#)

### Classes

- class [spot::couvreur99\\_check](#)  
An implementation of the Couvreur99 emptiness-check algorithm.
- class [spot::couvreur99\\_check\\_shy](#)



A version of `spot::couvreur99_check` that tries to visit known states first.

- struct `spot::couvreur99_check_shy::successor`
- struct `spot::couvreur99_check_shy::todo_item`

## Functions

- `emptiness_check * spot::couvreur99` (const tgba \*a, option\_map options=option\_map(), const numbered\_state\_heap\_factory \*nshf=numbered\_state\_heap\_hash\_map\_factory::instance())

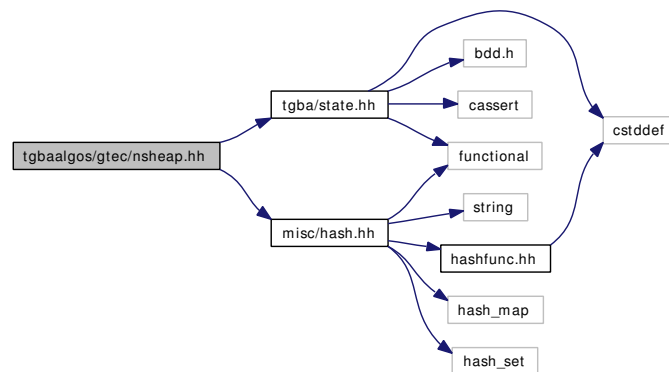
Check whether the language of an automate is empty.

## 13.98 tgbaalgos/gtec/nsheap.hh File Reference

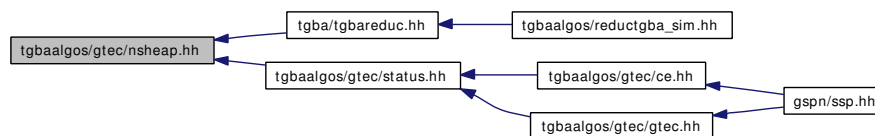
```
#include "tgba/state.hh"
```

```
#include "misc/hash.hh"
```

Include dependency graph for nsheap.hh:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace `spot`

## Classes

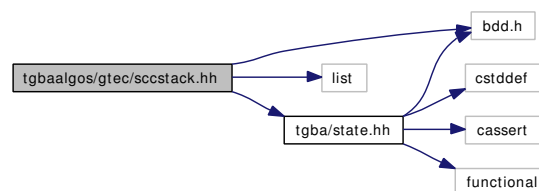
- class `spot::numbered_state_heap_const_iterator`  
Iterator on `numbered_state_heap` objects.

- class [spot::numbered\\_state\\_heap](#)  
*Keep track of a large quantity of indexed states.*
- class [spot::numbered\\_state\\_heap\\_factory](#)  
*Abstract factory for [numbered\\_state\\_heap](#).*
- class [spot::numbered\\_state\\_heap\\_hash\\_map](#)  
*A straightforward implementation of [numbered\\_state\\_heap](#) with a hash map.*
- class [spot::numbered\\_state\\_heap\\_hash\\_map\\_factory](#)  
*Factory for [numbered\\_state\\_heap\\_hash\\_map](#).*

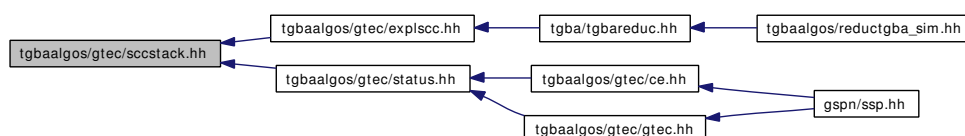
## 13.99 tgbaalgos/gtec/sccstack.hh File Reference

```
#include <bdd.h>
#include <list>
#include <tgba/state.hh>
```

Include dependency graph for sccstack.hh:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [spot](#)

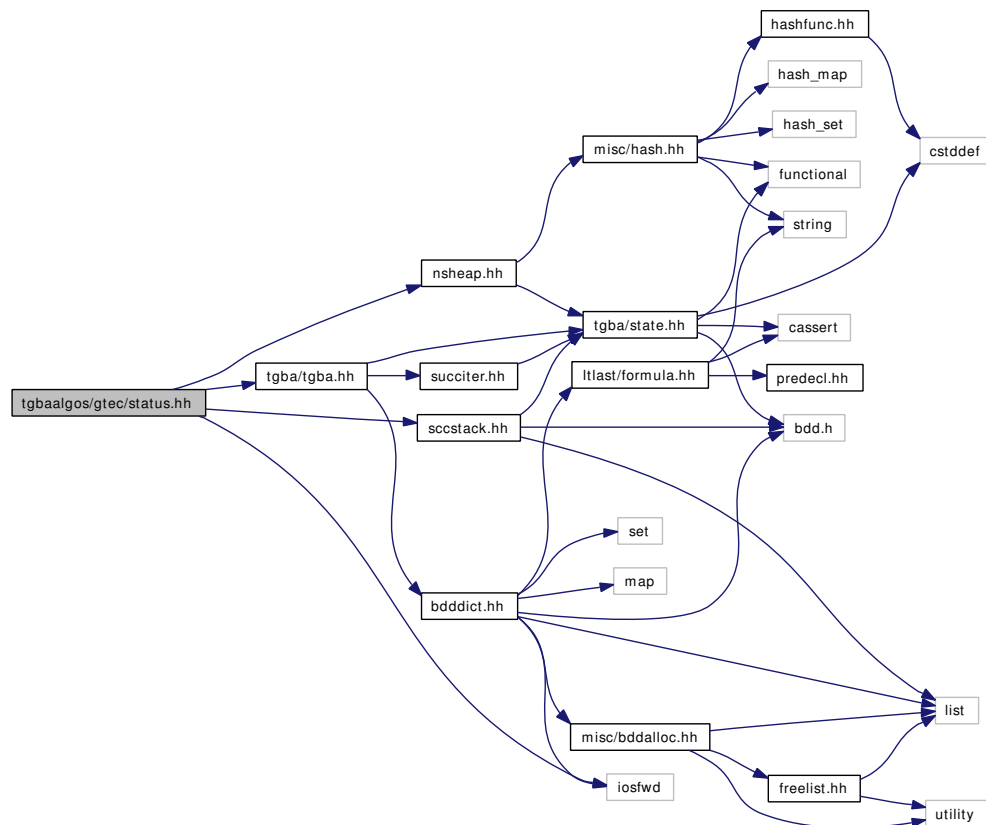
## Classes

- class [spot::scc\\_stack](#)
- struct [spot::scc\\_stack::connected\\_component](#)

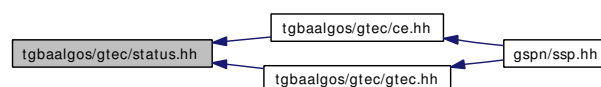
## 13.100 tgbaalgos/gtec/status.hh File Reference

```
#include "sccstack.hh"
#include "nsheap.hh"
#include "tgba/tgba.hh"
#include <iosfwd>
```

Include dependency graph for status.hh:



This graph shows which files directly or indirectly include this file:



### Namespaces

- namespace [spot](#)

### Classes

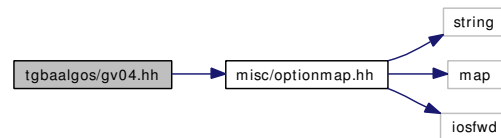
- class [spot::couvreur99\\_check\\_status](#)

*The status of the emptiness-check on success.*

### 13.101 tgbaalgos/gv04.hh File Reference

```
#include "misc/optionmap.hh"
```

Include dependency graph for gv04.hh:



#### Namespaces

- namespace [spot](#)

#### Functions

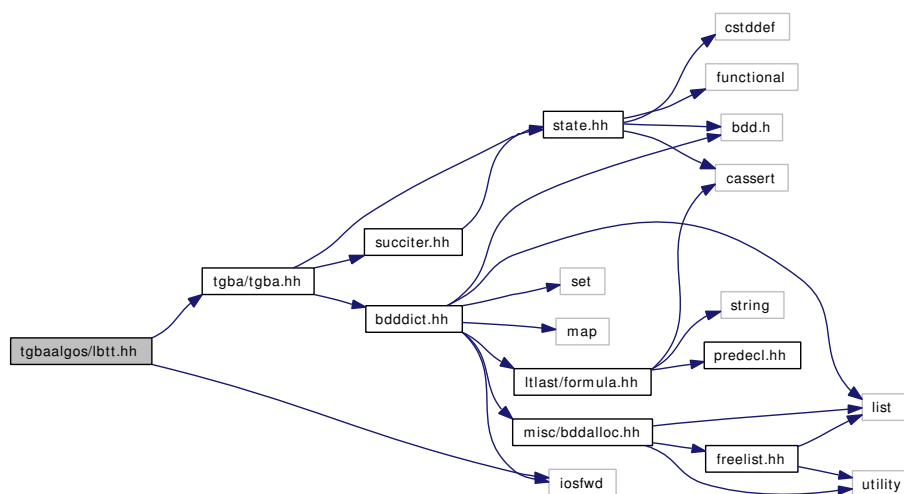
- emptiness\_check \* [spot::explicit\\_gv04\\_check](#) (const tgba \*a, option\_map o=option\_map())  
*Emptiness check based on Geldenhuys and Valmari's TACAS'04 paper.*

### 13.102 tgbaalgos/lbtt.hh File Reference

```
#include "tgba/tgba.hh"
```

```
#include <iosfwd>
```

Include dependency graph for lbtt.hh:



## Namespaces

- namespace [spot](#)

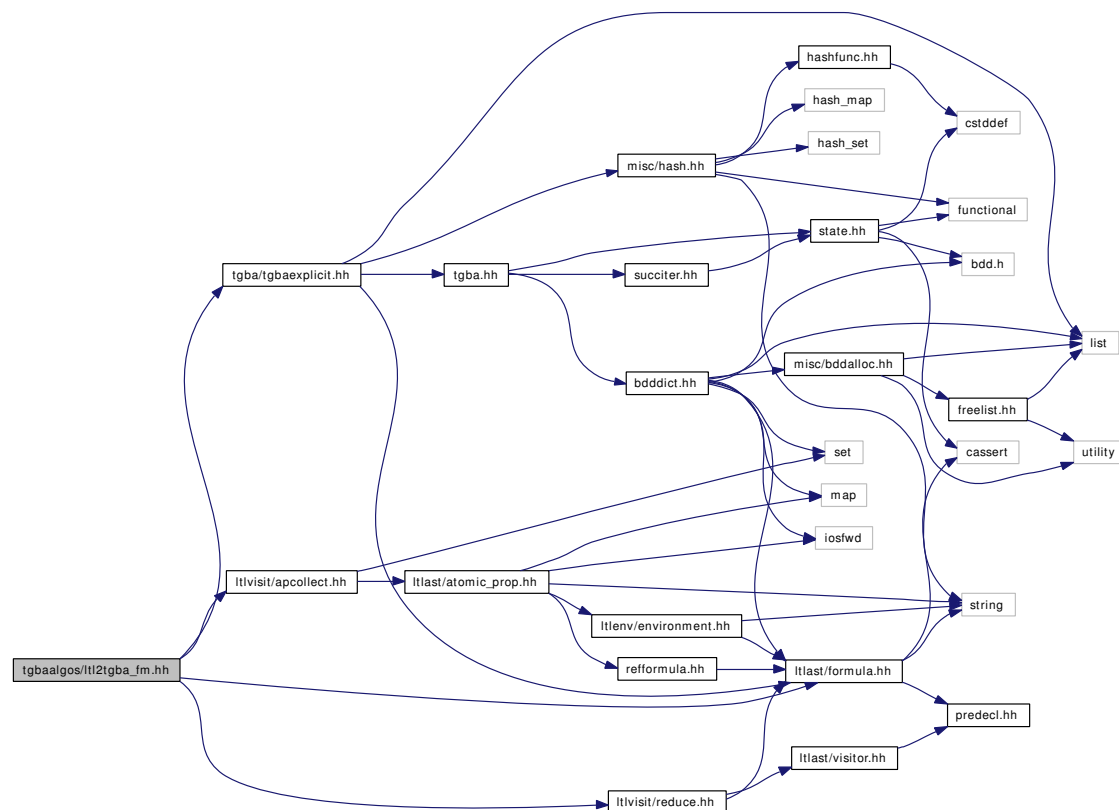
## Functions

- `std::ostream & spot::lbt\_reachable (std::ostream &os, const tgba *g)`  
*Print reachable states in LBT format.*

## 13.103 tgbaalgos/ltl2tgba\_fm.hh File Reference

```
#include "ltlast/formula.hh"
#include "tgba/tgbaexplicit.hh"
#include "ltlvisit/apcollect.hh"
#include "ltlvisit/reduce.hh"
```

Include dependency graph for ltl2tgba\_fm.hh:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [spot](#)

## Functions

- `tgba_explicit` \* [spot::ltl\\_to\\_tgba\\_fm](#) (const ltl::formula \*f, bdd\_dict \*dict, bool exprop=false, bool symb\_merge=true, bool branching\_postponement=false, bool fair\_loop\_approx=false, const ltl::atomic\_prop\_set \*unobs=0, int reduce\_ltl=ltl::Reduce\_None, bool containment\_checks=false)

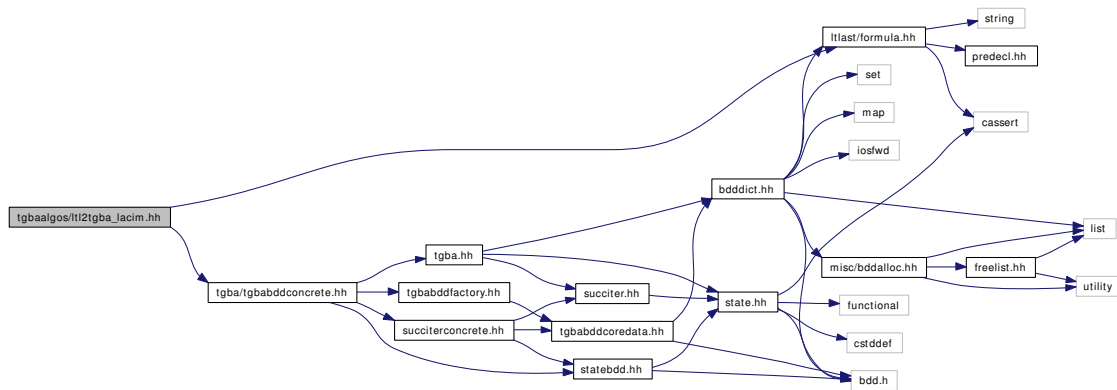
*Build a [spot::tgba\\_explicit](#) from an LTL formula.*

## 13.104 tgbaalgos/ltl2tgba\_lacim.hh File Reference

```
#include "ltlast/formula.hh"
```

```
#include "tgba/tgababddconcrete.hh"
```

Include dependency graph for ltl2tgba\_lacim.hh:



## Namespaces

- namespace [spot](#)

## Functions

- `tgba_bdd_concrete` \* [spot::ltl\\_to\\_tgba\\_lacim](#) (const ltl::formula \*f, bdd\_dict \*dict)

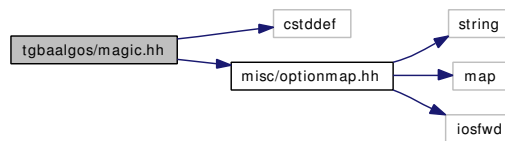
*Build a [spot::tgba\\_bdd\\_concrete](#) from an LTL formula.*

## 13.105 tgbaalgos/magic.hh File Reference

```
#include <cstddel>
```

```
#include "misc/optionmap.hh"
```

Include dependency graph for magic.hh:



## Namespaces

- namespace [spot](#)

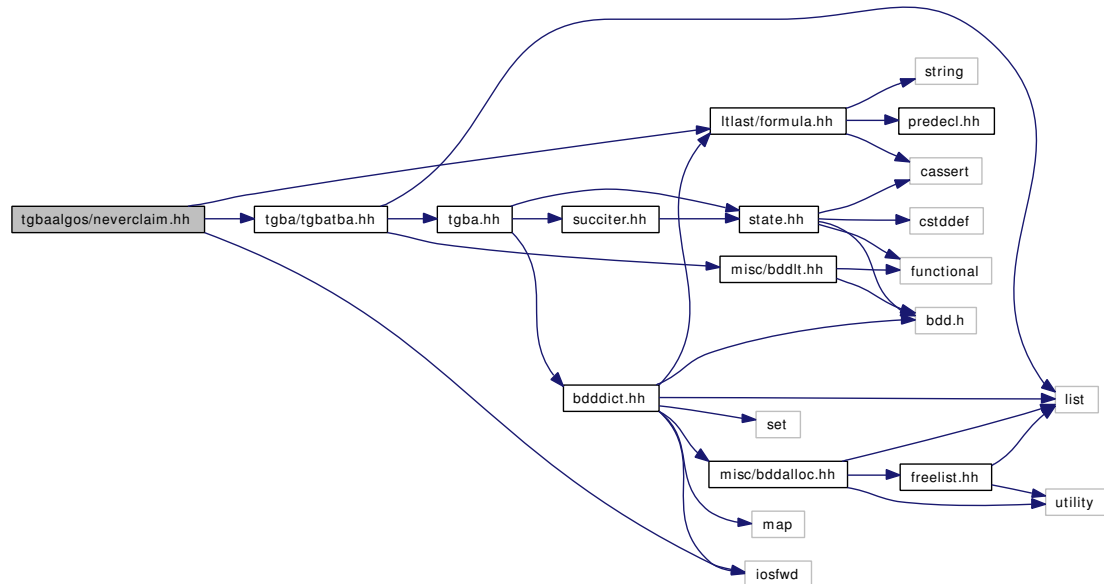
## Functions

- `emptiness_check * spot::explicit\_magic\_search (const tgba *a, option_map o=option_map())`  
*Returns an emptiness checker on the [spot::tgba](#) automaton a.*
- `emptiness_check * spot::bit\_state\_hashing\_magic\_search (const tgba *a, size_t size, option_map o=option_map())`  
*Returns an emptiness checker on the [spot::tgba](#) automaton a.*
- `emptiness_check * spot::magic\_search (const tgba *a, option_map o=option_map())`  
*Wrapper for the two `magic_search` implementations.*

## 13.106 tgbaalgos/neverclaim.hh File Reference

```
#include <iosfwd>
#include "ltlast/formula.hh"
#include "tgba/tgbatba.hh"
```

Include dependency graph for neverclaim.hh:



## Namespaces

- namespace [spot](#)

## Functions

- `std::ostream & spot::never\_claim\_reachable (std::ostream &os, const tgba_sba_proxy *g, const ltl::formula *f=0)`

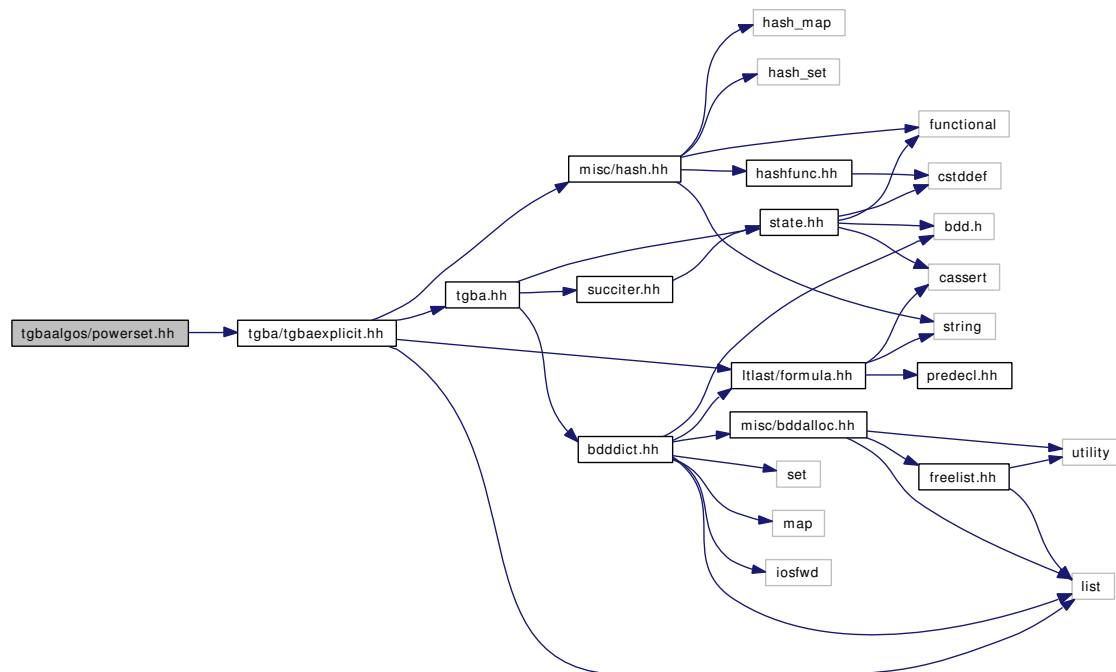
*Print reachable states in Spin never claim format.*

## 13.107 tgbaalgos/powerset.hh File Reference

```
#include "tgba/tgbaexplicit.hh"
```



Include dependency graph for powerset.hh:



## Namespaces

- namespace [spot](#)

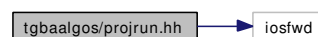
## Functions

- `tgba_explicit * spot::tgba\_powerset (const tgba *aut)`  
*Build a deterministic automaton, ignoring acceptance conditions.*

## 13.108 tgbaalgos/projrun.hh File Reference

```
#include <iosfwd>
```

Include dependency graph for projrun.hh:



## Namespaces

- namespace [spot](#)

## Functions

- `tgba_run * spot::project\_tgba\_run (const tgba *a_run, const tgba *a_proj, const tgba_run *run)`

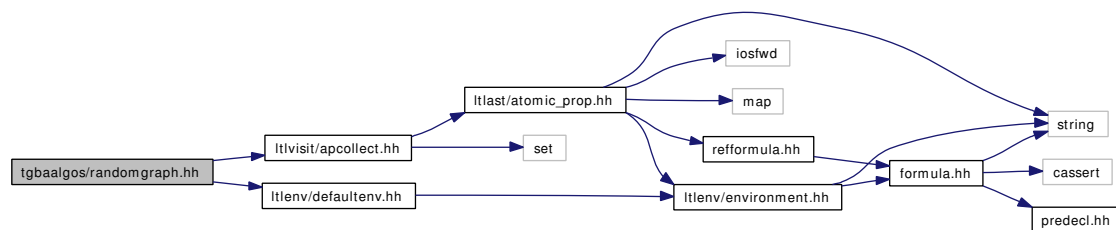
Project a [tgba\\_run](#) on a [tgba](#).

## 13.109 tgbaalgos/randomgraph.hh File Reference

```
#include "ltlvisit/apcollect.hh"
```

```
#include "ltlenv/defaultenv.hh"
```

Include dependency graph for randomgraph.hh:



### Namespaces

- namespace [spot](#)

### Functions

- [tgba](#) \* [spot::random\\_graph](#) (int n, float d, const ltl::atomic\_prop\_set \*ap, bdd\_dict \*dict, int n\_acc=0, float a=0.1, float t=0.5, ltl::environment \*env=&ltl::default\_environment::instance())

Construct a [tgba](#) randomly.

## 13.110 tgbaalgos/reducerun.hh File Reference

### Namespaces

- namespace [spot](#)

### Functions

- [tgba\\_run](#) \* [spot::reduce\\_run](#) (const [tgba](#) \*a, const [tgba\\_run](#) \*org)

Reduce an accepting run.

## 13.111 tgbaalgos/reductgba\_sim.hh File Reference

```
#include "tgba/tgbareduc.hh"
```

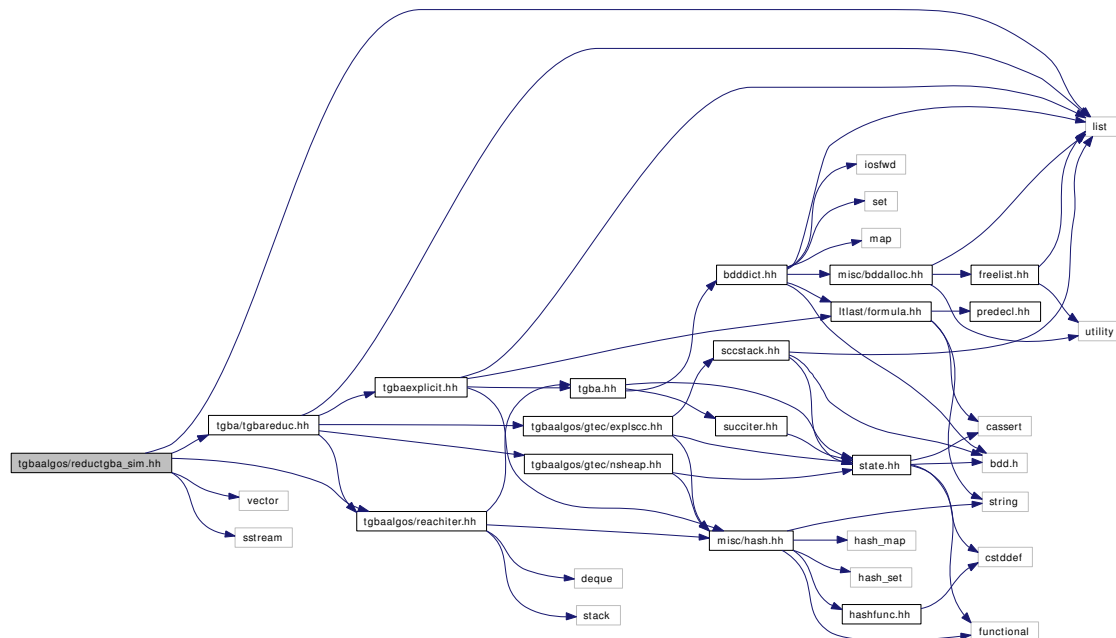
```
#include "tgbaalgos/reachiter.hh"
```

```
#include <vector>
```

```
#include <list>
```

```
#include <sstream>
```

Include dependency graph for reductgba\_sim.hh:



## Namespaces

- namespace [spot](#)

## Classes

- class [spot::parity\\_game\\_graph](#)  
*Parity game graph which compute a simulation relation.*
- class [spot::spoiler\\_node](#)  
*Spoiler node of parity game graph.*
- class [spot::duplicator\\_node](#)  
*Duplicator node of parity game graph.*
- class [spot::parity\\_game\\_graph\\_direct](#)  
*Parity game graph which compute the direct simulation relation.*
- class [spot::spoiler\\_node\\_delayed](#)  
*Spoiler node of parity game graph for delayed simulation.*
- class [spot::duplicator\\_node\\_delayed](#)  
*Duplicator node of parity game graph for delayed simulation.*

- class [spot::parity\\_game\\_graph\\_delayed](#)

### Typedefs

- typedef std::vector< spoiler\_node \* > [spot::sn\\_v](#)
- typedef std::vector< duplicator\_node \* > [spot::dn\\_v](#)
- typedef std::vector< const state \* > [spot::s\\_v](#)

### Enumerations

- enum [spot::reduce\\_tgba\\_options](#) {  
[spot::Reduce\\_None](#) = 0, [spot::Reduce\\_quotient\\_Dir\\_Sim](#) = 1, [spot::Reduce\\_transition\\_Dir\\_Sim](#) = 2,  
[spot::Reduce\\_quotient\\_Del\\_Sim](#) = 4,  
[spot::Reduce\\_transition\\_Del\\_Sim](#) = 8, [spot::Reduce\\_Scc](#) = 16, [spot::Reduce\\_All](#) = -1U }

*Options for reduce.*

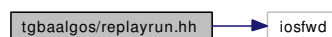
### Functions

- [tgba \\* spot::reduc\\_tgba\\_sim](#) (const tgba \*a, int opt=Reduce\_All)  
*Remove some node of the automata using a simulation relation.*
- [direct\\_simulation\\_relation \\* spot::get\\_direct\\_relation\\_simulation](#) (const tgba \*a, std::ostream &os, int opt=-1)  
*Compute a direct simulation relation on [state](#) of [tgba](#) f.*
- [delayed\\_simulation\\_relation \\* spot::get\\_delayed\\_relation\\_simulation](#) (const tgba \*a, std::ostream &os, int opt=-1)
- void [spot::free\\_relation\\_simulation](#) (direct\_simulation\_relation \*rel)  
*To free a simulation relation.*
- void [spot::free\\_relation\\_simulation](#) (delayed\_simulation\_relation \*rel)  
*To free a simulation relation.*

## 13.112 tgbaalgos/replayrun.hh File Reference

```
#include <iosfwd>
```

Include dependency graph for replayrun.hh:



### Namespaces

- namespace [spot](#)

## Functions

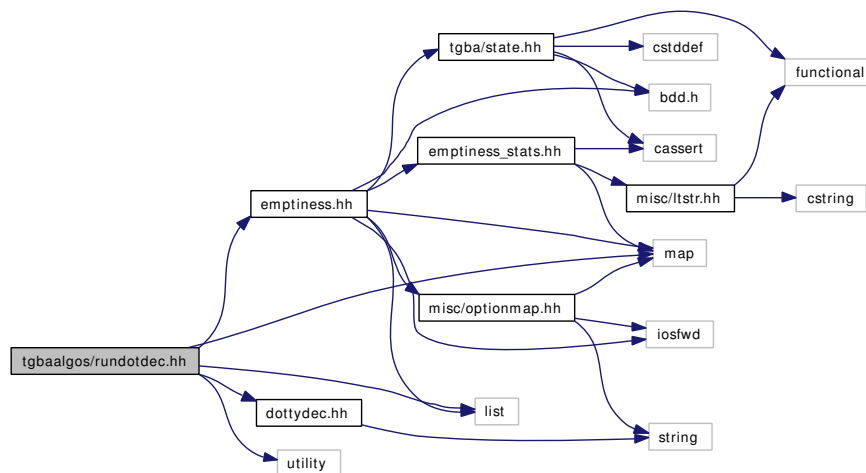
- bool [spot::replay\\_tgba\\_run](#) (std::ostream &os, const tgba \*a, const tgba\_run \*run, bool debug=false)

*Replay a [tgba\\_run](#) on a [tgba](#).*

## 13.113 tgbaalgos/rundotdec.hh File Reference

```
#include <map>
#include <utility>
#include <list>
#include "dottydec.hh"
#include "emptiness.hh"
```

Include dependency graph for rundotdec.hh:



## Namespaces

- namespace [spot](#)

## Classes

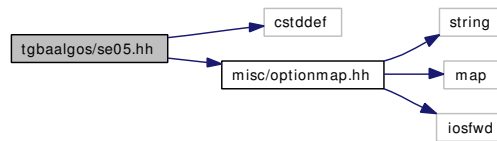
- class [spot::tgba\\_run\\_dotty\\_decorator](#)

*Highlight a [spot::tgba\\_run](#) on a [spot::tgba](#).*

## 13.114 tgbaalgos/se05.hh File Reference

```
#include <cstddef>
#include "misc/optionmap.hh"
```

Include dependency graph for se05.hh:



## Namespaces

- namespace [spot](#)

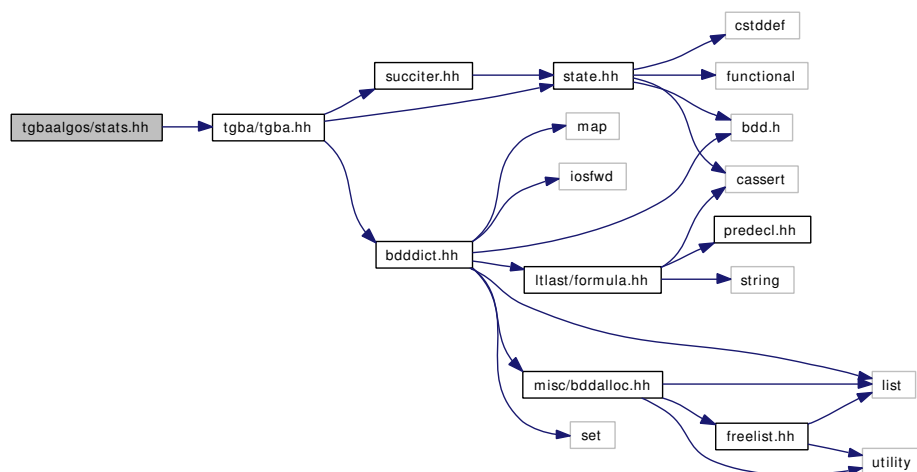
## Functions

- emptiness\_check \* [spot::explicit\\_se05\\_search](#) (const tgba \*a, option\_map o=option\_map())  
Returns an emptiness check on the [spot::tgba](#) automaton a.
- emptiness\_check \* [spot::bit\\_state\\_hashing\\_se05\\_search](#) (const tgba \*a, size\_t size, option\_map o=option\_map())  
Returns an emptiness checker on the [spot::tgba](#) automaton a.
- emptiness\_check \* [spot::se05](#) (const tgba \*a, option\_map o)  
Wrapper for the two se05 implementations.

## 13.115 tgbaalgos/stats.hh File Reference

```
#include "tgba/tgba.hh"
```

Include dependency graph for stats.hh:



## Namespaces

- namespace [spot](#)

## Classes

- struct [spot::tgba\\_statistics](#)

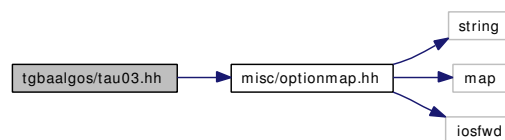
## Functions

- tgba\_statistics [spot::stats\\_reachable](#) (const tgba \*g)  
*Compute statistics for an automaton.*

## 13.116 tgbaalgos/tau03.hh File Reference

```
#include "misc/optionmap.hh"
```

Include dependency graph for tau03.hh:



## Namespaces

- namespace [spot](#)

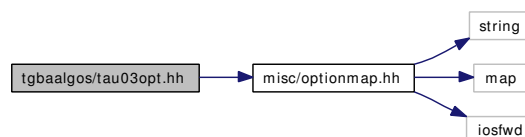
## Functions

- emptiness\_check \* [spot::explicit\\_tau03\\_search](#) (const tgba \*a, option\_map o=option\_map())  
*Returns an emptiness checker on the `spot::tgba` automaton a.*

## 13.117 tgbaalgos/tau03opt.hh File Reference

```
#include "misc/optionmap.hh"
```

Include dependency graph for tau03opt.hh:



## Namespaces

- namespace [spot](#)

## Functions

- emptiness\_check \* [spot::explicit\\_tau03\\_opt\\_search](#)(const tgba \*a, option\_map o=option\_map())  
Returns an emptiness checker on the [spot::tgba](#) automaton a.

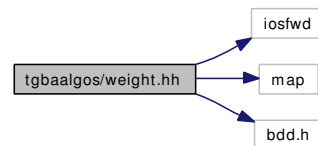
## 13.118 tgbaalgos/weight.hh File Reference

```
#include <iosfwd>
```

```
#include <map>
```

```
#include <bdd.h>
```

Include dependency graph for weight.hh:



## Namespaces

- namespace [spot](#)

## Classes

- class [spot::weight](#)  
Manage for a given automaton a vector of counter indexed by its acceptance condition.

# 14 spot Page Documentation

## 14.1 Bug List

Member [spot::get\\_delayed\\_relation\\_simulation](#)(const tgba \*a, std::ostream &os, int opt=-1) Does not work for generalized automata.

Member [spot::explicit\\_magic\\_search](#)(const tgba \*a, option\_map o=option\_map()) The name is misleading. Magic-search is the algorithm from [godefroid.93.pstv](#), not [courcoubetis.92.fmsd](#).



## Index

- ~acss\_statistics
  - spot::acss\_statistics, [88](#)
- ~atomic\_prop
  - spot::ltl::atomic\_prop, [95](#)
- ~bdd\_dict
  - spot::bdd\_dict, [109](#)
- ~bfs\_steps
  - spot::bfs\_steps, [119](#)
- ~binop
  - spot::ltl::binop, [125](#)
- ~clone\_visitor
  - spot::ltl::clone\_visitor, [129](#)
- ~connected\_component\_hash\_set
  - spot::connected\_component\_hash\_set, [132](#)
- ~connected\_component\_hash\_set\_factory
  - spot::connected\_component\_hash\_set\_factory, [134](#)
- ~const\_visitor
  - spot::ltl::const\_visitor, [135](#)
- ~constant
  - spot::ltl::constant, [138](#)
- ~couvreur99\_check
  - spot::couvreur99\_check, [144](#)
- ~couvreur99\_check\_shy
  - spot::couvreur99\_check\_shy, [156](#)
- ~couvreur99\_check\_status
  - spot::couvreur99\_check\_status, [162](#)
- ~declarative\_environment
  - spot::ltl::declarative\_environment, [164](#)
  - spot::ltl::read\_only\_environment, [296](#)
- ~default\_environment
  - spot::ltl::default\_environment, [166](#)
- ~dotty\_decorator
  - spot::dotty\_decorator, [168](#)
- ~duplicator\_node
  - spot::duplicator\_node, [172](#)
- ~duplicator\_node\_delayed
  - spot::duplicator\_node\_delayed, [176](#)
- ~emptiness\_check
  - spot::emptiness\_check, [185](#)
- ~emptiness\_check\_result
  - spot::emptiness\_check\_result, [191](#)
- ~environment
  - spot::ltl::environment, [193](#)
- ~evtgba
  - spot::evtgba, [194](#)
- ~evtgba\_explicit
  - spot::evtgba\_explicit, [198](#)
- ~evtgba\_iterator
  - spot::evtgba\_iterator, [202](#)
- ~evtgba\_product
  - spot::evtgba\_product, [204](#)
- ~evtgba\_reachable\_iterator
  - spot::evtgba\_reachable\_iterator, [207](#)
- ~explicit\_connected\_component
  - spot::explicit\_connected\_component, [217](#)
- ~explicit\_connected\_component\_factory
  - spot::explicit\_connected\_component\_factory, [219](#)
- ~formula
  - spot::ltl::formula, [221](#)
- ~free\_list
  - spot::free\_list, [225](#)
- ~gspn\_interface
  - spot::gspn\_interface, [229](#)
- ~gspn\_ssp\_interface
  - spot::gspn\_ssp\_interface, [231](#)
- ~language\_containment\_checker
  - spot::ltl::language\_containment\_checker, [234](#)
- ~loopless\_modular\_mixed\_radix\_gray\_code
  - spot::loopless\_modular\_mixed\_radix\_gray\_code, [241](#)
- ~multop
  - spot::ltl::multop, [251](#)
- ~numbered\_state\_heap
  - spot::numbered\_state\_heap, [255](#)
- ~numbered\_state\_heap\_const\_iterator
  - spot::numbered\_state\_heap\_const\_iterator, [257](#)
- ~numbered\_state\_heap\_factory
  - spot::numbered\_state\_heap\_factory, [258](#)
- ~numbered\_state\_heap\_hash\_map
  - spot::numbered\_state\_heap\_hash\_map, [260](#)
- ~numbered\_state\_heap\_hash\_map\_factory
  - spot::numbered\_state\_heap\_hash\_map\_factory, [263](#)
- ~parity\_game\_graph
  - spot::parity\_game\_graph, [269](#)
- ~parity\_game\_graph\_delayed
  - spot::parity\_game\_graph\_delayed, [276](#)
- ~parity\_game\_graph\_direct
  - spot::parity\_game\_graph\_direct, [282](#)
- ~postfix\_visitor
  - spot::ltl::postfix\_visitor, [289](#)
- ~random\_ltl
  - spot::ltl::random\_ltl, [292](#)
- ~ref\_formula
  - spot::ltl::ref\_formula, [299](#)
- ~rsymbol
  - spot::rsymbol, [302](#)

- ~simplify\_f\_g\_visitor
  - spot::ltl::simplify\_f\_g\_visitor, 307
- ~spoiler\_node
  - spot::spoiler\_node, 312
- ~spoiler\_node\_delayed
  - spot::spoiler\_node\_delayed, 315
- ~state
  - spot::state, 322
- ~state\_evtgba\_explicit
  - spot::state\_evtgba\_explicit, 327
- ~state\_explicit
  - spot::state\_explicit, 329
- ~state\_product
  - spot::state\_product, 332
- ~symbol
  - spot::symbol, 336
- ~tgba
  - spot::tgba, 340
- ~tgba\_bdd\_concrete
  - spot::tgba\_bdd\_concrete, 347
- ~tgba\_bdd\_concrete\_factory
  - spot::tgba\_bdd\_concrete\_factory, 353
- ~tgba\_bdd\_factory
  - spot::tgba\_bdd\_factory, 361
- ~tgba\_explicit
  - spot::tgba\_explicit, 367
- ~tgba\_explicit\_succ\_iterator
  - spot::tgba\_explicit\_succ\_iterator, 374
- ~tgba\_product
  - spot::tgba\_product, 379
- ~tgba\_reachable\_iterator
  - spot::tgba\_reachable\_iterator, 385
- ~tgba\_reduc
  - spot::tgba\_reduc, 402
- ~tgba\_run
  - spot::tgba\_run, 411
- ~tgba\_run\_dotty\_decorator
  - spot::tgba\_run\_dotty\_decorator, 414
- ~tgba\_succ\_iterator
  - spot::tgba\_succ\_iterator, 424
- ~tgba\_succ\_iterator\_concrete
  - spot::tgba\_succ\_iterator\_concrete, 429
- ~tgba\_succ\_iterator\_product
  - spot::tgba\_succ\_iterator\_product, 433
- ~tgba\_tba\_proxy
  - spot::tgba\_tba\_proxy, 439
- ~unabbreviate\_logic\_visitor
  - spot::ltl::unabbreviate\_logic\_visitor, 449
- ~unabbreviate\_ltl\_visitor
  - spot::ltl::unabbreviate\_ltl\_visitor, 452
- ~unop
  - spot::ltl::unop, 456
- ~unsigned\_statistics
  - spot::unsigned\_statistics, 460
- ~visitor
  - spot::ltl::visitor, 462
- a\_
  - spot::bfs\_steps, 120
  - spot::couvreur99\_check, 146
  - spot::couvreur99\_check\_result, 151
  - spot::couvreur99\_check\_shy, 159
  - spot::emptiness\_check, 186
  - spot::emptiness\_check\_result, 192
  - spot::loopless\_modular\_mixed\_radix\_gray\_code, 242
  - spot::tgba\_tba\_proxy, 442
- a\_first
  - spot::loopless\_modular\_mixed\_radix\_gray\_code, 242
- a\_last
  - spot::loopless\_modular\_mixed\_radix\_gray\_code, 242
- a\_next
  - spot::loopless\_modular\_mixed\_radix\_gray\_code, 242
- acc
  - spot::couvreur99\_check\_shy::successor, 160
  - spot::tgba\_run::step, 412
- acc\_
  - spot::duplicator\_node, 173
  - spot::duplicator\_node\_delayed, 178
  - spot::tgba\_bdd\_concrete\_factory, 355
  - spot::tgba\_reduc, 410
- acc\_cycle\_
  - spot::tgba\_sba\_proxy, 422
  - spot::tgba\_tba\_proxy, 442
- acc\_formula\_map
  - spot::bdd\_dict, 112
- acc\_map
  - spot::bdd\_dict, 112
- acc\_map\_
  - spot::tgba\_bdd\_concrete\_factory, 353
- acc\_set
  - spot::tgba\_bdd\_core\_data, 360
- acc\_set\_
  - spot::evtgba\_explicit, 200
- accept
  - spot::ltl::atomic\_prop, 95
  - spot::ltl::binop, 125
  - spot::ltl::constant, 138
  - spot::ltl::formula, 221
  - spot::ltl::multop, 251
  - spot::ltl::ref\_formula, 300
  - spot::ltl::unop, 457
- acceptance\_condition\_visited\_
  - spot::spoiler\_node\_delayed, 317
- acceptance\_conditions

- spot::evtgba\_explicit::transition, 201
- spot::tgba\_bdd\_core\_data, 359
- spot::tgba\_explicit::transition, 372
- accepting\_cycle
  - spot::couvreur99\_check\_result, 150
- accepting\_run
  - spot::couvreur99\_check\_result, 150
  - spot::emptiness\_check\_result, 191
- acss\_states
  - spot::acss\_statistics, 88
  - spot::couvreur99\_check\_result, 150
- acss\_statistics
  - spot::acss\_statistics, 88
- add\_acceptance\_condition
  - spot::tgba\_explicit, 367
  - spot::tgba\_reduc, 406
- add\_acceptance\_conditions
  - spot::tgba\_explicit, 368
  - spot::tgba\_reduc, 406
- add\_condition
  - spot::tgba\_explicit, 367
  - spot::tgba\_reduc, 405
- add\_conditions
  - spot::tgba\_explicit, 367
  - spot::tgba\_reduc, 405
- add\_duplicator\_node\_delayed
  - spot::parity\_game\_graph\_delayed, 276
- add\_pred
  - spot::duplicator\_node, 173
  - spot::duplicator\_node\_delayed, 177
  - spot::spoiler\_node, 312
  - spot::spoiler\_node\_delayed, 316
- add\_spoiler\_node\_delayed
  - spot::parity\_game\_graph\_delayed, 276
- add\_state
  - spot::evtgba\_reachable\_iterator, 208
  - spot::evtgba\_reachable\_iterator\_breadth\_first, 211
  - spot::evtgba\_reachable\_iterator\_depth\_first, 215
  - spot::parity\_game\_graph, 270
  - spot::parity\_game\_graph\_delayed, 277
  - spot::parity\_game\_graph\_direct, 283
  - spot::tgba\_explicit, 368
  - spot::tgba\_reachable\_iterator, 385
  - spot::tgba\_reachable\_iterator\_breadth\_first, 390
  - spot::tgba\_reachable\_iterator\_depth\_first, 394
  - spot::tgba\_reduc, 406, 409
- add\_succ
  - spot::duplicator\_node, 172
  - spot::duplicator\_node\_delayed, 177
  - spot::spoiler\_node, 312
  - spot::spoiler\_node\_delayed, 316
- add\_transition
  - spot::evtgba\_explicit, 199
- Algorithm patterns, 39
- Algorithms for LTL formulae, 21
- all\_acc\_
  - spot::evtgba\_product, 205
- all\_acceptance\_conditions
  - spot::evtgba, 195
  - spot::evtgba\_explicit, 198
  - spot::evtgba\_product, 205
  - spot::tgba, 342
  - spot::tgba\_bdd\_concrete, 349
  - spot::tgba\_bdd\_core\_data, 359
  - spot::tgba\_explicit, 369
  - spot::tgba\_product, 381
  - spot::tgba\_reduc, 407
  - spot::tgba\_sba\_proxy, 421
  - spot::tgba\_tba\_proxy, 440
- all\_acceptance\_conditions\_
  - spot::tgba\_explicit, 371
  - spot::tgba\_explicit\_succ\_iterator, 375
  - spot::tgba\_product, 382
  - spot::tgba\_reduc, 410
- all\_acceptance\_conditions\_computed\_
  - spot::tgba\_explicit, 371
  - spot::tgba\_reduc, 410
- allocate\_variables
  - spot::bdd\_allocator, 102
  - spot::bdd\_dict, 111
- alphabet
  - spot::evtgba, 195
  - spot::evtgba\_explicit, 198
  - spot::evtgba\_product, 205
- alphabet\_
  - spot::evtgba\_explicit, 200
  - spot::evtgba\_product, 205
- And
  - spot::ltl::multop, 251
- anon\_free\_list
  - spot::bdd\_dict::anon\_free\_list, 116
- ap
  - spot::ltl::random\_ltl, 293
- ap\_
  - spot::ltl::random\_ltl, 293
- arc
  - spot::couvreur99\_check\_shy, 158
- ars\_cycle\_states
  - spot::acss\_statistics, 89
  - spot::ars\_statistics, 91
  - spot::couvreur99\_check\_result, 151
- ars\_prefix\_states
  - spot::acss\_statistics, 88
  - spot::ars\_statistics, 91
  - spot::couvreur99\_check\_result, 151

- ars\_statistics
  - spot::ars\_statistics, 91
- as\_bdd
  - spot::state\_bdd, 325
- assert\_emptyiness
  - spot::bdd\_dict, 111
- atomic\_prop
  - spot::ltl::atomic\_prop, 95
- atomic\_prop\_collect
  - ltl\_misc, 27
- atomic\_prop\_set
  - ltl\_misc, 27
- aut
  - spot::couvreur99\_check\_status, 162
- automata\_
  - spot::evtgba\_reachable\_iterator, 209
  - spot::evtgba\_reachable\_iterator\_breadth\_first, 212
  - spot::evtgba\_reachable\_iterator\_depth\_first, 216
  - spot::parity\_game\_graph, 271
  - spot::parity\_game\_graph\_delayed, 278
  - spot::parity\_game\_graph\_direct, 284
  - spot::tgba\_reachable\_iterator, 387
  - spot::tgba\_reachable\_iterator\_breadth\_first, 392
  - spot::tgba\_reachable\_iterator\_depth\_first, 396
  - spot::tgba\_reduc, 410
- automaton
  - spot::couvreur99\_check, 145
  - spot::couvreur99\_check\_result, 150
  - spot::couvreur99\_check\_shy, 157
  - spot::emptiness\_check, 185
  - spot::emptiness\_check\_result, 191
  - spot::gspn\_interface, 229
  - spot::gspn\_ssp\_interface, 231
- barand
  - spot::barand, 98
- basic\_reduce
  - ltl\_rewriting, 25
- bdd\_allocator
  - spot::bdd\_allocator, 102
- bdd\_dict
  - spot::bdd\_dict, 109
- bdd\_format\_accset
  - spot, 78
- bdd\_format\_formula
  - spot, 78
- bdd\_format\_sat
  - spot, 78
- bdd\_format\_set
  - spot, 79
- bdd\_print\_acc
  - spot, 79
- bdd\_print\_accset
  - spot, 79
- bdd\_print\_dot
  - spot, 79
- bdd\_print\_formula
  - spot, 80
- bdd\_print\_sat
  - spot, 80
- bdd\_print\_set
  - spot, 80
- bdd\_print\_table
  - spot, 80
- bdd\_to\_formula
  - spot, 80
- bdd\_v
  - spot::parity\_game\_graph\_delayed, 275
- begin
  - ltlyy::location, 239
  - ltlyy::stack, 321
  - sautyy::location, 237
  - sautyy::stack, 319
- bfs\_steps
  - spot::bfs\_steps, 119
- binop
  - spot::ltl::binop, 125
- bit\_state\_hashing\_magic\_search
  - emptiness\_check\_algorithms, 46
- bit\_state\_hashing\_se05\_search
  - emptiness\_check\_algorithms, 46
- bmrand
  - random, 32
- branching\_postponement\_
  - spot::ltl::language\_containment\_checker, 234
- build
  - spot::connected\_component\_hash\_set\_factory, 134
  - spot::explicit\_connected\_component\_factory, 219
  - spot::ltl::random\_ltl::op\_proba, 294
  - spot::numbered\_state\_heap\_factory, 258
  - spot::numbered\_state\_heap\_hash\_map\_factory, 263
- build\_graph
  - spot::parity\_game\_graph, 270
  - spot::parity\_game\_graph\_delayed, 276
  - spot::parity\_game\_graph\_direct, 282
- build\_link
  - spot::parity\_game\_graph\_direct, 282
- build\_recurse\_successor\_duplicator
  - spot::parity\_game\_graph\_delayed, 276
- build\_recurse\_successor\_spoiler
  - spot::parity\_game\_graph\_delayed, 276
- builder

- spot::ltl::random\_ltl::op\_proba, 294
- cancel
  - spot::timer\_map, 445
- cc\_map
  - spot::bdd\_dict, 108
- check
  - spot::couvreur99\_check, 144
  - spot::couvreur99\_check\_shy, 156
  - spot::emptiness\_check, 185
- child
  - spot::ltl::unop, 457
- child\_
  - spot::ltl::unop, 458
- children\_
  - spot::ltl::multop, 253
- clear\_rem
  - spot::scc\_stack, 304
- clear\_todo
  - spot::couvreur99\_check\_shy, 156
- clone
  - ltl\_essential, 19
  - spot::state, 323
  - spot::state\_bdd, 325
  - spot::state\_evtgba\_explicit, 327
  - spot::state\_explicit, 330
  - spot::state\_product, 333
- clone\_counts
  - spot::bdd\_dict, 112
- clone\_visitor
  - spot::ltl::clone\_visitor, 129
- column
  - ltlyy::position, 286
  - sautyy::position, 287
- columns
  - ltlyy::location, 239
  - ltlyy::position, 285
  - sautyy::location, 237
  - sautyy::position, 287
- common\_symbol\_table
  - spot::evtgba\_product, 204
- common\_symbols\_
  - spot::evtgba\_product, 205
- compare
  - spot::duplicator\_node, 172
  - spot::duplicator\_node\_delayed, 177
  - spot::spoiler\_node, 313
  - spot::spoiler\_node\_delayed, 316
  - spot::state, 322
  - spot::state\_bdd, 325
  - spot::state\_evtgba\_explicit, 327
  - spot::state\_explicit, 329
  - spot::state\_product, 332
- complement\_all\_acceptance\_conditions
  - spot::tgba\_explicit, 368
  - spot::tgba\_reduc, 406
- compute\_scc
  - spot::tgba\_reduc, 403
- compute\_support\_conditions
  - spot::tgba, 343
  - spot::tgba\_bdd\_concrete, 349
  - spot::tgba\_explicit, 369
  - spot::tgba\_product, 381
  - spot::tgba\_reduc, 407
  - spot::tgba\_sba\_proxy, 421
  - spot::tgba\_tba\_proxy, 441
- compute\_support\_variables
  - spot::tgba, 343
  - spot::tgba\_bdd\_concrete, 349
  - spot::tgba\_explicit, 369
  - spot::tgba\_product, 381
  - spot::tgba\_reduc, 407
  - spot::tgba\_sba\_proxy, 421
  - spot::tgba\_tba\_proxy, 441
- condition
  - spot::connected\_component\_hash\_set, 133
  - spot::explicit\_connected\_component, 218
  - spot::scc\_stack::connected\_component, 305
  - spot::tgba\_explicit::transition, 372
- connected\_component
  - spot::scc\_stack::connected\_component, 305
- connected\_component\_hash\_set\_factory
  - spot::connected\_component\_hash\_set\_factory, 134
- const\_iterator
  - ltlyy::stack, 320
  - sautyy::stack, 318
- constant
  - spot::ltl::constant, 138
- constrain\_relation
  - spot::tgba\_bdd\_concrete\_factory, 354
- construct
  - spot::emptiness\_check\_instantiator, 188
- contained
  - spot::ltl::language\_containment\_checker, 234
- contained\_neg
  - spot::ltl::language\_containment\_checker, 234
- copy\_acceptance\_conditions\_of
  - spot::tgba\_explicit, 367
  - spot::tgba\_reduc, 406
- couvreur99
  - emptiness\_check\_algorithms, 47
- couvreur99\_check
  - spot::couvreur99\_check, 144
- couvreur99\_check\_result
  - spot::couvreur99\_check\_result, 150
- couvreur99\_check\_shy
  - spot::couvreur99\_check\_shy, 156

- couvreur99\_check\_ssp\_semi
  - emptiness\_check\_ssp, 19
- couvreur99\_check\_ssp\_shy
  - emptiness\_check\_ssp, 19
- couvreur99\_check\_ssp\_shy\_semi
  - emptiness\_check\_ssp, 19
- couvreur99\_check\_status
  - spot::couvreur99\_check\_status, 162
- create\_atomic\_prop
  - spot::tgba\_bdd\_concrete\_factory, 354
- create\_state
  - spot::tgba\_bdd\_concrete\_factory, 354
- create\_transition
  - spot::tgba\_explicit, 367
  - spot::tgba\_reduc, 404, 405
- cube\_
  - spot::minato\_isop, 244
- current\_
  - spot::tgba\_succ\_iterator\_concrete, 430
- current\_acc\_
  - spot::tgba\_succ\_iterator\_concrete, 430
- current\_acceptance\_conditions
  - spot::evtgba\_iterator, 202
  - spot::tgba\_explicit\_succ\_iterator, 375
  - spot::tgba\_succ\_iterator, 425
  - spot::tgba\_succ\_iterator\_concrete, 430
  - spot::tgba\_succ\_iterator\_product, 434
- current\_cond\_
  - spot::tgba\_succ\_iterator\_product, 435
- current\_condition
  - spot::tgba\_explicit\_succ\_iterator, 375
  - spot::tgba\_succ\_iterator, 425
  - spot::tgba\_succ\_iterator\_concrete, 430
  - spot::tgba\_succ\_iterator\_product, 434
- current\_label
  - spot::evtgba\_iterator, 202
- current\_state
  - spot::evtgba\_iterator, 202
  - spot::tgba\_explicit\_succ\_iterator, 374
  - spot::tgba\_succ\_iterator, 424
  - spot::tgba\_succ\_iterator\_concrete, 429
  - spot::tgba\_succ\_iterator\_product, 434
- current\_state\_
  - spot::tgba\_succ\_iterator\_concrete, 430
- cycle
  - spot::tgba\_run, 412
- cycle\_list
  - spot::tgba\_sba\_proxy, 419
  - spot::tgba\_tba\_proxy, 439
- cycle\_seed
  - spot::couvreur99\_check\_status, 162
- cycle\_states\_
  - spot::ars\_statistics, 91
- data\_
  - spot::tgba\_bdd\_concrete, 351
  - spot::tgba\_bdd\_concrete\_factory, 355
  - spot::tgba\_succ\_iterator\_concrete, 430
- dead\_
  - spot::gspn\_interface, 229
- dec\_depth
  - spot::couvreur99\_check, 146
  - spot::couvreur99\_check\_shy, 158
  - spot::ec\_statistics, 181
- dec\_weight\_handler
  - spot::weight, 465
- declarative\_environment
  - spot::ltl::declarative\_environment, 164
  - spot::ltl::read\_only\_environment, 296
- declare
  - spot::ltl::declarative\_environment, 164
  - spot::ltl::read\_only\_environment, 296
- declare\_acceptance\_condition
  - spot::evtgba\_explicit, 199
  - spot::tgba\_bdd\_concrete\_factory, 354
  - spot::tgba\_bdd\_core\_data, 359
  - spot::tgba\_explicit, 367
  - spot::tgba\_reduc, 405
- declare\_atomic\_prop
  - spot::tgba\_bdd\_core\_data, 358
- declare\_now\_next
  - spot::tgba\_bdd\_core\_data, 358
- declare\_state
  - spot::evtgba\_explicit, 199
- Decorating the dot output, 44
- default\_environment
  - spot::ltl::default\_environment, 166
- del\_pred
  - spot::duplicator\_node, 173
  - spot::duplicator\_node\_delayed, 177
  - spot::spoiler\_node, 312
  - spot::spoiler\_node\_delayed, 316
- del\_succ
  - spot::duplicator\_node, 173
  - spot::duplicator\_node\_delayed, 177
  - spot::spoiler\_node, 312
  - spot::spoiler\_node\_delayed, 316
- delete\_scc
  - spot::tgba\_reduc, 404
- delete\_transitions
  - spot::tgba\_reduc, 403
- depth
  - spot::couvreur99\_check, 146
  - spot::couvreur99\_check\_shy, 158
  - spot::ec\_statistics, 181
- depth\_
  - spot::ec\_statistics, 182
- Derivable visitors, 24

- dest
  - spot::tgba\_explicit::transition, 372
- destroy
  - ltl\_essential, 19
- dict
  - spot::tgba\_bdd\_core\_data, 360
- dict\_
  - spot::bdd\_dict::anon\_free\_list, 117
  - spot::gspn\_interface, 229
  - spot::gspn\_ssp\_interface, 231
  - spot::ltl::language\_containment\_checker, 234
  - spot::tgba\_explicit, 371
  - spot::tgba\_product, 382
  - spot::tgba\_reduc, 410
- display\_rel\_sim
  - spot::tgba\_reduc, 403
- display\_scc
  - spot::tgba\_reduc, 403
- dn\_v
  - tgba\_reduction, 41
- doit
  - spot::ltl::postfix\_visitor, 289, 290
- doit\_default
  - spot::ltl::postfix\_visitor, 290
- done
  - spot::evtgba\_iterator, 202
  - spot::loopless\_modular\_mixed\_radix\_gray\_code, 242
  - spot::numbered\_state\_heap\_const\_iterator, 257
  - spot::tgba\_explicit\_succ\_iterator, 374
  - spot::tgba\_succ\_iterator, 424
  - spot::tgba\_succ\_iterator\_concrete, 429
  - spot::tgba\_succ\_iterator\_product, 434
- done\_
  - spot::loopless\_modular\_mixed\_radix\_gray\_code, 242
- dotty
  - ltl\_io, 22
- dotty\_decorator
  - spot::dotty\_decorator, 168
- dotty\_reachable
  - spot, 81
  - tgba\_io, 36
- drand
  - random, 32
- dump
  - ltl\_io, 22
  - spot::bdd\_dict, 111
  - spot::ltl::atomic\_prop, 96
  - spot::ltl::binop, 126
  - spot::ltl::constant, 139
  - spot::ltl::formula, 221
  - spot::ltl::multop, 253
  - spot::ltl::ref\_formula, 300
  - spot::ltl::unop, 458
- dump\_
  - spot::ltl::atomic\_prop, 97
  - spot::ltl::binop, 127
  - spot::ltl::constant, 140
  - spot::ltl::formula, 222
  - spot::ltl::multop, 253
  - spot::ltl::ref\_formula, 300
  - spot::ltl::unop, 458
- dump\_free\_list
  - spot::bdd\_allocator, 102
  - spot::bdd\_dict::anon\_free\_list, 116
  - spot::free\_list, 226
- dump\_instances
  - spot::ltl::atomic\_prop, 96
  - spot::symbol, 336
- dump\_priorities
  - spot::ltl::random\_ltl, 292
- duplicator\_node
  - spot::duplicator\_node, 172
- duplicator\_node\_delayed
  - spot::duplicator\_node\_delayed, 176
- duplicator\_vertice\_
  - spot::parity\_game\_graph, 271
  - spot::parity\_game\_graph\_delayed, 278
  - spot::parity\_game\_graph\_direct, 284
- ec\_statistics
  - spot::ec\_statistics, 181
- ecs\_
  - spot::couvreur99\_check, 146
  - spot::couvreur99\_check\_result, 151
  - spot::couvreur99\_check\_shy, 158
- Emptiness-check algorithms, 45
- Emptiness-check algorithms for SSP, 19
- Emptiness-check statistics, 56
- Emptiness-checks, 44
- emptiness\_check
  - spot::emptiness\_check, 185
- emptiness\_check\_algorithms
  - bit\_state\_hashing\_magic\_search, 46
  - bit\_state\_hashing\_se05\_search, 46
  - couvreur99, 47
  - explicit\_gv04\_check, 48
  - explicit\_magic\_search, 49
  - explicit\_se05\_search, 50
  - explicit\_tau03\_opt\_search, 51
  - explicit\_tau03\_search, 52
  - magic\_search, 53
  - se05, 53
- emptiness\_check\_instantiator
  - spot::emptiness\_check\_instantiator, 188
- emptiness\_check\_result



- spot::emptiness\_check\_result, 191
- emptiness\_check\_ssp
  - couvreur99\_check\_ssp\_semi, 19
  - couvreur99\_check\_ssp\_shy, 19
  - couvreur99\_check\_ssp\_shy\_semi, 19
- empty
  - spot::scc\_stack, 304
  - spot::timer\_map, 446
- end
  - ltlyy::location, 239
  - ltlyy::stack, 321
  - sautyy::location, 237
  - sautyy::stack, 319
  - spot::evtgba\_reachable\_iterator, 208
  - spot::evtgba\_reachable\_iterator\_breadth\_first, 212
  - spot::evtgba\_reachable\_iterator\_depth\_first, 215
  - spot::parity\_game\_graph, 269
  - spot::parity\_game\_graph\_delayed, 276
  - spot::parity\_game\_graph\_direct, 283
  - spot::tgba\_reachable\_iterator, 386
  - spot::tgba\_reachable\_iterator\_breadth\_first, 391
  - spot::tgba\_reachable\_iterator\_depth\_first, 395
  - spot::tgba\_reduc, 403
- env
  - spot::ltl::atomic\_prop, 95
- env\_
  - spot::gspn\_interface, 229
  - spot::gspn\_ssp\_interface, 231
  - spot::ltl::atomic\_prop, 97
- equal
  - spot::ltl::language\_containment\_checker, 234
- Equiv
  - spot::ltl::binop, 125
- err\_
  - spot::gspn\_exception, 227
- escape\_str
  - misc\_tools, 30
- Essential LTL types, 19
- Essential TGBA types, 33
- evtgba
  - spot::evtgba, 194
- evtgba/ Directory Reference, 56
- evtgba/evtgba.hh, 468
- evtgba/evtgbaiter.hh, 469
- evtgba/explicit.hh, 469
- evtgba/product.hh, 470
- evtgba/symbol.hh, 471
- evtgba\_explicit
  - spot::evtgba\_explicit, 198
- evtgba\_parse
  - spot, 81
- evtgba\_parse\_error
  - spot, 78
- evtgba\_parse\_error\_list
  - spot, 78
- evtgba\_product
  - spot::evtgba\_product, 204
- evtgba\_product\_operands
  - spot::evtgba\_product, 204
- evtgba\_reachable\_iterator
  - spot::evtgba\_reachable\_iterator, 207
- evtgba\_reachable\_iterator\_breadth\_first
  - spot::evtgba\_reachable\_iterator\_breadth\_first, 211
- evtgba\_reachable\_iterator\_depth\_first
  - spot::evtgba\_reachable\_iterator\_depth\_first, 214
- evtgba\_save\_reachable
  - spot, 81
- evtgbalogs/ Directory Reference, 57
- evtgbalogs/dotty.hh, 472
- evtgbalogs/reachiter.hh, 474
- evtgbalogs/save.hh, 476
- evtgbalogs/tgba2evtgba.hh, 477
- evtgbaparse/ Directory Reference, 57
- evtgbaparse/public.hh, 477
- explicit\_gv04\_check
  - emptiness\_check\_algorithms, 48
- explicit\_magic\_search
  - emptiness\_check\_algorithms, 49
- explicit\_se05\_search
  - emptiness\_check\_algorithms, 50
- explicit\_tau03\_opt\_search
  - emptiness\_check\_algorithms, 51
- explicit\_tau03\_search
  - emptiness\_check\_algorithms, 52
- exprop\_
  - spot::ltl::language\_containment\_checker, 234
- extend
  - spot::bdd\_allocator, 102
  - spot::bdd\_dict::anon\_free\_list, 116
  - spot::free\_list, 226
- extvarnum
  - spot::bdd\_allocator, 102
- F
  - spot::ltl::unop, 456
- f0\_max
  - spot::minato\_isop::local\_vars, 246
- f0\_min
  - spot::minato\_isop::local\_vars, 246
- f1\_max
  - spot::minato\_isop::local\_vars, 246
- f1\_min
  - spot::minato\_isop::local\_vars, 246



- f\_
  - spot::loopless\_modular\_mixed\_radix\_gray\_code, 242
- f\_max
  - spot::minato\_isop::local\_vars, 245
- f\_min
  - spot::minato\_isop::local\_vars, 245
- fair\_loop\_approx\_
  - spot::ltl::language\_containment\_checker, 234
- False
  - spot::ltl::constant, 138
- false\_instance
  - spot::ltl::constant, 139
- filename
  - ltlyy::position, 285
  - sautyy::position, 287
- filter
  - spot::bfs\_steps, 119
- finalize
  - spot::bfs\_steps, 120
- find
  - spot::numbered\_state\_heap, 255
  - spot::numbered\_state\_heap\_hash\_map, 260, 261
- find\_state
  - spot::couvreur99\_check\_shy, 156
- finish
  - spot::tgba\_bdd\_concrete\_factory, 355
- first
  - spot::evtgba\_iterator, 202
  - spot::loopless\_modular\_mixed\_radix\_gray\_code, 242
  - spot::ltl::binop, 125
  - spot::numbered\_state\_heap\_const\_iterator, 257
  - spot::tgba\_explicit\_succ\_iterator, 374
  - spot::tgba\_succ\_iterator, 424
  - spot::tgba\_succ\_iterator\_concrete, 429
  - spot::tgba\_succ\_iterator\_product, 433
- first\_
  - spot::ltl::binop, 127
- FirstStep
  - spot::minato\_isop::local\_vars, 245
- fl
  - spot::bdd\_allocator, 103
  - spot::bdd\_dict::anon\_free\_list, 117
  - spot::free\_list, 226
- format\_acceptance\_condition
  - spot::evtgba, 195
  - spot::evtgba\_explicit, 199
  - spot::evtgba\_product, 205
- format\_acceptance\_conditions
  - spot::evtgba, 195
  - spot::evtgba\_explicit, 199
- spot::evtgba\_product, 205
- format\_evtgba\_parse\_errors
  - spot, 81
- format\_label
  - spot::evtgba, 195
  - spot::evtgba\_explicit, 199
  - spot::evtgba\_product, 205
- format\_parse\_errors
  - ltl\_io, 22
- format\_state
  - spot::evtgba, 195
  - spot::evtgba\_explicit, 198, 199
  - spot::evtgba\_product, 204
  - spot::tgba, 342
  - spot::tgba\_bdd\_concrete, 348
  - spot::tgba\_explicit, 369
  - spot::tgba\_product, 380
  - spot::tgba\_reduc, 403
  - spot::tgba\_sba\_proxy, 420
  - spot::tgba\_tba\_proxy, 440
- format\_tgba\_parse\_errors
  - tgba\_io, 36
- formula\_to\_bdd
  - spot, 81
- FourthStep
  - spot::minato\_isop::local\_vars, 245
- free\_anonymous\_list\_of
  - spot::bdd\_dict, 112
- free\_anonymous\_list\_of\_type
  - spot::bdd\_dict, 108
- free\_count
  - spot::bdd\_allocator, 103
  - spot::bdd\_dict::anon\_free\_list, 116
  - spot::free\_list, 226
- free\_list\_type
  - spot::bdd\_allocator, 101
  - spot::bdd\_dict::anon\_free\_list, 115
  - spot::free\_list, 225
- free\_relation\_simulation
  - tgba\_reduction, 41
- fv\_map
  - spot::bdd\_dict, 108
- G
  - spot::ltl::unop, 456
- g0
  - spot::minato\_isop::local\_vars, 246
- g1
  - spot::minato\_isop::local\_vars, 246
- generate
  - spot::ltl::random\_ltl, 292
- get
  - spot::acss\_statistics, 89
  - spot::ars\_statistics, 91

- spot::couvreur99\_check, 146
- spot::couvreur99\_check\_result, 151
- spot::couvreur99\_check\_shy, 158
- spot::ec\_statistics, 181
- spot::option\_map, 264
- spot::unsigned\_statistics, 460
- get\_acc
  - spot::duplicator\_node, 172
  - spot::duplicator\_node\_delayed, 177
- get\_acceptance\_condition
  - spot::tgba\_explicit, 369
  - spot::tgba\_reduc, 408
- get\_acceptance\_condition\_visited
  - spot::spoiler\_node\_delayed, 316
- get\_core\_data
  - spot::tgba\_bdd\_concrete, 349
  - spot::tgba\_bdd\_concrete\_factory, 354
  - spot::tgba\_bdd\_factory, 361
- get\_delayed\_relation\_simulation
  - tgba\_reduction, 42
- get\_dict
  - spot::tgba, 342
  - spot::tgba\_bdd\_concrete, 348
  - spot::tgba\_bdd\_concrete\_factory, 354
  - spot::tgba\_explicit, 368
  - spot::tgba\_product, 380
  - spot::tgba\_reduc, 407
  - spot::tgba\_sba\_proxy, 420
  - spot::tgba\_tba\_proxy, 440
- get\_direct\_relation\_simulation
  - tgba\_reduction, 42
- get\_duplicator\_node
  - spot::duplicator\_node, 173
  - spot::duplicator\_node\_delayed, 178
  - spot::spoiler\_node, 313
  - spot::spoiler\_node\_delayed, 316
- get\_err
  - spot::gspn\_exception, 227
- get\_index
  - spot::numbered\_state\_heap\_const\_iterator, 257
- get\_init\_bdd
  - spot::tgba\_bdd\_concrete, 348
- get\_init\_state
  - spot::tgba, 341
  - spot::tgba\_bdd\_concrete, 347
  - spot::tgba\_explicit, 368
  - spot::tgba\_product, 379
  - spot::tgba\_reduc, 406
  - spot::tgba\_sba\_proxy, 419
  - spot::tgba\_tba\_proxy, 439
- get\_label
  - spot::duplicator\_node, 172
  - spot::duplicator\_node\_delayed, 177
- get\_lead\_2\_acc\_all
  - spot::duplicator\_node\_delayed, 177
  - spot::spoiler\_node\_delayed, 316
- get\_nb\_succ
  - spot::duplicator\_node, 173
  - spot::duplicator\_node\_delayed, 177
  - spot::spoiler\_node, 312
  - spot::spoiler\_node\_delayed, 316
- get\_pair
  - spot::duplicator\_node, 173
  - spot::duplicator\_node\_delayed, 178
  - spot::spoiler\_node, 313
  - spot::spoiler\_node\_delayed, 317
- get\_progress\_measure
  - spot::duplicator\_node\_delayed, 177
  - spot::spoiler\_node\_delayed, 316
- get\_prop\_map
  - spot::ltl::declarative\_environment, 164
  - spot::ltl::read\_only\_environment, 296
- get\_relation
  - spot::parity\_game\_graph, 269
  - spot::parity\_game\_graph\_delayed, 276
  - spot::parity\_game\_graph\_direct, 282
- get\_removed\_components
  - spot::couvreur99\_check, 145
  - spot::couvreur99\_check\_shy, 157
- get\_spoiler\_node
  - spot::duplicator\_node, 173
  - spot::duplicator\_node\_delayed, 178
  - spot::spoiler\_node, 313
  - spot::spoiler\_node\_delayed, 316
- get\_state
  - spot::numbered\_state\_heap\_const\_iterator, 257
  - spot::state\_evtgba\_explicit, 327
  - spot::state\_explicit, 330
- get\_vmsize
  - spot::couvreur99\_check, 145
  - spot::couvreur99\_check\_shy, 157
- get\_where
  - spot::gspn\_exception, 227
- group2\_
  - spot::couvreur99\_check\_shy, 158
- group\_
  - spot::couvreur99\_check\_shy, 158
- gspn/ Directory Reference, 58
- gspn/common.hh, 465
- gspn/gspn.hh, 466
- gspn/ssp.hh, 467
- gspn\_exception
  - spot::gspn\_exception, 227
- gspn\_interface
  - spot::gspn\_interface, 229
- gspn\_ssp\_interface

- spot::gspn\_ssp\_interface, 231
- h
  - spot::couvreur99\_check\_status, 162
  - spot::numbered\_state\_heap\_hash\_map, 261
- h\_
  - spot::tgba\_reduc, 409
- has\_acceptance\_condition
  - spot::tgba\_explicit, 367
  - spot::tgba\_reduc, 406
- has\_state
  - spot::connected\_component\_hash\_set, 132
  - spot::explicit\_connected\_component, 217
- hash
  - spot::ltl::atomic\_prop, 96
  - spot::ltl::binop, 126
  - spot::ltl::constant, 139
  - spot::ltl::formula, 221
  - spot::ltl::multop, 253
  - spot::ltl::ref\_formula, 300
  - spot::ltl::unop, 458
  - spot::state, 322
  - spot::state\_bdd, 325
  - spot::state\_evtgba\_explicit, 327
  - spot::state\_explicit, 329
  - spot::state\_product, 332
- hash\_funcs
  - knuth32\_hash, 31
  - wang32\_hash, 31
- hash\_key\_
  - spot::ltl::atomic\_prop, 97
  - spot::ltl::binop, 127
  - spot::ltl::constant, 140
  - spot::ltl::formula, 222
  - spot::ltl::multop, 253
  - spot::ltl::ref\_formula, 301
  - spot::ltl::unop, 458
- hash\_type
  - spot::numbered\_state\_heap\_hash\_map, 260
- Hashing functions, 31
- height
  - ltlyy::stack, 321
  - sautyy::stack, 319
- i\_
  - spot::tgba\_explicit\_succ\_iterator, 375
- Implies
  - spot::ltl::binop, 124
- implies
  - spot::duplicator\_node, 172
  - spot::duplicator\_node\_delayed, 177
- implies\_acc
  - spot::duplicator\_node\_delayed, 177
- implies\_label
  - spot::duplicator\_node\_delayed, 177
- in
  - spot::evtgba\_explicit::state, 200
  - spot::evtgba\_explicit::transition, 201
- inc\_ars\_cycle\_states
  - spot::acss\_statistics, 89
  - spot::ars\_statistics, 91
  - spot::couvreur99\_check\_result, 151
- inc\_ars\_prefix\_states
  - spot::acss\_statistics, 88
  - spot::ars\_statistics, 91
  - spot::couvreur99\_check\_result, 151
- inc\_depth
  - spot::couvreur99\_check, 146
  - spot::couvreur99\_check\_shy, 158
  - spot::ec\_statistics, 181
- inc\_states
  - spot::couvreur99\_check, 146
  - spot::couvreur99\_check\_shy, 157
  - spot::ec\_statistics, 181
- inc\_transitions
  - spot::couvreur99\_check, 146
  - spot::couvreur99\_check\_shy, 158
  - spot::ec\_statistics, 181
- inc\_weight\_handler
  - spot::weight, 464
- incomp\_map
  - spot::ltl::language\_containment\_checker::record\_, 235
- incompatible
  - spot::ltl::language\_containment\_checker::record\_, 236
- incompatible\_
  - spot::ltl::language\_containment\_checker, 234
- index
  - spot::connected\_component\_hash\_set, 132
  - spot::explicit\_connected\_component, 218
  - spot::numbered\_state\_heap, 255, 256
  - spot::numbered\_state\_heap\_hash\_map, 261
  - spot::scc\_stack::connected\_component, 305
- info\_
  - spot::emptiness\_check\_instantiator, 188
- init\_
  - spot::tgba\_bdd\_concrete, 351
  - spot::tgba\_explicit, 371
  - spot::tgba\_reduc, 410
- init\_iter
  - spot::evtgba, 195
  - spot::evtgba\_explicit, 198
  - spot::evtgba\_product, 204
- init\_states\_
  - spot::evtgba\_explicit, 200
- initialize
  - ltlyy::location, 239

- ltlyy::position, 285
- sautyy::location, 237
- sautyy::position, 287
- spot::bdd\_allocator, 102
- spot::bdd\_dict, 111
- initialized
  - spot::bdd\_allocator, 103
  - spot::bdd\_dict, 112
- Input/Output of LTL formulae, 21
- Input/Output of TGBA, 35
- insert
  - spot::bdd\_allocator, 103
  - spot::bdd\_dict::anon\_free\_list, 116
  - spot::connected\_component\_hash\_set, 132
  - spot::explicit\_connected\_component, 217
  - spot::free\_list, 226
  - spot::numbered\_state\_heap, 256
  - spot::numbered\_state\_heap\_hash\_map, 261
- instance
  - spot::connected\_component\_hash\_set\_-factory, 134
  - spot::dotty\_decorator, 169
  - spot::ltl::atomic\_prop, 95
  - spot::ltl::binop, 125
  - spot::ltl::default\_environment, 167
  - spot::ltl::multop, 251
  - spot::ltl::unop, 457
  - spot::numbered\_state\_heap\_hash\_map\_-factory, 263
  - spot::symbol, 336
  - spot::tgba\_run\_dotty\_decorator, 415
- instance\_count
  - spot::ltl::atomic\_prop, 95
  - spot::ltl::binop, 126
  - spot::ltl::multop, 252
  - spot::ltl::unop, 457
  - spot::symbol, 336
- instances
  - spot::ltl::atomic\_prop, 97
  - spot::ltl::binop, 127
  - spot::ltl::multop, 253
  - spot::ltl::unop, 458
- instances\_
  - spot::symbol, 337
- instantiate
  - spot::emptiness\_check\_instantiator, 188
- is\_bare\_word
  - misc\_tools, 30
- is\_eventual
  - ltl\_misc, 27
- is\_FG
  - ltl\_misc, 27
- is\_GF
  - ltl\_misc, 28
- is\_not\_accepting
  - spot::tgba\_reduc, 405
- is\_registered\_acceptance\_variable
  - spot::bdd\_dict, 111
- is\_registered\_proposition
  - spot::bdd\_dict, 110
- is\_registered\_state
  - spot::bdd\_dict, 110
- is\_terminal
  - spot::tgba\_reduc, 404
- is\_universal
  - ltl\_misc, 28
- item\_type
  - spot::timer\_map, 445
- iterator
  - ltlyy::stack, 320
  - sautyy::stack, 318
  - spot::numbered\_state\_heap, 256
  - spot::numbered\_state\_heap\_hash\_map, 261
- knuth32\_hash
  - hash\_funcs, 31
- label
  - spot::evtgba\_explicit::transition, 201
  - spot::tgba\_run::step, 412
- label\_
  - spot::duplicator\_node, 173
  - spot::duplicator\_node\_delayed, 178
- language\_containment\_checker
  - spot::ltl::language\_containment\_checker, 234
- last
  - spot::loopless\_modular\_mixed\_radix\_gray\_-code, 242
- last\_support\_conditions\_input\_
  - spot::tgba, 343
- last\_support\_conditions\_output\_
  - spot::tgba, 343
- last\_support\_variables\_input\_
  - spot::tgba, 343
- last\_support\_variables\_output\_
  - spot::tgba, 343
- lbtt\_reachable
  - tgba\_io, 36
- lead\_2\_acc\_all\_
  - spot::duplicator\_node\_delayed, 178
  - spot::spoiler\_node\_delayed, 317
- left
  - spot::state\_product, 332
- left\_
  - spot::state\_product, 333
  - spot::tgba\_product, 382
  - spot::tgba\_succ\_iterator\_product, 435
- left\_acc\_complement\_

- spot::tgba\_product, 382
- left\_neg
  - spot::tgba\_succ\_iterator\_product, 435
- length
  - ltl\_misc, 28
- lift
  - spot::parity\_game\_graph, 270
  - spot::parity\_game\_graph\_delayed, 276
  - spot::parity\_game\_graph\_direct, 282
- line
  - ltlyy::position, 285
  - sautyy::position, 287
- lines
  - ltlyy::location, 239
  - ltlyy::position, 285
  - sautyy::location, 237
  - sautyy::position, 287
- link\_decl
  - spot::dotty\_decorator, 168
  - spot::tgba\_run\_dotty\_decorator, 415
- lnode\_pred
  - spot::duplicator\_node, 173
  - spot::duplicator\_node\_delayed, 178
  - spot::spoiler\_node, 313
  - spot::spoiler\_node\_delayed, 317
- lnode\_succ
  - spot::duplicator\_node, 173
  - spot::duplicator\_node\_delayed, 178
  - spot::spoiler\_node, 313
  - spot::spoiler\_node\_delayed, 317
- local\_vars
  - spot::minato\_isop::local\_vars, 245
- location
  - ltlyy::location, 239
  - sautyy::location, 237
- loopless\_modular\_mixed\_radix\_gray\_code
  - spot::loopless\_modular\_mixed\_radix\_gray\_code, 241
- LTL Abstract Syntax Tree, 20
- LTL environments, 20
- LTL formulae, 18
- ltl\_essential
  - clone, 19
  - destroy, 19
- ltl\_io
  - dotty, 22
  - dump, 22
  - format\_parse\_errors, 22
  - parse, 22
  - parse\_error, 22
  - parse\_error\_list, 22
  - to\_spin\_string, 23
  - to\_string, 23
- ltl\_misc
  - atomic\_prop\_collect, 27
  - atomic\_prop\_set, 27
  - is\_eventual, 27
  - is\_FG, 27
  - is\_GF, 28
  - is\_universal, 28
  - length, 28
  - syntactic\_implication, 28
  - syntactic\_implication\_neg, 28
- ltl\_rewriting
  - Reduce\_All, 25
  - Reduce\_Basics, 25
  - Reduce\_Containment\_Checks, 25
  - Reduce\_Containment\_Checks\_Stronger, 25
  - Reduce\_Eventuality\_And\_Universality, 25
  - Reduce\_None, 25
  - Reduce\_Syntactic\_Implications, 25
- ltl\_rewriting
  - basic\_reduce, 25
  - negative\_normal\_form, 25
  - reduce, 25
  - reduce\_options, 25
  - simplify\_f\_g, 26
  - unabbreviate\_logic, 26
- ltl\_to\_tgba\_fm
  - tgba\_ltl, 37
- ltl\_to\_tgba\_lacim
  - tgba\_ltl, 39
- ltlast/ Directory Reference, 59
- ltlast/allnodes.hh, 482
- ltlast/atomic\_prop.hh, 482
- ltlast/binop.hh, 483
- ltlast/constant.hh, 484
- ltlast/formula.hh, 485
- ltlast/multop.hh, 487
- ltlast/predecl.hh, 488
- ltlast/refformula.hh, 489
- ltlast/unop.hh, 489
- ltlast/visitor.hh, 490
- ltlenv/ Directory Reference, 60
- ltlenv/declenv.hh, 491
- ltlenv/defaultenv.hh, 492
- ltlenv/environment.hh, 492
- ltlenv/rodeco.hh, 493
- ltlparse/ Directory Reference, 60
- ltlparse/location.hh, 494
- ltlparse/position.hh, 496
- ltlparse/public.hh, 478
- ltlparse/stack.hh, 498
- ltlvisit/ Directory Reference, 61
- ltlvisit/apcollect.hh, 499
- ltlvisit/basicreduce.hh, 499
- ltlvisit/clone.hh, 500
- ltlvisit/contain.hh, 501

- ltlvisit/destroy.hh, 502
- ltlvisit/dotty.hh, 473
- ltlvisit/dump.hh, 503
- ltlvisit/length.hh, 503
- ltlvisit/lunabbrev.hh, 504
- ltlvisit/nenofrm.hh, 504
- ltlvisit/postfix.hh, 505
- ltlvisit/randomltl.hh, 505
- ltlvisit/reduce.hh, 506
- ltlvisit/simpfg.hh, 507
- ltlvisit/syntimpl.hh, 508
- ltlvisit/tostring.hh, 508
- ltlvisit/tunabbrev.hh, 509
- ltlyy, 64
  - operator+, 65
  - operator+=, 65
  - operator-, 65
  - operator-=, 65
  - operator<<, 65, 66
- ltlyy::location, 238
  - begin, 239
  - columns, 239
  - end, 239
  - initialize, 239
  - lines, 239
  - location, 239
  - step, 239
- ltlyy::position, 284
  - column, 286
  - columns, 285
  - filename, 285
  - initialize, 285
  - line, 285
  - lines, 285
  - position, 285
- ltlyy::slice, 308
  - operator[], 309
  - range\_, 309
  - slice, 309
  - stack\_, 309
- ltlyy::stack, 319
  - begin, 321
  - const\_iterator, 320
  - end, 321
  - height, 321
  - iterator, 320
  - operator[], 321
  - pop, 321
  - push, 321
  - seq\_, 321
  - stack, 321
- lvarnum
  - spot::bdd\_allocator, 103
  - spot::bdd\_dict, 112
- m
  - spot::weight, 465
- m\_
  - spot::barand, 98
  - spot::loopless\_modular\_mixed\_radix\_gray\_code, 242
- magic\_search
  - emptiness\_check\_algorithms, 53
- mainpage.dox, 465
- map
  - spot::ltl::atomic\_prop, 95
  - spot::ltl::binop, 124
  - spot::ltl::multop, 250
  - spot::ltl::unop, 456
  - spot::symbol, 336
- map\_
  - spot::tgba\_run\_dotty\_decorator, 415
- match
  - spot::bfs\_steps, 119
  - spot::duplicator\_node, 172
  - spot::duplicator\_node\_delayed, 177
- max\_acceptance\_conditions
  - spot::emptiness\_check\_instantiator, 188
- max\_depth
  - spot::couvereur99\_check, 146
  - spot::couvereur99\_check\_shy, 158
  - spot::ec\_statistics, 181
- max\_depth\_
  - spot::ec\_statistics, 182
- memusage
  - spot, 81
- merge\_state
  - spot::tgba\_reduc, 404
- merge\_state\_delayed
  - spot::tgba\_reduc, 404
- merge\_transitions
  - spot::tgba\_explicit, 368
  - spot::tgba\_reduc, 406
- min\_acceptance\_conditions
  - spot::emptiness\_check\_instantiator, 188
- min\_n
  - spot::ltl::random\_ltl::op\_proba, 294
- minato\_isop
  - spot::minato\_isop, 244
- misc/ Directory Reference, 61
- misc/bareword.hh, 509
- misc/bddalloc.hh, 510
- misc/bddlt.hh, 511
- misc/escape.hh, 511
- misc/freelist.hh, 512
- misc/hash.hh, 513
- misc/hashfunc.hh, 514
- misc/ltstr.hh, 514
- misc/memusage.hh, 515

- misc/minato.hh, 515
- misc/modgray.hh, 516
- misc/optionmap.hh, 516
- misc/random.hh, 517
- misc/timer.hh, 518
- misc/version.hh, 519
- misc\_tools
  - escape\_str, 30
  - is\_bare\_word, 30
  - quote\_unless\_bare\_word, 30
  - version, 30
- Miscellaneous algorithms for LTL formulae, 26
- Miscellaneous algorithms on TGBA, 42
- Miscellaneous helper algorithms, 29
- mrnd
  - random, 32
- multop
  - spot::ltl::multop, 251
- n
  - spot::couvreur99\_check\_shy::todo\_item, 160
- n\_
  - spot::barand, 98
  - spot::loopless\_modular\_mixed\_radix\_gray\_code, 242
- name
  - spot::ltl::atomic\_prop, 95
  - spot::ltl::declarative\_environment, 164
  - spot::ltl::default\_environment, 166
  - spot::ltl::environment, 193
  - spot::ltl::random\_ltl::op\_proba, 294
  - spot::ltl::read\_only\_environment, 296
  - spot::symbol, 336
- name\_
  - spot::ltl::atomic\_prop, 97
  - spot::symbol, 337
- name\_state\_map\_
  - spot::evtgba\_explicit, 199
  - spot::tgba\_explicit, 371
  - spot::tgba\_reduc, 410
- nb\_node\_parity\_game
  - spot::parity\_game\_graph, 271
  - spot::parity\_game\_graph\_delayed, 278
  - spot::parity\_game\_graph\_direct, 284
- nb\_set\_acc\_cond
  - spot::parity\_game\_graph\_delayed, 276
  - spot::tgba\_reduc, 405
- neg\_acceptance\_conditions
  - spot::tgba, 343
  - spot::tgba\_bdd\_concrete, 349
  - spot::tgba\_explicit, 369
  - spot::tgba\_product, 381
  - spot::tgba\_reduc, 407
  - spot::tgba\_sba\_proxy, 421
  - spot::tgba\_tba\_proxy, 441
- neg\_acceptance\_conditions\_
  - spot::tgba\_explicit, 371
  - spot::tgba\_product, 382
  - spot::tgba\_reduc, 410
- neg\_all\_acc
  - spot::weight, 465
- neg\_contained
  - spot::ltl::language\_containment\_checker, 234
- negacc\_set
  - spot::tgba\_bdd\_core\_data, 360
- negative\_normal\_form
  - ltl\_rewriting, 25
- never\_claim\_reachable
  - tgba\_io, 36
- next
  - spot::evtgba\_iterator, 202
  - spot::loopless\_modular\_mixed\_radix\_gray\_code, 242
  - spot::minato\_isop, 244
  - spot::numbered\_state\_heap\_const\_iterator, 257
  - spot::tgba\_explicit\_succ\_iterator, 374
  - spot::tgba\_succ\_iterator, 424
  - spot::tgba\_succ\_iterator\_concrete, 429
  - spot::tgba\_succ\_iterator\_product, 433
- next\_non\_false\_
  - spot::tgba\_succ\_iterator\_product, 434
- next\_set
  - spot::tgba\_bdd\_core\_data, 360
- next\_state
  - spot::evtgba\_reachable\_iterator, 208
  - spot::evtgba\_reachable\_iterator\_breadth\_first, 211
  - spot::evtgba\_reachable\_iterator\_depth\_first, 215
  - spot::parity\_game\_graph, 270
  - spot::parity\_game\_graph\_delayed, 277
  - spot::parity\_game\_graph\_direct, 283
  - spot::tgba\_reachable\_iterator, 386
  - spot::tgba\_reachable\_iterator\_breadth\_first, 390
  - spot::tgba\_reachable\_iterator\_depth\_first, 394
  - spot::tgba\_reduc, 409
- next\_to\_now
  - spot::bdd\_dict, 112
- non\_one\_radixes\_
  - spot::loopless\_modular\_mixed\_radix\_gray\_code, 243
- Not
  - spot::ltl::unop, 456
- not\_win
  - spot::duplicator\_node, 173
  - spot::duplicator\_node\_delayed, 178



- spot::spoiler\_node, 313
- spot::spoiler\_node\_delayed, 317
- notacc\_set
  - spot::tgba\_bdd\_core\_data, 360
- notnext\_set
  - spot::tgba\_bdd\_core\_data, 360
- notnow\_set
  - spot::tgba\_bdd\_core\_data, 360
- notvar\_set
  - spot::tgba\_bdd\_core\_data, 360
- now\_formula\_map
  - spot::bdd\_dict, 111
- now\_map
  - spot::bdd\_dict, 111
- now\_set
  - spot::tgba\_bdd\_core\_data, 359
- now\_to\_next
  - spot::bdd\_dict, 112
- nownext\_set
  - spot::tgba\_bdd\_core\_data, 360
- nrand
  - random, 32
- ns\_map
  - spot::evtgba\_explicit, 198
  - spot::tgba\_explicit, 367
  - spot::tgba\_reduc, 402
- nth
  - spot::ltl::multop, 252
- num
  - spot::couvreur99\_check\_shy, 158
- num\_
  - spot::duplicator\_node, 173
  - spot::duplicator\_node\_delayed, 178
  - spot::spoiler\_node, 313
  - spot::spoiler\_node\_delayed, 317
- num\_acc\_
  - spot::tgba, 343
- number\_of\_acceptance\_conditions
  - spot::tgba, 343
  - spot::tgba\_bdd\_concrete, 350
  - spot::tgba\_explicit, 370
  - spot::tgba\_product, 382
  - spot::tgba\_reduc, 408
  - spot::tgba\_sba\_proxy, 422
  - spot::tgba\_tba\_proxy, 442
- numbered\_state\_heap\_hash\_map\_factory
  - spot::numbered\_state\_heap\_hash\_map\_-factory, 263
- o\_
  - spot::couvreur99\_check, 146
  - spot::couvreur99\_check\_result, 151
  - spot::couvreur99\_check\_shy, 159
  - spot::emptiness\_check, 186
  - spot::emptiness\_check\_instantiator, 188
  - spot::emptiness\_check\_result, 192
- onepass\_
  - spot::couvreur99\_check\_shy, 158
- op
  - spot::ltl::binop, 125
  - spot::ltl::multop, 252
  - spot::ltl::unop, 457
- op\_
  - spot::evtgba\_product, 205
  - spot::ltl::binop, 127
  - spot::ltl::multop, 253
  - spot::ltl::unop, 458
- op\_name
  - spot::ltl::binop, 125
  - spot::ltl::multop, 252
  - spot::ltl::unop, 457
- operator const symbol \*
  - spot::rsymbol, 302
- operator!=
  - spot::rsymbol, 302
  - spot::unsigned\_statistics\_copy, 461
- operator()
  - spot::bdd\_less\_than, 117
  - spot::char\_ptr\_less\_than, 127
  - spot::ltl::formula\_ptr\_hash, 222
  - spot::ltl::formula\_ptr\_less\_than, 223
  - spot::ltl::multop::paircmp, 254
  - spot::ptr\_hash, 290
  - spot::state\_ptr\_equal, 333
  - spot::state\_ptr\_hash, 334
  - spot::state\_ptr\_less\_than, 335
  - spot::string\_hash, 335
- operator+
  - ltlyy, 65
  - sautyy, 67
- operator+=
  - ltlyy, 65
  - sautyy, 67
  - spot::weight, 464
- operator-
  - ltlyy, 65
  - sautyy, 67
  - spot::weight, 464
- operator-=
  - ltlyy, 65
  - sautyy, 67
  - spot::weight, 464
- operator<
  - spot::rsymbol, 302
- operator<<
  - ltlyy, 65, 66
  - sautyy, 67, 68
  - spot, 82



- spot::option\_map, 265
- spot::weight, 465
- operator=
  - spot::bdd\_dict, 111
  - spot::rsymbol, 302
  - spot::tgba\_bdd\_concrete, 349
  - spot::tgba\_bdd\_core\_data, 358
  - spot::tgba\_explicit, 369
  - spot::tgba\_product, 381
  - spot::tgba\_run, 411
  - spot::tgba\_tba\_proxy, 441
- operator==
  - spot::rsymbol, 302
  - spot::unsigned\_statistics\_copy, 461
- operator[]
  - ltlyy::slice, 309
  - ltlyy::stack, 321
  - sautyy::slice, 310
  - sautyy::stack, 319
  - spot::option\_map, 264, 265
- options
  - spot::couvreur99\_check, 145
  - spot::couvreur99\_check\_result, 150
  - spot::couvreur99\_check\_shy, 157
  - spot::emptiness\_check, 185
  - spot::emptiness\_check\_instantiator, 188
  - spot::emptiness\_check\_result, 191
- options\_
  - spot::option\_map, 265
- options\_updated
  - spot::couvreur99\_check, 145
  - spot::couvreur99\_check\_result, 150
  - spot::couvreur99\_check\_shy, 157
  - spot::emptiness\_check, 186
  - spot::emptiness\_check\_result, 192
- Or
  - spot::ltl::multop, 251
- out
  - spot::evtgba\_explicit::state, 200
  - spot::evtgba\_explicit::transition, 201
- pair
  - spot::ltl::atomic\_prop, 95
  - spot::ltl::binop, 124
  - spot::ltl::multop, 250
  - spot::ltl::unop, 456
- pairf
  - spot::ltl::binop, 124
- parity\_game\_graph
  - spot::parity\_game\_graph, 269
- parity\_game\_graph\_delayed
  - spot::parity\_game\_graph\_delayed, 276
- parity\_game\_graph\_direct
  - spot::parity\_game\_graph\_direct, 282
- parse
  - ltl\_io, 22
- parse\_error
  - ltl\_io, 22
- parse\_error\_list
  - ltl\_io, 22
- parse\_options
  - spot::couvreur99\_check, 145
  - spot::couvreur99\_check\_result, 150
  - spot::couvreur99\_check\_shy, 157
  - spot::emptiness\_check, 185
  - spot::emptiness\_check\_result, 191
  - spot::ltl::random\_ltl, 292
  - spot::option\_map, 264
- pm
  - spot::weight, 465
- pop
  - ltlyy::stack, 321
  - sautyy::stack, 319
  - spot::scc\_stack, 303
- poprem\_
  - spot::couvreur99\_check, 146
  - spot::couvreur99\_check\_shy, 158
- pos
  - spot::couvreur99\_check\_shy, 158
- pos\_lenght\_pair
  - spot::bdd\_allocator, 101
  - spot::bdd\_dict::anon\_free\_list, 115
  - spot::free\_list, 225
- position
  - ltlyy::position, 285
  - sautyy::position, 287
- postfix\_visitor
  - spot::ltl::postfix\_visitor, 289
- prand
  - random, 32
- pred\_iter
  - spot::evtgba, 195
  - spot::evtgba\_explicit, 198, 199
  - spot::evtgba\_product, 204
- prefix
  - spot::tgba\_run, 412
- prefix\_states\_
  - spot::ars\_statistics, 91
- print
  - spot::parity\_game\_graph, 269
  - spot::parity\_game\_graph\_delayed, 276
  - spot::parity\_game\_graph\_direct, 282
  - spot::timer\_map, 446
- print\_stats
  - spot::couvreur99\_check, 144
  - spot::couvreur99\_check\_result, 150
  - spot::couvreur99\_check\_shy, 156
  - spot::couvreur99\_check\_status, 162

- spot::emptiness\_check, 186
- print\_tgba\_run
  - tgba\_run, 54
- proba
  - spot::ltl::random\_ltl::op\_proba, 294
- proba\_
  - spot::ltl::random\_ltl, 293
- proba\_2\_
  - spot::ltl::random\_ltl, 293
- process\_link
  - spot::evtgba\_reachable\_iterator, 208
  - spot::evtgba\_reachable\_iterator\_breadth\_first, 212
  - spot::evtgba\_reachable\_iterator\_depth\_first, 215
  - spot::parity\_game\_graph, 269, 270
  - spot::parity\_game\_graph\_delayed, 277
  - spot::parity\_game\_graph\_direct, 283
  - spot::tgba\_reachable\_iterator, 386
  - spot::tgba\_reachable\_iterator\_breadth\_first, 391
  - spot::tgba\_reachable\_iterator\_depth\_first, 395
  - spot::tgba\_reduc, 404, 409
- process\_state
  - spot::evtgba\_reachable\_iterator, 208
  - spot::evtgba\_reachable\_iterator\_breadth\_first, 212
  - spot::evtgba\_reachable\_iterator\_depth\_first, 215
  - spot::parity\_game\_graph, 269
  - spot::parity\_game\_graph\_delayed, 277
  - spot::parity\_game\_graph\_direct, 283
  - spot::tgba\_reachable\_iterator, 386
  - spot::tgba\_reachable\_iterator\_breadth\_first, 391
  - spot::tgba\_reachable\_iterator\_depth\_first, 395
  - spot::tgba\_reduc, 403
- product
  - tgba\_algorithms, 34
- progress\_measure\_
  - spot::duplicator\_node\_delayed, 178
  - spot::spoiler\_node\_delayed, 317
- project\_state
  - spot::tgba, 342
  - spot::tgba\_bdd\_concrete, 350
  - spot::tgba\_explicit, 370
  - spot::tgba\_product, 380
  - spot::tgba\_reduc, 408
  - spot::tgba\_sba\_proxy, 420
  - spot::tgba\_tba\_proxy, 440
- project\_tgba\_run
  - tgba\_run, 54
- prop\_map
  - spot::ltl::declarative\_environment, 164
  - spot::ltl::read\_only\_environment, 296
- props\_
  - spot::ltl::declarative\_environment, 165
  - spot::ltl::read\_only\_environment, 297
- prune
  - spot::duplicator\_node, 173
  - spot::duplicator\_node\_delayed, 177
  - spot::spoiler\_node, 312
  - spot::spoiler\_node\_delayed, 316
- prune\_acc
  - spot::tgba\_reduc, 403
- prune\_scc
  - spot::tgba\_reduc, 403
- push
  - ltlyy::stack, 321
  - sautyy::stack, 319
  - spot::scc\_stack, 303
- q
  - spot::couvreur99\_check\_shy::todo\_item, 160
- quote\_unless\_bare\_word
  - misc\_tools, 30
- quotient\_state
  - spot::tgba\_reduc, 403
- R
  - spot::ltl::binop, 125
- rand
  - spot::barand, 98
- random
  - bmrand, 32
  - drand, 32
  - mrnd, 32
  - nrnd, 32
  - prand, 32
  - rrand, 33
  - srnd, 33
- Random functions, 31
- random\_graph
  - tgba\_misc, 43
- random\_ltl
  - spot::ltl::random\_ltl, 292
- range\_
  - ltlyy::slice, 309
  - sautyy::slice, 310
- recurse
  - spot::ltl::clone\_visitor, 130
  - spot::ltl::simplify\_f\_g\_visitor, 307
  - spot::ltl::unabbreviate\_logic\_visitor, 449
  - spot::ltl::unabbreviate\_ltl\_visitor, 452
- redirect\_transition
  - spot::tgba\_reduc, 404
- reduc\_tgba\_sim
  - tgba\_reduction, 42

- reduce
  - ltl\_rewriting, 25
- Reduce\_All
  - ltl\_rewriting, 25
  - tgba\_reduction, 41
- Reduce\_Basics
  - ltl\_rewriting, 25
- Reduce\_Containment\_Checks
  - ltl\_rewriting, 25
- Reduce\_Containment\_Checks\_Stronger
  - ltl\_rewriting, 25
- Reduce\_Eventuality\_And\_Universality
  - ltl\_rewriting, 25
- Reduce\_None
  - ltl\_rewriting, 25
  - tgba\_reduction, 41
- reduce\_options
  - ltl\_rewriting, 25
- Reduce\_quotient\_Del\_Sim
  - tgba\_reduction, 41
- Reduce\_quotient\_Dir\_Sim
  - tgba\_reduction, 41
- reduce\_run
  - tgba\_run, 55
- Reduce\_Scc
  - tgba\_reduction, 41
- Reduce\_Syntactic\_Implications
  - ltl\_rewriting, 25
- reduce\_tau03
  - spot::ltl, 85
- reduce\_tgba\_options
  - tgba\_reduction, 41
- Reduce\_transition\_Del\_Sim
  - tgba\_reduction, 41
- Reduce\_transition\_Dir\_Sim
  - tgba\_reduction, 41
- ref
  - spot::ltl::atomic\_prop, 96
  - spot::ltl::binop, 126
  - spot::ltl::constant, 139
  - spot::ltl::formula, 221
  - spot::ltl::multop, 252
  - spot::ltl::ref\_formula, 300
  - spot::ltl::unop, 457
  - spot::symbol, 337
- ref\_
  - spot::ltl::atomic\_prop, 96
  - spot::ltl::binop, 126
  - spot::ltl::constant, 139
  - spot::ltl::formula, 221
  - spot::ltl::multop, 252
  - spot::ltl::ref\_formula, 299
  - spot::ltl::unop, 457
- ref\_count\_
  - spot::ltl::atomic\_prop, 96
  - spot::ltl::binop, 126
  - spot::ltl::multop, 252
  - spot::ltl::ref\_formula, 299
  - spot::ltl::unop, 457
  - spot::symbol, 337
- ref\_counter\_
  - spot::ltl::ref\_formula, 300
- ref\_formula
  - spot::ltl::ref\_formula, 299
- ref\_set
  - spot::bdd\_dict, 108
- refs\_
  - spot::symbol, 337
- register\_acceptance\_variable
  - spot::bdd\_dict, 109
- register\_acceptance\_variables
  - spot::bdd\_dict, 110
- register\_all\_variables\_of
  - spot::bdd\_dict, 110
- register\_anonymous\_variables
  - spot::bdd\_dict, 110
- register\_clone\_acc
  - spot::bdd\_dict, 110
- register\_formula\_
  - spot::ltl::language\_containment\_checker, 234
- register\_n
  - spot::bdd\_allocator, 102
  - spot::bdd\_dict::anon\_free\_list, 116
  - spot::free\_list, 226
- register\_proposition
  - spot::bdd\_dict, 109
- register\_propositions
  - spot::bdd\_dict, 109
- register\_state
  - spot::bdd\_dict, 109
- relation
  - spot::tgba\_bdd\_core\_data, 359
- release\_n
  - spot::bdd\_allocator, 102
  - spot::bdd\_dict::anon\_free\_list, 116
  - spot::free\_list, 226
- release\_variables
  - spot::bdd\_allocator, 102
  - spot::bdd\_dict, 111
- rem
  - spot::connected\_component\_hash\_set, 133
  - spot::explicit\_connected\_component, 218
  - spot::scc\_stack, 304
  - spot::scc\_stack::connected\_component, 305
- remove
  - spot::bdd\_allocator, 103
  - spot::bdd\_dict::anon\_free\_list, 116
  - spot::free\_list, 226

- remove\_acc
  - spot::tgba\_reduc, 405
- remove\_component
  - spot::couvreur99\_check, 145
  - spot::couvreur99\_check\_shy, 157
  - spot::tgba\_reduc, 405
- remove\_predecessor\_state
  - spot::tgba\_reduc, 404
- remove\_scc
  - spot::tgba\_reduc, 405
- remove\_state
  - spot::tgba\_reduc, 404
- removed\_components
  - spot::couvreur99\_check, 146
  - spot::couvreur99\_check\_shy, 158
- replay\_tgba\_run
  - tgba\_run, 55
- require
  - spot::ltd::declarative\_environment, 164
  - spot::ltd::default\_environment, 166
  - spot::ltd::environment, 193
  - spot::ltd::read\_only\_environment, 296
- result
  - spot::couvreur99\_check, 145
  - spot::couvreur99\_check\_shy, 156
  - spot::ltd::clone\_visitor, 130
  - spot::ltd::simplify\_f\_g\_visitor, 307
  - spot::ltd::unabbreviate\_logic\_visitor, 449
  - spot::ltd::unabbreviate\_ltl\_visitor, 453
- result\_
  - spot::ltd::clone\_visitor, 130
  - spot::ltd::simplify\_f\_g\_visitor, 308
  - spot::ltd::unabbreviate\_logic\_visitor, 450
  - spot::ltd::unabbreviate\_ltl\_visitor, 453
- ret\_
  - spot::minato\_isop, 244
- Rewriting LTL formulae, 24
- right
  - spot::state\_product, 332
- right\_
  - spot::state\_product, 333
  - spot::tgba\_product, 382
  - spot::tgba\_succ\_iterator\_product, 435
- right\_acc\_complement\_
  - spot::tgba\_product, 382
- right\_common\_acc\_
  - spot::tgba\_product, 382
  - spot::tgba\_succ\_iterator\_product, 435
- right\_neg\_
  - spot::tgba\_succ\_iterator\_product, 435
- root
  - spot::couvreur99\_check\_status, 162
- root\_
  - spot::tgba\_reduc, 409
- rrand
  - random, 33
- rsymbol
  - spot::rsymbol, 302
- rsymbol\_set
  - spot, 78
- run
  - spot::evtgba\_reachable\_iterator, 208
  - spot::evtgba\_reachable\_iterator\_breadth\_first, 211
  - spot::evtgba\_reachable\_iterator\_depth\_first, 215
  - spot::parity\_game\_graph, 270
  - spot::parity\_game\_graph\_delayed, 277
  - spot::parity\_game\_graph\_direct, 283
  - spot::tgba\_reachable\_iterator, 385
  - spot::tgba\_reachable\_iterator\_breadth\_first, 391
  - spot::tgba\_reachable\_iterator\_depth\_first, 395
  - spot::tgba\_reduc, 409
- run\_
  - spot::couvreur99\_check\_result, 151
  - spot::tgba\_run\_dotty\_decorator, 415
- s
  - spot::couvreur99\_check\_shy::successor, 160
  - spot::couvreur99\_check\_shy::todo\_item, 160
  - spot::scc\_stack, 304
  - spot::tgba\_run::step, 412
- s\_
  - spot::barand, 98
  - spot::loopless\_modular\_mixed\_radix\_gray\_code, 243
  - spot::rsymbol, 302
  - spot::tgba\_explicit\_succ\_iterator, 375
- s\_v
  - tgba\_reduction, 41
- safe
  - spot::couvreur99\_check, 145
  - spot::couvreur99\_check\_shy, 157
  - spot::emptiness\_check, 185
- sautparse/ Directory Reference, 62
- sautparse/location.hh, 495
- sautparse/position.hh, 497
- sautparse/stack.hh, 498
- sautyy, 66
  - operator+, 67
  - operator+=, 67
  - operator-, 67
  - operator=, 67
  - operator<<, 67, 68
- sautyy::location, 236
  - begin, 237
  - columns, 237

- end, 237
- initialize, 237
- lines, 237
- location, 237
- step, 237
- sauty::position, 286
  - column, 287
  - columns, 287
  - filename, 287
  - initialize, 287
  - line, 287
  - lines, 287
  - position, 287
- sauty::slice, 309
  - operator[], 310
  - range\_, 310
  - slice, 310
  - stack\_, 310
- sauty::stack, 317
  - begin, 319
  - const\_iterator, 318
  - end, 319
  - height, 319
  - iterator, 318
  - operator[], 319
  - pop, 319
  - push, 319
  - seq\_, 319
  - stack, 319
- sc\_
  - spot::duplicator\_node, 173
  - spot::duplicator\_node\_delayed, 178
  - spot::spoiler\_node, 313
  - spot::spoiler\_node\_delayed, 317
- scc\_computed\_
  - spot::tgba\_reduc, 409
- se05
  - emptiness\_check\_algorithms, 53
- search
  - spot::bfs\_steps, 119
- second
  - spot::ltl::binop, 125
- second\_
  - spot::ltl::binop, 127
- SecondStep
  - spot::minato\_isop::local\_vars, 245
- seen
  - spot::evtgba\_reachable\_iterator, 209
  - spot::evtgba\_reachable\_iterator\_breadth\_first, 212
  - spot::evtgba\_reachable\_iterator\_depth\_first, 216
  - spot::parity\_game\_graph, 271
  - spot::parity\_game\_graph\_delayed, 278
  - spot::parity\_game\_graph\_direct, 284
  - spot::tgba\_reachable\_iterator, 387
  - spot::tgba\_reachable\_iterator\_breadth\_first, 392
  - spot::tgba\_reachable\_iterator\_depth\_first, 396
  - spot::tgba\_reduc, 410
- seen\_
  - spot::duplicator\_node\_delayed, 178
  - spot::spoiler\_node\_delayed, 317
  - spot::tgba\_reduc, 410
- seen\_map
  - spot::evtgba\_reachable\_iterator, 207
  - spot::evtgba\_reachable\_iterator\_breadth\_first, 211
  - spot::evtgba\_reachable\_iterator\_depth\_first, 214
  - spot::parity\_game\_graph, 269
  - spot::parity\_game\_graph\_delayed, 275
  - spot::parity\_game\_graph\_direct, 282
  - spot::tgba\_reachable\_iterator, 385
  - spot::tgba\_reachable\_iterator\_breadth\_first, 390
  - spot::tgba\_reachable\_iterator\_depth\_first, 394
  - spot::tgba\_reduc, 402
- seq\_
  - ltlyy::stack, 321
  - sauty::stack, 319
- set
  - spot::option\_map, 265
  - spot::unsigned\_statistics\_copy, 461
- set\_init\_state
  - spot::evtgba\_explicit, 199
  - spot::tgba\_bdd\_concrete, 347
  - spot::tgba\_explicit, 367
  - spot::tgba\_reduc, 405
- set\_key\_
  - spot::ltl::atomic\_prop, 96
  - spot::ltl::binop, 126
  - spot::ltl::constant, 139
  - spot::ltl::formula, 222
  - spot::ltl::multop, 253
  - spot::ltl::ref\_formula, 300
  - spot::ltl::unop, 458
- set\_lead\_2\_acc\_all
  - spot::duplicator\_node\_delayed, 177
  - spot::spoiler\_node\_delayed, 316
- set\_states
  - spot::couvreur99\_check, 146
  - spot::couvreur99\_check\_shy, 157
  - spot::ec\_statistics, 181
- set\_type
  - spot::connected\_component\_hash\_set, 132
- set\_win
  - spot::duplicator\_node, 172

- spot::duplicator\_node\_delayed, 176
- spot::spoiler\_node, 312
- spot::spoiler\_node\_delayed, 316
- seteq
  - spot::unsigned\_statistics\_copy, 461
- setup
  - spot::ltl::random\_ltl::op\_proba, 294
- si\_
  - spot::tgba\_reduc, 410
- simplify\_f\_g
  - ltl\_rewriting, 26
- simplify\_f\_g\_visitor
  - spot::ltl::simplify\_f\_g\_visitor, 307
- simulation\_relation
  - spot, 78
- size
  - spot::ltl::multop, 251
  - spot::numbered\_state\_heap, 256
  - spot::numbered\_state\_heap\_hash\_map, 261
  - spot::scc\_stack, 304
- slice
  - ltlyy::slice, 309
  - sautyy::slice, 310
- sn\_map
  - spot::evtgba\_explicit, 198
  - spot::tgba\_explicit, 367
  - spot::tgba\_reduc, 402
- sn\_v
  - tgba\_reduction, 41
- sp\_map
  - spot::tgba\_reduc, 402
- spoiler\_node
  - spot::spoiler\_node, 312
- spoiler\_node\_delayed
  - spot::spoiler\_node\_delayed, 315
- spoiler\_vertice\_
  - spot::parity\_game\_graph, 271
  - spot::parity\_game\_graph\_delayed, 278
  - spot::parity\_game\_graph\_direct, 284
- spot, 68
  - bdd\_format\_accset, 78
  - bdd\_format\_formula, 78
  - bdd\_format\_sat, 78
  - bdd\_format\_set, 79
  - bdd\_print\_acc, 79
  - bdd\_print\_accset, 79
  - bdd\_print\_dot, 79
  - bdd\_print\_formula, 80
  - bdd\_print\_sat, 80
  - bdd\_print\_set, 80
  - bdd\_print\_table, 80
  - bdd\_to\_formula, 80
  - dotty\_reachable, 81
  - evtgba\_parse, 81
  - evtgba\_parse\_error, 78
  - evtgba\_parse\_error\_list, 78
  - evtgba\_save\_reachable, 81
  - format\_evtgba\_parse\_errors, 81
  - formula\_to\_bdd, 81
  - memusage, 81
  - operator<<, 82
  - rsymbol\_set, 78
  - simulation\_relation, 78
  - state\_couple, 78
  - symbol\_set, 78
  - tgba\_to\_evtgba, 82
- spot::acss\_statistics, 86
  - ~acss\_statistics, 88
  - acss\_states, 88
  - acss\_statistics, 88
  - ars\_cycle\_states, 89
  - ars\_prefix\_states, 88
  - get, 89
  - inc\_ars\_cycle\_states, 89
  - inc\_ars\_prefix\_states, 88
  - stats, 89
  - stats\_map, 88
  - unsigned\_fun, 88
- spot::ars\_statistics, 89
  - ars\_cycle\_states, 91
  - ars\_prefix\_states, 91
  - ars\_statistics, 91
  - cycle\_states\_, 91
  - get, 91
  - inc\_ars\_cycle\_states, 91
  - inc\_ars\_prefix\_states, 91
  - prefix\_states\_, 91
  - stats, 91
  - stats\_map, 90
  - unsigned\_fun, 90
- spot::barand, 97
  - barand, 98
  - m\_, 98
  - n\_, 98
  - rand, 98
  - s\_, 98
- spot::bdd\_allocator, 98
  - allocate\_variables, 102
  - bdd\_allocator, 102
  - dump\_free\_list, 102
  - extend, 102
  - extvarnum, 102
  - fl, 103
  - free\_count, 103
  - free\_list\_type, 101
  - initialize, 102
  - initialized, 103
  - insert, 103

- lvarnum, 103
- pos\_lenght\_pair, 101
- register\_n, 102
- release\_n, 102
- release\_variables, 102
- remove, 103
- spot::bdd\_dict, 103
  - ~bdd\_dict, 109
  - acc\_formula\_map, 112
  - acc\_map, 112
  - allocate\_variables, 111
  - assert\_emptiness, 111
  - bdd\_dict, 109
  - cc\_map, 108
  - clone\_counts, 112
  - dump, 111
  - free\_anonymous\_list\_of, 112
  - free\_anonymous\_list\_of\_type, 108
  - fv\_map, 108
  - initialize, 111
  - initialized, 112
  - is\_registered\_acceptance\_variable, 111
  - is\_registered\_proposition, 110
  - is\_registered\_state, 110
  - lvarnum, 112
  - next\_to\_now, 112
  - now\_formula\_map, 111
  - now\_map, 111
  - now\_to\_next, 112
  - operator=, 111
  - ref\_set, 108
  - register\_acceptance\_variable, 109
  - register\_acceptance\_variables, 110
  - register\_all\_variables\_of, 110
  - register\_anonymous\_variables, 110
  - register\_clone\_acc, 110
  - register\_proposition, 109
  - register\_propositions, 109
  - register\_state, 109
  - release\_variables, 111
  - unregister\_all\_my\_variables, 110
  - unregister\_variable, 110, 111
  - var\_formula\_map, 111
  - var\_map, 111
  - var\_refs, 112
  - vf\_map, 108
  - vr\_map, 108
- spot::bdd\_dict::anon\_free\_list, 112
  - anon\_free\_list, 116
  - dict\_, 117
  - dump\_free\_list, 116
  - extend, 116
  - fl, 117
  - free\_count, 116
  - free\_list\_type, 115
  - insert, 116
  - pos\_lenght\_pair, 115
  - register\_n, 116
  - release\_n, 116
  - remove, 116
- spot::bdd\_less\_than, 117
  - operator(), 117
- spot::bfs\_steps, 117
  - ~bfs\_steps, 119
  - a\_, 120
  - bfs\_steps, 119
  - filter, 119
  - finalize, 120
  - match, 119
  - search, 119
- spot::char\_ptr\_less\_than, 127
  - operator(), 127
- spot::connected\_component\_hash\_set, 130
  - ~connected\_component\_hash\_set, 132
  - condition, 133
  - has\_state, 132
  - index, 132
  - insert, 132
  - rem, 133
  - set\_type, 132
  - states, 132
- spot::connected\_component\_hash\_set\_factory, 133
  - ~connected\_component\_hash\_set\_factory, 134
  - build, 134
  - connected\_component\_hash\_set\_factory, 134
  - instance, 134
- spot::couvreur99\_check, 140
  - ~couvreur99\_check, 144
  - a\_, 146
  - automaton, 145
  - check, 144
  - couvreur99\_check, 144
  - dec\_depth, 146
  - depth, 146
  - ecs\_, 146
  - get, 146
  - get\_removed\_components, 145
  - get\_vmsize, 145
  - inc\_depth, 146
  - inc\_states, 146
  - inc\_transitions, 146
  - max\_depth, 146
  - o\_, 146
  - options, 145
  - options\_updated, 145
  - parse\_options, 145
  - poprem\_, 146

- print\_stats, 144
- remove\_component, 145
- removed\_components, 146
- result, 145
- safe, 145
- set\_states, 146
- states, 146
- statistics, 145
- stats, 146
- stats\_map, 144
- transitions, 146
- unsigned\_fun, 144
- spot::couvreur99\_check\_result, 147
  - a\_, 151
  - accepting\_cycle, 150
  - accepting\_run, 150
  - acss\_states, 150
  - ars\_cycle\_states, 151
  - ars\_prefix\_states, 151
  - automaton, 150
  - couvreur99\_check\_result, 150
  - ecs\_, 151
  - get, 151
  - inc\_ars\_cycle\_states, 151
  - inc\_ars\_prefix\_states, 151
  - o\_, 151
  - options, 150
  - options\_updated, 150
  - parse\_options, 150
  - print\_stats, 150
  - run\_, 151
  - statistics, 150
  - stats, 151
  - stats\_map, 149
  - unsigned\_fun, 149
- spot::couvreur99\_check\_shy, 151
  - ~couvreur99\_check\_shy, 156
  - a\_, 159
  - arc, 158
  - automaton, 157
  - check, 156
  - clear\_todo, 156
  - couvreur99\_check\_shy, 156
  - dec\_depth, 158
  - depth, 158
  - ecs\_, 158
  - find\_state, 156
  - get, 158
  - get\_removed\_components, 157
  - get\_vmsize, 157
  - group2\_, 158
  - group\_, 158
  - inc\_depth, 158
  - inc\_states, 157
  - inc\_transitions, 158
  - max\_depth, 158
  - num, 158
  - o\_, 159
  - onepass\_, 158
  - options, 157
  - options\_updated, 157
  - parse\_options, 157
  - poprem\_, 158
  - pos, 158
  - print\_stats, 156
  - remove\_component, 157
  - removed\_components, 158
  - result, 156
  - safe, 157
  - set\_states, 157
  - states, 158
  - statistics, 157
  - stats, 159
  - stats\_map, 156
  - succ\_queue, 156
  - todo, 158
  - todo\_list, 156
  - transitions, 158
  - unsigned\_fun, 156
- spot::couvreur99\_check\_shy::successor, 159
  - acc, 160
  - s, 160
  - successor, 159
- spot::couvreur99\_check\_shy::todo\_item, 160
  - n, 160
  - q, 160
  - s, 160
  - todo\_item, 160
- spot::couvreur99\_check\_status, 161
  - ~couvreur99\_check\_status, 162
  - aut, 162
  - couvreur99\_check\_status, 162
  - cycle\_seed, 162
  - h, 162
  - print\_stats, 162
  - root, 162
  - states, 162
- spot::delayed\_simulation\_relation, 167
- spot::direct\_simulation\_relation, 167
- spot::dotty\_decorator, 167
  - ~dotty\_decorator, 168
  - dotty\_decorator, 168
  - instance, 169
  - link\_decl, 168
  - state\_decl, 168
- spot::duplicator\_node, 169
  - ~duplicator\_node, 172
  - acc\_, 173



- add\_pred, 173
- add\_succ, 172
- compare, 172
- del\_pred, 173
- del\_succ, 173
- duplicator\_node, 172
- get\_acc, 172
- get\_duplicator\_node, 173
- get\_label, 172
- get\_nb\_succ, 173
- get\_pair, 173
- get\_spoiler\_node, 173
- implies, 172
- label\_, 173
- lnode\_pred, 173
- lnode\_succ, 173
- match, 172
- not\_win, 173
- num\_, 173
- prune, 173
- sc\_, 173
- set\_win, 172
- succ\_to\_string, 173
- to\_string, 172
- spot::duplicator\_node\_delayed, 174
  - ~duplicator\_node\_delayed, 176
  - acc\_, 178
  - add\_pred, 177
  - add\_succ, 177
  - compare, 177
  - del\_pred, 177
  - del\_succ, 177
  - duplicator\_node\_delayed, 176
  - get\_acc, 177
  - get\_duplicator\_node, 178
  - get\_label, 177
  - get\_lead\_2\_acc\_all, 177
  - get\_nb\_succ, 177
  - get\_pair, 178
  - get\_progress\_measure, 177
  - get\_spoiler\_node, 178
  - implies, 177
  - implies\_acc, 177
  - implies\_label, 177
  - label\_, 178
  - lead\_2\_acc\_all\_, 178
  - lnode\_pred, 178
  - lnode\_succ, 178
  - match, 177
  - not\_win, 178
  - num\_, 178
  - progress\_measure\_, 178
  - prune, 177
  - sc\_, 178
  - seen\_, 178
  - set\_lead\_2\_acc\_all, 177
  - set\_win, 176
  - succ\_to\_string, 177
  - to\_string, 176
- spot::ec\_statistics, 178
  - dec\_depth, 181
  - depth, 181
  - depth\_, 182
  - ec\_statistics, 181
  - get, 181
  - inc\_depth, 181
  - inc\_states, 181
  - inc\_transitions, 181
  - max\_depth, 181
  - max\_depth\_, 182
  - set\_states, 181
  - states, 181
  - states\_, 182
  - stats, 182
  - stats\_map, 181
  - transitions, 181
  - transitions\_, 182
  - unsigned\_fun, 181
- spot::emptiness\_check, 182
  - ~emptiness\_check, 185
  - a\_, 186
  - automaton, 185
  - check, 185
  - emptiness\_check, 185
  - o\_, 186
  - options, 185
  - options\_updated, 186
  - parse\_options, 185
  - print\_stats, 186
  - safe, 185
  - statistics, 186
- spot::emptiness\_check\_instantiator, 186
  - construct, 188
  - emptiness\_check\_instantiator, 188
  - info\_, 188
  - instantiate, 188
  - max\_acceptance\_conditions, 188
  - min\_acceptance\_conditions, 188
  - o\_, 188
  - options, 188
- spot::emptiness\_check\_result, 189
  - ~emptiness\_check\_result, 191
  - a\_, 192
  - accepting\_run, 191
  - automaton, 191
  - emptiness\_check\_result, 191
  - o\_, 192
  - options, 191

- options\_updated, 192
- parse\_options, 191
- statistics, 192
- spot::evtgba, 193
  - ~evtgba, 194
  - all\_acceptance\_conditions, 195
  - alphabet, 195
  - evtgba, 194
  - format\_acceptance\_condition, 195
  - format\_acceptance\_conditions, 195
  - format\_label, 195
  - format\_state, 195
  - init\_iter, 195
  - pred\_iter, 195
  - succ\_iter, 195
- spot::evtgba\_explicit, 196
  - ~evtgba\_explicit, 198
  - acc\_set\_, 200
  - add\_transition, 199
  - all\_acceptance\_conditions, 198
  - alphabet, 198
  - alphabet\_, 200
  - declare\_acceptance\_condition, 199
  - declare\_state, 199
  - evtgba\_explicit, 198
  - format\_acceptance\_condition, 199
  - format\_acceptance\_conditions, 199
  - format\_label, 199
  - format\_state, 198, 199
  - init\_iter, 198
  - init\_states\_, 200
  - name\_state\_map\_, 199
  - ns\_map, 198
  - pred\_iter, 198, 199
  - set\_init\_state, 199
  - sn\_map, 198
  - state\_name\_map\_, 199
  - succ\_iter, 198, 199
  - transition\_list, 198
- spot::evtgba\_explicit::state, 200
  - in, 200
  - out, 200
- spot::evtgba\_explicit::transition, 200
  - acceptance\_conditions, 201
  - in, 201
  - label, 201
  - out, 201
- spot::evtgba\_iterator, 201
  - ~evtgba\_iterator, 202
  - current\_acceptance\_conditions, 202
  - current\_label, 202
  - current\_state, 202
  - done, 202
  - first, 202
  - next, 202
- spot::evtgba\_product, 202
  - ~evtgba\_product, 204
  - all\_acc\_, 205
  - all\_acceptance\_conditions, 205
  - alphabet, 205
  - alphabet\_, 205
  - common\_symbol\_table, 204
  - common\_symbols\_, 205
  - evtgba\_product, 204
  - evtgba\_product\_operands, 204
  - format\_acceptance\_condition, 205
  - format\_acceptance\_conditions, 205
  - format\_label, 205
  - format\_state, 204
  - init\_iter, 204
  - op\_, 205
  - pred\_iter, 204
  - succ\_iter, 204
- spot::evtgba\_reachable\_iterator, 205
  - ~evtgba\_reachable\_iterator, 207
  - add\_state, 208
  - automata\_, 209
  - end, 208
  - evtgba\_reachable\_iterator, 207
  - next\_state, 208
  - process\_link, 208
  - process\_state, 208
  - run, 208
  - seen, 209
  - seen\_map, 207
  - start, 208
- spot::evtgba\_reachable\_iterator\_breadth\_first, 209
  - add\_state, 211
  - automata\_, 212
  - end, 212
  - evtgba\_reachable\_iterator\_breadth\_first, 211
  - next\_state, 211
  - process\_link, 212
  - process\_state, 212
  - run, 211
  - seen, 212
  - seen\_map, 211
  - start, 211
  - todo, 212
- spot::evtgba\_reachable\_iterator\_depth\_first, 213
  - add\_state, 215
  - automata\_, 216
  - end, 215
  - evtgba\_reachable\_iterator\_depth\_first, 214
  - next\_state, 215
  - process\_link, 215
  - process\_state, 215
  - run, 215

- seen, 216
- seen\_map, 214
- start, 215
- todo, 216
- spot::explicit\_connected\_component, 216
  - ~explicit\_connected\_component, 217
  - condition, 218
  - has\_state, 217
  - index, 218
  - insert, 217
  - rem, 218
- spot::explicit\_connected\_component\_factory, 218
  - ~explicit\_connected\_component\_factory, 219
  - build, 219
- spot::free\_list, 223
  - ~free\_list, 225
  - dump\_free\_list, 226
  - extend, 226
  - fl, 226
  - free\_count, 226
  - free\_list\_type, 225
  - insert, 226
  - pos\_lenght\_pair, 225
  - register\_n, 226
  - release\_n, 226
  - remove, 226
- spot::gspn\_exception, 227
  - err\_, 227
  - get\_err, 227
  - get\_where, 227
  - gspn\_exception, 227
  - where\_, 227
- spot::gspn\_interface, 227
  - ~gspn\_interface, 229
  - automaton, 229
  - dead\_, 229
  - dict\_, 229
  - env\_, 229
  - gspn\_interface, 229
- spot::gspn\_ssp\_interface, 229
  - ~gspn\_ssp\_interface, 231
  - automaton, 231
  - dict\_, 231
  - env\_, 231
  - gspn\_ssp\_interface, 231
- spot::loopless\_modular\_mixed\_radix\_gray\_code, 240
  - ~loopless\_modular\_mixed\_radix\_gray\_code, 241
  - a\_, 242
  - a\_first, 242
  - a\_last, 242
  - a\_next, 242
  - done, 242
  - done\_, 242
  - f\_, 242
  - first, 242
  - last, 242
  - loopless\_modular\_mixed\_radix\_gray\_code, 241
  - m\_, 242
  - n\_, 242
  - next, 242
  - non\_one\_radixes\_, 243
  - s\_, 243
- spot::ltl, 82
  - reduce\_tau03, 85
  - unabbreviate\_ltl, 86
- spot::ltl::atomic\_prop, 91
  - ~atomic\_prop, 95
  - accept, 95
  - atomic\_prop, 95
  - dump, 96
  - dump\_, 97
  - dump\_instances, 96
  - env, 95
  - env\_, 97
  - hash, 96
  - hash\_key\_, 97
  - instance, 95
  - instance\_count, 95
  - instances, 97
  - map, 95
  - name, 95
  - name\_, 97
  - pair, 95
  - ref, 96
  - ref\_, 96
  - ref\_count\_, 96
  - set\_key\_, 96
  - unref, 96
  - unref\_, 96
- spot::ltl::binop, 120
  - ~binop, 125
  - accept, 125
  - binop, 125
  - dump, 126
  - dump\_, 127
  - Equiv, 125
  - first, 125
  - first\_, 127
  - hash, 126
  - hash\_key\_, 127
  - Implies, 124
  - instance, 125
  - instance\_count, 126
  - instances, 127
  - map, 124

- op, 125
- op\_, 127
- op\_name, 125
- pair, 124
- pairf, 124
- R, 125
- ref, 126
- ref\_, 126
- ref\_count\_, 126
- second, 125
- second\_, 127
- set\_key\_, 126
- type, 124
- U, 125
- unref, 126
- unref\_, 126
- Xor, 124
- spot::ltl::clone\_visitor, 128
  - ~clone\_visitor, 129
  - clone\_visitor, 129
  - recurse, 130
  - result, 130
  - result\_, 130
  - visit, 130
- spot::ltl::const\_visitor, 134
  - ~const\_visitor, 135
  - visit, 135
- spot::ltl::constant, 135
  - ~constant, 138
  - accept, 138
  - constant, 138
  - dump, 139
  - dump\_, 140
  - False, 138
  - false\_instance, 139
  - hash, 139
  - hash\_key\_, 140
  - ref, 139
  - ref\_, 139
  - set\_key\_, 139
  - True, 138
  - true\_instance, 139
  - type, 138
  - unref, 139
  - unref\_, 139
  - val, 138
  - val\_, 140
  - val\_name, 138
- spot::ltl::declarative\_environment, 163
  - ~declarative\_environment, 164
  - declarative\_environment, 164
  - declare, 164
  - get\_prop\_map, 164
  - name, 164
  - prop\_map, 164
  - props\_, 165
  - require, 164
- spot::ltl::default\_environment, 165
  - ~default\_environment, 166
  - default\_environment, 166
  - instance, 167
  - name, 166
  - require, 166
- spot::ltl::environment, 192
  - ~environment, 193
  - name, 193
  - require, 193
- spot::ltl::formula, 219
  - ~formula, 221
  - accept, 221
  - dump, 221
  - dump\_, 222
  - hash, 221
  - hash\_key\_, 222
  - ref, 221
  - ref\_, 221
  - set\_key\_, 222
  - unref, 221
  - unref\_, 221
- spot::ltl::formula\_ptr\_hash, 222
  - operator(), 222
- spot::ltl::formula\_ptr\_less\_than, 223
  - operator(), 223
- spot::ltl::language\_containment\_checker, 231
  - ~language\_containment\_checker, 234
  - branching\_postponement\_, 234
  - contained, 234
  - contained\_neg, 234
  - dict\_, 234
  - equal, 234
  - exprop\_, 234
  - fair\_loop\_approx\_, 234
  - incompatible\_, 234
  - language\_containment\_checker, 234
  - neg\_contained, 234
  - register\_formula\_, 234
  - symb\_merge\_, 234
  - trans\_map, 233
  - translated\_, 235
- spot::ltl::language\_containment\_checker::record\_, 235
  - incomp\_map, 235
  - incompatible, 236
  - translation, 236
- spot::ltl::multop, 246
  - ~multop, 251
  - accept, 251
  - And, 251

- children\_, 253
- dump, 253
- dump\_, 253
- hash, 253
- hash\_key\_, 253
- instance, 251
- instance\_count, 252
- instances, 253
- map, 250
- multop, 251
- nth, 252
- op, 252
- op\_, 253
- op\_name, 252
- Or, 251
- pair, 250
- ref, 252
- ref\_, 252
- ref\_count\_, 252
- set\_key\_, 253
- size, 251
- type, 251
- unref, 252
- unref\_, 252
- vec, 250
- spot::ltl::multop::pairemp, 253
  - operator(), 254
- spot::ltl::postfix\_visitor, 287
  - ~postfix\_visitor, 289
  - doit, 289, 290
  - doit\_default, 290
  - postfix\_visitor, 289
  - visit, 289
- spot::ltl::random\_ltl, 290
  - ~random\_ltl, 292
  - ap, 293
  - ap\_, 293
  - dump\_priorities, 292
  - generate, 292
  - parse\_options, 292
  - proba\_, 293
  - proba\_2\_, 293
  - random\_ltl, 292
  - total\_1\_, 293
  - total\_2\_, 293
  - total\_2\_and\_more\_, 293
  - update\_sums, 293
- spot::ltl::random\_ltl::op\_proba, 293
  - build, 294
  - builder, 294
  - min\_n, 294
  - name, 294
  - proba, 294
  - setup, 294
- spot::ltl::read\_only\_environment, 294
  - ~declarative\_environment, 296
  - declarative\_environment, 296
  - declare, 296
  - get\_prop\_map, 296
  - name, 296
  - prop\_map, 296
  - props\_, 297
  - require, 296
- spot::ltl::ref\_formula, 297
  - ~ref\_formula, 299
  - accept, 300
  - dump, 300
  - dump\_, 300
  - hash, 300
  - hash\_key\_, 301
  - ref, 300
  - ref\_, 299
  - ref\_count\_, 299
  - ref\_counter\_, 300
  - ref\_formula, 299
  - set\_key\_, 300
  - unref, 300
  - unref\_, 299
- spot::ltl::simplify\_f\_g\_visitor, 305
  - ~simplify\_f\_g\_visitor, 307
  - recurse, 307
  - result, 307
  - result\_, 308
  - simplify\_f\_g\_visitor, 307
  - super, 307
  - visit, 307, 308
- spot::ltl::unabbreviate\_logic\_visitor, 446
  - ~unabbreviate\_logic\_visitor, 449
  - recurse, 449
  - result, 449
  - result\_, 450
  - super, 449
  - unabbreviate\_logic\_visitor, 449
  - visit, 449
- spot::ltl::unabbreviate\_ltl\_visitor, 450
  - ~unabbreviate\_ltl\_visitor, 452
  - recurse, 452
  - result, 453
  - result\_, 453
  - super, 452
  - unabbreviate\_ltl\_visitor, 452
  - visit, 452
- spot::ltl::unop, 453
  - ~unop, 456
  - accept, 457
  - child, 457
  - child\_, 458
  - dump, 458

- dump\_, 458
- F, 456
- G, 456
- hash, 458
- hash\_key\_, 458
- instance, 457
- instance\_count, 457
- instances, 458
- map, 456
- Not, 456
- op, 457
- op\_, 458
- op\_name, 457
- pair, 456
- ref, 457
- ref\_, 457
- ref\_count\_, 457
- set\_key\_, 458
- type, 456
- unop, 456
- unref, 458
- unref\_, 457
- X, 456
- spot::ltl::visitor, 461
  - ~visitor, 462
  - visit, 463
- spot::minato\_isop, 243
  - cube\_, 244
  - minato\_isop, 244
  - next, 244
  - ret\_, 244
  - todo\_, 244
- spot::minato\_isop::local\_vars
  - FirstStep, 245
  - FourthStep, 245
  - SecondStep, 245
  - ThirdStep, 245
- spot::minato\_isop::local\_vars, 244
  - f0\_max, 246
  - f0\_min, 246
  - f1\_max, 246
  - f1\_min, 246
  - f\_max, 245
  - f\_min, 245
  - g0, 246
  - g1, 246
  - local\_vars, 245
  - step, 245
  - v1, 246
  - vars, 246
- spot::numbered\_state\_heap, 254
  - ~numbered\_state\_heap, 255
  - find, 255
  - index, 255, 256
  - insert, 256
  - iterator, 256
  - size, 256
  - state\_index, 255
  - state\_index\_p, 255
- spot::numbered\_state\_heap\_const\_iterator, 256
  - ~numbered\_state\_heap\_const\_iterator, 257
  - done, 257
  - first, 257
  - get\_index, 257
  - get\_state, 257
  - next, 257
- spot::numbered\_state\_heap\_factory, 257
  - ~numbered\_state\_heap\_factory, 258
  - build, 258
- spot::numbered\_state\_heap\_hash\_map, 258
  - ~numbered\_state\_heap\_hash\_map, 260
  - find, 260, 261
  - h, 261
  - hash\_type, 260
  - index, 261
  - insert, 261
  - iterator, 261
  - size, 261
  - state\_index, 260
  - state\_index\_p, 260
- spot::numbered\_state\_heap\_hash\_map\_factory, 262
  - ~numbered\_state\_heap\_hash\_map\_factory, 263
  - build, 263
  - instance, 263
  - numbered\_state\_heap\_hash\_map\_factory, 263
- spot::option\_map, 263
  - get, 264
  - operator<<, 265
  - operator[], 264, 265
  - options\_, 265
  - parse\_options, 264
  - set, 265
- spot::parity\_game\_graph, 265
  - ~parity\_game\_graph, 269
  - add\_state, 270
  - automata\_, 271
  - build\_graph, 270
  - duplicator\_vertice\_, 271
  - end, 269
  - get\_relation, 269
  - lift, 270
  - nb\_node\_parity\_game, 271
  - next\_state, 270
  - parity\_game\_graph, 269
  - print, 269
  - process\_link, 269, 270

- process\_state, 269
- run, 270
- seen, 271
- seen\_map, 269
- spoiler\_vertice\_, 271
- start, 269
- tgba\_state\_, 271
- todo, 271
- spot::parity\_game\_graph\_delayed, 271
  - ~parity\_game\_graph\_delayed, 276
  - add\_duplicator\_node\_delayed, 276
  - add\_spoiler\_node\_delayed, 276
  - add\_state, 277
  - automata\_, 278
  - bdd\_v, 275
  - build\_graph, 276
  - build\_recurse\_successor\_duplicator, 276
  - build\_recurse\_successor\_spoiler, 276
  - duplicator\_vertice\_, 278
  - end, 276
  - get\_relation, 276
  - lift, 276
  - nb\_node\_parity\_game, 278
  - nb\_set\_acc\_cond, 276
  - next\_state, 277
  - parity\_game\_graph\_delayed, 276
  - print, 276
  - process\_link, 277
  - process\_state, 277
  - run, 277
  - seen, 278
  - seen\_map, 275
  - spoiler\_vertice\_, 278
  - start, 276
  - sub\_set\_acc\_cond\_, 278
  - tgba\_state\_, 278
  - todo, 278
- spot::parity\_game\_graph\_direct, 278
  - ~parity\_game\_graph\_direct, 282
  - add\_state, 283
  - automata\_, 284
  - build\_graph, 282
  - build\_link, 282
  - duplicator\_vertice\_, 284
  - end, 283
  - get\_relation, 282
  - lift, 282
  - nb\_node\_parity\_game, 284
  - next\_state, 283
  - parity\_game\_graph\_direct, 282
  - print, 282
  - process\_link, 283
  - process\_state, 283
  - run, 283
  - seen, 284
  - seen\_map, 282
  - spoiler\_vertice\_, 284
  - start, 282
  - tgba\_state\_, 284
  - todo, 284
- spot::ptr\_hash, 290
  - operator(), 290
- spot::rsymbol, 301
  - ~rsymbol, 302
  - operator const symbol \*, 302
  - operator!=, 302
  - operator<, 302
  - operator=, 302
  - operator==, 302
  - rsymbol, 302
  - s\_, 302
- spot::scc\_stack, 302
  - clear\_rem, 304
  - empty, 304
  - pop, 303
  - push, 303
  - rem, 304
  - s, 304
  - size, 304
  - stack\_type, 303
  - top, 303
- spot::scc\_stack::connected\_component, 304
  - condition, 305
  - connected\_component, 305
  - index, 305
  - rem, 305
- spot::spoiler\_node, 310
  - ~spoiler\_node, 312
  - add\_pred, 312
  - add\_succ, 312
  - compare, 313
  - del\_pred, 312
  - del\_succ, 312
  - get\_duplicator\_node, 313
  - get\_nb\_succ, 312
  - get\_pair, 313
  - get\_spoiler\_node, 313
  - lnode\_pred, 313
  - lnode\_succ, 313
  - not\_win, 313
  - num\_, 313
  - prune, 312
  - sc\_, 313
  - set\_win, 312
  - spoiler\_node, 312
  - succ\_to\_string, 313
  - to\_string, 312
- spot::spoiler\_node\_delayed, 313

- ~spoiler\_node\_delayed, 315
- acceptance\_condition\_visited\_, 317
- add\_pred, 316
- add\_succ, 316
- compare, 316
- del\_pred, 316
- del\_succ, 316
- get\_acceptance\_condition\_visited, 316
- get\_duplicator\_node, 316
- get\_lead\_2\_acc\_all, 316
- get\_nb\_succ, 316
- get\_pair, 317
- get\_progress\_measure, 316
- get\_spoiler\_node, 316
- lead\_2\_acc\_all\_, 317
- lnode\_pred, 317
- lnode\_succ, 317
- not\_win, 317
- num\_, 317
- progress\_measure\_, 317
- prune, 316
- sc\_, 317
- seen\_, 317
- set\_lead\_2\_acc\_all, 316
- set\_win, 316
- spoiler\_node\_delayed, 315
- succ\_to\_string, 316
- to\_string, 316
- spot::state, 321
  - ~state, 322
  - clone, 323
  - compare, 322
  - hash, 322
- spot::state\_bdd, 323
  - as\_bdd, 325
  - clone, 325
  - compare, 325
  - hash, 325
  - state\_, 325
  - state\_bdd, 324
- spot::state\_evtgba\_explicit, 325
  - ~state\_evtgba\_explicit, 327
  - clone, 327
  - compare, 327
  - get\_state, 327
  - hash, 327
  - state\_, 328
  - state\_evtgba\_explicit, 327
- spot::state\_explicit, 328
  - ~state\_explicit, 329
  - clone, 330
  - compare, 329
  - get\_state, 330
  - hash, 329
  - state\_, 330
  - state\_explicit, 329
- spot::state\_product, 330
  - ~state\_product, 332
  - clone, 333
  - compare, 332
  - hash, 332
  - left, 332
  - left\_, 333
  - right, 332
  - right\_, 333
  - state\_product, 332
- spot::state\_ptr\_equal, 333
  - operator(), 333
- spot::state\_ptr\_hash, 334
  - operator(), 334
- spot::state\_ptr\_less\_than, 334
  - operator(), 335
- spot::string\_hash, 335
  - operator(), 335
- spot::symbol, 335
  - ~symbol, 336
  - dump\_instances, 336
  - instance, 336
  - instance\_count, 336
  - instances\_, 337
  - map, 336
  - name, 336
  - name\_, 337
  - ref, 337
  - ref\_count\_, 337
  - refs\_, 337
  - symbol, 336
  - unref, 337
- spot::tgba, 337
  - ~tgba, 340
  - all\_acceptance\_conditions, 342
  - compute\_support\_conditions, 343
  - compute\_support\_variables, 343
  - format\_state, 342
  - get\_dict, 342
  - get\_init\_state, 341
  - last\_support\_conditions\_input\_, 343
  - last\_support\_conditions\_output\_, 343
  - last\_support\_variables\_input\_, 343
  - last\_support\_variables\_output\_, 343
  - neg\_acceptance\_conditions, 343
  - num\_acc\_, 343
  - number\_of\_acceptance\_conditions, 343
  - project\_state, 342
  - succ\_iter, 341
  - support\_conditions, 341
  - support\_variables, 341
  - tgba, 340



- transition\_annotation, 342
- spot::tgba\_bdd\_concrete, 344
  - ~tgba\_bdd\_concrete, 347
  - all\_acceptance\_conditions, 349
  - compute\_support\_conditions, 349
  - compute\_support\_variables, 349
  - data\_, 351
  - format\_state, 348
  - get\_core\_data, 349
  - get\_dict, 348
  - get\_init\_bdd, 348
  - get\_init\_state, 347
  - init\_, 351
  - neg\_acceptance\_conditions, 349
  - number\_of\_acceptance\_conditions, 350
  - operator=, 349
  - project\_state, 350
  - set\_init\_state, 347
  - succ\_iter, 348
  - support\_conditions, 349
  - support\_variables, 350
  - tgba\_bdd\_concrete, 347
  - transition\_annotation, 350
- spot::tgba\_bdd\_concrete\_factory, 351
  - ~tgba\_bdd\_concrete\_factory, 353
  - acc\_, 355
  - acc\_map\_, 353
  - constrain\_relation, 354
  - create\_atomic\_prop, 354
  - create\_state, 354
  - data\_, 355
  - declare\_acceptance\_condition, 354
  - finish, 355
  - get\_core\_data, 354
  - get\_dict, 354
  - tgba\_bdd\_concrete\_factory, 353
- spot::tgba\_bdd\_core\_data, 355
  - acc\_set, 360
  - acceptance\_conditions, 359
  - all\_acceptance\_conditions, 359
  - declare\_acceptance\_condition, 359
  - declare\_atomic\_prop, 358
  - declare\_now\_next, 358
  - dict, 360
  - negacc\_set, 360
  - next\_set, 360
  - notacc\_set, 360
  - notnext\_set, 360
  - notnow\_set, 360
  - notvar\_set, 360
  - now\_set, 359
  - nownext\_set, 360
  - operator=, 358
  - relation, 359
  - tgba\_bdd\_core\_data, 358
  - var\_set, 360
  - varandnext\_set, 360
- spot::tgba\_bdd\_factory, 361
  - ~tgba\_bdd\_factory, 361
  - get\_core\_data, 361
- spot::tgba\_explicit, 362
  - ~tgba\_explicit, 367
  - add\_acceptance\_condition, 367
  - add\_acceptance\_conditions, 368
  - add\_condition, 367
  - add\_conditions, 367
  - add\_state, 368
  - all\_acceptance\_conditions, 369
  - all\_acceptance\_conditions\_, 371
  - all\_acceptance\_conditions\_computed\_, 371
  - complement\_all\_acceptance\_conditions, 368
  - compute\_support\_conditions, 369
  - compute\_support\_variables, 369
  - copy\_acceptance\_conditions\_of, 367
  - create\_transition, 367
  - declare\_acceptance\_condition, 367
  - dict\_, 371
  - format\_state, 369
  - get\_acceptance\_condition, 369
  - get\_dict, 368
  - get\_init\_state, 368
  - has\_acceptance\_condition, 367
  - init\_, 371
  - merge\_transitions, 368
  - name\_state\_map\_, 371
  - neg\_acceptance\_conditions, 369
  - neg\_acceptance\_conditions\_, 371
  - ns\_map, 367
  - number\_of\_acceptance\_conditions, 370
  - operator=, 369
  - project\_state, 370
  - set\_init\_state, 367
  - sn\_map, 367
  - state, 367
  - state\_name\_map\_, 371
  - succ\_iter, 368
  - support\_conditions, 369
  - support\_variables, 370
  - tgba\_explicit, 367
  - transition\_annotation, 370
- spot::tgba\_explicit::transition, 371
  - acceptance\_conditions, 372
  - condition, 372
  - dest, 372
- spot::tgba\_explicit\_succ\_iterator, 372
  - ~tgba\_explicit\_succ\_iterator, 374
  - all\_acceptance\_conditions\_, 375
  - current\_acceptance\_conditions, 375

- current\_condition, 375
- current\_state, 374
- done, 374
- first, 374
- i\_, 375
- next, 374
- s\_, 375
- tgba\_explicit\_succ\_iterator, 374
- spot::tgba\_product, 375
  - ~tgba\_product, 379
  - all\_acceptance\_conditions, 381
  - all\_acceptance\_conditions\_, 382
  - compute\_support\_conditions, 381
  - compute\_support\_variables, 381
  - dict\_, 382
  - format\_state, 380
  - get\_dict, 380
  - get\_init\_state, 379
  - left\_, 382
  - left\_acc\_complement\_, 382
  - neg\_acceptance\_conditions, 381
  - neg\_acceptance\_conditions\_, 382
  - number\_of\_acceptance\_conditions, 382
  - operator=, 381
  - project\_state, 380
  - right\_, 382
  - right\_acc\_complement\_, 382
  - right\_common\_acc\_, 382
  - succ\_iter, 379
  - support\_conditions, 381
  - support\_variables, 382
  - tgba\_product, 379
  - transition\_annotation, 380
- spot::tgba\_reachable\_iterator, 383
  - ~tgba\_reachable\_iterator, 385
  - add\_state, 385
  - automata\_, 387
  - end, 386
  - next\_state, 386
  - process\_link, 386
  - process\_state, 386
  - run, 385
  - seen, 387
  - seen\_map, 385
  - start, 386
  - tgba\_reachable\_iterator, 385
- spot::tgba\_reachable\_iterator\_breadth\_first, 387
  - add\_state, 390
  - automata\_, 392
  - end, 391
  - next\_state, 390
  - process\_link, 391
  - process\_state, 391
  - run, 391
  - seen, 392
  - seen\_map, 390
  - start, 391
  - tgba\_reachable\_iterator\_breadth\_first, 390
  - todo, 392
- spot::tgba\_reachable\_iterator\_depth\_first, 392
  - add\_state, 394
  - automata\_, 396
  - end, 395
  - next\_state, 394
  - process\_link, 395
  - process\_state, 395
  - run, 395
  - seen, 396
  - seen\_map, 394
  - start, 395
  - tgba\_reachable\_iterator\_depth\_first, 394
  - todo, 396
- spot::tgba\_reduc, 396
  - ~tgba\_reduc, 402
  - acc\_, 410
  - add\_acceptance\_condition, 406
  - add\_acceptance\_conditions, 406
  - add\_condition, 405
  - add\_conditions, 405
  - add\_state, 406, 409
  - all\_acceptance\_conditions, 407
  - all\_acceptance\_conditions\_, 410
  - all\_acceptance\_conditions\_computed\_, 410
  - automata\_, 410
  - complement\_all\_acceptance\_conditions, 406
  - compute\_scc, 403
  - compute\_support\_conditions, 407
  - compute\_support\_variables, 407
  - copy\_acceptance\_conditions\_of, 406
  - create\_transition, 404, 405
  - declare\_acceptance\_condition, 405
  - delete\_scc, 404
  - delete\_transitions, 403
  - dict\_, 410
  - display\_rel\_sim, 403
  - display\_scc, 403
  - end, 403
  - format\_state, 403
  - get\_acceptance\_condition, 408
  - get\_dict, 407
  - get\_init\_state, 406
  - h\_, 409
  - has\_acceptance\_condition, 406
  - init\_, 410
  - is\_not\_accepting, 405
  - is\_terminal, 404
  - merge\_state, 404
  - merge\_state\_delayed, 404

- merge\_transitions, 406
- name\_state\_map\_, 410
- nb\_set\_acc\_cond, 405
- neg\_acceptance\_conditions, 407
- neg\_acceptance\_conditions\_, 410
- next\_state, 409
- ns\_map, 402
- number\_of\_acceptance\_conditions, 408
- process\_link, 404, 409
- process\_state, 403
- project\_state, 408
- prune\_acc, 403
- prune\_scc, 403
- quotient\_state, 403
- redirect\_transition, 404
- remove\_acc, 405
- remove\_component, 405
- remove\_predecessor\_state, 404
- remove\_scc, 405
- remove\_state, 404
- root\_, 409
- run, 409
- scc\_computed\_, 409
- seen, 410
- seen\_, 410
- seen\_map, 402
- set\_init\_state, 405
- si\_, 410
- sn\_map, 402
- sp\_map, 402
- start, 403
- state, 402
- state\_name\_map\_, 410
- state\_predecessor\_map\_, 410
- state\_scc\_, 409
- state\_scc\_v\_, 409
- succ\_iter, 406
- support\_conditions, 408
- support\_variables, 408
- tgba\_reduc, 402
- todo, 410
- transition\_annotation, 408
- spot::tgba\_run, 411
  - ~tgba\_run, 411
  - cycle, 412
  - operator=, 411
  - prefix, 412
  - steps, 411
  - tgba\_run, 411
- spot::tgba\_run::step, 412
  - acc, 412
  - label, 412
  - s, 412
- spot::tgba\_run\_dotty\_decorator, 413
  - ~tgba\_run\_dotty\_decorator, 414
- instance, 415
- link\_decl, 415
- map\_, 415
- run\_, 415
- state\_decl, 414
- step\_map, 414
- step\_num, 414
- step\_set, 414
- tgba\_run\_dotty\_decorator, 414
- spot::tgba\_sba\_proxy, 416
  - acc\_cycle\_, 422
  - all\_acceptance\_conditions, 421
  - compute\_support\_conditions, 421
  - compute\_support\_variables, 421
  - cycle\_list, 419
  - format\_state, 420
  - get\_dict, 420
  - get\_init\_state, 419
  - neg\_acceptance\_conditions, 421
  - number\_of\_acceptance\_conditions, 422
  - project\_state, 420
  - state\_is\_accepting, 419
  - succ\_iter, 419
  - support\_conditions, 421
  - support\_variables, 421
  - tgba\_sba\_proxy, 419
  - transition\_annotation, 420
- spot::tgba\_statistics, 422
  - states, 422
  - transitions, 422
- spot::tgba\_succ\_iterator, 422
  - ~tgba\_succ\_iterator, 424
  - current\_acceptance\_conditions, 425
  - current\_condition, 425
  - current\_state, 424
  - done, 424
  - first, 424
  - next, 424
- spot::tgba\_succ\_iterator\_concrete, 425
  - ~tgba\_succ\_iterator\_concrete, 429
  - current\_, 430
  - current\_acc\_, 430
  - current\_acceptance\_conditions, 430
  - current\_condition, 430
  - current\_state, 429
  - current\_state\_, 430
  - data\_, 430
  - done, 429
  - first, 429
  - next, 429
  - succ\_set\_, 430
  - succ\_set\_left\_, 430
  - tgba\_succ\_iterator\_concrete, 429

- spot::tgba\_succ\_iterator\_product, 431
  - ~tgba\_succ\_iterator\_product, 433
  - current\_acceptance\_conditions, 434
  - current\_cond\_, 435
  - current\_condition, 434
  - current\_state, 434
  - done, 434
  - first, 433
  - left\_, 435
  - left\_neg\_, 435
  - next, 433
  - next\_non\_false\_, 434
  - right\_, 435
  - right\_common\_acc\_, 435
  - right\_neg\_, 435
  - step\_, 434
  - tgba\_product, 435
  - tgba\_succ\_iterator\_product, 433
- spot::tgba\_tba\_proxy, 435
  - ~tgba\_tba\_proxy, 439
  - a\_, 442
  - acc\_cycle\_, 442
  - all\_acceptance\_conditions, 440
  - compute\_support\_conditions, 441
  - compute\_support\_variables, 441
  - cycle\_list, 439
  - format\_state, 440
  - get\_dict, 440
  - get\_init\_state, 439
  - neg\_acceptance\_conditions, 441
  - number\_of\_acceptance\_conditions, 442
  - operator=, 441
  - project\_state, 440
  - succ\_iter, 439
  - support\_conditions, 441
  - support\_variables, 441
  - tgba\_tba\_proxy, 439
  - the\_acceptance\_cond\_, 442
  - transition\_annotation, 440
- spot::time\_info, 442
  - stime, 442
  - time\_info, 442
  - utime, 442
- spot::timer, 443
  - start, 444
  - start\_, 444
  - stime, 444
  - stop, 444
  - total\_, 444
  - utime, 444
- spot::timer\_map, 444
  - cancel, 445
  - empty, 446
  - item\_type, 445
  - print, 446
  - start, 445
  - stop, 445
  - timer, 446
  - tm, 446
  - tm\_type, 445
- spot::unsigned\_statistics, 459
  - ~unsigned\_statistics, 460
  - get, 460
  - stats, 460
  - stats\_map, 460
  - unsigned\_fun, 460
- spot::unsigned\_statistics\_copy, 460
  - operator!=, 461
  - operator==, 461
  - set, 461
  - seteq, 461
  - stats, 461
  - stats\_map, 461
  - unsigned\_statistics\_copy, 461
- spot::weight, 463
  - dec\_weight\_handler, 465
  - inc\_weight\_handler, 464
  - m, 465
  - neg\_all\_acc, 465
  - operator+=, 464
  - operator-, 464
  - operator=, 464
  - operator<<, 465
  - pm, 465
  - weight, 464
  - weight\_vector, 464
- srand
  - random, 33
- stack
  - ltlyy::stack, 321
  - sautyy::stack, 319
- stack\_
  - ltlyy::slice, 309
  - sautyy::slice, 310
- stack\_type
  - spot::scc\_stack, 303
- start
  - spot::evtgba\_reachable\_iterator, 208
  - spot::evtgba\_reachable\_iterator\_breadth\_first, 211
  - spot::evtgba\_reachable\_iterator\_depth\_first, 215
  - spot::parity\_game\_graph, 269
  - spot::parity\_game\_graph\_delayed, 276
  - spot::parity\_game\_graph\_direct, 282
  - spot::tgba\_reachable\_iterator, 386
  - spot::tgba\_reachable\_iterator\_breadth\_first, 391

- spot::tgba\_reachable\_iterator\_depth\_first, 395
- spot::tgba\_reduc, 403
- spot::timer, 444
- spot::timer\_map, 445
- start\_
  - spot::timer, 444
- state
  - spot::tgba\_explicit, 367
  - spot::tgba\_reduc, 402
- state\_
  - spot::state\_bdd, 325
  - spot::state\_evtgba\_explicit, 328
  - spot::state\_explicit, 330
- state\_bdd
  - spot::state\_bdd, 324
- state\_couple
  - spot, 78
- state\_decl
  - spot::dotty\_decorator, 168
  - spot::tgba\_run\_dotty\_decorator, 414
- state\_evtgba\_explicit
  - spot::state\_evtgba\_explicit, 327
- state\_explicit
  - spot::state\_explicit, 329
- state\_index
  - spot::numbered\_state\_heap, 255
  - spot::numbered\_state\_heap\_hash\_map, 260
- state\_index\_p
  - spot::numbered\_state\_heap, 255
  - spot::numbered\_state\_heap\_hash\_map, 260
- state\_is\_accepting
  - spot::tgba\_sba\_proxy, 419
- state\_name\_map\_
  - spot::evtgba\_explicit, 199
  - spot::tgba\_explicit, 371
  - spot::tgba\_reduc, 410
- state\_predecessor\_map\_
  - spot::tgba\_reduc, 410
- state\_product
  - spot::state\_product, 332
- state\_scc\_
  - spot::tgba\_reduc, 409
- state\_scc\_v\_
  - spot::tgba\_reduc, 409
- states
  - spot::connected\_component\_hash\_set, 132
  - spot::couvreur99\_check, 146
  - spot::couvreur99\_check\_shy, 158
  - spot::couvreur99\_check\_status, 162
  - spot::ec\_statistics, 181
  - spot::tgba\_statistics, 422
- states\_
  - spot::ec\_statistics, 182
- statistics
  - spot::couvreur99\_check, 145
  - spot::couvreur99\_check\_result, 150
  - spot::couvreur99\_check\_shy, 157
  - spot::emptiness\_check, 186
  - spot::emptiness\_check\_result, 192
- stats
  - spot::acss\_statistics, 89
  - spot::ars\_statistics, 91
  - spot::couvreur99\_check, 146
  - spot::couvreur99\_check\_result, 151
  - spot::couvreur99\_check\_shy, 159
  - spot::ec\_statistics, 182
  - spot::unsigned\_statistics, 460
  - spot::unsigned\_statistics\_copy, 461
- stats\_map
  - spot::acss\_statistics, 88
  - spot::ars\_statistics, 90
  - spot::couvreur99\_check, 144
  - spot::couvreur99\_check\_result, 149
  - spot::couvreur99\_check\_shy, 156
  - spot::ec\_statistics, 181
  - spot::unsigned\_statistics, 460
  - spot::unsigned\_statistics\_copy, 461
- stats\_reachable
  - tgba\_misc, 43
- step
  - ltlyy::location, 239
  - sautyy::location, 237
  - spot::minato\_isop::local\_vars, 245
- step\_
  - spot::tgba\_succ\_iterator\_product, 434
- step\_map
  - spot::tgba\_run\_dotty\_decorator, 414
- step\_num
  - spot::tgba\_run\_dotty\_decorator, 414
- step\_set
  - spot::tgba\_run\_dotty\_decorator, 414
- steps
  - spot::tgba\_run, 411
- stime
  - spot::time\_info, 442
  - spot::timer, 444
- stop
  - spot::timer, 444
  - spot::timer\_map, 445
- sub\_set\_acc\_cond\_
  - spot::parity\_game\_graph\_delayed, 278
- succ\_iter
  - spot::evtgba, 195
  - spot::evtgba\_explicit, 198, 199
  - spot::evtgba\_product, 204
  - spot::tgba, 341
  - spot::tgba\_bdd\_concrete, 348
  - spot::tgba\_explicit, 368

- spot::tgba\_product, 379
- spot::tgba\_reduc, 406
- spot::tgba\_sba\_proxy, 419
- spot::tgba\_tba\_proxy, 439
- succ\_queue
  - spot::couvreur99\_check\_shy, 156
- succ\_set\_
  - spot::tgba\_succ\_iterator\_concrete, 430
- succ\_set\_left\_
  - spot::tgba\_succ\_iterator\_concrete, 430
- succ\_to\_string
  - spot::duplicator\_node, 173
  - spot::duplicator\_node\_delayed, 177
  - spot::spoiler\_node, 313
  - spot::spoiler\_node\_delayed, 316
- successor
  - spot::couvreur99\_check\_shy::successor, 159
- super
  - spot::ltl::simplify\_f\_g\_visitor, 307
  - spot::ltl::unabbreviate\_logic\_visitor, 449
  - spot::ltl::unabbreviate\_ltl\_visitor, 452
- support\_conditions
  - spot::tgba, 341
  - spot::tgba\_bdd\_concrete, 349
  - spot::tgba\_explicit, 369
  - spot::tgba\_product, 381
  - spot::tgba\_reduc, 408
  - spot::tgba\_sba\_proxy, 421
  - spot::tgba\_tba\_proxy, 441
- support\_variables
  - spot::tgba, 341
  - spot::tgba\_bdd\_concrete, 350
  - spot::tgba\_explicit, 370
  - spot::tgba\_product, 382
  - spot::tgba\_reduc, 408
  - spot::tgba\_sba\_proxy, 421
  - spot::tgba\_tba\_proxy, 441
- symb\_merge\_
  - spot::ltl::language\_containment\_checker, 234
- symbol
  - spot::symbol, 336
- symbol\_set
  - spot, 78
- syntactic\_implication
  - ltl\_misc, 28
- syntactic\_implication\_neg
  - ltl\_misc, 28
- tgba
  - spot::tgba, 340
- TGBA (Transition-based Generalized Büchi Automata), 18
- TGBA algorithms, 34
- TGBA on-the-fly algorithms, 34
- TGBA representations, 34
- TGBA runs and supporting functions, 54
- TGBA simplifications, 40
- tgba/ Directory Reference, 62
- tgba/bdddict.hh, 519
- tgba/bddprint.hh, 520
- tgba/formula2bdd.hh, 522
- tgba/public.hh, 480
- tgba/state.hh, 522
- tgba/statebdd.hh, 524
- tgba/succiter.hh, 524
- tgba/succiterconcrete.hh, 525
- tgba/tgba.hh, 526
- tgba/tgabddconcrete.hh, 528
- tgba/tgabddconcretefactory.hh, 529
- tgba/tgabddconcreteproduct.hh, 529
- tgba/tgabddcoredata.hh, 530
- tgba/tgabddfactory.hh, 531
- tgba/tgbaexplicit.hh, 532
- tgba/tgabproduct.hh, 534
- tgba/tgbareduc.hh, 534
- tgba/tgbatba.hh, 536
- tgba\_algorithms
  - product, 34
- tgba\_bdd\_concrete
  - spot::tgba\_bdd\_concrete, 347
- tgba\_bdd\_concrete\_factory
  - spot::tgba\_bdd\_concrete\_factory, 353
- tgba\_bdd\_core\_data
  - spot::tgba\_bdd\_core\_data, 358
- tgba\_dupexp\_bfs
  - tgba\_misc, 44
- tgba\_dupexp\_dfs
  - tgba\_misc, 44
- tgba\_explicit
  - spot::tgba\_explicit, 367
- tgba\_explicit\_succ\_iterator
  - spot::tgba\_explicit\_succ\_iterator, 374
- tgba\_io
  - dotty\_reachable, 36
  - format\_tgba\_parse\_errors, 36
  - lbt\_reachable, 36
  - never\_claim\_reachable, 36
  - tgba\_parse, 36
  - tgba\_parse\_error, 36
  - tgba\_parse\_error\_list, 36
  - tgba\_save\_reachable, 37
- tgba\_ltl
  - ltl\_to\_tgba\_fm, 37
  - ltl\_to\_tgba\_lacim, 39
- tgba\_misc
  - random\_graph, 43
  - stats\_reachable, 43
  - tgba\_dupexp\_bfs, 44

- tgba\_dupexp\_dfs, 44
  - tgba\_powerset, 44
- tgba\_parse
  - tgba\_io, 36
- tgba\_parse\_error
  - tgba\_io, 36
- tgba\_parse\_error\_list
  - tgba\_io, 36
- tgba\_powerset
  - tgba\_misc, 44
- tgba\_product
  - spot::tgba\_product, 379
  - spot::tgba\_succ\_iterator\_product, 435
- tgba\_reachable\_iterator
  - spot::tgba\_reachable\_iterator, 385
- tgba\_reachable\_iterator\_breadth\_first
  - spot::tgba\_reachable\_iterator\_breadth\_first, 390
- tgba\_reachable\_iterator\_depth\_first
  - spot::tgba\_reachable\_iterator\_depth\_first, 394
- tgba\_reduc
  - spot::tgba\_reduc, 402
- tgba\_reduction
  - Reduce\_All, 41
  - Reduce\_None, 41
  - Reduce\_quotient\_Del\_Sim, 41
  - Reduce\_quotient\_Dir\_Sim, 41
  - Reduce\_Scc, 41
  - Reduce\_transition\_Del\_Sim, 41
  - Reduce\_transition\_Dir\_Sim, 41
- tgba\_reduction
  - dn\_v, 41
  - free\_relation\_simulation, 41
  - get\_delayed\_relation\_simulation, 42
  - get\_direct\_relation\_simulation, 42
  - reduc\_tgba\_sim, 42
  - reduce\_tgba\_options, 41
  - s\_v, 41
  - sn\_v, 41
- tgba\_run
  - print\_tgba\_run, 54
  - project\_tgba\_run, 54
  - reduce\_run, 55
  - replay\_tgba\_run, 55
  - spot::tgba\_run, 411
  - tgba\_run\_to\_tgba, 55
- tgba\_run\_dotty\_decorator
  - spot::tgba\_run\_dotty\_decorator, 414
- tgba\_run\_to\_tgba
  - tgba\_run, 55
- tgba\_save\_reachable
  - tgba\_io, 37
- tgba\_sba\_proxy
  - spot::tgba\_sba\_proxy, 419
- tgba\_state\_
  - spot::parity\_game\_graph, 271
  - spot::parity\_game\_graph\_delayed, 278
  - spot::parity\_game\_graph\_direct, 284
- tgba\_succ\_iterator\_concrete
  - spot::tgba\_succ\_iterator\_concrete, 429
- tgba\_succ\_iterator\_product
  - spot::tgba\_succ\_iterator\_product, 433
- tgba\_tba\_proxy
  - spot::tgba\_tba\_proxy, 439
- tgba\_to\_evtgba
  - spot, 82
- tgbaalgorithms/ Directory Reference, 63
- tgbaalgorithms/bfssteps.hh, 536
- tgbaalgorithms/dotty.hh, 473
- tgbaalgorithms/dottydec.hh, 537
- tgbaalgorithms/dupexp.hh, 538
- tgbaalgorithms/emptiness.hh, 538
- tgbaalgorithms/emptiness\_stats.hh, 540
- tgbaalgorithms/gtec/ Directory Reference, 58
- tgbaalgorithms/gtec/ce.hh, 541
- tgbaalgorithms/gtec/explscch.hh, 542
- tgbaalgorithms/gtec/gtec.hh, 543
- tgbaalgorithms/gtec/nsheap.hh, 544
- tgbaalgorithms/gtec/sccstack.hh, 545
- tgbaalgorithms/gtec/status.hh, 546
- tgbaalgorithms/gv04.hh, 547
- tgbaalgorithms/lbtt.hh, 547
- tgbaalgorithms/ltl2tgba\_fm.hh, 548
- tgbaalgorithms/ltl2tgba\_lacim.hh, 549
- tgbaalgorithms/magic.hh, 549
- tgbaalgorithms/neverclaim.hh, 550
- tgbaalgorithms/powerset.hh, 551
- tgbaalgorithms/projrun.hh, 552
- tgbaalgorithms/randomgraph.hh, 553
- tgbaalgorithms/reachiter.hh, 475
- tgbaalgorithms/reducerun.hh, 553
- tgbaalgorithms/reducttgba\_sim.hh, 553
- tgbaalgorithms/replayrun.hh, 555
- tgbaalgorithms/rundotdec.hh, 556
- tgbaalgorithms/save.hh, 476
- tgbaalgorithms/se05.hh, 556
- tgbaalgorithms/stats.hh, 557
- tgbaalgorithms/tau03.hh, 558
- tgbaalgorithms/tau03opt.hh, 558
- tgbaalgorithms/weight.hh, 559
- tgbaalgorithms/ Directory Reference, 64
- tgbaalgorithms/public.hh, 480
- the\_acceptance\_cond\_
  - spot::tgba\_tba\_proxy, 442
- ThirdStep
  - spot::minato\_isop::local\_vars, 245
- time\_info
  - spot::time\_info, 442

- timer
  - spot::timer\_map, 446
- tm
  - spot::timer\_map, 446
- tm\_type
  - spot::timer\_map, 445
- to\_spin\_string
  - ltl\_io, 23
- to\_string
  - ltl\_io, 23
  - spot::duplicator\_node, 172
  - spot::duplicator\_node\_delayed, 176
  - spot::spoiler\_node, 312
  - spot::spoiler\_node\_delayed, 316
- todo
  - spot::couvreur99\_check\_shy, 158
  - spot::evtgba\_reachable\_iterator\_breadth\_first, 212
  - spot::evtgba\_reachable\_iterator\_depth\_first, 216
  - spot::parity\_game\_graph, 271
  - spot::parity\_game\_graph\_delayed, 278
  - spot::parity\_game\_graph\_direct, 284
  - spot::tgba\_reachable\_iterator\_breadth\_first, 392
  - spot::tgba\_reachable\_iterator\_depth\_first, 396
  - spot::tgba\_reduc, 410
- todo\_
  - spot::minato\_isop, 244
- todo\_item
  - spot::couvreur99\_check\_shy::todo\_item, 160
- todo\_list
  - spot::couvreur99\_check\_shy, 156
- top
  - spot::scc\_stack, 303
- total\_
  - spot::timer, 444
- total\_1\_
  - spot::ltl::random\_ltl, 293
- total\_2\_
  - spot::ltl::random\_ltl, 293
- total\_2\_and\_more\_
  - spot::ltl::random\_ltl, 293
- trans\_map
  - spot::ltl::language\_containment\_checker, 233
- transition\_annotation
  - spot::tgba, 342
  - spot::tgba\_bdd\_concrete, 350
  - spot::tgba\_explicit, 370
  - spot::tgba\_product, 380
  - spot::tgba\_reduc, 408
  - spot::tgba\_sba\_proxy, 420
  - spot::tgba\_tba\_proxy, 440
- transition\_list
  - spot::evtgba\_explicit, 198
- transitions
  - spot::couvreur99\_check, 146
  - spot::couvreur99\_check\_shy, 158
  - spot::ec\_statistics, 181
  - spot::tgba\_statistics, 422
- transitions\_
  - spot::ec\_statistics, 182
- translated\_
  - spot::ltl::language\_containment\_checker, 235
- Translating LTL formulae into TGBA, 37
- translation
  - spot::ltl::language\_containment\_checker::record\_, 236
- True
  - spot::ltl::constant, 138
- true\_instance
  - spot::ltl::constant, 139
- type
  - spot::ltl::binop, 124
  - spot::ltl::constant, 138
  - spot::ltl::multop, 251
  - spot::ltl::unop, 456
- U
  - spot::ltl::binop, 125
- unabbreviate\_logic
  - ltl\_rewriting, 26
- unabbreviate\_logic\_visitor
  - spot::ltl::unabbreviate\_logic\_visitor, 449
- unabbreviate\_ltl
  - spot::ltl, 86
- unabbreviate\_ltl\_visitor
  - spot::ltl::unabbreviate\_ltl\_visitor, 452
- unop
  - spot::ltl::unop, 456
- unref
  - spot::ltl::atomic\_prop, 96
  - spot::ltl::binop, 126
  - spot::ltl::constant, 139
  - spot::ltl::formula, 221
  - spot::ltl::multop, 252
  - spot::ltl::ref\_formula, 300
  - spot::ltl::unop, 458
  - spot::symbol, 337
- unref\_
  - spot::ltl::atomic\_prop, 96
  - spot::ltl::binop, 126
  - spot::ltl::constant, 139
  - spot::ltl::formula, 221
  - spot::ltl::multop, 252
  - spot::ltl::ref\_formula, 299
  - spot::ltl::unop, 457
- unregister\_all\_my\_variables



- spot::bdd\_dict, [110](#)
- unregister\_variable
  - spot::bdd\_dict, [110](#), [111](#)
- unsigned\_fun
  - spot::acss\_statistics, [88](#)
  - spot::ars\_statistics, [90](#)
  - spot::couvreur99\_check, [144](#)
  - spot::couvreur99\_check\_result, [149](#)
  - spot::couvreur99\_check\_shy, [156](#)
  - spot::ec\_statistics, [181](#)
  - spot::unsigned\_statistics, [460](#)
- unsigned\_statistics\_copy
  - spot::unsigned\_statistics\_copy, [461](#)
- update\_sums
  - spot::ltl::random\_ltl, [293](#)
- utime
  - spot::time\_info, [442](#)
  - spot::timer, [444](#)
- vl
  - spot::minato\_isop::local\_vars, [246](#)
- val
  - spot::ltl::constant, [138](#)
- val\_
  - spot::ltl::constant, [140](#)
- val\_name
  - spot::ltl::constant, [138](#)
- var\_formula\_map
  - spot::bdd\_dict, [111](#)
- var\_map
  - spot::bdd\_dict, [111](#)
- var\_refs
  - spot::bdd\_dict, [112](#)
- var\_set
  - spot::tgba\_bdd\_core\_data, [360](#)
- varandnext\_set
  - spot::tgba\_bdd\_core\_data, [360](#)
- vars
  - spot::minato\_isop::local\_vars, [246](#)
- vec
  - spot::ltl::multop, [250](#)
- version
  - misc\_tools, [30](#)
- vf\_map
  - spot::bdd\_dict, [108](#)
- visit
  - spot::ltl::clone\_visitor, [130](#)
  - spot::ltl::const\_visitor, [135](#)
  - spot::ltl::postfix\_visitor, [289](#)
  - spot::ltl::simplify\_f\_g\_visitor, [307](#), [308](#)
  - spot::ltl::unabbreviate\_logic\_visitor, [449](#)
  - spot::ltl::unabbreviate\_ltl\_visitor, [452](#)
  - spot::ltl::visitor, [463](#)
- vr\_map
  - spot::bdd\_dict, [108](#)
- wang32\_hash
  - hash\_funcs, [31](#)
- weight
  - spot::weight, [464](#)
- weight\_vector
  - spot::weight, [464](#)
- where\_
  - spot::gspn\_exception, [227](#)
- X
  - spot::ltl::unop, [456](#)
- Xor
  - spot::ltl::binop, [124](#)