

Parsybone

Generated by Doxygen 1.8.1.2

Tue Jul 17 2012 23:01:19

Contents

1	Class Index	1
1.1	Class Hierarchy	1
2	Class Index	3
2.1	Class List	3
3	File Index	5
3.1	File List	5
4	Class Documentation	7
4.1	ArgumentParser Class Reference	7
4.1.1	Detailed Description	7
4.1.2	Member Function Documentation	7
4.1.2.1	parseArguments	7
4.2	rapidxml::attribute_iterator< Ch > Class Template Reference	7
4.2.1	Detailed Description	8
4.3	AutomatonBuilder Class Reference	8
4.3.1	Constructor & Destructor Documentation	8
4.3.1.1	AutomatonBuilder	8
4.3.2	Member Function Documentation	9
4.3.2.1	buildAutomaton	9
4.4	AutomatonInterface Class Reference	9
4.4.1	Constructor & Destructor Documentation	9
4.4.1.1	~AutomatonInterface	9
4.4.2	Member Function Documentation	9
4.4.2.1	getFinalStates	9
4.4.2.2	getInitialStates	9
4.4.2.3	isFinal	10
4.4.2.4	isInitial	10
4.5	AutomatonStructure Class Reference	10
4.5.1	Member Function Documentation	11
4.5.1.1	getAllowedValues	11

4.5.1.2	getFinalStates	11
4.5.1.3	getInitialStates	11
4.5.1.4	getStateCount	11
4.5.1.5	getString	11
4.5.1.6	getTargetID	11
4.5.1.7	getTransitionCount	11
4.5.1.8	isFinal	12
4.5.1.9	isInitial	12
4.5.1.10	isTransitionFeasible	12
4.6	BasicStructure Class Reference	12
4.6.1	Member Function Documentation	12
4.6.1.1	getDirection	12
4.6.1.2	getSpecieID	13
4.6.1.3	getStateCount	13
4.6.1.4	getStateLevels	13
4.6.1.5	getString	13
4.6.1.6	getTargetID	13
4.6.1.7	getTransitionCount	13
4.7	BasicStructureBuilder Class Reference	14
4.7.1	Constructor & Destructor Documentation	14
4.7.1.1	BasicStructureBuilder	14
4.7.2	Member Function Documentation	14
4.7.2.1	buildStructure	14
4.8	ClassName Class Reference	14
4.9	ColoringAnalyzer Class Reference	14
4.9.1	Constructor & Destructor Documentation	14
4.9.1.1	ColoringAnalyzer	14
4.9.2	Member Function Documentation	15
4.9.2.1	getColors	15
4.9.2.2	getColors	15
4.9.2.3	getUnion	15
4.9.2.4	storeResults	15
4.9.2.5	startNewRound	15
4.10	ColoringParser Class Reference	15
4.10.1	Constructor & Destructor Documentation	16
4.10.1.1	ColoringParser	16
4.10.2	Member Function Documentation	16
4.10.2.1	createOutput	16
4.10.2.2	getColors	16
4.10.2.3	getColorsCount	16

4.10.2.4	input	16
4.10.2.5	openFile	16
4.10.2.6	output	17
4.10.2.7	outputComputed	17
4.10.2.8	parseMask	17
4.11	ColorStorage Class Reference	17
4.11.1	Member Function Documentation	17
4.11.1.1	addFrom	17
4.11.1.2	getColor	18
4.11.1.3	getColor	18
4.11.1.4	getMarking	18
4.11.1.5	getNeighbours	18
4.11.1.6	reset	19
4.11.1.7	update	19
4.11.1.8	update	19
4.12	ConstrainsParser Class Reference	19
4.12.1	Member Function Documentation	20
4.12.1.1	getAllColorsNum	20
4.12.1.2	getColor	20
4.12.1.3	getColorsNum	20
4.12.1.4	getSpecieNum	20
4.12.1.5	getTargetVals	20
4.12.1.6	parseConstrains	20
4.13	rapidxml::file< Ch > Class Template Reference	20
4.13.1	Detailed Description	21
4.13.2	Constructor & Destructor Documentation	21
4.13.2.1	file	21
4.13.2.2	file	21
4.13.3	Member Function Documentation	21
4.13.3.1	data	21
4.13.3.2	data	21
4.13.3.3	size	21
4.14	FunctionsBuilder Class Reference	22
4.14.1	Constructor & Destructor Documentation	22
4.14.1.1	FunctionsBuilder	22
4.14.2	Member Function Documentation	22
4.14.2.1	buildFunctions	22
4.15	FunctionsStructure Class Reference	22
4.15.1	Member Function Documentation	23
4.15.1.1	getParametersCount	23

4.15.1.2	getPossibleValues	23
4.15.1.3	getRegulationsCount	23
4.15.1.4	getSourceSpecies	23
4.15.1.5	getSourceValues	23
4.15.1.6	getSpecieName	23
4.15.1.7	getSpeciesCount	23
4.15.1.8	getSpecieValues	23
4.15.1.9	getStepSize	24
4.16	GraphInterface Class Reference	24
4.16.1	Constructor & Destructor Documentation	24
4.16.1.1	~GraphInterface	24
4.16.2	Member Function Documentation	24
4.16.2.1	getStateCount	24
4.16.2.2	getString	24
4.16.2.3	getTargetID	25
4.16.2.4	getTransitionCount	25
4.17	Model::Interaction Struct Reference	25
4.18	rapidxml::memory_pool< Ch > Class Template Reference	26
4.18.1	Detailed Description	26
4.18.2	Constructor & Destructor Documentation	27
4.18.2.1	~memory_pool	27
4.18.3	Member Function Documentation	27
4.18.3.1	allocate_attribute	27
4.18.3.2	allocate_node	27
4.18.3.3	allocate_string	28
4.18.3.4	clear	28
4.18.3.5	clone_node	28
4.18.3.6	set_allocator	28
4.19	Model Class Reference	29
4.19.1	Member Function Documentation	29
4.19.1.1	findID	29
4.19.1.2	getEdges	30
4.19.1.3	getInteractions	30
4.19.1.4	getMax	30
4.19.1.5	getMin	30
4.19.1.6	getName	30
4.19.1.7	getRegulations	30
4.19.1.8	getSpeciesCount	30
4.19.1.9	getStateCount	30
4.19.1.10	isFinal	31

4.20	ModelChecker Class Reference	31
4.20.1	Constructor & Destructor Documentation	31
4.20.1.1	ModelChecker	31
4.20.2	Member Function Documentation	31
4.20.2.1	startColoring	31
4.20.2.2	startColoring	31
4.21	ModelParser Class Reference	32
4.21.1	Constructor & Destructor Documentation	32
4.21.1.1	ModelParser	32
4.21.2	Member Function Documentation	32
4.21.2.1	parseInput	32
4.22	rapidxml::node_iterator< Ch > Class Template Reference	32
4.22.1	Detailed Description	33
4.23	OutputManager Class Reference	33
4.23.1	Member Function Documentation	33
4.23.1.1	outputRound	33
4.23.1.2	outputRoundNum	33
4.23.1.3	outputSummary	33
4.24	OutputStreamer Class Reference	33
4.24.1	Constructor & Destructor Documentation	34
4.24.1.1	OutputStreamer	34
4.24.1.2	~OutputStreamer	34
4.24.2	Member Function Documentation	34
4.24.2.1	createStreamFile	34
4.24.2.2	isResultInFile	34
4.24.2.3	output	34
4.24.2.4	output	35
4.24.2.5	testTrait	35
4.25	ParametrizedStructure Class Reference	35
4.25.1	Member Function Documentation	36
4.25.1.1	getStateCount	36
4.25.1.2	getStateLevels	36
4.25.1.3	getStepSize	36
4.25.1.4	getString	36
4.25.1.5	getTargetID	36
4.25.1.6	getTransitionCount	36
4.25.1.7	getTransitive	36
4.26	ParametrizedStructureBuilder Class Reference	37
4.26.1	Constructor & Destructor Documentation	37
4.26.1.1	ParametrizedStructureBuilder	37

4.26.2	Member Function Documentation	37
4.26.2.1	buildStructure	37
4.27	rapidxml::parse_error Class Reference	37
4.27.1	Detailed Description	37
4.27.2	Member Function Documentation	38
4.27.2.1	what	38
4.27.2.2	where	38
4.28	PerColorStorage Class Reference	38
4.28.1	Member Function Documentation	38
4.28.1.1	getNeighbours	38
4.29	ProductBuilder Class Reference	39
4.29.1	Constructor & Destructor Documentation	39
4.29.1.1	ProductBuilder	39
4.29.2	Member Function Documentation	39
4.29.2.1	buildProduct	39
4.30	ProductStructure Class Reference	39
4.30.1	Member Function Documentation	40
4.30.1.1	getBA	40
4.30.1.2	getBAID	40
4.30.1.3	getCons	40
4.30.1.4	getFinalStates	40
4.30.1.5	getFunc	40
4.30.1.6	getInitialStates	41
4.30.1.7	getKS	41
4.30.1.8	getKSID	41
4.30.1.9	getProductID	41
4.30.1.10	getStateCount	41
4.30.1.11	getStateLevels	41
4.30.1.12	getStepSize	41
4.30.1.13	getString	41
4.30.1.14	getTargetID	42
4.30.1.15	getTransitionCount	42
4.30.1.16	getTransitive	42
4.30.1.17	isFinal	42
4.30.1.18	isInitial	42
4.31	SplitManager Class Reference	42
4.31.1	Constructor & Destructor Documentation	43
4.31.1.1	SplitManager	43
4.31.2	Member Function Documentation	43
4.31.2.1	createStartingParameters	43

4.31.2.2	getAllColorsCount	43
4.31.2.3	getProcColorsCount	43
4.31.2.4	getRoundCount	43
4.31.2.5	getRoundNum	43
4.31.2.6	getRoundRange	43
4.31.2.7	getRoundSize	44
4.31.2.8	increaseRound	44
4.31.2.9	lastRound	44
4.31.2.10	setStartPositions	44
4.31.2.11	valid	44
4.32	SynthesisManager Class Reference	44
4.32.1	Constructor & Destructor Documentation	44
4.32.1.1	SynthesisManager	44
4.32.2	Member Function Documentation	44
4.32.2.1	doSynthesis	44
4.33	TimeManager Class Reference	45
4.33.1	Member Function Documentation	45
4.33.1.1	ouputClock	45
4.33.1.2	startClock	45
4.34	UserOptions Class Reference	45
4.34.1	Constructor & Destructor Documentation	46
4.34.1.1	UserOptions	46
4.34.2	Member Function Documentation	46
4.34.2.1	BA	46
4.34.2.2	coloring	46
4.34.2.3	displayWintess	46
4.34.2.4	negation	46
4.34.2.5	procCount	46
4.34.2.6	procNum	46
4.34.2.7	robustness	46
4.34.2.8	stats	47
4.34.2.9	timeSerie	47
4.34.2.10	verbose	47
4.34.2.11	witnesses	47
4.35	WitnessSearcher Class Reference	47
4.35.1	Constructor & Destructor Documentation	47
4.35.1.1	WitnessSearcher	47
4.35.2	Member Function Documentation	47
4.35.2.1	display	47
4.36	rapidxml::xml_attribute< Ch > Class Template Reference	48

4.36.1 Detailed Description	48
4.36.2 Constructor & Destructor Documentation	48
4.36.2.1 xml_attribute	48
4.36.3 Member Function Documentation	48
4.36.3.1 document	48
4.36.3.2 next_attribute	49
4.36.3.3 previous_attribute	49
4.37 rapidxml::xml_base< Ch > Class Template Reference	49
4.37.1 Detailed Description	50
4.37.2 Member Function Documentation	50
4.37.2.1 name	50
4.37.2.2 name	50
4.37.2.3 name	51
4.37.2.4 name_size	51
4.37.2.5 parent	51
4.37.2.6 value	51
4.37.2.7 value	51
4.37.2.8 value	52
4.37.2.9 value_size	52
4.38 rapidxml::xml_document< Ch > Class Template Reference	52
4.38.1 Detailed Description	52
4.38.2 Member Function Documentation	53
4.38.2.1 clear	53
4.38.2.2 parse	53
4.39 rapidxml::xml_node< Ch > Class Template Reference	53
4.39.1 Detailed Description	54
4.39.2 Constructor & Destructor Documentation	54
4.39.2.1 xml_node	54
4.39.3 Member Function Documentation	55
4.39.3.1 append_attribute	55
4.39.3.2 append_node	55
4.39.3.3 document	55
4.39.3.4 first_attribute	55
4.39.3.5 first_node	55
4.39.3.6 insert_attribute	56
4.39.3.7 insert_node	56
4.39.3.8 last_attribute	56
4.39.3.9 last_node	57
4.39.3.10 next_sibling	57
4.39.3.11 prepend_attribute	57

4.39.3.12	prepend_node	57
4.39.3.13	previous_sibling	58
4.39.3.14	remove_attribute	58
4.39.3.15	remove_first_attribute	58
4.39.3.16	remove_first_node	58
4.39.3.17	remove_last_attribute	58
4.39.3.18	remove_last_node	58
4.39.3.19	type	59
4.39.3.20	type	59
5	File Documentation	61
5.1	parsing/rapidxml-1.13/rapidxml.hpp File Reference	61
5.1.1	Detailed Description	62
5.2	parsing/rapidxml-1.13/rapidxml_iterators.hpp File Reference	62
5.2.1	Detailed Description	62
5.3	parsing/rapidxml-1.13/rapidxml_print.hpp File Reference	63
5.3.1	Detailed Description	63
5.4	parsing/rapidxml-1.13/rapidxml_utils.hpp File Reference	63
5.4.1	Detailed Description	64

Chapter 1

Class Index

1.1 Class Hierarchy

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

ArgumentParser	7
rapidxml::attribute_iterator< Ch >	7
AutomatonBuilder	8
BasicStructureBuilder	14
ClassName	14
ColoringAnalyzer	14
ColoringParser	15
ColorStorage	17
ConstrainsParser	19
rapidxml::file< Ch >	20
FunctionsBuilder	22
FunctionsStructure	22
GraphInterface	24
AutomatonInterface	9
AutomatonStructure	10
ProductStructure	39
BasicStructure	12
ParametrizedStructure	35
Model::Interaction	25
rapidxml::memory_pool< Ch >	26
rapidxml::xml_document< Ch >	52
Model	29
ModelChecker	31
ModelParser	32
rapidxml::node_iterator< Ch >	32
OutputManager	33
OutputStreamer	33
ParametrizedStructureBuilder	37
rapidxml::parse_error	37
PerColorStorage	38
ProductBuilder	39
SplitManager	42
SynthesisManager	44
TimeManager	45
UserOptions	45
WitnessSearcher	47
rapidxml::xml_base< Ch >	49

rapidxml::xml_attribute< Ch >	48
rapidxml::xml_node< Ch >	53
rapidxml::xml_document< Ch >	52

Chapter 2

Class Index

2.1 Class List

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

ArgumentParser	7
rapidxml::attribute_iterator< Ch >	
Iterator of child attributes of xml_node	7
AutomatonBuilder	8
AutomatonInterface	9
AutomatonStructure	10
BasicStructure	12
BasicStructureBuilder	14
ClassName	14
ColoringAnalyzer	14
ColoringParser	15
ColorStorage	17
ConstrainsParser	19
rapidxml::file< Ch >	
Represents data loaded from a file	20
FunctionsBuilder	22
FunctionsStructure	22
GraphInterface	24
Model::Interaction	25
rapidxml::memory_pool< Ch >	26
Model	29
ModelChecker	31
ModelParser	32
rapidxml::node_iterator< Ch >	
Iterator of child nodes of xml_node	32
OutputManager	33
OutputStreamer	33
ParametrizedStructure	35
ParametrizedStructureBuilder	37
rapidxml::parse_error	37
PerColorStorage	38
ProductBuilder	39
ProductStructure	39
SplitManager	42
SynthesisManager	44
TimeManager	45
UserOptions	45
WitnessSearcher	47

rapidxml::xml_attribute< Ch >	48
rapidxml::xml_base< Ch >	49
rapidxml::xml_document< Ch >	52
rapidxml::xml_node< Ch >	53

Chapter 3

File Index

3.1 File List

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

auxiliary/ common_functions.hpp	??
auxiliary/ data_types.hpp	??
auxiliary/ output_streamer.hpp	??
auxiliary/ template.hpp	??
auxiliary/ time_manager.hpp	??
auxiliary/ user_options.hpp	??
coloring/ model_checker.hpp	??
coloring/ parameters_functions.hpp	??
coloring/ split_manager.hpp	??
coloring/ synthesis_manager.hpp	??
parsing/ argument_parser.hpp	??
parsing/ coloring_parser.hpp	??
parsing/ constrains_parser.hpp	??
parsing/ model.hpp	??
parsing/ model_parser.hpp	??
parsing/rapidxml-1.13/ rapidxml.hpp	
This file contains rapidxml parser and DOM implementation	61
parsing/rapidxml-1.13/ rapidxml_iterators.hpp	
This file contains rapidxml iterators	62
parsing/rapidxml-1.13/ rapidxml_print.hpp	
This file contains rapidxml printer implementation	63
parsing/rapidxml-1.13/ rapidxml_utils.hpp	63
reforging/ automaton_builder.hpp	??
reforging/ automaton_interface.hpp	??
reforging/ automaton_structure.hpp	??
reforging/ basic_structure.hpp	??
reforging/ basic_structure_builder.hpp	??
reforging/ color_storage.hpp	??
reforging/ functions_builder.hpp	??
reforging/ functions_structure.hpp	??
reforging/ graph_interface.hpp	??
reforging/ parametrized_structure.hpp	??
reforging/ parametrized_structure_builder.hpp	??
reforging/ product_builder.hpp	??
reforging/ product_structure.hpp	??
results/ coloring_analyzer.hpp	??
results/ output_manager.hpp	??
results/ per_color_storage.hpp	??

results/witness_searcher.hpp ??

Chapter 4

Class Documentation

4.1 ArgumentParser Class Reference

```
#include <argument_parser.hpp>
```

Public Member Functions

- void [parseArguments](#) ([UserOptions](#) &user_options, int argc, char *argv[])

4.1.1 Detailed Description

Parses arguments on the input and changes switches accordingly. If there is a file on the input, it is created.

Parameters

<i>user_options</i>	parsed data will be saved here
<i>argc</i>	same as at main
<i>argv</i>	same as at main
<i>result_stream</i>	pointer to the stream that will get the output

4.1.2 Member Function Documentation

4.1.2.1 void [ArgumentParser::parseArguments](#) ([UserOptions](#) & *user_options*, int *argc*, char * *argv*[]) [\[inline\]](#)

Take all the arguments on the input and store information from them

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

- parsing/argument_parser.hpp

4.2 rapidxml::attribute_iterator< Ch > Class Template Reference

Iterator of child attributes of [xml_node](#).

```
#include <rapidxml_iterators.hpp>
```

Public Types

- typedef `rapidxml::xml_attribute< Ch > value_type`
- typedef `rapidxml::xml_attribute< Ch > & reference`
- typedef `rapidxml::xml_attribute< Ch > * pointer`
- typedef `std::ptrdiff_t difference_type`
- typedef `std::bidirectional_iterator_tag iterator_category`

Public Member Functions

- `attribute_iterator` (`xml_node< Ch > *node`)
- `reference operator*` () const
- `pointer operator->` () const
- `attribute_iterator & operator++` ()
- `attribute_iterator operator++` (int)
- `attribute_iterator & operator--` ()
- `attribute_iterator operator--` (int)
- bool `operator==` (const `attribute_iterator< Ch > &rhs`)
- bool `operator!=` (const `attribute_iterator< Ch > &rhs`)

4.2.1 Detailed Description

`template<class Ch>class rapidxml::attribute_iterator< Ch >`

Iterator of child attributes of `xml_node`.

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

- `parsing/rapidxml-1.13/rapidxml_iterators.hpp`

4.3 AutomatonBuilder Class Reference

Public Member Functions

- `AutomatonBuilder` (const `Model` &`_model`, `AutomatonStructure` &`_automaton`)
- void `buildAutomaton` ()

4.3.1 Constructor & Destructor Documentation

4.3.1.1 `AutomatonBuilder::AutomatonBuilder (const Model & _model, AutomatonStructure & _automaton)`
`[inline]`

Constructor just attaches the references to data holders

4.3.2 Member Function Documentation

4.3.2.1 void AutomatonBuilder::buildAutomaton () [inline]

Create the transitions from the model and fill the automaton with them

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

- [reforging/automaton_builder.hpp](#)

4.4 AutomatonInterface Class Reference

Inherits [GraphInterface](#).

Inherited by [AutomatonStructure](#), and [ProductStructure](#).

Public Member Functions

- virtual const bool [isFinal](#) (const StateID ID) const =0
- virtual const bool [isInitial](#) (const StateID ID) const =0
- virtual const std::vector< StateID > & [getFinalStates](#) () const =0
- virtual const std::vector< StateID > & [getInitialStates](#) () const =0
- virtual [~AutomatonInterface](#) ()

4.4.1 Constructor & Destructor Documentation

4.4.1.1 virtual AutomatonInterface::~~AutomatonInterface () [inline], [virtual]

Virtual destructor.

4.4.2 Member Function Documentation

4.4.2.1 virtual const std::vector<StateID>& AutomatonInterface::getFinalStates () const [inline], [pure virtual]

Get IDs of all states that are marked as final.

Returns

vector of final states' IDs

Implemented in [AutomatonStructure](#), and [ProductStructure](#).

4.4.2.2 virtual const std::vector<StateID>& AutomatonInterface::getInitialStates () const [inline], [pure virtual]

Get IDs of all states that are marked as initial.

Returns

vector of initial states' IDs

Implemented in [AutomatonStructure](#), and [ProductStructure](#).

4.4.2.3 `virtual const bool AutomatonInterface::isFinal (const StateID ID) const` `[inline],[pure virtual]`

For given state find out if it is marked as final.

Parameters

<i>ID</i>	state to test
-----------	---------------

Returns

true if the state is final

Implemented in [AutomatonStructure](#), and [ProductStructure](#).

4.4.2.4 `virtual const bool AutomatonInterface::isInitial (const StateID ID) const` `[inline],[pure virtual]`

For given state find out if it is marked as initial.

Parameters

<i>ID</i>	state to test
-----------	---------------

Returns

true if the state is initial

Implemented in [AutomatonStructure](#), and [ProductStructure](#).

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

- `reforging/automaton_interface.hpp`

4.5 AutomatonStructure Class Reference

Inherits [AutomatonInterface](#).

Classes

- struct **State**
- struct **Transition**

Public Member Functions

- bool [isTransitionFeasible](#) (const std::size_t state_num, const std::size_t transition_num, const Levels &levels) const
- const std::size_t [getStateCount](#) () const
- const std::size_t [getTransitionCount](#) (const StateID ID) const
- const std::size_t [getTargetID](#) (const StateID ID, const std::size_t transition_num) const
- const std::string [getString](#) (const StateID ID) const
- virtual const bool [isFinal](#) (const StateID ID) const
- virtual const bool [isInitial](#) (const StateID ID) const
- virtual const std::vector
< StateID > & [getFinalStates](#) () const

- virtual const std::vector
 < StateID > & [getInitialStates](#) () const
- const std::vector< std::set
 < std::size_t > > & [getAllowedValues](#) (const StateID ID, const std::size_t transition_num) const

Friends

- class **AutomatonBuilder**

4.5.1 Member Function Documentation

4.5.1.1 const std::vector<std::set<std::size_t> > & AutomatonStructure::getAllowedValues (const StateID ID, const std::size_t transition_num) const [\[inline\]](#)

Parameters

<i>transition_num</i>	number of transition to get the data from
-----------------------	---

Returns

ID of the target state of this transition

4.5.1.2 virtual const std::vector<StateID>& AutomatonStructure::getFinalStates () const [\[inline\]](#),[\[virtual\]](#)

Implements [AutomatonInterface](#).

4.5.1.3 virtual const std::vector<StateID>& AutomatonStructure::getInitialStates () const [\[inline\]](#),[\[virtual\]](#)

Only the first state is considered initial.

Implements [AutomatonInterface](#).

4.5.1.4 const std::size_t AutomatonStructure::getStateCount () const [\[inline\]](#),[\[virtual\]](#)

Implements [GraphInterface](#).

4.5.1.5 const std::string AutomatonStructure::getString (const StateID ID) const [\[inline\]](#),[\[virtual\]](#)

Return string representing the state in the form: (ID).

Implements [GraphInterface](#).

4.5.1.6 const std::size_t AutomatonStructure::getTargetID (const StateID ID, const std::size_t transition_num) const [\[inline\]](#),[\[virtual\]](#)

Implements [GraphInterface](#).

4.5.1.7 const std::size_t AutomatonStructure::getTransitionCount (const StateID ID) const [\[inline\]](#),[\[virtual\]](#)

Implements [GraphInterface](#).

4.5.1.8 `virtual const bool AutomatonStructure::isFinal (const StateID ID) const` `[inline],[virtual]`

Implements [AutomatonInterface](#).

4.5.1.9 `virtual const bool AutomatonStructure::isInitial (const StateID ID) const` `[inline],[virtual]`

Only the first state is considered initial.

Implements [AutomatonInterface](#).

4.5.1.10 `bool AutomatonStructure::isTransitionFeasible (const std::size_t state_num, const std::size_t transition_num, const Levels & levels) const` `[inline]`

Parameters

<i>levels</i>	current levels of species i.e. the state of the KS
---------------	--

Returns

true if the transition is feasible

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

- `reforging/automaton_structure.hpp`

4.6 BasicStructure Class Reference

Inherits [GraphInterface](#).

Classes

- struct **State**
- struct **Transition**

Public Member Functions

- `const std::size_t getStateCount () const`
- `const std::size_t getTransitionCount (const StateID ID) const`
- `const std::size_t getTargetID (const StateID ID, const std::size_t trans_number) const`
- `const std::string getString (const StateID ID) const`
- `const Levels & getStateLevels (const StateID ID) const`
- `const std::size_t getSpecieID (const StateID ID, const std::size_t neighbour_index) const`
- `const Direction getDirection (const StateID ID, const std::size_t neighbour_index) const`

Friends

- class **BasicStructureBuilder**

4.6.1 Member Function Documentation

4.6.1.1 `const Direction BasicStructure::getDirection (const StateID ID, const std::size_t neighbour_index) const` `[inline]`

Parameters

<i>ID</i>	ID of the state to get the neighbour from
<i>neighbour_index</i>	index in the vector of neighbours

Returns

Direction in which the specie changes

4.6.1.2 `const std::size_t BasicStructure::getSpecieID (const StateID ID, const std::size_t neighbour_index) const` `[inline]`

Parameters

<i>ID</i>	ID of the state to get the neighbour from
<i>neighbour_index</i>	index in the vector of neighbours

Returns

ID of the specie that vary between the two states

4.6.1.3 `const std::size_t BasicStructure::getStateCount () const` `[inline]`, `[virtual]`

Implements [GraphInterface](#).

4.6.1.4 `const Levels& BasicStructure::getStateLevels (const StateID ID) const` `[inline]`

Parameters

<i>ID</i>	ID of the state to get
-----------	------------------------

Returns

levels of the state

4.6.1.5 `const std::string BasicStructure::getString (const StateID ID) const` `[inline]`, `[virtual]`

Unused, return empty string.

Implements [GraphInterface](#).

4.6.1.6 `const std::size_t BasicStructure::getTargetID (const StateID ID, const std::size_t trans_number) const` `[inline]`, `[virtual]`

Implements [GraphInterface](#).

4.6.1.7 `const std::size_t BasicStructure::getTransitionCount (const StateID ID) const` `[inline]`, `[virtual]`

Implements [GraphInterface](#).

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

- `reforging/basic_structure.hpp`

4.7 BasicStructureBuilder Class Reference

Public Member Functions

- [BasicStructureBuilder](#) (const [Model](#) &_model, [BasicStructure](#) &_structure)
- void [buildStructure](#) ()

4.7.1 Constructor & Destructor Documentation

4.7.1.1 [BasicStructureBuilder::BasicStructureBuilder](#) (const [Model](#) & _model, [BasicStructure](#) & _structure)
[inline]

Constructor initializes basic information from the model

4.7.2 Member Function Documentation

4.7.2.1 void [BasicStructureBuilder::buildStructure](#) () [inline]

Create the states from the model and fill the structure with them.

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

- [reforging/basic_structure_builder.hpp](#)

4.8 ClassName Class Reference

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

- [auxiliary/template.hpp](#)

4.9 ColoringAnalyzer Class Reference

Public Member Functions

- [ColoringAnalyzer](#) (const [ProductStructure](#) &_product)
- void [startNewRound](#) (const Range &round_range)
- void **display** () const
- void [storeResults](#) (const Coloring &results)
- const Parameters [getUnion](#) () const
- std::vector< std::pair
< std::size_t, std::string > > [getColors](#) (Parameters result_parameters) const
- std::vector< std::pair
< std::size_t, std::string > > [getColors](#) () const

4.9.1 Constructor & Destructor Documentation

4.9.1.1 [ColoringAnalyzer::ColoringAnalyzer](#) (const [ProductStructure](#) & _product) [inline]

Get reference data and create final states that will hold all the computed data

4.9.2 Member Function Documentation

4.9.2.1 `std::vector<std::pair<std::size_t, std::string> > ColoringAnalyzer::getColors (Parameters result_parameters) const` `[inline]`

Obtain colors given parameters in the form [fun1, fun2, ...] for specified parameters

Parameters

<i>result_parameters</i>	parameters to use
--------------------------	-------------------

Returns

vector of masks and strings of feasible colors

4.9.2.2 `std::vector<std::pair<std::size_t, std::string> > ColoringAnalyzer::getColors () const` `[inline]`

Obtain colors given parameters in the form [fun1, fun2, ...] for all parameters in this round

Returns

vector of numbers and strings of colors

4.9.2.3 `const Parameters ColoringAnalyzer::getUnion () const` `[inline]`

Compute merge of all final colors creating a coloring with all feasible colors

Returns

all feasible colors in this round

4.9.2.4 `void ColoringAnalyzer::storeResults (const Coloring & results)` `[inline]`

Store requested results for a give state of product

4.9.2.5 `void ColoringAnalyzer::startNewRound (const Range & round_range)` `[inline]`

Iterates color until it responds to the first parameter of this round

Parameters

<i>round_range</i>	first and one behind last parameter of this round
--------------------	---

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

- results/coloring_analyzer.hpp

4.10 ColoringParser Class Reference

Public Member Functions

- [ColoringParser](#) ()
- void [openFile](#) (const std::string filename)
- void [createOutput](#) (const std::string filename)
- void [parseMask](#) ()
- void [outputComputed](#) (const Parameters parameters)
- const bool [input](#) () const
- const bool [output](#) () const
- const std::vector< Parameters > & [getColors](#) () const
- const std::size_t [getColorsCount](#) ()

4.10.1 Constructor & Destructor Documentation

4.10.1.1 ColoringParser::ColoringParser () [inline]

Basic constructor - should be used only for the single object shared throught the program

4.10.2 Member Function Documentation

4.10.2.1 void ColoringParser::createOutput (const std::string *filename*) [inline]

Create a file to output bitmasks to.

Parameters

<i>filename</i>	path to the file to read from
-----------------	-------------------------------

4.10.2.2 const std::vector<Parameters>& ColoringParser::getColors () const [inline]

Returns

masks for all colors that can be used

4.10.2.3 const std::size_t ColoringParser::getColorsCount () [inline]

Returns

number of Parameters e.g. number of rounds of computation

4.10.2.4 const bool ColoringParser::input () const [inline]

Returns

true if the mask was provided on the input

4.10.2.5 void ColoringParser::openFile (const std::string *filename*) [inline]

Only opens the file with the data stream.

Parameters

<i>filename</i>	path to the file to read from
-----------------	-------------------------------

4.10.2.6 `const bool ColoringParser::output () const` `[inline]`

Returns

true if the mask is requested on the output

4.10.2.7 `void ColoringParser::outputComputed (const Parameters parameters)` `[inline]`

Send computed data for this round on the ouput

Parameters

<i>parameters</i>	bitmask of computed feasible colors
-------------------	-------------------------------------

4.10.2.8 `void ColoringParser::parseMask ()` `[inline]`

Main parsing function that creates parameters vector.

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

- parsing/coloring_parser.hpp

4.11 ColorStorage Class Reference

Classes

- struct **State**

Public Member Functions

- void [reset](#) ()
- void [addFrom](#) (const [ColorStorage](#) &other)
- bool [update](#) (const Parameters parameters, const StateID ID)
- bool [soft_update](#) (const Parameters parameters, const StateID ID)
- bool [update](#) (const StateID source_ID, const Parameters parameters, const StateID target_ID)
- const Parameters & [getColor](#) (const StateID ID) const
- const std::vector< Coloring > [getColor](#) (const std::vector< StateID > &states) const
- std::set< StateID > [getColored](#) () const
- const Neighbours [getNeighbours](#) (const StateID ID, const bool successors, const Parameters color_mask=~0) const
- const std::vector< Parameters > [getMarking](#) (const StateID ID, const bool successors) const

Friends

- class **ProductBuilder**

4.11.1 Member Function Documentation

4.11.1.1 `void ColorStorage::addFrom (const ColorStorage & other)` `[inline]`

Add all values from one coloring structure to another

Parameters

<i>other</i>	structure to copy from
--------------	------------------------

4.11.1.2 `const Parameters& ColorStorage::getColor (const StateID ID) const` `[inline]`

Parameters

<i>ID</i>	index of the state to ask for parameters
-----------	--

Returns

parameters assigned to the state

4.11.1.3 `const std::vector<Coloring> ColorStorage::getColor (const std::vector< StateID > & states) const` `[inline]`

Parameters

<i>states</i>	indexes of states to ask for parameters
---------------	---

Returns

queue with all colorings of states

4.11.1.4 `const std::vector<Parameters> ColorStorage::getMarking (const StateID ID, const bool successors) const`
`[inline]`

Get all the neighbours for this color from this state.

Parameters

<i>ID</i>	index of the state to ask for predecessors
<i>successors</i>	true if successors are required, false if predecessors

Returns

neighbours for given state and their color

4.11.1.5 `const Neighbours ColorStorage::getNeighbours (const StateID ID, const bool successors, const Parameters color_mask = ~0) const` `[inline]`

Get all the neighbours for this color from this state.

Parameters

<i>ID</i>	index of the state to ask for predecessors
<i>successors</i>	true if successors are required, false if predecessors
<i>color_mask</i>	bitmask for a given color, if it is not specified, all colors are required

Returns

neighbours for given state

4.11.1.6 void ColorStorage::reset () [inline]

Sets all values for all the states to zero

4.11.1.7 bool ColorStorage::update (const Parameters *parameters*, const StateID *ID*) [inline]

Add passed colors to the state

Parameters

<i>ID</i>	index of the state to fill
<i>parameters</i>	to add - if empty, add all, otherwise use bitwise or

Returns

true if there was an actual update

4.11.1.8 bool ColorStorage::update (const StateID *source_ID*, const Parameters *parameters*, const StateID *target_ID*) [inline]

Add passed colors to the state

Parameters

<i>source_ID</i>	index of the state that passed this update
<i>target_ID</i>	index of the state to fill
<i>parameters</i>	to add - if empty, add all, otherwise use bitwise or

Returns

true if there was an actual update

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

- reforging/color_storage.hpp

4.12 ConstrainsParser Class Reference

Classes

- struct **SpecieColors**

Public Member Functions

- **ConstrainsParser** (const [Model](#) &_model)
- void [parseConstrains](#) ()
- const std::size_t [getSpecieNum](#) () const
- const std::size_t [getAllColorsNum](#) (const SpecieID ID) const
- const std::size_t [getColorsNum](#) (const SpecieID ID) const
- const std::vector< std::size_t > & [getColor](#) (const SpecieID ID, const std::size_t color_num) const
- const std::vector< std::size_t > [getTargetVals](#) (const SpecieID ID, const std::size_t regul_num) const

4.12.1 Member Function Documentation

4.12.1.1 `const std::size_t ConstrainsParser::getAllColorsNum (const SpecieID ID) const` `[inline]`

Returns

total number of subcolors this specie could have (all regulatory contexts' combinations)

4.12.1.2 `const std::vector<std::size_t>& ConstrainsParser::getColor (const SpecieID ID, const std::size_t color_num) const` `[inline]`

Returns

requested subcolor from the vector of subcolors of given specie

4.12.1.3 `const std::size_t ConstrainsParser::getColorsNum (const SpecieID ID) const` `[inline]`

Returns

total number of subcolors this specie has (allowed regulatory contexts' combinations)

4.12.1.4 `const std::size_t ConstrainsParser::getSpecieNum () const` `[inline]`

Returns

total number of species

4.12.1.5 `const std::vector<std::size_t> ConstrainsParser::getTargetVals (const SpecieID ID, const std::size_t regul_num) const` `[inline]`

Returns

total number of subcolors this specie has (allowed regulatory contexts' combinations)

4.12.1.6 `void ConstrainsParser::parseConstrains ()` `[inline]`

Entry function of parsing, tests and stores subcolors for all the species

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

- parsing/constrains_parser.hpp

4.13 rapidxml::file< Ch > Class Template Reference

Represents data loaded from a file.

```
#include <rapidxml_utils.hpp>
```

Public Member Functions

- `file` (const char *filename)
- `file` (std::basic_istream< Ch > &stream)
- Ch * `data` ()
- const Ch * `data` () const
- std::size_t `size` () const

4.13.1 Detailed Description

template<class Ch = char>class rapidxml::file< Ch >

Represents data loaded from a file.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 template<class Ch = char> rapidxml::file< Ch >::file (const char * *filename*) [inline]

Loads file into the memory. Data will be automatically destroyed by the destructor.

Parameters

<i>filename</i>	Filename to load.
-----------------	-------------------

4.13.2.2 template<class Ch = char> rapidxml::file< Ch >::file (std::basic_istream< Ch > & *stream*) [inline]

Loads file into the memory. Data will be automatically destroyed by the destructor

Parameters

<i>stream</i>	Stream to load from
---------------	---------------------

4.13.3 Member Function Documentation

4.13.3.1 template<class Ch = char> Ch* rapidxml::file< Ch >::data () [inline]

Gets file data.

Returns

Pointer to data of file.

4.13.3.2 template<class Ch = char> const Ch* rapidxml::file< Ch >::data () const [inline]

Gets file data.

Returns

Pointer to data of file.

4.13.3.3 template<class Ch = char> std::size_t rapidxml::file< Ch >::size () const [inline]

Gets file data size.

Returns

Size of file data, in characters.

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

- [parsing/rapidxml-1.13/rapidxml_utils.hpp](#)

4.14 FunctionsBuilder Class Reference

Public Member Functions

- [FunctionsBuilder](#) (const [Model](#) &_model, const [ConstrainsParser](#) &_constrains, [FunctionsStructure](#) &_functions_structure)
- void [buildFunctions](#) ()

4.14.1 Constructor & Destructor Documentation

4.14.1.1 [FunctionsBuilder::FunctionsBuilder](#) (const [Model](#) &_model, const [ConstrainsParser](#) &_constrains, [FunctionsStructure](#) &_functions_structure) [\[inline\]](#)

Constructor just attaches the references to data holders

4.14.2 Member Function Documentation

4.14.2.1 void [FunctionsBuilder::buildFunctions](#) () [\[inline\]](#)

For each specie recreate all its regulatory functions

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

- [reforging/functions_builder.hpp](#)

4.15 FunctionsStructure Class Reference

Classes

- struct **RegulatoryFunction**
- struct **Specie**

Public Member Functions

- const std::size_t [getParametersCount](#) () const
- const std::size_t [getSpeciesCount](#) () const
- const std::string & [getSpecieName](#) (const std::size_t ID) const
- const std::vector< std::size_t > & [getSpecieValues](#) (const std::size_t ID) const
- const std::vector< std::size_t > & [getSourceSpecies](#) (const std::size_t ID) const
- const std::size_t [getRegulationsCount](#) (const std::size_t ID) const
- const std::size_t [getStepSize](#) (const std::size_t ID, const std::size_t regulation) const
- const std::vector< std::size_t > & [getPossibleValues](#) (const std::size_t ID, const std::size_t regulation) const
- const std::vector< std::vector< std::size_t > > & [getSourceValues](#) (const std::size_t ID, const std::size_t regulation) const

Friends

- class **FunctionsBuilder**

4.15.1 Member Function Documentation

4.15.1.1 `const std::size_t FunctionsStructure::getParametersCount () const [inline]`

Returns

size of the parameter space

4.15.1.2 `const std::vector<std::size_t>& FunctionsStructure::getPossibleValues (const std::size_t ID, const std::size_t regulation) const [inline]`

Returns

values this function can possibly regulate to

4.15.1.3 `const std::size_t FunctionsStructure::getRegulationsCount (const std::size_t ID) const [inline]`

Returns

number of regulations for this specie (two to power of number of source species)

4.15.1.4 `const std::vector<std::size_t>& FunctionsStructure::getSourceSpecies (const std::size_t ID) const [inline]`

Returns

IDs of all the species that regulate this specie

4.15.1.5 `const std::vector<std::vector<std::size_t> >& FunctionsStructure::getSourceValues (const std::size_t ID, const std::size_t regulation) const [inline]`

Returns

for each source specie all the values that if it is within them, it allows this function

4.15.1.6 `const std::string& FunctionsStructure::getSpecieName (const std::size_t ID) const [inline]`

Returns

name of the specie with given ID

4.15.1.7 `const std::size_t FunctionsStructure::getSpeciesCount () const [inline]`

Returns

number of the species

4.15.1.8 `const std::vector<std::size_t>& FunctionsStructure::getSpecieValues (const std::size_t ID) const [inline]`

Returns

all the values the specie can occur in

4.15.1.9 `const std::size_t FunctionsStructure::getStepSize (const std::size_t ID, const std::size_t regulation) const`
`[inline]`

Returns

`step_size` (how many neighbour parameters share the same value for this regulation)

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

- `reforging/functions_structure.hpp`

4.16 GraphInterface Class Reference

Inherited by [AutomatonInterface](#), [BasicStructure](#), and [ParametrizedStructure](#).

Public Member Functions

- virtual const std::size_t [getStateCount](#) () const =0
- virtual const std::size_t [getTransitionCount](#) (const StateID ID) const =0
- virtual const StateID [getTargetID](#) (const StateID ID, const std::size_t transition_number) const =0
- virtual const std::string [getString](#) (const std::size_t StateID) const =0
- virtual [~GraphInterface](#) ()

4.16.1 Constructor & Destructor Documentation

4.16.1.1 `virtual GraphInterface::~~GraphInterface ()` `[inline]`, `[virtual]`

Virtual destructor.

4.16.2 Member Function Documentation

4.16.2.1 `virtual const std::size_t GraphInterface::getStateCount () const` `[inline]`, `[pure virtual]`

Obtains number of states of the graph.

Returns

integer with size of the graph

Implemented in [AutomatonStructure](#), [ProductStructure](#), [ParametrizedStructure](#), and [BasicStructure](#).

4.16.2.2 `virtual const std::string GraphInterface::getString (const std::size_t StateID) const` `[inline]`, `[pure virtual]`

Returns given state as a string.

Parameters

<i>ID</i>	ID of the state to turn into the string
-----------	---

Returns

given state as a string

Implemented in [AutomatonStructure](#), [ProductStructure](#), [ParametrizedStructure](#), and [BasicStructure](#).

4.16.2.3 `virtual const StateID GraphInterface::getTargetID (const StateID ID, const std::size_t transition_number) const [inline], [pure virtual]`

Obtains ID of the target of given transition for given state.

Parameters

<i>ID</i>	ID of the state to get the neighbour from
<i>trans_number</i>	index in the vector of transitions

Returns

ID of the requested target

Implemented in [AutomatonStructure](#), [ProductStructure](#), [ParametrizedStructure](#), and [BasicStructure](#).

4.16.2.4 `virtual const std::size_t GraphInterface::getTransitionCount (const StateID ID) const [inline], [pure virtual]`

Obtains number of transitions for given state.

Parameters

<i>ID</i>	ID of the state to get the number from
-----------	--

Returns

integer with number of outgoing transitions

Implemented in [AutomatonStructure](#), [ProductStructure](#), [ParametrizedStructure](#), and [BasicStructure](#).

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

- `reforging/graph_interface.hpp`

4.17 Model::Interaction Struct Reference

Public Member Functions

- **Interaction** (const StateID _source, const std::size_t _threshold, const EdgeConstrain _constrain, const bool _observable)

Public Attributes

- StateID **source**
- std::size_t **threshold**
- EdgeConstrain **constrain**
- bool **observable**

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

- `parsing/model.hpp`

4.18 rapidxml::memory_pool< Ch > Class Template Reference

```
#include <rapidxml.hpp>
```

Inherited by [rapidxml::xml_document< Ch >](#).

Classes

- struct **header**

Public Member Functions

- [memory_pool](#) ()
Constructs empty pool with default allocator functions.
- [~memory_pool](#) ()
- [xml_node< Ch > * allocate_node](#) (node_type type, const Ch *name=0, const Ch *value=0, std::size_t name_size=0, std::size_t value_size=0)
- [xml_attribute< Ch > * allocate_attribute](#) (const Ch *name=0, const Ch *value=0, std::size_t name_size=0, std::size_t value_size=0)
- Ch * [allocate_string](#) (const Ch *source=0, std::size_t size=0)
- [xml_node< Ch > * clone_node](#) (const [xml_node< Ch >](#) *source, [xml_node< Ch >](#) *result=0)
- void [clear](#) ()
- void [set_allocator](#) (alloc_func *af, free_func *ff)

4.18.1 Detailed Description

```
template<class Ch = char>class rapidxml::memory_pool< Ch >
```

This class is used by the parser to create new nodes and attributes, without overheads of dynamic memory allocation. In most cases, you will not need to use this class directly. However, if you need to create nodes manually or modify names/values of nodes, you are encouraged to use [memory_pool](#) of relevant [xml_document](#) to allocate the memory. Not only is this faster than allocating them by using `new` operator, but also their lifetime will be tied to the lifetime of document, possibly simplifying memory management.

Call [allocate_node\(\)](#) or [allocate_attribute\(\)](#) functions to obtain new nodes or attributes from the pool. You can also call [allocate_string\(\)](#) function to allocate strings. Such strings can then be used as names or values of nodes without worrying about their lifetime. Note that there is no `free()` function – all allocations are freed at once when [clear\(\)](#) function is called, or when the pool is destroyed.

It is also possible to create a standalone [memory_pool](#), and use it to allocate nodes, whose lifetime will not be tied to any document.

Pool maintains `RAPIDXML_STATIC_POOL_SIZE` bytes of statically allocated memory. Until static memory is exhausted, no dynamic memory allocations are done. When static memory is exhausted, pool allocates additional blocks of memory of size `RAPIDXML_DYNAMIC_POOL_SIZE` each, by using global `new[]` and `delete[]` operators. This behaviour can be changed by setting custom allocation routines. Use [set_allocator\(\)](#) function to set them.

Allocations for nodes, attributes and strings are aligned at `RAPIDXML_ALIGNMENT` bytes. This value defaults to the size of pointer on target architecture.

To obtain absolutely top performance from the parser, it is important that all nodes are allocated from a single, contiguous block of memory. Otherwise, cache misses when jumping between two (or more) disjoint blocks of

memory can slow down parsing quite considerably. If required, you can tweak `RAPIDXML_STATIC_POOL_SIZE`, `RAPIDXML_DYNAMIC_POOL_SIZE` and `RAPIDXML_ALIGNMENT` to obtain best wasted memory to performance compromise. To do it, define their values before `rapidxml.hpp` file is included.

Parameters

<i>Ch</i>	Character type of created nodes.
-----------	----------------------------------

4.18.2 Constructor & Destructor Documentation

4.18.2.1 `template<class Ch = char> rapidxml::memory_pool< Ch >::~~memory_pool() [inline]`

Destroys pool and frees all the memory. This causes memory occupied by nodes allocated by the pool to be freed. Nodes allocated from the pool are no longer valid.

4.18.3 Member Function Documentation

4.18.3.1 `template<class Ch = char> xml_attribute<Ch>* rapidxml::memory_pool< Ch >::allocate_attribute(const Ch * name = 0, const Ch * value = 0, std::size_t name_size = 0, std::size_t value_size = 0) [inline]`

Allocates a new attribute from the pool, and optionally assigns name and value to it. If the allocation request cannot be accomodated, this function will throw `std::bad_alloc`. If exceptions are disabled by defining `RAPIDXML_NO_EXCEPTIONS`, this function will call `rapidxml::parse_error_handler()` function.

Parameters

<i>name</i>	Name to assign to the attribute, or 0 to assign no name.
<i>value</i>	Value to assign to the attribute, or 0 to assign no value.
<i>name_size</i>	Size of name to assign, or 0 to automatically calculate size from name string.
<i>value_size</i>	Size of value to assign, or 0 to automatically calculate size from value string.

Returns

Pointer to allocated attribute. This pointer will never be NULL.

4.18.3.2 `template<class Ch = char> xml_node<Ch>* rapidxml::memory_pool< Ch >::allocate_node(node_type type, const Ch * name = 0, const Ch * value = 0, std::size_t name_size = 0, std::size_t value_size = 0) [inline]`

Allocates a new node from the pool, and optionally assigns name and value to it. If the allocation request cannot be accomodated, this function will throw `std::bad_alloc`. If exceptions are disabled by defining `RAPIDXML_NO_EXCEPTIONS`, this function will call `rapidxml::parse_error_handler()` function.

Parameters

<i>type</i>	Type of node to create.
<i>name</i>	Name to assign to the node, or 0 to assign no name.
<i>value</i>	Value to assign to the node, or 0 to assign no value.
<i>name_size</i>	Size of name to assign, or 0 to automatically calculate size from name string.
<i>value_size</i>	Size of value to assign, or 0 to automatically calculate size from value string.

Returns

Pointer to allocated node. This pointer will never be NULL.

4.18.3.3 `template<class Ch = char> Ch* rapidxml::memory_pool< Ch >::allocate_string (const Ch * source = 0, std::size_t size = 0) [inline]`

Allocates a char array of given size from the pool, and optionally copies a given string to it. If the allocation request cannot be accomodated, this function will throw `std::bad_alloc`. If exceptions are disabled by defining `RAPIDXML_NO_EXCEPTIONS`, this function will call `rapidxml::parse_error_handler()` function.

Parameters

<i>source</i>	String to initialize the allocated memory with, or 0 to not initialize it.
<i>size</i>	Number of characters to allocate, or zero to calculate it automatically from source string length; if size is 0, source string must be specified and null terminated.

Returns

Pointer to allocated char array. This pointer will never be NULL.

4.18.3.4 `template<class Ch = char> void rapidxml::memory_pool< Ch >::clear () [inline]`

Clears the pool. This causes memory occupied by nodes allocated by the pool to be freed. Any nodes or strings allocated from the pool will no longer be valid.

Reimplemented in [rapidxml::xml_document< Ch >](#).

4.18.3.5 `template<class Ch = char> xml_node<Ch>* rapidxml::memory_pool< Ch >::clone_node (const xml_node< Ch > * source, xml_node< Ch > * result = 0) [inline]`

Clones an [xml_node](#) and its hierarchy of child nodes and attributes. Nodes and attributes are allocated from this memory pool. Names and values are not cloned, they are shared between the clone and the source. Result node can be optionally specified as a second parameter, in which case its contents will be replaced with cloned source node. This is useful when you want to clone entire document.

Parameters

<i>source</i>	Node to clone.
<i>result</i>	Node to put results in, or 0 to automatically allocate result node

Returns

Pointer to cloned node. This pointer will never be NULL.

4.18.3.6 `template<class Ch = char> void rapidxml::memory_pool< Ch >::set_allocator (alloc_func * af, free_func * ff) [inline]`

Sets or resets the user-defined memory allocation functions for the pool. This can only be called when no memory is allocated from the pool yet, otherwise results are undefined. Allocation function must not return invalid pointer on failure. It should either throw, stop the program, or use `longjmp()` function to pass control to other place of program. If it returns invalid pointer, results are undefined.

User defined allocation functions must have the following forms:

```
void *allocate(std::size_t size);
void free(void *pointer);
```


Parameters

<i>af</i>	Allocation function, or 0 to restore default function
<i>ff</i>	Free function, or 0 to restore default function

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

- parsing/rapidxml-1.13/[rapidxml.hpp](#)

4.19 Model Class Reference

Classes

- struct **AdditionalInformation**
- struct **BuchiAutomatonState**
- struct [Interaction](#)
- struct **ModelSpecie**

Public Types

- typedef std::pair< std::vector< bool >, int > **Regulation**
- typedef std::pair< StateID, std::string > **Egde**

Public Member Functions

- const std::size_t [getSpeciesCount](#) () const
- const std::size_t [getStateCount](#) () const
- const int [findID](#) (const std::string name) const
- const std::string & [getName](#) (const std::size_t ID) const
- const std::size_t [getMin](#) (const std::size_t ID) const
- const std::size_t [getMax](#) (const std::size_t ID) const
- const std::vector< [Interaction](#) > & [getInteractions](#) (const std::size_t ID) const
- const std::vector< Regulation > & [getRegulations](#) (const std::size_t ID) const
- const bool [isFinal](#) (const std::size_t ID) const
- const std::vector< Egde > & [getEdges](#) (const std::size_t ID) const

Friends

- class **ModelParser**

4.19.1 Member Function Documentation

4.19.1.1 const int Model::findID (const std::string *name*) const [inline]

Returns

ID of the specie with the specified name if there is such, otherwise -1

4.19.1.2 `const std::vector<Edge>& Model::getEdges (const std::size_t ID) const` `[inline]`

Returns

edges of the state

4.19.1.3 `const std::vector<Interaction>& Model::getInteractions (const std::size_t ID) const` `[inline]`

Returns

interactions of the specie

4.19.1.4 `const std::size_t Model::getMax (const std::size_t ID) const` `[inline]`

Returns

maximal value of the specie

4.19.1.5 `const std::size_t Model::getMin (const std::size_t ID) const` `[inline]`

Returns

minimal value of the specie (always 0)

4.19.1.6 `const std::string& Model::getName (const std::size_t ID) const` `[inline]`

Returns

name of the specie

4.19.1.7 `const std::vector<Regulation>& Model::getRegulations (const std::size_t ID) const` `[inline]`

Returns

regulations of the specie

4.19.1.8 `const std::size_t Model::getSpeciesCount () const` `[inline]`

Returns

number of the species

4.19.1.9 `const std::size_t Model::getStateCount () const` `[inline]`

Returns

number of the states

4.19.1.10 `const bool Model::isFinal (const std::size_t ID) const` `[inline]`

Returns

true if the state is final

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

- parsing/model.hpp

4.20 ModelChecker Class Reference

Public Member Functions

- [ModelChecker](#) (const [ProductStructure](#) &_product, [ColorStorage](#) &_storage)
- `const std::vector< std::size_t > startColoring (const StateID ID, const Parameters parameters, const Range &_range, const WitnessUse _witness_use=none_wit)`
- `const std::vector< std::size_t > startColoring (const Parameters parameters, const std::set< StateID > &_updates, const Range &_range, const WitnessUse _witness_use=none_wit)`

4.20.1 Constructor & Destructor Documentation

4.20.1.1 `ModelChecker::ModelChecker (const ProductStructure &_product, ColorStorage &_storage)` `[inline]`

Constructor, passes the data

4.20.2 Member Function Documentation

4.20.2.1 `const std::vector<std::size_t> ModelChecker::startColoring (const StateID ID, const Parameters parameters, const Range &_range, const WitnessUse _witness_use = none_wit)` `[inline]`

Start a new coloring round for cycle detection from a single state.

Parameters

<i>ID</i>	ID of the state to start cycle detection from
<i>parameters</i>	starting parameters for the cycle detection
<i>_range</i>	range of parameters for this coloring round
<i>_witness_use</i>	how to manage witnesses in this coloring round

4.20.2.2 `const std::vector<std::size_t> ModelChecker::startColoring (const Parameters parameters, const std::set< StateID > &_updates, const Range &_range, const WitnessUse _witness_use = none_wit)` `[inline]`

Start a new coloring round for cycle detection from a single state.

Parameters

<i>_updates</i>	states that are will be scheduled for an update in this round
<i>_range</i>	range of parameters for this coloring round
<i>_witness_use</i>	how to manage witnesses in this coloring round

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

- coloring/model_checker.hpp

4.21 ModelParser Class Reference

Public Member Functions

- [ModelParser](#) ([Model](#) &_model)
- void [parseInput](#) ()

4.21.1 Constructor & Destructor Documentation

4.21.1.1 ModelParser::ModelParser ([Model](#) & *_model*) `[inline]`

Constructor has to provide references to an input stream to read from and model object to store parsed information.

4.21.2 Member Function Documentation

4.21.2.1 void ModelParser::parseInput () `[inline]`

Functions that causes the parser to read the input from the stream, parse it and store model information in the model object.

Returns

version of the parsed file.

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

- parsing/model_parser.hpp

4.22 rapidxml::node_iterator< Ch > Class Template Reference

Iterator of child nodes of [xml_node](#).

```
#include <rapidxml_iterators.hpp>
```

Public Types

- typedef [rapidxml::xml_node](#)< Ch > **value_type**
- typedef [rapidxml::xml_node](#)< Ch > & **reference**
- typedef [rapidxml::xml_node](#)< Ch > * **pointer**
- typedef std::ptrdiff_t **difference_type**
- typedef
std::bidirectional_iterator_tag **iterator_category**

Public Member Functions

- **node_iterator** ([xml_node](#)< Ch > *node)
- **reference operator*** () const
- **pointer operator->** () const
- **node_iterator** & **operator++** ()
- **node_iterator** **operator++** (int)
- **node_iterator** & **operator--** ()
- **node_iterator** **operator--** (int)
- bool **operator==** (const [node_iterator](#)< Ch > &rhs)
- bool **operator!=** (const [node_iterator](#)< Ch > &rhs)

4.22.1 Detailed Description

template<class Ch>class rapidxml::node_iterator< Ch >

Iterator of child nodes of [xml_node](#).

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

- parsing/rapidxml-1.13/rapidxml_iterators.hpp

4.23 OutputManager Class Reference

Public Member Functions

- **OutputManager** (const [ColoringAnalyzer](#) &_analyzer, const [ProductStructure](#) &_product, const [SplitManager](#) &_split_manager, [WitnessSearcher](#) &_searcher)
- void [outputSummary](#) (const std::size_t total_count)
- void [outputRoundNum](#) ()
- void [outputRound](#) (const std::vector< std::size_t > &BFS_reach) const

4.23.1 Member Function Documentation

4.23.1.1 void OutputManager::outputRound (const std::vector< std::size_t > &BFS_reach) const [inline]

Display colors synthetized during current round

4.23.1.2 void OutputManager::outputRoundNum () [inline]

Outputs round number - if there are no data within, then erase the line each round

4.23.1.3 void OutputManager::outputSummary (const std::size_t total_count) [inline]

Output summary after the computation

Parameters

<i>total_count</i>	number of all feasible colors
--------------------	-------------------------------

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

- results/output_manager.hpp

4.24 OutputStreamer Class Reference

Public Types

- typedef const unsigned int **Trait**

Public Member Functions

- bool [testTrait](#) (const unsigned int tested, const unsigned int traits) const

- const bool `isResultInFile` () const
- `OutputStreamer` ()
- `~OutputStreamer` ()
- void `createStreamFile` (StreamType stream_type, std::string filename)
- void `flush` ()
- template<class outputType >
const `OutputStreamer` & `output` (StreamType stream_type, const outputType &stream_data, const unsigned int trait_mask=0)
- template<class outputType >
const `OutputStreamer` & `output` (const outputType &stream_data, const unsigned int trait_mask=0) const

Static Public Attributes

- static Trait `no_newl` = 1
- static Trait `important` = 2
- static Trait `rewrite_In` = 4

4.24.1 Constructor & Destructor Documentation

4.24.1.1 `OutputStreamer::OutputStreamer ()` [inline]

Basic constructor - should be used only for the single object shared throught the program

4.24.1.2 `OutputStreamer::~~OutputStreamer ()` [inline]

If some of the streams has been assigned a file, delete that file object

4.24.2 Member Function Documentation

4.24.2.1 void `OutputStreamer::createStreamFile` (StreamType *stream_type*, std::string *filename*)

 [inline]

output on a specified stream

Parameters

<i>stream_type</i>	enumeration type specifying the type of stream to output to
<i>data</i>	data to output - should be any possible ostream data

4.24.2.2 const bool `OutputStreamer::isResultInFile` () const

 [inline]

Returns

true if there is a file to output the results

4.24.2.3 template<class outputType > const `OutputStreamer& OutputStreamer::output` (StreamType *stream_type*, const outputType & *stream_data*, const unsigned int *trait_mask* = 0)

 [inline]

output on a specified stream

Parameters

<i>stream_type</i>	enumeration type specifying the type of stream to output to
<i>data</i>	data to output - should be any possible ostream data
<i>trait_mask</i>	bitmask of traits for output

4.24.2.4 `template<class outputType > const OutputStreamer& OutputStreamer::output (const outputType & stream_data, const unsigned int trait_mask = 0) const` `[inline]`

overloaded method that uses the same stream as the last output

Parameters

<i>data</i>	data to output - should be any possible ostream data
<i>trait_mask</i>	bitmask of traits for output

4.24.2.5 `bool OutputStreamer::testTrait (const unsigned int tested, const unsigned int traits) const` `[inline]`

test if given trait is present

Parameters

<i>tested</i>	number of the tested trait
<i>traits</i>	traits given with the function

Returns

bool if the trait is present

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

- auxiliary/output_streamer.hpp

4.25 ParametrizedStructure Class Reference

Inherits [GraphInterface](#).

Classes

- struct **State**
- struct **Transition**

Public Member Functions

- const std::size_t [getStateCount](#) () const
- const std::size_t [getTransitionCount](#) (const StateID ID) const
- const std::size_t [getTargetID](#) (const StateID ID, const std::size_t transtion_num) const
- const std::string [getString](#) (const StateID ID) const
- const Levels & [getStateLevels](#) (const StateID ID) const
- const std::size_t [getStepSize](#) (const StateID ID, const std::size_t transtion_num) const
- const std::vector< bool > & [getTransitive](#) (const StateID ID, const std::size_t transtion_num) const

Friends

- class **ParametrizedStructureBuilder**

4.25.1 Member Function Documentation

4.25.1.1 `const std::size_t ParametrizedStructure::getStateCount () const` `[inline], [virtual]`

Implements [GraphInterface](#).

4.25.1.2 `const Levels& ParametrizedStructure::getStateLevels (const StateID ID) const` `[inline]`

Parameters

<i>ID</i>	ID of the state to get the data from
-----------	--------------------------------------

Returns

species level

4.25.1.3 `const std::size_t ParametrizedStructure::getStepSize (const StateID ID, const std::size_t transition_num) const` `[inline]`

Parameters

<i>ID</i>	ID of the state to get the data from
<i>transition_num</i>	index of the transition to get the data from

Returns

number of neighbour parameters that share the same value of the function

4.25.1.4 `const std::string ParametrizedStructure::getString (const StateID ID) const` `[inline], [virtual]`

Return string representing given state in the form (specie1_val, specie2_val, ...)

Implements [GraphInterface](#).

4.25.1.5 `const std::size_t ParametrizedStructure::getTargetID (const StateID ID, const std::size_t transition_num) const` `[inline], [virtual]`

Implements [GraphInterface](#).

4.25.1.6 `const std::size_t ParametrizedStructure::getTransitionCount (const StateID ID) const` `[inline], [virtual]`

Implements [GraphInterface](#).

4.25.1.7 `const std::vector<bool>& ParametrizedStructure::getTransitive (const StateID ID, const std::size_t transition_num) const` `[inline]`

Parameters

<i>ID</i>	ID of the state to get the data from
<i>transition_num</i>	index of the transition to get the data from

Returns

target values that are include in non-transitive parameters that have to be removed

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

- `reforging/parametrized_structure.hpp`

4.26 ParametrizedStructureBuilder Class Reference

Public Member Functions

- `ParametrizedStructureBuilder` (const `BasicStructure` &_basic_structure, const `FunctionsStructure` &_regulatory_functions, `ParametrizedStructure` &_structure)
- void `buildStructure` ()

4.26.1 Constructor & Destructor Documentation

4.26.1.1 `ParametrizedStructureBuilder::ParametrizedStructureBuilder (const BasicStructure &_basic_structure, const FunctionsStructure &_regulatory_functions, ParametrizedStructure &_structure) [inline]`

Constructor just attaches the references to data holders

4.26.2 Member Function Documentation

4.26.2.1 `void ParametrizedStructureBuilder::buildStructure () [inline]`

Create the states from the model and fill the structure with them.

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

- `reforging/parametrized_structure_builder.hpp`

4.27 rapidxml::parse_error Class Reference

```
#include <rapidxml.hpp>
```

Public Member Functions

- `parse_error` (const char **what*, void **where*)
Constructs parse error.
- virtual const char * *what* () const throw ()
- template<class Ch >
Ch * *where* () const

4.27.1 Detailed Description

Parse error exception. This exception is thrown by the parser when an error occurs. Use `what()` function to get human-readable error message. Use `where()` function to get a pointer to position within source text where error was detected.

If throwing exceptions by the parser is undesirable, it can be disabled by defining `RAPIDXML_NO_EXCEPTIONS` macro before `rapidxml.hpp` is included. This will cause the parser to call `rapidxml::parse_error_handler()` function instead of throwing an exception. This function must be defined by the user.

This class derives from `std::exception` class.

4.27.2 Member Function Documentation

4.27.2.1 `virtual const char* rapidxml::parse_error::what () const throw ()` `[inline],[virtual]`

Gets human readable description of error.

Returns

Pointer to null terminated description of the error.

4.27.2.2 `template<class Ch> Ch* rapidxml::parse_error::where () const` `[inline]`

Gets pointer to character data where error happened. Ch should be the same as char type of `xml_document` that produced the error.

Returns

Pointer to location within the parsed string where error occurred.

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

- parsing/rapidxml-1.13/[rapidxml.hpp](#)

4.28 PerColorStorage Class Reference

Classes

- struct **ColorData**
- struct **State**

Public Member Functions

- **PerColorStorage** (const [ColoringAnalyzer](#) &_analyzer, const [ColorStorage](#) &_storage, const [ProductStructure](#) &_product)
- const Neighbours & [getNeighbours](#) (const StateID ID, const bool successors, std::size_t number) const

4.28.1 Member Function Documentation

4.28.1.1 `const Neighbours& PerColorStorage::getNeighbours (const StateID ID, const bool successors, std::size_t number)` `const` `[inline]`

Get all the neighbours for this color from this state.

Parameters

<i>ID</i>	index of the state to ask for predecessors
<i>successors</i>	true if successors are required, false if predecessors
<i>number</i>	ordinal number of stored coloring

Returns

neighbours for given state

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

- results/per_color_storage.hpp

4.29 ProductBuilder Class Reference

Public Member Functions

- [ProductBuilder](#) (const [ParametrizedStructure](#) &_structure, const [AutomatonStructure](#) &_automaton, [ProductStructure](#) &_product, [ColorStorage](#) &_storage)
- void [buildProduct](#) ()

4.29.1 Constructor & Destructor Documentation

4.29.1.1 [ProductBuilder::ProductBuilder](#) (const [ParametrizedStructure](#) &_structure, const [AutomatonStructure](#) &_automaton, [ProductStructure](#) &_product, [ColorStorage](#) &_storage) `[inline]`

Constructor just attaches the references to data holders

4.29.2 Member Function Documentation

4.29.2.1 void [ProductBuilder::buildProduct](#) () `[inline]`

Create the the product from BA and KS together.

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

- reforging/product_builder.hpp

4.30 ProductStructure Class Reference

Inherits [AutomatonInterface](#).

Classes

- struct **State**
- struct **Transition**

Public Member Functions

- **ProductStructure** (const [FunctionsStructure](#) &_functions, const [ConstrainsParser](#) &_constrains, const [ParametrizedStructure](#) &_structure, const [AutomatonStructure](#) &_automaton)
- const std::size_t [getStateCount](#) () const
- const std::size_t [getTransitionCount](#) (const StateID ID) const
- const std::size_t [getTargetID](#) (const StateID ID, const std::size_t trans_number) const
- const std::string [getString](#) (const StateID ID) const
- virtual const bool [isFinal](#) (const StateID ID) const

- virtual const bool [isInitial](#) (const StateID ID) const
- virtual const std::vector< StateID > & [getFinalStates](#) () const
- virtual const std::vector< StateID > & [getInitialStates](#) () const
- const StateID [getProductID](#) (const StateID KS_ID, const StateID BA_ID) const
- const StateID [getBAID](#) (const StateID ID) const
- const StateID [getKSID](#) (const StateID ID) const
- const [ParametrizedStructure](#) & [getKS](#) () const
- const [AutomatonStructure](#) & [getBA](#) () const
- const [FunctionsStructure](#) & [getFunc](#) () const
- const [ConstrainsParser](#) & [getCons](#) () const
- const Levels & [getStateLevels](#) (const StateID ID) const
- const std::size_t [getStepSize](#) (const StateID ID, const std::size_t transtion_num) const
- const std::vector< bool > & [getTransitive](#) (const StateID ID, const std::size_t transtion_num) const

Friends

- class **ProductBuilder**

4.30.1 Member Function Documentation

4.30.1.1 const [AutomatonStructure](#)& [ProductStructure::getBA](#) () const [\[inline\]](#)

Returns

constant reference to Buchi automaton stored within the product

4.30.1.2 const StateID [ProductStructure::getBAID](#) (const StateID *ID*) const [\[inline\]](#)

Returns

index of BA state form the product

4.30.1.3 const [ConstrainsParser](#)& [ProductStructure::getCons](#) () const [\[inline\]](#)

Returns

constant reference to structure with interactions constrains

4.30.1.4 virtual const std::vector<StateID>& [ProductStructure::getFinalStates](#) () const [\[inline\]](#),[\[virtual\]](#)

Implements [AutomatonInterface](#).

4.30.1.5 const [FunctionsStructure](#)& [ProductStructure::getFunc](#) () const [\[inline\]](#)

Returns

constant reference to structure with regulatory functions

4.30.1.6 `virtual const std::vector<StateID>& ProductStructure::getInitialStates () const` `[inline],[virtual]`

Implements [AutomatonInterface](#).

4.30.1.7 `const ParametrizedStructure& ProductStructure::getKS () const` `[inline]`

Returns

constant reference to Kripke structure stored within the product

4.30.1.8 `const StateID ProductStructure::getKSID (const StateID ID) const` `[inline]`

Returns

index of BA state form the product

4.30.1.9 `const StateID ProductStructure::getProductID (const StateID KS_ID, const StateID BA_ID) const` `[inline]`

Returns

index of this combination of states in the product

4.30.1.10 `const std::size_t ProductStructure::getStateCount () const` `[inline],[virtual]`

Implements [GraphInterface](#).

4.30.1.11 `const Levels& ProductStructure::getStateLevels (const StateID ID) const` `[inline]`

Parameters

<i>ID</i>	ID of the state to get the data from
-----------	--------------------------------------

Returns

species level

4.30.1.12 `const std::size_t ProductStructure::getStepSize (const StateID ID, const std::size_t transtion_num) const` `[inline]`

Parameters

<i>ID</i>	ID of the state to get the data from
<i>transtion_num</i>	index of the transition to get the data from

Returns

number of neighbour parameters that share the same value of the function

4.30.1.13 `const std::string ProductStructure::getString (const StateID ID) const` `[inline],[virtual]`

Create string in the form KSstateBAstate or KSstate based on if user requests BA as well

Implements [GraphInterface](#).

4.30.1.14 `const std::size_t ProductStructure::getTargetID (const StateID ID, const std::size_t trans_number) const` `[inline], [virtual]`

Implements [GraphInterface](#).

4.30.1.15 `const std::size_t ProductStructure::getTransitionCount (const StateID ID) const` `[inline], [virtual]`

Implements [GraphInterface](#).

4.30.1.16 `const std::vector<bool>& ProductStructure::getTransitive (const StateID ID, const std::size_t transition_num) const` `[inline]`

Parameters

<i>ID</i>	ID of the state to get the data from
<i>transition_num</i>	index of the transition to get the data from

Returns

target values that are include in non-transitive parameters that have to be removed

4.30.1.17 `virtual const bool ProductStructure::isFinal (const StateID ID) const` `[inline], [virtual]`

Implements [AutomatonInterface](#).

4.30.1.18 `virtual const bool ProductStructure::isInitial (const StateID ID) const` `[inline], [virtual]`

Implements [AutomatonInterface](#).

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

- `reforging/product_structure.hpp`

4.31 SplitManager Class Reference

Public Member Functions

- [SplitManager](#) (ColorNum `_all_colors_count`)
- void [setStartPositions](#) ()
- void [increaseRound](#) ()
- const ColorNum [getAllColorsCount](#) () const
- const Range [getRoundRange](#) () const
- const ColorNum [getRoundSize](#) () const
- const ColorNum [getProcColorsCount](#) () const
- const bool [lastRound](#) () const
- const bool [valid](#) () const
- const long long [getRoundNum](#) () const
- const long long [getRoundCount](#) () const
- Parameters [createStartingParameters](#) () const

4.31.1 Constructor & Destructor Documentation

4.31.1.1 SplitManager::SplitManager (ColorNum *_all_colors_count*) [inline]

Computes splitting for both process (in case of a distributed computation) and its rounds that are of a size of the Parameters data type.

Parameters

<i>_processes_ - count</i>	how many processes compute the coloring
<i>_process_ - number</i>	index of this process
<i>_parameters_ - count</i>	complete number of parameters that have to be tested by all the processes

4.31.2 Member Function Documentation

4.31.2.1 Parameters SplitManager::createStartingParameters () const [inline]

Returns

All the parameters of the current round.

4.31.2.2 const ColorNum SplitManager::getAllColorsCount () const [inline]

Returns

total number of parameters for all the processes

4.31.2.3 const ColorNum SplitManager::getProcColorsCount () const [inline]

Returns

range with first and one before last parameter to compute for this process

4.31.2.4 const long long SplitManager::getRoundCount () const [inline]

Returns

total number of rounds

4.31.2.5 const long long SplitManager::getRoundNum () const [inline]

Returns

number of this round

4.31.2.6 const Range SplitManager::getRoundRange () const [inline]

Returns

range with first and one before last parameter to compute this round

4.31.2.7 `const ColorNum SplitManager::getRoundSize () const` `[inline]`

Returns

number of bits in current round

4.31.2.8 `void SplitManager::increaseRound ()` `[inline]`

Increase parameter positions so a new round can be computed.

4.31.2.9 `const bool SplitManager::lastRound () const` `[inline]`

Returns

true if this round is not the last

4.31.2.10 `void SplitManager::setStartPositions ()` `[inline]`

Set values for the first round of computation.

4.31.2.11 `const bool SplitManager::valid () const` `[inline]`

Returns

true if current round is valid (this round does not correspond to any paramteres)

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

- coloring/split_manager.hpp

4.32 SynthesisManager Class Reference

Public Member Functions

- [SynthesisManager](#) (const [ProductStructure](#) &_product, [ColorStorage](#) &_storage)
- void [doSynthesis](#) ()

4.32.1 Constructor & Destructor Documentation

4.32.1.1 `SynthesisManager::SynthesisManager (const ProductStructure & _product, ColorStorage & _storage)` `[inline]`

Constructor builds all the data objects that are used within

4.32.2 Member Function Documentation

4.32.2.1 `void SynthesisManager::doSynthesis ()` `[inline]`

Main synthesis function that iterates through all the rounds of the synthesis

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

- coloring/synthesis_manager.hpp

4.33 TimeManager Class Reference

Public Member Functions

- void [startClock](#) (const std::string clock_name)
- void [ouputClock](#) (const std::string clock_name) const

4.33.1 Member Function Documentation

4.33.1.1 void TimeManager::ouputClock (const std::string *clock_name*) const `[inline]`

Outputs current runtime of the clock

Parameters

<i>clock_name</i>	name of the clock to output (also appears on the output)
-------------------	--

4.33.1.2 void TimeManager::startClock (const std::string *clock_name*) `[inline]`

Starts a clock with given name and, if it is requested by user, outputs the info.

Parameters

<i>clock_name</i>	unique ID of the clock that will also be send on the output
-------------------	---

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

- auxiliary/time_manager.hpp

4.34 UserOptions Class Reference

Public Member Functions

- [UserOptions](#) ()
- const bool [coloring](#) () const
- const WitnessUse [witnesses](#) () const
- const bool [BA](#) () const
- const bool [verbose](#) () const
- const bool [stats](#) () const
- const bool [negation](#) () const
- const bool [timeSerie](#) () const
- const std::size_t [procNum](#) () const
- const std::size_t [procCount](#) () const
- const bool [robustness](#) () const
- const bool [displayWintess](#) () const

Friends

- class **ArgumentParser**

4.34.1 Constructor & Destructor Documentation

4.34.1.1 UserOptions::UserOptions () [inline]

Constructor, sets up default values

4.34.2 Member Function Documentation

4.34.2.1 const bool UserOptions::BA () const [inline]

Returns

true if add_BA_to_witness is set (displaying path with BA states as well)

4.34.2.2 const bool UserOptions::coloring () const [inline]

Returns

true if show_coloring is set (displaying each accepting color)

4.34.2.3 const bool UserOptions::displayWintess () const [inline]

Returns

true if display_wintess (witness output is requested)

4.34.2.4 const bool UserOptions::negation () const [inline]

Returns

true if time_serie (checking only reachability)

4.34.2.5 const std::size_t UserOptions::procCount () const [inline]

Returns

total number of processes in distributed computation

4.34.2.6 const std::size_t UserOptions::procNum () const [inline]

Returns

number of this process in distributed computation (indexed from 1)

4.34.2.7 const bool UserOptions::robustness () const [inline]

Returns

true if compute_robustness (robustness output is requested)

4.34.2.8 `const bool UserOptions::stats () const [inline]`

Returns

true if `display_stats` is set (displaying statistics of the model)

4.34.2.9 `const bool UserOptions::timeSerie () const [inline]`

Returns

true if `negative_check` (switching feasible for non-feasible)

4.34.2.10 `const bool UserOptions::verbose () const [inline]`

Returns

true if `verbose` is set (displaying additional information during computation)

4.34.2.11 `const WitnessUse UserOptions::witnesses () const [inline]`

Returns

how to manage witnesses

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

- `auxiliary/user_options.hpp`

4.35 WitnessSearcher Class Reference

Public Member Functions

- [WitnessSearcher](#) (const [ColoringAnalyzer](#) &*_analyzer*, const [ColorStorage](#) &*_storage*, const [ProductStructure](#) &*_product*)
- void [display](#) (const std::vector< std::size_t > &*BFS_reach*)

4.35.1 Constructor & Destructor Documentation

4.35.1.1 `WitnessSearcher::WitnessSearcher (const ColoringAnalyzer & _analyzer, const ColorStorage & _storage, const ProductStructure & _product) [inline]`

Get reference data and create final states that will hold all the computed data

4.35.2 Member Function Documentation

4.35.2.1 `void WitnessSearcher::display (const std::vector< std::size_t > & BFS_reach) [inline]`

Output all witnesses for all colors, might be together with the colors as well.

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

- `results/witness_searcher.hpp`

4.36 rapidxml::xml_attribute< Ch > Class Template Reference

#include <rapidxml.hpp>

Inherits [rapidxml::xml_base< Ch >](#).

Public Member Functions

- [xml_attribute](#) ()
- [xml_document](#)< Ch > * [document](#) () const
- [xml_attribute](#)< Ch > * [previous_attribute](#) (const Ch *[name](#)=0, std::size_t [name_size](#)=0, bool case_sensitive=true) const
- [xml_attribute](#)< Ch > * [next_attribute](#) (const Ch *[name](#)=0, std::size_t [name_size](#)=0, bool case_sensitive=true) const

Friends

- class [xml_node](#)< Ch >

Additional Inherited Members

4.36.1 Detailed Description

```
template<class Ch = char>class rapidxml::xml_attribute< Ch >
```

Class representing attribute node of XML document. Each attribute has name and value strings, which are available through [name\(\)](#) and [value\(\)](#) functions (inherited from [xml_base](#)). Note that after parse, both name and value of attribute will point to interior of source text used for parsing. Thus, this text must persist in memory for the lifetime of attribute.

Parameters

<i>Ch</i>	Character type to use.
-----------	------------------------

4.36.2 Constructor & Destructor Documentation

4.36.2.1 `template<class Ch = char> rapidxml::xml_attribute< Ch >::xml_attribute () [inline]`

Constructs an empty attribute with the specified type. Consider using [memory_pool](#) of appropriate [xml_document](#) if allocating attributes manually.

4.36.3 Member Function Documentation

4.36.3.1 `template<class Ch = char> xml_document<Ch>* rapidxml::xml_attribute< Ch >::document () const [inline]`

Gets document of which attribute is a child.

Returns

Pointer to document that contains this attribute, or 0 if there is no parent document.

4.36.3.2 `template<class Ch = char> xml_attribute<Ch>* rapidxml::xml_attribute< Ch >::next_attribute (const Ch * name = 0, std::size_t name_size = 0, bool case_sensitive = true) const [inline]`

Gets next attribute, optionally matching attribute name.

Parameters

<i>name</i>	Name of attribute to find, or 0 to return next attribute regardless of its name; this string doesn't have to be zero-terminated if name_size is non-zero
<i>name_size</i>	Size of name, in characters, or 0 to have size calculated automatically from string
<i>case_sensitive</i>	Should name comparison be case-sensitive; non case-sensitive comparison works properly only for ASCII characters

Returns

Pointer to found attribute, or 0 if not found.

4.36.3.3 `template<class Ch = char> xml_attribute<Ch>* rapidxml::xml_attribute< Ch >::previous_attribute (const Ch * name = 0, std::size_t name_size = 0, bool case_sensitive = true) const [inline]`

Gets previous attribute, optionally matching attribute name.

Parameters

<i>name</i>	Name of attribute to find, or 0 to return previous attribute regardless of its name; this string doesn't have to be zero-terminated if name_size is non-zero
<i>name_size</i>	Size of name, in characters, or 0 to have size calculated automatically from string
<i>case_sensitive</i>	Should name comparison be case-sensitive; non case-sensitive comparison works properly only for ASCII characters

Returns

Pointer to found attribute, or 0 if not found.

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

- [parsing/rapidxml-1.13/rapidxml.hpp](#)

4.37 rapidxml::xml_base< Ch > Class Template Reference

```
#include <rapidxml.hpp>
```

Inherited by [rapidxml::xml_attribute< Ch >](#), and [rapidxml::xml_node< Ch >](#).

Public Member Functions

- `Ch * name () const`
- `std::size_t name_size () const`
- `Ch * value () const`
- `std::size_t value_size () const`
- `void name (const Ch *name, std::size_t size)`
- `void name (const Ch *name)`
- `void value (const Ch *value, std::size_t size)`
- `void value (const Ch *value)`
- `xml_node< Ch > * parent () const`

Static Protected Member Functions

- static Ch * **nullstr** ()

Protected Attributes

- Ch * **m_name**
- Ch * **m_value**
- std::size_t **m_name_size**
- std::size_t **m_value_size**
- [xml_node](#)< Ch > * **m_parent**

4.37.1 Detailed Description

template<class Ch = char>class rapidxml::xml_base< Ch >

Base class for [xml_node](#) and [xml_attribute](#) implementing common functions: [name\(\)](#), [name_size\(\)](#), [value\(\)](#), [value_size\(\)](#) and [parent\(\)](#).

Parameters

<i>Ch</i>	Character type to use
-----------	-----------------------

4.37.2 Member Function Documentation

4.37.2.1 template<class Ch = char> Ch* rapidxml::xml_base< Ch >::name () const [inline]

Gets name of the node. Interpretation of name depends on type of node. Note that name will not be zero-terminated if rapidxml::parse_no_string_terminators option was selected during parse.

Use [name_size\(\)](#) function to determine length of the name.

Returns

Name of node, or empty string if node has no name.

4.37.2.2 template<class Ch = char> void rapidxml::xml_base< Ch >::name (const Ch * *name*, std::size_t *size*) [inline]

Sets name of node to a non zero-terminated string. See ownership_of_strings.

Note that node does not own its name or value, it only stores a pointer to it. It will not delete or otherwise free the pointer on destruction. It is responsibility of the user to properly manage lifetime of the string. The easiest way to achieve it is to use [memory_pool](#) of the document to allocate the string - on destruction of the document the string will be automatically freed.

Size of name must be specified separately, because name does not have to be zero terminated. Use [name\(const Ch *\)](#) function to have the length automatically calculated (string must be zero terminated).

Parameters

<i>name</i>	Name of node to set. Does not have to be zero terminated.
<i>size</i>	Size of name, in characters. This does not include zero terminator, if one is present.

4.37.2.3 `template<class Ch = char> void rapidxml::xml_base< Ch >::name (const Ch * name) [inline]`

Sets name of node to a zero-terminated string. See also `ownership_of_strings` and `xml_node::name(const Ch *, std::size_t)`.

Parameters

<i>name</i>	Name of node to set. Must be zero terminated.
-------------	---

4.37.2.4 `template<class Ch = char> std::size_t rapidxml::xml_base< Ch >::name_size () const [inline]`

Gets size of node name, not including terminator character. This function works correctly irrespective of whether name is or is not zero terminated.

Returns

Size of node name, in characters.

4.37.2.5 `template<class Ch = char> xml_node<Ch>* rapidxml::xml_base< Ch >::parent () const [inline]`

Gets node parent.

Returns

Pointer to parent node, or 0 if there is no parent.

4.37.2.6 `template<class Ch = char> Ch* rapidxml::xml_base< Ch >::value () const [inline]`

Gets value of node. Interpretation of value depends on type of node. Note that value will not be zero-terminated if `rapidxml::parse_no_string_terminators` option was selected during parse.

Use `value_size()` function to determine length of the value.

Returns

Value of node, or empty string if node has no value.

4.37.2.7 `template<class Ch = char> void rapidxml::xml_base< Ch >::value (const Ch * value, std::size_t size) [inline]`

Sets value of node to a non zero-terminated string. See `ownership_of_strings`.

Note that node does not own its name or value, it only stores a pointer to it. It will not delete or otherwise free the pointer on destruction. It is responsibility of the user to properly manage lifetime of the string. The easiest way to achieve it is to use `memory_pool` of the document to allocate the string - on destruction of the document the string will be automatically freed.

Size of value must be specified separately, because it does not have to be zero terminated. Use `value(const Ch *)` function to have the length automatically calculated (string must be zero terminated).

If an element has a child node of type `node_data`, it will take precedence over element value when printing. If you want to manipulate data of elements using values, use parser flag `rapidxml::parse_no_data_nodes` to prevent creation of data nodes by the parser.

Parameters

<i>value</i>	value of node to set. Does not have to be zero terminated.
<i>size</i>	Size of value, in characters. This does not include zero terminator, if one is present.

4.37.2.8 `template<class Ch = char> void rapidxml::xml_base< Ch >::value (const Ch * value) [inline]`

Sets value of node to a zero-terminated string. See also `ownership_of_strings` and `xml_node::value(const Ch *, std::size_t)`.

Parameters

<i>value</i>	Value of node to set. Must be zero terminated.
--------------	--

4.37.2.9 `template<class Ch = char> std::size_t rapidxml::xml_base< Ch >::value_size () const [inline]`

Gets size of node value, not including terminator character. This function works correctly irrespective of whether value is or is not zero terminated.

Returns

Size of node value, in characters.

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

- `parsing/rapidxml-1.13/rapidxml.hpp`

4.38 rapidxml::xml_document< Ch > Class Template Reference

`#include <rapidxml.hpp>`

Inherits `rapidxml::xml_node< Ch >`, and `rapidxml::memory_pool< Ch >`.

Classes

- struct `attribute_name_pred`
- struct `attribute_value_pred`
- struct `attribute_value_pure_pred`
- struct `node_name_pred`
- struct `text_pred`
- struct `text_pure_no_ws_pred`
- struct `text_pure_with_ws_pred`
- struct `whitespace_pred`

Public Member Functions

- `xml_document ()`
Constructs empty XML document.
- `template<int Flags> void parse (Ch *text)`
- `void clear ()`

Additional Inherited Members

4.38.1 Detailed Description


```
template<class Ch = char>class rapidxml::xml_document< Ch >
```

This class represents root of the DOM hierarchy. It is also an [xml_node](#) and a [memory_pool](#) through public inheritance. Use [parse\(\)](#) function to build a DOM tree from a zero-terminated XML text string. [parse\(\)](#) function allocates memory for nodes and attributes by using functions of [xml_document](#), which are inherited from [memory_pool](#). To access root node of the document, use the document itself, as if it was an [xml_node](#).

Parameters

<i>Ch</i>	Character type to use.
-----------	------------------------

4.38.2 Member Function Documentation

4.38.2.1 `template<class Ch = char> void rapidxml::xml_document< Ch >::clear () [inline]`

Clears the document by deleting all nodes and clearing the memory pool. All nodes owned by document pool are destroyed.

Reimplemented from [rapidxml::memory_pool< Ch >](#).

4.38.2.2 `template<class Ch = char> template<int Flags> void rapidxml::xml_document< Ch >::parse (Ch * text) [inline]`

Parses zero-terminated XML string according to given flags. Passed string will be modified by the parser, unless `rapidxml::parse_non_destructive` flag is used. The string must persist for the lifetime of the document. In case of error, [rapidxml::parse_error](#) exception will be thrown.

If you want to parse contents of a file, you must first load the file into the memory, and pass pointer to its beginning. Make sure that data is zero-terminated.

Document can be parsed into multiple times. Each new call to parse removes previous nodes and attributes (if any), but does not clear memory pool.

Parameters

<i>text</i>	XML data to parse; pointer is non-const to denote fact that this data may be modified by the parser.
-------------	--

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

- [parsing/rapidxml-1.13/rapidxml.hpp](#)

4.39 rapidxml::xml_node< Ch > Class Template Reference

```
#include <rapidxml.hpp>
```

Inherits [rapidxml::xml_base< Ch >](#).

Inherited by [rapidxml::xml_document< Ch >](#).

Public Member Functions

- [xml_node](#) (node_type type)
- node_type type () const
- [xml_document< Ch > * document](#) () const
- [xml_node< Ch > * first_node](#) (const Ch *name=0, std::size_t name_size=0, bool case_sensitive=true) const
- [xml_node< Ch > * last_node](#) (const Ch *name=0, std::size_t name_size=0, bool case_sensitive=true) const

- `xml_node< Ch > * previous_sibling` (const Ch *name=0, std::size_t name_size=0, bool case_sensitive=true) const
- `xml_node< Ch > * next_sibling` (const Ch *name=0, std::size_t name_size=0, bool case_sensitive=true) const
- `xml_attribute< Ch > * first_attribute` (const Ch *name=0, std::size_t name_size=0, bool case_sensitive=true) const
- `xml_attribute< Ch > * last_attribute` (const Ch *name=0, std::size_t name_size=0, bool case_sensitive=true) const
- void `type` (node_type type)
- void `prepend_node` (xml_node< Ch > *child)
- void `append_node` (xml_node< Ch > *child)
- void `insert_node` (xml_node< Ch > *where, xml_node< Ch > *child)
- void `remove_first_node` ()
- void `remove_last_node` ()
- void `remove_node` (xml_node< Ch > *where)
Removes specified child from the node.
- void `remove_all_nodes` ()
Removes all child nodes (but not attributes).
- void `prepend_attribute` (xml_attribute< Ch > *attribute)
- void `append_attribute` (xml_attribute< Ch > *attribute)
- void `insert_attribute` (xml_attribute< Ch > *where, xml_attribute< Ch > *attribute)
- void `remove_first_attribute` ()
- void `remove_last_attribute` ()
- void `remove_attribute` (xml_attribute< Ch > *where)
- void `remove_all_attributes` ()
Removes all attributes of node.

Additional Inherited Members

4.39.1 Detailed Description

template<class Ch = char>class rapidxml::xml_node< Ch >

Class representing a node of XML document. Each node may have associated name and value strings, which are available through `name()` and `value()` functions. Interpretation of name and value depends on type of the node. Type of node can be determined by using `type()` function.

Note that after parse, both name and value of node, if any, will point interior of source text used for parsing. Thus, this text must persist in the memory for the lifetime of node.

Parameters

<i>Ch</i>	Character type to use.
-----------	------------------------

4.39.2 Constructor & Destructor Documentation

4.39.2.1 template<class Ch = char> rapidxml::xml_node< Ch >::xml_node (node_type type) [inline]

Constructs an empty node with the specified type. Consider using `memory_pool` of appropriate document to allocate nodes manually.

Parameters

<i>type</i>	Type of node to construct.
-------------	----------------------------

4.39.3 Member Function Documentation

4.39.3.1 `template<class Ch = char> void rapidxml::xml_node< Ch >::append_attribute (xml_attribute< Ch > * attribute) [inline]`

Appends a new attribute to the node.

Parameters

<i>attribute</i>	Attribute to append.
------------------	----------------------

4.39.3.2 `template<class Ch = char> void rapidxml::xml_node< Ch >::append_node (xml_node< Ch > * child) [inline]`

Appends a new child node. The appended child becomes the last child.

Parameters

<i>child</i>	Node to append.
--------------	-----------------

4.39.3.3 `template<class Ch = char> xml_document<Ch>* rapidxml::xml_node< Ch >::document () const [inline]`

Gets document of which node is a child.

Returns

Pointer to document that contains this node, or 0 if there is no parent document.

4.39.3.4 `template<class Ch = char> xml_attribute<Ch>* rapidxml::xml_node< Ch >::first_attribute (const Ch * name = 0, std::size_t name_size = 0, bool case_sensitive = true) const [inline]`

Gets first attribute of node, optionally matching attribute name.

Parameters

<i>name</i>	Name of attribute to find, or 0 to return first attribute regardless of its name; this string doesn't have to be zero-terminated if name_size is non-zero
<i>name_size</i>	Size of name, in characters, or 0 to have size calculated automatically from string
<i>case_sensitive</i>	Should name comparison be case-sensitive; non case-sensitive comparison works properly only for ASCII characters

Returns

Pointer to found attribute, or 0 if not found.

4.39.3.5 `template<class Ch = char> xml_node<Ch>* rapidxml::xml_node< Ch >::first_node (const Ch * name = 0, std::size_t name_size = 0, bool case_sensitive = true) const [inline]`

Gets first child node, optionally matching node name.

Parameters

<i>name</i>	Name of child to find, or 0 to return first child regardless of its name; this string doesn't have to be zero-terminated if <i>name_size</i> is non-zero
<i>name_size</i>	Size of name, in characters, or 0 to have size calculated automatically from string
<i>case_sensitive</i>	Should name comparison be case-sensitive; non case-sensitive comparison works properly only for ASCII characters

Returns

Pointer to found child, or 0 if not found.

4.39.3.6 `template<class Ch = char> void rapidxml::xml_node< Ch >::insert_attribute (xml_attribute< Ch > * where, xml_attribute< Ch > * attribute) [inline]`

Inserts a new attribute at specified place inside the node. All attributes after and including the specified attribute are moved one position back.

Parameters

<i>where</i>	Place where to insert the attribute, or 0 to insert at the back.
<i>attribute</i>	Attribute to insert.

4.39.3.7 `template<class Ch = char> void rapidxml::xml_node< Ch >::insert_node (xml_node< Ch > * where, xml_node< Ch > * child) [inline]`

Inserts a new child node at specified place inside the node. All children after and including the specified node are moved one position back.

Parameters

<i>where</i>	Place where to insert the child, or 0 to insert at the back.
<i>child</i>	Node to insert.

4.39.3.8 `template<class Ch = char> xml_attribute<Ch>* rapidxml::xml_node< Ch >::last_attribute (const Ch * name = 0, std::size_t name_size = 0, bool case_sensitive = true) const [inline]`

Gets last attribute of node, optionally matching attribute name.

Parameters

<i>name</i>	Name of attribute to find, or 0 to return last attribute regardless of its name; this string doesn't have to be zero-terminated if <i>name_size</i> is non-zero
<i>name_size</i>	Size of name, in characters, or 0 to have size calculated automatically from string
<i>case_sensitive</i>	Should name comparison be case-sensitive; non case-sensitive comparison works properly only for ASCII characters

Returns

Pointer to found attribute, or 0 if not found.

4.39.3.9 `template<class Ch = char> xml_node<Ch>* rapidxml::xml_node< Ch >::last_node (const Ch * name = 0, std::size_t name_size = 0, bool case_sensitive = true) const [inline]`

Gets last child node, optionally matching node name. Behaviour is undefined if node has no children. Use [first_node\(\)](#) to test if node has children.

Parameters

<i>name</i>	Name of child to find, or 0 to return last child regardless of its name; this string doesn't have to be zero-terminated if name_size is non-zero
<i>name_size</i>	Size of name, in characters, or 0 to have size calculated automatically from string
<i>case_sensitive</i>	Should name comparison be case-sensitive; non case-sensitive comparison works properly only for ASCII characters

Returns

Pointer to found child, or 0 if not found.

4.39.3.10 `template<class Ch = char> xml_node<Ch>* rapidxml::xml_node< Ch >::next_sibling (const Ch * name = 0, std::size_t name_size = 0, bool case_sensitive = true) const [inline]`

Gets next sibling node, optionally matching node name. Behaviour is undefined if node has no parent. Use [parent\(\)](#) to test if node has a parent.

Parameters

<i>name</i>	Name of sibling to find, or 0 to return next sibling regardless of its name; this string doesn't have to be zero-terminated if name_size is non-zero
<i>name_size</i>	Size of name, in characters, or 0 to have size calculated automatically from string
<i>case_sensitive</i>	Should name comparison be case-sensitive; non case-sensitive comparison works properly only for ASCII characters

Returns

Pointer to found sibling, or 0 if not found.

4.39.3.11 `template<class Ch = char> void rapidxml::xml_node< Ch >::prepend_attribute (xml_attribute< Ch > * attribute) [inline]`

Prepends a new attribute to the node.

Parameters

<i>attribute</i>	Attribute to prepend.
------------------	-----------------------

4.39.3.12 `template<class Ch = char> void rapidxml::xml_node< Ch >::prepend_node (xml_node< Ch > * child) [inline]`

Prepends a new child node. The prepended child becomes the first child, and all existing children are moved one position back.

Parameters

<i>child</i>	Node to prepend.
--------------	------------------

4.39.3.13 `template<class Ch = char> xml_node<Ch>* rapidxml::xml_node< Ch >::previous_sibling (const Ch * name = 0, std::size_t name_size = 0, bool case_sensitive = true) const [inline]`

Gets previous sibling node, optionally matching node name. Behaviour is undefined if node has no parent. Use [parent\(\)](#) to test if node has a parent.

Parameters

<i>name</i>	Name of sibling to find, or 0 to return previous sibling regardless of its name; this string doesn't have to be zero-terminated if name_size is non-zero
<i>name_size</i>	Size of name, in characters, or 0 to have size calculated automatically from string
<i>case_sensitive</i>	Should name comparison be case-sensitive; non case-sensitive comparison works properly only for ASCII characters

Returns

Pointer to found sibling, or 0 if not found.

4.39.3.14 `template<class Ch = char> void rapidxml::xml_node< Ch >::remove_attribute (xml_attribute< Ch > * where) [inline]`

Removes specified attribute from node.

Parameters

<i>where</i>	Pointer to attribute to be removed.
--------------	-------------------------------------

4.39.3.15 `template<class Ch = char> void rapidxml::xml_node< Ch >::remove_first_attribute () [inline]`

Removes first attribute of the node. If node has no attributes, behaviour is undefined. Use [first_attribute\(\)](#) to test if node has attributes.

4.39.3.16 `template<class Ch = char> void rapidxml::xml_node< Ch >::remove_first_node () [inline]`

Removes first child node. If node has no children, behaviour is undefined. Use [first_node\(\)](#) to test if node has children.

4.39.3.17 `template<class Ch = char> void rapidxml::xml_node< Ch >::remove_last_attribute () [inline]`

Removes last attribute of the node. If node has no attributes, behaviour is undefined. Use [first_attribute\(\)](#) to test if node has attributes.

4.39.3.18 `template<class Ch = char> void rapidxml::xml_node< Ch >::remove_last_node () [inline]`

Removes last child of the node. If node has no children, behaviour is undefined. Use [first_node\(\)](#) to test if node has children.

4.39.3.19 `template<class Ch = char> node_type rapidxml::xml_node< Ch >::type () const` `[inline]`

Gets type of node.

Returns

Type of node.

4.39.3.20 `template<class Ch = char> void rapidxml::xml_node< Ch >::type (node_type type)` `[inline]`

Sets type of node.

Parameters

<i>type</i>	Type of node to set.
-------------	----------------------

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

- parsing/rapidxml-1.13/[rapidxml.hpp](#)

Chapter 5

File Documentation

5.1 parsing/rapidxml-1.13/rapidxml.hpp File Reference

This file contains rapidxml parser and DOM implementation.

```
#include <cstdlib>
#include <cassert>
#include <new>
#include <iterator>
#include <exception>
```

Classes

- class [rapidxml::parse_error](#)
- class [rapidxml::memory_pool< Ch >](#)
- struct **rapidxml::memory_pool< Ch >::header**
- class [rapidxml::xml_base< Ch >](#)
- class [rapidxml::xml_attribute< Ch >](#)
- class [rapidxml::xml_node< Ch >](#)
- class [rapidxml::xml_document< Ch >](#)
- struct **rapidxml::xml_document< Ch >::whitespace_pred**
- struct **rapidxml::xml_document< Ch >::node_name_pred**
- struct **rapidxml::xml_document< Ch >::attribute_name_pred**
- struct **rapidxml::xml_document< Ch >::text_pred**
- struct **rapidxml::xml_document< Ch >::text_pure_no_ws_pred**
- struct **rapidxml::xml_document< Ch >::text_pure_with_ws_pred**
- struct **rapidxml::xml_document< Ch >::attribute_value_pred< Quote >**
- struct **rapidxml::xml_document< Ch >::attribute_value_pure_pred< Quote >**

Macros

- **#define RAPIDXML_PARSE_ERROR**(what, where) throw parse_error(what, where)
- **#define RAPIDXML_STATIC_POOL_SIZE** (64 * 1024)
- **#define RAPIDXML_DYNAMIC_POOL_SIZE** (64 * 1024)
- **#define RAPIDXML_ALIGNMENT** sizeof(void *)

Enumerations

- enum **node_type** {
rapidxml::node_document, **rapidxml::node_element**, **rapidxml::node_data**, **rapidxml::node_cdata**,
rapidxml::node_comment, **rapidxml::node_declaration**, **rapidxml::node_doctype**, **rapidxml::node_pi** }

Variables

- const int **rapidxml::parse_no_data_nodes** = 0x1
- const int **rapidxml::parse_no_element_values** = 0x2
- const int **rapidxml::parse_no_string_terminators** = 0x4
- const int **rapidxml::parse_no_entity_translation** = 0x8
- const int **rapidxml::parse_no_utf8** = 0x10
- const int **rapidxml::parse_declaration_node** = 0x20
- const int **rapidxml::parse_comment_nodes** = 0x40
- const int **rapidxml::parse_doctype_node** = 0x80
- const int **rapidxml::parse_pi_nodes** = 0x100
- const int **rapidxml::parse_validate_closing_tags** = 0x200
- const int **rapidxml::parse_trim_whitespace** = 0x400
- const int **rapidxml::parse_normalize_whitespace** = 0x800
- const int **rapidxml::parse_default** = 0
- const int **rapidxml::parse_non_destructive** = parse_no_string_terminators | parse_no_entity_translation
- const int **rapidxml::parse_fastest** = parse_non_destructive | parse_no_data_nodes
- const int **rapidxml::parse_full** = parse_declaration_node | parse_comment_nodes | parse_doctype_node | parse_pi_nodes | parse_validate_closing_tags

5.1.1 Detailed Description

This file contains rapidxml parser and DOM implementation.

5.2 parsing/rapidxml-1.13/rapidxml_iterators.hpp File Reference

This file contains rapidxml iterators.

```
#include "rapidxml.hpp"
```

Classes

- class **rapidxml::node_iterator**< Ch >
Iterator of child nodes of [xml_node](#).
- class **rapidxml::attribute_iterator**< Ch >
Iterator of child attributes of [xml_node](#).

5.2.1 Detailed Description

This file contains rapidxml iterators.

5.3 parsing/rapidxml-1.13/rapidxml_print.hpp File Reference

This file contains rapidxml printer implementation.

```
#include "rapidxml.hpp"
#include <ostream>
#include <iterator>
```

Functions

- template<class OutIt, class Ch >
OutIt **rapidxml::print** (OutIt out, const xml_node< Ch > &node, int flags=0)
- template<class Ch >
std::basic_ostream< Ch > & **rapidxml::print** (std::basic_ostream< Ch > &out, const xml_node< Ch > &node, int flags=0)
- template<class Ch >
std::basic_ostream< Ch > & **rapidxml::operator<<** (std::basic_ostream< Ch > &out, const xml_node< Ch > &node)

Variables

- const int **rapidxml::print_no_indenting** = 0x1
Printer flag instructing the printer to suppress indenting of XML. See print() function.

5.3.1 Detailed Description

This file contains rapidxml printer implementation.

5.4 parsing/rapidxml-1.13/rapidxml_utils.hpp File Reference

```
#include "rapidxml.hpp"
#include <vector>
#include <string>
#include <fstream>
#include <stdexcept>
```

Classes

- class **rapidxml::file**< Ch >
Represents data loaded from a file.

Functions

- template<class Ch >
std::size_t **rapidxml::count_children** (xml_node< Ch > *node)
- template<class Ch >
std::size_t **rapidxml::count_attributes** (xml_node< Ch > *node)

5.4.1 Detailed Description

This file contains high-level rapidxml utilities that can be useful in certain simple scenarios. They should probably not be used if maximizing performance is the main objective.

Index

- ~AutomatonInterface
 - AutomatonInterface, 9
- ~GraphInterface
 - GraphInterface, 24
- ~OutputStreamer
 - OutputStreamer, 34
- ~memory_pool
 - rapidxml::memory_pool, 27
- addFrom
 - ColorStorage, 17
- allocate_attribute
 - rapidxml::memory_pool, 27
- allocate_node
 - rapidxml::memory_pool, 27
- allocate_string
 - rapidxml::memory_pool, 27
- append_attribute
 - rapidxml::xml_node, 55
- append_node
 - rapidxml::xml_node, 55
- ArgumentParser, 7
 - parseArguments, 7
- AutomatonBuilder, 8
 - AutomatonBuilder, 8
 - AutomatonBuilder, 8
 - buildAutomaton, 9
- AutomatonInterface, 9
 - ~AutomatonInterface, 9
 - getFinalStates, 9
 - getInitialStates, 9
 - isFinal, 9
 - isInitial, 10
- AutomatonStructure, 10
 - getAllowedValues, 11
 - getFinalStates, 11
 - getInitialStates, 11
 - getStateCount, 11
 - getString, 11
 - getTargetID, 11
 - getTransitionCount, 11
 - isFinal, 11
 - isInitial, 12
 - isTransitionFeasible, 12
- BA
 - UserOptions, 46
- BasicStructure, 12
 - getDirection, 12
 - getSpecieID, 13
- getStateCount, 13
- getStateLevels, 13
- getString, 13
- getTargetID, 13
- getTransitionCount, 13
- BasicStructureBuilder, 14
 - BasicStructureBuilder, 14
 - BasicStructureBuilder, 14
 - buildStructure, 14
- buildAutomaton
 - AutomatonBuilder, 9
- buildFunctions
 - FunctionsBuilder, 22
- buildProduct
 - ProductBuilder, 39
- buildStructure
 - BasicStructureBuilder, 14
 - ParametrizedStructureBuilder, 37
- ClassName, 14
- clear
 - rapidxml::memory_pool, 28
 - rapidxml::xml_document, 53
- clone_node
 - rapidxml::memory_pool, 28
- ColorStorage, 17
 - addFrom, 17
 - getColor, 18
 - getMarking, 18
 - getNeighbours, 18
 - reset, 18
 - update, 19
- coloring
 - UserOptions, 46
- ColoringAnalyzer, 14
 - ColoringAnalyzer, 14
 - ColoringAnalyzer, 14
 - getColors, 15
 - getUnion, 15
 - storeResults, 15
 - startNewRound, 15
- ColoringParser, 15
 - ColoringParser, 16
 - ColoringParser, 16
 - createOutput, 16
 - getColors, 16
 - getColorsCount, 16
 - input, 16
 - openFile, 16
 - output, 17

- outputComputed, 17
- parseMask, 17
- ConstrainsParser, 19
 - getAllColorsNum, 20
 - getColor, 20
 - getColorsNum, 20
 - getSpecieNum, 20
 - getTargetVals, 20
 - parseConstrains, 20
- createOutput
 - ColoringParser, 16
- createStartingParameters
 - SplitManager, 43
- createStreamFile
 - OutputStreamer, 34
- data
 - rapidxml::file, 21
- display
 - WitnessSearcher, 47
- displayWintess
 - UserOptions, 46
- doSynthesis
 - SynthesisManager, 44
- document
 - rapidxml::xml_attribute, 48
 - rapidxml::xml_node, 55
- file
 - rapidxml::file, 21
- findID
 - Model, 29
- first_attribute
 - rapidxml::xml_node, 55
- first_node
 - rapidxml::xml_node, 55
- FunctionsBuilder, 22
 - buildFunctions, 22
 - FunctionsBuilder, 22
 - FunctionsBuilder, 22
- FunctionsStructure, 22
 - getParametersCount, 23
 - getPossibleValues, 23
 - getRegulationsCount, 23
 - getSourceSpecies, 23
 - getSourceValues, 23
 - getSpecieName, 23
 - getSpecieValues, 23
 - getSpeciesCount, 23
 - getStepSize, 23
- getAllColorsCount
 - SplitManager, 43
- getAllColorsNum
 - ConstrainsParser, 20
- getAllowedValues
 - AutomatonStructure, 11
- getBA
 - ProductStructure, 40
- getBAID
 - ProductStructure, 40
- getColor
 - ColorStorage, 18
 - ConstrainsParser, 20
- getColors
 - ColoringAnalyzer, 15
 - ColoringParser, 16
- getColorsCount
 - ColoringParser, 16
- getColorsNum
 - ConstrainsParser, 20
- getCons
 - ProductStructure, 40
- getDirection
 - BasicStructure, 12
- getEdges
 - Model, 29
- getFinalStates
 - AutomatonInterface, 9
 - AutomatonStructure, 11
 - ProductStructure, 40
- getFunc
 - ProductStructure, 40
- getInitialStates
 - AutomatonInterface, 9
 - AutomatonStructure, 11
 - ProductStructure, 40
- getInteractions
 - Model, 30
- getKS
 - ProductStructure, 41
- getKSID
 - ProductStructure, 41
- getMarking
 - ColorStorage, 18
- getMax
 - Model, 30
- getMin
 - Model, 30
- getName
 - Model, 30
- getNeighbours
 - ColorStorage, 18
 - PerColorStorage, 38
- getParametersCount
 - FunctionsStructure, 23
- getPossibleValues
 - FunctionsStructure, 23
- getProcColorsCount
 - SplitManager, 43
- getProductID
 - ProductStructure, 41
- getRegulations
 - Model, 30
- getRegulationsCount
 - FunctionsStructure, 23
- getRoundCount

- SplitManager, 43
- getRoundNum
 - SplitManager, 43
- getRoundRange
 - SplitManager, 43
- getRoundSize
 - SplitManager, 43
- getSourceSpecies
 - FunctionsStructure, 23
- getSourceValues
 - FunctionsStructure, 23
- getSpecieID
 - BasicStructure, 13
- getSpecieName
 - FunctionsStructure, 23
- getSpecieNum
 - ConstrainsParser, 20
- getSpecieValues
 - FunctionsStructure, 23
- getSpeciesCount
 - FunctionsStructure, 23
 - Model, 30
- getStateCount
 - AutomatonStructure, 11
 - BasicStructure, 13
 - GraphInterface, 24
 - Model, 30
 - ParametrizedStructure, 36
 - ProductStructure, 41
- getStateLevels
 - BasicStructure, 13
 - ParametrizedStructure, 36
 - ProductStructure, 41
- getStepSize
 - FunctionsStructure, 23
 - ParametrizedStructure, 36
 - ProductStructure, 41
- getString
 - AutomatonStructure, 11
 - BasicStructure, 13
 - GraphInterface, 24
 - ParametrizedStructure, 36
 - ProductStructure, 41
- getTargetID
 - AutomatonStructure, 11
 - BasicStructure, 13
 - GraphInterface, 25
 - ParametrizedStructure, 36
 - ProductStructure, 41
- getTargetVals
 - ConstrainsParser, 20
- getTransitionCount
 - AutomatonStructure, 11
 - BasicStructure, 13
 - GraphInterface, 25
 - ParametrizedStructure, 36
 - ProductStructure, 42
- getTransitive
 - ParametrizedStructure, 36
 - ProductStructure, 42
- getUnion
 - ColoringAnalyzer, 15
- GraphInterface, 24
 - ~GraphInterface, 24
- getStateCount, 24
- getString, 24
- getTargetID, 25
- getTransitionCount, 25
- increaseRound
 - SplitManager, 44
- input
 - ColoringParser, 16
- insert_attribute
 - rapidxml::xml_node, 56
- insert_node
 - rapidxml::xml_node, 56
- isFinal
 - AutomatonInterface, 9
 - AutomatonStructure, 11
 - Model, 30
 - ProductStructure, 42
- isInitial
 - AutomatonInterface, 10
 - AutomatonStructure, 12
 - ProductStructure, 42
- isResultInFile
 - OutputStreamer, 34
- isTransitionFeasible
 - AutomatonStructure, 12
- last_attribute
 - rapidxml::xml_node, 56
- last_node
 - rapidxml::xml_node, 57
- lastRound
 - SplitManager, 44
- Model, 29
 - findID, 29
 - getEdges, 29
 - getInteractions, 30
 - getMax, 30
 - getMin, 30
 - getName, 30
 - getRegulations, 30
 - getSpeciesCount, 30
 - getStateCount, 30
 - isFinal, 30
- Model::Interaction, 25
- ModelChecker, 31
 - ModelChecker, 31
 - ModelChecker, 31
 - startColoring, 31
- ModelParser, 32
 - ModelParser, 32
 - ModelParser, 32

- parseInput, 32
- name
 - rapidxml::xml_base, 50
- name_size
 - rapidxml::xml_base, 51
- negation
 - UserOptions, 46
- next_attribute
 - rapidxml::xml_attribute, 48
- next_sibling
 - rapidxml::xml_node, 57
- openFile
 - ColoringParser, 16
- ouputClock
 - TimeManager, 45
- output
 - ColoringParser, 17
 - OutputStreamer, 34, 35
- outputComputed
 - ColoringParser, 17
- OutputManager, 33
 - outputRound, 33
 - outputRoundNum, 33
 - outputSummary, 33
- outputRound
 - OutputManager, 33
- outputRoundNum
 - OutputManager, 33
- OutputStreamer, 33
 - ~OutputStreamer, 34
 - createStreamFile, 34
 - isResultInFile, 34
 - output, 34, 35
 - OutputStreamer, 34
 - OutputStreamer, 34
 - testTrait, 35
- outputSummary
 - OutputManager, 33
- ParametrizedStructure, 35
 - getStateCount, 36
 - getStateLevels, 36
 - getStepSize, 36
 - getString, 36
 - getTargetID, 36
 - getTransitionCount, 36
 - getTransitive, 36
- ParametrizedStructureBuilder, 37
 - buildStructure, 37
 - ParametrizedStructureBuilder, 37
 - ParametrizedStructureBuilder, 37
- parent
 - rapidxml::xml_base, 51
- parse
 - rapidxml::xml_document, 53
- parseArguments
 - ArgumentParser, 7
- parseConstrains
 - ConstrainsParser, 20
- parseInput
 - ModelParser, 32
- parseMask
 - ColoringParser, 17
- parsing/rapidxml-1.13/rapidxml.hpp, 61
- parsing/rapidxml-1.13/rapidxml_iterators.hpp, 62
- parsing/rapidxml-1.13/rapidxml_print.hpp, 63
- parsing/rapidxml-1.13/rapidxml_utils.hpp, 63
- PerColorStorage, 38
 - getNeighbours, 38
- prepend_attribute
 - rapidxml::xml_node, 57
- prepend_node
 - rapidxml::xml_node, 57
- previous_attribute
 - rapidxml::xml_attribute, 49
- previous_sibling
 - rapidxml::xml_node, 58
- procCount
 - UserOptions, 46
- procNum
 - UserOptions, 46
- ProductBuilder, 39
 - buildProduct, 39
 - ProductBuilder, 39
 - ProductBuilder, 39
- ProductStructure, 39
 - getBA, 40
 - getBAID, 40
 - getCons, 40
 - getFinalStates, 40
 - getFunc, 40
 - getInitialStates, 40
 - getKS, 41
 - getKSID, 41
 - getProductID, 41
 - getStateCount, 41
 - getStateLevels, 41
 - getStepSize, 41
 - getString, 41
 - getTargetID, 41
 - getTransitionCount, 42
 - getTransitive, 42
 - isFinal, 42
 - isInitial, 42
- rapidxml::attribute_iterator< Ch >, 7
- rapidxml::file
 - data, 21
 - file, 21
 - size, 21
- rapidxml::file< Ch >, 20
- rapidxml::memory_pool
 - ~memory_pool, 27
 - allocate_attribute, 27
 - allocate_node, 27
 - allocate_string, 27

- clear, 28
- clone_node, 28
- set_allocator, 28
- rapidxml::memory_pool< Ch >, 26
- rapidxml::node_iterator< Ch >, 32
- rapidxml::parse_error, 37
 - what, 38
 - where, 38
- rapidxml::xml_attribute
 - document, 48
 - next_attribute, 48
 - previous_attribute, 49
 - xml_attribute, 48
- rapidxml::xml_attribute< Ch >, 48
- rapidxml::xml_base
 - name, 50
 - name_size, 51
 - parent, 51
 - value, 51, 52
 - value_size, 52
- rapidxml::xml_base< Ch >, 49
- rapidxml::xml_document
 - clear, 53
 - parse, 53
- rapidxml::xml_document< Ch >, 52
- rapidxml::xml_node
 - append_attribute, 55
 - append_node, 55
 - document, 55
 - first_attribute, 55
 - first_node, 55
 - insert_attribute, 56
 - insert_node, 56
 - last_attribute, 56
 - last_node, 57
 - next_sibling, 57
 - prepend_attribute, 57
 - prepend_node, 57
 - previous_sibling, 58
 - remove_attribute, 58
 - remove_first_attribute, 58
 - remove_first_node, 58
 - remove_last_attribute, 58
 - remove_last_node, 58
 - type, 58, 59
 - xml_node, 54
- rapidxml::xml_node< Ch >, 53
- remove_attribute
 - rapidxml::xml_node, 58
- remove_first_attribute
 - rapidxml::xml_node, 58
- remove_first_node
 - rapidxml::xml_node, 58
- remove_last_attribute
 - rapidxml::xml_node, 58
- remove_last_node
 - rapidxml::xml_node, 58
- reset
 - ColorStorage, 18
 - robustness
 - UserOptions, 46
 - set_allocator
 - rapidxml::memory_pool, 28
 - setStartPositions
 - SplitManager, 44
 - size
 - rapidxml::file, 21
 - SplitManager, 42
 - createStartingParameters, 43
 - getAllColorsCount, 43
 - getProcColorsCount, 43
 - getRoundCount, 43
 - getRoundNum, 43
 - getRoundRange, 43
 - getRoundSize, 43
 - increaseRound, 44
 - lastRound, 44
 - setStartPositions, 44
 - SplitManager, 43
 - SplitManager, 43
 - valid, 44
 - startClock
 - TimeManager, 45
 - startColoring
 - ModelChecker, 31
 - stats
 - UserOptions, 46
 - storeResults
 - ColoringAnalyzer, 15
 - strartNewRound
 - ColoringAnalyzer, 15
 - SynthesisManager, 44
 - doSynthesis, 44
 - SynthesisManager, 44
 - SynthesisManager, 44
 - testTrait
 - OutputStreamer, 35
 - TimeManager, 45
 - ouputClock, 45
 - startClock, 45
 - timeSerie
 - UserOptions, 47
 - type
 - rapidxml::xml_node, 58, 59
 - update
 - ColorStorage, 19
 - UserOptions, 45
 - BA, 46
 - coloring, 46
 - displayWintess, 46
 - negation, 46
 - procCount, 46
 - procNum, 46
 - robustness, 46

- stats, [46](#)
- timeSerie, [47](#)
- UserOptions, [46](#)
- UserOptions, [46](#)
- verbose, [47](#)
- witnesses, [47](#)
- valid
 - SplitManager, [44](#)
- value
 - rapidxml::xml_base, [51](#), [52](#)
- value_size
 - rapidxml::xml_base, [52](#)
- verbose
 - UserOptions, [47](#)
- what
 - rapidxml::parse_error, [38](#)
- where
 - rapidxml::parse_error, [38](#)
- WitnessSearcher, [47](#)
 - display, [47](#)
 - WitnessSearcher, [47](#)
 - WitnessSearcher, [47](#)
- witnesses
 - UserOptions, [47](#)
- xml_attribute
 - rapidxml::xml_attribute, [48](#)
- xml_node
 - rapidxml::xml_node, [54](#)