stv_v2

Generated by Doxygen 1.9.6

1 README 1
2 Hierarchical Index 3
2.1 Class Hierarchy
3 Class Index 5
3.1 Class List
4 File Index 7
4.1 File List
5 Class Documentation 9
5.1 Agent Class Reference
5.1.1 Detailed Description
5.1.2 Constructor & Destructor Documentation
5.1.2.1 Agent()
5.1.3 Member Function Documentation
5.1.3.1 includesState()
5.2 AgentTemplate Class Reference
5.2.1 Detailed Description
5.2.2 Member Function Documentation
5.2.2.1 addInitial()
5.2.2.2 addLocal()
5.2.2.3 addPersistent()
5.2.2.4 addTransition()
5.2.2.5 generateAgent()
5.2.2.6 setIdent()
5.2.2.7 setInitState()
5.3 Assignment Class Reference
5.3.1 Detailed Description
5.3.2 Constructor & Destructor Documentation
5.3.2.1 Assignment()
5.3.3 Member Function Documentation
5.3.3.1 assign()
5.4 Cfg Struct Reference
5.5 Condition Struct Reference
5.5.1 Detailed Description
5.6 EpistemicClass Struct Reference
5.6.1 Detailed Description
5.7 ExprAdd Class Reference
5.7.1 Detailed Description
5.7.2 Constructor & Destructor Documentation
5.7.2.1 ExprAdd()
5.7.3 Member Function Documentation

5.7.3.1 eval()	 . 19
5.8 ExprAnd Class Reference	 . 19
5.8.1 Detailed Description	 . 19
5.8.2 Constructor & Destructor Documentation	 . 20
5.8.2.1 ExprAnd()	 . 20
5.8.3 Member Function Documentation	 . 20
5.8.3.1 eval()	 . 20
5.9 ExprConst Class Reference	 . 20
5.9.1 Detailed Description	 . 21
5.9.2 Constructor & Destructor Documentation	 . 21
5.9.2.1 ExprConst()	 . 21
5.9.3 Member Function Documentation	 . 21
5.9.3.1 eval()	 . 21
5.10 ExprDiv Class Reference	 . 22
5.10.1 Detailed Description	 . 22
5.10.2 Constructor & Destructor Documentation	 . 22
5.10.2.1 ExprDiv()	 . 22
5.10.3 Member Function Documentation	 . 23
5.10.3.1 eval()	 . 23
5.11 ExprEq Class Reference	 . 23
5.11.1 Detailed Description	 . 24
5.11.2 Constructor & Destructor Documentation	 . 24
5.11.2.1 ExprEq()	 . 24
5.11.3 Member Function Documentation	 . 24
5.11.3.1 eval()	 . 24
5.12 ExprGe Class Reference	 . 25
5.12.1 Detailed Description	 . 25
5.12.2 Constructor & Destructor Documentation	 . 25
5.12.2.1 ExprGe()	 . 25
5.12.3 Member Function Documentation	 . 25
5.12.3.1 eval()	 . 25
5.13 ExprGt Class Reference	 . 26
5.13.1 Detailed Description	 . 26
5.13.2 Constructor & Destructor Documentation	 . 26
5.13.2.1 ExprGt()	 . 26
5.13.3 Member Function Documentation	 . 27
5.13.3.1 eval()	 . 27
5.14 Exprident Class Reference	 . 27
5.14.1 Detailed Description	 . 28
5.14.2 Constructor & Destructor Documentation	 . 28
5.14.2.1 ExprIdent()	 . 28
5.14.3 Member Function Documentation	 . 28

5.14.3.1 eval()	. 28
5.15 ExprLe Class Reference	. 29
5.15.1 Detailed Description	. 29
5.15.2 Constructor & Destructor Documentation	. 29
5.15.2.1 ExprLe()	. 29
5.15.3 Member Function Documentation	. 30
5.15.3.1 eval()	. 30
5.16 ExprLt Class Reference	. 30
5.16.1 Detailed Description	. 31
5.16.2 Constructor & Destructor Documentation	. 31
5.16.2.1 ExprLt()	. 31
5.16.3 Member Function Documentation	. 31
5.16.3.1 eval()	. 31
5.17 ExprMul Class Reference	. 32
5.17.1 Detailed Description	. 32
5.17.2 Constructor & Destructor Documentation	. 32
5.17.2.1 ExprMul()	. 32
5.17.3 Member Function Documentation	. 32
5.17.3.1 eval()	. 32
5.18 ExprNe Class Reference	. 33
5.18.1 Detailed Description	. 33
5.18.2 Constructor & Destructor Documentation	. 33
5.18.2.1 ExprNe()	. 33
5.18.3 Member Function Documentation	. 34
5.18.3.1 eval()	. 34
5.19 ExprNode Class Reference	. 34
5.19.1 Detailed Description	. 34
5.19.2 Member Function Documentation	. 35
5.19.2.1 eval()	. 35
5.20 ExprNot Class Reference	. 35
5.20.1 Detailed Description	. 35
5.20.2 Constructor & Destructor Documentation	. 36
5.20.2.1 ExprNot()	. 36
5.20.3 Member Function Documentation	. 36
5.20.3.1 eval()	. 36
5.21 ExprOr Class Reference	. 36
5.21.1 Detailed Description	. 37
5.21.2 Constructor & Destructor Documentation	. 37
5.21.2.1 ExprOr()	. 37
5.21.3 Member Function Documentation	. 37
5.21.3.1 eval()	. 37
5.22 EvnrRem Class Reference	38

5.22.1 Detailed Description	38
5.22.2 Constructor & Destructor Documentation	38
5.22.2.1 ExprRem()	38
5.22.3 Member Function Documentation	39
5.22.3.1 eval()	39
5.23 ExprSub Class Reference	39
5.23.1 Detailed Description	40
5.23.2 Constructor & Destructor Documentation	40
5.23.2.1 ExprSub()	40
5.23.3 Member Function Documentation	40
5.23.3.1 eval()	40
5.24 Formula Struct Reference	41
5.24.1 Detailed Description	41
5.25 GlobalModel Struct Reference	41
5.25.1 Detailed Description	41
5.26 GlobalModelGenerator Class Reference	42
5.26.1 Detailed Description	43
5.26.2 Member Function Documentation	43
5.26.2.1 checkLocalTransitionConditions()	43
5.26.2.2 computeEpistemicClassHash()	43
5.26.2.3 computeGlobalStateHash()	44
5.26.2.4 expandState()	44
5.26.2.5 findGlobalStateInEpistemicClass()	44
5.26.2.6 findOrCreateEpistemicClass()	45
5.26.2.7 generateGlobalTransitions()	45
5.26.2.8 generateInitState()	45
5.26.2.9 generateStateFromLocalStates()	46
5.26.2.10 getCurrentGlobalModel()	46
5.26.2.11 getFormula()	46
5.26.2.12 initModel()	47
5.27 GlobalState Struct Reference	47
5.27.1 Detailed Description	48
5.28 GlobalTransition Struct Reference	48
5.28.1 Detailed Description	48
5.29 HistoryDbg Class Reference	48
5.29.1 Detailed Description	49
5.29.2 Member Function Documentation	49
5.29.2.1 addEntry()	49
5.29.2.2 cloneEntry()	49
5.29.2.3 markEntry()	50
5.29.2.4 print()	50
5.30 HistoryEntry Struct Reference	50

5.30.1 Detailed Description	51
5.30.2 Member Function Documentation	51
5.30.2.1 toString()	51
5.31 LocalModels Struct Reference	52
5.31.1 Detailed Description	52
5.32 LocalState Class Reference	52
5.32.1 Detailed Description	53
5.32.2 Member Function Documentation	53
5.32.2.1 compare()	53
5.33 LocalStateTemplate Class Reference	53
5.33.1 Detailed Description	53
5.34 LocalTransition Struct Reference	54
5.34.1 Detailed Description	54
5.35 SeleneFormula Class Reference	54
5.36 SeleneFormula1 Class Reference	55
5.36.1 Member Function Documentation	55
5.36.1.1 verifyLocalStates()	55
5.37 TestParser Class Reference	55
5.37.1 Detailed Description	56
5.37.2 Member Function Documentation	56
5.37.2.1 parse()	56
5.38 TransitionTemplate Class Reference	56
5.38.1 Detailed Description	57
5.38.2 Constructor & Destructor Documentation	57
5.38.2.1 TransitionTemplate()	57
5.39 Var Struct Reference	58
5.39.1 Detailed Description	58
5.40 VarAssignment Struct Reference	58
5.41 Verification Class Reference	59
5.41.1 Detailed Description	60
5.41.2 Constructor & Destructor Documentation	60
5.41.2.1 Verification()	60
5.41.3 Member Function Documentation	60
5.41.3.1 addHistoryContext()	60
5.41.3.2 addHistoryDecision()	61
5.41.3.3 addHistoryMarkDecisionAsInvalid()	61
5.41.3.4 addHistoryStateStatus()	61
5.41.3.5 areGlobalStatesInTheSameEpistemicClass()	62
5.41.3.6 getEpistemicClassForGlobalState()	62
5.41.3.7 isAgentInCoalition()	63
5.41.3.8 isGlobalTransitionControlledByCoalition()	63
5.41.3.9 newHistoryMarkDecisionAsInvalid()	63

	5.41.3.10 printCurrentHistory()	64
	5.41.3.11 revertLastDecision()	64
	5.41.3.12 undoHistoryUntil()	64
	5.41.3.13 undoLastHistoryEntry()	65
	5.41.3.14 verify()	65
	5.41.3.15 verifyGlobalState()	65
	5.41.3.16 verifyLocalStates()	66
_		
b	File Documentation	67
	6.1 Constants.hpp	67
	6.2 GlobalModelGenerator.cpp File Reference	67
	6.2.1 Detailed Description	67
	6.3 GlobalModelGenerator.hpp File Reference	67
	6.3.1 Detailed Description	68
	6.4 GlobalModelGenerator.hpp	68
	6.5 expressions.cc File Reference	68
	6.5.1 Detailed Description	68
	6.6 expressions.hpp File Reference	69
	6.6.1 Detailed Description	70
	6.7 expressions.hpp	70
	6.8 nodes.cc File Reference	73
	6.8.1 Detailed Description	73
	6.9 nodes.hpp File Reference	73
	6.9.1 Detailed Description	73
	6.10 nodes.hpp	74
	6.11 SeleneFormula.hpp	75
	6.12 TestParser.hpp	76
	6.13 Types.cc File Reference	76
	6.13.1 Detailed Description	76
	6.14 Types.hpp File Reference	76
	6.14.1 Detailed Description	77
	6.14.2 Enumeration Type Documentation	77
	6.14.2.1 ConditionOperator	77
	6.14.2.2 GlobalStateVerificationStatus	78
	6.14.2.3 VarAssignmentType	78
	6.15 Types.hpp	78
	6.16 Utils.cpp File Reference	81
	6.16.1 Detailed Description	82
	6.16.2 Function Documentation	82
	6.16.2.1 agentToString()	82
	6.16.2.2 envToString()	82
	6.16.2.3 localModelsToString()	82

6.16.2.4 outputGlobalModel()	83
6.17 Utils.hpp File Reference	83
6.17.1 Function Documentation	84
6.17.1.1 agentToString()	84
6.17.1.2 envToString()	84
6.17.1.3 localModelsToString()	84
6.17.1.4 outputGlobalModel()	85
6.18 Utils.hpp	85
6.19 Verification.cpp File Reference	85
6.19.1 Detailed Description	86
6.19.2 Function Documentation	86
6.19.2.1 dbgHistEnt()	86
6.19.2.2 dbgVerifStatus()	
6.19.2.3 verStatusToStr()	
6.20 Verification.hpp File Reference	
6.20.1 Enumeration Type Documentation	
6.20.1.1 HistoryEntryType	
6.20.1.2 Mode	
6.20.2 Function Documentation	
6.20.2.1 verStatusToStr()	
6.21 Verification.hpp	
0.21 Verilloation.hpp	09
Index	91

Chapter 1

README

To run: cd build make clean make ./stv

Configuration file: build/config.txt

2 README

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Agent	9
AgentTemplate	10
Assignment	15
Ofg	16
Condition	17
EpistemicClass	17
ExprNode	34
ExprAdd	18
ExprAnd	19
ExprConst	20
ExprDiv	22
ExprEq	23
ExprGe	25
ExprGt	26
Exprident	27
ExprLe	29
ExprLt	30
ExprMul	32
ExprNe	33
ExprNot	35
ExprOr	36
ExprRem	38
ExprSub	39
Formula	41
GlobalModel	41
GlobalModelGenerator	42
GlobalState	47
GlobalTransition	48
HistoryDbg	48
HistoryEntry	50
ocalModels	52
ocalState	52
ocalStateTemplate	53
ocalTransition	54
SeleneFormula	54

Hierarchical Index

S	eleneFormula1								 												5	5
TestF	arser	 		 			 														5	5
Trans	itionTemplate	 		 			 										 				5	6
Var .		 		 			 										 				5	8
VarA	ssignment .	 		 			 										 				5	8
\/orifi	cation																				5	a

Chapter 3

Class Index

3.1 Class List

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

Agent	
Contains all data for a single Agent, including id, name and all of the agents' variables	9
AgentTemplate	
Represents a single agent loaded from the description from a file	10
Assignment	
Represents an assingment	15
Cfg	16
Condition	
Represents a condition for LocalTransition	17
EpistemicClass	
Represents a single epistemic class	17
ExprAdd	
Node for addition	18
ExprAnd	
Node for AND operator	19
ExprConst Node for a constant	20
ExprDiv	20
Node for division	22
ExprEq	
Node for "==" operator	23
ExprGe	
Node for ">=" operator	25
ExprGt	
Node for ">" operator	26
Exprident	
Node for an identifier	27
ExprLe	
Node for "<=" operator	29
ExprLt	
Node for "<" operator	30
ExprMul	
Node for multiplication	32
ExprNe	
Node for "!=" operator	33

6 Class Index

ExprNod	8	
□NI=4	Base node for expressions	34
ExprNot	Node for NOT operator	35
ExprOr	Node for NOT operator	00
	Node for OR operator	36
ExprRem		
	Node for modulo	38
ExprSub		
Formula	Node for subtraction	39
Formula	Contains a coalition of Agent from the formula	41
GlobalMo		71
	Represents a global model, containing agents and a formula	41
GlobalMo	odelGenerator	
	Stores the local models, formula and a global model	42
GlobalSta		
OL 1 IT	Represents a single global state	47
GlobalTra		40
HistoryDI	Represents a single global transition	48
i iistoi yDi	Stores history and allows displaying it to the console	48
HistoryE		
	Structure used to save model traversal history	50
LocalMod		
	Represents a single local model, contains all agents and variables	52
LocalSta		
LocalStat	Represents a single LocalState, containing id, name and internal variables teTemplate	52
LUCAISIA	A template for the local state	53
LocalTrar	•	00
	Represents a single local transition, containing id, global name, local name, is shared and count	
	of the appearances	54
SeleneFo	ormula	54
SeleneFo		55
TestPars		
Transition	A parser for converting a text file into a model	55
Iransilior	Represents a meta-transition	56
Var	riepiesenis a meta-transition	50
·u	Represents a variable in the model, containing name, initial value and persistence	58
VarAssig	nment	58
Verification		
	A class that verifies if the model fulfills the formula. Also can do some operations on decision	
	history	59

Chapter 4

File Index

4.1 File List

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

Constants.hpp	37
GlobalModelGenerator.cpp	
Generator of a global model. Class for initializing and generating a global model 6	67
GlobalModelGenerator.hpp	
Generator of a global model. Class for initializing and generating a global model 6	67
expressions.cc	
	8
expressions.hpp	
Eval and helper class for expressions. Eval and helper class for expressions 6	39
nodes.cc	
Parser templates. Class for setting up a new objects from a parser	73
nodes.hpp	
Parser templates. Class for setting up a new objects from a parser	73
SeleneFormula.hpp	75
TestParser.hpp	76
Types.cc	
Custom data structures. Data structures and classes containing model data	76
Types.hpp	
Custom data structures. Data structures and classes containing model data	76
Utils.cpp	
Utility functions. A collection of utility functions to use in the project	31
Utils.hpp	33
Verification.cpp	
Class for verification of the formula on a model. Class for verification of the specified formula on	
a specified model	35
Verification.hpp	37

8 File Index

Chapter 5

Class Documentation

5.1 Agent Class Reference

Contains all data for a single Agent, including id, name and all of the agents' variables.

```
#include <Types.hpp>
```

Public Member Functions

• Agent (int _id, string _name)

Constructor for the Agent class, assigning it an id and name.

LocalState * includesState (LocalState *state)

Checks if there is an equivalent LocalState in the model to the one passed as an argment.

Public Attributes

int id

Identifier of the agent.

• string name

Name of the agent.

set< Var * > vars

Variable names for the agent.

• LocalState * initState

Initial state of the agent.

vector < LocalState * > localStates

Local states for this agent.

vector < LocalTransition * > localTransitions

Local transitions for this agent.

5.1.1 Detailed Description

Contains all data for a single Agent, including id, name and all of the agents' variables.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 Agent()

Constructor for the Agent class, assigning it an id and name.

Parameters

_id	Identifier of the new agent.
_name	Name of the new agent.

5.1.3 Member Function Documentation

5.1.3.1 includesState()

Checks if there is an equivalent LocalState in the model to the one passed as an argment.

Parameters

state	A pointer to LocalState to be checked.
-------	--

Returns

Returns a pointer to an equivalent LocalState if such exists, otherwise returns NULL.

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

- · Types.hpp
- Types.cc

5.2 AgentTemplate Class Reference

Represents a single agent loaded from the description from a file.

```
#include <nodes.hpp>
```

Public Member Functions

• AgentTemplate ()

Constructor for an AgentTemplate.

virtual AgentTemplate & setIdent (string _ident)

Set the identifier of an agent.

virtual AgentTemplate & setInitState (string _startState)

Set the initial state of the agent.

virtual AgentTemplate & addLocal (set< string > *variables)

Adds local variables to an agent.

virtual AgentTemplate & addPersistent (set< string > *variables)

Adds persistent variables to an agent.

virtual AgentTemplate & addInitial (set< Assignment * > *assigns)

Adds initial assignments.

virtual AgentTemplate & addTransition (TransitionTemplate *_transition)

Adds a transition to the agent.

virtual Agent * generateAgent (int id)

Generate a new agent for the model.

5.2.1 Detailed Description

Represents a single agent loaded from the description from a file.

5.2.2 Member Function Documentation

5.2.2.1 addInitial()

Adds initial assignments.

Sets initial values of agent's variables.

Parameters

assigns	Assignments to be added.

Returns

Returns a pointer to self.

Parameters

assigns	Set of variables to assign.

Returns

Returns itself.

5.2.2.2 addLocal()

```
AgentTemplate & AgentTemplate::addLocal ( set < string > * \ variables \ ) \quad [virtual]
```

Adds local variables to an agent.

Adds local variables to the agent.

Parameters

Returns

Returns a pointer to self.

Parameters

variables A pointer to a set of strings w	vith the variables to be added.
---	---------------------------------

Returns

Returns itself.

5.2.2.3 addPersistent()

Adds persistent variables to an agent.

Adds persistent variables to the agent.

Parameters

Returns

Returns a pointer to self.

Parameters

variables	A pointer to a set of strings with the variables to be added.
-----------	---

Returns

Returns itself.

5.2.2.4 addTransition()

Adds a transition to the agent.

Adds a transition for the agent.

Parameters

Returns

Returns a pointer to self.

Parameters

_transition	Transition to be added.
-------------	-------------------------

Returns

Returns itself.

5.2.2.5 generateAgent()

Generate a new agent for the model.

Generates an agent for the model.

Parameters

id Identification number defining a new Agent.

Returns

Returns a pointer to a new Agent.

Parameters

id Identifier of the new Agent.

Returns

Returns a pointer to a newly created Agent.

5.2.2.6 setIdent()

```
AgentTemplate & AgentTemplate::setIdent (
          string _ident ) [virtual]
```

Set the identifier of an agent.

Sets the identifier of an agent.

Parameters

_ident	New agent identifier.

Returns

Returns a pointer to self.

Parameters

_ident	String with a new identifier.
--------	-------------------------------

Returns

Returns itself.

5.2.2.7 setInitState()

Set the initial state of the agent.

Sets initial state of an agent.

Parameters

_startState	New inital agent state.
-------------	-------------------------

Returns

Returns a pointer to self.

Parameters

_initState	String with a new state.
------------	--------------------------

Returns

Returns itself.

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

- nodes.hpp
- nodes.cc

5.3 Assignment Class Reference

Represents an assingment.

```
#include <nodes.hpp>
```

Public Member Functions

Assignment (string _ident, ExprNode *_exp)

Constructor for an Assignment class.

• virtual void assign (Environment &env)

Make an assignment in a given environment.

Public Attributes

· string ident

To what we should assign a value.

ExprNode * value

A value to be assigned.

5.3.1 Detailed Description

Represents an assingment.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 Assignment()

Constructor for an Assignment class.

Parameters

_ident	To what we should assign a value.
_exp	A value to be assigned.

5.3.3 Member Function Documentation

5.3.3.1 assign()

Make an assignment in a given environment.

Parameters

env	Environment in which to make an assignment.

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

• nodes.hpp

5.4 Cfg Struct Reference

Public Attributes

• char * fname

- · char stv_mode
- bool output_local_models
- bool output_global_model
- int model_id

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

· Constants.hpp

5.5 Condition Struct Reference

Represents a condition for LocalTransition.

```
#include <Types.hpp>
```

Public Attributes

Var * var

Pointer to a variable.

• ConditionOperator conditionOperator

Conditional operator for the variable.

int comparedValue

Condition value to be met.

5.5.1 Detailed Description

Represents a condition for LocalTransition.

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

• Types.hpp

5.6 EpistemicClass Struct Reference

Represents a single epistemic class.

```
#include <Types.hpp>
```

Public Attributes

· string hash

Hash of that epistemic class.

map< string, GlobalState * > globalStates

Map of GlobalState hashes to according GlobalState pointers bound to this epistemic class.

GlobalTransition * fixedCoalitionTransition

Transition that was already selected in this epistemic class. Model has to choose this transition if it is already set.

5.6.1 Detailed Description

Represents a single epistemic class.

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

• Types.hpp

5.7 ExprAdd Class Reference

Node for addition.

```
#include <expressions.hpp>
```

Public Member Functions

```
• ExprAdd (ExprNode *_larg, ExprNode *_rarg)

Addition expression constructor.
```

• virtual int eval (Environment &env)

Calculates the expression value.

virtual int eval (Environment &env)=0
 Calculates the expression value.

5.7.1 Detailed Description

Node for addition.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 ExprAdd()

Addition expression constructor.

Parameters

_larg	Left argument of the expression.
_rarg	Right argument of the expression.

5.7.3 Member Function Documentation

5.7.3.1 eval()

Calculates the expression value.

Parameters

env Environment values.

Returns

Returns an integer.

Implements ExprNode.

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

- expressions.hpp
- expressions.cc

5.8 ExprAnd Class Reference

Node for AND operator.

```
#include <expressions.hpp>
```

Public Member Functions

```
    ExprAnd (ExprNode *_larg, ExprNode *_rarg)
```

Logic AND expression constructor.

• virtual int eval (Environment &env)

Calculates the expression value.

• virtual int eval (Environment &env)=0

Calculates the expression value.

5.8.1 Detailed Description

Node for AND operator.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 ExprAnd()

Logic AND expression constructor.

Parameters

_larg	Left argument of the expression.
_rarg	Right argument of the expression.

5.8.3 Member Function Documentation

5.8.3.1 eval()

Calculates the expression value.

Parameters

env	Environment values.
-----	---------------------

Returns

Returns an integer.

Implements ExprNode.

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

- · expressions.hpp
- expressions.cc

5.9 ExprConst Class Reference

Node for a constant.

```
#include <expressions.hpp>
```

Public Member Functions

```
    ExprConst (int _val)
```

Constant expression constructor.

• virtual int eval (Environment &env)

Calculates the expression value.

• virtual int eval (Environment &env)=0

Calculates the expression value.

5.9.1 Detailed Description

Node for a constant.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 ExprConst()

Constant expression constructor.

Parameters

_val ExprConst value.

5.9.3 Member Function Documentation

5.9.3.1 eval()

Calculates the expression value.

Parameters

env Environment values.

Returns

Returns an integer.

Implements ExprNode.

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

- · expressions.hpp
- expressions.cc

5.10 ExprDiv Class Reference

Node for division.

```
#include <expressions.hpp>
```

Public Member Functions

```
    ExprDiv (ExprNode *_larg, ExprNode *_rarg)
    Division expression constructor.
```

• virtual int eval (Environment &env)

Calculates the expression value.

• virtual int eval (Environment &env)=0

Calculates the expression value.

5.10.1 Detailed Description

Node for division.

5.10.2 Constructor & Destructor Documentation

5.10.2.1 ExprDiv()

Division expression constructor.

Parameters

_larg	Left argument of the expression.
_rarg	Right argument of the expression.

5.10.3 Member Function Documentation

5.10.3.1 eval()

Calculates the expression value.

Parameters

Returns

Returns an integer.

Implements ExprNode.

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

- · expressions.hpp
- expressions.cc

5.11 ExprEq Class Reference

```
Node for "==" operator.
#include <expressions.hpp>
```

Public Member Functions

- ExprEq (ExprNode *_larg, ExprNode *_rarg)
 - Equals expression constructor.
- virtual int eval (Environment &env)

Calculates the expression value.

• virtual int eval (Environment &env)=0

Calculates the expression value.

5.11.1 Detailed Description

Node for "==" operator.

5.11.2 Constructor & Destructor Documentation

5.11.2.1 ExprEq()

Equals expression constructor.

Parameters

_larg	Left argument of the expression.
_rarg	Right argument of the expression.

5.11.3 Member Function Documentation

5.11.3.1 eval()

Calculates the expression value.

Parameters

env	Environment values.

Returns

Returns an integer.

Implements ExprNode.

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

- expressions.hpp
- expressions.cc

5.12 ExprGe Class Reference

```
Node for ">=" operator.
#include <expressions.hpp>
```

Public Member Functions

```
• ExprGe (ExprNode *_larg, ExprNode *_rarg)

Greater or equal expression constructor.
```

• virtual int eval (Environment &env)

Calculates the expression value.

virtual int eval (Environment &env)=0
 Calculates the expression value.

5.12.1 Detailed Description

Node for ">=" operator.

5.12.2 Constructor & Destructor Documentation

5.12.2.1 ExprGe()

Greater or equal expression constructor.

Parameters

_larg	Left argument of the expression.
_rarg	Right argument of the expression.

5.12.3 Member Function Documentation

5.12.3.1 eval()

Calculates the expression value.

Parameters

```
env Environment values.
```

Returns

Returns an integer.

Implements ExprNode.

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

- · expressions.hpp
- expressions.cc

5.13 ExprGt Class Reference

```
Node for ">" operator.
#include <expressions.hpp>
```

Public Member Functions

```
    ExprGt (ExprNode *_larg, ExprNode *_rarg)
        Greater than expression constructor.
    virtual int eval (Environment &env)
```

Calculates the expression value.

virtual int eval (Environment &env)=0
 Calculates the expression value.

5.13.1 Detailed Description

Node for ">" operator.

5.13.2 Constructor & Destructor Documentation

5.13.2.1 ExprGt()

Greater than expression constructor.

Parameters

_larg	Left argument of the expression.
_rarg	Right argument of the expression.

5.13.3 Member Function Documentation

5.13.3.1 eval()

Calculates the expression value.

Parameters

env	Environment values.
-----	---------------------

Returns

Returns an integer.

Implements ExprNode.

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

- · expressions.hpp
- expressions.cc

5.14 Exprident Class Reference

Node for an identifier.

```
#include <expressions.hpp>
```

Public Member Functions

• Exprident (string _ident)

Identifier expression constructor.

virtual int eval (Environment &env)

Calculates the expression value.

• virtual int eval (Environment &env)=0

Calculates the expression value.

5.14.1 Detailed Description

Node for an identifier.

5.14.2 Constructor & Destructor Documentation

5.14.2.1 Exprident()

Identifier expression constructor.

Parameters

_ident | Exprldent value.

5.14.3 Member Function Documentation

5.14.3.1 eval()

Calculates the expression value.

Parameters

env Environment values.

Returns

Returns an integer.

Parameters

env

Returns

Implements ExprNode.

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

- · expressions.hpp
- expressions.cc

5.15 ExprLe Class Reference

```
Node for "<=" operator.
#include <expressions.hpp>
```

Public Member Functions

```
• ExprLe (ExprNode *_larg, ExprNode *_rarg)

Less or equal expression constructor.
```

• virtual int eval (Environment &env)

Calculates the expression value.

virtual int eval (Environment &env)=0
 Calculates the expression value.

5.15.1 Detailed Description

```
Node for "<=" operator.
```

5.15.2 Constructor & Destructor Documentation

5.15.2.1 ExprLe()

Less or equal expression constructor.

Parameters

_larg	Left argument of the expression.
_rarg	Right argument of the expression.

5.15.3 Member Function Documentation

5.15.3.1 eval()

Calculates the expression value.

Parameters

<i>env</i> Environment values.

Returns

Returns an integer.

Implements ExprNode.

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

- · expressions.hpp
- expressions.cc

5.16 ExprLt Class Reference

```
Node for "<" operator.
#include <expressions.hpp>
```

Public Member Functions

```
• ExprLt (ExprNode *_larg, ExprNode *_rarg)
```

Less than expression constructor.
• virtual int eval (Environment & env)

Calculates the expression value.

• virtual int eval (Environment &env)=0

Calculates the expression value.

5.16.1 Detailed Description

Node for "<" operator.

5.16.2 Constructor & Destructor Documentation

5.16.2.1 ExprLt()

Less than expression constructor.

Parameters

_larg	Left argument of the expression.
_rarg	Right argument of the expression.

5.16.3 Member Function Documentation

5.16.3.1 eval()

Calculates the expression value.

Parameters

env	Environment values.

Returns

Returns an integer.

Implements ExprNode.

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

- expressions.hpp
- expressions.cc

5.17 ExprMul Class Reference

Node for multiplication.

```
#include <expressions.hpp>
```

Public Member Functions

```
    ExprMul (ExprNode *_larg, ExprNode *_rarg)
```

Multiplication expression constructor.

virtual int eval (Environment &env)

Calculates the expression value.

virtual int eval (Environment &env)=0

Calculates the expression value.

5.17.1 Detailed Description

Node for multiplication.

5.17.2 Constructor & Destructor Documentation

5.17.2.1 ExprMul()

Multiplication expression constructor.

Parameters

_larg	Left argument of the expression.
_rarg	Right argument of the expression.

5.17.3 Member Function Documentation

5.17.3.1 eval()

Calculates the expression value.

Parameters

```
env Environment values.
```

Returns

Returns an integer.

Implements ExprNode.

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

- · expressions.hpp
- expressions.cc

5.18 ExprNe Class Reference

```
Node for "!=" operator.

#include <expressions.hpp>
```

Public Member Functions

```
    ExprNe (ExprNode *_larg, ExprNode *_rarg)
        Not equals expression constructor.
        virtual int eval (Environment &env)
```

Calculates the expression value.

• virtual int eval (Environment &env)=0

Calculates the expression value.

5.18.1 Detailed Description

Node for "!=" operator.

5.18.2 Constructor & Destructor Documentation

5.18.2.1 ExprNe()

Not equals expression constructor.

Parameters

_larg	Left argument of the expression.
_rarg	Right argument of the expression.

5.18.3 Member Function Documentation

5.18.3.1 eval()

Calculates the expression value.

Parameters

Returns

Returns an integer.

Implements ExprNode.

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

- expressions.hpp
- expressions.cc

5.19 ExprNode Class Reference

Base node for expressions.

```
#include <expressions.hpp>
```

Public Member Functions

• virtual int eval (Environment &env)=0

Calculates the expression value.

5.19.1 Detailed Description

Base node for expressions.

5.19.2 Member Function Documentation

5.19.2.1 eval()

Calculates the expression value.

Parameters

env Environment values.

Returns

Returns an integer.

Implemented in ExprConst, ExprIdent, ExprAdd, ExprSub, ExprMul, ExprDiv, ExprRem, ExprAnd, ExprOr, ExprNot, ExprEq, ExprNe, ExprLe, ExprLe, ExprGt, and ExprGe.

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

• expressions.hpp

5.20 ExprNot Class Reference

Node for NOT operator.

```
#include <expressions.hpp>
```

Public Member Functions

```
    ExprNot (ExprNode *_arg)
```

Logic NOT expression constructor.

virtual int eval (Environment &env)

Calculates the expression value.

• virtual int eval (Environment &env)=0

Calculates the expression value.

5.20.1 Detailed Description

Node for NOT operator.

5.20.2 Constructor & Destructor Documentation

5.20.2.1 ExprNot()

Logic NOT expression constructor.

Parameters

5.20.3 Member Function Documentation

5.20.3.1 eval()

Calculates the expression value.

Parameters

env Environment values.

Returns

Returns an integer.

Implements ExprNode.

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

- expressions.hpp
- expressions.cc

5.21 ExprOr Class Reference

Node for OR operator.

#include <expressions.hpp>

Public Member Functions

```
• ExprOr (ExprNode *_larg, ExprNode *_rarg)

Logic OR expression constructor.
```

• virtual int eval (Environment &env)

Calculates the expression value.

• virtual int eval (Environment &env)=0

Calculates the expression value.

5.21.1 Detailed Description

Node for OR operator.

5.21.2 Constructor & Destructor Documentation

5.21.2.1 ExprOr()

Logic OR expression constructor.

Parameters

_larg	Left argument of the expression.
_rarg	Right argument of the expression.

5.21.3 Member Function Documentation

5.21.3.1 eval()

Calculates the expression value.

Parameters

env	Environment values.

Returns

Returns an integer.

Implements ExprNode.

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

- · expressions.hpp
- expressions.cc

5.22 ExprRem Class Reference

Node for modulo.

```
#include <expressions.hpp>
```

Public Member Functions

```
• ExprRem (ExprNode *_larg, ExprNode *_rarg)
```

Modulo expression constructor.

• virtual int eval (Environment &env)

Calculates the expression value.

• virtual int eval (Environment &env)=0

Calculates the expression value.

5.22.1 Detailed Description

Node for modulo.

5.22.2 Constructor & Destructor Documentation

5.22.2.1 ExprRem()

Modulo expression constructor.

Parameters

_larg	Left argument of the expression.
_rarg	Right argument of the expression.

5.22.3 Member Function Documentation

5.22.3.1 eval()

Calculates the expression value.

Parameters

env	Environment values.
-----	---------------------

Returns

Returns an integer.

Implements ExprNode.

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

- · expressions.hpp
- expressions.cc

5.23 ExprSub Class Reference

Node for subtraction.

```
#include <expressions.hpp>
```

Public Member Functions

- ExprSub (ExprNode *_larg, ExprNode *_rarg)
 - Subtraction expression constructor.
- virtual int eval (Environment &env)

Calculates the expression value.

• virtual int eval (Environment &env)=0

Calculates the expression value.

5.23.1 Detailed Description

Node for subtraction.

5.23.2 Constructor & Destructor Documentation

5.23.2.1 ExprSub()

Subtraction expression constructor.

Parameters

_larg	Left argument of the expression.
_rarg	Right argument of the expression.

5.23.3 Member Function Documentation

5.23.3.1 eval()

Calculates the expression value.

Parameters

env	Environment values.

Returns

Returns an integer.

Implements ExprNode.

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

- expressions.hpp
- expressions.cc

5.24 Formula Struct Reference

Contains a coalition of Agent from the formula.

```
#include <Types.hpp>
```

Public Attributes

set < Agent * > coalition
 Coalition of Agent from the formula.

5.24.1 Detailed Description

Contains a coalition of Agent from the formula.

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

• Types.hpp

5.25 GlobalModel Struct Reference

Represents a global model, containing agents and a formula.

```
#include <Types.hpp>
```

Public Attributes

vector< Agent * > agents

Pointers to all agents in a model.

• Formula * formula

A pointer to a Formula.

GlobalState * initState

Pointer to the initial state of the model.

vector < GlobalState * > globalStates

Every GlobalState in the model.

vector < GlobalTransition * > globalTransitions

Every GlobalTransition in the model.

map < Agent *, map < string, EpistemicClass * > > epistemicClasses
 Map of Agent pointers to a map of EpistemicClass.

5.25.1 Detailed Description

Represents a global model, containing agents and a formula.

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

Types.hpp

5.26 GlobalModelGenerator Class Reference

Stores the local models, formula and a global model.

#include <GlobalModelGenerator.hpp>

Public Member Functions

· GlobalModelGenerator ()

Constructor for GlobalModelGenerator class.

∼GlobalModelGenerator ()

Destructor for GlobalModelGenerator class.

GlobalState * initModel (LocalModels *localModels, Formula *formula)

Initializes a global model from local models and a formula.

void expandState (GlobalState *state)

Goes through all GlobalTransition in a given GlobalState and creates new GlobalStates connected to the given one.

void expandAllStates ()

Expands the states starting from the initial GlobalState and continues until there are no more states to expand.

GlobalModel * getCurrentGlobalModel ()

Get for a GlobalModel used in initialization.

Formula * getFormula ()

Get for the Formula used in initialization.

Protected Member Functions

GlobalState * generateInitState ()

Generates initial state of the model from GlobalModel in memory.

GlobalState * generateStateFromLocalStates (set< LocalState * > *localStates, set< LocalTransition * > *viaLocalTransitions, GlobalState *prevGlobalState)

Creates a new GlobalState using some of the internally known model data and given local states, transitions that were uset to get there and the previous global state.

 void generateGlobalTransitions (GlobalState *fromGlobalState, set< LocalTransition * > localTransitions, map< Agent *, vector< LocalTransition * > > transitionsByAgent)

Adds all shared global transitions to a GlobalState.

bool checkLocalTransitionConditions (LocalTransition *localTransition, GlobalState *globalState)

Checks if all conditions for a given local transition in a given global state are fulfilled.

string computeEpistemicClassHash (set< LocalState * > *localStates, Agent *agent)

Creates a hash from a set of LocalState and an Agent.

string computeGlobalStateHash (set< LocalState * > *localStates)

Creates a hash from a set of LocalState.

EpistemicClass * findOrCreateEpistemicClass (set< LocalState * > *localStates, Agent *agent)

Checks if a set of LocalState is already an epistemic class for a given Agent, if not, creates a new one.

GlobalState * findGlobalStateInEpistemicClass (set< LocalState * > *localStates, EpistemicClass *epistemicClass)

Gets a GlobalState from an EpistemicClass if it exists in that episcemic class.

Protected Attributes

LocalModels * localModels

LocalModels used in initModel.

• Formula * formula

Formula used in initModel.

• GlobalModel * globalModel

GlobalModel created in initModel.

5.26.1 Detailed Description

Stores the local models, formula and a global model.

5.26.2 Member Function Documentation

5.26.2.1 checkLocalTransitionConditions()

```
\label{localTransition} bool \ \mbox{GlobalModelGenerator::checkLocalTransitionConditions} \ \ ( \\ \mbox{LocalTransition} * \mbox{localTransition,} \\ \mbox{GlobalState} * \mbox{globalState} \ ) \ \ \mbox{[protected]}
```

Checks if all conditions for a given local transition in a given global state are fulfilled.

Parameters

localTransition	Local transition to traverse.
globalState	Current global state.

Returns

Returns true if all of the necessary conditions to use that transition are fulfilled, otherwise false.

5.26.2.2 computeEpistemicClassHash()

Creates a hash from a set of LocalState and an Agent.

Parameters

localStates	Pointer to a set of pointers of LocalState and pointer to and Agent to turn into a hash.

Returns

Returns a string with a hash.

5.26.2.3 computeGlobalStateHash()

```
string GlobalModelGenerator::computeGlobalStateHash ( set < LocalState * > * localStates ) \quad [protected]
```

Creates a hash from a set of LocalState.

Parameters

Returns

Returns a string with a hash.

5.26.2.4 expandState()

Goes through all GlobalTransition in a given GlobalState and creates new GlobalStates connected to the given one.

Parameters

state A state from which the expansion should start.

5.26.2.5 findGlobalStateInEpistemicClass()

Gets a GlobalState from an EpistemicClass if it exists in that episcemic class.

Parameters

localStates	Pointer to a set of pointers to LocalState, from which will be generated a global state hash.
epistemicClass	Epistemic class in which to check if a GlobalState exists.

Returns

Returns a pointer to a GlobalState if it exists in given epistemic class, otherwise returns nullptr.

5.26.2.6 findOrCreateEpistemicClass()

Checks if a set of LocalState is already an epistemic class for a given Agent, if not, creates a new one.

Parameters

localStates	Local states from agent.	1
agent	Agent for which to check the existence of an epistemic class.]

Returns

A pointer to a new or existing EpistemicClass.

5.26.2.7 generateGlobalTransitions()

Adds all shared global transitions to a GlobalState.

Parameters

fromGlobalState	Global state to add transitions to.
localTransitions	Initially empty, avaliable local transitions by each agent from transitionsByAgent.
transitionsByAgent	Mapped transitions to an agent, only with transitions avaliable for the agent at this moment.

5.26.2.8 generateInitState()

```
GlobalState * GlobalModelGenerator::generateInitState ( ) [protected]
```

Generates initial state of the model from GlobalModel in memory.

Returns

Returns a pointer to an initial GlobalState.

5.26.2.9 generateStateFromLocalStates()

```
GlobalState * GlobalModelGenerator::generateStateFromLocalStates (
    set < LocalState * > * localStates,
    set < LocalTransition * > * viaLocalTransitions,
    GlobalState * prevGlobalState ) [protected]
```

Creates a new GlobalState using some of the internally known model data and given local states, transitions that were uset to get there and the previous global state.

Parameters

localStates	LocalStates from which the new GlobalState will be built.
viaLocalTransitions	Pointer to a set of pointers to LocalTransition from which the changes in variables, as a result of traversing through the transition, will be made in a new GlobalState.
prevGlobalState	Pointer to GlobalState from which all persistent variables will be copied over from to the new GlobalState.

Returns

Returns a pointer to a new or already existing in the same epistemic class GlobalModel.

5.26.2.10 getCurrentGlobalModel()

```
GlobalModel * GlobalModelGenerator::getCurrentGlobalModel ( )
```

Get for a GlobalModel used in initialization.

Returns

Returns a pointer to a global model.

5.26.2.11 getFormula()

```
Formula * GlobalModelGenerator::getFormula ( )
```

Get for the Formula used in initialization.

Returns

Returns a pointer to the formula structure.

5.26.2.12 initModel()

Initializes a global model from local models and a formula.

Parameters

localModels	Pointer to LocalModels that will construct a global model.
formula	Pointer to a Formula to include into the model.

Returns

Returns a pointer to initial state of the global model.

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

- · GlobalModelGenerator.hpp
- GlobalModelGenerator.cpp

5.27 GlobalState Struct Reference

Represents a single global state.

```
#include <Types.hpp>
```

Public Attributes

int id

Identifier of the global state.

· string hash

Hash of the global state used in quick checks if the states are in the same epistemic class.

map< Var *, int > vars

Map of model variables and their current values.

map< Agent *, EpistemicClass * > epistemicClasses

Map of agents and the epistemic classes that belongs to the respective agent.

bool isExpanded

If false, the state can be still expanded, potentially creating new states, otherwise the expansion of the state already occured and is not necessary.

• GlobalStateVerificationStatus verificationStatus

Current verifivation status of this state.

set< GlobalTransition * > globalTransitions

Every GlobalTransition in the model.

• set< LocalState * > localStates

Local states of each agent that define this global state.

5.27.1 Detailed Description

Represents a single global state.

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

• Types.hpp

5.28 GlobalTransition Struct Reference

Represents a single global transition.

```
#include <Types.hpp>
```

Public Attributes

int id

Identifier of the transition.

• bool isInvalidDecision

Marks if the transition is invalid, true if there is no point in traversing that transition, otherwise false.

GlobalState * from

Binding to a GlobalState from which this transition goes from.

• GlobalState * to

Binding to a GlobalState from which this transition goes to.

set < LocalTransition * > localTransitions

Local transitions that define this global transition. A single transition or more in case of shared transitions.

5.28.1 Detailed Description

Represents a single global transition.

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

· Types.hpp

5.29 HistoryDbg Class Reference

Stores history and allows displaying it to the console.

```
#include <Verification.hpp>
```

Public Member Functions

· HistoryDbg ()

A constructor for HistoryDbg.

∼HistoryDbg ()

A destructor for History Dbg.

void addEntry (HistoryEntry *entry)

Adds a HistoryEntry to the debug history.

void markEntry (HistoryEntry *entry, char chr)

Marks an entry in the degug history with a char.

void print (string prefix)

Prints every entry from the algorithm's path.

HistoryEntry * cloneEntry (HistoryEntry *entry)

Checks if the HistoryEntry pointer exists in the debug history.

Public Attributes

vector< pair< HistoryEntry *, char >> entries
 A pair of history entries and a char marking history type.

5.29.1 Detailed Description

Stores history and allows displaying it to the console.

5.29.2 Member Function Documentation

5.29.2.1 addEntry()

Adds a HistoryEntry to the debug history.

Parameters

entry A pointer to the HistoryEntry that will be added to the history.

5.29.2.2 cloneEntry()

Checks if the HistoryEntry pointer exists in the debug history.

Parameters

Returns

Identity function if the entry is in history, otherwise returns nullptr.

5.29.2.3 markEntry()

Marks an entry in the degug history with a char.

Parameters

entry	A pointer to a HistoryEntry that is supposed to be marked.
chr	A char that will be made into a pair with a HistoryEntry.

5.29.2.4 print()

Prints every entry from the algorithm's path.

Parameters

prefix A prefix string to append to the front of every entry.

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

- · Verification.hpp
- · Verification.cpp

5.30 HistoryEntry Struct Reference

Structure used to save model traversal history.

```
#include <Verification.hpp>
```

Public Member Functions

• string toString ()

Converts HistoryEntry to string.

Public Attributes

• HistoryEntryType type

Type of the history record.

• GlobalState * globalState

Saved global state.

• GlobalTransition * decision

Selected transition.

· bool globalTransitionControlled

Is the transition controlled by an agent in coalition.

• GlobalStateVerificationStatus prevStatus

Previous model verification state.

GlobalStateVerificationStatus newStatus

Next model verification state.

· int depth

Recursion depth.

HistoryEntry * prev

Pointer to the previous HistoryEntry.

HistoryEntry * next

Pointer to the next HistoryEntry.

5.30.1 Detailed Description

Structure used to save model traversal history.

5.30.2 Member Function Documentation

5.30.2.1 toString()

```
string HistoryEntry::toString ( ) [inline]
```

Converts HistoryEntry to string.

Returns

A string with the descriprion of this history record.

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

Verification.hpp

5.31 LocalModels Struct Reference

Represents a single local model, contains all agents and variables.

```
#include <Types.hpp>
```

Public Attributes

vector< Agent * > agents

A vector of agents for the current model.

map< string, Var * > vars

A map of variable names to Var.

5.31.1 Detailed Description

Represents a single local model, contains all agents and variables.

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

· Types.hpp

5.32 LocalState Class Reference

Represents a single LocalState, containing id, name and internal variables.

```
#include <Types.hpp>
```

Public Member Functions

bool compare (LocalState *state)

Function comparing two states.

Public Attributes

int id

State identifier.

string name

State name.

map< Var *, int > vars

Local variables and their values.

• map< string, int > environment

Local variables as a name and their current values.

Agent * agent

Binding to an Agent.

set < LocalTransition * > localTransitions

Binding to the set of LocalTransition.

5.32.1 Detailed Description

Represents a single LocalState, containing id, name and internal variables.

5.32.2 Member Function Documentation

5.32.2.1 compare()

Function comparing two states.

Parameters

state A pointer to LocalState to which this state should be compared to.

Returns

Returns true if the current LocalState is the same as the passed one, otherwise false.

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

- Types.hpp
- Types.cc

5.33 LocalStateTemplate Class Reference

A template for the local state.

```
#include <nodes.hpp>
```

Public Attributes

· string name

Name of the local state.

set< TransitionTemplate * > transitions

Local transitions going out from this state.

5.33.1 Detailed Description

A template for the local state.

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

nodes.hpp

5.34 LocalTransition Struct Reference

Represents a single local transition, containing id, global name, local name, is shared and count of the appearances.

```
#include <Types.hpp>
```

Public Attributes

int id

Identifier of the transition.

string name

Name of the transition (global).

string localName

Name of the transition (local).

· bool isShared

Is the transition appearing somewhere else, true if yes, false if no.

· int sharedCount

Count of recurring appearances of this transition.

set < Condition * > conditions

Conditions that have to be fulfilled for the transition to be avaliable.

set < VarAssignment * > varAsssignments

Values to be set as a result of the traversal.

Agent * agent

Binding to an Agent.

LocalState * from

Binding to a LocalState from which this transition goes from.

LocalState * to

Binding to a LocalState from which this transition goes to.

set < LocalTransition * > sharedLocalTransitions

Stores shared transitions from different models.

5.34.1 Detailed Description

Represents a single local transition, containing id, global name, local name, is shared and count of the appearances.

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

· Types.hpp

5.35 SeleneFormula Class Reference

Public Member Functions

- virtual bool verifyLocalStates (set< LocalState * > *localStates)=0
- LocalState * getLocalStateForAgent (string agentName, set < LocalState * > *localStates)
- int getLocalStateVar (string varName, LocalState *localState)
- · bool implication (bool left, bool right)

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

- SeleneFormula.hpp
- · SeleneFormula.cpp

5.36 SeleneFormula1 Class Reference

Public Member Functions

bool verifyLocalStates (set< LocalState * > *localStates)

Public Member Functions inherited from SeleneFormula

- virtual bool verifyLocalStates (set< LocalState * > *localStates)=0
- LocalState * getLocalStateForAgent (string agentName, set< LocalState * > *localStates)
- int getLocalStateVar (string varName, LocalState *localState)
- bool implication (bool left, bool right)

5.36.1 Member Function Documentation

5.36.1.1 verifyLocalStates()

Implements SeleneFormula.

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

- · SeleneFormula.hpp
- · SeleneFormula.cpp

5.37 TestParser Class Reference

A parser for converting a text file into a model.

```
#include <TestParser.hpp>
```

Public Member Functions

• TestParser ()

TestParser constructor.

∼TestParser ()

TestParser destructor.

• LocalModels * parse (string fileName)

Parses a file with given name into a usable model.

5.37.1 Detailed Description

A parser for converting a text file into a model.

5.37.2 Member Function Documentation

5.37.2.1 parse()

Parses a file with given name into a usable model.

Parameters

e of the file to be converted into a	e of the file to be converted into a model.
--------------------------------------	---

Returns

Pointer to a model created from a given file.

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

- · TestParser.hpp
- TestParser.cc

5.38 TransitionTemplate Class Reference

Represents a meta-transition.

```
#include <nodes.hpp>
```

Public Member Functions

TransitionTemplate (int _shared, string _patternName, string _matchName, string _startState, string _end
 State, ExprNode *_cond, set< Assignment * > *_assign)

TransitionTemplate constructor.

Public Attributes

· int shared

Needed amound of needed agents. -1 if not shared.

string patternName

Name of the pattern.

• string matchName

Global name for shared transitions.

· string startState

Start state name.

· string endState

End state name.

• ExprNode * condition

Condition expression that has do be fulfilled in that transition.

set < Assignment * > * assignments

Set of assignments.

5.38.1 Detailed Description

Represents a meta-transition.

5.38.2 Constructor & Destructor Documentation

5.38.2.1 TransitionTemplate()

```
TransitionTemplate::TransitionTemplate (
    int _shared,
    string _patternName,
    string _matchName,
    string _startState,
    string _endState,
    ExprNode * _cond,
    set< Assignment * > * _assign ) [inline]
```

TransitionTemplate constructor.

Parameters

_shared	Needed amound of needed agents1 if not shared.
_patternName	Name of the pattern.
_matchName	Global name for shared transitions.
_startState	Start state name.
_endState	End state name.
_cond	Condition expression that has do be fulfilled in that transition.
_assign	Set of assignments.

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

· nodes.hpp

5.39 Var Struct Reference

Represents a variable in the model, containing name, initial value and persistence.

```
#include <Types.hpp>
```

Public Attributes

· string name

Variable name.

• int initialValue

Initial value of the variable.

· bool persistent

True if variable is persistent, i.e. it should appear in all states in the model, false otherwise.

· Agent * agent

Reference to an agent, to which this variable belongs to.

5.39.1 Detailed Description

Represents a variable in the model, containing name, initial value and persistence.

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

• Types.hpp

5.40 VarAssignment Struct Reference

Public Attributes

- Var * dstVar
- VarAssignmentType type
- Var * srcVar
- int value

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

Types.hpp

5.41 Verification Class Reference

A class that verifies if the model fulfills the formula. Also can do some operations on decision history.

#include <Verification.hpp>

Public Member Functions

Verification (GlobalModelGenerator *generator)

Constructor for Verification.

∼Verification ()

Destructor for Verification.

· bool verify ()

Starts the process of formula verification on a model.

Protected Member Functions

bool verifyLocalStates (set< LocalState * > *localStates)

Verifies a set of LocalState that a GlobalState is composed of with a hardcoded formula.

• bool verifyGlobalState (GlobalState *globalState, int depth)

Recursively verifies GlobalState.

• bool isGlobalTransitionControlledByCoalition (GlobalTransition *globalTransition)

Checks if any of the LocalTransition in a given GlobalTransition has an Agent in a coalition in the formula.

bool isAgentInCoalition (Agent *agent)

Checks if the Agent is in a coalition based on the formula in a GlobalModelGenerator.

• EpistemicClass * getEpistemicClassForGlobalState (GlobalState *globalState)

Gets the EpistemicClass for the agent in passed GlobalState, i.e. transitions from indistinguishable state from certain other states for an agent to other states.

• bool areGlobalStatesInTheSameEpistemicClass (GlobalState *globalState1, GlobalState *globalState2)

Compares two GlobalState and checks if their EpistemicClass is the same.

• void addHistoryDecision (GlobalState *globalState, GlobalTransition *ecision)

Creates a HistoryEntry of the type DECISION and puts it on top of the stack of the decision history.

void addHistoryStateStatus (GlobalState *globalState, GlobalStateVerificationStatus prevStatus, GlobalStateVerificationStatus newStatus)

Creates a HistoryEntry of the type STATE_STATUS and puts it to the top of the decision history.

Creates a HistoryEntry of the type CONTEXT and puts it to the top of the decision history.

void addHistoryMarkDecisionAsInvalid (GlobalState *globalState, GlobalTransition *decision)

Creates a HistoryEntry of the type MARK_DECISION_AS_INVALID and puts it to the top of the decision history.

• HistoryEntry * newHistoryMarkDecisionAsInvalid (GlobalState *globalState, GlobalTransition *decision)

Creates a HistoryEntry of the type MARK_DECISION_AS_INVALID and returns it.

bool revertLastDecision (int depth)

Reverts GlobalState and history to the previous decision state.

void undoLastHistoryEntry (bool freeMemory)

Removes the top entry of the history stack.

void undoHistoryUntil (HistoryEntry *historyEntry, bool inclusive, int depth)

Rolls back the history entries up to the certain HistoryEntry.

void printCurrentHistory (int depth)

Prints current history to the console.

Protected Attributes

Mode mode

Current mode of model traversal.

GlobalState * revertToGlobalState

Global state to which revert will rollback to.

stack< HistoryEntry * > historyToRestore

A history of decisions to be rolled back.

• GlobalModelGenerator * generator

Holds current model and formula.

• SeleneFormula * seleneFormula

Temporary solve for data input.

• HistoryEntry * historyStart

Pointer to the start of model traversal history.

HistoryEntry * historyEnd

Pointer to the end of model traversal history.

5.41.1 Detailed Description

A class that verifies if the model fulfills the formula. Also can do some operations on decision history.

5.41.2 Constructor & Destructor Documentation

5.41.2.1 Verification()

Constructor for Verification.

Parameters

```
generator Pointer to GlobalModelGenerator
```

5.41.3 Member Function Documentation

5.41.3.1 addHistoryContext()

```
GlobalTransition * decision,
bool globalTransitionControlled ) [protected]
```

Creates a HistoryEntry of the type CONTEXT and puts it to the top of the decision history.

Parameters

globalState	Pointer to a GlobalState of the model.
depth	Depth of the recursion of the validation algorithm.
decision	Pointer to a transition GlobalTransition selected by the algorithm.
globalTransitionControlled	True if the GlobalTransition is in the set of global transitions controlled by a coalition and it is not a fixed global transition.

5.41.3.2 addHistoryDecision()

Creates a HistoryEntry of the type DECISION and puts it on top of the stack of the decision history.

Parameters

globalState	Pointer to a GlobalState of the model.
decision	Pointer to a GlobalTransition that is to be recorded in the decision history.

5.41.3.3 addHistoryMarkDecisionAsInvalid()

Creates a HistoryEntry of the type MARK_DECISION_AS_INVALID and puts it to the top of the decision history.

Parameters

global	State Po	inter to a GlobalState of the model.
decisio	on Po	inter to a transition GlobalTransition selected by the algorithm.

5.41.3.4 addHistoryStateStatus()

```
GlobalStateVerificationStatus prevStatus,
GlobalStateVerificationStatus newStatus ) [protected]
```

Creates a HistoryEntry of the type STATE_STATUS and puts it to the top of the decision history.

Parameters

globalState	Pointer to a GlobalState of the model.
prevStatus	Previous GlobalStateVerificationStatus to be logged.
newStatus	New GlobalStateVerificationStatus to be logged.

5.41.3.5 areGlobalStatesInTheSameEpistemicClass()

Compares two GlobalState and checks if their EpistemicClass is the same.

Parameters

globalState1	Pointer to the first GlobalState.
globalState2	Pointer to the second GlobalState.

Returns

Returns true if the EpistemicClass is the same for both of the GlobalState. Returns false if they are different or at least one of them has no EpistemicClass.

5.41.3.6 getEpistemicClassForGlobalState()

Gets the EpistemicClass for the agent in passed GlobalState, i.e. transitions from indistinguishable state from certain other states for an agent to other states.

Parameters

globalState	Pointer to a GlobalState of the model.
-------------	--

Returns

Pointer to the EpistemicClass that a coalition of agents from the formula belong to. If there is no such EpistemicClass, returns false.

5.41.3.7 isAgentInCoalition()

Checks if the Agent is in a coalition based on the formula in a GlobalModelGenerator.

Parameters

agent Pointer to an Agent that is to be checked.

Returns

Returns true if the Agent is in a coalition, otherwise returns false.

5.41.3.8 isGlobalTransitionControlledByCoalition()

```
\begin{tabular}{ll} bool Verification:: is Global Transition Controlled By Coalition ( \\ Global Transition * global Transition ) & [protected] \end{tabular}
```

Checks if any of the LocalTransition in a given GlobalTransition has an Agent in a coalition in the formula.

Parameters

globalTransition	Pointer to a GlobalTransition in a model.
------------------	---

Returns

Returns true if the Agent is in coalition in the formula, otherwise returns false.

5.41.3.9 newHistoryMarkDecisionAsInvalid()

Creates a HistoryEntry of the type MARK_DECISION_AS_INVALID and returns it.

Parameters

globalState	Pointer to a GlobalState of the model.
decision	Pointer to a transition GlobalTransition selected by the algorithm.

64 Class Documentation

Returns

Returns pointer to a new HistoryEntry.

5.41.3.10 printCurrentHistory()

Prints current history to the console.

Parameters

depth	Integer that will be multiplied by 4 and appended as a prefix to the optional debug log.
	The state of the s

5.41.3.11 revertLastDecision()

Reverts GlobalState and history to the previous decision state.

Parameters

depth Integer that will be multiplied by 4 and appended as a prefix to the optional debug log.

Returns

Returns true if rollback is successful, otherwise returns false.

5.41.3.12 undoHistoryUntil()

Rolls back the history entries up to the certain HistoryEntry.

Parameters

historyEntry	Pointer to a HistoryEntry that the history has to be rolled back to.
inclusive	True if the rollback has to remove the specified entry too.
depth	Integer that will be multiplied by 4 and appended as a prefix to the optional debug log.

5.41.3.13 undoLastHistoryEntry()

Removes the top entry of the history stack.

Parameters

freeMemory True if the entry has to be removed from mem-	ory.
--	------

5.41.3.14 verify()

```
bool Verification::verify ( )
```

Starts the process of formula verification on a model.

Returns

Returns true if the verification is PENDING or VERIFIED_OK, otherwise returns false.

5.41.3.15 verifyGlobalState()

Recursively verifies GlobalState.

Parameters

globalState	Pointer to a GlobalState of the model.
depth	Current depth of the recursion.

Returns

Returns true if the verification is PENDING or VERIFIED_OK, otherwise returns false.

66 Class Documentation

5.41.3.16 verifyLocalStates()

```
bool Verification::verifyLocalStates ( set < LocalState * > * localStates ) \quad [protected]
```

Verifies a set of LocalState that a GlobalState is composed of with a hardcoded formula.

Parameters

localStates	A pointer to a set of pointers to LocalState.
-------------	---

Returns

Returns true if there is a LocalState with a specific set of values, fulfilling the criteria, otherwise returns false.

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

- · Verification.hpp
- · Verification.cpp

Chapter 6

File Documentation

6.1 Constants.hpp

```
00001 #ifndef SELENE_CONSTANTS
00002 #define SELENE_CONSTANTS
00003
00004 #define VERBOSE 0
00005 // #define OUTPUT_LOCAL_MODELS 1
00006 // #define OUTPUT_GLOBAL_MODEL 0 // warning: it will call expandAllStates()
00007 // \#define MODE 2 // 1 = only generate; 2 = verify
80000
00009 // Model id
00010 // 1 = src/examples/trains/Trains.txt
00011 // 2 = src/examples/ssvr/Selene_Select_Vote_Revoting_1v_1cv_3c_3rev_share.txt
00012 // 3 = src/examples/svote/Simple_voting.txt
00013 // #define MODEL_ID 1
00014
00015 struct Cfg{
        char* fname;
char stv_mode;
00016
00018
          bool output_local_models;
         bool output_global_model; int model_id; // <-- this is temporary member (used in Verification.cpp for a hardcoded formula)
00019
00020
00021 };
00022
00023 #endif // SELENE_CONSTANTS
```

6.2 GlobalModelGenerator.cpp File Reference

Generator of a global model. Class for initializing and generating a global model.

```
#include "GlobalModelGenerator.hpp"
#include <iostream>
```

6.2.1 Detailed Description

Generator of a global model. Class for initializing and generating a global model.

6.3 GlobalModelGenerator.hpp File Reference

Generator of a global model. Class for initializing and generating a global model.

```
#include "Constants.hpp"
#include "Types.hpp"
```

Classes

· class GlobalModelGenerator

Stores the local models, formula and a global model.

6.3.1 Detailed Description

Generator of a global model. Class for initializing and generating a global model.

6.4 GlobalModelGenerator.hpp

Go to the documentation of this file.

```
00007 #ifndef SELENE_GLOBAL_MODEL_GENERATOR
00008 #define SELENE_GLOBAL_MODEL_GENERATOR
00009
00010 #include "Constants.hpp"
00011 #include "Types.hpp"
00013 using namespace std;
00014
00016 class GlobalModelGenerator {
00017 public:
00018
                          GlobalModelGenerator();
00019
                            ~GlobalModelGenerator();
00020
                           GlobalState* initModel(LocalModels* localModels, Formula* formula);
00021
                          void expandState(GlobalState* state);
00022
                           void expandAllStates();
                          GlobalModel* getCurrentGlobalModel();
00023
00024
                          Formula* getFormula();
00026 protected:
00028
                          LocalModels* localModels;
00030
                          Formula* formula:
00032
                           GlobalModel* globalModel;
00033
                           GlobalState* generateInitState();
00034
                           GlobalState* generateStateFromLocalStates(set<LocalState*>* localStates, set<LocalTransition*>*
               viaLocalTransitions, GlobalState* prevGlobalState);
00035
                           void generateGlobalTransitions(GlobalState* fromGlobalState, set<LocalTransition*>
               localTransitions, map<Agent*, vector<LocalTransition*» transitionsByAgent);
bool checkLocalTransitionConditions(LocalTransition* localTransition, GlobalState* globalState);
string computeEpistemicClassHash(set<LocalState*>* localStates, Agent* agent);
00036
00037
                           string computeGlobalStateHash(set<LocalState*>* localStates);
00039
                           EpistemicClass* findOrCreateEpistemicClass(set<LocalState*>* localStates, Agent* agent);
00040
                           {\tt GlobalState*} \  \, {\tt findGlobalStateInEpistemicClass} \\ ({\tt set<LocalState*>*} \  \, {\tt localStates}, \  \, {\tt EpistemicClass*} \\ ({\tt set<LocalState*>*} \  \, {\tt localStates}, \  \, {\tt EpistemicClass*} \\ ({\tt epistemicClass*}, \  \, {\tt epistemicClass*}) \\ ({\tt epistemicClass*}, \  \, {\tt epistemicC
                epistemicClass);
00041 };
00042
00043 #endif // SELENE_GLOBAL_MODEL_GENERATOR
```

6.5 expressions.cc File Reference

Eval and helper class for expressions. Eval and helper class for expressions.

```
#include "expressions.hpp"
```

6.5.1 Detailed Description

Eval and helper class for expressions. Eval and helper class for expressions.

6.6 expressions.hpp File Reference

Eval and helper class for expressions. Eval and helper class for expressions.

```
#include <string>
#include <map>
```

Classes

class ExprNode

Base node for expressions.

class ExprConst

Node for a constant.

class ExprIdent

Node for an identifier.

· class ExprAdd

Node for addition.

class ExprSub

Node for subtraction.

class ExprMul

Node for multiplication.

class ExprDiv

Node for division.

• class ExprRem

Node for modulo.

class ExprAnd

Node for AND operator.

· class ExprOr

Node for OR operator.

class ExprNot

Node for NOT operator.

class ExprEq

Node for "==" operator.

• class ExprNe

Node for "!=" operator.

class ExprLt

Node for "<" operator.

• class ExprLe

Node for "<=" operator.

class ExprGt

Node for ">" operator.

• class ExprGe

Node for ">=" operator.

Typedefs

• typedef map< string, int > Environment

Variable names with their values.

6.6.1 Detailed Description

Eval and helper class for expressions. Eval and helper class for expressions.

6.7 expressions.hpp

Go to the documentation of this file.

```
00001
00007 #ifndef ___EXPRESSIONS_H
00008 #define __EXPRESSIONS_H
00009
00010 #include <string>
00011 #include <map>
00012
00013 using namespace std;
00014
00016 typedef map<string, int> Environment;
00018 // węzeł bazowy dla wyrażeń
00019
00021 class ExprNode {
00022
00023
         public:
00024
            // metoda do wyliczenia wartości wyrażenia zależna od typu węzła
00025
00029
            virtual int eval( Environment& env ) = 0;
00030 };
00031
00032 // węzeł dla stałej
00033
00035 class ExprConst: public ExprNode {
00036
00037
         // argumenty
00038
00040
         int val;
00041
00042
         public:
00045
            ExprConst(int _val): val(_val) {};
00046
00050
            virtual int eval( Environment& env );
00051 };
00052
00053 // węzeł dla identyfikatora
00054
00056 class ExprIdent: public ExprNode {
00057
00058
         // argumenty
00059
00061
         string ident;
00062
00063
         public:
00066
            ExprIdent(string _ident): ident(_ident) {};
00067
00071
            virtual int eval( Environment& env );
00072 };
00073
00074 // węzeł dla dodawań
00075
00077 class ExprAdd: public ExprNode {
00078
00079
         // argumenty
00080
00082
         ExprNode *larg, *rarg;
00083
00084
         public:
00088
            ExprAdd(ExprNode *_larg, ExprNode *_rarg): larg(_larg), rarg(_rarg) {};
00089
00093
            virtual int eval( Environment& env );
00094 };
00095
00096 // węzeł dla odejmowań
00097
00099 class ExprSub: public ExprNode {
00100
00101
         // argumenty
00102
00104
         ExprNode *larg, *rarg;
00105
00106
         public:
```

6.7 expressions.hpp 71

```
00110
            ExprSub(ExprNode *_larg, ExprNode *_rarg): larg(_larg), rarg(_rarg) {};
00111
00115
            virtual int eval( Environment& env );
00116 };
00117
00118 // wezeł dla mnożeń
00119
00121 class ExprMul: public ExprNode {
00122
00123
         // argumenty
00124
00126
         ExprNode *larg, *rarg;
00127
00128
         public:
00132
            ExprMul(ExprNode *_larg, ExprNode *_rarg): larg(_larg), rarg(_rarg) {};
00133
            virtual int eval( Environment& env );
00137
00138 };
00139
00140 // węzeł dla dzieleń
00141
00143 class ExprDiv: public ExprNode {
00144
00145
         // argumenty
00146
00148
         ExprNode *larg, *rarg;
00149
00150
         public:
00154
            ExprDiv(ExprNode *_larg, ExprNode *_rarg): larg(_larg), rarg(_rarg) {};
00155
00159
            virtual int eval( Environment& env );
00160 };
00161
00162 // węzeł dla reszty z dzielenia
00163
00165 class ExprRem: public ExprNode {
00166
00167
         // argumenty
00168
00170
         ExprNode *larg, *rarg;
00171
         public:
00172
00176
            ExprRem(ExprNode *_larg, ExprNode *_rarg): larg(_larg), rarg(_rarg) {};
00177
00181
            virtual int eval( Environment& env );
00182 };
00183
00184 // węzeł dla operatora AND
00185
00187 class ExprAnd: public ExprNode {
00188
00189
         // argumenty
00190
00192
         ExprNode *larg, *rarg;
00193
00194
         public:
00198
            ExprAnd(ExprNode *_larg, ExprNode *_rarg): larg(_larg), rarg(_rarg) {};
00199
00203
            virtual int eval( Environment& env );
00204 };
00205
00206 // węzeł dla operatora OR
00207
00209 class ExprOr: public ExprNode {
00210
00211
         // argumenty
00212
00214
         ExprNode *larg, *rarg;
00215
00216
         public:
00220
           ExprOr(ExprNode *_larg, ExprNode *_rarg): larg(_larg), rarg(_rarg) {};
00221
00225
            virtual int eval( Environment& env );
00226 };
00227
00228 // węzeł dla operatora NOT
00229
00231 class ExprNot: public ExprNode {
00232
00233
         // argumenty
00234
00236
         ExprNode *arg;
00237
         public:
00238
00241
            ExprNot (ExprNode *_arg): arg(_arg) {};
00242
00246
            virtual int eval( Environment& env );
```

```
00247 };
00248
00249 // węzeł dla operatora "=="
00250
00252 class ExprEq: public ExprNode {
00253
00254
         // argumenty
00255
00257
         ExprNode *larg, *rarg;
00258
00259
         public:
           ExprEq(ExprNode *_larg, ExprNode *_rarg): larg(_larg), rarg(_rarg) {};
00263
00264
00268
            virtual int eval( Environment& env );
00269 };
00270
00271 // węzeł dla operatora "!="
00272
00274 class ExprNe: public ExprNode {
00275
00276
         // argumenty
00277
00279
         ExprNode *larg, *rarg;
00280
00281
         public:
00285
           ExprNe(ExprNode *_larg, ExprNode *_rarg): larg(_larg), rarg(_rarg) {};
00286
00290
            virtual int eval( Environment& env );
00291 };
00292
00293 // wezeł dla operatora "<"
00294
00296 class ExprLt: public ExprNode {
00297
00298
         // argumenty
00299
00301
         ExprNode *larg, *rarg;
00302
00303
00307
           ExprLt(ExprNode *_larg, ExprNode *_rarg): larg(_larg), rarg(_rarg) {};
00308
00312
            virtual int eval( Environment& env );
00313 };
00314
00315 // węzeł dla operatora "<="
00316
00318 class ExprLe: public ExprNode {
00319
00320
         // argumenty
00321
00323
         ExprNode *larg, *rarg;
00324
00325
         public:
00329
            ExprLe(ExprNode *_larg, ExprNode *_rarg): larg(_larg), rarg(_rarg) {};
00330
00334
            virtual int eval( Environment& env );
00335 };
00336
00337 // węzeł dla operatora ">"
00338
00340 class ExprGt: public ExprNode {
00341
00342
         // argumenty
00343
00345
         ExprNode *larg, *rarg;
00346
00347
         public:
00351
            ExprGt(ExprNode *_larg, ExprNode *_rarg): larg(_larg), rarg(_rarg) {};
00352
00356
            virtual int eval( Environment& env );
00357 };
00358
00359 // węzeł dla operatora ">="
00360
00362 class ExprGe: public ExprNode {
00363
00364
         // argumenty
00365
00367
         ExprNode *larg, *rarg;
00368
00369
         public:
00373
            ExprGe(ExprNode *_larg, ExprNode *_rarg): larg(_larg), rarg(_rarg) {};
00374
00378
            virtual int eval( Environment& env );
00379 };
00380
00381
```

73

```
00382 #endif
```

6.8 nodes.cc File Reference

Parser templates. Class for setting up a new objects from a parser.

```
#include "expressions.hpp"
#include "nodes.hpp"
#include <queue>
```

6.8.1 Detailed Description

Parser templates. Class for setting up a new objects from a parser.

6.9 nodes.hpp File Reference

Parser templates. Class for setting up a new objects from a parser.

```
#include <string>
#include <set>
#include <map>
#include "expressions.hpp"
#include "../Types.hpp"
```

Classes

· class Assignment

Represents an assingment.

class TransitionTemplate

Represents a meta-transition.

class LocalStateTemplate

A template for the local state.

class AgentTemplate

Represents a single agent loaded from the description from a file.

6.9.1 Detailed Description

Parser templates. Class for setting up a new objects from a parser.

6.10 nodes.hpp

Go to the documentation of this file.

```
00001
00007 #ifndef __NODES_H
00008 #define __NODES_H
00009
00010 #include <string>
00011 #include <set>
00012 #include <map>
00013 #include "expressions.hpp"
00014
00015 #include "../Types.hpp"
00016
00017 using namespace std;
00018
00019 /* Klasa reprezentująca przypisanie */
00020
00022 class Assignment {
00023
        public:
00025
           string ident;
00026
            // do czego przypisujemy
00027
00029
           ExprNode *value;
00030
           // co przypisujemy
00031
00035
           Assignment(string _ident, ExprNode *_exp): ident(_ident), value(_exp) {};
00036
00037
           // wykonaj przypisanie w danym środowisku
00038
00041
           virtual void assign(Environment &env) {
00042
              env[ident]=value->eval(env);
00043
00044 };
00045
00046 /* Klasa reprezentująca meta-tranzycję */
00047
00049 class TransitionTemplate {
00050
00051
           // jeśli -1 to nie ma dzielenia, w p.p. wartość określa łączną liczbę wymaganych agentów
00052
00054
           int shared:
00055
00056
           // nazwa wzorca
00057
00059
           string patternName;
00060
00061
           // nazwa do wyszukiwania dla shared
00062
00064
           string matchName;
00065
00066
           // nazwa stanu początkowego i końcowego
00067
00069
           string startState;
00070
00072
           string endState:
00073
00074
           // wyrażenie warunkowe
00075
00077
           ExprNode *condition;
00078
00079
           // lista przypisań wartości
00082
           set<Assignment*> *assignments;
00083
00092
           TransitionTemplate(int _shared, string _patternName, string _matchName, string _startState,
     00093
00094
                 startState(_startState), endState(_endState), condition(_cond), assignments(_assign) {};
00095
00096 };
00097
00099 class LocalStateTemplate {
        public:
00100
00102
           string name:
00103
00105
           set<TransitionTemplate*> transitions;
00106 };
00107
00108 /* Klasa reprezentująca pojedynczego agenta po wczytaniu jego opisu z pliku */
00109
00111 class AgentTemplate {
00112
           // identyfikator agenta
00113
00115
            string ident;
```

```
00116
00117
            // stan startowy
00118
00120
            string initState;
00121
00122
            // zbiór zmiennych lokalnych (local)
00123
00125
            set<string>* localVars;
00126
00127
            // zbiór zmiennych trwałych (persistent)
00128
            set<string>* persistentVars;
00130
00131
00132
            // początkowa inicjacja
00133
00135
            set<Assignment*>* initialAssignments;
00136
00137
            // zbiór tranzvcji
00138
00140
            set<TransitionTemplate*>* transitions;
00141
00142
            // mapa stanów lokalnych potrzebna do wygenerowania modelu
00143
00145
            map<string,LocalStateTemplate*> localStateTemplates;
00146
00147
            // metoda wyznaczająca węzeł kolejny do danego, zależnie od tranzycji
00148
00149
            virtual LocalState* genNextState(LocalState *state, TransitionTemplate *trans);
00150
00151
         public:
00152
            AgentTemplate();
00153
00154
            // ustaw identyfikator agenta
00155
00159
            virtual AgentTemplate& setIdent(string _ident);
00160
00161
            // ustaw identyfikator agenta
00162
00166
            virtual AgentTemplate& setInitState(string _startState);
00167
00168
            // dodaj zmienną/zmienne lokalne
00169
00173
            virtual AgentTemplate& addLocal(set<string> *variables);
00174
00175
            // dodaj zmienne trwałe
00176
00180
            virtual AgentTemplate& addPersistent(set<string> *variables);
00181
00182
            // dodaj początkowe inicjacje
00183
00187
            virtual AgentTemplate& addInitial(set<Assignment*> *assigns);
00188
00189
            // dodaj tranzycję
00190
            virtual AgentTemplate& addTransition(TransitionTemplate *_transition);
00194
00195
00196
            // wygeneruj agenta do modelu
00197
00201
            virtual Agent* generateAgent(int id) ;
00202 };
00203
00204 #endif
```

6.11 SeleneFormula.hpp

```
00001 #ifndef SELENE_SELENE_FORMULA
00002 #define SELENE_SELENE_FORMULA
00003
00004 #include "Types.hpp"
00005
00006
00007
00008
00009
00010 class SeleneFormula {
00011 public:
00012
          SeleneFormula();
          ~SeleneFormula();
00013
00014
          virtual bool verifyLocalStates(set<LocalState*>* localStates) = 0;
00015
          LocalState* getLocalStateForAgent(string agentName, set<LocalState*>* localStates);
00016
          int getLocalStateVar(string varName, LocalState* localState);
00017
          inline bool implication (bool left, bool right);
00018 };
```

```
00020
00021
00022
00023
00024 class SeleneFormula1 : public SeleneFormula {
00025 public:
00026
         SeleneFormula1();
00027
          ~SeleneFormula1();
        bool verifyLocalStates(set<LocalState*>* localStates);
00028
00029 protected:
00030 };
00031
00032
00033
00034
00035
00036 #endif // SELENE_SELENE_FORMULA
```

6.12 TestParser.hpp

```
00001
00007 #ifndef ___TESTPARSER_HPP
00008 #define __TESTPARSER_HPP
00010 #include "Types.hpp"
00011
00012 using namespace std;
00013
00015 class TestParser {
00016 public:
00017 TestParser();
00018 ~TestParser();
00019
        LocalModels* parse(string fileName);
00020
00021 protected:
00022
          // @internal
00023 };
00024
00025 #endif
```

6.13 Types.cc File Reference

Custom data structures. Data structures and classes containing model data.

```
#include "Types.hpp"
```

6.13.1 Detailed Description

Custom data structures. Data structures and classes containing model data.

6.14 Types.hpp File Reference

Custom data structures. Data structures and classes containing model data.

```
#include <map>
#include <set>
#include <stack>
#include <string>
#include <utility>
#include <vector>
```

Classes

struct Var

Represents a variable in the model, containing name, initial value and persistence.

• struct Condition

Represents a condition for LocalTransition.

struct Formula

Contains a coalition of Agent from the formula.

class Agent

Contains all data for a single Agent, including id, name and all of the agents' variables.

· class LocalState

Represents a single LocalState, containing id, name and internal variables.

- struct VarAssignment
- struct LocalTransition

Represents a single local transition, containing id, global name, local name, is shared and count of the appearances.

struct LocalModels

Represents a single local model, contains all agents and variables.

struct GlobalModel

Represents a global model, containing agents and a formula.

struct GlobalState

Represents a single global state.

· struct GlobalTransition

Represents a single global transition.

struct EpistemicClass

Represents a single epistemic class.

Enumerations

enum ConditionOperator { Equals , NotEquals }

Conditional operator for the variable.

 $\bullet \ \ \text{enum GlobalStateVerificationStatus \{ \ UNVERIFIED\ ,\ PENDING\ ,\ VERIFIED_OK\ ,\ VERIFIED_ERR\ \} }$

Verification status of a GlobalState.

enum VarAssignmentType { FromVar , FromValue }

Handles if the Var value is from srcVar or from value.

6.14.1 Detailed Description

Custom data structures. Data structures and classes containing model data.

6.14.2 Enumeration Type Documentation

6.14.2.1 ConditionOperator

enum ConditionOperator

Conditional operator for the variable.

Enumerator

Equals	Variable should be equal to the value.
NotEquals	Variable should be not equal to the value.

6.14.2.2 GlobalStateVerificationStatus

enum GlobalStateVerificationStatus

Verification status of a GlobalState.

Enumerator

UNVERIFIED	State is not verified.
PENDING	Entered the state but it is not verified as correct or incorrect yet.
VERIFIED_OK	The state has been verified and is correct.

6.14.2.3 VarAssignmentType

enum VarAssignmentType

Handles if the Var value is from srcVar or from value.

Enumerator

FromVar	Take value from srcVar.
FromValue	Take value from value.

6.15 Types.hpp

Go to the documentation of this file.

```
00001
00007 #ifndef SELENE_TYPES
00008 #define SELENE_TYPES
00009
00010 #include <map>
00011 #include <set>
00012
00013 #include <stack>
00014 #include <string>
00015 #include <utility>
00016 #include <vector>
00017
00018 using namespace std;
00019
00020 class Agent;
00022 class LocalState;
00023
```

6.15 Types.hpp 79

```
00024 struct Condition;
00025 struct EpistemicClass;
00026 struct Formula;
00027 struct GlobalModel;
00028 struct GlobalState;
00029 struct GlobalTransition;
00030 struct LocalTransition;
00031 struct LocalModels;
00032 struct Var;
00033 struct VarAssignment;
00034
00036 enum ConditionOperator {
00037
          Equals,
00038
          NotEquals,
00039 };
00040
00042 enum GlobalStateVerificationStatus {
00043
          UNVERIFIED,
00044
          PENDING,
00045
          VERIFIED_OK,
00046
          VERIFIED_ERR,
00047 };
00048
00050 enum VarAssignmentType {
00051
          FromVar,
00052
          FromValue,
00053 };
00054
00056 struct Var {
00058
          string name;
00059
00061
          int initialValue;
00062
00064
          bool persistent;
00065
00067
          Agent *agent;
00068 };
00069
00071 struct Condition {
00073
          Var* var;
00074
00076
          ConditionOperator conditionOperator;
00077
00079
          int comparedValue;
00080 };
00081
00083 struct Formula {
00085
          set<Agent*> coalition;
00086 };
00087
00089 class Agent {
00090
         public:
00092
              int id;
00093
00095
              string name;
00096
00098
              set<Var*> vars;
00099
00103
              Agent(int _id, string _name):id(_id), name(_name) {};
00104
00106
              LocalState* initState:
00107
00109
              vector<LocalState*> localStates; // localStates[i].id == i
00110
00112
              vector<LocalTransition*> localTransitions; // localTransitions[i].id == i
00113
00114
              // sprawdź, czy stan nie został już wygenerowany.
00115
00116
              LocalState* includesState(LocalState *state);
00117 };
00118
00120 class LocalState {
00121
         public:
             // Data
00122
00123
00125
              int id;
00126
00128
              string name;
00129
00131
              map<Var*, int> vars;
00132
00133
              // alternatywna wersja - może wystarczy
00134
00136
              map<string, int> environment;
00137
              // komparator
00138
00139
```

```
bool compare(LocalState *state);
00141
00142
               // Bindings
00143
00145
               Agent * agent;
00146
00148
               set<LocalTransition*> localTransitions;
00149 };
00150
00151 // to jest zbędne
00152 struct VarAssignment {
          Var* dstVar;
00153
          VarAssignmentType type; // zbędne
00154
00155
          Var* srcVar;
00156
          int value;
00157 };
00158
00160 struct LocalTransition {
00161 // Data
00162
00164
          int id;
00165
          string name;
00167
00168
00170
          string localName;
00171
          // if empty => same as name
00172
00174
          bool isShared;
00175
00177
          int sharedCount:
00178
00180
          set < Condition *> conditions;
00181
00183
          set<VarAssignment*> varAsssignments;
00184
00185
          // Bindings
00186
00188
          Agent* agent;
00189
00191
          LocalState* from;
00192
00194
          LocalState* to;
00195
00197
          set<LocalTransition*> sharedLocalTransitions;
00198 };
00199
00201 struct LocalModels {
00203
          vector<Agent*> agents;
00204
          // agents[i].id == i
00205
00207
          map<string, Var*> vars;
00208
          // vars[str].name == str
00209 };
00210
00212 struct GlobalModel {
00213
          // Data
00214
00216
          vector<Agent*> agents;
00217
          // agents[i].id == i
00218
00220
          Formula* formula;
00221
00222
          // Bindings
00223
00225
          GlobalState* initState;
00226
          vector<GlobalState*> globalStates;
00228
          // globalStates[i].id == i
00229
00230
00232
          vector<GlobalTransition*> globalTransitions;
00233
          // globalTransitions[i].id == i
00234
          map<Agent*, map<string, EpistemicClass*» epistemicClasses;
// Agent* => (EpistemicClass->hash => EpistemicClass*)
00236
00237
00238 };
00239
00241 struct GlobalState {
00242
          // Data
00243
00245
          int id:
00246
00248
          string hash;
00249
00251
          map<Var*, int> vars;
00252
00254
          map<Agent*, EpistemicClass*> epistemicClasses;
00255
```

```
00257
          bool isExpanded;
00258
00260
          GlobalStateVerificationStatus verificationStatus;
00261
00262
          // Bindings
00263
00265
          set<GlobalTransition*> globalTransitions;
00266
00268
          set < LocalState* > localStates;
00269 };
00270
00272 struct GlobalTransition {
00273
          // Data
00274
00276
          int id;
00277
00279
          bool isInvalidDecision:
00280
00281
          // Bindings
00282
00284
          GlobalState* from;
00285
00287
          GlobalState* to;
00288
00290
          set<LocalTransition*> localTransitions;
00291 };
00292
00294 struct EpistemicClass {
00296
          string hash;
00297
00299
          map<string, GlobalState*> globalStates;
00300
          // GlobalState->hash => GlobalState*
00301
00303
          GlobalTransition* fixedCoalitionTransition;
00304 };
00305
00306 #endif // SELENE_TYPES
```

6.16 Utils.cpp File Reference

Utility functions. A collection of utility functions to use in the project.

```
#include "Utils.hpp"
```

Functions

- string envToString (map< string, int > env)
 - Converts a map of string and int to a string.
- string agentToString (Agent *agt)

Converts pointer to an Agant into a string containing name of the agent, its initial state, transitions with their local and global names, shared count and conditions.

- string localModelsToString (LocalModels *Im)
 - Converts pointer to the LocalModels into a string cointaining all Agent instances from the model, initial values of the variables and names of the persistent values.
- void outputGlobalModel (GlobalModel *globalModel)

Prints the whole GlobalModel into the console. Contains global states with hashes, local states, variables inside those states, global variables, global transitions, local transitions and epistemic classes of agents.

• unsigned long getMemCap ()

Variables

· Cfg config

6.16.1 Detailed Description

Utility functions. A collection of utility functions to use in the project.

6.16.2 Function Documentation

6.16.2.1 agentToString()

```
string agentToString ( {\tt Agent} \ * \ agt \ )
```

Converts pointer to an Agant into a string containing name of the agent, its initial state, transitions with their local and global names, shared count and conditions.

Parameters

```
agt Pointer to an Agent to parse into a string.
```

Returns

String containing all of Agent data.

6.16.2.2 envToString()

```
string envToString ( \label{eq:map} \texttt{map} < \texttt{string, int} \ > \ env \ )
```

Converts a map of string and int to a string.

Parameters

```
env Map to be converted into a string.
```

Returns

Returns string " (first_name, second_name, ..., last_name=int_value)"

6.16.2.3 localModelsToString()

```
string localModelsToString ( {\color{red}{\bf LocalModels}} \ * \ {\it lm} \ )
```

Converts pointer to the LocalModels into a string cointaining all Agent instances from the model, initial values of the variables and names of the persistent values.

Parameters

Im Pointer to the local model to parse into a string.

Returns

String containing all of LocalModels data.

6.16.2.4 outputGlobalModel()

Prints the whole GlobalModel into the console. Contains global states with hashes, local states, variables inside those states, global variables, global transitions, local transitions and epistemic classes of agents.

Parameters

globalModel Pointer to a GlobalModel to print into the console.

6.17 Utils.hpp File Reference

```
#include "Types.hpp"
#include "Constants.hpp"
#include <map>
#include <string>
#include <unistd.h>
#include <sys/time.h>
#include <iostream>
#include <fstream>
```

Functions

- string envToString (map< string, int > env)
 - Converts a map of string and int to a string.
- string agentToString (Agent *agt)

Converts pointer to an Agant into a string containing name of the agent, its initial state, transitions with their local and global names, shared count and conditions.

string localModelsToString (LocalModels *Im)

Converts pointer to the LocalModels into a string cointaining all Agent instances from the model, initial values of the variables and names of the persistent values.

void outputGlobalModel (GlobalModel *globalModel)

Prints the whole GlobalModel into the console. Contains global states with hashes, local states, variables inside those states, global variables, global transitions, local transitions and epistemic classes of agents.

• unsigned long getMemCap ()

6.17.1 Function Documentation

6.17.1.1 agentToString()

Converts pointer to an Agant into a string containing name of the agent, its initial state, transitions with their local and global names, shared count and conditions.

Parameters

agt Pointer to an Agent to parse into a string.

Returns

String containing all of Agent data.

6.17.1.2 envToString()

```
string envToString ( \label{eq:map} \texttt{map} < \; \texttt{string, int} \; > \; env \; )
```

Converts a map of string and int to a string.

Parameters

env Map to be converted into a string.

Returns

Returns string " (first_name, second_name, ..., last_name=int_value)"

6.17.1.3 localModelsToString()

```
string localModelsToString ( {\color{red}{\rm LocalModels}} \ * \ {\it lm} \ )
```

Converts pointer to the LocalModels into a string cointaining all Agent instances from the model, initial values of the variables and names of the persistent values.

6.18 Utils.hpp 85

Parameters

Im Pointer to the local model to parse into a string.

Returns

String containing all of LocalModels data.

6.17.1.4 outputGlobalModel()

Prints the whole GlobalModel into the console. Contains global states with hashes, local states, variables inside those states, global variables, global transitions, local transitions and epistemic classes of agents.

Parameters

globalModel | Pointer to a GlobalModel to print into the console.

6.18 Utils.hpp

Go to the documentation of this file.

```
00005 #ifndef STV_TYPES
00006 #define STV_TYPES
00007
00008 #include "Types.hpp"
00009 #include "Constants.hpp"
00010 #include <map>
00011 #include <string>
00012 #include <unistd.h>
00013 #include <sys/time.h>
00014 #include <iostream>
00015 #include <fstream>
00016
00017 using namespace std;
00019 string envToString(map<string, int> env);
00020 string agentToString(Agent* agt);
00021 string localModelsToString(LocalModels* lm);
00022 void outputGlobalModel(GlobalModel* globalModel);
00023 unsigned long getMemCap();
00025 #endif // STV_TYPES
```

6.19 Verification.cpp File Reference

Class for verification of the formula on a model. Class for verification of the specified formula on a specified model.

```
#include "Verification.hpp"
```

Functions

• string verStatusToStr (GlobalStateVerificationStatus status)

Converts global verification status into a string.

• void dbgVerifStatus (string prefix, GlobalState *gs, GlobalStateVerificationStatus st, string reason)

Print a debug message of a verification status to the console.

void dbgHistEnt (string prefix, HistoryEntry *h)

Print a single debug message with a history entry to the console.

Variables

· Cfg config

6.19.1 Detailed Description

Class for verification of the formula on a model. Class for verification of the specified formula on a specified model.

6.19.2 Function Documentation

6.19.2.1 dbgHistEnt()

```
void dbgHistEnt (  \mbox{string } prefix, \\ \mbox{HistoryEntry * } h \mbox{ )}
```

Print a single debug message with a history entry to the console.

Parameters

K	orefix	A prefix string to append to the front of the entry.
ŀ	h	A pointer to the HistoryEntry struct which will be printed out.

6.19.2.2 dbgVerifStatus()

Print a debug message of a verification status to the console.

Parameters

prefix	A prefix string to append to the front of every entry.
gs	Pointer to a GlobalState.
st	Enum with a verification status of a global state.
reason	String with a reason why the function was called, e.g. "entered state", "all passed".

6.19.2.3 verStatusToStr()

Converts global verification status into a string.

Parameters

status Enum value to be conve	erted.
---------------------------------	--------

Returns

Verification status converted into a string.

6.20 Verification.hpp File Reference

```
#include <stack>
#include "Types.hpp"
#include "GlobalModelGenerator.hpp"
#include "SeleneFormula.hpp"
```

Classes

struct HistoryEntry

Structure used to save model traversal history.

class HistoryDbg

Stores history and allows displaying it to the console.

class Verification

A class that verifies if the model fulfills the formula. Also can do some operations on decision history.

Enumerations

- enum HistoryEntryType { DECISION , STATE_STATUS , CONTEXT , MARK_DECISION_AS_INVALID }
 HistoryEntry entry type.
- enum Mode { NORMAL , REVERT , RESTORE }

Current model traversal mode.

Functions

• string verStatusToStr (GlobalStateVerificationStatus status)

Converts global verification status into a string.

6.20.1 Enumeration Type Documentation

6.20.1.1 HistoryEntryType

```
enum HistoryEntryType
```

HistoryEntry entry type.

Enumerator

DECISION	Made the decision to go to a state using a transition.
STATE_STATUS	Changed verification status.
CONTEXT	Recursion has gone deeper.
MARK_DECISION_AS_INVALID	Marking a transition as invalid.

6.20.1.2 Mode

enum Mode

Current model traversal mode.

Enumerator

NC	DRMAL	Normal model traversal.	
R	EVERT	Backtracking through recursion with state rollback.	
RES	STORE	Backtracking through recursion.	

6.20.2 Function Documentation

6.20.2.1 verStatusToStr()

```
string verStatusToStr ( {\tt GlobalStateVerificationStatus}\ status\ )
```

Converts global verification status into a string.

6.21 Verification.hpp 89

Parameters

status | Enum value to be converted.

Returns

Verification status converted into a string.

6.21 Verification.hpp

Go to the documentation of this file.

```
00001
00005 #ifndef SELENE_VERIFICATION
00006 #define SELENE_VERIFICATION
00007
00008 #include <stack>
00009 #include "Types.hpp"
00010 #include "GlobalModelGenerator.hpp"
00011 #include "SeleneFormula.hpp"
00013 string verStatusToStr(GlobalStateVerificationStatus status);
00014
00016 enum HistoryEntryType {
                   DECISION.
00017
00018
                     STATE STATUS.
00019
                     CONTEXT,
                    MARK_DECISION_AS_INVALID,
00021 };
00022
00024 struct HistoryEntry {
              HistoryEntryType type;
GlobalState* globalState;
00026
00028
                     GlobalTransition* decision;
00032
                     bool globalTransitionControlled;
00034
                     GlobalStateVerificationStatus prevStatus;
00036
                    GlobalStateVerificationStatus newStatus;
00038
                     int depth;
                    HistoryEntry* prev;
00040
                    HistoryEntry* next;
00042
00045
                    string toString()
                            char buff[1024] = { 0 };
if (this->type == HistoryEntryType::DECISION) {
    snprintf(buff, sizeof(buff), "decision in %s: to %s", this->globalState->hash.c_str(),
00046
00047
00048
           this->decision->to->hash.c str());
00050
                             else if (this->type == HistoryEntryType::STATE_STATUS) {
00051
                                     snprintf(buff, sizeof(buff), "stateVerStatus of %s: %s -> %s",
            this->globalState->hash.c_str(), verStatusToStr(this->prevStatus).c_str(),
            verStatusToStr(this->newStatus).c_str());
00052
00053
                             else if (this->type == HistoryEntryType::CONTEXT) {
                                     snprintf(buff, sizeof(buff), "context in %s at depth %i: to %s (%s)",
            \label{this-sqlobalState-hash.c_str(), this-sqlobalState-hash.c_str(), this-sqlobalState-has
            this->globalTransitionControlled ? "controlled" : "uncontrolled");
00055
00056
                             else if (this->type == HistoryEntryType::MARK_DECISION_AS_INVALID) {
                                     snprintf(buff, sizeof(buff), "markInvalid in %s: to %s", this->globalState->hash.c_str(),
00057
           this->decision->to->hash.c_str());
00058
00059
                             return string(buff);
00060
                    };
00061 };
00062
00064 class HistoryDbg {
00067
                    vector<pair<HistoryEntry*, char» entries;</pre>
00068
                    HistoryDbg();
                     ~HistoryDbg();
00069
00070
                    void addEntry(HistoryEntry* entry);
void markEntry(HistoryEntry* entry, char chr);
00071
00072
                     void print(string prefix);
00073
                     HistoryEntry* cloneEntry(HistoryEntry* entry);
00074 };
00075
00076 // On-the-fly traversal mode
00078 enum Mode {
                    NORMAL,
```

```
08000
           REVERT,
00081
           RESTORE,
00082 };
00083
00085 class Verification {
00086 public:
           Verification(GlobalModelGenerator* generator);
00088
           ~Verification();
00089
           bool verify();
00090 protected:
00092
           Mode mode;
00094
           GlobalState* revertToGlobalState;
00096
           stack<HistoryEntry*> historyToRestore;
00098
           GlobalModelGenerator* generator;
00100
           SeleneFormula* seleneFormula;
           HistoryEntry* historyStart;
HistoryEntry* historyEnd;
00102
00104
00105
           bool verifyLocalStates(set<LocalState*>* localStates);
           bool verifyGlobalState(GlobalState* globalState, int depth);
00107
           bool isGlobalTransitionControlledByCoalition(GlobalTransition* globalTransition);
00108
           bool isAgentInCoalition(Agent* agent);
00109
           EpistemicClass* getEpistemicClassForGlobalState(GlobalState* globalState);
           bool areGlobalStatesInTheSameEpistemicClass(GlobalState* globalState1, GlobalState* globalState2); void addHistoryDecision(GlobalState* globalState, GlobalTransition* ecision);
00110
00111
00112
           void addHistoryStateStatus(GlobalState* globalState, GlobalStateVerificationStatus prevStatus,
      GlobalStateVerificationStatus newStatus);
00113
           void addHistoryContext(GlobalState* globalState, int depth, GlobalTransition* decision, bool
      globalTransitionControlled);
           void addHistoryMarkDecisionAsInvalid(GlobalState* globalState, GlobalTransition* decision);
HistoryEntry* newHistoryMarkDecisionAsInvalid(GlobalState* globalState, GlobalTransition*
00114
00115
      decision);
00116
           bool revertLastDecision(int depth);
00117
           void undoLastHistoryEntry(bool freeMemory);
00118
           void undoHistoryUntil(HistoryEntry* historyEntry, bool inclusive, int depth);
00119
           void printCurrentHistory(int depth);
00120 };
00121
00122 #endif // SELENE_VERIFICATION
```

Index

I-IC	01-1-1114-1-10
addEntry	GlobalModelGenerator, 44
HistoryDbg, 49	Condition, 17
addHistoryContext	ConditionOperator
Verification, 60	Types.hpp, 77
addHistoryDecision	Constants.hpp, 67
Verification, 61	CONTEXT
addHistoryMarkDecisionAsInvalid	Verification.hpp, 88
Verification, 61	
addHistoryStateStatus	dbgHistEnt
Verification, 61	Verification.cpp, 86
addInitial	dbgVerifStatus
AgentTemplate, 11	Verification.cpp, 86
addLocal	DECISION
AgentTemplate, 12	Verification.hpp, 88
addPersistent	
AgentTemplate, 12	envToString
addTransition	Utils.cpp, 82
AgentTemplate, 13	Utils.hpp, 84
Agent, 9	EpistemicClass, 17
Agent, 10	Equals
includesState, 10	Types.hpp, 78
AgentTemplate, 10	eval
addInitial, 11	ExprAdd, 19
•	ExprAnd, 20
addLocal, 12	ExprConst, 21
addPersistent, 12	ExprDiv, 23
addTransition, 13	ExprEq, 24
generateAgent, 13	ExprGe, 25
setIdent, 14	ExprGt, 27
setInitState, 14	Exprident, 28
agentToString	Exprise 11, 20
Utils.cpp, 82	•
Utils.hpp, 84	ExprLt, 31
areGlobalStatesInTheSameEpistemicClass	ExprMul, 32
Verification, 62	ExprNe, 34
assign	ExprNode, 35
Assignment, 16	ExprNot, 36
Assignment, 15	ExprOr, 37
assign, 16	ExprRem, 39
Assignment, 16	ExprSub, 40
•	expandState
Cfg, 16	GlobalModelGenerator, 44
checkLocalTransitionConditions	ExprAdd, 18
GlobalModelGenerator, 43	eval, 19
cloneEntry	ExprAdd, 18
HistoryDbg, 49	ExprAnd, 19
compare	eval, 20
LocalState, 53	ExprAnd, 20
computeEpistemicClassHash	ExprConst, 20
GlobalModelGenerator, 43	eval, 21
computeGlobalStateHash	ExprConst, 21
· called a contact terms of te	The state of the s

92 INDEX

ExprDiv, 22	GlobalModelGenerator, 45
eval, 23	generateStateFromLocalStates
ExprDiv, 22	GlobalModelGenerator, 46
ExprEq, 23	getCurrentGlobalModel
eval, 24	GlobalModelGenerator, 46
ExprEq, 24	getEpistemicClassForGlobalState
expressions.cc, 68	Verification, 62
expressions.hpp, 69, 70	getFormula
ExprGe, 25	GlobalModelGenerator, 46
eval, 25	GlobalModel, 41
· · · · · · · · · · · · · · · · · · ·	
ExprGe, 25	GlobalModelGenerator, 42
ExprGt, 26	checkLocalTransitionConditions, 43
eval, 27	computeEpistemicClassHash, 43
ExprGt, 26	computeGlobalStateHash, 44
Exprident, 27	expandState, 44
eval, 28	findGlobalStateInEpistemicClass, 44
Exprident, 28	findOrCreateEpistemicClass, 45
ExprLe, 29	generateGlobalTransitions, 45
eval, 30	generateInitState, 45
ExprLe, 29	generateStateFromLocalStates, 46
ExprLt, 30	getCurrentGlobalModel, 46
eval, 31	getFormula, 46
ExprLt, 31	initModel, 46
ExprMul, 32	GlobalModelGenerator.cpp, 67
eval, 32	GlobalModelGenerator.hpp, 67, 68
ExprMul, 32	GlobalState, 47
ExprNe, 33	GlobalStateVerificationStatus
eval, 34	Types.hpp, 78
ExprNe, 33	GlobalTransition, 48
ExprNode, 34	
eval, 35	HistoryDbg, 48
ExprNot, 35	addEntry, 49
eval, 36	cloneEntry, 49
ExprNot, 36	markEntry, 50
ExprOr, 36	print, 50
eval, 37	HistoryEntry, 50
ExprOr, 37	toString, 51
	HistoryEntryType
ExprRem, 38	Verification.hpp, 88
eval, 39	vormoutioninpp, oo
ExprRem, 38	includesState
ExprSub, 39	Agent, 10
eval, 40	initModel
ExprSub, 40	GlobalModelGenerator, 46
findClabalCtatala Enistancia Class	isAgentInCoalition
findGlobalStateInEpistemicClass	Verification, 63
GlobalModelGenerator, 44	isGlobalTransitionControlledByCoalition
findOrCreateEpistemicClass	
GlobalModelGenerator, 45	Verification, 63
Formula, 41	LocalModels, 52
FromValue	localModelsToString
Types.hpp, 78	Utils.cpp, 82
FromVar	• •
Types.hpp, 78	Utils.hpp, 84
	LocalState, 52
generateAgent	compare, 53
AgentTemplate, 13	LocalStateTemplate, 53
generateGlobalTransitions	LocalTransition, 54
GlobalModelGenerator, 45	MARK DECISION AS INVALID
generateInitState	MARK_DECISION_AS_INVALID

INDEX 93

newHistoryMarkDecisionAsInvalid undoHistoryUntil Verification, 64 nodes.cc, 73 undoLastHistoryEntry Verification, 65 NOPMAL Verification, 65 NVERIFIEED Voregrate Types hpp, 78 Verification, 65 NOEquals Uils.cpp, 81 Types ppp, 78 OutputGlobalModel Uils.cpp, 83 Uils.cpp, 81 2 Utils.pp, 83 Uils.hpp, 83 55 2 apentToString, 82 outputGlobalModel, 83 3 Uils.hpp, 83 3 10 coalModels ToString, 82 outputGlobalModel, 83 3 Uils.hpp, 83 85 2 outputGlobalModel, 83 3 10 coalModels ToString, 82 outputGlobalModel, 83 10 coalModels ToString, 84 outputGlobalModel, 83 10 coalModels ToString, 84 outputGlobalModel, 83 10 coalModels ToString, 84 outputGlobalModel, 85 10 coalModels ToString, 84 10 coalModels ToString, 84 10 coa	Verification.hpp, 88 markEntry HistoryDbg, 50 Mode Verification.hpp, 88	NotEquals, 78 PENDING, 78 UNVERIFIED, 78 VarAssignmentType, 78 VERIFIED_OK, 78
Verification, 63 Verification, 64 undoLastHistoryEntry Verification, 65 Verification, 66 Verification, 67 Verific	newHistoryMarkDecisionAsInvalid	undoHistoryUntil
nodes.cc, 73 nodes.hpp, 73, 74 NORMAL Verification.hpp, 88 NotEquals Types.hpp, 78 outputGlobalModel Utils.cpp, 83 Julis.hpp, 85 Julis.cpp, 81 Julis.cpp, 83 Julis.hpp, 85 Julis.cpp, 83 Julis.hpp, 8	-	
nodes hpp, 73, 74 Verification, 65 NORMAL UNVERIFIED Verification, hpp, 88 Types hpp, 78 NotEquals Uilis c.pp, 81 Types hpp, 78 uilis c.pp, 81 outputGlobalModel Quilis c.pp, 83 Uilis c.pp, 83 Uilis hpp, 83, 85 parse TostParser, 56 PENDING Jypes hpp, 78 print Var, 58 PENDING Var, 58 print HistoryDbg, 50 Var, 58 Print HistoryDe, 50 Var, 58 Persone Verification hpp, 88 Var, 58 REVERT Verification, 59 Verification, 59, 88 devertLastDecision, 61 REVERT AddHistoryDecision, 61 Verification, 64 devertLastDecision, 61 SeleneFormula, 54 devertLastDecision, 61 SeleneFormula, 54 devertLastDecision, 61 SeleneFormula, 54 devertLastDecision, 62 SeleneFormula, 54 devertLastDecision, 62 AgentTemplate, 14 devertLastDecision, 62 Settledent devertLastDecision, 64 <		
NORMAL UNVERIFIED Verification.hpp, 88 Types.hpp, 78 NotEquals Utils.cpp, 81 Types.hpp, 78 agentToString, 82 outputGlobalModel utils.cpp, 83 Utils.cpp, 83 utils.hpp, 85 parse agentToString, 84 TostParser, 56 envToString, 84 PENDING Types.hpp, 78 print Var, 58 HistoryDbg, 50 Var, 58 printCurrentHistory Var/ssignment, 58 Verification, 64 Var/ssignmentType RESTORE Verification.hpp, 88 REVERT Verification, 59 Verification, 64 verification, 59 Verification, 64 addHistoryDecision, 61 Verification, 64 addHistoryMarkDecisionAsInvalid, 61 Verification, 64 addHistoryStateStatus, 61 Reserbormula, 54 seleneFormula, 54 SeleneFormula, 55 seleneFormula, 54 SeleneFormula, 54 siagentification, 63 SeleneFormula, 55 serification, 60 Verification, 60 verifyColoalState, 65		
Verification.hpp, 88 NotEquals Types.hpp, 78 outputGlobalModel Utils.cpp, 83 Utils.hpp, 85 Utils.hpp, 85 Utils.hpp, 83 Utils.hpp, 85 Utils.hpp, 83 Utils.hpp, 83 Utils.hpp, 83 Utils.hpp, 83, 85 parse TestParser, 56 PENDING Types.hpp, 78 Print HistoryDbg, 50 Verification, 84 Verification, hpp, 88 REVERT Verification hpp, 88 REVERT Verification, 64 SeleneFormula, 54 SeleneFormula 1, 55 verifyLocalStates, 55 setIdent AgentTemplate, 14 SetInitState AgentTemplate, 15 Types.hpp, 78 Utils.cpp, 81 Suchidation Agent Ag	• •	
NotEquals Types.hpp, 78 Types.hpp, 78 Types.hpp, 85 Dutlis.cpp, 83 Utils.hpp, 85 Utils.cpp, 83 Utils.hpp, 85 Utils.hpp, 85 Utils.hpp, 85 Utils.hpp, 85 Utils.hpp, 83 Utils.hpp, 85 Utils.hpp, 83 Utils.hpp, 84 UocalModels ToString, 84 UocalModels ToSt	Verification.hpp. 88	
Types.hpp, 78 outputGlobalModel Utils.cpp, 83 Utils.hpp, 85 parse TestParser, 56 PENDING Types.hpp, 78 Print HistoryDbg, 50 PrintCurrentHistory Verification, 64 RESTORE Verification, 64 REVERT Verification, 64 SeleneFormula, 54 SeleneFormula, 55 verifyLocalStates, 55 setIdent AgentTemplate, 14 Striates AgentTemplate, 14 Striates AgentTemplate, 14 Striates AgentTemplate, 14 Striates AgentTemplate, 15 TestParser, 55 parse, 56 TestParser, 55 parse, 56 TestParser, 57 Types.cp, 76 Types.cp, 76 Types.cp, 76 Types.cp, 76 TransitionTemplate, 57 Types.cc, 76 Types.cp, 78 TrowValue, 78 FromValue, 78 F	• •	
outputGlobalModel Utils.cpp, 83 Utils.hpp, 85 Utils.hpp, 85 Utils.hpp, 85 Utils.hpp, 83, 85 parse TestParser, 56 PENDING Types.hpp, 78 Thittion, 64 TestParser, 56 Perfication.hpp, 88 REVERT Verification.hpp, 75 SeleneFormula, 54 SeleneFormula, 55 SeleneFormula, 56 Verification.hpp, 75 SeleneFormula, 56 Verification.hpp, 75 SeleneFormula, 56 Verification.hpp, 88 Verification.hpp, 87, 89 ConditionOperator, 77 Equals, 78 NORMAL, 88 RESTORE, 88 REVERT, 88	•	• •
outputGlobalModel localModels ToString, 82 outputGlobalModel, 83 Utils.hpp, 85 Utils.hpp, 83, 85 parse agentToString, 84 envToString, 84 outputGlobalModel, 85 FENDING outputGlobalModels ToString, 84 outputGlobalModels ToString, 84 outputGlobalModels ToString, 84 outputGlobalModel, 85 PENDING Var, 58 print Var, 58 HistoryDbg, 50 Var, 58 printCurrentHistory VarAssignment, 58 Verification, 64 VarAssignment, 58 Verification, 64 VarAssignmentType Verification, 59 addHistoryOntext, 60 adHistoryDecision, 61 addHistoryObateStates, 61 adHistoryObateStates, 61 addHistoryObateStates, 61 adHistoryObateStates, 61 addHistoryObateStates, 61 areGlobalStatesInTneSameEpistemicClass, 62 getEpistemicClassForGlobalState, 62 seleneFormula, 54 seleneFormula, 54 SeleneFormula, 1, 55 seleneFormula, 54 seleneFormula, 1, 55 seleneFormula, 1, 55 setIdent AgentTemplate, 14 setInitState AgentTemplate, 14 setInitState undoHistoryUntil, 64	21	-
Utils.hpp, 85 utils.hpp, 83, 85 Utils.hpp, 85 Utils.hpp, 83, 85 parse agentToString, 84 TestParser, 56 envToString, 84 PENDING outputGlobalModel, 85 Pint Var, 58 HistoryDbg, 50 Var, 58 Verification, 64 Var/assignment, 58 Verification, 64 Var/assignmentType Verification.hpp, 88 Verification, 59 REVERT addHistoryOcntext, 60 Verification, hpp, 88 addHistoryOcntext, 61 revertLastDecision addHistoryOcntext, 61 Verification, 64 addHistoryOcntext, 60 SeleneFormula, 54 addHistoryOcntext, 61 SeleneFormula, 55 areGlobalStatesInTheSameEpistemicClass, 62 yetEpistemicClassForGlobalState, 62 isGlobalTransitionControlledByCoalition, 63 seleneFormula, 55 newHistoryMarkDecisionAsInvalid, 63 yerifyLocalStates, 55 setIdent werifyLocalStates, 65 printCurrentHistory, 64 verifyIncurrentHistory, 64 undoLastHistoryPurty, 65 verifyIncurrentHistory verifyIncurrentHistory, 64	outputGlobalModel	_
Utils.hpp, 85 bparse TestParser, 56 PENDING Types.hpp, 78 print HistoryDbg, 50 printCurrentHistory Verification.hpp, 88 RESTORE Verification.hpp, 88 REVERT Verification, 64 SeleneFormula, 55 SeleneFormula, 1, 55 verifyLocalStates, 55 setIdent AgentTemplate, 14 SetInitState AgentTemplate, 14 State_STATUS Verification.hpp, 88 CestParser, 55 Destarser, 55 TestParser, 55 TestParser, 56 TestParser, 56 TransitionTemplate, 57 Types.hpp, 76 Types.hpp, 76 Types.hpp, 78 Types.hpp, 78 Var, 58 Var/Assignment, 58 Var/Assignment, 58 Var/AssignmentType Types.hpp, 78 Varification, 59 Var/AssignmentType Types.hpp, 78 Varification, 59 Var/AssignmentType Types.hpp, 78 Varification, 59 Var/Assignment, 58 Var/Bable Var/Bable Var/Assignment, 58 Var/Assignment, 58 Var/Assignment, 58 Va	··	
parse TestParser, 56 PENDING Types.hpp, 78 print HistoryDbg, 50 printCurrentHistory Verification, 64 RESTORE Verification.hpp, 88 REVERT Verification.hpp, 88 revertLastDecision Verification, 64 SeleneFormula, 54 SeleneFormula, 55 SeleneFormula1, 55 verifyLocalStates, 55 setIdent AgentTemplate, 14 setInitState AgentTemplate, 14 STATE_STATUS Verification.hpp, 88 TestParser, 55 parse, 56 TestParser, 156 TiransitionTemplate, 57 Types.hpp, 76 Types.hpp, 76 RESTORE AgentToString, 84 outputGlobalModel, 85 Var, 58 Var/Assignment, 59 var/Bull, 59 Var/Assignment, 58 Var/Assignment, 58 Var/Assignment, 58 Var/Assignment, 59 var/Bull, 60 addHistoryDotx, 60 addHis	Utils.hpp, 85	•
TestParser, 56 PENDING Types.hpp, 78 print HistoryDbg, 50 printCurrentHistory Verification.hpp, 88 REVERT Verification.hpp, 88 revertLastDecision Verification, 64 SeleneFormula, 54 SeleneFormula, 55 verifyLocalStates, 55 setIdent AgentTemplate, 14 setIntiState AgentTemplate, 14 STATE_STATUS Verification.hpp, 88 TestParser, 55 parse, 56 TestParser, 15 TransitionTemplate, 56 TransitionTemplate, 56 TransitionTemplate, 56 TransitionTemplate, 56 TransitionTemplate, 57 Types.hpp, 78 Var, 58 VarAssignment, 58 VarIssignment, 58 VarIssignment, 58 VarIssignment, 58 VarAssignment, 58 VarIssignment, 58 VarIssignment, 58 VarIssignment, 58 VarAssignment, 58 VarAssignment, 58 VarIssignment, 58 VarIssignment, 58 VarIssignment, 58 VarAssignment, 58 VarIssignment, 58 VarIssignment, 58 VarAssignment, 58 VarIssignment, 58 VarIssignment, 58 VarAssignment, 58 VarAssignment, 58 VarIssignment, 58 VarIssignment, 58 VarIssignment, 58 VarIssignment, 58 VarIssignment, 59 VarIssig		• •
InestFarser, 56 PENDING Types.hpp, 78 print HistoryDbg, 50 printCurrentHistory Verification, 64 RESTORE Verification.hpp, 88 REVERT Verification.hpp, 88 revertLastDecision Verification, 64 SeleneFormula, 54 SeleneFormula, 1,55 verifyLocalStates, 55 setIddent AgentTemplate, 14 setInitState AgentTemplate, 14 StatE_STATUS Verification.hpp, 88 TestParser, 55 parse, 56 TestParser, 150 TransitionTemplate, 56 TransitionTemplate, 56 TransitionTemplate, 56 TransitionTemplate, 56 TransitionTemplate, 57 Types.cc, 76 Types.cc, 76 Types.cc, 76 Types.cp, 76, 78 Condition, 78 Parser, 58 PromVaure, 78 Person Agent Parser, 78 79 Person Agent	•	envToString, 84
Types.hpp, 78 print HistoryDbg, 50 printCurrentHistory Verification, 64 RESTORE Verification.hpp, 88 REVERT Verification.pp, 88 revertLastDecision Verification, 64 SeleneFormula, 54 SeleneFormula, 55 verifyLocalStates, 55 setIdent AgentTemplate, 14 setInitistate AgentTemplate, 14 STATE_STATUS Verification.hpp, 88 TestParser, 55 parse, 56 TestParser, 157 TransitionTemplate, 56 TransitionTemplate, 56 TransitionTemplate, 57 Types.hpp, 78 Var, 58 VarAssignment, 58 Varification, 59 addHistoryCentext, 60 addHistoryDecision, 61 addHistoryDecision, 61 addHistoryDecisionAsInvalid, 61 addHistoryStateStatus, 61 areGlobalStatesInTheSameEpistemicClass, 62 getEpistemicClassForGlobalState, 62 getEpistemicClassForGlobalState, 62 getEpistemicClassForGlobalState, 62 getEpistemicClassForGlobalState, 63 isGlobalTransitionControlledByCoalition, 63 isGlobalTransitionControlledByCoalition, 63 newHistoryMarkDecisionAsInvalid, 63 printCurrentHistory, 64 revertLastDecision, 64 undoHistoryUntil, 64 revertLastDecision, 64 undoHistoryUntil, 64 verification, 60 verify, 65 Verification.hpp, 88 verifylocalState, 65 verifyLocalStates, 65 Verification.pp, 88 verifylocalState, 65 verifyLocalStates, 65 Verification.pp, 87 dbgHistEnt, 86 dbgVerifStatus, 86 verStatusToStr, 87 Verification.pp, 87, 89 CONTEXT, 88 DECISION, 88 HistoryEntryType, 88 MARK_DECISION_AS_INVALID, 88 ConditionOperator, 77 Equals, 78 REVERT, 88 REVERT, 88	,	_
lypes.npp, 78 print HistoryDbg, 50 PrintCurrentHistory Verification, 64 RESTORE Verification.hpp, 88 REVERT Verification.hpp, 88 revertLastDecision Verification, 64 SeleneFormula, 54 SeleneFormula, 55 SeleneFormulat, 55 verifyLocalStates, 55 setIdent AgentTemplate, 14 SetInitState AgentTemplate, 14 STATE_STATUS Verification.hpp, 88 TestParser, 55 parse, 56 TestParser, 55 parse, 56 TestParser.hpp, 76 toString HistoryEnty, 51 TransitionTemplate, 56 TransitionTemplate, 57 Types.hpp, 76, 78 ConditionOperator, 77 Equals, 78 FromWar, 78 FromWar, 78 FromWar, 78 FromWar, 78 Verification Var, 58 VarAssignment, 58 Varification, 50 addHistoryDecision, 61 addHistoryDecision, 61 addHistoryDecision, 61 addHistoryDecision, 61 addHistoryDecision, 61 addHistoryDecision, 62 gelEpistemicClass ForGlobalState, 62 gelEpistemicClass ForGlobalState, 62 gelEpistemicClass ForGlobalState, 63 perible calculation, 63 is GlobalTransitionControlledByCoalition, 63 newHistoryMarkDecisionAsInvalid, 61 addHistoryDucki, 60 addHistoryDucki, 60 addHistoryDucki, 60 addHistoryDucki, 60 addHistoryDucki, 61 addHistoryDuckine, 61 addHistoryDuckine, 61 addHistoryDuckine, 61 addHistoryDuckine, 61 addHistoryDuckine, 61 addHistoryDuck		
HistoryDbg, 50 printCurrentHistory Verification, 64 RESTORE Verification.hpp, 88 REVERT Verification.hpp, 88 REVERT Verification, 64 Verification, 64 RESTORE Verification.hpp, 88 REVERT Verification, 64 RESTORE Verification.hpp, 88 REVERT Verification, 64 RESTORE Verification.hpp, 88 REVERT Verification, 64 RESTORE Verification, 65 REVERT Verification, 64 REVERT Verification, 64 REVERT Verification, 64 REVERT Verification, 64 REVERT Verification, 65 REVERT Verification, 64 REVERT		
printCurrentHistory Verification, 64 RESTORE Verification.hpp, 88 REVERT Verification, 64 REVERT REVERT, 88 AddHistoryContext, 60 addHistoryDentext, 60 addHistoryDecision, 61 addHistoryDecision, 62 addHistoryDecision, 61 addHistoryDecision, 62 arcGloabletes, 62 arcGl	•	Var, 58
Verification, 64 RESTORE Verification.hpp, 88 REVERT Verification.hpp, 88 REVERT Verification, 64 REVERT Verification, 64 REVERT Verification, 64 SeleneFormula, 54 SeleneFormula, 55 verifyLocalStates, 55 setIdent AgentTemplate, 14 setInitState AgentTemplate, 14 STATE_STATUS Verification.hpp, 88 Types.hpp, 76 Seleparser, 55 parse, 56 TestParser, 55 parse, 56 TestParser, 57 Types.oc, 76 Types.hpp, 78 Verification, 69 addHistoryDecision, 61 addHistoryDecisionAsInvalid, 61 areGlobalStatesInTheSameEpistemicClass, 62 getEpistemicClassForGlobalState, 62 isAgentInCoallition, 63 isGlobalTransitionControlledByCoallition, 63 newHistoryMarkDecisionAsInvalid, 63 printCurrentHistory, 64 revertLastDecision, 64 undoHistoryUntil, 64 undoLastHistoryEntry, 65 verification, 60 verification, 60 verification, 60 verify, 65 verifyLocalState, 65 verifyLocalState, 65 verifyLocalState, 65 verifyLocalState, 65 verifyLocalState, 86 dbgVerifStatus, 86 verStatusToStr, 87 Verification.hpp, 87, 89 CONTEXT, 88 TransitionTemplate, 57 Types.oc, 76 Types.pp, 76, 78 ConditionOperator, 77 Equals, 78 FromValue, 78 FromVar, 78 REVERT, 88	· · ·	VarAssignment, 58
RESTORE Verification.hpp, 88 REVERT Verification.hpp, 88 REVERT Verification.hpp, 88 revertLastDecision Verification, 64 SeleneFormula, 54 SeleneFormula, 1, 55 verifyLocalStates, 55 setIdent AgentTemplate, 14 setInitState AgentTemplate, 14 STATE_STATUS Verification.hpp, 88 TestParser, 55 parse, 56 TestParser, 55 parse, 56 TestParser, 15 TransitionTemplate, 56 TransitionTemplate, 57 Types.cc, 76 Types.hpp, 76, 78 ConditionOperator, 77 Equals, 78 FromVar, 78 RESTORE Verification.hpp, 88 revertLastDecision, 61 addHistoryDecisionAsInvalid, 61 addHistoryStates Status, 61 areGlobalStatesIntheSameEpistemicClass, 62 getEpistemicClassForGlobalState, 63 printCurrentHistory_Unit, 64 addHistoryDamid, 61 addHistoryDamid, 62 getEpistemicClass, 62	•	
RESTORE Verification.hpp, 88 REVERT Verification.hpp, 88 revertLastDecision Verification, 64 RevertLastDecision Verification, 64 Verification, 64 SeleneFormula, 54 SeleneFormula, 1,55 SeleneFormula, 1,55 verifyLocalStates, 55 setIdent AgentTemplate, 14 STATE_STATUS Verification.hpp, 88 TestParser, 55 parse, 56 TestParser, hpp, 76 toString HistoryEntry, 51 TransitionTemplate, 56 TransitionTemplate, 56 TransitionTemplate, 56 TransitionTemplate, 56 TransitionTemplate, 57 Types.cc, 76 Types.hpp, 76, 78 ConditionOperator, 77 Equals, 78 FromVar, 78 REVERT, 88 AddHistoryContext, 60 addHistoryDecision, 61 addHistoryDecision, 61 addHistoryDecisionAsInvalid, 61 addHistoryDecisionAsInvalid, 61 addHistoryDecisionAsInvalid, 61 addHistoryDecisionAsInvalid, 61 addHistoryDecisionAsInvalid, 61 addHistoryEntrestatus, 62 setders addHistoryEntreclass, 62 getEpistemicClassForGlobalState, 62 getEpistemicClassForGlobalState, 63 perintCurrentHistory_netalled is AgentTroControlledByCoalition, 63 newHistoryMill, 64 undoHistoryUntil, 64 undoHistoryUntil, 64 undoHistoryUntil, 64 undoHistoryUntil, 64 undoHistoryDecisionAsInvalid, 63 printCurrentHistory_netaltory_neta	verification, 64	• • • • • • • • • • • • • • • • • • • •
Verification.hpp, 88 REVERT Verification.hpp, 88 revertLastDecision Verification, 64 Verification, 64 Verification, 64 SeleneFormula, 54 SeleneFormula, 1, 55 verifyLocalStates, 55 setIdent AgentTemplate, 14 setInitState AgentTemplate, 14 STATE_STATUS Verification.hpp, 88 TestParser, 55 parse, 56 TestParser, 55 restParser, 55 restParser, 55 restParser, 55 restParser, 56 TestParser, 57 TransitionTemplate, 56 TransitionTemplate, 57 Types.cc, 76 Types.hpp, 76, 78 ConditionOperator, 77 Equals, 78 FromVar, 78 RestParser, 88 REVERT, 88	BESTORE	
REVERT Verification.hpp, 88 revertLastDecision Verification, 64 SeleneFormula, 54 SeleneFormula, 55 SeleneFormula, 55 verifyLocalStates, 55 setIdent AgentTemplate, 14 setInitState AgentTemplate, 14 Senere, 55 parse, 56 TestParser, 55 parse, 56 TestParser, hpp, 76 toString HistoryEntry, 51 TransitionTemplate, 56 TransitionTemplate, 57 Types.cc, 76 Types.cc, 76 Types.hpp, 76, 78 ConditionOperator, 77 Equals, 78 FromVar, 78 RestDats dadHistoryDecisionAsInvalid, 61 areGlobalStates latus, 61 areGlobalStates latus, 61 areGlobalStatesInTheSameEpistemicClass, 62 getEpistemicClassForGlobalState, 63 isAgentInCoalition, 63 areGlobalStatesIntheSameEpistemicClass, 62 getEpistemicClassForGlobalState, 63 perbloalStates, 65 printCurrentHistoryDutlledByCoalition, 63 areGlobalStatesIntheSameEpistemicClass, 62 getEpistemicClassForGlobalState, 63 perbloalState, 63 printCurrentHistoryDutlledByCoalition, 63 areGlobalState, 63 printCurrentHistory, 64 revertLastDecision, 64 undoHistoryDutll, 64 undoLastHistoryEntry, 65 verify, 65 veri		
Verification.hpp, 88 revertLastDecision Verification, 64 Verification, 64 SeleneFormula, 54 SeleneFormula.hpp, 75 SeleneFormula1, 55 verifyLocalStates, 55 setIdent AgentTemplate, 14 setInitState AgentTemplate, 14 SetParser, 55 parse, 56 TestParser.hpp, 76 toString HistoryEntry, 51 TransitionTemplate, 57 Types.cr, 76 Types.npp, 76, 78 Coddition, 63 addHistoryStateStatus, 61 areGlobalStateStatus, 61 areGlobalStatesInTheSameEpistemicClass, 62 getEpistemicClassForGlobalState, 62 isAgentInCoalition, 63 isGlobalTransitionControlledByCoalition, 63 newHistoryMarkDecisionAsInvalid, 63 printCurrentHistory, 64 revertLastDecision, 64 undoHistoryUntil, 64 undoLastHistoryEntry, 65 Verification, 60 verify, 65 verifyGlobalState, 65 verifyLocalStates, 65 Verification.cpp, 85 dbgHistEnt, 86 dbgVerifStatus, 86 toString verStatusToStr, 87 HistoryEntry, 51 TransitionTemplate, 56 TransitionTemplate, 57 Types.cc, 76 Types.cc, 76 Types.npp, 76, 78 ConditionOperator, 77 Mode, 88 FromValue, 78 RESTORE, 88 FromVar, 78 RESTORE, 88 FromVar, 78 RESTORE, 88 FromVar, 78 REVERT, 88	··	
revertLastDecision Verification, 64 SeleneFormula, 54 SeleneFormula, 55 SeleneFormula, 55 SeleneFormula, 55 SeleneFormula, 55 SeleneFormula, 55 verifyLocalStates, 55 setIdent AgentTemplate, 14 setInitState AgentTemplate, 14 STATE_STATUS Verification.hpp, 88 VerifyGlobalStates, 65 VerifyCoalStates, 55 TestParser, 55 parse, 56 TestParser.hpp, 76 toString HistoryEntry, 51 TransitionTemplate, 57 Types.cc, 76 Types.hpp, 76, 78 ConditionOperator, 77 Equals, 78 FromVar, 78 SeleneFormula, 54 areGlobalStatesInTheSameEpistemicClass, 62 getEpistemicClassForGlobalState, 62 isAgentInCoalition, 63 areGlobalState, 62 isAgentInCoalition, 63 areGlobalState, 63 petEpistemicClassForGlobalState, 63 printCurrentHistoryCntrolledByCoalition, 63 newHistoryMarkDecisionAsInvalid, 63 printCurrentHistory, 64 revertLastDecision, 64 undoHistoryUntil, 64 undoHistoryUntil, 64 undoLastHistoryEntry, 65 Verification, 60 verify, 65 verifyGlobalState, 65 verifyGlobalState, 65 verifyGlobalState, 65 verifyGlobalState, 65 verifyGlobalState, 65 Verification.cpp, 85 dbgHistent, 86 dbgVerifStatus, 86 verStatusToStr, 87 Verification.pp, 87, 89 CONTEXT, 88 TransitionTemplate, 56 TestParser.hpp, 76, 78 DECISION, 88 HistoryEntryType, 88 MARK_DECISION_AS_INVALID, 88 Mode, 88 RESTORE, 88 FromVar, 78 RESTORE, 88 REVERT, 88		
Verification, 64 getEpistemicClassForGlobalStates, 62 getEpistemicClassForGlobalState, 62 isAgentInCoalition, 63 isGlobalTransitionControlledByCoalition, 63 SeleneFormula.hpp, 75 SeleneFormula1, 55 verifyLocalStates, 55 setIdent AgentTemplate, 14 setInitState AgentTemplate, 14 STATE_STATUS Verification.hpp, 88 VerifyGlobalState, 65 verifyLocalStates, 65 TestParser, 55 parse, 56 TestParser.hpp, 76 toString HistoryEntry, 51 TransitionTemplate, 56 TransitionTemplate, 57 Types.cc, 76 Types.cc, 76 Types.hpp, 76, 78 ConditionOperator, 77 Equals, 78 FromValue, 78 FromValue, 78 FromValue, 78 FromValue, 78 FromVar, 78 SeleneFormula, 14 siagettipitemicClass, 62 getEpistemicClassForGlobalState, 63 isAgentInCoalition, 63 isGlobalTransitionCendByCoalition, 63 isGlobalTransitionCentrolledbyCoalition, 63 isGlobalTransitionCendralition, 63 isGlobalTransitionControlledbyCoalition, 63 isGlobalTansitionControlledbyCoalition, 63 isGlobalTansitionControlledbyCoalition, 63 isGlobalTansitionControlledbyCoalition, 63 isGlobalTansitionControlledbyCoalition, 63 isGlobalTansitionControlledbyCoalition, 63 isGlobalTansitionControlledbyCoalition, 64 inductionControlledbyCoalition, 64 inductionControlledbyCoalition, 64 inductionControlledbyCoalition, 64 inductionCon		
SeleneFormula, 54 SeleneFormula, 54 SeleneFormula, hpp, 75 SeleneFormula, 55 SeleneFormula, 55 verifyLocalStates, 55 setIdent AgentTemplate, 14 setInitState AgentTemplate, 14 STATE_STATUS Verification, hpp, 88 Verification, hpp, 88 Verification, hpp, 88 Verification, hpp, 88 Verification, pp, 85 parse, 56 TestParser, 55 parse, 56 TestParser, 55 TestParser, 55 TestParser, 55 TestParser, 56 TestParser, 57 TestParser, 57 Types. C, 76 Types. hpp, 76 ConditionOperator, 77 Equals, 78 FromValue, 78 FromValue, 78 FromValue, 78 FromValue, 78 FromVal, 78 SeleneFormula, 54 isAgentIncoalition, 63 isAgentInCoalition, 64 isAgentInCoalition, 64 undolation, 64 undolat		
Selene Formula, 1975 Selene Formula, 1975 Selene Formula, 1975 Selene Formula, 1, 55 verifyLocalStates, 55 setIdent AgentTemplate, 14 setInitState AgentTemplate, 14 STATE_STATUS Verification.hpp, 88 Verification.cpp, 85 parse, 56 TestParser, 55 parse, 56 TestParser.hpp, 76 toString HistoryEntry, 51 TransitionTemplate, 57 Types.cc, 76 Types.hpp, 76, 78 ConditionOperator, 77 Equals, 78 FromValue, 78 FromValue, 78 FromValue, 78 FromValue, 78 Setloer Inexperser instition ControlledByCoalition, 63 newHistoryMarkDecisionAsInvalid, 63		= .
SeleneFormula1, 55 verifyLocalStates, 55 setIdent AgentTemplate, 14 setInitState AgentTemplate, 14 STATE_STATUS Verification.hpp, 88 Verification.cpp, 85 parse, 56 TestParser, 55 parse, 56 TestParser.hpp, 76 toString HistoryEntry, 51 TransitionTemplate, 56 TransitionTemplate, 57 Types.cc, 76 Types.hpp, 76, 78 ConditionOperator, 77 Equals, 78 FromValue, 78 FromValu	SeleneFormula, 54	
verifyLocalStates, 55 setIdent AgentTemplate, 14 setInitState AgentTemplate, 14 STATE_STATUS Verification.hpp, 88 Verification.hpp, 88 Verification.cpp, 85 parse, 56 TestParser, 55 parse, 56 TestParser.hpp, 76 toString HistoryEntry, 51 TransitionTemplate, 57 Types.cc, 76 Types.hpp, 76, 78 ConditionOperator, 77 Equals, 78 FromValue, 78 F	SeleneFormula.hpp, 75	
setIdent revertLastDecision, 64 AgentTemplate, 14 setInitState undoLastHistoryEntry, 65 AgentTemplate, 14 STATE_STATUS verification.hpp, 88 Verification.hpp, 88 Verification.cpp, 85 parse, 56 TestParser, 55 parse, 56 TestParser.hpp, 76 dbgVerifStatus, 86 toString verStatusToStr, 87 HistoryEntry, 51 VerificationTemplate, 56 TransitionTemplate, 57 Types.cc, 76 Types.hpp, 76, 78 ConditionOperator, 77 Equals, 78 FromValue, 78 FromValue, 78 FromVar, 78 revertLastDecision, 64 undoHistoryUntil, 64 undoHistoryUntil, 64 undoHistoryUntil, 64 undoHistoryEntry, 65 Verification, 60 Verification, 60 Verify, 65 VerifyclobalState, 65 VerifyclobalState, 65 Verification.cpp, 85 dbgHistEnt, 86 Condition.cpp, 85 Condition.npp, 87 Verification.hpp, 87, 89 CONTEXT, 88 FromValue, 78 NORMAL, 88 RESTORE, 88 REVERT, 88	SeleneFormula1, 55	
AgentTemplate, 14 setInitState AgentTemplate, 14 SetInitState AgentTemplate, 14 STATE_STATUS Verification.hpp, 88 Verification.hpp, 88 Verification.cpp, 85 parse, 56 TestParser.hpp, 76 toString HistoryEntry, 51 TransitionTemplate, 56 TransitionTemplate, 57 Types.cc, 76 Types.hpp, 76, 78 ConditionOperator, 77 Equals, 78 FromValue, 78 FromVar, 78 undoHistoryUntil, 64 undoLastHistoryEntry, 65 Verification, 60 verify, 65 verifyGlobalState, 65 verifyGlobalState, 65 verifyLocalStates, 65 Verification.cpp, 85 dbgHistEnt, 86 dbgVerifStatus, 86 verStatusToStr, 87 Verification.hpp, 87, 89 CONTEXT, 88 DECISION, 88 HistoryEntryType, 88 MARK_DECISION_AS_INVALID, 88 Mode, 88 RESTORE, 88 RESTORE, 88 REVERT, 88	verifyLocalStates, 55	
setInitState undoLastHistoryEntry, 65 AgentTemplate, 14 STATE_STATUS Verification.hpp, 88 Verification.hpp, 88 Verification.cpp, 85 parse, 56 TestParser, 55 parse, 56 TestParser.hpp, 76 dbgHistEnt, 86 TestParser.hpp, 76 dbgVerifStatus, 86 verStatusToStr, 87 HistoryEntry, 51 Verification.hpp, 87, 89 TransitionTemplate, 56 TransitionTemplate, 57 Types.cc, 76 Types.hpp, 76, 78 ConditionOperator, 77 Equals, 78 FromValue, 78 FromValue, 78 FromVar, 78 Verification, 60 Verification, 60 Verification, 60 Verify, 65 Verification.cpp, 85 dbgHistEnt, 86 ConditionOperator, 87 HistoryEntry, 88 Verification.hpp, 87, 89 CONTEXT, 88 FromValue, 78 RESTORE, 88 RESTORE, 88 REVERT, 88		
AgentTemplate, 14 STATE_STATUS Verification.hpp, 88 Verification.hpp, 88 Verification.cpp, 85 VerifyLocalStates, 65 VerifyLocalStates, 65 Verification.cpp, 85 parse, 56 TestParser.hpp, 76 toString HistoryEntry, 51 TransitionTemplate, 56 TransitionTemplate, 57 Types.cc, 76 Types.hpp, 76, 78 ConditionOperator, 77 Equals, 78 FromValue, 78 FromVar, 78 Verification, 60 Verification, 60 verifyLocalStates, 65 VerifyLocalStates, 65 Verification.cpp, 85 dbgHistEnt, 86 Verification.cpp, 85 Verification.pp, 87, 89 Coldition.pp, 87, 89 CONTEXT, 88 FromValue, 78 NARK_DECISION_AS_INVALID, 88 RESTORE, 88 RESTORE, 88 REVERT, 88	AgentTemplate, 14	
Verification.hpp, 88 Verification.hpp, 88 Verification.hpp, 88 Verification.cpp, 85 Verification.cpp, 85 parse, 56 TestParser.hpp, 76 toString HistoryEntry, 51 TransitionTemplate, 56 TransitionTemplate, 57 Types.cc, 76 Types.hpp, 76, 78 ConditionOperator, 77 Equals, 78 FromValue, 78 FromVar, 78 Verification.hpp, 87, 89 Verification.hpp, 87, 89 CONTEXT, 88 CONTEXT, 88 DECISION, 88 HistoryEntryType, 88 MARK_DECISION_AS_INVALID, 88 RESTORE, 88 FromVar, 78 RESTORE, 88 FromVar, 78		
Verification.hpp, 88 Verification.hpp, 88 Verification.cpp, 85 Verification.cpp, 85 parse, 56 TestParser.hpp, 76 toString HistoryEntry, 51 Verification.hpp, 87, 89 TransitionTemplate, 56 TransitionTemplate, 57 Types.cc, 76 Types.hpp, 76, 78 ConditionOperator, 77 Equals, 78 FromValue, 78 FromVar, 78 Verification.cpp, 85 dbgHistEnt, 86 dbgVerifStatus, 86 verStatus ToStr, 87 Verification.hpp, 87, 89 CONTEXT, 88 DECISION, 88 HistoryEntryType, 88 MARK_DECISION_AS_INVALID, 88 Mode, 88 RESTORE, 88 RESTORE, 88 REVERT, 88	- ·	
verifyLocalStates, 65 TestParser, 55 parse, 56 TestParser.hpp, 76 toString HistoryEntry, 51 TransitionTemplate, 56 TransitionTemplate, 57 Types.cc, 76 Types.hpp, 76, 78 ConditionOperator, 77 Equals, 78 FromValue, 78 FromVar, 78 Verification.cpp, 85 dbgHistEnt, 86 dbgVerifStatus, 86 verStatusToStr, 87 Verification.hpp, 87, 89 CONTEXT, 88 CONTEXT, 88 DECISION, 88 HistoryEntryType, 88 MARK_DECISION_AS_INVALID, 88 Mode, 88 RESTORE, 88 RESTORE, 88 REVERT, 88	_	• 1
TestParser, 55 parse, 56 dbgHistEnt, 86 TestParser.hpp, 76 dbgVerifStatus, 86 toString verStatusToStr, 87 HistoryEntry, 51 Verification.hpp, 87, 89 TransitionTemplate, 56 TransitionTemplate, 57 DECISION, 88 Types.cc, 76 Types.hpp, 76, 78 ConditionOperator, 77 Equals, 78 FromValue, 78 FromVar, 78 Verification.cpp, 85 dbgHistEnt, 86 dbgVerifStatus, 86 verStatusToStr, 87 Verification.hpp, 87, 89 CONTEXT, 88 HistoryEntryType, 88 MARK_DECISION, 88 MARK_DECISION_AS_INVALID, 88 Mode, 88 NORMAL, 88 RESTORE, 88 RESTORE, 88 REVERT, 88	Verification.hpp, 88	•
parse, 56 TestParser.hpp, 76 toString HistoryEntry, 51 TransitionTemplate, 56 TransitionTemplate, 57 Types.cc, 76 Types.hpp, 76, 78 ConditionOperator, 77 Equals, 78 FromValue, 78 FromVar, 78 dbgHistEnt, 86 dbgVerifStatus, 86 verStatusToStr, 87 Verification.hpp, 87, 89 CONTEXT, 88 CONTEXT, 88 DECISION, 88 HistoryEntryType, 88 MARK_DECISION_AS_INVALID, 88 Mode, 88 RESTORE, 88 RESTORE, 88 REVERT, 88	Toot Doroor EE	-
TestParser.hpp, 76 toString HistoryEntry, 51 Verification.hpp, 87, 89 TransitionTemplate, 56 TransitionTemplate, 57 Types.cc, 76 Types.hpp, 76, 78 ConditionOperator, 77 Equals, 78 FromValue, 78 FromVar, 78 dbgVerifStatus, 86 verStatusToStr, 87 Verification.hpp, 87, 89 CONTEXT, 88 DECISION, 88 HistoryEntryType, 88 MARK_DECISION_AS_INVALID, 88 Mode, 88 RESTORE, 88 RESTORE, 88 REVERT, 88		• •
toString verStatusToStr, 87 HistoryEntry, 51 Verification.hpp, 87, 89 TransitionTemplate, 56 TransitionTemplate, 57 Types.cc, 76 Types.hpp, 76, 78 ConditionOperator, 77 Equals, 78 FromValue, 78 FromVar, 78 Verification.hpp, 87, 89 CONTEXT, 88 DECISION, 88 HistoryEntryType, 88 MARK_DECISION_AS_INVALID, 88 Mode, 88 NORMAL, 88 RESTORE, 88 REVERT, 88	•	_
HistoryEntry, 51 Verification.hpp, 87, 89 TransitionTemplate, 56 TransitionTemplate, 57 Types.cc, 76 Types.hpp, 76, 78 ConditionOperator, 77 Equals, 78 FromValue, 78 FromVar, 78 Verification.hpp, 87, 89 CONTEXT, 88 DECISION, 88 HistoryEntryType, 88 MARK_DECISION_AS_INVALID, 88 Mode, 88 NORMAL, 88 RESTORE, 88 REVERT, 88	• •	
TransitionTemplate, 56 TransitionTemplate, 57 DECISION, 88 Types.cc, 76 HistoryEntryType, 88 Types.hpp, 76, 78 ConditionOperator, 77 Equals, 78 FromValue, 78 FromVar, 78 CONTEXT, 88 DECISION, 88 MARK_DECISION_AS_INVALID, 88 Mode, 88 NORMAL, 88 RESTORE, 88 REVERT, 88	_	
TransitionTemplate, 57 Types.cc, 76 Types.hpp, 76, 78 ConditionOperator, 77 Equals, 78 FromValue, 78 FromVar, 78 DECISION, 88 HistoryEntryType, 88 MARK_DECISION_AS_INVALID, 88 Mode, 88 NORMAL, 88 RESTORE, 88 REVERT, 88		·
Types.cc, 76 Types.hpp, 76, 78 MARK_DECISION_AS_INVALID, 88 ConditionOperator, 77 Mode, 88 Equals, 78 FromValue, 78 FromVar, 78 HistoryEntryType, 88 MARK_DECISION_AS_INVALID, 88 Mode, 88 RESTORE, 88 RESTORE, 88 REVERT, 88	•	
Types.hpp, 76, 78 ConditionOperator, 77 Equals, 78 FromValue, 78 FromVar, 78 MARK_DECISION_AS_INVALID, 88 Mode, 88 NORMAL, 88 RESTORE, 88 REVERT, 88	•	
ConditionOperator, 77 Equals, 78 FromValue, 78 FromVar, 78 Mode, 88 NORMAL, 88 RESTORE, 88 REVERT, 88	• •	
Equals, 78 NORMAL, 88 FromValue, 78 RESTORE, 88 FromVar, 78 REVERT, 88		
FromValue, 78 RESTORE, 88 FromVar, 78 REVERT, 88	·	•
FromVar, 78 REVERT, 88	·	
•		

94 INDEX

verStatusToStr, 88
VERIFIED_OK
Types.hpp, 78
verify
Verification, 65
verifyGlobalState
Verification, 65
verifyLocalStates
SeleneFormula1, 55
Verification, 65
verStatusToStr
Verification.cpp, 87
Verification.hpp, 88