MindEngine

Generated by Doxygen 1.8.11

Contents

1	Todo	o List											1
2	Nam	nespace	Index										3
	2.1	Names	space List					 	 	 	 		3
3	Hier	archica	l Index										5
	3.1	Class	Hierarchy					 	 	 	 		5
4	Clas	s Index											7
	4.1	Class	List					 	 	 	 		7
5	Nam	espace	Docume	ntation									9
	5.1	Databa	ase Name	space Refere	ence			 	 	 	 		9
		5.1.1	Detailed	Description				 	 	 	 		9
	5.2	Rules	Namespa	ce Reference)			 	 	 	 		9
		5.2.1	Detailed	Description				 	 	 	 		9
6	Clas	s Docu	mentatio	1									11
	6.1	Databa	ase::Data0	Group Class I	Reference	э		 	 	 	 		11
		6.1.1	Detailed	Description				 	 	 	 		12
		6.1.2	Construc	ctor & Destru	ctor Docu	ımenta	tion .	 	 	 	 		12
			6.1.2.1	DataGroup Sibling, Da									12
		6.1.3	Member	Function Do	cumentati	ion .		 	 	 	 		12
			6.1.3.1	getLeftMos	stChild() .			 	 	 	 		12
			6.1.3.2	isGroup() .				 	 	 	 		12

iv CONTENTS

6.2	Databa	ase::DataN	lode Class Reference	13
	6.2.1	Detailed	Description	13
	6.2.2	Construc	etor & Destructor Documentation	14
		6.2.2.1	DataNode(const std::string &identifier, DataNode *parent, DataNode *rightSibling)	14
	6.2.3	Member	Function Documentation	14
		6.2.3.1	getIdentifier()	14
		6.2.3.2	getParent()	14
		6.2.3.3	getRightSibling()	14
		6.2.3.4	isDatum()	14
		6.2.3.5	isGroup()	15
	6.2.4	Member	Data Documentation	15
		6.2.4.1	identifier	15
6.3	Databa	ase::Datun	n< T > Class Template Reference	15
	6.3.1	Detailed	Description	16
	6.3.2	Construc	etor & Destructor Documentation	16
		6.3.2.1	Datum(T value)	16
		6.3.2.2	Datum(const std::string &identifier, DataNode *parent, DataNode *rightSibling, T value)	16
	6.3.3	Member	Function Documentation	16
		6.3.3.1	getValue()	16
		6.3.3.2	isDatum()	17
		6.3.3.3	setValue(T newValue)	17
6.4	Rules:	:Match Str	uct Reference	17
	6.4.1	Detailed	Description	17
	6.4.2	Member	Function Documentation	17
		6.4.2.1	matches(const Database::DataNode *database, void *bindings)=0	17
6.5	Rules:	:Rule Clas	s Reference	18
	6.5.1	Detailed	Description	18
	6.5.2	Member	Function Documentation	18
		6.5.2.1	action()=0	18

Index

19

Todo List

Member Database::DataNode::identifier

Use identifier with strings may decrease performance with large systems due to string-matching operations. Will change this to integer or unsigned integer as soon as I find a way to match identifier with human-readable strings.

Member Rules::Rule::action ()=0

Examine the performance and reusability of using a method. If it is hard to expand, find a way to use a struct/class instead.

2 Todo List

Namespace Index

2.1 Namespace List

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

Databas	se	
Rules	Contains classes to represent Rule-based system's database, which stores knowledge available to the AI agent	9
	Contains the implementation of rules as well as the mechanism to match the rules and the data in the database	9

4 Namespace Index

Hierarchical Index

3.1 Class Hierarchy

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

Database::DataNode	13
Database::DataGroup	. 11
Database::Datum <t></t>	15
Rules::Match	17
Rules::Rule	18

6 Hierarchical Index

Class Index

4.1 Class List

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

Databas	se::DataGroup	
	Represents a non-leaf node, which contains children. Its children can be any DataNode object: either another DataGroup, or a Datum only	11
Databas	se::DataNode	
	Base class of each node in the database tree. Since every node needs an identifier, but non-leaf nodes contain their children, while leaf nodes store values	13
Databas	se::Datum< T >	
	A Datum consists of an identifier and and value. In Database's tree structure, a leaf node is a	4.5
Rules::M	Datum	10
Hulesiv	Match	
	Provides the mechanism to match the data item from the rule with any item inside the database	17
Rules::F	Rule	
	Represent a rule in a Rule-based system. A rule has two components: an if clause is going to be used to match against the database and a function to perform any action required	18

8 Class Index

Namespace Documentation

5.1 Database Namespace Reference

Contains classes to represent Rule-based system's database, which stores knowledge available to the AI agent.

Classes

class DataGroup

Represents a non-leaf node, which contains children. Its children can be any DataNode object: either another Data Group, or a Datum only.

class DataNode

Base class of each node in the database tree. Since every node needs an identifier, but non-leaf nodes contain their children, while leaf nodes store values.

· class Datum

A Datum consists of an identifier and and value. In Database's tree structure, a leaf node is a Datum.

5.1.1 Detailed Description

Contains classes to represent Rule-based system's database, which stores knowledge available to the AI agent.

5.2 Rules Namespace Reference

Contains the implementation of rules as well as the mechanism to match the rules and the data in the database.

Classes

· struct Match

Provides the mechanism to match the data item from the rule with any item inside the database.

class Rule

Represent a rule in a Rule-based system. A rule has two components: an if clause is going to be used to match against the database and a function to perform any action required.

5.2.1 Detailed Description

Contains the implementation of rules as well as the mechanism to match the rules and the data in the database.

Class Documentation

6.1 Database::DataGroup Class Reference

Represents a non-leaf node, which contains children. Its children can be any DataNode object: either another DataGroup, or a Datum only.

```
#include <DataGroup.h>
```

Inheritance diagram for Database::DataGroup:



Public Member Functions

• DataGroup ()

DataGroup default constructor.

DataGroup (const std::string &identifier, DataNode *parent, DataNode *rightSibling, DataNode *leftMost
 — Child)

DataGroup constructor.

virtual ~DataGroup ()

 \sim DataGroup destructor.

const DataNode * getLeftMostChild ()

Data nodes are put into a left-most child, right sibling tree. This function returns the pointer to the left most child of this node.

• bool isGroup ()

Allows user to check whether this node is a DataGroup or not.

12 Class Documentation

Additional Inherited Members

6.1.1 Detailed Description

Represents a non-leaf node, which contains children. Its children can be any DataNode object: either another DataGroup, or a Datum only.

See also

DataNode Datum

6.1.2 Constructor & Destructor Documentation

6.1.2.1 Database::DataGroup::DataGroup (const std::string & identifier, DataNode * parent, DataNode * rightSibling, DataNode * leftMostChild)

DataGroup constructor.

Parameters

identifier	a string parameter.
parent	a DataNode pointer representing this node's parent in tree structure.
rightSibling	a DataNode pointer representing this node's right sibling in tree.
leftMostChild	a DataNode pointer representing this node's left most child in tree.

6.1.3 Member Function Documentation

6.1.3.1 const DataNode * Database::DataGroup::getLeftMostChild ()

Data nodes are put into a left-most child, right sibling tree. This function returns the pointer to the left most child of this node.

Returns

The left most child of this data group node.

```
6.1.3.2 bool Database::DataGroup::isGroup( ) [virtual]
```

Allows user to check whether this node is a DataGroup or not.

Returns

true if this node is a DataGroup, otherwise returns false.

Reimplemented from Database::DataNode.

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

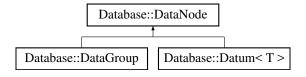
- Engine/Database/DataGroup.h
- Engine/Database/DataGroup.cpp

6.2 Database::DataNode Class Reference

Base class of each node in the database tree. Since every node needs an identifier, but non-leaf nodes contain their children, while leaf nodes store values.

#include <DataNode.h>

Inheritance diagram for Database::DataNode:



Public Member Functions

• DataNode ()

DataNode default constructor.

DataNode (const std::string &identifier, DataNode *parent, DataNode *rightSibling)

DataNode constructor with parameters.

- virtual ~DataNode ()
 - \sim DataNode destructor.
- const std::string & getIdentifier ()

The data nodes have unique identifiers.

const DataNode * getParent ()

Data nodes are put into a left-most child, right sibling tree. This function returns the pointer to the parent of this node.

const DataNode * getRightSibling ()

Data nodes are put into a left-most child, right sibling tree. This function is used to get the sibling next to current node.

• virtual bool isGroup ()

Allows user to check whether this node is a DataGroup or not.

virtual bool isDatum ()

Allows user to check whether this node is a Datum or not.

Protected Attributes

· std::string identifier

Each item of data should be identified to know what the value means.

DataNode * parent

The parent node of this node, or NULL if this node is the root of the tree.

DataNode * rightSibling

The right sibling node of this node, or NULL if this node is the right most child.

6.2.1 Detailed Description

Base class of each node in the database tree. Since every node needs an identifier, but non-leaf nodes contain their children, while leaf nodes store values.

See also

DataGroup Datum 14 Class Documentation

6.2.2 Constructor & Destructor Documentation

6.2.2.1 Database::DataNode::DataNode (const std::string & identifier, DataNode * parent, DataNode * rightSibling)

DataNode constructor with parameters.

Parameters

identifier	a string parameter.
parent	a DataNode pointer representing this node's parent in tree structure.
rightSibling	a DataNode pointer representing this node's right sibling in tree. structure

6.2.3 Member Function Documentation

6.2.3.1 const std::string & Database::DataNode::getIdentifier ()

The data nodes have unique identifiers.

Returns

The identifier of this node.

6.2.3.2 const DataNode * Database::DataNode::getParent ()

Data nodes are put into a left-most child, right sibling tree. This function returns the pointer to the parent of this node.

Returns

The parent node of this node, or NULL if this node is the root of the tree.

```
6.2.3.3 const DataNode * Database::DataNode::getRightSibling ( )
```

Data nodes are put into a left-most child, right sibling tree. This function is used to get the sibling next to current node.

Returns

The right sibling node of this node, or NULL if this node is the right most child.

6.2.3.4 bool Database::DataNode::isDatum() [virtual]

Allows user to check whether this node is a Datum or not.

Returns

true if this node is a Datum, otherwise returns false.

Reimplemented in Database::Datum< T >.

6.2.3.5 bool Database::DataNode::isGroup() [virtual]

Allows user to check whether this node is a DataGroup or not.

Returns

true if this node is a DataGroup, otherwise returns false.

Reimplemented in Database::DataGroup.

6.2.4 Member Data Documentation

6.2.4.1 std::string Database::DataNode::identifier [protected]

Each item of data should be identified to know what the value means.

Todo Use identifier with strings may decrease performance with large systems due to string-matching operations. Will change this to integer or unsigned integer as soon as I find a way to match identifier with human-readable strings.

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

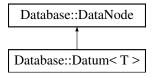
- Engine/Database/DataNode.h
- Engine/Database/DataNode.cpp

6.3 Database::Datum < T > Class Template Reference

A Datum consists of an identifier and and value. In Database's tree structure, a leaf node is a Datum.

```
#include <Datum.h>
```

Inheritance diagram for Database::Datum< T >:



Public Member Functions

• Datum (T value)

Datum constructor.

Datum (const std::string &identifier, DataNode *parent, DataNode *rightSibling, T value)

Datum constructor.

virtual ~Datum ()

 \sim Datum destructor

void setValue (T newValue)

To change the value of the Datum.

• T getValue ()

To get the value of the Datum.

· bool isDatum ()

Allows user to check whether this node is a Datum or not.

16 Class Documentation

Additional Inherited Members

6.3.1 Detailed Description

```
\label{template} \begin{split} \text{template} \! < \! \text{class T} \! > \\ \text{class Database::Datum} \! < \! \text{T} \! > \end{split}
```

A Datum consists of an identifier and and value. In Database's tree structure, a leaf node is a Datum.

See also

DataNode DataGroup

6.3.2 Constructor & Destructor Documentation

```
6.3.2.1 template < class T > Database::Datum < T >::Datum ( T value )
```

Datum constructor.

Parameters

```
value that the Datum holds.
```

 $\textbf{6.3.2.2} \quad \textbf{template} < \textbf{class T} > \textbf{Database} :: \textbf{Datum} < \textbf{T} > :: \textbf{Datum} \ (\ \textbf{const std} :: \textbf{string \& identifier}, \ \textbf{DataNode} * \textit{parent}, \\ \textbf{DataNode} * \textit{rightSibling}, \ \textbf{T} \textit{value} \)$

Datum constructor.

Parameters

identifier	a string parameter.
parent	a DataNode pointer representing this node's parent in tree structure.
rightSibling	a DataNode pointer representing this node's right sibling in tree.
value	the value that the Datum holds.

6.3.3 Member Function Documentation

```
6.3.3.1 template < class T > T Database::Datum < T >::getValue ( )
```

To get the value of the Datum.

Returns

The value that the datum is currently holding.

```
6.3.3.2 template < class T > bool Database::Datum < T >::isDatum ( ) [virtual]
```

Allows user to check whether this node is a Datum or not.

Returns

true if this node is a Datum, otherwise returns false.

Reimplemented from Database::DataNode.

```
6.3.3.3 template < class T > void Database::Datum < T >::setValue ( T newValue )
```

To change the value of the Datum.

Parameters

```
newValue The new value that is going to be assigned to the Datum.
```

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

- · Engine/Database/Datum.h
- · Engine/Database/Datum.cpp

6.4 Rules::Match Struct Reference

Provides the mechanism to match the data item from the rule with any item inside the database.

```
#include <Match.h>
```

Public Member Functions

virtual bool matches (const Database::DataNode *database, void *bindings)=0
 Check the match on the database.

6.4.1 Detailed Description

Provides the mechanism to match the data item from the rule with any item inside the database.

6.4.2 Member Function Documentation

6.4.2.1 virtual bool Rules::Match::matches (const Database::DataNode * database, void * bindings) [pure virtual]

Check the match on the database.

18 Class Documentation

Parameters

database	The database to match on.			
bindings	When part of the if clause matches a wild card, it is added to the bindings. This parameter is both			
	input and output parameter.			

Returns

true if matches, else returns false.

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

· Engine/Rules/Match.h

6.5 Rules::Rule Class Reference

Represent a rule in a Rule-based system. A rule has two components: an if clause is going to be used to match against the database and a function to perform any action required.

```
#include <Rule.h>
```

Public Member Functions

• virtual void action ()=0

The action is going to be carried out when the rule matches.

Public Attributes

Match * ifClause

Consist of a set of data items, in a similar format to those in the database.

6.5.1 Detailed Description

Represent a rule in a Rule-based system. A rule has two components: an if clause is going to be used to match against the database and a function to perform any action required.

6.5.2 Member Function Documentation

```
6.5.2.1 virtual void Rules::Rule::action ( ) [pure virtual]
```

The action is going to be carried out when the rule matches.

Todo Examine the performance and reusability of using a method. If it is hard to expand, find a way to use a struct/class instead.

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

• Engine/Rules/Rule.h

Index

matches

action	Rules::Match, 17
Rules::Rule, 18	Rules, 9
DataGroup Database::DataGroup, 12 DataNode	Rules::Match, 17 matches, 17 Rules::Rule, 18
Database::DataNode, 14 Database, 9	action, 18
Database::DataGroup, 11 DataGroup, 12 getLeftMostChild, 12 isGroup, 12	setValue Database::Datum, 17
Database::DataNode, 13 DataNode, 14 getIdentifier, 14 getParent, 14 getRightSibling, 14 identifier, 15 isDatum, 14 isGroup, 14	
Database::Datum Datum, 16 getValue, 16 isDatum, 16 setValue, 17	
Database::Datum< T >, 15	
Datum Database::Datum, 16	
getIdentifier Database::DataNode, 14 getLeftMostChild	
Database::DataGroup, 12 getParent Database::DataNode, 14	
getRightSibling Database::DataNode, 14 getValue Database::Datum, 16	
identifier Database::DataNode, 15 isDatum Database::DataNode, 14 Database::Datum, 16 isGroup	
Database::DataGroup, 12 Database::DataNode, 14	