Emotional Appraisal Asset

Generated by Doxygen 1.8.11

Tue May 10 2016 20:58:10

Contents

1 Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

2 Hierarchical Index

2.1 Class Hierarchy

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

3 Class Index

3.1 Class List

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

4 Namespace Documentation

4.1 ActionLibrary Namespace Reference

Namespaces

Classes

- interface IAction

  Interface used to represent an action execution request.

4.2 ActionLibrary.DTOs Namespace Reference

Classes

- class ActionDefinitionDTO

  Data Type Object Class for defining an Action.

4.3 EmotionalDecisionMaking Namespace Reference

Namespaces

Classes

- class EmotionalDecisionMakingAsset

  Main class of the Emotional Decision Making Asset

4.4 EmotionalDecisionMaking.DTOs Namespace Reference

Classes

- class ReactionDTO

  Data Type Object Class for defining a reactive action.

4.5 KnowledgeBase Namespace Reference

Namespaces

4.6 KnowledgeBase.Conditions Namespace Reference

Enumerations

- enum LogicalQuantifier : byte { LogicalQuantifier.Existential, LogicalQuantifier.Universal }

  Represents logical quantification modes

4.6.1 Enumeration Type Documentation

4.6.1.1 enum KnowledgeBase.Conditions.LogicalQuantifier : byte [strong]

Represents logical quantification modes

Enumerator

Existential Sets of conditions evaluated in this mode, return true if at least on possible case is considered valid.

Universal Sets of conditions evaluated in this mode, return true only if all the possible cases are considered valid.

4.7 KnowledgeBase.DTOs Namespace Reference

Namespaces

4.8 KnowledgeBase.DTOs.Conditions Namespace Reference

Classes

- class ConditionSetDTO

  Data Type Object Class for the representation of a condition set

4.9 KnowledgeBase.WellFormedNames Namespace Reference

Classes

- class Name

  Well Formed Name Class.
- class Substitution

  Represents a substitution of a variable Name for another Name object.
- class SubstitutionSet

  Class representing a set of Substitution objects.

5 Class Documentation

5.1 ActionLibrary.DTOs.ActionDefinitionDTO Class Reference

Data Type Object Class for defining an Action.

Inherited by EmotionalDecisionMaking.DTOs.ReactionDTO.

Properties

- Guid Id `[get, set]`

  The unique identifier of the action that this DTO is describing
- string Action `[get, set]`

  The action template, as a well formed string.
- string Target `[get, set]`

  The target of the action, if any.
- ConditionSetDTO Conditions `[get, set]`

  The set of conditions that must be true for this action execution.

5.1.1 Detailed Description

Data Type Object Class for defining an Action.

5.1.2 Property Documentation

5.1.2.1 string ActionLibrary.DTOs.ActionDefinitionDTO.Action `[get]`, `[set]`

The action template, as a well formed string.

Attack([type],[strength])

5.1.2.2 ConditionSetDTO ActionLibrary.DTOs.ActionDefinitionDTO.Conditions `[get]`, `[set]`

The set of conditions that must be true for this action execution.

5.1.2.3 Guid ActionLibrary.DTOs.ActionDefinitionDTO.Id [get], [set]

The unique identifier of the action that this DTO is describing

5.1.2.4 string ActionLibrary.DTOs.ActionDefinitionDTO.Target [get], [set]

The target of the action, if any.

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

- ActionDefinitionDTO.cs

5.2 ActionLibrary.IAction Interface Reference

Interface used to represent an action execution request.

Properties

- Name ActionName [get]

  The name of the action to execute
- Name Target [get]

  The target of the action, if appliable
- IList< Name > Parameters [get]

  The parameters values that the action needs in order for it to be executed. The parameter order is equal to the order of the variables in the action template, defined in the correspondent ActionDefinitionDTO.

5.2.1 Detailed Description

Interface used to represent an action execution request.

5.2.2 Property Documentation

5.2.2.1 Name ActionLibrary.IAction.ActionName [get]

The name of the action to execute

5.2.2.2 IList<Name> ActionLibrary.IAction.Parameters [get]

The parameters values that the action needs in order for it to be executed. The parameter order is equal to the order of the variables in the action template, defined in the correspondent ActionDefinitionDTO.

5.2.2.3 Name ActionLibrary.IAction.Target [get]

The target of the action, if appliable

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

- IAction.cs

5.3 EmotionalDecisionMaking.DTOs.ReactionDTO Class Reference

Data Type Object Class for defining a reactive action.

Inherits ActionLibrary.DTOs.ActionDefinitionDTO.

Properties

- float Cooldown `[get, set]`

  The cooldown time that must pass before reusing this same action, after using it once.

5.3.1 Detailed Description

Data Type Object Class for defining a reactive action.

5.3.2 Property Documentation

5.3.2.1 float EmotionalDecisionMaking.DTOs.ReactionDTO.Cooldown `[get],[set]`

The cooldown time that must pass before reusing this same action, after using it once.

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

- ReactionDTO.cs

5.4 EmotionalDecisionMaking.EmotionalDecisionMakingAsset Class Reference

Main class of the Emotional Decision Making Asset

Inherits BaseAsset.

Public Member Functions

- EmotionalDecisionMakingAsset ()

  Asset constructor. Creates a new empty Emotional Decision Making asset.
- void SaveToFile (Stream stream)

  Save the asset in a data stream
- void RegisterEmotionalAppraisalAsset (EmotionalAppraisalAsset eaa)

  Registers an Emotional Appraisal Asset to be used by this Emotional Decision Making asset.
- IEnumerable< IAction > Decide ()

  Performs the decision making process, returning the actions that the assets deems to be executed. Actual action execution is left in the responsibility of the application running this asset.
- Guid AddReaction (ReactionDTO newReaction)

  Adds a new reactive action to the asset.
- void UpdateReaction (ReactionDTO reactionToEdit, ReactionDTO newReaction)

  Updates a reaction definition.
- IEnumerable< ReactionDTO > GetAllReactions ()

Retrives the definitions of all the stored reactions.

- ReactionDTO GetReaction (Guid id)

  Retrieves the definitions of a single reaction.
- void RemoveReactions (IList< Guid > reactionsToRemove)

  Removes a set of reactions from the asset.
- void AddReactionCondition (Guid selectedReactionId, string newCondition)

  Adds a new activation condition to a reaction.
- void RemoveReactionConditions (Guid selectedReactionId, IEnumerable< string > conditionsToRemove)

  Removes a set of activation conditions from a reaction.
- void UpdateRectionCondition (Guid selectedReactionID, string conditionToEdit, string newCondition)

  Swaps a condition from a reaction for another.

Static Public Member Functions

- static EmotionalDecisionMakingAsset LoadFromFile (string filename)

  Static method used to load an Emotional Decision Making Asset state from a file.

5.4.1 Detailed Description

Main class of the Emotional Decision Making Asset

5.4.2 Constructor & Destructor Documentation

5.4.2.1 EmotionalDecisionMaking.EmotionalDecisionMakingAsset.EmotionalDecisionMakingAsset ( )

Asset constructor. Creates a new empty Emotional Decision Making asset.

5.4.3 Member Function Documentation

5.4.3.1 Guid EmotionalDecisionMaking.EmotionalDecisionMakingAsset.AddReaction (ReactionDTO *newReaction*)

Adds a new reactive action to the asset.

Parameters

newReaction	The DTO containing the parameters needed to generate a reaction.

Returns

The unique identifier of the newly created reaction.

5.4.3.2 void EmotionalDecisionMaking.EmotionalDecisionMakingAsset.AddReactionCondition (Guid *selectedReactionId,* string *newCondition*)

Adds a new activation condition to a reaction.

Parameters

selected↩ ReactionId	The unique identifier of the reaction we want to modify.
newCondition	The condition we want to add to the requested reaction.

5.4.3.3 IEnumerable<IAction> EmotionalDecisionMaking.EmotionalDecisionMakingAsset.Decide ( )

Performs the decision making process, returning the actions that the assets deems to be executed. Actual action execution is left in the responsibility of the application running this asset.

Returns

The set of actions that the assets wants to execute

Exceptions

Exception	Thrown if there is no Emotional Appraisal Asset registed in this asset.

5.4.3.4 IEnumerable<ReactionDTO> EmotionalDecisionMaking.EmotionalDecisionMakingAsset.GetAllReactions ( )

Retrives the definitions of all the stored reactions.

Returns

A set of DTOs containing the data of all reactions.

5.4.3.5 ReactionDTO EmotionalDecisionMaking.EmotionalDecisionMakingAsset.GetReaction (Guid *id*)

Retrieves the definitions of a single reaction.

Parameters

id	The unique identifier of the reaction to retrieve.

Returns

The DTO containing the data of the requested action, or null if no reaction with the given id was found.

5.4.3.6 static EmotionalDecisionMakingAsset EmotionalDecisionMaking.EmotionalDecisionMakingAsset.LoadFromFile (string *filename*) [static]

Static method used to load an Emotional Decision Making Asset state from a file.

Parameters

filename	The file path from which to load the asset.

Returns

The loaded instance of a Emotional Decision Making Asset.

5.4.3.7 void EmotionalDecisionMaking.EmotionalDecisionMakingAsset.RegisterEmotionalAppraisalAsset (EmotionalAppraisalAsset *eaa*)

Registers an Emotional Appraisal Asset to be used by this Emotional Decision Making asset.

To understand Emotional Appaisal Asset functionalities, please refer to its documentation.

Parameters

eaa	The instance of an Emotional Appaisal Asset to regist in this asset.

5.4.3.8 void EmotionalDecisionMaking.EmotionalDecisionMakingAsset.RemoveReactionConditions (Guid *selectedReactionId,* IEnumerable< string > *conditionsToRemove*)

Removes a set of activation conditions from a reaction.

Parameters

selectedReactionId	The unique identifier of the reaction we want to modify.
conditionsToRemove	The condition we want to remove from the requested reaction.

5.4.3.9 void EmotionalDecisionMaking.EmotionalDecisionMakingAsset.RemoveReactions (IList< Guid > *reactionsToRemove*)

Removes a set of reactions from the asset.

Parameters

reactionsToRemove	A set of unique identifiers of the reactions we want to remove.

5.4.3.10 void EmotionalDecisionMaking.EmotionalDecisionMakingAsset.SaveToFile (Stream *stream*)

Save the asset in a data stream

Parameters

stream	the stream to which to save the asset

5.4.3.11 void EmotionalDecisionMaking.EmotionalDecisionMakingAsset.UpdateReaction (ReactionDTO *reactionToEdit,* ReactionDTO *newReaction*)

Updates a reaction definition.

Parameters

reactionToEdit	The DTO of the reaction we want to update
newReaction	The DTO containing the new reaction data

5.4.3.12 void EmotionalDecisionMaking.EmotionalDecisionMakingAsset.UpdateRectionCondition (Guid *selectedReactionID,* string *conditionToEdit,* string *newCondition*)

Swaps a condition from a reaction for another.

Parameters

selectedReactionID	The unique identifier of the reaction we want to modify.
conditionToEdit	The condition of the reaction we want to be substituted.
newCondition	The new condition we want the reaction to have.

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

- EmotionalDecisionMakingAsset.cs

5.5 KnowledgeBase.DTOs.Conditions.ConditionSetDTO Class Reference

Data Type Object Class for the representation of a condition set

Public Attributes

- string[ ] ConditionSet
  The conditions to be evaluated as a single set.

Properties

- LogicalQuantifier Quantifier `[get, set]`
  The logical quantifier of this condition set. Used to change how the entier condition set is evaluated.

5.5.1 Detailed Description

Data Type Object Class for the representation of a condition set

5.5.2 Member Data Documentation

5.5.2.1 string [ ] KnowledgeBase.DTOs.Conditions.ConditionSetDTO.ConditionSet

The conditions to be evaluated as a single set.

5.5.3 Property Documentation

5.5.3.1 LogicalQuantifier KnowledgeBase.DTOs.Conditions.ConditionSetDTO.Quantifier [get], [set]

The logical quantifier of this condition set. Used to change how the entier condition set is evaluated.

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

- ConditionSetDTO.cs

5.6 KnowledgeBase.WellFormedNames.Name Class Reference

Well Formed Name Class.

Inherits IGroundable< Name >, IComparable< Name >, IPerspective< Name >, and ICloneable.

Public Member Functions

- abstract Name GetFirstTerm ()

  Returns the first term of this Name. Primitive and Variable Names will always return them selfs.
- abstract IEnumerable< Name > GetTerms ()

  Return all terms contained inside this Name.
- abstract Name GetNTerm (int index)

  Return the term at the specified index.
- abstract IEnumerable< Name > GetLiterals ()

  Generates a sequence of all Names contained inside this Name.
- abstract IEnumerable< Name > GetVariables ()

  Generates a sequence of all variables contained inside this Name.
- abstract bool HasGhostVariable ()

  Tells if this name contains a Ghost variable
- abstract bool HasSelf ()

  Tells if this name contains a SELF primitive.
- bool ContainsVariable (Name variable)

  Verifies if a specific variable is contained inside this Name.
- Name ApplyPerspective (Name name)

  Swaps every instance of the given Name with the SELF primitive.
- Name RemovePerspective (Name name)

  Swaps every instance of the SELF primitive with the given Name.
- abstract Name SwapPerspective (Name original, Name newName)

  Swaps every instance of the requested Name with another.
- abstract Name MakeGround (SubstitutionSet bindings)

  Given a SubstitutionSet, tries to ground this Name by substituting every variable with the corresponding value.
- abstract Name ReplaceUnboundVariables (string id)

  Adds a tag to the end of every variable inside this Name, effectively modifying their identifier.
- abstract Name RemoveBoundedVariables (string id)

  Removes a tag from the end of every variable inside this Name, effectively modifying their identifier.
- abstract object Clone ()

  Clones this Name, returning an equal copy. If this clone is changed afterwards, the original object remains the same.
- abstract bool Match (Name name)

  Determines if this matches the given name template. Both Names are matched to each other if all their Symbols are equal to one another or if a Symbol matches a universal Symbol.
- abstract Name ApplyToTerms (Func< Name, Name > transformFunction)

  Apply a transformation function to this Name.

Static Public Member Functions

- static Name GenerateUniqueGhostVariable ()

  Creates a new Name, representing a variable without a proper human readable identifier. Usefull to create temporary substitution variables.
- static operator Name (string definition)

  Explicit cast from a string to a Name. Similar from calling Name.Build(string)
- static bool operator== (Name n1, Name n2)

  Name comparison operator. Tells if two names are equal to one another.
- static bool operator!= (Name n1, Name n2)

  Name comparison operator. Tells if two names are diferent from one another.
- static Name BuildName (Name rootTerm, Name firstTerm, params Name[] otherTerms)

  Creates a composed Name, using two or more Names
- static Name BuildName (IEnumerable< Name > terms)

  Creates a Name, using a sequence of Names.
- static Name BuildName (string str)

  Creates a new Name instance by parsing a string.

Public Attributes

- const string NIL_STRING = "-"

  *The string representation of a **NIL** value Name.*
- const string SELF_STRING = "SELF"

  *The string representation of the **"SELF"** primitive Name.*
- const string UNIVERSAL_STRING = "*"

  The string representation of the Universal matching Name.
- readonly bool IsUniversal

  Tells if this is name the Universal Matching Symbol
- readonly bool IsConstant

  Tells if this name does not contain universal or variable Symbols
- readonly bool IsVariable

  Tells if this name is a variable definition
- readonly bool IsPrimitive

  Tells if this name is a primitive value
- readonly bool IsComposed

  Tells if this name is a composition of other names

Static Public Attributes

- static readonly Name NIL_SYMBOL

  A constant containing an instance of a NIL Name
- static readonly Name SELF_SYMBOL

  A constant containing an instance of a SELF Name
- static readonly Name UNIVERSAL_SYMBOL

  A constant containing an instance of a Universal matching Name

Properties

- bool IsGrounded `[get]`

  Tells if this name is grounded. A grounded Name is one that do not contain variables.
- abstract int NumberOfTerms `[get]`

  The number of terms that compose this name. Primitive and Variable Names will always return 1.

5.6.1 Detailed Description

Well Formed Name Class.

A well formed name is used to specify goal/action names, objects, properties, constants, and relations.

Its syntax is based on first order logic symbols, variables and predicates.

Names can be generated from strings, or from composition with other names. All names are case-insensitive.

Even though the Name is class type, its underlying behaviour is similar to a value type structure. This means that every modification to its values, returns a new instance of a Name object, preserving the state of the original one.

By default, Names separated in the following categories:

- Primitives
  - John
  - Dog
  - Blue
  - 34.5
- Variables
  - [x]
  - [strength]
- Composed Names
  - Color(Sky)
  - Likes(John)
  - Size(Ball)
  - Kick(Hard, Low)

5.6.2 Member Function Documentation

5.6.2.1 Name KnowledgeBase.WellFormedNames.Name.ApplyPerspective (Name *name*)

Swaps every instance of the given Name with the SELF primitive.

Parameters

name	The Name instance to swap from.

Returns

A new instance, which is a clone of this Name, but with every instance of the given Name swaped with SELF.

5.6.2.2 abstract Name KnowledgeBase.WellFormedNames.Name.ApplyToTerms (Func< Name, Name > *transformFunction*) [pure virtual]

Apply a transformation function to this Name.

The function will receive every term of this name, and should return a name to be swapped with the old one.

Parameters

transformFunction	The function we want to apply to this Name.

Returns

A new Name instance, which is the original one with the transformed function applied.

5.6.2.3 static Name KnowledgeBase.WellFormedNames.Name.BuildName (Name *rootTerm,* Name *firstTerm,* params Name[ ] *otherTerms*) [static]

Creates a composed Name, using two or more Names

Parameters

rootTerm	The Name that will be root of the composed Name.
firstTerm	The first term of the composed Name.
otherTerms	The remaining terms of the composed Name.

Exceptions

ArgumentException	Thrown if the rootTerm is not a primitive Name.

5.6.2.4 static Name KnowledgeBase.WellFormedNames.Name.BuildName (IEnumerable< Name > *terms*) [static]

Creates a Name, using a sequence of Names.

Parameters

terms	The Name set used to generate the new one.

Exceptions

ArgumentException	Thrown if the first element of the set is not a primitive Name.

5.6.2.5 static Name KnowledgeBase.WellFormedNames.Name.BuildName (string *str*) [static]

Creates a new Name instance by parsing a string.

Parameters

str	The string to parse.

Exceptions

ArgumentException	Thrown if the given string is empty.

5.6.2.6 abstract object KnowledgeBase.WellFormedNames.Name.Clone ( ) [pure virtual]

Clones this Name, returning an equal copy. If this clone is changed afterwards, the original object remains the same.

Returns

The Name's copy.

5.6.2.7 bool KnowledgeBase.WellFormedNames.Name.ContainsVariable (Name *variable*)

Verifies if a specific variable is contained inside this Name.

Parameters

variable	The variable Name we want to verify

Exceptions

ArgumentException	Thrown if the given argument is not a variable definition.

5.6.2.8 static Name KnowledgeBase.WellFormedNames.Name.GenerateUniqueGhostVariable ( ) [static]

Creates a new Name, representing a variable without a proper human readable identifier. Usefull to create temporary substitution variables.

5.6.2.9 abstract Name KnowledgeBase.WellFormedNames.Name.GetFirstTerm ( ) [pure virtual]

Returns the first term of this Name. Primitive and Variable Names will always return them selfs.

5.6.2.10 abstract IEnumerable<Name> KnowledgeBase.WellFormedNames.Name.GetLiterals ( ) [pure virtual]

Generates a sequence of all Names contained inside this Name.

5.6.2.11 abstract Name KnowledgeBase.WellFormedNames.Name.GetNTerm (int *index*) [pure virtual]

Return the term at the specified index.

Parameters

index	The zero-based index of the term to get.

Exceptions

IndexOutOfRangeException	Thrown if the given index is out of bounds.

- For Primitive or Variable Names, any index different from 0, will throw an IndexOutOfRangeException.

- Using this method with a 0 index is the same as using GetFirstTerm()

5.6.2.12 abstract IEnumerable<Name> KnowledgeBase.WellFormedNames.Name.GetTerms ( ) [pure virtual]

Return all terms contained inside this Name.

5.6.2.13 abstract IEnumerable<Name> KnowledgeBase.WellFormedNames.Name.GetVariables ( ) [pure virtual]

Generates a sequence of all variables contained inside this Name.

5.6.2.14 abstract bool KnowledgeBase.WellFormedNames.Name.HasGhostVariable ( ) [pure virtual]

Tells if this name contains a Ghost variable

GenerateUniqueGhostVariable()

5.6.2.15 abstract bool KnowledgeBase.WellFormedNames.Name.HasSelf ( ) [pure virtual]

Tells if this name contains a SELF primitive.

5.6.2.16 abstract Name KnowledgeBase.WellFormedNames.Name.MakeGround (SubstitutionSet *bindings*) [pure virtual]

Given a SubstitutionSet, tries to ground this Name by substituting every variable with the corresponding value.

Parameters

bindings	The SubstitutionSet to be used to ground this Name.

Returns

A new instance, which is a clone of this Name, but grounded as much as possible.

- If this instance is already grounded before calling this method, it will just return the same Name.
- This method does not warrant that this Name will be fully grounded, as the given SubstitutionSet may not contain the substitution variables needed to perform the task.

5.6.2.17 abstract bool KnowledgeBase.WellFormedNames.Name.Match (Name *name*) [pure virtual]

Determines if this matches the given name template. Both Names are matched to each other if all their Symbols are equal to one another or if a Symbol matches a universal Symbol.

Parameters

name	The Name to match with this instance.

Returns

True if both Names match with each other, false otherwise.

5.6.2.18 static KnowledgeBase.WellFormedNames.Name.operator Name (string *definition*) [explicit], [static]

Explicit cast from a string to a Name. Similar from calling Name.Build(string)

5.6.2.19 static bool KnowledgeBase.WellFormedNames.Name.operator!= (Name *n1*, Name *n2*) [static]

Name comparison operator. Tells if two names are diferent from one another.

5.6.2.20 static bool KnowledgeBase.WellFormedNames.Name.operator== (Name *n1*, Name *n2*) [static]

Name comparison operator. Tells if two names are equal to one another.

5.6.2.21 abstract Name KnowledgeBase.WellFormedNames.Name.RemoveBoundedVariables (string *id*) [pure virtual]

Removes a tag from the end of every variable inside this Name, effectively modifying their identifier.

Parameters

id	The tag to remove from every variable.

Returns

A new instance, which is a clone of this Name, but with every variable identifier changed in order to exclude the requested tag.

///

- If this instance is already grounded before calling this method, it will just return the same Name.
- The tag is only removed if, and only if, the variable identifier ends with the requested tag.

5.6.2.22 Name KnowledgeBase.WellFormedNames.Name.RemovePerspective (Name *name*)

Swaps every instance of the SELF primitive with the given Name.

Parameters

name	The Name instance to swap to.

Returns

A new instance, which is a clone of this Name, but with every instance of SELF swaped with the given Name.

5.6.2.23 abstract Name KnowledgeBase.WellFormedNames.Name.ReplaceUnboundVariables (string *id*) [pure virtual]

Adds a tag to the end of every variable inside this Name, effectively modifying their identifier.

Parameters

id	The tag to add to every variable.

Returns

A new instance, which is a clone of this Name, but with every variable identifier changed in order to include the new tag.

///

- If this instance is already grounded before calling this method, it will just return the same Name.

5.6.2.24 abstract Name KnowledgeBase.WellFormedNames.Name.SwapPerspective (Name *original,* Name *newName*) [pure virtual]

Swaps every instance of the requested Name with another.

Parameters

original	The Name instance to swap from.
newName	The Name instance to swap to.

Returns

A new instance, which is a clone of this Name, but with every instance of the original Name swaped with the new one.

5.6.3 Member Data Documentation

5.6.3.1 readonly bool KnowledgeBase.WellFormedNames.Name.IsComposed

Tells if this name is a composition of other names

5.6.3.2 readonly bool KnowledgeBase.WellFormedNames.Name.IsConstant

Tells if this name does not contain universal or variable Symbols

5.6.3.3 readonly bool KnowledgeBase.WellFormedNames.Name.IsPrimitive

Tells if this name is a primitive value

5.6.3.4 readonly bool KnowledgeBase.WellFormedNames.Name.IsUniversal

Tells if this is name the Universal Matching Symbol

5.6.3.5 readonly bool KnowledgeBase.WellFormedNames.Name.IsVariable

Tells if this name is a variable definition

5.6.3.6 const string KnowledgeBase.WellFormedNames.Name.NIL_STRING = "-"

The string representation of a **NIL** value Name.

5.6.3.7 const string KnowledgeBase.WellFormedNames.Name.SELF_STRING = "SELF"

The string representation of the **"SELF"** primitive Name.

5.6.3.8 const string KnowledgeBase.WellFormedNames.Name.UNIVERSAL_STRING = "*"

The string representation of the Universal matching Name.

5.6.4 Property Documentation

5.6.4.1 bool KnowledgeBase.WellFormedNames.Name.IsGrounded [get]

Tells if this name is grounded. A grounded Name is one that do not contain variables.

5.6.4.2 abstract int KnowledgeBase.WellFormedNames.Name.NumberOfTerms [get]

The number of terms that compose this name. Primitive and Variable Names will always return 1.

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

- Name.cs

5.7 KnowledgeBase.WellFormedNames.Substitution Class Reference

Represents a substitution of a variable Name for another Name object.

Inherits IVariableRenamer< Substitution >, and ICloneable.

Public Member Functions

- Substitution (Name variable, Name value)
  Substitution Constructor
- Substitution (string substitutionDefinition)
  Constructs a Substitution using a string definition
- Substitution (string variable, string value)
  Substitution Constructor
- Substitution ReplaceUnboundVariables (string id)
  Adds a tag to the end of every variable inside this Substitution, effectively modifying their identifier.
- Substitution RemoveBoundedVariables (string id)
  Removes a tag from the end of every variable inside this Substitution, effectively modifying their identifier.

Properties

- Name Variable [get]

  The Name variable to substitute.
- Name Value [get]

  The Name value to substitute the variable with.

5.7.1 Detailed Description

Represents a substitution of a variable Name for another Name object.

The variable can be substituted by any type of Name object, meaning that grounding a Name with this substitution will not guarantee a grounded Name.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 KnowledgeBase.WellFormedNames.Substitution.Substitution (Name *variable,* Name *value*)

Substitution Constructor

Parameters

variable	the variable to be replaced
value	the new value to apply in the place of the old variable

Exceptions

BadSubstitutionException	Thrown if the Name given for the variable, is not an actual variable.

5.7.2.2 KnowledgeBase.WellFormedNames.Substitution.Substitution (string *substitutionDefinition*)

Constructs a Substitution using a string definition

Parameters

substitutionDefinition	The string to be parsed as a Substitution

Exceptions

FAtiMA.Core.Exceptions.BadSubstitutionException	Thrown if the given definition is not a valid substitution

5.7.2.3 KnowledgeBase.WellFormedNames.Substitution.Substitution (string *variable,* string *value*)

Substitution Constructor

Parameters

variable	the variable to be replaced
value	the new value to apply in the place of the old variable

Exceptions

FAtiMA.Core.Exceptions.BadSubstitutionException	Thrown if the variable symbol is grounded (ie. is not a valid variable)

5.7.3 Member Function Documentation

5.7.3.1 Substitution KnowledgeBase.WellFormedNames.Substitution.RemoveBoundedVariables (string *id*)

Removes a tag from the end of every variable inside this Substitution, effectively modifying their identifier.

Parameters

id	The tag to remove from every variable.

Returns

A new instance, which is a clone of this Substitution, but with every variable identifier changed in order to exclude the requested tag.

///

- The tag is only removed if, and only if, the variable identifier ends with the requested tag.

5.7.3.2 Substitution KnowledgeBase.WellFormedNames.Substitution.ReplaceUnboundVariables (string *id*)

Adds a tag to the end of every variable inside this Substitution, effectively modifying their identifier.

Parameters

id	The tag to add to every variable.

Returns

A new instance, which is a clone of this Substitution, but with every variable identifier changed in order to include the new tag.

5.7.4 Property Documentation

5.7.4.1 Name KnowledgeBase.WellFormedNames.Substitution.Value [get]

The Name value to substitute the variable with.

5.7.4.2 Name KnowledgeBase.WellFormedNames.Substitution.Variable [get]

The Name variable to substitute.

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

- Substitution.cs

5.8 KnowledgeBase.WellFormedNames.SubstitutionSet Class Reference

Class representing a set of Substitution objects.

Inherits IEnumerable< Substitution >.

Public Member Functions

- SubstitutionSet ()

  Creates an empty SubstitutionSet
- bool Contains (Name variable)

  Tells if a given variable is contained within this SubstitutionSet.
- int Count ()

  Returns how many substitutions are in this set.
- bool AddSubstitution (Substitution substitution)

  Adds a new Substitution to this set. The adding process might fail, if the addition of the new Substitution would create a conflict.
- bool AddSubstitutions (SubstitutionSet substitutions)

  Merge a Substitution set with this one. The merging will only ocurr if no conflicts between the two sets are found.
- bool AddSubstitutions (IEnumerable< Substitution > substitutions)

  Adds a set of Substitution objects to this set. If conflics are detected, the original SubstitutionSet object in not changed.
- bool Conflicts (Substitution substitution)

  Determines if a given Substitution object will conflict with this set.
- bool Conflicts (SubstitutionSet substitutions)

  Determines if a given SubstitutionSet object will conflict with this set.
- IEnumerator< Substitution > GetEnumerator ()

  Returns the enumerator of this set.

- SubstitutionSet (params Substitution[ ] substitutions)

  Creates a new SubstitutionSet, given a set of Substitution objects.
- **SubstitutionSet** (IEnumerable< Substitution > substitutions)

Properties

- Name this[Name variable] [get]

  Gets/Sets the Name to substitute for a give variable.

5.8.1 Detailed Description

Class representing a set of Substitution objects.

SubstitutionSet objects cannot contain conflicting substitutions, like:

- [x] -> John
- [x] -> Mary

5.8.2 Constructor & Destructor Documentation

5.8.2.1 KnowledgeBase.WellFormedNames.SubstitutionSet.SubstitutionSet ( )

Creates an empty SubstitutionSet

5.8.2.2 KnowledgeBase.WellFormedNames.SubstitutionSet.SubstitutionSet (params Substitution[ ] *substitutions*)

Creates a new SubstitutionSet, given a set of Substitution objects.

Parameters

substitutions	The set of Substitution objects.

Exceptions

ArgumentException	Thrown if the given set will create substitution conflicts.

5.8.3 Member Function Documentation

5.8.3.1 bool KnowledgeBase.WellFormedNames.SubstitutionSet.AddSubstitution (Substitution *substitution*)

Adds a new Substitution to this set. The adding process might fail, if the addition of the new Substitution would create a conflict.

Parameters

substitution	The Substitution to add to this set.

Returns

True if the substitution was sucessfully added to the set. False otherwise.

If the given Substitution already exists in this set, this method will return true as if it was successfully added, but it would not produce any changes to the underlying set.

5.8.3.2 bool KnowledgeBase.WellFormedNames.SubstitutionSet.AddSubstitutions (SubstitutionSet *substitutions*)

Merge a Substitution set with this one. The merging will only ocurr if no conflicts between the two sets are found.

Parameters

substitution	The SubstitutionSet to merge with this one.

Returns

True if the substitutions was sucessfully merged. False otherwise.

5.8.3.3 bool KnowledgeBase.WellFormedNames.SubstitutionSet.AddSubstitutions (IEnumerable< **Substitution** > *substitutions*)

Adds a set of Substitution objects to this set. If conflics are detected, the original SubstitutionSet object in not changed.

Parameters

substitutions	The Substitution objects set to add.

Returns

True if all the substitutions were added to this object. False if conflics were detected.

5.8.3.4 bool KnowledgeBase.WellFormedNames.SubstitutionSet.Conflicts (**Substitution** *substitution*)

Determines if a given Substitution object will conflict with this set.

Parameters

substitution	The Substitution object to test.

5.8.3.5 bool KnowledgeBase.WellFormedNames.SubstitutionSet.Conflicts (**SubstitutionSet** *substitutions*)

Determines if a given SubstitutionSet object will conflict with this set.

Parameters

substitution	The SubstitutionSet object to test.

5.8.3.6 bool KnowledgeBase.WellFormedNames.SubstitutionSet.Contains (**Name** *variable*)

Tells if a given variable is contained within this SubstitutionSet.

Parameters

variable	The Name of variable to test.

Exceptions

ArgumentException	Thrown if the given Name is not a variable definition.

5.8.3.7 int KnowledgeBase.WellFormedNames.SubstitutionSet.Count ( )

Returns how many substitutions are in this set.

5.8.3.8 IEnumerator<Substitution> KnowledgeBase.WellFormedNames.SubstitutionSet.GetEnumerator ( )

Returns the enumerator of this set.

5.8.4 Property Documentation

5.8.4.1 Name KnowledgeBase.WellFormedNames.SubstitutionSet.this[Name variable] [get]

Gets/Sets the Name to substitute for a give variable.

Parameters

variable	The Name of the variable of the substitution value to get

Returns

The Name that will substitute the given variable Name, or null if the variable is not contained within this SubstitutionSet.

Exceptions

ArgumentException	Thrown if the given Name does not represent a variable.

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

- SubstitutionSet.cs

Index