GGTK: The GO Graph Took Kit for working with the Gene Ontology

Generated by Doxygen 1.8.11

Contents

Chapter 1

Namespace Index

1.1 Namespace List

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

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

Namespace Documentation

4.1 Accumulators Namespace Reference

The Accumulators namespace provides min, max, and average accumulators to the broader code base.

Typedefs

- typedef boost::accumulators::accumulator_set< double, boost::accumulators::stats< boost::accumulators↩ ::tag::min > > MinAccumulator

  A helper type wrapping boost accumulators.
- typedef boost::accumulators::accumulator_set< double, boost::accumulators::stats< boost::accumulators↩ ::tag::max > > MaxAccumulator

  A helper type wrapping boost accumulators.
- typedef boost::accumulators::accumulator_set< double, boost::accumulators::stats< boost::accumulators↩ ::tag::mean > > MeanAccumulator

  A helper type wrapping boost accumulators.
- typedef boost::accumulators::accumulator_set< double, boost::accumulators::stats< boost::accumulators↩ ::tag::max, boost::accumulators::tag::min, boost::accumulators::tag::mean > > SimpleAccumulator

  A helper type wrapping min, max, and mean accumulators.
- typedef boost::accumulators::accumulator_set< double, boost::accumulators::stats< boost::accumulators↩ ::tag::covariance< double, boost::accumulators::tag::covariate1 > > > CovarianceAccumulator

  A helper type wrapping the covariance accumulator.
- typedef boost::accumulators::accumulator_set< double, boost::accumulators::stats< boost::accumulators↩ ::tag::variance > > VarianceAccumulator

  A helper type wrapping the variance accumulator.

Functions

- double extractMin (const MinAccumulator &acc)

  A helper helper function to extract the min.
- double extractMax (const MaxAccumulator &acc)

  A helper helper function to extract the max.
- double extractMean (const MeanAccumulator &acc)

  A helper helper function to extract the mean.
- double extractMin (const SimpleAccumulator &acc)

An overlaoded helper helper function to extract the min.

- double extractMax (const SimpleAccumulator &acc)

  An overlaoded helper helper function to extract the max.
- double extractMean (const SimpleAccumulator &acc)

  An overlaoded helper helper function to extract the mean.
- double extractCovariance (const CovarianceAccumulator &acc)

  A helper helper function to extract the covariance.
- double extractVariance (const VarianceAccumulator &acc)

  A helper helper function to extract the variance.
- double extractSD (const VarianceAccumulator &acc)

  A helper helper function to extract the variance.

4.1.1 Detailed Description

The Accumulators namespace provides min, max, and average accumulators to the broader code base.

This namespace defines accumulator types from boost. Also provided are specific extactors that will return the accumulator's current value.

4.1.2 Typedef Documentation

4.1.2.1 typedef boost::accumulators::accumulator_set< double, boost::accumulators::stats< boost::accumulators::tag::covariance<double, boost::accumulators::tag::covariate1> > > Accumulators::CovarianceAccumulator

A helper type wrapping the covariance accumulator.

CovarianceAccumulator

4.1.2.2 typedef boost::accumulators::accumulator_set< double, boost::accumulators::stats< boost::accumulators::tag::max > > Accumulators::MaxAccumulator

A helper type wrapping boost accumulators.

MaxAccumulator

4.1.2.3 typedef boost::accumulators::accumulator_set< double, boost::accumulators::stats< boost::accumulators::tag::mean > > Accumulators::MeanAccumulator

A helper type wrapping boost accumulators.

MeanAccumulator

4.1.2.4 typedef boost::accumulators::accumulator_set< double, boost::accumulators::stats< boost::accumulators::tag::min > > Accumulators::MinAccumulator

A helper type wrapping boost accumulators.

MinAccumulator

4.1.2.5 typedef boost::accumulators::accumulator_set< double, boost::accumulators::stats< boost::accumulators::tag::max, boost::accumulators::tag::min, boost::accumulators::tag::mean > > Accumulators::SimpleAccumulator

A helper type wrapping min, max, and mean accumulators.

SimpleAccumulator

4.1.2.6 typedef boost::accumulators::accumulator_set< double, boost::accumulators::stats< boost::accumulators::tag::variance> > Accumulators::VarianceAccumulator

A helper type wrapping the variance accumulator.

VarianceAccumulator

4.1.3 Function Documentation

4.1.3.1 double Accumulators::extractCovariance (const CovarianceAccumulator & *acc*) [inline]

A helper helper function to extract the covariance.

extractCovariance

4.1.3.2 double Accumulators::extractMax (const MaxAccumulator & *acc*) [inline]

A helper helper function to extract the max.

extractMax

4.1.3.3 double Accumulators::extractMax (const SimpleAccumulator & *acc*) [inline]

An overlaoded helper helper function to extract the max.

extractMax

4.1.3.4 double Accumulators::extractMean (const MeanAccumulator & *acc*) [inline]

A helper helper function to extract the mean.

extractMean

4.1.3.5 double Accumulators::extractMean (const SimpleAccumulator & *acc*) [inline]

An overlaoded helper helper function to extract the mean.

extractMean

4.1.3.6 double Accumulators::extractMin (const MinAccumulator & *acc*) [inline]

A helper helper function to extract the min.

extractMin

4.1.3.7 double Accumulators::extractMin (const SimpleAccumulator & *acc*) [inline]

An overlaoded helper helper function to extract the min.

extractMin

4.1.3.8 double Accumulators::extractSD (const VarianceAccumulator & *acc*) [inline]

A helper helper function to extract the variance.

extractVariance

4.1.3.9 double Accumulators::extractVariance (const VarianceAccumulator & *acc*) [inline]

A helper helper function to extract the variance.

extractVariance

4.2 AppUtilities Namespace Reference

The AppUtilities namespace provides utility functions that facilitate application creation and integration.

Functions

- void parseParamFile (const std::string fname, std::map< std::string, std::string > &paramMap)

  A method for parsing a parameter file.
- bool checkParams (const std::map< std::string, std::string > &paramMap, const std::vector< std::string > &params, std::string &message)

  A method for checking the parameters loaded in a map.
- std::vector< std::string > paramList (std::string param_str)

  A method for extracting a comma separated string of paramters.
- std::vector< std::string > parseSimpleFile (const std::string fname)

  A method for parsing a simple single column file.
- std::string splitTake (const std::string &name, const char &sep, const size_t &n)

  A method to split a string and return the nth element.
- std::vector< std::string > setToVec (const boost::unordered_set< std::string > &uset)

  A method to split a string and return the nth element.
- boost::unordered_set< std::string > vecToSet (const std::vector< std::string > &vec)

  A method to split a string and return the nth element.

4.2.1 Detailed Description

The AppUtilities namespace provides utility functions that facilitate application creation and integration.

This namespace defines free functions that aid in constructing applicaitons using this tool kit. The main use for this name space is to construct and check the validity of paramaters used as intput to GO applicaitons and to provide certain container converstion (vector -> set etc.).

4.2.2 Function Documentation

4.2.2.1 bool AppUtilities::checkParams (const std::map< std::string, std::string > & *paramMap,* const std::vector< std::string > & *params,* std::string & *message*) [inline]

A method for checking the parameters loaded in a map.

This method checks the existence of specific keys in the parameter map. If all specified keys exist, true is returned. Otherwise false is returned and specific error messages are placed int the message parameter.

4.2.2.2 std::vector<std::string> AppUtilities::paramList (std::string *param_str*) [inline]

A method for extracting a comma separated string of paramters.

This method checks provides an easy method for inputing a list of madatory params to be checked with check↩ Params

4.2.2.3 void AppUtilities::parseParamFile (const std::string *fname,* std::map< std::string, std::string > & *paramMap*) [inline]

A method for parsing a parameter file.

This method parses a tab delimited parameter file and returns a map of parameters.

4.2.2.4 std::vector<std::string> AppUtilities::parseSimpleFile (const std::string *fname*) [inline]

A method for parsing a simple single column file.

This method parses a simple single column file.

4.2.2.5 std::vector< std::string > AppUtilities::setToVec (const boost::unordered_set< std::string > & *uset*) [inline]

A method to split a string and return the nth element.

This method splits a string by the given character and return the element at n.

4.2.2.6 std::string AppUtilities::splitTake (const std::string & *name*, const char & *sep*, const size_t & *n*) [inline]

A method to split a string and return the nth element.

This method splits a string by the given character and return the element at n.

4.2.2.7 boost::unordered_set< std::string > AppUtilities::vecToSet (const std::vector< std::string > & *vec*) [inline]

A method to split a string and return the nth element.

This method splits a string by the given character and return the element at n.

4.3 EnrichmentTools Namespace Reference

The EnrichmentTools namespace provides simple functions for calulating GO term enrichment.

Functions

- boost::unordered_set< std::string > getDescendantGenes (GoGraph *go, AnnotationData *data, const std::string &term)

  A method for determining which genes are annotated with the given term or a child of that term.
- double oneSidedRawPvalue_hyper (size_t sample, size_t success, size_t population, size_t test_value)

  A method for calculating the result of a hypergeometic test.
- double enrichmentSignificance (GoGraph *go, AnnotationData *data, boost::unordered_set< std::string > &genes, const std::string &term)

  A method to calculate the enrichment of a specific term in a sample of genes.

4.3.1 Detailed Description

The EnrichmentTools namespace provides simple functions for calulating GO term enrichment.

This namespace defines free functions that allow enrichment p-values to be calculated. These funcitons can serve as the foundation for more sophisticated enrichment anlayis.

4.3.2 Function Documentation

4.3.2.1 double EnrichmentTools::enrichmentSignificance (GoGraph * *go*, AnnotationData * *data*, boost::unordered_set< std::string > & *genes*, const std::string & *term*) [inline]

A method to calculate the enrichment of a specific term in a sample of genes.

This method performs a hypergeometic test of enrichment for a term given a set of genes that serves as the sample. The population is taken as all genes in the annotation database.

4.3.2.2 boost::unordered_set<std::string> EnrichmentTools::getDescendantGenes (GoGraph * *go,* AnnotationData * *data,* const std::string & *term*) [inline]

A method for determining which genes are annotated with the given term or a child of that term.

This method calculates the set of the genes annotated with a given term or transatively with a child of that term.

4.3.2.3 double EnrichmentTools::oneSidedRawPvalue_hyper (size_t *sample,* size_t *success,* size_t *population,* size_t *test_value*) [inline]

A method for calculating the result of a hypergeometic test.

This method calculates p-value of a hypergeometice test give 4 values. The sample size, n The population success K The the population size N The test value k

Answers the question: "What is probability of seeing value of k or more successes in a sample of size n, given that the population of size N contains K total successes."

4.4 GO Namespace Reference

GO namespaces.

Enumerations

- enum Onto { **BP** =0, **MF** =1, **CC** =2, **ONTO_ERROR** =3 }

  Ontology enum type.

- enum EvidenceCode {
  EXP =0, **IDA** =1, **IPI** =2, **IMP** =3,
  IGI =4, **IEP** =5, **ISS** =6, **ISO** =7,
  ISA =8, **ISM** =9, **IGC** =10, **IBA** =11,
  IBD =12, **IKR** =13, **IRD** =14, **RCA** =15,
  TAS =16, **NAS** =17, **IC** =18, **ND** =19,
  IEA =20, **NR** =21, **ECODE_ERROR** =22 }

  Evcidence Code enum type.

- enum Relationship {
  IS_A =0, **PART_OF** =1, **REGULATES** =2, **POSITIVELY_REGULATES** =3,
  NEGATIVELY_REGULATES =4, **REL_ERROR** =5 }

  Relationship codes enum.

Functions

- std::string getRootTermBP ()

  function that returns strings representing the root ontology term biological_process
- std::string getRootTermMF ()

  function that returns strings representing the root ontology term molecular_function
- std::string getRootTermCC ()

  function that returns strings representing the root ontology term cellular_component
- Onto ontologyStringToCode (std::string code)

  A method for returning the ontology code based on string.
- std::string ontologyToString (const Onto &o)

  A method for returning a human readable string from the ontology code.
- EvidenceCode evidenceStringToCode (std::string code)

  A method for converting evidence code strings to enums.
- std::string evidenceToString (const EvidenceCode &evidence)

  A method for returning a human readable string from an evidence code.
- Relationship relationshipStringToCode (std::string code)

  A method to convert relationship codes from string to enum.
- std::string relationshipToString (const Relationship &relationship)

  A method for returning a human readable string from a Relationship.

4.4.1 Detailed Description

GO namespaces.

This namespace is a set of static variables related to go terms and relationships.

4.4.2 Enumeration Type Documentation

4.4.2.1 enum GO::EvidenceCode

Evcidence Code enum type.

This enum defines a type for evidence codes. Defined at http://www.geneontology.org/GO.↩evidence.shtml

4.4.2.2 enum GO::Onto

Ontology enum type.

This enum defines a type for sub-ontologies.

4.4.2.3 enum GO::Relationship

Relationship codes enum.

This enum represents the relationship codes for ontology edges.

4.4.3 Function Documentation

4.4.3.1 EvidenceCode GO::evidenceStringToCode (std::string *code*) [inline]

A method for converting evidence code strings to enums.

This method takes a string representing the evidence code and converts it to an enum.

4.4.3.2 std::string GO::evidenceToString (const EvidenceCode & *evidence*) [inline]

A method for returning a human readable string from an evidence code.

This method takes an evidence code enum value and returns a string

4.4.3.3 Onto GO::ontologyStringToCode (std::string *code*) [inline]

A method for returning the ontology code based on string.

This method takes a string and returns the proper enum

4.4.3.4 std::string GO::ontologyToString (const Onto & *o*) [inline]

A method for returning a human readable string from the ontology code.

This method takes an ontology enum value and returns a string

4.4.3.5 Relationship GO::relationshipStringToCode (std::string *code*) [inline]

A method to convert relationship codes from string to enum.

This method converts the string representation of a relationship to an enum.

4.4.3.6 std::string GO::relationshipToString (const Relationship & *relationship*) [inline]

A method for returning a human readable string from a Relationship.

This method takes an evidence code enum pointer value and returns a string

4.5 SetUtilities Namespace Reference

The SetUtilities namespace provides useful opperations on generic boost::unordered_set.

4.5.1 Detailed Description

The SetUtilities namespace provides useful opperations on generic boost::unordered_set.

SetUtilities provides useful operations on boost::unordered_set. set_intersection is optimized to traverse the smaller set and test exisitence in the larger set.

Chapter 5

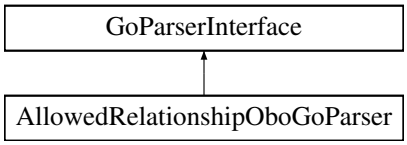
Class Documentation

5.1 AllowedRelationshipOboGoParser Class Reference

A class to parse only a specified set of relationships.

`#include <ggtk/AllowedRelationshipOboGoParser.hpp>`

Inheritance diagram for AllowedRelationshipOboGoParser:



Public Member Functions

- GoGraph * parseGoFile (std::string filename)

  Method to parse the go file, should be an OBO file.
- bool isFileGood (const std::string &filename)

  A method to test if a file fits the accepted format.
- void splitWith (const std::string &instr, const std::string &splitStr, std::string &attr, std::string &value)

  a helper method
- GoParserInterface * clone ()

  a method to create a new instance of this class for use in a factory
- void setPolicy (RelationshipPolicyInterface *policy)

  a method to set the policy
- AllowedRelationshipOboGoParser (RelationshipPolicyInterface *policy)

  A parameterized constructor.

5.1.1 Detailed Description

A class to parse only a specified set of relationships.

This class will read a Gene Ontology OBO file and add only those relationship which are specified to the graph. The most important method of this class if the parseGoFile which takes the file name as a parameter.

Implements GoParserInterface

5.1.2 Constructor & Destructor Documentation

5.1.2.1 AllowedRelationshipOboGoParser::AllowedRelationshipOboGoParser (RelationshipPolicyInterface * *policy*) [inline]

A parameterized constructor.

constructor that sets the policy

5.1.3 Member Function Documentation

5.1.3.1 GoParserInterface* AllowedRelationshipOboGoParser::clone ( ) [inline], [virtual]

a method to create a new instance of this class for use in a factory

creats a new pointer to the parser, used by the factory for go parsers.

Implements GoParserInterface.

5.1.3.2 bool AllowedRelationshipOboGoParser::isFileGood (const std::string & *filename*) [inline], [virtual]

A method to test if a file fits the accepted format.

Returns true if the file matches accepted format, false otherwise

Implements GoParserInterface.

5.1.3.3 GoGraph* AllowedRelationshipOboGoParser::parseGoFile (std::string *filename*) [inline], [virtual]

Method to parse the go file, should be an OBO file.

This method will read a Gene Ontology OBO file and add only those relationship which are specified to the graph.

Implements GoParserInterface.

5.1.3.4 void AllowedRelationshipOboGoParser::setPolicy (RelationshipPolicyInterface * *policy*) [inline]

a method to set the policy

sets the policy of the parser

5.1.3.5 void AllowedRelationshipOboGoParser::splitWith (const std::string & *instr,* const std::string & *splitStr,* std::string & *attr,* std::string & *value*) [inline]

a helper method

splits strings on the given string pattern, splitStr.

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

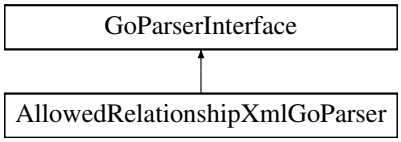
- ggtk/AllowedRelationshipOboGoParser.hpp

5.2 AllowedRelationshipXmlGoParser Class Reference

A class to parse only a specified set of relationships.

`#include <ggtk/AllowedRelationshipXmlGoParser.hpp>`

Inheritance diagram for AllowedRelationshipXmlGoParser:



Public Member Functions

- GoGraph * parseGoFile (std::string filename)

  Method to parse the go file, should be an XML file.
- bool isFileGood (const std::string &filename)

  A method to test if a file fits the accepted format.
- GoParserInterface * clone ()

  a method to create a new instance of this class for use in a factory
- void setPolicy (RelationshipPolicyInterface *policy)

  a method to set the policy
- AllowedRelationshipXmlGoParser (RelationshipPolicyInterface *policy)

  A parameterized constructor.

5.2.1 Detailed Description

A class to parse only a specified set of relationships.

This class will read a Gene Ontology XML file and add only those relationship which are specified to the graph. The most important method of this class if the parseGoFile which takes the file name as a parameter.

Implements GoParserInterface

5.2.2 Constructor & Destructor Documentation

5.2.2.1 AllowedRelationshipXmlGoParser::AllowedRelationshipXmlGoParser (RelationshipPolicyInterface * *policy*) [inline]

A parameterized constructor.

constructor that sets the policy

5.2.3 Member Function Documentation

5.2.3.1 GoParserInterface* AllowedRelationshipXmlGoParser::clone ( ) [inline], [virtual]

a method to create a new instance of this class for use in a factory

creats a new pointer to the parser, used by the factory for go parsers.

Implements GoParserInterface.

5.2.3.2 bool AllowedRelationshipXmlGoParser::isFileGood (const std::string & *filename*) [inline], [virtual]

A method to test if a file fits the accepted format.

Returns true if the file matches accepted format, false otherwise

Implements GoParserInterface.

5.2.3.3 GoGraph* AllowedRelationshipXmlGoParser::parseGoFile (std::string *filename*) [inline], [virtual]

Method to parse the go file, should be an XML file.

This method will read a Gene Ontology XML file and add only those relationship which are specified to the graph.

Implements GoParserInterface.

5.2.3.4 void AllowedRelationshipXmlGoParser::setPolicy (RelationshipPolicyInterface * *policy*) [inline]

a method to set the policy

sets the policy of the parser

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

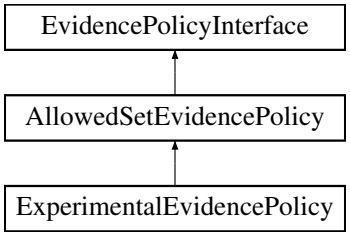
- ggtk/AllowedRelationshipXmlGoParser.hpp

5.3 AllowedSetEvidencePolicy Class Reference

A class to allow only a set of evidence codes for annotations.

`#include <ggtk/AllowedSetEvidencePolicy.hpp>`

Inheritance diagram for AllowedSetEvidencePolicy:



Public Member Functions

- AllowedSetEvidencePolicy ()

  A constructor.
- AllowedSetEvidencePolicy (std::vector< GO::EvidenceCode > evidenceCodes)

  A parameterized constructor.
- bool isAllowed (GO::EvidenceCode evidenceCode)

  a method to test if an eviddence code is allowed or not
- void addEvidence (GO::EvidenceCode evidenceCode)

  a method to add a evidence to the set of evidence codes allowed
- void addEvidence (const std::string &stringCode)

  a method to add a evidence to the set of evidence codes allowed
- bool isEmpty ()

  a method to determine if the Policy is empty

5.3.1 Detailed Description

A class to allow only a set of evidence codes for annotations.

A class to allow only certain evidence codes in the go graph. It uses a set of enums to restric the types of evidence codes considered for annotations.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 AllowedSetEvidencePolicy::AllowedSetEvidencePolicy ( ) [inline]

A constructor.

Creates the default(empty) AllowedSetEvidencePolicy

5.3.2.2 AllowedSetEvidencePolicy::AllowedSetEvidencePolicy (std::vector< GO::EvidenceCode > *evidenceCodes*) [inline]

A parameterized constructor.

Creats the AllowedSetEvidencePolicy using a list(vector) of evidence codes to allow

5.3.3 Member Function Documentation

5.3.3.1 void AllowedSetEvidencePolicy::addEvidence (GO::EvidenceCode *evidenceCode*) [inline]

a method to add a evidence to the set of evidence codes allowed

adds a evidence to the set of evidence codes allowed by setting its mapped value to true

5.3.3.2 void AllowedSetEvidencePolicy::addEvidence (const std::string & *stringCode*) [inline]

a method to add a evidence to the set of evidence codes allowed

adds a evidence to the set of evidence codes allowed by setting its mapped value to true

5.3.3.3 bool AllowedSetEvidencePolicy::isAllowed (GO::EvidenceCode *evidenceCode*) [inline], [virtual]

a method to test if an eviddence code is allowed or not

tests if the evidence is allowed. Overridden to fulfill the EvidencePolicyInterface

Implements EvidencePolicyInterface.

5.3.3.4 bool AllowedSetEvidencePolicy::isEmpty ( ) [inline]

a method to determine if the Policy is empty

Determines if the Policy is empty

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

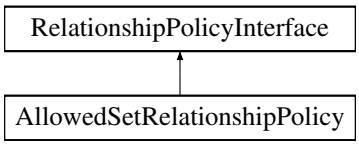
- ggtk/AllowedSetEvidencePolicy.hpp

5.4 AllowedSetRelationshipPolicy Class Reference

A class to allow only a set of relationships.

`#include <ggtk/AllowedSetRelationshipPolicy.hpp>`

Inheritance diagram for AllowedSetRelationshipPolicy:



Public Member Functions

- AllowedSetRelationshipPolicy ()
  A constructor.
- AllowedSetRelationshipPolicy (std::vector< GO::Relationship > relationships)
  A parameterized constructor.
- bool isAllowed (GO::Relationship relationship)
  a method to test if a relatinoship is allowed or not
- void addRelationship (GO::Relationship relationship)
  a method to add a relationship to the set of relationships allowed
- void addRelationship (const std::string &relString)
  a method to add a relationship to the set of relationships allowed
- bool isEmpty ()
  a method to determine if the Policy is empty

5.4.1 Detailed Description

A class to allow only a set of relationships.

A class to allow only certain relationships in the go graph. It uses a set of enums to restric the types of relationships considered in a graph.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 AllowedSetRelationshipPolicy::AllowedSetRelationshipPolicy ( ) [inline]

A constructor.

Creates the default(empty) AllowedSetRelationshipPolicy

5.4.2.2 AllowedSetRelationshipPolicy::AllowedSetRelationshipPolicy (std::vector< GO::Relationship > *relationships*) [inline]

A parameterized constructor.

Creats the AllowedSetRelationshipPolicy using a list(vector) of relationships to allow

5.4.3 Member Function Documentation

5.4.3.1 void AllowedSetRelationshipPolicy::addRelationship (GO::Relationship *relationship*) [inline]

a method to add a relationship to the set of relationships allowed

adds a relationship to the set of relationships allowed by setting its mapped value to true

5.4.3.2 void AllowedSetRelationshipPolicy::addRelationship (const std::string & *relString*) [inline]

a method to add a relationship to the set of relationships allowed

adds a relationship to the set of relationships allowed by setting its mapped value to true

5.4.3.3 bool AllowedSetRelationshipPolicy::isAllowed (GO::Relationship *relationship*) [inline], [virtual]

a method to test if a relatinoship is allowed or not

tests if the relationship is allowed. Overridden to fulfill the RelationshipPolicyInterface

Implements RelationshipPolicyInterface.

5.4.3.4 bool AllowedSetRelationshipPolicy::isEmpty ( ) [inline]

a method to determine if the Policy is empty

Determines if the Policy is empty

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

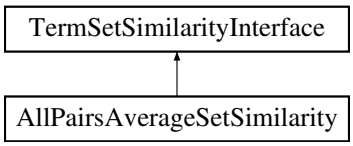
- ggtk/AllowedSetRelationshipPolicy.hpp

5.5 AllPairsAverageSetSimilarity Class Reference

A class to calculate the average similarity between all pairs of go terms for 2 sets.

`#include <ggtk/AllPairsAverageSetSimilarity.hpp>`

Inheritance diagram for AllPairsAverageSetSimilarity:



Public Member Functions

- AllPairsAverageSetSimilarity (TermSimilarityInterface *simMeasure)
  Constructor.
- double calculateSimilarity (const boost::unordered_set< std::string > &termsA, const boost::unordered_set< std::string > &termsB)
  A method for calculating term set to term set similarity for GO terms;.

5.5.1 Detailed Description

A class to calculate the average similarity between all pairs of go terms for 2 sets.

This class defines the all pairs average similarity getween two sets of terms. Put forth by Lord et al.

P. W. Lord, R. D. Stevens, A. Brass, and C. A. Goble, "Investigating semantic similarity measures across the Gene Ontology: the relationship between sequence and annotation," Bioinformatics, vol. 19, pp. 1275-83, Jul 1 2003.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 AllPairsAverageSetSimilarity::AllPairsAverageSetSimilarity (TermSimilarityInterface * *simMeasure*) [inline]

Constructor.

Creates the AllPairsAverageSetSimilarity class assigning the similarity measure private memeber.

5.5.3 Member Function Documentation

5.5.3.1 double AllPairsAverageSetSimilarity::calculateSimilarity (const boost::unordered_set< std::string > & *termsA*, const boost::unordered_set< std::string > & *termsB*) [inline], [virtual]

A method for calculating term set to term set similarity for GO terms;.

This method returns the Relevance similarity.

Implements TermSetSimilarityInterface.

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

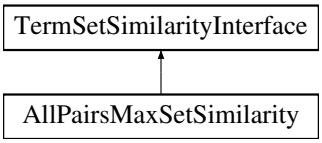
- ggtk/AllPairsAverageSetSimilarity.hpp

5.6 AllPairsMaxSetSimilarity Class Reference

A class to calculate the max similarity between all pairs of go terms for 2 sets.

`#include <ggtk/AllPairsMaxSetSimilarity.hpp>`

Inheritance diagram for AllPairsMaxSetSimilarity:



Public Member Functions

- AllPairsMaxSetSimilarity (TermSimilarityInterface *simMeasure)

  Constructor.
- double calculateSimilarity (const boost::unordered_set< std::string > &termsA, const boost::unordered_set< std::string > &termsB)

  A method for calculating term set to term set similarity for GO terms;.

5.6.1 Detailed Description

A class to calculate the max similarity between all pairs of go terms for 2 sets.

This class defines the all pairs max similarity between two sets of terms. Used by Sevilla et al.

J. L. Sevilla, V. Segura, A. Podhorski, E. Guruceaga, J. M. Mato, L. A. Martinez-Cruz, et al., "Correlation between gene expression and GO semantic similarity," IEEE/ACM Trans Comput Biol Bioinform, vol. 2, pp. 330-8, Oct-Dec 2005.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 AllPairsMaxSetSimilarity::AllPairsMaxSetSimilarity (TermSimilarityInterface * *simMeasure*) [inline]

Constructor.

Creates the AllPairsMaxSetSimilarity class assigning the similarity measure private memeber.

5.6.3 Member Function Documentation

5.6.3.1 double AllPairsMaxSetSimilarity::calculateSimilarity (const boost::unordered_set< std::string > & *termsA*, const boost::unordered_set< std::string > & *termsB*) [inline], [virtual]

A method for calculating term set to term set similarity for GO terms;.

This method returns the all pairs max similarity.

Implements TermSetSimilarityInterface.

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

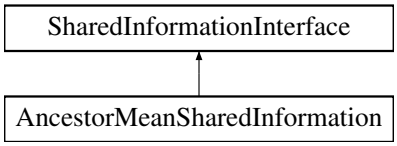
- ggtk/AllPairsMaxSetSimilarity.hpp

5.7 AncestorMeanSharedInformation Class Reference

A class to calculate shared infromation as the average information conent of all common ancestors.

`#include <ggtk/AncestorMeanSharedInformation.hpp>`

Inheritance diagram for AncestorMeanSharedInformation:



Public Member Functions

- AncestorMeanSharedInformation (GoGraph *goGraph, TermInformationContentMap &icMap)
  A constructor.
- double sharedInformation (const std::string &termA, const std::string &termB)
  A method for calculating the shared infromation between two concepts.
- double sharedInformation (const std::string &term)
  An interface method for returning the shared information of a single terms,or information content.
- double maxInformationContent (const std::string &term)
  An interface method for returning the maximum information content for a term.
- bool hasTerm (const std::string &term)
  An interface method for determining if a term can be found.
- bool isSameOntology (const std::string &termA, const std::string &termB)
  An interface method for determining if the two terms are of like ontologies.

5.7.1 Detailed Description

A class to calculate shared infromation as the average information conent of all common ancestors.

This class calculates shared infromation by averaging the information content of all common ancestors.

This shared information method is used a baseline for comparison and may not be meaningful.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 AncestorMeanSharedInformation::AncestorMeanSharedInformation (GoGraph * *goGraph,* TermInformationContentMap & *icMap*) [inline]

A constructor.

Creates the AncestorMeanSharedInformation class

5.7.3 Member Function Documentation

5.7.3.1 bool AncestorMeanSharedInformation::hasTerm (const std::string & *term*) [inline], [virtual]

An interface method for determining if a term can be found.

Determines if the term can be found in the current map.

Implements SharedInformationInterface.

5.7.3.2 bool AncestorMeanSharedInformation::isSameOntology (const std::string & *termA,* const std::string & *termB*) [inline], [virtual]

An interface method for determining if the two terms are of like ontologies.

Determine if two terms are of the same ontology.

Implements SharedInformationInterface.

5.7.3.3 double AncestorMeanSharedInformation::maxInformationContent (const std::string & *term*) [inline], [virtual]

An interface method for returning the maximum information content for a term.

This method provides the absolute max information content within a corpus for normalization purposes.

Implements SharedInformationInterface.

5.7.3.4 double AncestorMeanSharedInformation::sharedInformation (const std::string & *termA*, const std::string & *termB*) [inline], [virtual]

A method for calculating the shared infromation between two concepts.

This method returns the shared information between two concepts.

Implements SharedInformationInterface.

5.7.3.5 double AncestorMeanSharedInformation::sharedInformation (const std::string & *term*) [inline], [virtual]

An interface method for returning the shared information of a single terms,or information content.

This method privdes a mechanism for returing a term's infromation content.

Implements SharedInformationInterface.

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

- ggtk/AncestorMeanSharedInformation.hpp

5.8 AnnotationData Class Reference

A class for storing information about genes annotated with go terms.

`#include <ggtk/AnnotationData.hpp>`

Public Member Functions

- AnnotationData ()
  Class constructor.
- ~AnnotationData ()
  class descructor
- void addAssociation (const std::string &gene, const std::string &goTerm, const std::string &evidenceCode)
  A Method to add annotations to the dataset.
- bool hasGoTerm (const std::string &goTerm)
  This method tests the existence of a term in the database.
- bool hasGene (const std::string &gene)
  This method tests the existence of a gene in the database.
- std::vector< std::string > getAllGoTerms ()
  This method returns all the go terms in the database.
- std::vector< std::string > getAllGenes ()
  This method returns all genes in the database.
- std::vector< std::string > getGoTermsForGene (const std::string &gene)
  This method gets the go terms for a gene.
- boost::unordered_set< std::string > getGoTermsForGeneByOntology (const std::string &gene, GO::Onto filterOntology, GoGraph *G)
  This method gets the go terms for a gene within the specified onotlogy.
- boost::unordered_set< std::string > getGoTermsForGeneBP (const std::string &gene, GoGraph *G)

This method gets the biological process go terms for a gene.

- boost::unordered_set< std::string > getGoTermsForGeneMF (const std::string &gene, GoGraph *G)

  This method gets the molecular function go terms for a gene.
- boost::unordered_set< std::string > getGoTermsForGeneCC (const std::string &gene, GoGraph *G)

  This method gets the cellular component go terms for a gene.
- std::vector< std::string > getGoTermsEvidenceForGene (const std::string &gene)

  A method to get the evidence codes for a list of go terms.
- std::vector< std::string > getGenesForGoTerm (const std::string &goTerm)

  This method gets the genes for a go term.
- void addGenesForGoTerm (const std::string &goTerm, boost::unordered_set< std::string > &geneSet)

  This method adds the genes for a go term to a set.
- std::vector< std::string > getGenesEvidenceForGoTerm (const std::string &goTerm)

  A method to get the evidence codes for a list of genes.
- std::size_t getNumAnnotationsForGoTerm (const std::string &goTerm)

  A method to get the number of annotations of a particular go term.
- std::size_t getNumAnnotationsForGene (const std::string &gene)

  A method to get the number of annotations of a particular gene.
- std::size_t getNumGenes ()

  A helper method to get the number of genes in the db.
- std::size_t getNumGoTerms ()

  A helper method to get the number of go terms in the db.
- std::vector< std::string > getOntologyTerms (GoGraph *graph, GO::Onto ontology)

  A helper method to return only the terms of the give ontology.

Public Attributes

- std::vector< std::string > _genes

  A list of genes stored by the annotation data object.
- std::vector< std::string > _goTerms

  A list of go terms stored by the annotation data object.
- boost::unordered_map< std::string, std::size_t > _stringToGene

  A map from a gene strings to a gene index.
- boost::unordered_map< std::string, std::size_t > _stringToGo

  A map from a go term strings to a go term index.
- std::vector< std::vector< std::size_t > > _goToGenes

  A list of lists of genes, one for each go term.
- std::vector< std::vector< GO::EvidenceCode > > _goToGenesEvidence

  A list of lists of evidence codes, one for each go term. Parallel to _goToGenes.
- std::vector< std::vector< std::size_t > > _geneToGos

  A list of lists of go terms, one for each gene.
- std::vector< std::vector< GO::EvidenceCode > > _geneToGosEvidence

  A list of lists of evidence codes, one for each gene. Parallel to _geneToGos.

5.8.1 Detailed Description

A class for storing information about genes annotated with go terms.

This class hold all information about a set of annotations for genes annotated with go terms. It holds a list of genes, a list of go terms, as well as mappings from a gene to a list of go terms, and mappings from a go term to a list of annotated genes. This class allows querying go annotations and their evidence codes.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 AnnotationData::AnnotationData ( ) [inline]

Class constructor.

This constructor initialized each vector as an empty vector of the correct type.

5.8.2.2 AnnotationData::~AnnotationData ( ) [inline]

class descructor

This destructor clears all maps and vectors.

5.8.3 Member Function Documentation

5.8.3.1 void AnnotationData::addAssociation (const std::string & *gene,* const std::string & *goTerm,* const std::string & *evidenceCode*) [inline]

A Method to add annotations to the dataset.

This method adds annotations to the database. It takes a gene, a goTerm, and an evidence code. This method checks existence and indexing to remove the burden from parser implementations.

5.8.3.2 void AnnotationData::addGenesForGoTerm (const std::string & *goTerm,* boost::unordered_set< std::string > & *geneSet*) [inline]

This method adds the genes for a go term to a set.

A helper method to add genes associated to a term to a set of genes. Used in enrichment calculation

5.8.3.3 std::vector<std::string> AnnotationData::getAllGenes ( ) [inline]

This method returns all genes in the database.

A helper method to return all the genes in the databse

5.8.3.4 std::vector<std::string> AnnotationData::getAllGoTerms ( ) [inline]

This method returns all the go terms in the database.

A helper method to return all the GO terms in the databse

5.8.3.5 std::vector<std::string> AnnotationData::getGenesEvidenceForGoTerm (const std::string & *goTerm*) [inline]

A method to get the evidence codes for a list of genes.

This method returns the evidence codes for a list of genes. It parallels the getGenesForGoTerm method and is used for printing and testing.

5.8.3.6 std::vector<std::string> AnnotationData::getGenesForGoTerm (const std::string & *goTerm*) [inline]

This method gets the genes for a go term.

A helper method to return, for a go term, a list of genes as a vector of strings.

5.8.3.7 std::vector<std::string> AnnotationData::getGoTermsEvidenceForGene (const std::string & *gene*) [inline]

A method to get the evidence codes for a list of go terms.

This method returns the evidence codes for a list of go terms. It parallels the getGoTermsForGene method and is used for printing and testing.

5.8.3.8 std::vector<std::string> AnnotationData::getGoTermsForGene (const std::string & *gene*) [inline]

This method gets the go terms for a gene.

A helper method to return, for a gene, a list of go terms as a vector of strings.

5.8.3.9 boost::unordered_set<std::string> AnnotationData::getGoTermsForGeneBP (const std::string & *gene,* GoGraph * *G*) [inline]

This method gets the biological process go terms for a gene.

A helper method to return a list of BP go terms for a gene.

5.8.3.10 boost::unordered_set<std::string> AnnotationData::getGoTermsForGeneByOntology (const std::string & *gene,* GO::Onto *filterOntology,* GoGraph * *G*) [inline]

This method gets the go terms for a gene within the specified onotlogy.

A helper method to return a list of go terms as a set of strings for a gene given the sub ontology BP, MF, or CC.

5.8.3.11 boost::unordered_set<std::string> AnnotationData::getGoTermsForGeneCC (const std::string & *gene,* GoGraph * *G*) [inline]

This method gets the cellular component go terms for a gene.

A helper method to return a list of CC go terms for a gene.

5.8.3.12 boost::unordered_set<std::string> AnnotationData::getGoTermsForGeneMF (const std::string & *gene,* GoGraph * *G*) [inline]

This method gets the molecular function go terms for a gene.

A helper method to return a list of MF go terms for a gene.

5.8.3.13 std::size_t AnnotationData::getNumAnnotationsForGene (const std::string & *gene*) [inline]

A method to get the number of annotations of a particular gene.

This method returns the number of annotations for a go term. Queries the data base rather than extracting a vector.

5.8.3.14 std::size_t AnnotationData::getNumAnnotationsForGoTerm (const std::string & *goTerm*) [inline]

A method to get the number of annotations of a particular go term.

This method returns the number of annotations for a go term. Queries the data base rather than extracting a vector. Used to calculate information content.

5.8.3.15 std::size_t AnnotationData::getNumGenes ( ) [inline]

A helper method to get the number of genes in the db.

This method reutrns the size of the _genes vector.

5.8.3.16 std::size_t AnnotationData::getNumGoTerms ( ) [inline]

A helper method to get the number of go terms in the db.

This method reutrns the size of the _goTerms vector.

5.8.3.17 std::vector<std::string> AnnotationData::getOntologyTerms (GoGraph * *graph,* GO::Onto *ontology*) [inline]

A helper method to return only the terms of the give ontology.

Returns only those terms used that occur for the given ontology.

5.8.3.18 bool AnnotationData::hasGene (const std::string & *gene*) [inline]

This method tests the existence of a gene in the database.

A helper method to check if a gene exists in the database.

5.8.3.19 bool AnnotationData::hasGoTerm (const std::string & *goTerm*) [inline]

This method tests the existence of a term in the database.

A helper method to check if a term exists in the database.

5.8.4 Member Data Documentation

5.8.4.1 std::vector<std::string> AnnotationData::_genes

A list of genes stored by the annotation data object.

This storage variable stores the gene names.

5.8.4.2 std::vector<std::vector<std::size_t> > AnnotationData::_geneToGos

A list of lists of go terms, one for each gene.

This vector holds one entry for each gene. Each entry holds a list of go terms annotated to that gene.

5.8.4.3 std::vector<std::vector<GO::EvidenceCode> > AnnotationData::_geneToGosEvidence

A list of lists of evidence codes, one for each gene. Parallel to _geneToGos.

This vector holds one entry for each gene. Each entry holds a list of evidence codes for each go term annotated to that gene. It parallels the _geneToGos vectors having the same size and dimensions for each element.

5.8.4.4 std::vector<std::string> AnnotationData::_goTerms

A list of go terms stored by the annotation data object.

This storage variable stores the go terms.

5.8.4.5 std::vector<std::vector<std::size_t> > AnnotationData::_goToGenes

A list of lists of genes, one for each go term.

This vector holds one entry for each go term. Each entry holds a list of genes annotated to that go term.

5.8.4.6 std::vector<std::vector<GO::EvidenceCode> > AnnotationData::_goToGenesEvidence

A list of lists of evidence codes, one for each go term. Parallel to _goToGenes.

This vector holds one entry for each go term. Each entry holds a list of evidence codes for each gene annotated to that go term. It parallels the _goToGenes vectors having the same size and dimensions for each element.

5.8.4.7 boost::unordered_map<std::string,std::size_t> AnnotationData::_stringToGene

A map from a gene strings to a gene index.

This map accespts gene strings and returns gene indices. boost unordered_map ensures O(1) constant time find/has_key queries (hash table).

5.8.4.8 boost::unordered_map<std::string,std::size_t> AnnotationData::_stringToGo

A map from a go term strings to a go term index.

This map accespts go term strings and returns go term indices. boost unordered_map ensures O(1) constant time find/has_key queries (hash table).

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

- ggtk/AnnotationData.hpp

5.9 AnnotationParserFactory Class Reference

A class to return an instance of AnnotationParserInterface at runtime based on an argument.

`#include <ggtk/AnnotationParserFactory.hpp>`

Public Member Functions

- AnnotationParserFactory ()

  Class constructor.
- ~AnnotationParserFactory ()

  Class destructor.
- void addParser (std::string name, AnnotationParserInterface *parser)

  A Method to add a parser to the factory.
- AnnotationParserInterface * getParser (std::string name)

  A method to return a parser based on a query string.

5.9.1 Detailed Description

A class to return an instance of AnnotationParserInterface at runtime based on an argument.

This class holds a set of parser classes. When queried using the getParser method, it returns an instance of AnnotationParserInterface based on a string key. This allows parsers to be easily added to a larger system and switched at runtime.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 AnnotationParserFactory::AnnotationParserFactory ( ) [inline]

Class constructor.

This constructor initializes the private lists to empty vectors.

5.9.2.2 AnnotationParserFactory::~AnnotationParserFactory ( ) [inline]

Class destructor.

This destructor clears the names vector. It also deletes each parser pointer explicitly. Finally it clears the parser list.

5.9.3 Member Function Documentation

5.9.3.1 void AnnotationParserFactory::addParser (std::string *name,* AnnotationParserInterface * *parser*) [inline]

A Method to add a parser to the factory.

This method adds a pointer to a parser and a string to the factory. This string is used to query the appropriate parser.

5.9.3.2 AnnotationParserInterface* AnnotationParserFactory::getParser (std::string *name*) [inline]

A method to return a parser based on a query string.

If the string supplied matches one of the keys in the database, the appropriate parser will be returned. If not, a NULL pointer is returned. The calling environment must check against NULL before using.

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

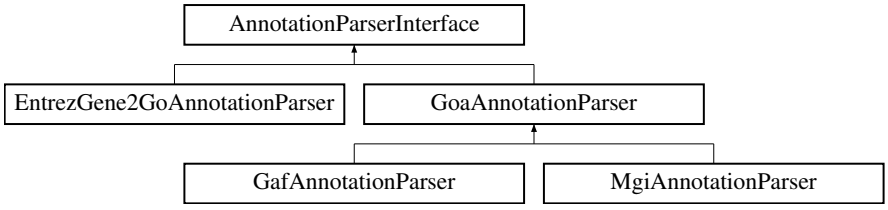
- ggtk/AnnotationParserFactory.hpp

5.10 AnnotationParserInterface Class Reference

An interface class to define annotation parsers.

`#include <ggtk/AnnotationParserInterface.hpp>`

Inheritance diagram for AnnotationParserInterface:



Public Member Functions

- virtual AnnotationData * parseAnnotationFile (std::string fileName)=0

  A pure virtual method for parsing the file and returning an AnnotationData object.
- virtual bool isFileGood (const std::string &fileName)=0

  A pure virtual method for checking if a file exists and is formatted correctly.
- virtual AnnotationParserInterface * clone ()=0

  A pure virtual clone function for factory pattern.

5.10.1 Detailed Description

An interface class to define annotation parsers.

This class defines the interface of an annotation parser. Pure virtual methods require that parsers implement these methods.

5.10.2 Member Function Documentation

5.10.2.1 virtual AnnotationParserInterface* AnnotationParserInterface::clone ( ) [pure virtual]

A pure virtual clone function for factory pattern.

This pure virtual method returns an instance of this interface. This method is used in a factory class to have the ability to decide the parser at runtime.

Implemented in GoaAnnotationParser, and EntrezGene2GoAnnotationParser.

5.10.2.2 virtual bool AnnotationParserInterface::isFileGood (const std::string & *fileName*) [pure virtual]

A pure virtual method for checking if a file exists and is formatted correctly.

This pure virtual function delegates format checking to the implementing class.

Implemented in GoaAnnotationParser, and EntrezGene2GoAnnotationParser.

5.10.2.3 virtual AnnotationData* AnnotationParserInterface::parseAnnotationFile (std::string *fileName*) [pure virtual]

A pure virtual method for parsing the file and returning an AnnotationData object.

This pure virtual method requires any parser to have a method that takes a filename string and returns an AnnotationData object pointer.

Implemented in GoaAnnotationParser, and EntrezGene2GoAnnotationParser.

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

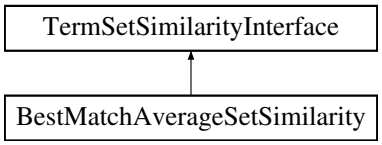
- ggtk/AnnotationParserInterface.hpp

5.11 BestMatchAverageSetSimilarity Class Reference

A class to calculate the best match average similarity between go terms for 2 sets.

`#include <ggtk/BestMatchAverageSetSimilarity.hpp>`

Inheritance diagram for BestMatchAverageSetSimilarity:



Public Member Functions

- BestMatchAverageSetSimilarity (TermSimilarityInterface *simMeasure)
  Constructor.
- double calculateSimilarity (const boost::unordered_set< std::string > &termsA, const boost::unordered_set< std::string > &termsB)
  A method for calculating term set to term set similarity for GO terms;.

5.11.1 Detailed Description

A class to calculate the best match average similarity between go terms for 2 sets.

This class defines the best match average similarity getween two sets of terms. Used by Couto et al.

F. M. Couto, M. J. Silva, and P. M. Coutinho, "Measuring semantic similarity between Gene Ontology terms," Data & Knowledge Engineering, vol. 61, pp. 137-152, Apr 2007.

5.11.2 Constructor & Destructor Documentation

5.11.2.1 BestMatchAverageSetSimilarity::BestMatchAverageSetSimilarity (TermSimilarityInterface * *simMeasure*) [inline]

Constructor.

Creates the BestMatchAverageSetSimilarity class assigning the similarity measure private memeber.

5.11.3 Member Function Documentation

5.11.3.1 double BestMatchAverageSetSimilarity::calculateSimilarity (const boost::unordered_set< std::string > & *termsA*, const boost::unordered_set< std::string > & *termsB*) [inline], [virtual]

A method for calculating term set to term set similarity for GO terms;.

This method returns the best match average similarity.

Implements TermSetSimilarityInterface.

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

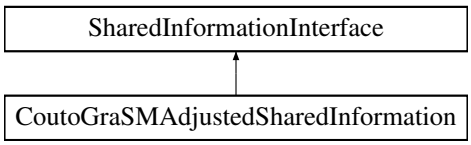
- ggtk/BestMatchAverageSetSimilarity.hpp

5.12 CoutoGraSMAdjustedSharedInformation Class Reference

A class to calculate shared infromation accross disjoint common ancetors using an adjusted algorithm.

`#include <ggtk/CoutoGraSMAdjustedSharedInformation.hpp>`

Inheritance diagram for CoutoGraSMAdjustedSharedInformation:



Public Member Functions

- CoutoGraSMAdjustedSharedInformation (GoGraph *goGraph, TermInformationContentMap &icMap)

  Constructor.
- boost::unordered_set< std::string > getCommonDisjointAncestors (const std::string &termC1, const std::string &termC2)

  Calculate disjunctive ancestors.
- bool isDisjoint (const std::string &termC, const std::string &termA1, const std::string &termA2)

  Determine if a terms are disjoint in a concept.
- std::size_t getNumPaths (const std::string &termA, const std::string &termB)

  Calculate the number of paths between two concept terms.
- double sharedInformation (const std::string &termA, const std::string &termB)

  Shared infromation between two conecepts.
- double sharedInformation (const std::string &term)

  Term information content.
- double maxInformationContent (const std::string &term)

  Maximum Ontology IC for normalization.
- bool hasTerm (const std::string &term)

  An interface method for determining if a term can be found.
- bool isSameOntology (const std::string &termA, const std::string &termB)

  An interface method for determining if the two terms are of like ontologies.

5.12.1 Detailed Description

A class to calculate shared infromation accross disjoint common ancetors using an adjusted algorithm.

This class calculates shared infromation accross disjoint common ancetors. This is a modificaiton of the original algorithm provided by Couto. The adjustment changes the contrain to path lengths to strictly greater than. See line 150.

F. M. Couto, M. J. Silva, and P. M. Coutinho, "Measuring semantic similarity between Gene Ontology terms," Data & Knowledge Engineering, vol. 61, pp. 137-152, Apr 2007.

Couto proposing calculating this value a subsituite for the IC of the MICA in calculating Resnik, Lin, and Jiang-Conrath

5.12.2 Constructor & Destructor Documentation

5.12.2.1 CoutoGraSMAdjustedSharedInformation::CoutoGraSMAdjustedSharedInformation (GoGraph * *goGraph*, TermInformationContentMap & *icMap*) [inline]

Constructor.

Creates the CoutoGraSMAdjustedSharedInformation class

5.12.3 Member Function Documentation

5.12.3.1 boost::unordered_set<std::string> CoutoGraSMAdjustedSharedInformation::getCommonDisjointAncestors (const std::string & *termC1*, const std::string & *termC2*) [inline]

Calculate disjunctive ancestors.

A method for determining common disjunctive ancestors for two terms

5.12.3.2 std::size_t CoutoGraSMAdjustedSharedInformation::getNumPaths (const std::string & *termA*, const std::string & *termB*) [inline]

Calculate the number of paths between two concept terms.

A method for calculating the number of paths from one term to another.

5.12.3.3 bool CoutoGraSMAdjustedSharedInformation::hasTerm (const std::string & *term*) [inline], [virtual]

An interface method for determining if a term can be found.

Determines if the term can be found in the current map.

Implements SharedInformationInterface.

5.12.3.4 bool CoutoGraSMAdjustedSharedInformation::isDisjoint (const std::string & *termC*, const std::string & *termA1*, const std::string & *termA2*) [inline]

Determine if a terms are disjoint in a concept.

A method for determining if, for a term c, a pair (a1,a2) is disjoint in c

5.12.3.5 bool CoutoGraSMAdjustedSharedInformation::isSameOntology (const std::string & *termA*, const std::string & *termB*) [inline], [virtual]

An interface method for determining if the two terms are of like ontologies.

Determine if two terms are of the same ontology.

Implements SharedInformationInterface.

5.12.3.6 double CoutoGraSMAdjustedSharedInformation::maxInformationContent (const std::string & *term*) [inline], [virtual]

Maximum Ontology IC for normalization.

An interface method for returning the maximum information content for a term within a corpus for normalization purposes.

Implements SharedInformationInterface.

5.12.3.7 double CoutoGraSMAdjustedSharedInformation::sharedInformation (const std::string & *termA,* const std::string & *termB*) [inline],[virtual]

Shared infromation between two conecepts.

A method for calculating the shared infromation between two concepts.

Implements SharedInformationInterface.

5.12.3.8 double CoutoGraSMAdjustedSharedInformation::sharedInformation (const std::string & *term*) [inline], [virtual]

Term information content.

An interface method to conventiently get information content of a single term

Implements SharedInformationInterface.

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

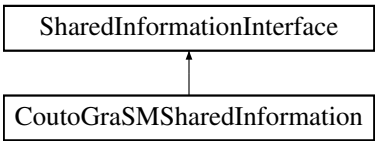
- ggtk/CoutoGraSMAdjustedSharedInformation.hpp

5.13 CoutoGraSMSharedInformation Class Reference

A class to calculate shared infromation accross disjoint common ancetors using the exact algorithm as written in the paper.

`#include <ggtk/CoutoGraSMSharedInformation.hpp>`

Inheritance diagram for CoutoGraSMSharedInformation:



Public Member Functions

- CoutoGraSMSharedInformation (GoGraph *goGraph, TermInformationContentMap &icMap)

  A constructor.
- boost::unordered_set< std::string > getCommonDisjointAncestors (const std::string &termC1, const std::string &termC2)

  A method for determining the common disjunctive ancestors.
- bool isDisjoint (const std::string &termC, const std::string &termA1, const std::string &termA2)

  A method for determining if for a term c, a pair (a1,a2) is disjoint in c.
- std::size_t getNumPaths (const std::string &termA, const std::string &termB)

  A method for calculating the number of paths for one term to another.
- double sharedInformation (const std::string &termA, const std::string &termB)

  An method for returning the shared information of two terms.
- double sharedInformation (const std::string &term)

  An interface method for returning the shared information of a single terms,or information content.
- double maxInformationContent (const std::string &term)

  An interface method for returning the maximum information content for a term.
- bool hasTerm (const std::string &term)

  An interface method for determining if a term can be found.
- bool isSameOntology (const std::string &termA, const std::string &termB)

  An interface method for determining if the two terms are of like ontologies.

5.13.1 Detailed Description

A class to calculate shared infromation accross disjoint common ancetors using the exact algorithm as written in the paper.

This class calculates shared infromation accross disjoint common ancetors.

F. M. Couto, M. J. Silva, and P. M. Coutinho, "Measuring semantic similarity between Gene Ontology terms," Data & Knowledge Engineering, vol. 61, pp. 137-152, Apr 2007.

Couto proposing calculating this value a subsituite for the IC of the MICA in calculating Resnik, Lin, and Jiang-Conrath

5.13.2 Constructor & Destructor Documentation

5.13.2.1 CoutoGraSMSharedInformation::CoutoGraSMSharedInformation (GoGraph * *goGraph,* TermInformationContentMap & *icMap*) [inline]

A constructor.

Creates the CoutoGraSMGreaterOrEqual class

5.13.3 Member Function Documentation

5.13.3.1 boost::unordered_set<std::string> CoutoGraSMSharedInformation::getCommonDisjointAncestors (const std::string & *termC1,* const std::string & *termC2*) [inline]

A method for determining the common disjunctive ancestors.

This method returns the common disjunctive ancestors for two terms

5.13.3.2 std::size_t CoutoGraSMSharedInformation::getNumPaths (const std::string & *termA,* const std::string & *termB*) [inline]

A method for calculating the number of paths for one term to another.

This method returns the number of paths between two terms

5.13.3.3 bool CoutoGraSMSharedInformation::hasTerm (const std::string & *term*) [inline], [virtual]

An interface method for determining if a term can be found.

Determines if the term can be found in the current map.

Implements SharedInformationInterface.

5.13.3.4 bool CoutoGraSMSharedInformation::isDisjoint (const std::string & *termC,* const std::string & *termA1,* const std::string & *termA2*) [inline]

A method for determining if for a term c, a pair (a1,a2) is disjoint in c.

This method returns

5.13.3.5 bool CoutoGraSMSharedInformation::isSameOntology (const std::string & *termA,* const std::string & *termB*) [inline], [virtual]

An interface method for determining if the two terms are of like ontologies.

Determine if two terms are of the same ontology.

Implements SharedInformationInterface.

5.13.3.6 double CoutoGraSMSharedInformation::maxInformationContent (const std::string & *term*) [inline], [virtual]

An interface method for returning the maximum information content for a term.

This method provides the absolute max information content within a corpus for normalization purposes.

Implements SharedInformationInterface.

5.13.3.7 double CoutoGraSMSharedInformation::sharedInformation (const std::string & *termA,* const std::string & *termB*) [inline], [virtual]

An method for returning the shared information of two terms.

This method returns the mean information content disjoint common ancestors

Implements SharedInformationInterface.

5.13.3.8 double CoutoGraSMSharedInformation::sharedInformation (const std::string & *term*) [inline], [virtual]

An interface method for returning the shared information of a single terms,or information content.

This method privdes a mechanism for returing a term's infromation content.

Implements SharedInformationInterface.

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

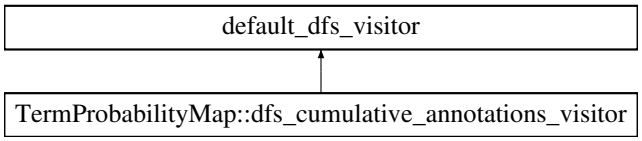
- ggtk/CoutoGraSMSharedInformation.hpp

5.14 TermProbabilityMap::dfs_cumulative_annotations_visitor Class Reference

Depth first search boost visitor.

`#include <ggtk/TermProbabilityMap.hpp>`

Inheritance diagram for TermProbabilityMap::dfs_cumulative_annotations_visitor:



Public Member Functions

- dfs_cumulative_annotations_visitor (GoGraph ∗inGraph, AnnotationData ∗inData, std::vector< std::size_t > ∗annotations, boost::unordered_map< std::string, std::size_t > ∗nameToIndex)

  A parameterized constructor passing parameters to the boost default_dfs_visitor.
- template<typename Vertex , typename Graph >
  void finish_vertex (Vertex u, const Graph &g)

  The extended method of the default_dfs_visitor, finish_vertex.

Public Attributes

- GoGraph ∗ goGraph

  The go graph object.
- AnnotationData ∗ annoData

  An AnnotationData object for accessing annotations.
- std::vector< std::size_t > ∗ annoList

  The annotaiton list to hold the and query the cummulative annotations.
- boost::unordered_map< std::string, std::size_t > ∗ nameIndexMap

  A map from name To index, initialized in this visitor.

5.14.1 Detailed Description

Depth first search boost visitor.

This defines a class used by TermProabilityMap to calculate the cumulative annotations of a term based on the true path rule. Basically this class is passed to a boost Depth first search algorithm to add the number of a child's annotaitons to the parent.

5.14.2 Member Function Documentation

5.14.2.1 template<typename Vertex , typename Graph > void TermProbabilityMap::dfs_cumulative_annotations_visitor
::finish_vertex (Vertex *u*, const Graph & *g*) [inline]

The extended method of the default_dfs_visitor, finish_vertex.

This method is called during a depth first search traversal of a graph and called when the visitor finished or leaves a node. This is the last time the algorithm touches the node.

Here we calculate the number of cummulative annotations for a term (node) by adding the current term's annotations to the annotations of the term's children.

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

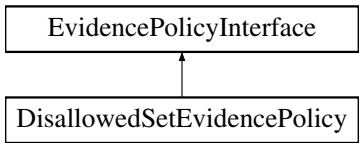
- ggtk/TermProbabilityMap.hpp

5.15 DisallowedSetEvidencePolicy Class Reference

A class to allow only a set of evidence codes for annotations.

`#include <ggtk/DisallowedSetEvidencePolicy.hpp>`

Inheritance diagram for DisallowedSetEvidencePolicy:



Public Member Functions

- DisallowedSetEvidencePolicy ()
  A constructor.
- DisallowedSetEvidencePolicy (std::vector< GO::EvidenceCode > evidenceCodes)
  A parameterized constructor.
- bool isAllowed (GO::EvidenceCode evidenceCode)
  a method to test if an eviddence code is allowed or not
- void addEvidence (GO::EvidenceCode evidenceCode)
  a method to add a evidence to the set of evidence codes not allowed
- void addEvidence (const std::string &stringCode)
  a method to add a evidence to the set of evidence codes allowed
- bool isEmpty ()
  a method to determine if the Policy is empty, disallowed set is never empty

5.15.1 Detailed Description

A class to allow only a set of evidence codes for annotations.

A class to allow only certain evidence codes in the go graph. It uses a set of enums to restric the types of evidence codes considered for annotations.

5.15.2 Constructor & Destructor Documentation

5.15.2.1 DisallowedSetEvidencePolicy::DisallowedSetEvidencePolicy ( ) [inline]

A constructor.

Creates the default(empty) DisallowedSetEvidencePolicy

5.15.2.2 DisallowedSetEvidencePolicy::DisallowedSetEvidencePolicy (std::vector< GO::EvidenceCode > *evidenceCodes*) [inline]

A parameterized constructor.

Creats the DisallowedSetEvidencePolicy using a list(vector) of evidence codes to allow

5.15.3 Member Function Documentation

5.15.3.1 void DisallowedSetEvidencePolicy::addEvidence (GO::EvidenceCode *evidenceCode*) [inline]

a method to add a evidence to the set of evidence codes not allowed

adds a evidence to the set of evidence codes not allowed

5.15.3.2 void DisallowedSetEvidencePolicy::addEvidence (const std::string & *stringCode*) [inline]

a method to add a evidence to the set of evidence codes allowed

adds a evidence to the set of evidence codes allowed by setting its mapped value to true

5.15.3.3 bool DisallowedSetEvidencePolicy::isAllowed (GO::EvidenceCode *evidenceCode*) [inline], [virtual]

a method to test if an eviddence code is allowed or not

tests if the evidence is allowed. Overridden to fulfill the EvidencePolicyInterface. This class disallows a set and so returns false if the evidence code is found.

Implements EvidencePolicyInterface.

5.15.3.4 bool DisallowedSetEvidencePolicy::isEmpty ( ) [inline]

a method to determine if the Policy is empty, disallowed set is never empty

Determines if the Policy is empty

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

- ggtk/DisallowedSetEvidencePolicy.hpp

5.16 GoGraph::EdgeProps Struct Reference

An Edge Property object.

`#include <ggtk/GoGraph.hpp>`

Public Member Functions

- ~EdgeProps ()

Public Attributes

- GO::Relationship relType

5.16.1 Detailed Description

An Edge Property object.

This struct represent the data needed by each edge. Boost provides constant time access to these members by querying them using the vertex and graph objects (graphVar[edge].relType).

5.16.2 Constructor & Destructor Documentation

5.16.2.1 GoGraph::EdgeProps::~EdgeProps ( ) [inline]

Desctructor

5.16.3 Member Data Documentation

5.16.3.1 GO::Relationship GoGraph::EdgeProps::relType

The type of relationship between the terms, is_a, part_of etc.

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

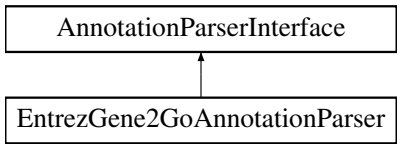
- ggtk/GoGraph.hpp

5.17 EntrezGene2GoAnnotationParser Class Reference

A class to parse an Entrez gene2go annotation file.

`#include <ggtk/EntrezGene2GoAnnotationParser.hpp>`

Inheritance diagram for EntrezGene2GoAnnotationParser:



Public Member Functions

- AnnotationData * parseAnnotationFile (std::string filename)
  An interface method for parsing an annotation file.
- bool isFileGood (const std::string &fileName)
  A method for checking if a file exists and is formatted correctly.
- AnnotationParserInterface * clone ()
  An interface method for creating a new instance of the parser.
- EntrezGene2GoAnnotationParser (EvidencePolicyInterface *policy)
  A parameterized constructor method for creating the parser with a policy.
- EntrezGene2GoAnnotationParser ()
  A default constructor method for creating the parser with the default policy.

5.17.1 Detailed Description

A class to parse an Entrez gene2go annotation file.

This class will read an Entrez gene2go file and add those annoations to an AnnotationData class. Available at: `ftp://ftp.ncbi.nih.gov/gene/DATA/`

Implements AnnotationParserInterface

5.17.2 Constructor & Destructor Documentation

5.17.2.1 EntrezGene2GoAnnotationParser::EntrezGene2GoAnnotationParser (EvidencePolicyInterface * *policy*) [inline]

A parameterized constructor method for creating the parser with a policy.

Creates the parser

5.17.2.2 EntrezGene2GoAnnotationParser::EntrezGene2GoAnnotationParser ( ) [inline]

A default constructor method for creating the parser with the default policy.

Creates the parser with the default evidence policy, everything is allowed.

5.17.3 Member Function Documentation

5.17.3.1 AnnotationParserInterface* EntrezGene2GoAnnotationParser::clone ( ) [inline], [virtual]

An interface method for creating a new instance of the parser.

This method returns a new instance of the class. This method partially fulfills the interface contract.

Implements AnnotationParserInterface.

5.17.3.2 bool EntrezGene2GoAnnotationParser::isFileGood (const std::string & *fileName*) [inline], [virtual]

A method for checking if a file exists and is formatted correctly.

This function checks that the file exists and its format can be recognized.

Implements AnnotationParserInterface.

5.17.3.3 AnnotationData* EntrezGene2GoAnnotationParser::parseAnnotationFile (std::string *filename*) [inline], [virtual]

An interface method for parsing an annotation file.

This method takes a filename as in put and returns a pointer to an AnnotationData object. This method fulfills part of the interface contract.

Implements AnnotationParserInterface.

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

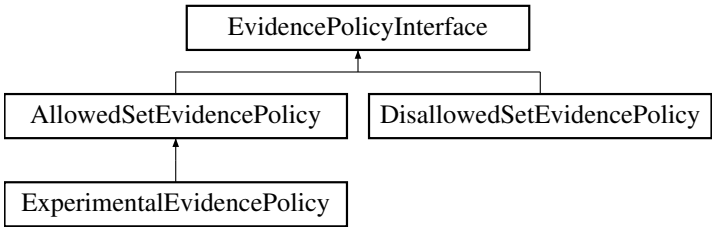
- ggtk/EntrezGene2GoAnnotationParser.hpp

5.18 EvidencePolicyInterface Class Reference

An interface to check evidence codes for GO annotations.

`#include <ggtk/EvidencePolicyInterface.hpp>`

Inheritance diagram for EvidencePolicyInterface:



Public Member Functions

- virtual bool isAllowed (GO::EvidenceCode evidenceCode)=0

  A pure virtual method to test if an evidence code is allowed.

5.18.1 Detailed Description

An interface to check evidence codes for GO annotations.

This is interface is used to create parsers which will only use a specific set of evidence codes when parsing annotations

5.18.2 Member Function Documentation

5.18.2.1 virtual bool EvidencePolicyInterface::isAllowed (GO::EvidenceCode *evidenceCode*) [pure virtual]

A pure virtual method to test if an evidence code is allowed.

This pure virtual method requires any subclass to imlement an isAllowed method to enforce the evidence pollicy.

Implemented in AllowedSetEvidencePolicy, and DisallowedSetEvidencePolicy.

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

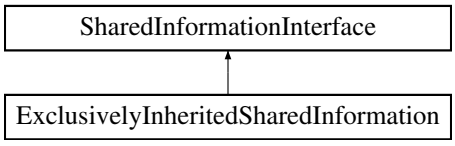
- ggtk/EvidencePolicyInterface.hpp

5.19 ExclusivelyInheritedSharedInformation Class Reference

A class to calculate shared infromation in linear time after Zhang and Lai.

```
#include <ggtk/ExclusivelyInheritedSharedInformation.hpp>
```

Inheritance diagram for ExclusivelyInheritedSharedInformation:



Public Member Functions

- ExclusivelyInheritedSharedInformation (GoGraph *goGraph, TermInformationContentMap &icMap)

  A constructor.
- boost::unordered_set< std::string > getCommonDisjointAncestors (const std::string &termC1, const std::string &termC2)

  A method for determining the common disjunctive ancestors.
- double sharedInformation (const std::string &termA, const std::string &termB)

  An method for returning the shared information of two terms.
- double sharedInformation (const std::string &term)

  An interface method for returning the shared information of a single terms,or information content.
- double maxInformationContent (const std::string &term)

  An interface method for returning the maximum information content for a term.
- bool hasTerm (const std::string &term)

  An interface method for determining if a term can be found.
- bool isSameOntology (const std::string &termA, const std::string &termB)

  An interface method for determining if the two terms are of like ontologies.

5.19.1 Detailed Description

A class to calculate shared infromation in linear time after Zhang and Lai.

Shu-Bo Zhang and Jian-Huang Lai. Semantic Similarity measurement between gene ontology terms based on exclusively inherited shared informaiton. Gene 558 (2015) 108-117.

5.19.2 Constructor & Destructor Documentation

5.19.2.1 ExclusivelyInheritedSharedInformation::ExclusivelyInheritedSharedInformation (GoGraph * *goGraph,* TermInformationContentMap & *icMap*) [inline]

A constructor.

Creates the CoutoGraSMSharedInformation class

5.19.3 Member Function Documentation

5.19.3.1 boost::unordered_set<std::string> ExclusivelyInheritedSharedInformation::getCommonDisjointAncestors (const std::string & *termC1,* const std::string & *termC2*) [inline]

A method for determining the common disjunctive ancestors.

This method returns the common disjunctive ancestors for two terms

5.19.3.2 bool ExclusivelyInheritedSharedInformation::hasTerm (const std::string & *term*) [inline], [virtual]

An interface method for determining if a term can be found.

Determines if the term can be found in the current map.

Implements SharedInformationInterface.

5.19.3.3 bool ExclusivelyInheritedSharedInformation::isSameOntology (const std::string & *termA*, const std::string & *termB*) [inline], [virtual]

An interface method for determining if the two terms are of like ontologies.

Determine if two terms are of the same ontology.

Implements SharedInformationInterface.

5.19.3.4 double ExclusivelyInheritedSharedInformation::maxInformationContent (const std::string & *term*) [inline], [virtual]

An interface method for returning the maximum information content for a term.

This method provides the absolute max information content within a corpus for normalization purposes.

Implements SharedInformationInterface.

5.19.3.5 double ExclusivelyInheritedSharedInformation::sharedInformation (const std::string & *termA*, const std::string & *termB*) [inline], [virtual]

An method for returning the shared information of two terms.

This method returns the mean information content of the frontier ancestors

Implements SharedInformationInterface.

5.19.3.6 double ExclusivelyInheritedSharedInformation::sharedInformation (const std::string & *term*) [inline], [virtual]

An interface method for returning the shared information of a single terms,or information content.

This method privdes a mechanism for returing a term's infromation content.

Implements SharedInformationInterface.

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

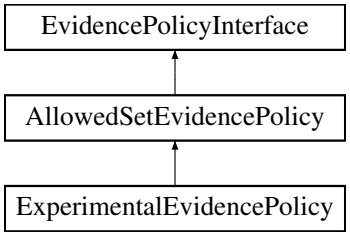
- ggtk/ExclusivelyInheritedSharedInformation.hpp

5.20 ExperimentalEvidencePolicy Class Reference

A class to allow experimental evidence codes for annotations.

`#include <ggtk/ExperimentalEvidencePolicy.hpp>`

Inheritance diagram for ExperimentalEvidencePolicy:



Public Member Functions

- ExperimentalEvidencePolicy ()

  A constructor.

5.20.1 Detailed Description

A class to allow experimental evidence codes for annotations.

A class to allow only experimental evidence codes for gene annotations. This class extends the AllowedSet↩ EvidencePolicy and adds the experimental evidence codes to the allowed set.

5.20.2 Constructor & Destructor Documentation

5.20.2.1 ExperimentalEvidencePolicy::ExperimentalEvidencePolicy ( ) [inline]

A constructor.

Creates the default(empty) AllowedSetEvidencePolicy

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

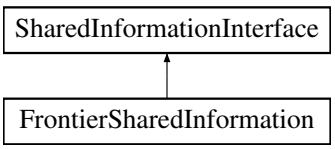
- ggtk/ExperimentalEvidencePolicy.hpp

5.21 FrontierSharedInformation Class Reference

A class to calculate shared infromation across disjoint common ancestors in linear time.

`#include <ggtk/FrontierSharedInformation.hpp>`

Inheritance diagram for FrontierSharedInformation:



Public Member Functions

- FrontierSharedInformation (GoGraph *goGraph, TermInformationContentMap &icMap)

  A constructor.
- boost::unordered_set< std::string > getCommonDisjointAncestors (const std::string &termC1, const std↩ ::string &termC2)

  A method for determining the common disjunctive ancestors.
- double sharedInformation (const std::string &termA, const std::string &termB)

  An method for returning the shared information of two terms.
- double sharedInformation (const std::string &term)

  An interface method for returning the shared information of a single terms,or information content.
- double maxInformationContent (const std::string &term)

  An interface method for returning the maximum information content for a term.
- bool hasTerm (const std::string &term)

  An interface method for determining if a term can be found.
- bool isSameOntology (const std::string &termA, const std::string &termB)

  An interface method for determining if the two terms are of like ontologies.

5.21.1 Detailed Description

A class to calculate shared infromation across disjoint common ancestors in linear time.

This class calculates shared infromation along a semantic frontier between terms.

5.21.2 Constructor & Destructor Documentation

5.21.2.1 FrontierSharedInformation::FrontierSharedInformation (GoGraph * *goGraph,* TermInformationContentMap & *icMap*) [inline]

A constructor.

Creates the CoutoGraSMSharedInformation class

5.21.3 Member Function Documentation

5.21.3.1 boost::unordered_set<std::string> FrontierSharedInformation::getCommonDisjointAncestors (const std::string & *termC1,* const std::string & *termC2*) [inline]

A method for determining the common disjunctive ancestors.

This method returns the common disjunctive ancestors for two terms

5.21.3.2 bool FrontierSharedInformation::hasTerm (const std::string & *term*) [inline], [virtual]

An interface method for determining if a term can be found.

Determines if the term can be found in the current map.

Implements SharedInformationInterface.

5.21.3.3 bool FrontierSharedInformation::isSameOntology (const std::string & *termA,* const std::string & *termB*) [inline], [virtual]

An interface method for determining if the two terms are of like ontologies.

Determine if two terms are of the same ontology.

Implements SharedInformationInterface.

5.21.3.4 double FrontierSharedInformation::maxInformationContent (const std::string & *term*) [inline], [virtual]

An interface method for returning the maximum information content for a term.

This method provides the absolute max information content within a corpus for normalization purposes.

Implements SharedInformationInterface.

5.21.3.5 double FrontierSharedInformation::sharedInformation (const std::string & *termA,* const std::string & *termB*) [inline], [virtual]

An method for returning the shared information of two terms.

This method returns the mean information content of the frontier ancestors

Implements SharedInformationInterface.

5.21.3.6 double FrontierSharedInformation::sharedInformation (const std::string & *term*) [inline], [virtual]

An interface method for returning the shared information of a single terms,or information content.

This method privdes a mechanism for returing a term's infromation content.

Implements SharedInformationInterface.

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

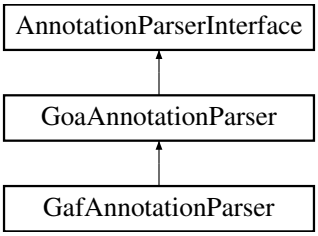
- ggtk/FrontierSharedInformation.hpp

5.22 GafAnnotationParser Class Reference

A class to parse a GO Annotation File (GAF, Format 2.0).

`#include <ggtk/GafAnnotationParser.hpp>`

Inheritance diagram for GafAnnotationParser:



Public Member Functions

- GafAnnotationParser ()

  A default constructor method for creating the parser.
- GafAnnotationParser (EvidencePolicyInterface *policy)

  A parameterized constructor method for creating the parser with a policy.

5.22.1 Detailed Description

A class to parse a GO Annotation File (GAF, Format 2.0).

This class will read a GAF file and return an AnnotationData object pointer. Defined at: http://geneontology.org/page/go-annotation-file-format-20

Implements AnnotationParserInterface.

For now, the important aspects of the GAF file and GOA file are the same. The GafAnnotationParser inherits all functionality from GoaAnnotationParser. This may change in the future.

5.22.2 Constructor & Destructor Documentation

5.22.2.1 GafAnnotationParser::GafAnnotationParser ( ) [inline]

A default constructor method for creating the parser.

Creates the parser

5.22.2.2 GafAnnotationParser::GafAnnotationParser (EvidencePolicyInterface * *policy*) [inline]

A parameterized constructor method for creating the parser with a policy.

Creates the parser with a custom policy

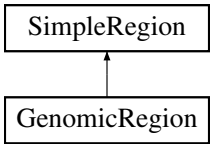
The documentation for this class was generated from the following file:

- ggtk/GafAnnotationParser.hpp

5.23 GenomicRegion Class Reference

`#include <ggtk/SimpleRegion.hpp>`

Inheritance diagram for GenomicRegion:



Public Member Functions

- GenomicRegion (const std::string chrom, const size_t start, const size_t end, const size_t id, std::string name, std::string desc)
- GenomicRegion (const size_t id, const size_t start, const size_t end)
- GenomicRegion ()
- std::string getChrom ()
- std::string getName ()
- std::string getDesc ()

Public Attributes

- char _chrom
- std::string _name
- std::string _desc

5.23.1 Detailed Description

EXPERIMENTAL: GenomicRegion class inheriting from SimpleRegion Adds a few vaibles to SimpleRegion for use in modeling genomic regions. This class is experimental and will be updated at some point in the future.

5.23.2 Constructor & Destructor Documentation

5.23.2.1 GenomicRegion::GenomicRegion (const std::string *chrom*, const size_t *start*, const size_t *end*, const size_t *id*, std::string *name*, std::string *desc*) [inline]

a parameterized constructor for the genomic region.

5.23.2.2 GenomicRegion::GenomicRegion (const size_t *id*, const size_t *start*, const size_t *end*) [inline]

a simple parametrized constructor for the genomic region

5.23.2.3 GenomicRegion::GenomicRegion ( ) [inline]

a default constructor for the genomic region

5.23.3 Member Function Documentation

5.23.3.1 std::string GenomicRegion::getChrom ( ) [inline]

returns the chromosome for the genomic region

5.23.3.2 std::string GenomicRegion::getDesc ( ) [inline]

accessor for the genomic region's description

5.23.3.3 std::string GenomicRegion::getName ( ) [inline]

accessor for the genomic region's name

5.23.4 Member Data Documentation

5.23.4.1 char GenomicRegion::_chrom

a character representing the chromosome

5.23.4.2 std::string GenomicRegion::_desc

a description for the region

5.23.4.3 std::string GenomicRegion::_name

the name or accession of the region

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

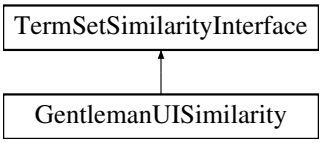
- ggtk/SimpleRegion.hpp

5.24 GentlemanUISimilarity Class Reference

A class to calculate Gentleman's UI similarity between go terms for 2 sets.

`#include <ggtk/GentlemanSimUISetSimilarity.hpp>`

Inheritance diagram for GentlemanUISimilarity:



Public Member Functions

- GentlemanUISimilarity (TermSimilarityInterface *simMeasure)

  Constructor.
- double calculateSimilarity (boost::unordered_set< std::string > termsA, boost::unordered_set< std::string > termsB)

  A method for calculating term set to term set similarity for GO terms;.

5.24.1 Detailed Description

A class to calculate Gentleman's UI similarity between go terms for 2 sets.

This is a stub.

5.24.2 Constructor & Destructor Documentation

5.24.2.1 GentlemanUISimilarity::GentlemanUISimilarity (TermSimilarityInterface * *simMeasure*) [inline]

Constructor.

Creates the GentlemanUISimilarity class assigning the similarity measure private memeber.

5.24.3 Member Function Documentation

5.24.3.1 double GentlemanUISimilarity::calculateSimilarity (boost::unordered_set< std::string > *termsA*, boost::unordered_set< std::string > *termsB*) [inline]

A method for calculating term set to term set similarity for GO terms;.

This method returns the best match average similarity.

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

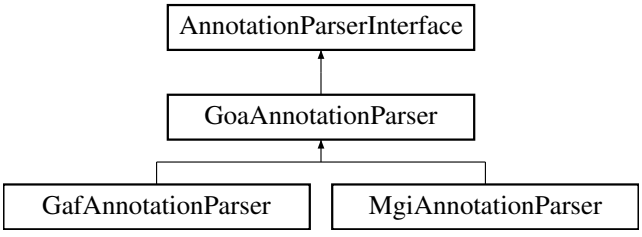
- ggtk/GentlemanSimUISetSimilarity.hpp

5.25 GoaAnnotationParser Class Reference

A class to parse a Uniprot Gene Ontolog Annotation (GOA) file.

`#include <ggtk/GoaAnnotationParser.hpp>`

Inheritance diagram for GoaAnnotationParser:



Public Member Functions

- AnnotationData * parseAnnotationFile (std::string filename)
  An interface method for parsing an annotation file.
- bool isFileGood (const std::string &fileName)
  A method for checking if a file exists and is formatted correctly.
- GoaAnnotationParser (EvidencePolicyInterface *policy)
  A parameterized constructor method for creating the parser with a policy.
- GoaAnnotationParser ()
  A default constructor method for creating the parser with a policy.
- AnnotationParserInterface * clone ()
  An interface method for creating a new instance of the parser.

5.25.1 Detailed Description

A class to parse a Uniprot Gene Ontolog Annotation (GOA) file.

This class will read a GOA file an return an AnnotationData object pointer. Defined at: http://www.ebi.ac.uk/GOA

Implements AnnotationParserInterface

5.25.2 Constructor & Destructor Documentation

5.25.2.1 GoaAnnotationParser::GoaAnnotationParser (EvidencePolicyInterface * *policy*) [inline]

A parameterized constructor method for creating the parser with a policy.

Creates the parser

5.25.2.2 GoaAnnotationParser::GoaAnnotationParser ( ) [inline]

A default constructor method for creating the parser with a policy.

Creates the parser

5.25.3 Member Function Documentation

5.25.3.1 AnnotationParserInterface* GoaAnnotationParser::clone ( ) [inline], [virtual]

An interface method for creating a new instance of the parser.

This method returns a new instance of the class. This method partially fulfills the interface contract.

Implements AnnotationParserInterface.

5.25.3.2 bool GoaAnnotationParser::isFileGood (const std::string & *fileName*) [inline], [virtual]

A method for checking if a file exists and is formatted correctly.

This function checks that the file exists and its format can be recognized.

Implements AnnotationParserInterface.

5.25.3.3 AnnotationData* GoaAnnotationParser::parseAnnotationFile (std::string *filename*) [inline], [virtual]

An interface method for parsing an annotation file.

This method takes a filename as in put and returns a pointer to an AnnotationData object. This method fulfills part of the interface contract.

Implements AnnotationParserInterface.

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

- ggtk/GoaAnnotationParser.hpp

5.26 GoGraph Class Reference

This class holds the Gene Ontology directed acyclic graph.

`#include <ggtk/GoGraph.hpp>`

Classes

- struct EdgeProps

  An Edge Property object.
- struct VertexProps

  A Vertex Property object.

Public Types

- typedef boost::adjacency_list< boost::vecS, boost::vecS, boost::bidirectionalS, boost::property< boost::vertex_index_t, size_t, VertexProps >, boost::property< boost::edge_index_t, size_t, EdgeProps > > Graph_t

  A Graph type representing Go.
- typedef boost::subgraph< Graph_t > Graph

  The main Graph type representing Go.
- typedef boost::graph_traits< Graph >::vertex_descriptor GoVertex

  A vertex object.
- typedef boost::graph_traits< Graph >::edge_descriptor GoEdge

  An edge object.
- typedef boost::graph_traits< Graph >::vertex_iterator GoVertexIterator

  A vertex iterator.
- typedef boost::graph_traits< Graph >::in_edge_iterator InEdgeIterator

  An in edge iterator.
- typedef boost::graph_traits< Graph >::out_edge_iterator OutEdgeIterator

  An out edge iterator.
- typedef boost::property_map< Graph, boost::vertex_index_t >::type VertexIndexMap

  A vertex to index map.
- typedef boost::property_map< Graph, boost::edge_index_t >::type EdgeIndexMap

  An edge to index map.

Public Member Functions

- ~GoGraph ()

  A destructor.
- void insertTerm (const std::string &termId, const std::string &name, const std::string &description, const std::string &ontology)

  Method to insert terms into the graph.
- void insertRelationship (const std::string &termParent, const std::string &termChild, const std::string &relationship)

  Method to insert relationship edges into the graph.
- std::size_t getNumVertices ()

  Helper method to get number of vertices.
- std::size_t getNumEdges ()

Helper method to get number of edges.

- void initMaps ()

  A method to initialize internal index maps.

- Graph * getGraph ()

  A method to return the boost graph.

- bool hasTerm (const std::string &term)

  A helper method to test term existence.

- size_t getTermIndex (const std::string &term)

  A helper method to return the index of the term.

- std::string getTermStringIdByIndex (std::size_t index)

  A helper method to return the string id based on the index.

- std::string getTermNameByIndex (std::size_t index)

  A helper method to return the string name based on the index.

- std::string getTermName (std::string term)

  A helper method to return the string name based on the go term.

- std::string getTermDescriptionByIndex (std::size_t index)

  A helper method to return the string description based on the index.

- std::string getTermDescription (std::string term)

  A helper method to return the string description based on the go term.

- GoVertex getRoot ()

  A helper method to return the root of the graph.

- GO::Onto getTermOntology (const std::string &term)

  A helper method to return the ontology of a term by term string.

- GO::Onto getTermOntologyByIndex (std::size_t index)

  A helper method to return the ontology of a term by index.

- GO::Onto getTermOntologyByVertex (GoVertex vertex)

  A helper method to return the ontology of a term by GoVertex.

- std::size_t getVertexIndex (GoVertex vertex)

  A helper method to return the index of a GoVertex.

- GoVertex getVertexByIndex (std::size_t index)

  A helper method to return the GoVertex for the given index.

- GoVertex getVertexByName (const std::string &term)

  A helper method to return the GoVertex for the given term.

- boost::unordered_set< std::string > getDescendantTerms (const std::string &term)

  A helper method to get the desendant terms for a given term.

- boost::unordered_set< std::string > getAncestorTerms (const std::string &term)

  A helper method to get the ancestor terms for a given term.

- boost::unordered_set< std::string > getParentTerms (const std::string &term)

  A helper method to get the parent terms for a given term.

- boost::unordered_set< std::string > getChildTerms (const std::string &term)

  A helper method to get the child terms for a given term.

- boost::unordered_set< std::string > getAllTerms ()

  A helper method to retrieve all terms in the GoGraph.

- boost::unordered_set< std::string > getAllTermsBP ()

  A helper method to retrieve all terms in the GoGraph belonging to the BP ontology.

- boost::unordered_set< std::string > getAllTermsMF ()

  A helper method to retrieve all terms in the GoGraph belonging to the MF ontology.

- boost::unordered_set< std::string > getAllTermsCC ()

  A helper method to retrieve all terms in the GoGraph belonging to the CC ontology.

- boost::unordered_set< std::string > filterSetForOntology (const boost::unordered_set< std::string > &inSet, GO::Onto onto)

A helper method to filter out all terms not belonging to a particular ontology.

- boost::unordered_set< std::string > filterSetForOntology (const std::vector< std::string > &inSet, GO::Onto onto)

  A helper method to filter out all terms not belonging to a particular ontology from a vector.
- boost::unordered_set< std::string > filterSetForBP (const std::vector< std::string > &inSet)

  A helper method to retrun only BP terms from a vector.
- boost::unordered_set< std::string > filterSetForBP (const boost::unordered_set< std::string > &inSet)

  A helper method to retrun only BP terms from a set.
- boost::unordered_set< std::string > filterSetForMF (const std::vector< std::string > &inSet)

  A helper method to retrun only MF terms from a vector.
- boost::unordered_set< std::string > filterSetForMF (const boost::unordered_set< std::string > &inSet)

  A helper method to retrun only MF terms from a set.
- boost::unordered_set< std::string > filterSetForCC (const std::vector< std::string > &inSet)

  A helper method to retrun only CC terms from a vector.
- boost::unordered_set< std::string > filterSetForCC (const boost::unordered_set< std::string > &inSet)

  A helper method to retrun only CC terms from a set.
- boost::unordered_set< std::string > getOntologyTerms (GO::Onto ontology)

  A helper method to return only the terms of the give ontology.
- Graph * getInducedSubgraph2 (const std::string &termId)

  A method to return the induced subgraph of a given term, ancestor graph.
- Graph * getInducedSubgraph (const std::string &termId)

  A method to return the induced subgraph of a given term, ancestor graph.
- std::size_t getNumComponents ()

  A method to calculate the number of connected components of the graph.

5.26.1 Detailed Description

This class holds the Gene Ontology directed acyclic graph.

This class holds the Gene Ontology as a boost graph. It provides the graph data, as well as other strucutres which make working with the graph easier.

5.26.2 Member Typedef Documentation

5.26.2.1 typedef boost::property_map<Graph, boost::edge_index_t >::type GoGraph::EdgeIndexMap

An edge to index map.

A typedef of the boost property_map. Saves typing by using EdgeIndexMap.

5.26.2.2 typedef boost::graph_traits<Graph>::edge_descriptor GoGraph::GoEdge

An edge object.

A typedef of the boost edge_descriptor. Saves typing by using GoEdge.

5.26.2.3 typedef boost::graph_traits<Graph>::vertex_descriptor GoGraph::GoVertex

A vertex object.

A typedef of the boost vertex_descriptor. Saves typing by using GoVertex.

5.26.2.4 typedef boost::graph_traits<Graph>::vertex_iterator GoGraph::GoVertexIterator

A vertex iterator.

A typedef of the boost vertex_iterator. Saves typing by using GoVertexIterator.

5.26.2.5 typedef boost::subgraph<Graph_t> GoGraph::Graph

The main Graph type representing Go.

This typedef defines the main type as a subgraph of Graph_t. This allows the the graph to be divided into subgraphs if needed. Virtually not differnece but can cause problems with some boost constructors such as random graph generators.

5.26.2.6 typedef boost::adjacency_list<boost::vecS, boost::vecS, boost::bidirectionalS, boost::property< boost::vertex_index_t, size_t, VertexProps>, boost::property< boost::edge_index_t, size_t, EdgeProps> > GoGraph::Graph_t

A Graph type representing Go.

This typedef defines a graph type used as the basic go graph. This typedef takes VertexProps and EdgeProps as templete arguments.

5.26.2.7 typedef boost::graph_traits<Graph>::in_edge_iterator GoGraph::InEdgeIterator

An in edge iterator.

A typedef of the boost in_edge_iterator. Saves typing by using InEdgeIterator.

5.26.2.8 typedef boost::graph_traits<Graph>::out_edge_iterator GoGraph::OutEdgeIterator

An out edge iterator.

A typedef of the boost out_edge_iterator. Saves typing by using OutEdgeIterator.

5.26.2.9 typedef boost::property_map<Graph, boost::vertex_index_t >::type GoGraph::VertexIndexMap

A vertex to index map.

A typedef of the boost property_map. Saves typing by using VertexIndexMap.

5.26.3 Constructor & Destructor Documentation

5.26.3.1 GoGraph::~GoGraph ( ) [inline]

A destructor.

Destroying the graph calls clear on all the containers. Other data should be destroyed when leaving scope.

5.26.4 Member Function Documentation

5.26.4.1 boost::unordered_set<std::string> GoGraph::filterSetForBP (const std::vector< std::string > & *inSet*) [inline]

A helper method to retrun only BP terms from a vector.

This method returns a filtered set containing only BP terms

5.26.4.2 boost::unordered_set<std::string> GoGraph::filterSetForBP (const boost::unordered_set< std::string > & *inSet*) [inline]

A helper method to retrun only BP terms from a set.

This method returns a filtered set containing only BP terms

5.26.4.3 boost::unordered_set<std::string> GoGraph::filterSetForCC (const std::vector< std::string > & *inSet*) [inline]

A helper method to retrun only CC terms from a vector.

This method returns a filtered set containing only CC terms

5.26.4.4 boost::unordered_set<std::string> GoGraph::filterSetForCC (const boost::unordered_set< std::string > & *inSet*) [inline]

A helper method to retrun only CC terms from a set.

This method returns a filtered set containing only CC terms

5.26.4.5 boost::unordered_set<std::string> GoGraph::filterSetForMF (const std::vector< std::string > & *inSet*) [inline]

A helper method to retrun only MF terms from a vector.

This method returns a filtered set containing only MF terms

5.26.4.6 boost::unordered_set<std::string> GoGraph::filterSetForMF (const boost::unordered_set< std::string > & *inSet*) [inline]

A helper method to retrun only MF terms from a set.

This method returns a filtered set containing only MF terms

5.26.4.7 boost::unordered_set<std::string> GoGraph::filterSetForOntology (const boost::unordered_set< std::string > & *inSet,* GO::Onto *onto*) [inline]

A helper method to filter out all terms not belonging to a particular ontology.

This method returns a filtered set of ontology terms matching the given ontology

5.26.4.8 boost::unordered_set<std::string> GoGraph::filterSetForOntology (const std::vector< std::string > & *inSet,* GO::Onto *onto*) [inline]

A helper method to filter out all terms not belonging to a particular ontology from a vector.

This method returns a filtered set of ontology terms matching the given ontology

5.26.4.9 boost::unordered_set<std::string> GoGraph::getAllTerms ( ) [inline]

A helper method to retrieve all terms in the GoGraph.

This method returns a set of term strings

5.26.4.10 boost::unordered_set<std::string> GoGraph::getAllTermsBP ( ) [inline]

A helper method to retrieve all terms in the GoGraph belonging to the BP ontology.

This method returns a set of BP terms in the graph

5.26.4.11 boost::unordered_set<std::string> GoGraph::getAllTermsCC ( ) [inline]

A helper method to retrieve all terms in the GoGraph belonging to the CC ontology.

This method returns a set of CC terms in the graph

5.26.4.12 boost::unordered_set<std::string> GoGraph::getAllTermsMF ( ) [inline]

A helper method to retrieve all terms in the GoGraph belonging to the MF ontology.

This method returns a set of MF terms in the graph

5.26.4.13 boost::unordered_set<std::string> GoGraph::getAncestorTerms (const std::string & *term*) [inline]

A helper method to get the ancestor terms for a given term.

This method takes a term and returns a list of ancestor terms.

5.26.4.14 boost::unordered_set<std::string> GoGraph::getChildTerms (const std::string & *term*) [inline]

A helper method to get the child terms for a given term.

This method takes a term and returns a list of child terms.

5.26.4.15 boost::unordered_set<std::string> GoGraph::getDescendantTerms (const std::string & *term*) [inline]

A helper method to get the desendant terms for a given term.

This method takes a term and returns a list of desendant terms.

5.26.4.16 Graph* GoGraph::getGraph ( ) [inline]

A method to return the boost graph.

This method is needed to return the graph to other boost algorithms. As stated by boost, subclasses of adjacency_↩ list are not recommended. http://www.boost.org/doc/libs/1_54_0/libs/graph/doc/graph↩ _concepts.html

5.26.4.17 Graph* GoGraph::getInducedSubgraph (const std::string & *termId*) [inline]

A method to return the induced subgraph of a given term, ancestor graph.

This method returns a subgraph of the graph induced by traversing the ancestors of the given vertex.

5.26.4.18 Graph* GoGraph::getInducedSubgraph2 (const std::string & *termId*) [inline]

A method to return the induced subgraph of a given term, ancestor graph.

This method returns a subgraph of the graph induced by traversing the ancestors of the given vertex.

5.26.4.19 std::size_t GoGraph::getNumComponents ( ) [inline]

A method to calculate the number of connected components of the graph.

This method calculates the number of connected components in the graph. This is used to check if the GO graph conatains only the 3 sub-ontologies.

5.26.4.20 std::size_t GoGraph::getNumEdges ( ) [inline]

Helper method to get number of edges.

This method calls boost num_edges on the go graph

5.26.4.21 std::size_t GoGraph::getNumVertices ( ) [inline]

Helper method to get number of vertices.

This method calls boost num_vertices on the go graph

5.26.4.22 boost::unordered_set<std::string> GoGraph::getOntologyTerms (GO::Onto *ontology*) [inline]

A helper method to return only the terms of the give ontology.

Returns only those terms used that occur for the given ontology.

5.26.4.23 boost::unordered_set<std::string> GoGraph::getParentTerms (const std::string & *term*) [inline]

A helper method to get the parent terms for a given term.

This method takes a term and returns a list of parent terms (immediate ancestors).

5.26.4.24 GoVertex GoGraph::getRoot ( ) [inline]

A helper method to return the root of the graph.

This method returns the root vertex or first root vertex of a graph.

5.26.4.25 std::string GoGraph::getTermDescription (std::string *term*) [inline]

A helper method to return the string description based on the go term.

This method returns the term's description string using the go term.

5.26.4.26 std::string GoGraph::getTermDescriptionByIndex (std::size_t *index*) [inline]

A helper method to return the string description based on the index.

This method returns the term's description string using its index. Used mainly for testing.

5.26.4.27 size_t GoGraph::getTermIndex (const std::string & *term*) [inline]

A helper method to return the index of the term.

This method returns the index of the given term.

5.26.4.28 std::string GoGraph::getTermName (std::string *term*) [inline]

A helper method to return the string name based on the go term.

This method returns the term's string name using the go term.

5.26.4.29 std::string GoGraph::getTermNameByIndex (std::size_t *index*) [inline]

A helper method to return the string name based on the index.

This method returns the term's string name using its index. Used mainly for testing.

5.26.4.30 GO::Onto GoGraph::getTermOntology (const std::string & *term*) [inline]

A helper method to return the ontology of a term by term string.

This method returns the term's ontoogy taking a string term as an argument

5.26.4.31 GO::Onto GoGraph::getTermOntologyByIndex (std::size_t *index*) [inline]

A helper method to return the ontology of a term by index.

This method returns the term's ontoogy taking an index as an argument

5.26.4.32 GO::Onto GoGraph::getTermOntologyByVertex (GoVertex *vertex*) [inline]

A helper method to return the ontology of a term by GoVertex.

This method returns the term's ontoogy taking GoVertex as an argument

5.26.4.33 std::string GoGraph::getTermStringIdByIndex (std::size_t *index*) [inline]

A helper method to return the string id based on the index.

This method returns the term's string id using its index. Used mainly for testing.

5.26.4.34 GoVertex GoGraph::getVertexByIndex (std::size_t *index*) [inline]

A helper method to return the GoVertex for the given index.

This method returns the GoVertex based on the given index

5.26.4.35 GoVertex GoGraph::getVertexByName (const std::string & *term*) [inline]

A helper method to return the GoVertex for the given term.

This method returns the GoVertex based on the given term

5.26.4.36 std::size_t GoGraph::getVertexIndex (GoVertex *vertex*) [inline]

A helper method to return the index of a GoVertex.

This method returns the index of a GoVertex

5.26.4.37 bool GoGraph::hasTerm (const std::string & *term*) [inline]

A helper method to test term existence.

Tests the map for existence of the term.

5.26.4.38 void GoGraph::initMaps ( ) [inline]

A method to initialize internal index maps.

This method sets the private map variables by call calling boost get on the property maps.

5.26.4.39 void GoGraph::insertRelationship (const std::string & *termParent,* const std::string & *termChild,* const std::string & *relationship*) [inline]

Method to insert relationship edges into the graph.

This method takes a parent term, child term, and relationshp type as arguments. The method will insert the edge into the graph, setting the relationship type based on the data provided.

5.26.4.40 void GoGraph::insertTerm (const std::string & *termId,* const std::string & *name,* const std::string & *description,* const std::string & *ontology*) [inline]

Method to insert terms into the graph.

This method takes a go term, description, and ontology information (MF,BP,CC). The method will check if the term already exists in the graph then add the vertex or update the meta data accordingly. The parser can call this method without having to consider if terms have already been added or not.

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

- ggtk/GoGraph.hpp

5.27 GoParserFactory Class Reference

A class to return an instance of GoParserInterface at runtime based on an argument.

`#include <ggtk/GoParserFactory.hpp>`

Public Member Functions

- GoParserFactory ()

  Class constructor.
- ∼GoParserFactory ()

  Class destructor.
- void addParser (std::string name, GoParserInterface ∗parser)

  A Method to add a parser to the factory.
- GoParserInterface ∗ getParser (std::string name)

  A method to return a parser based on a query string.

5.27.1 Detailed Description

A class to return an instance of GoParserInterface at runtime based on an argument.

This class holds a set of parser classes. When queried using the getParser method, it returns an instance of GoParserInterface based on a string key. This allows parsers to be easily added to a larger system and switched at runtime.

5.27.2 Constructor & Destructor Documentation

5.27.2.1 GoParserFactory::GoParserFactory ( ) [inline]

Class constructor.

This constructor initializes the private lists to empty vectors.

5.27.2.2 GoParserFactory::∼GoParserFactory ( ) [inline]

Class destructor.

This destructor clears the names vector. It also deletes each parser pointer explicitly. Finally it clears the parser list.

5.27.3 Member Function Documentation

5.27.3.1 void GoParserFactory::addParser (std::string *name,* GoParserInterface ∗ *parser*) [inline]

A Method to add a parser to the factory.

This method adds a pointer to a parser and a string to the factory. This string is used to query the appropriate parser.

5.27.3.2 GoParserInterface* GoParserFactory::getParser (std::string *name*) [inline]

A method to return a parser based on a query string.

If the string supplied matches one of the keys in the database, the appropriate parser will be returned. If not, a NULL pointer is returned. The calling environment must check against NULL before using.

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

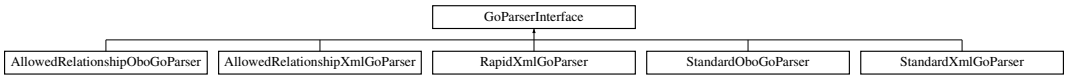
- ggtk/GoParserFactory.hpp

5.28 GoParserInterface Class Reference

An interface class to define go graph parsers.

`#include <ggtk/GoParserInterface.hpp>`

Inheritance diagram for GoParserInterface:



Public Member Functions

- virtual GoGraph * parseGoFile (std::string fileName)=0

  A pure virtual method for parsing the file and returning a GoGraph object.
- virtual bool isFileGood (const std::string &filename)=0

  A pure virtual method for parsing the file and returning a GoGraph object.
- virtual GoParserInterface * clone ()=0

  A pure virtual clone function for factory pattern.

5.28.1 Detailed Description

An interface class to define go graph parsers.

This class defines the interface of a go graph parser. Pure virtual methods require that parsers implement these methods.

5.28.2 Member Function Documentation

5.28.2.1 virtual GoParserInterface* GoParserInterface::clone ( ) [pure virtual]

A pure virtual clone function for factory pattern.

This pure virtual method returns an instance of this interface. This method is used in a factory class to have the ability to decide the parser at runtime.

Implemented in AllowedRelationshipOboGoParser, AllowedRelationshipXmlGoParser, RapidXmlGoParser, StandardXmlGoParser, and StandardOboGoParser.

5.28.2.2 virtual bool GoParserInterface::isFileGood (const std::string & *filename*) [pure virtual]

A pure virtual method for parsing the file and returning a GoGraph object.

This pure virtual method requires any parser to have a method that takes a filename string and returns a GoGraph object pointer.

Implemented in AllowedRelationshipOboGoParser, AllowedRelationshipXmlGoParser, RapidXmlGoParser, StandardXmlGoParser, and StandardOboGoParser.

5.28.2.3 virtual GoGraph* GoParserInterface::parseGoFile (std::string *fileName*) [pure virtual]

A pure virtual method for parsing the file and returning a GoGraph object.

This pure virtual method requires any parser to have a method that takes a filename string and returns a GoGraph object pointer.

Implemented in AllowedRelationshipOboGoParser, AllowedRelationshipXmlGoParser, StandardXmlGoParser, RapidXmlGoParser, and StandardOboGoParser.

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

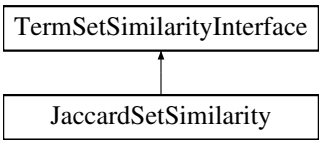
- ggtk/GoParserInterface.hpp

5.29 JaccardSetSimilarity Class Reference

A class to calculate jaccard similarity between 2 sets.

`#include <ggtk/JaccardSetSimilarity.hpp>`

Inheritance diagram for JaccardSetSimilarity:



Public Member Functions

- JaccardSetSimilarity ()
  Constructor.
- double calculateSimilarity (const boost::unordered_set< std::string > &termsA, const boost::unordered_set< std::string > &termsB)
  A method for calculating term set to term set similarity for GO terms;.

5.29.1 Detailed Description

A class to calculate jaccard similarity between 2 sets.

This class calculates jaccard set similarity between two sets of terms.

5.29.2 Constructor & Destructor Documentation

5.29.2.1 JaccardSetSimilarity::JaccardSetSimilarity ( ) [inline]

Constructor.

Creates the JaccardSetSimilarity class.

5.29.3 Member Function Documentation

5.29.3.1 double JaccardSetSimilarity::calculateSimilarity (const boost::unordered_set< std::string > & *termsA*, const boost::unordered_set< std::string > & *termsB*) [inline], [virtual]

A method for calculating term set to term set similarity for GO terms;.

This method returns the Jaccard set similarity.

Implements TermSetSimilarityInterface.

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

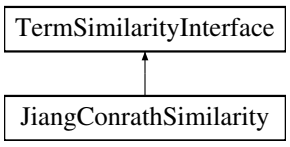
- ggtk/JaccardSetSimilarity.hpp

5.30 JiangConrathSimilarity Class Reference

A class to calculate Jiang Conrath similarity between 2 terms.

`#include <ggtk/JiangConrathSimilarity.hpp>`

Inheritance diagram for JiangConrathSimilarity:



Public Member Functions

- JiangConrathSimilarity (GoGraph *goGraph, TermInformationContentMap &icMap)
  A constructor.
- double calculateTermSimilarity (std::string goTermA, std::string goTermB)
  A method for calculating term-to-term similarity for GO terms using JiangConrath similarity.
- double calculateNormalizedTermSimilarity (std::string goTermA, std::string goTermB)
  A method for calculating term-to-term similarity for GO terms using Normalized JiangConrath similarity.
- std::string getMICA (boost::unordered_set< std::string > &ancestorsA, boost::unordered_set< std::string > &ancestorsB)
  A method for calculating the most informative common ancestor.

5.30.1 Detailed Description

A class to calculate Jiang Conrath similarity between 2 terms.

This class calculates Jiang Conrath similarity.

Jiang, J. J., & Conrath, D. W. (1997). Semantic similarity based on corpus statistics and lexical taxonomy. In Proc. of 10th International Conference on Research on Computational Linguistics, Taiwan.

P. W. Lord, R. D. Stevens, A. Brass, and C. A. Goble, "Semantic similarity measures as tools for exploring the gene ontology," Pac Symp Biocomput, pp. 601-12, 2003.

distance = IC(termA) + IC(termB) - 2∗IC(MICA) maxDistance = 2∗IC(single annotaiotn) similarity = 1 - distance/maxDistance (see Lord et al.)

5.30.2 Constructor & Destructor Documentation

5.30.2.1 JiangConrathSimilarity::JiangConrathSimilarity (GoGraph ∗ *goGraph,* TermInformationContentMap & *icMap*) [inline]

A constructor.

Creates the default(empty) StandardRelationshipPolicy

5.30.3 Member Function Documentation

5.30.3.1 double JiangConrathSimilarity::calculateNormalizedTermSimilarity (std::string *goTermA,* std::string *goTermB*) [inline], [virtual]

A method for calculating term-to-term similarity for GO terms using Normalized JiangConrath similarity.

This method returns the JiangConrath similarity scaled between 0 and 1 [0,1] inclusive

Implements TermSimilarityInterface.

5.30.3.2 double JiangConrathSimilarity::calculateTermSimilarity (std::string *goTermA,* std::string *goTermB*) [inline], [virtual]

A method for calculating term-to-term similarity for GO terms using JiangConrath similarity.

This method returns the JiangConrath similarity or the information content of the most informative common ancestor.

Implements TermSimilarityInterface.

5.30.3.3 std::string JiangConrathSimilarity::getMICA (boost::unordered_set< std::string > & *ancestorsA,* boost::unordered_set< std::string > & *ancestorsB*) [inline]

A method for calculating the most informative common ancestor.

This method searches the sets to determine the most informatics ancestor.

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

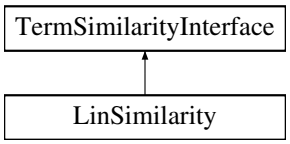
- ggtk/JiangConrathSimilarity.hpp

5.31 LinSimilarity Class Reference

A class to calculate Lin similarity between 2 terms.

`#include <ggtk/LinSimilarity.hpp>`

Inheritance diagram for LinSimilarity:



Public Member Functions

- LinSimilarity (GoGraph *goGraph, TermInformationContentMap &icMap)

  A constructor.
- double calculateTermSimilarity (std::string goTermA, std::string goTermB)

  A method for calculating term-to-term similarity for GO terms using Lin similarity.
- double calculateNormalizedTermSimilarity (std::string goTermA, std::string goTermB)

  A method for calculating term-to-term similarity for GO terms using Normalized Lin similarity.
- std::string getMICA (boost::unordered_set< std::string > &ancestorsA, boost::unordered_set< std::string > &ancestorsB)

  A method for calculating the most informative common ancestor.

5.31.1 Detailed Description

A class to calculate Lin similarity between 2 terms.

This class calculates Lin similarity.

Lin, D. (1998) An information theoretic definition of similarity. In: Proc. of the 15th Inernational Conference on Machine Learning. San Franscisco, CA: Morgan Kaufman. pp 296-304

P. W. Lord, R. D. Stevens, A. Brass, and C. A. Goble, "Semantic similarity measures as tools for exploring the gene ontology," Pac Symp Biocomput, pp. 601-12, 2003.

2 * IC(MICA) / (IC(termA) + IC(termB))

5.31.2 Constructor & Destructor Documentation

5.31.2.1 LinSimilarity::LinSimilarity (GoGraph * *goGraph,* TermInformationContentMap & *icMap*) [inline]

A constructor.

Creates the LinSimilarity calculator

5.31.3 Member Function Documentation

5.31.3.1 double LinSimilarity::calculateNormalizedTermSimilarity (std::string *goTermA*, std::string *goTermB*) [inline], [virtual]

A method for calculating term-to-term similarity for GO terms using Normalized Lin similarity.

This method returns the Lin similarity scaled between 0 and 1 [0,1] inclusive

Implements TermSimilarityInterface.

5.31.3.2 double LinSimilarity::calculateTermSimilarity (std::string *goTermA*, std::string *goTermB*) [inline], [virtual]

A method for calculating term-to-term similarity for GO terms using Lin similarity.

This method returns the Lin similarity.

Implements TermSimilarityInterface.

5.31.3.3 std::string LinSimilarity::getMICA (boost::unordered_set< std::string > & *ancestorsA*, boost::unordered_set< std::string > & *ancestorsB*) [inline]

A method for calculating the most informative common ancestor.

This method searches the sets to determine the most informatics ancestor.

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

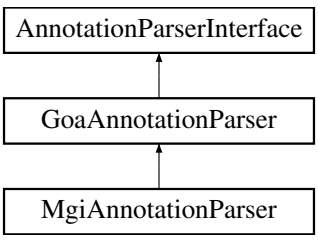
- ggtk/LinSimilarity.hpp

5.32 MgiAnnotationParser Class Reference

A class to parse an Mouse Genome Informatics go annotation file.

```
#include <ggtk/MgiAnnotationParser.hpp>
```

Inheritance diagram for MgiAnnotationParser:



Public Member Functions

- MgiAnnotationParser ()

  A default constructor method for creating the parser.
- MgiAnnotationParser (EvidencePolicyInterface *policy)

  A parameterized constructor method for creating the parser with a policy.

5.32.1 Detailed Description

A class to parse an Mouse Genome Informatics go annotation file.

This class will read an mgi annotation file and add those annoations to an AnnotationData class. Available at: ftp://ftp.informatics.jax.org/pub/reports/index.html#go

MGI uses GAF format which is GOA.

Implements AnnotationParserInterface

5.32.2 Constructor & Destructor Documentation

5.32.2.1 MgiAnnotationParser::MgiAnnotationParser ( ) [inline]

A default constructor method for creating the parser.

Creates the parser

5.32.2.2 MgiAnnotationParser::MgiAnnotationParser (EvidencePolicyInterface * *policy*) [inline]

A parameterized constructor method for creating the parser with a policy.

Creates the parser with a custom policy

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

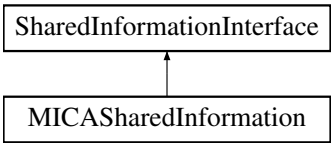
- ggtk/MgiAnnotationParser.hpp

5.33 MICASharedInformation Class Reference

A class to calculate shared infromation as the most informative common ancestor (MICA)

`#include <ggtk/MICASharedInformation.hpp>`

Inheritance diagram for MICASharedInformation:



Public Member Functions

- MICASharedInformation (GoGraph *goGraph, TermInformationContentMap &icMap)

  A constructor.
- double sharedInformation (const std::string &termA, const std::string &termB)

  A method for calculating the shared infromation between two concepts.
- double sharedInformation (const std::string &term)

  An interface method for returning the shared information of a single terms,or information content.
- double maxInformationContent (const std::string &term)

  An interface method for returning the maximum information content for a term.
- bool hasTerm (const std::string &term)

  An interface method for determining if a term can be found.
- bool isSameOntology (const std::string &termA, const std::string &termB)

  An interface method for determining if the two terms are of like ontologies.

5.33.1 Detailed Description

A class to calculate shared infromation as the most informative common ancestor (MICA)

This class calculates shared infromation using the most informative common ancestor (MICA). The MICA is a term that is also known as the minimum subsumer.

This shared information method forms the basis of 3 inforamtion content measures put forward by Lord el al.

P. W. Lord, R. D. Stevens, A. Brass, and C. A. Goble, "Semantic similarity measures as tools for exploring the gene ontology," Pac Symp Biocomput, pp. 601-12, 2003.

5.33.2 Constructor & Destructor Documentation

5.33.2.1 MICASharedInformation::MICASharedInformation (GoGraph * *goGraph,* TermInformationContentMap & *icMap*) [inline]

A constructor.

Creates the MICASharedInformation class

5.33.3 Member Function Documentation

5.33.3.1 bool MICASharedInformation::hasTerm (const std::string & *term*) [inline], [virtual]

An interface method for determining if a term can be found.

Determines if the term can be found in the current map.

Implements SharedInformationInterface.

5.33.3.2 bool MICASharedInformation::isSameOntology (const std::string & *termA*, const std::string & *termB*) [inline], [virtual]

An interface method for determining if the two terms are of like ontologies.

Determine if two terms are of the same ontology.

Implements SharedInformationInterface.

5.33.3.3 double MICASharedInformation::maxInformationContent (const std::string & *term*) [inline], [virtual]

An interface method for returning the maximum information content for a term.

This method provides the absolute max information content within a corpus for normalization purposes.

Implements SharedInformationInterface.

5.33.3.4 double MICASharedInformation::sharedInformation (const std::string & *termA*, const std::string & *termB*) [inline], [virtual]

A method for calculating the shared infromation between two concepts.

This method returns the shared information between two concepts.

Implements SharedInformationInterface.

5.33.3.5 double MICASharedInformation::sharedInformation (const std::string & *term*) [inline], [virtual]

An interface method for returning the shared information of a single terms,or information content.

This method privdes a mechanism for returing a term's infromation content.

Implements SharedInformationInterface.

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

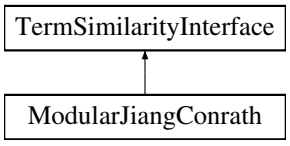
- ggtk/MICASharedInformation.hpp

5.34 ModularJiangConrath Class Reference

A class to calculate Jiang Conrath similarity between 2 terms.

`#include <ggtk/ModularJiangConrath.hpp>`

Inheritance diagram for ModularJiangConrath:



Public Member Functions

- ModularJiangConrath (SharedInformationInterface *sharedInformationCalculator)

  A constructor.
- double calculateTermSimilarity (std::string goTermA, std::string goTermB)

  A method for calculating term-to-term similarity for GO terms using Lin similarity.
- double calculateNormalizedTermSimilarity (std::string goTermA, std::string goTermB)

  A method for calculating term-to-term similarity for GO terms using normalized Lin similarity.
- void setSharedInformationCalculator (SharedInformationInterface *newSharedInformationCalulator)

  A method to set alternative methods of shared information calculators.

5.34.1 Detailed Description

A class to calculate Jiang Conrath similarity between 2 terms.

This class calculates Jiang Conrath similarity.

Jiang, J. J., & Conrath, D. W. (1997). Semantic similarity based on corpus statistics and lexical taxonomy. In Proc. of 10th International Conference on Research on Computational Linguistics, Taiwan.

P. W. Lord, R. D. Stevens, A. Brass, and C. A. Goble, "Semantic similarity measures as tools for exploring the gene ontology," Pac Symp Biocomput, pp. 601-12, 2003.

distance = IC(termA) + IC(termB) - 2*IC(MICA) maxDistance = 2*IC(single annotaiotn) similarity = 1 - distance/maxDistance (see Lord et al.)

5.34.2 Constructor & Destructor Documentation

5.34.2.1 ModularJiangConrath::ModularJiangConrath (SharedInformationInterface * *sharedInformationCalculator*) [inline]

A constructor.

Creates the Jiang Conrath simialrity measure using a given shared infromation calculator

5.34.3 Member Function Documentation

5.34.3.1 double ModularJiangConrath::calculateNormalizedTermSimilarity (std::string *goTermA*, std::string *goTermB*) [inline], [virtual]

A method for calculating term-to-term similarity for GO terms using normalized Lin similarity.

This method returns the Lin similarity. Lin similarity is already normalized

Implements TermSimilarityInterface.

5.34.3.2 double ModularJiangConrath::calculateTermSimilarity (std::string *goTermA*, std::string *goTermB*) [inline], [virtual]

A method for calculating term-to-term similarity for GO terms using Lin similarity.

This method returns the Resnik similarity or the information content of the most informative common ancestor.

Implements TermSimilarityInterface.

5.34.3.3 void ModularJiangConrath::setSharedInformationCalculator (SharedInformationInterface * *newSharedInformationCalulator*) [inline]

A method to set alternative methods of shared information calculators.

This method accepts a new method for calculating the shared information of two terms.

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

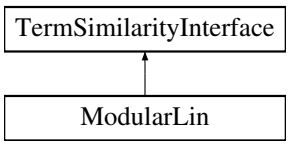
- ggtk/ModularJiangConrath.hpp

5.35 ModularLin Class Reference

A class to calculate Lin similarity between 2 terms.

`#include <ggtk/ModularLin.hpp>`

Inheritance diagram for ModularLin:



Public Member Functions

- ModularLin (SharedInformationInterface *sharedInformationCalculator)
  A constructor.
- double calculateTermSimilarity (std::string goTermA, std::string goTermB)
  A method for calculating term-to-term similarity for GO terms using Lin similarity.
- double calculateNormalizedTermSimilarity (std::string goTermA, std::string goTermB)
  A method for calculating term-to-term similarity for GO terms using normalized Lin similarity.
- void setSharedInformationCalculator (SharedInformationInterface *newSharedInformationCalulator)
  A method to set alternative methods of shared information calculators.

5.35.1 Detailed Description

A class to calculate Lin similarity between 2 terms.

This class calculates Lin similarity.

Lin, D. (1998) An information theoretic definition of similarity. In: Proc. of the 15th Inernational Conference on Machine Learning. San Franscisco, CA: Morgan Kaufman. pp 296-304

P. W. Lord, R. D. Stevens, A. Brass, and C. A. Goble, "Semantic similarity measures as tools for exploring the gene ontology," Pac Symp Biocomput, pp. 601-12, 2003.

2 ∗ IC(MICA) / (IC(termA) + IC(termB))

5.35.2 Constructor & Destructor Documentation

5.35.2.1 ModularLin::ModularLin (SharedInformationInterface ∗ *sharedInformationCalculator*) [inline]

A constructor.

Creates the LinSimilarity calculator with a particular shared information calculator

5.35.3 Member Function Documentation

5.35.3.1 double ModularLin::calculateNormalizedTermSimilarity (std::string *goTermA,* std::string *goTermB*) [inline], [virtual]

A method for calculating term-to-term similarity for GO terms using normalized Lin similarity.

This method returns the Lin similarity. Lin similarity is already normalized

Implements TermSimilarityInterface.

5.35.3.2 double ModularLin::calculateTermSimilarity (std::string *goTermA,* std::string *goTermB*) [inline], [virtual]

A method for calculating term-to-term similarity for GO terms using Lin similarity.

This method returns the Resnik similarity or the information content of the most informative common ancestor.

Implements TermSimilarityInterface.

5.35.3.3 void ModularLin::setSharedInformationCalculator (SharedInformationInterface ∗ *newSharedInformationCalulator*) [inline]

A method to set alternative methods of shared information calculators.

This method accepts a new method for calculating the shared information of two terms.

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

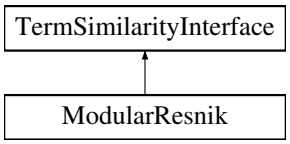
- ggtk/ModularLin.hpp

5.36 ModularResnik Class Reference

A class to calculate resnik similarity between 2 terms using a shared information interface.

`#include <ggtk/ModularResnik.hpp>`

Inheritance diagram for ModularResnik:



Public Member Functions

- ModularResnik (SharedInformationInterface ∗sharedInformationCalculator)

  A constructor.
- double calculateTermSimilarity (std::string goTermA, std::string goTermB)

  A method for calculating term-to-term similarity for GO terms using Resnik similarity.
- double calculateNormalizedTermSimilarity (std::string goTermA, std::string goTermB)

  A method for calculating term-to-term similarity for GO terms using Normalized Resnik similarity.
- void setSharedInformationCalculator (SharedInformationInterface ∗newSharedInformationCalulator)

  A constructor.

5.36.1 Detailed Description

A class to calculate resnik similarity between 2 terms using a shared information interface.

This class calculates Resnik similarity. Philip Resnik (1995). "Using information content to evaluate semantic similarity in a taxonomy". In Chris S. Mellish (Ed.). Proceedings of the 14th international joint conference on Artificial intelligence (IJCAI'95)

P. W. Lord, R. D. Stevens, A. Brass, and C. A. Goble, "Semantic similarity measures as tools for exploring the gene ontology," Pac Symp Biocomput, pp. 601-12, 2003.

maximun information content of all shared ancestors IC(MICA)

5.36.2 Constructor & Destructor Documentation

5.36.2.1 ModularResnik::ModularResnik (SharedInformationInterface ∗ *sharedInformationCalculator*) [inline]

A constructor.

Creates the default(empty) StandardRelationshipPolicy

5.36.3 Member Function Documentation

5.36.3.1 double ModularResnik::calculateNormalizedTermSimilarity (std::string *goTermA*, std::string *goTermB*) [inline], [virtual]

A method for calculating term-to-term similarity for GO terms using Normalized Resnik similarity.

This method returns the Resnik similarity divided by the maximum possible similarity

Implements TermSimilarityInterface.

5.36.3.2 double ModularResnik::calculateTermSimilarity (std::string *goTermA*, std::string *goTermB*) [inline], [virtual]

A method for calculating term-to-term similarity for GO terms using Resnik similarity.

This method returns the Resnik similarity or the information content of the most informative common ancestor.

Implements TermSimilarityInterface.

5.36.3.3 void ModularResnik::setSharedInformationCalculator (SharedInformationInterface * *newSharedInformationCalulator*) [inline]

A constructor.

Creates the default(empty) StandardRelationshipPolicy

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

- ggtk/ModularResnik.hpp

5.37 NCList Class Reference

A container class for quickly finding intersections of genomic intervals.

```
#include <ggtk/NCList.hpp>
```

Classes

- class NCNode

Public Member Functions

- NCList (std::vector< GenomicRegion > &regions)
- size_t getOverlapIndex (const GenomicRegion r)
- size_t getOverlapIndex (const std::pair< size_t, size_t > p)
- size_t getOverlapIndex (const size_t start, const size_t end)
- size_t getOverlapIndex (const size_t point)
- std::vector< GenomicRegion > getFeatures ()
- std::vector< GenomicRegion > getFeaturesInRange (const GenomicRegion r)
- std::vector< GenomicRegion > getFeaturesInRange (const std::pair< size_t, size_t > p)
- std::vector< GenomicRegion > getFeaturesInRange (const size_t start, const size_t end)
- std::vector< GenomicRegion > getFeaturesAt (const size_t index)
- size_t size ()
- size_t numFeatures ()
- std::vector< NCNode > getNodes ()
- GenomicRegion getRegionAt (size_t i)

Public Attributes

- std::vector< NCNode > _items

5.37.1 Detailed Description

A container class for quickly finding intersections of genomic intervals.

This class prepresents an implementation of Nested Containment Lists by Alekseyenko and Lee.

Alekseyenko AV and Lee CJ. Nested Containment List (NCList): a new algorithm for accelerating interval query of genome alignment and interval databases. Bioinformatics. 2007 Jun 1;23(11):1386-93. Epub 2007 Jan 18.

5.37.2 Constructor & Destructor Documentation

5.37.2.1 NCList::NCList (std::vector< GenomicRegion > & *regions*) [inline]

Constructor for NCList that performs the work of sorting and inserting intevals into the NCNodes.

5.37.3 Member Function Documentation

5.37.3.1 std::vector<GenomicRegion> NCList::getFeatures ( ) [inline]

A method to return all genes as a vector

5.37.3.2 std::vector<GenomicRegion> NCList::getFeaturesAt (const size_t *index*) [inline]

Method to return all genes at a particular index in the list

5.37.3.3 std::vector<GenomicRegion> NCList::getFeaturesInRange (const GenomicRegion *r*) [inline]

Method to return all genes in a given range

5.37.3.4 std::vector<GenomicRegion> NCList::getFeaturesInRange (const std::pair< size_t, size_t > *p*) [inline]

Method to return all genes between a start end pair

5.37.3.5 std::vector<GenomicRegion> NCList::getFeaturesInRange (const size_t *start,* const size_t *end*) [inline]

Method to return all genes between a start end interval

5.37.3.6 std::vector<NCNode> NCList::getNodes ( ) [inline]

return the vector of NCNodes

5.37.3.7 size_t NCList::getOverlapIndex (const GenomicRegion *r*) [inline]

Get the list index of the overlapped region

5.37.3.8 size_t NCList::getOverlapIndex (const std::pair< size_t, size_t > *p*) [inline]

Get the list index of the overlapped start end pair

5.37.3.9 size_t NCList::getOverlapIndex (const size_t *start,* const size_t *end*) [inline]

Get the list index of the overlapped start end interval

5.37.3.10 size_t NCList::getOverlapIndex (const size_t *point*) [inline]

Get the list index of the overlapped coordinate

5.37.3.11 GenomicRegion NCList::getRegionAt (size_t *i*) [inline]

returns the top level region at an index i

5.37.3.12 size_t NCList::numFeatures ( ) [inline]

numFeatures returns number of all genes/features

5.37.3.13 size_t NCList::size ( ) [inline]

size returns number of top level nodes

5.37.4 Member Data Documentation

5.37.4.1 std::vector<NCNode> NCList::_items

List of NCNodes as a vector

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

- ggtk/NCList.hpp

5.38 NCList::NCNode Class Reference

`#include <ggtk/NCList.hpp>`

Public Member Functions

- NCNode (GenomicRegion region, std::vector< GenomicRegion > &containments)
- bool contains (const GenomicRegion g)
- bool contains (const std::pair< size_t, size_t > p)
- bool contains (const size_t start, const size_t end)
- bool contains (const size_t point)
- bool overlaps (const GenomicRegion g)
- bool overlaps (const std::pair< size_t, size_t > p)
- bool overlaps (const size_t start, const size_t end)
- GenomicRegion getRegion ()

Public Attributes

- GenomicRegion _region
- NCList * _containments

5.38.1 Detailed Description

Data structure NCNode, NCList = List of NCNodes

5.38.2 Constructor & Destructor Documentation

5.38.2.1 NCList::NCNode::NCNode (GenomicRegion *region,* std::vector< GenomicRegion > & *containments*) [inline]

Node Constructor Most of the work done by NCList Constructor

5.38.3 Member Function Documentation

5.38.3.1 bool NCList::NCNode::contains (const GenomicRegion *g*) [inline]

contains a Genomic Region Overloaded methods to check for containment or overlap

Contianment |----------—| |----------—|

5.38.3.2 bool NCList::NCNode::contains (const std::pair< size_t, size_t > *p*) [inline]

contains a start end pair

5.38.3.3 bool NCList::NCNode::contains (const size_t *start,* const size_t *end*) [inline]

contains start and end interval

5.38.3.4 bool NCList::NCNode::contains (const size_t *point*) [inline]

contains a single coordinate

5.38.3.5 GenomicRegion NCList::NCNode::getRegion ( ) [inline]

get the current region

5.38.3.6 bool NCList::NCNode::overlaps (const GenomicRegion *g*) [inline]

overlaps a genomic region

Overlap |----------—| |--------| or |---------—|

5.38.3.7 bool NCList::NCNode::overlaps (const std::pair< size_t, size_t > *p*) [inline]

overlaps a start end pair

5.38.3.8 bool NCList::NCNode::overlaps (const size_t *start,* const size_t *end*) [inline]

overlaps a start and end interval

5.38.4 Member Data Documentation

5.38.4.1 NCList* NCList::NCNode::_containments

List of all regions completely contained by the top level regions interval.

5.38.4.2 GenomicRegion NCList::NCNode::_region

Top level region interval.

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

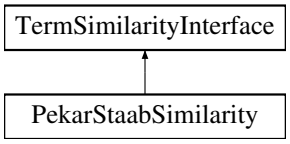
- ggtk/NCList.hpp

5.39 PekarStaabSimilarity Class Reference

A class to calculate PekarStaab similarity between 2 terms.

`#include <ggtk/PekarStaabSimilarity.hpp>`

Inheritance diagram for PekarStaabSimilarity:



Public Member Functions

- PekarStaabSimilarity (GoGraph *goGraph, TermDepthMap &icMap)

  A constructor.
- double calculateTermSimilarity (std::string goTermA, std::string goTermB)

  A method for calculating term-to-term similarity for GO terms using Pekar Staab similarity.
- double calculateNormalizedTermSimilarity (std::string goTermA, std::string goTermB)

  A method for calculating term-to-term similarity for GO terms using Normalized Pekar Staab similarity.
- std::string getLCA (boost::unordered_set< std::string > &ancestorsA, boost::unordered_set< std::string > &ancestorsB)

  A method for calculating the least common ancestor.

5.39.1 Detailed Description

A class to calculate PekarStaab similarity between 2 terms.

This class calculates Pekar Staab similarity.

V. Pekar and S. Staab, "Taxonomy learning: factoring the structure of a taxonomy into a semantic classification decision," in Proc. of 19th International Conference on Computational Linguistics. Morristown NJ USA: Association for Computational Linguistics, pp. 1-7, 2002.

H. Yu, L. Gao, K. Tu, and Z. Guo, "Broadly predicting specific gene functions with expression similarity and taxonomy similarity," Gene, vol. 352, pp. 75-81, Jun 6 2005.

lowest common ancestor (LCA) GraphDist(LCA,root)/(GraphDist(a,LCA)+GraphDist(b,LCA)+GraphDist(LCA,root))

5.39.2 Constructor & Destructor Documentation

5.39.2.1 PekarStaabSimilarity::PekarStaabSimilarity (GoGraph * *goGraph,* TermDepthMap & *icMap*) [inline]

A constructor.

Creates the default(empty) StandardRelationshipPolicy

5.39.3 Member Function Documentation

5.39.3.1 double PekarStaabSimilarity::calculateNormalizedTermSimilarity (std::string *goTermA,* std::string *goTermB*) [inline], [virtual]

A method for calculating term-to-term similarity for GO terms using Normalized Pekar Staab similarity.

This method returns the PekarStaab similarity scaled between 0 and 1 [0,1] inclusive

Implements TermSimilarityInterface.

5.39.3.2 double PekarStaabSimilarity::calculateTermSimilarity (std::string *goTermA,* std::string *goTermB*) [inline], [virtual]

A method for calculating term-to-term similarity for GO terms using Pekar Staab similarity.

This method returns the PekarStaab similarity.

Implements TermSimilarityInterface.

5.39.3.3 std::string PekarStaabSimilarity::getLCA (boost::unordered_set< std::string > & *ancestorsA,* boost::unordered_set< std::string > & *ancestorsB*) [inline]

A method for calculating the least common ancestor.

This method searches the sets to determine the deepest common ancestor

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

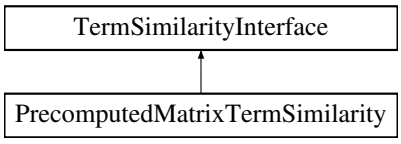
- ggtk/PekarStaabSimilarity.hpp

5.40 PrecomputedMatrixTermSimilarity Class Reference

A class to calculate similarity between go terms for 2 sets using a precomuted term similarity matrix.

`#include <ggtk/PrecomputedMatrixTermSimilarity.hpp>`

Inheritance diagram for PrecomputedMatrixTermSimilarity:



Public Member Functions

- PrecomputedMatrixTermSimilarity (std::string matrix_file)

  A constructor.
- double calculateTermSimilarity (std::string goTermA, std::string goTermB)

  A method for calculating term-to-term similarity for GO terms using a precomputed similarity matrix.
- double calculateNormalizedTermSimilarity (std::string goTermA, std::string goTermB)

  A method for calculating term-to-term similarity for GO terms using a precomputed similarity matrix.
- std::vector< double > projectTermSet (const std::vector< std::string > &terms)

  This method projects a set of terms into it the kernel space.

5.40.1 Detailed Description

A class to calculate similarity between go terms for 2 sets using a precomuted term similarity matrix.

This class allows the term similarity calculation to be decoupled from term set (gene) similarity measure that use them. Term similarity is loaded from a matrix file.

5.40.2 Constructor & Destructor Documentation

5.40.2.1 PrecomputedMatrixTermSimilarity::PrecomputedMatrixTermSimilarity (std::string *matrix_file*) [inline]

A constructor.

Parses a matrix file and creates the PrecomputedMatrixTermSimilarity object

5.40.3 Member Function Documentation

5.40.3.1 double PrecomputedMatrixTermSimilarity::calculateNormalizedTermSimilarity (std::string *goTermA*, std::string *goTermB*) [inline], [virtual]

A method for calculating term-to-term similarity for GO terms using a precomputed similarity matrix.

This method returns the similarity scaled between 0 and 1 [0,1] inclusive

Implements TermSimilarityInterface.

5.40.3.2 double PrecomputedMatrixTermSimilarity::calculateTermSimilarity (std::string *goTermA*, std::string *goTermB*) [inline], [virtual]

A method for calculating term-to-term similarity for GO terms using a precomputed similarity matrix.

This method returns the term similarity as defined by the matrix.

Implements TermSimilarityInterface.

5.40.3.3 std::vector<double> PrecomputedMatrixTermSimilarity::projectTermSet (const std::vector< std::string > & *terms*) [inline]

This method projects a set of terms into it the kernel space.

This method treats the term similarity matrix as a kernel and projects a set of terms into it.

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

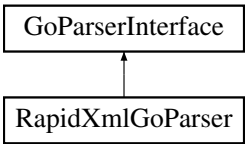
- ggtk/PrecomputedMatrixTermSimilarity.hpp

5.41 RapidXmlGoParser Class Reference

This class parses a go XML file using RapidXML library.

`#include <ggtk/RapidXmlGoParser.hpp>`

Inheritance diagram for RapidXmlGoParser:



Public Member Functions

- GoGraph * parseGoFile (std::string filename)
  Method to parse the go file, should be an XML file.
- bool isFileGood (const std::string &filename)
  A method to test if a file fits the accepted format.
- GoParserInterface * clone ()
  A method to create a new instance of this class for use in a factory.

5.41.1 Detailed Description

This class parses a go XML file using RapidXML library.

This class parses a Gene Ontology XML file using RapidXML library.

Implements GoParserInterface

5.41.2 Member Function Documentation

5.41.2.1 GoParserInterface* RapidXmlGoParser::clone ( ) [inline], [virtual]

A method to create a new instance of this class for use in a factory.

creats a new pointer to the parser, used by the factory for go parsers.

Implements GoParserInterface.

5.41.2.2 bool RapidXmlGoParser::isFileGood (const std::string & *filename*) [inline], [virtual]

A method to test if a file fits the accepted format.

Returns true if the file matches accepted format, false otherwise

Implements GoParserInterface.

5.41.2.3 GoGraph* RapidXmlGoParser::parseGoFile (std::string *filename*) [inline], [virtual]

Method to parse the go file, should be an XML file.

This method will read a Gene Ontology XML file and add all relationships to the graph.

Implements GoParserInterface.

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

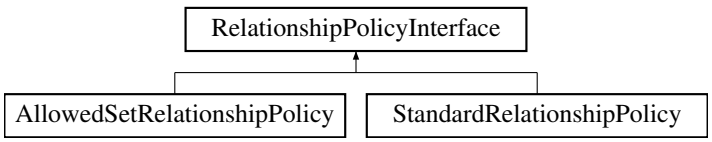
- ggtk/RapidXmlGoParser.hpp

5.42 RelationshipPolicyInterface Class Reference

An interface to check relationships between GO terms.

`#include <ggtk/RelationshipPolicyInterface.hpp>`

Inheritance diagram for RelationshipPolicyInterface:



Public Member Functions

- virtual bool isAllowed (GO::Relationship relationship)=0
  A pure virtual meethod to test if a relationship is allowed.

5.42.1 Detailed Description

An interface to check relationships between GO terms.

This is interface is used to create parsers which will only use a specific set of relationship when parsing a GO graph file.

5.42.2 Member Function Documentation

5.42.2.1 virtual bool RelationshipPolicyInterface::isAllowed (GO::Relationship *relationship*) [pure virtual]

A pure virtual meethod to test if a relationship is allowed.

This pure virtual method requires any subclass to imlement an isAllowed method to enforce the relationship pollicy.

Implemented in AllowedSetRelationshipPolicy, and StandardRelationshipPolicy.

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

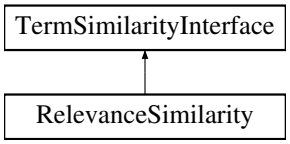
- ggtk/RelationshipPolicyInterface.hpp

5.43 RelevanceSimilarity Class Reference

A class to calculate Relevance similarity between 2 terms.

`#include <ggtk/RelevanceSimilarity.hpp>`

Inheritance diagram for RelevanceSimilarity:



Public Member Functions

- RelevanceSimilarity (GoGraph *goGraph, TermInformationContentMap &icMap)
  A constructor.
- double calculateTermSimilarity (std::string goTermA, std::string goTermB)
  A method for calculating term-to-term similarity for GO terms using Relevance similarity.
- double calculateNormalizedTermSimilarity (std::string goTermA, std::string goTermB)
  A method for calculating term-to-term similarity for GO terms using Normalized Relevance similarity.
- std::string getMICA (boost::unordered_set< std::string > &ancestorsA, boost::unordered_set< std::string > &ancestorsB)
  A method for calculating the most informative common ancestor.

5.43.1 Detailed Description

A class to calculate Relevance similarity between 2 terms.

This class calculates Relevance similarity.

A. Schlicker, F. S. Domingues, J. Rahnenfuhrer, and T. Lengauer, "A new measure for functional similarity of gene products based on Gene Ontology," BMC Bioinformatics, vol. 7, p. 302, 2006.

P. W. Lord, R. D. Stevens, A. Brass, and C. A. Goble, "Semantic similarity measures as tools for exploring the gene ontology," Pac Symp Biocomput, pp. 601-12, 2003.

Basically this is Lin similarity scaled by the complement of the probability of the mica 2 * IC(MICA) / (IC(termA) + IC(termB))*(1-p(Mica))

5.43.2 Constructor & Destructor Documentation

5.43.2.1 RelevanceSimilarity::RelevanceSimilarity (GoGraph * *goGraph,* TermInformationContentMap & *icMap*) [inline]

A constructor.

Creates the default(empty) StandardRelationshipPolicy

5.43.3 Member Function Documentation

5.43.3.1 double RelevanceSimilarity::calculateNormalizedTermSimilarity (std::string *goTermA,* std::string *goTermB*) [inline], [virtual]

A method for calculating term-to-term similarity for GO terms using Normalized Relevance similarity.

This method returns the Relevance similarity scaled between 0 and 1 [0,1] inclusive

Implements TermSimilarityInterface.

5.43.3.2 double RelevanceSimilarity::calculateTermSimilarity (std::string *goTermA,* std::string *goTermB*) [inline], [virtual]

A method for calculating term-to-term similarity for GO terms using Relevance similarity.

This method returns the Relevance similarity.

Implements TermSimilarityInterface.

5.43.3.3 std::string RelevanceSimilarity::getMICA (boost::unordered_set< std::string > & *ancestorsA,* boost::unordered_set< std::string > & *ancestorsB*) [inline]

A method for calculating the most informative common ancestor.

This method searches the sets to determine the most informatics ancestor.

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

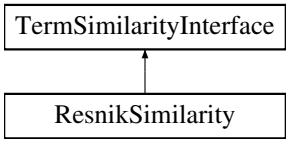
- ggtk/RelevanceSimilarity.hpp

5.44 ResnikSimilarity Class Reference

A class to calculate resnik similarity between 2 terms.

`#include <ggtk/ResnikSimilarity.hpp>`

Inheritance diagram for ResnikSimilarity:



Public Member Functions

- ResnikSimilarity (GoGraph *goGraph, TermInformationContentMap &icMap)

  A constructor.
- double calculateTermSimilarity (std::string goTermA, std::string goTermB)

  A method for calculating term-to-term similarity for GO terms using Resnik similarity.
- double calculateNormalizedTermSimilarity (std::string goTermA, std::string goTermB)

  A method for calculating term-to-term similarity for GO terms using Normalized Resnik similarity.
- std::string getMICA (boost::unordered_set< std::string > &ancestorsA, boost::unordered_set< std::string > &ancestorsB)

  A method for calculating the most informative common ancestor.

5.44.1 Detailed Description

A class to calculate resnik similarity between 2 terms.

This class calculates Resnik similarity. Philip Resnik (1995). "Using information content to evaluate semantic similarity in a taxonomy". In Chris S. Mellish (Ed.). Proceedings of the 14th international joint conference on Artificial intelligence (IJCAI'95)

P. W. Lord, R. D. Stevens, A. Brass, and C. A. Goble, "Semantic similarity measures as tools for exploring the gene ontology," Pac Symp Biocomput, pp. 601-12, 2003.

maximun information content of all shared ancestors IC(MICA)

5.44.2 Constructor & Destructor Documentation

5.44.2.1 ResnikSimilarity::ResnikSimilarity (GoGraph * *goGraph,* TermInformationContentMap & *icMap*) [inline]

A constructor.

Creates the default(empty) StandardRelationshipPolicy

5.44.3 Member Function Documentation

5.44.3.1 double ResnikSimilarity::calculateNormalizedTermSimilarity (std::string *goTermA,* std::string *goTermB*) [inline], [virtual]

A method for calculating term-to-term similarity for GO terms using Normalized Resnik similarity.

This method returns the Resnik similarity divided by the maximum possible similarity

Implements TermSimilarityInterface.

5.44.3.2 double ResnikSimilarity::calculateTermSimilarity (std::string *goTermA*, std::string *goTermB*) [inline], [virtual]

A method for calculating term-to-term similarity for GO terms using Resnik similarity.

This method returns the Resnik similarity or the information content of the most informative common ancestor.

Implements TermSimilarityInterface.

5.44.3.3 std::string ResnikSimilarity::getMICA (boost::unordered_set< std::string > & *ancestorsA*, boost::unordered_set< std::string > & *ancestorsB*) [inline]

A method for calculating the most informative common ancestor.

This method searches the sets to determine the most informatics ancestor.

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

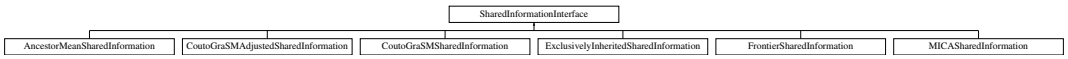
- ggtk/ResnikSimilarity.hpp

5.45 SharedInformationInterface Class Reference

An interface class to define shared information calculations.

`#include <ggtk/SharedInformationInterface.hpp>`

Inheritance diagram for SharedInformationInterface:



Public Member Functions

- virtual double sharedInformation (const std::string &termA, const std::string &termB)=0

  A pure virtual method for returning the shared information of two terms.
- virtual double sharedInformation (const std::string &term)=0

  A pure virtual method for returning the shared information of a single terms,or information content.
- virtual double maxInformationContent (const std::string &term)=0

  A pure virtual method for returning the maximum information content for a term.
- virtual bool hasTerm (const std::string &term)=0

  A pure virtual method for determining if a term can be found.
- virtual bool isSameOntology (const std::string &termA, const std::string &termB)=0

  A pure virtual method for determining if the two terms are of like ontologies.

5.45.1 Detailed Description

An interface class to define shared information calculations.

This class defines the interface for shared information calculations. Pure virtual methods require that shared information methods implement these.

5.45.2 Member Function Documentation

5.45.2.1 virtual bool SharedInformationInterface::hasTerm (const std::string & *term*) [pure virtual]

A pure virtual method for determining if a term can be found.

This pure virtual method requires any shared information class to implement this method. This method provides a method for client classes to determine if a term can be found by the method.

Implemented in CoutoGraSMSharedInformation, CoutoGraSMAdjustedSharedInformation, FrontierSharedInformation, ExclusivelyInheritedSharedInformation, MICASharedInformation, and AncestorMeanSharedInformation.

5.45.2.2 virtual bool SharedInformationInterface::isSameOntology (const std::string & *termA,* const std::string & *termB*) [pure virtual]

A pure virtual method for determining if the two terms are of like ontologies.

This pure virtual method requires any shared information class to implement this method. This method provides a method for client classes to determine if two terms are of the same ontology.

Implemented in CoutoGraSMSharedInformation, CoutoGraSMAdjustedSharedInformation, FrontierSharedInformation, ExclusivelyInheritedSharedInformation, MICASharedInformation, and AncestorMeanSharedInformation.

5.45.2.3 virtual double SharedInformationInterface::maxInformationContent (const std::string & *term*) [pure virtual]

A pure virtual method for returning the maximum information content for a term.

This pure virtual method requires any shared information class to implement this method. This method provides the absolute max information content with in a corpus for normalization purposes.

Implemented in CoutoGraSMSharedInformation, CoutoGraSMAdjustedSharedInformation, FrontierSharedInformation, ExclusivelyInheritedSharedInformation, MICASharedInformation, and AncestorMeanSharedInformation.

5.45.2.4 virtual double SharedInformationInterface::sharedInformation (const std::string & *termA,* const std::string & *termB*) [pure virtual]

A pure virtual method for returning the shared information of two terms.

This pure virtual method requires any shared information class to implement this method.

Implemented in CoutoGraSMSharedInformation, CoutoGraSMAdjustedSharedInformation, FrontierSharedInformation, ExclusivelyInheritedSharedInformation, MICASharedInformation, and AncestorMeanSharedInformation.

5.45.2.5 virtual double SharedInformationInterface::sharedInformation (const std::string & *term*) [pure virtual]

A pure virtual method for returning the shared information of a single terms,or information content.

This pure virtual method privdes a mechanism for returing a term's infromation content.

Implemented in CoutoGraSMSharedInformation, CoutoGraSMAdjustedSharedInformation, FrontierSharedInformation, ExclusivelyInheritedSharedInformation, MICASharedInformation, and AncestorMeanSharedInformation.

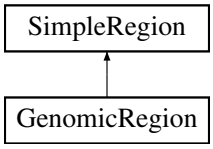
The documentation for this class was generated from the following file:

- ggtk/SharedInformationInterface.hpp

5.46 SimpleRegion Class Reference

`#include <ggtk/SimpleRegion.hpp>`

Inheritance diagram for SimpleRegion:



Public Member Functions

- SimpleRegion ()
- SimpleRegion (const size_t id, const size_t start, const size_t end)
- const size_t midpoint () const
- int distance (const SimpleRegion &rhs)
- bool contains (const SimpleRegion &rhs)
- bool contains (const std::pair< size_t, size_t > &rhs)
- bool contains (const size_t rhs_start, const size_t rhs_end)
- bool contains (const size_t point)
- bool overlaps (const SimpleRegion &rhs)
- bool overlaps (const std::pair< size_t, size_t > &rhs)
- bool overlaps (const size_t rhs_start, const size_t rhs_end)
- size_t getStart ()
- size_t getEnd ()
- size_t getId ()

Public Attributes

- size_t _id
- size_t _start
- size_t _end

5.46.1 Detailed Description

A Region class to be used with NCList This class is used to represent region and genome data.

5.46.2 Constructor & Destructor Documentation

5.46.2.1 SimpleRegion::SimpleRegion ( ) [inline]

Default constructor Initializes a simple region.

5.46.2.2 SimpleRegion::SimpleRegion (const size_t *id*, const size_t *start*, const size_t *end*) [inline]

Parameterized constructor Allows the creation of a regions with user specified values

5.46.3 Member Function Documentation

5.46.3.1 bool SimpleRegion::contains (const **SimpleRegion** & *rhs*) [inline]

contains test with SimpleRegion objects. Does the region contain rhs?

5.46.3.2 bool SimpleRegion::contains (const std::pair< size_t, size_t > & *rhs*) [inline]

Tests the contains relationship with a start end pair Does the region contain the start end pair?

5.46.3.3 bool SimpleRegion::contains (const size_t *rhs_start*, const size_t *rhs_end*) [inline]

Tests the contains relationship with start and end coordinates Does the region contain the interval between start and end?

5.46.3.4 bool SimpleRegion::contains (const size_t *point*) [inline]

Tests the contains relationship with a single point Does the region contain the point coordinate?

5.46.3.5 int SimpleRegion::distance (const **SimpleRegion** & *rhs*) [inline]

Simple distance between two regions. Calculates the distance between two regions by calculating the distance between their midpoints.

5.46.3.6 size_t SimpleRegion::getEnd ( ) [inline]

accessor for end coordinate returns the interval's end coordinate

5.46.3.7 size_t SimpleRegion::getId ( ) [inline]

accessor for the id returns the interval's id

5.46.3.8 size_t SimpleRegion::getStart ( ) [inline]

accessor for start coordinate returns the interval's start coordinate

5.46.3.9 const size_t SimpleRegion::midpoint ( ) const [inline]

Calculates the midpoint of a region Simple mid point of a region. (end - start)/2

5.46.3.10 bool SimpleRegion::overlaps (const SimpleRegion & *rhs*) [inline]

Tests the overlap relationship with a region Does the region overlap the rhs region?

5.46.3.11 bool SimpleRegion::overlaps (const std::pair< size_t, size_t > & *rhs*) [inline]

Tests the overlap relationship with a start end pair Does the region overlap the start end pair?

5.46.3.12 bool SimpleRegion::overlaps (const size_t *rhs_start,* const size_t *rhs_end*) [inline]

Tests the overlap relationship with start and end coordinates Does the region overlap the interval defines by start and end?

5.46.4 Member Data Documentation

5.46.4.1 size_t SimpleRegion::_end

the end coordinate for the region

5.46.4.2 size_t SimpleRegion::_id

a generic id for the region

5.46.4.3 size_t SimpleRegion::_start

the start coordinate for the region

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

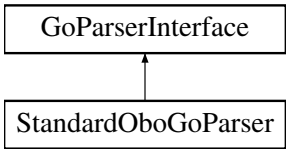
- ggtk/SimpleRegion.hpp

5.47 StandardOboGoParser Class Reference

A class to parse only is_a or part_of relationships.

`#include <ggtk/StandardOboGoParser.hpp>`

Inheritance diagram for StandardOboGoParser:



Public Member Functions

- GoGraph * parseGoFile (std::string filename)
  Method to parse the go file, should be an XML file.
- bool isFileGood (const std::string &filename)
  A method to test if a file fits the accepted format.
- GoParserInterface * clone ()
  a method to create a new instance of this class for use in a factory
- StandardOboGoParser ()
  A parameterized constructor.

5.47.1 Detailed Description

A class to parse only is_a or part_of relationships.

Implements GoParserInterface

5.47.2 Constructor & Destructor Documentation

5.47.2.1 StandardOboGoParser::StandardOboGoParser ( ) [inline]

A parameterized constructor.

constructor that sets the policy

5.47.3 Member Function Documentation

5.47.3.1 GoParserInterface* StandardOboGoParser::clone ( ) [inline], [virtual]

a method to create a new instance of this class for use in a factory

creats a new pointer to the parser, used by the factory for go parsers.

Implements GoParserInterface.

5.47.3.2 bool StandardOboGoParser::isFileGood (const std::string & *filename*) [inline], [virtual]

A method to test if a file fits the accepted format.

Returns true if the file matches accepted format, false otherwise

Implements GoParserInterface.

5.47.3.3 GoGraph* StandardOboGoParser::parseGoFile (std::string *filename*) [inline], [virtual]

Method to parse the go file, should be an XML file.

This method will read a Gene Ontology XML file and add only those relationship which are specified to the graph.

Implements GoParserInterface.

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

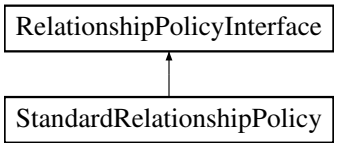
- ggtk/StandardOboGoParser.hpp

5.48 StandardRelationshipPolicy Class Reference

A class to allow only a set of relationships.

`#include <ggtk/StandardRelationshipPolicy.hpp>`

Inheritance diagram for StandardRelationshipPolicy:



Public Member Functions

- StandardRelationshipPolicy ()
  A constructor.
- StandardRelationshipPolicy (std::vector< GO::Relationship > relationships)
  A parameterized constructor.
- bool isAllowed (GO::Relationship relationship)
  a method to test if a relatinoship is allowed or not

5.48.1 Detailed Description

A class to allow only a set of relationships.

A class to allow only certain relationships in the go graph. It uses a set of enums to restric the types of relationships considered in a graph.

5.48.2 Constructor & Destructor Documentation

5.48.2.1 StandardRelationshipPolicy::StandardRelationshipPolicy ( ) [inline]

A constructor.

Creates the default(empty) StandardRelationshipPolicy

5.48.2.2 StandardRelationshipPolicy::StandardRelationshipPolicy (std::vector< GO::Relationship > *relationships*) [inline]

A parameterized constructor.

Creats the StandardRelationshipPolicy using a list(vector) of relationships to allow

5.48.3 Member Function Documentation

5.48.3.1 bool StandardRelationshipPolicy::isAllowed (GO::Relationship *relationship*) [inline], [virtual]

a method to test if a relatinoship is allowed or not

tests if the relationship is allowed. Overridden to fulfill the RelationshipPolicyInterface

Implements RelationshipPolicyInterface.

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

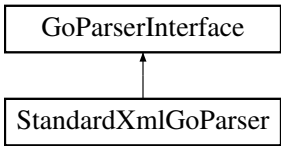
- ggtk/StandardRelationshipPolicy.hpp

5.49 StandardXmlGoParser Class Reference

A class to parse only is_a or part_of relationships.

`#include <ggtk/StandardXmlGoParser.hpp>`

Inheritance diagram for StandardXmlGoParser:



Public Member Functions

- GoGraph * parseGoFile (std::string filename)

  Method to parse the go file, should be an XML file.
- bool isFileGood (const std::string &filename)

  A method to test if a file fits the accepted format.
- GoParserInterface * clone ()

  a method to create a new instance of this class for use in a factory
- StandardXmlGoParser ()

  A parameterized constructor.

5.49.1 Detailed Description

A class to parse only is_a or part_of relationships.

Implements GoParserInterface

5.49.2 Constructor & Destructor Documentation

5.49.2.1 StandardXmlGoParser::StandardXmlGoParser ( ) [inline]

A parameterized constructor.

constructor that sets the policy

5.49.3 Member Function Documentation

5.49.3.1 GoParserInterface* StandardXmlGoParser::clone ( ) [inline], [virtual]

a method to create a new instance of this class for use in a factory

creats a new pointer to the parser, used by the factory for go parsers.

Implements GoParserInterface.

5.49.3.2 bool StandardXmlGoParser::isFileGood (const std::string & *filename*) [inline], [virtual]

A method to test if a file fits the accepted format.

Returns true if the file matches accepted format, false otherwise

Implements GoParserInterface.

5.49.3.3 GoGraph* StandardXmlGoParser::parseGoFile (std::string *filename*) [inline], [virtual]

Method to parse the go file, should be an XML file.

This method will read a Gene Ontology XML file and add only those relationship which are specified to the graph.

Implements GoParserInterface.

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

- ggtk/StandardXmlGoParser.hpp

5.50 TermDepthMap Class Reference

A class to calculate the depth of a GO term in the ontology.

`#include <ggtk/TermDepthMap.hpp>`

Public Member Functions

- TermDepthMap (GoGraph *graph)
  A parameterized constructor.
- TermDepthMap ()
  A default constructor.
- ~TermDepthMap ()
  A default desctructor.
- std::vector< double > getValues ()
  Accessor for probablities vector.
- std::vector< std::string > getKeys ()
  Function to return all the keys in the map.
- bool hasTerm (std::string testTerm)
  Method to test if the id exists in the map.
- double operator[ ] (std::string termId)
  Overloaded [] bracket operator to mimic Map.
- double getValue (std::string termId)
  Mapping function to return the value mapped by key.

Protected Attributes

- boost::unordered_map< std::string, std::size_t > _nameToIndex
  A private map that returns the index of a term.
- std::vector< double > _depths
  A private list of term depths.

5.50.1 Detailed Description

A class to calculate the depth of a GO term in the ontology.

This class provides a map that returns the depth of a GO term. This method is used in graph and edge based similarity methods to calculate a node's depth

5.50.2 Constructor & Destructor Documentation

5.50.2.1 TermDepthMap::TermDepthMap (GoGraph * *graph*) [inline]

A parameterized constructor.

This constructor takes pointers to GoGraph and AnnotationData objects. Only the parameterized construtor is allowed to ensure these objects are created with valid parameters.

5.50.2.2 TermDepthMap::TermDepthMap ( ) [inline]

A default constructor.

This constructor initialized the storage structures. Should not be used.

5.50.2.3 TermDepthMap::~TermDepthMap ( ) [inline]

A default desctructor.

This desctructor clears the containters

5.50.3 Member Function Documentation

5.50.3.1 std::vector<std::string> TermDepthMap::getKeys ( ) [inline]

Function to return all the keys in the map.

Returns all valid keys in the map.

5.50.3.2 double TermDepthMap::getValue (std::string *termId*) [inline]

Mapping function to return the value mapped by key.

Get the value mapped by the given key. A specified function for the [] operator

5.50.3.3 std::vector<double> TermDepthMap::getValues ( ) [inline]

Accessor for probablities vector.

Get the vector of values

5.50.3.4 bool TermDepthMap::hasTerm (std::string *testTerm*) [inline]

Method to test if the id exists in the map.

Return true the id is found, false if not

5.50.3.5 double TermDepthMap::operator[ ] (std::string *termId*) [inline]

Overloaded [] bracket operator to mimic Map.

This defines a bracket operator to access the data inside of the map. This is done to mimic the behavior of the map class

5.50.4 Member Data Documentation

5.50.4.1 std::vector<double> TermDepthMap::_depths [protected]

A private list of term depths.

This vector of doubles holds the depth for each term

5.50.4.2 boost::unordered_map<std::string,std::size_t> TermDepthMap::_nameToIndex [protected]

A private map that returns the index of a term.

This map takes string term ids and returns the index for annotation count access.

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

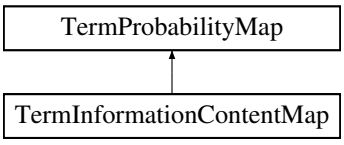
- ggtk/TermDepthMap.hpp

5.51 TermInformationContentMap Class Reference

A class to calculate the information content of a GO term.

`#include <ggtk/TermInformationContentMap.hpp>`

Inheritance diagram for TermInformationContentMap:



Public Member Functions

- TermInformationContentMap (GoGraph *graph, AnnotationData *annoData)

  A parameterized constructor.
- double badIdValue ()

  Return a default value for a term that does not exist.
- TermInformationContentMap ()

  A default constructor.

Additional Inherited Members

5.51.1 Detailed Description

A class to calculate the information content of a GO term.

This class provides a map that returns the information content of a GO term. This class is used by Information Content methods.

5.51.2 Constructor & Destructor Documentation

5.51.2.1 TermInformationContentMap::TermInformationContentMap (GoGraph * *graph*, AnnotationData * *annoData*) [inline]

A parameterized constructor.

This constructor takes pointers to GoGraph and AnnotationData objects. Only the parameterized construtor is allowed to ensure these objects are created with valid parameters. This constructor relies on the TermProbability↩ Map.

5.51.2.2 TermInformationContentMap::TermInformationContentMap ( ) [inline]

A default constructor.

This constructor creates an empty IC map. Should not be used.

5.51.3 Member Function Documentation

5.51.3.1 double TermInformationContentMap::badIdValue ( ) [inline], [virtual]

Return a default value for a term that does not exist.

A value to return if the term is not found (does not exist in the map). Returns informaiton content 0. This may not be the ideal behavior.

Reimplemented from TermProbabilityMap.

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

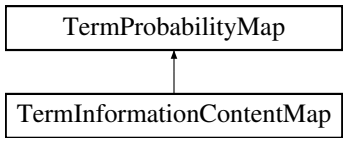
- ggtk/TermInformationContentMap.hpp

5.52 TermProbabilityMap Class Reference

A class to calculate the probability of a GO term.

`#include <ggtk/TermProbabilityMap.hpp>`

Inheritance diagram for TermProbabilityMap:



Classes

- class dfs_cumulative_annotations_visitor

  Depth first search boost visitor.

Public Member Functions

- TermProbabilityMap (GoGraph *graph, AnnotationData *annoData)

  A parameterized constructor.
- TermProbabilityMap ()

  A default constructor.
- ~TermProbabilityMap ()

  A default desctructor.
- std::vector< double > getValues ()

  Accessor for probablities vector.
- std::vector< std::string > getKeys ()

  Function to return all the keys in the map.
- bool hasTerm (const std::string &testTerm)

  Method to test if the id exists in the map.
- virtual double badIdValue ()

  Return a default value for a term that does not exist.
- double operator[ ] (std::string termId)

  Overloaded [] bracket operator to mimic Map.
- double getValue (std::string termId)

  Mapping function to return the value mapped by key.
- double getMinBP ()

  Get the specific minimum probability for BP.
- double getMinMF ()

  Get the specific minimum probability for MF.
- double getMinCC ()

  Get the specific minimum probability for CC.

Protected Attributes

- boost::unordered_map< std::string, std::size_t > _nameToIndex

  A private map that returns the index of a term.
- std::vector< double > _probabilities

  A private list of term probabilities.
- bool _isSingleAnnoMin

  A flag designating the minimum policy.
- double _bp_normalization_min_1anno

  Normalization factor for calculating normalized simialrites Biological Process.
- double _bp_normalization_min_minAnno

  Normalization factor for calculating normalized simialrites for Biological Process.
- double _mf_normalization_min_1anno

  Normalization factor for calculating normalized simialrites Molecular Function.
- double _mf_normalization_min_minAnno

  Normalization factor for calculating normalized simialrites for Molecular Function.
- double _cc_normalization_min_1anno

  Normalization factor for calculating normalized simialrites Cellular Component.
- double _cc_normalization_min_minAnno

  Normalization factor for calculating normalized simialrites for Cellular Component.

5.52.1 Detailed Description

A class to calculate the probability of a GO term.

This class provides a map that returns the probability of GO term. This class is used by Information Content methods to determine the prior probability of a term give an instance of AnnotationData.

5.52.2 Constructor & Destructor Documentation

5.52.2.1 TermProbabilityMap::TermProbabilityMap (GoGraph * *graph*, AnnotationData * *annoData*) [inline]

A parameterized constructor.

This constructor takes pointers to GoGraph and AnnotationData objects. Only the parameterized construtor is allowed to ensure these objects are created with valid parameters.

5.52.2.2 TermProbabilityMap::TermProbabilityMap ( ) [inline]

A default constructor.

This constructor initialized the storage structures. Should not be used.

5.52.2.3 TermProbabilityMap::~TermProbabilityMap ( ) [inline]

A default desctructor.

This desctructor clears the containters

5.52.3 Member Function Documentation

5.52.3.1 virtual double TermProbabilityMap::badIdValue ( ) [inline], [virtual]

Return a default value for a term that does not exist.

A value to return if the term is not found (does not exist in the map). Returns probability 1 or certanty. This may not be the ideal behavior.

Reimplemented in TermInformationContentMap.

5.52.3.2 std::vector<std::string> TermProbabilityMap::getKeys ( ) [inline]

Function to return all the keys in the map.

Returns all valid keys in the map.

5.52.3.3 double TermProbabilityMap::getMinBP ( ) [inline]

Get the specific minimum probability for BP.

This function returns the minimum probablity for the bp ontology

5.52.3.4 double TermProbabilityMap::getMinCC ( ) [inline]

Get the specific minimum probability for CC.

This function returns the minimum probablity for the cc ontology

5.52.3.5 double TermProbabilityMap::getMinMF ( ) [inline]

Get the specific minimum probability for MF.

This function returns the minimum probablity for the mf ontology

5.52.3.6 double TermProbabilityMap::getValue (std::string *termId*) [inline]

Mapping function to return the value mapped by key.

Get the value mapped by the given key. A specified function for the [] operator

5.52.3.7 std::vector<double> TermProbabilityMap::getValues ( ) [inline]

Accessor for probablities vector.

Get the vector of values

5.52.3.8 bool TermProbabilityMap::hasTerm (const std::string & *testTerm*) [inline]

Method to test if the id exists in the map.

Return true the id is found, false if not

5.52.3.9 double TermProbabilityMap::operator[ ] (std::string *termId*) [inline]

Overloaded [] bracket operator to mimic Map.

This defines a bracket operator to access the data inside of the map. This is done to mimic the behavior of the map class

5.52.4 Member Data Documentation

5.52.4.1 double TermProbabilityMap::_bp_normalization_min_1anno [protected]

Normalization factor for calculating normalized simialrites Biological Process.

Normalization factor representing the minimum probability using a single annotation devided by the cumulative annotations.

5.52.4.2 double TermProbabilityMap::_bp_normalization_min_minAnno [protected]

Normalization factor for calculating normalized simialrites for Biological Process.

Normalization factor representing the minimum probability using the number of annotations of the least probable term devided by the cumulative annotations.

5.52.4.3 double TermProbabilityMap::_cc_normalization_min_1anno [protected]

Normalization factor for calculating normalized simialrites Cellular Component.

Normalization factor representing the minimum probability using a single annotation devided by the cumulative annotations.

5.52.4.4 double TermProbabilityMap::_cc_normalization_min_minAnno [protected]

Normalization factor for calculating normalized simialrites for Cellular Component.

Normalization factor representing the minimum probability using the number of annotations of the least probable term devided by the cumulative annotations.

5.52.4.5 bool TermProbabilityMap::_isSingleAnnoMin [protected]

A flag designating the minimum policy.

This flag will be true and return true is single annotation probability is used, false otherwise.

5.52.4.6 double TermProbabilityMap::_mf_normalization_min_1anno [protected]

Normalization factor for calculating normalized simialrites Molecular Function.

Normalization factor representing the minimum probability using a single annotation devided by the cumulative annotations.

5.52.4.7 double TermProbabilityMap::_mf_normalization_min_minAnno [protected]

Normalization factor for calculating normalized simialrites for Molecular Function.

Normalization factor representing the minimum probability using the number of annotations of the least probable term devided by the cumulative annotations.

5.52.4.8 boost::unordered_map<std::string,std::size_t> TermProbabilityMap::_nameToIndex [protected]

A private map that returns the index of a term.

This map takes string term ids and returns the index for annotation count access.

5.52.4.9 std::vector<double> TermProbabilityMap::_probabilities [protected]

A private list of term probabilities.

This vector of doubles holds the prior probability for each term

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

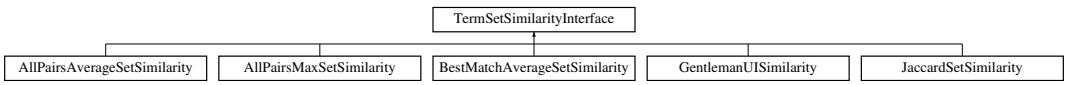
- ggtk/TermProbabilityMap.hpp

5.53 TermSetSimilarityInterface Class Reference

An interface class for comparing semantic similarity of sets of GO terms.

`#include <ggtk/TermSetSimilarityInterface.hpp>`

Inheritance diagram for TermSetSimilarityInterface:



Public Member Functions

- virtual double calculateSimilarity (const boost::unordered_set< std::string > &termsA, const boost::unordered_set< std::string > &termsB)=0

  A pure virtual method for calculating term set to term set similarity for sets of GO terms.

5.53.1 Detailed Description

An interface class for comparing semantic similarity of sets of GO terms.

This class defines the interface for comparing term set to term set similarity. This is the most useful case for comparing genes with multiple annotations to each other.

5.53.2 Member Function Documentation

5.53.2.1 virtual double TermSetSimilarityInterface::calculateSimilarity (const boost::unordered_set< std::string > & *termsA*, const boost::unordered_set< std::string > & *termsB*) [pure virtual]

A pure virtual method for calculating term set to term set similarity for sets of GO terms.

This pure virtual method requires similarity measure to implement the basic interface that returns a similarity value for two sets of go terms.

Implemented in AllPairsMaxSetSimilarity, AllPairsAverageSetSimilarity, BestMatchAverageSetSimilarity, and JaccardSetSimilarity.

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

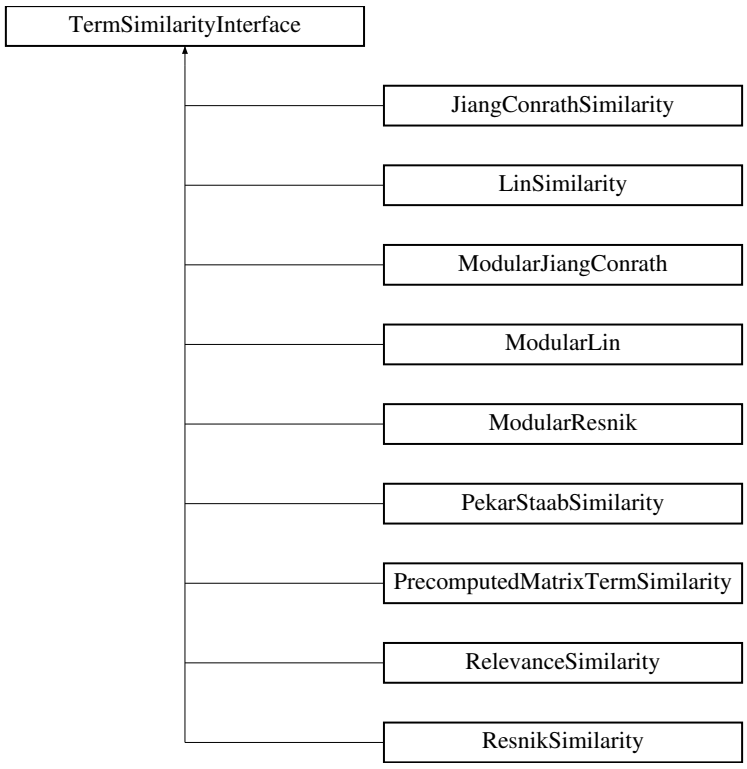
- ggtk/TermSetSimilarityInterface.hpp

5.54 TermSimilarityInterface Class Reference

An interface class for comparing semantic similarity of GO terms.

`#include <ggtk/TermSimilarityInterface.hpp>`

Inheritance diagram for TermSimilarityInterface:



Public Member Functions

- virtual double calculateTermSimilarity (std::string goTermA, std::string goTermB)=0

  A pure virtual method for calculating term-to-term similarity for GO terms.
- virtual double calculateNormalizedTermSimilarity (std::string goTermA, std::string goTermB)=0

  A pure virtual method for calculating term-to-term similarity for GO terms.

5.54.1 Detailed Description

An interface class for comparing semantic similarity of GO terms.

This class defines the interface for comparing term-to-term GO similarity.

5.54.2 Member Function Documentation

5.54.2.1 virtual double TermSimilarityInterface::calculateNormalizedTermSimilarity (std::string *goTermA,* std::string *goTermB*) [pure virtual]

A pure virtual method for calculating term-to-term similarity for GO terms.

This pure virtual method requires similarity measure to implement the basic interface that returns a similarity value for two go terms. This version of the function must be normalzied. Returning similarity between 0 and 1

Implemented in PrecomputedMatrixTermSimilarity, JiangConrathSimilarity, PekarStaabSimilarity, LinSimilarity, RelevanceSimilarity, ResnikSimilarity, ModularJiangConrath, ModularLin, and ModularResnik.

5.54.2.2 virtual double TermSimilarityInterface::calculateTermSimilarity (std::string *goTermA,* std::string *goTermB*) [pure virtual]

A pure virtual method for calculating term-to-term similarity for GO terms.

This pure virtual method requires similarity measure to implement the basic interface that returns a similarity value for two go terms.

Implemented in PrecomputedMatrixTermSimilarity, JiangConrathSimilarity, PekarStaabSimilarity, ModularJiang↩Conrath, RelevanceSimilarity, LinSimilarity, ModularLin, ResnikSimilarity, and ModularResnik.

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

- ggtk/TermSimilarityInterface.hpp

5.55 TermSimilarityWriter Class Reference

A class write a term similarity matrix to file. Companion to PrecomputedMatrixTermSimilarity.

```
#include <ggtk/TermSimilarityWriter.hpp>
```

Public Member Functions

- TermSimilarityWriter (GoGraph *goGraph, AnnotationData *annoData)

  Parameterized constructor.
- void writeSimilarityMatrix (TermSimilarityInterface *termSim, std::string fileName, long ontology_code)

  A method to write a term similarity matrix.
- void writeSimilarityMatrixBP (TermSimilarityInterface *termSim, std::string fileName)

  A method to write a term similarity matrix for Biological Process.
- void writeSimilarityMatrixMF (TermSimilarityInterface *termSim, std::string fileName)

  A method to write a term similarity matrix for Molecular Function.
- void writeSimilarityMatrixCC (TermSimilarityInterface *termSim, std::string fileName)

  A method to write a term similarity matrix for Cellular Component.

5.55.1 Detailed Description

A class write a term similarity matrix to file. Companion to PrecomputedMatrixTermSimilarity.

This class writes a term similarity matrix to file by first calculating all similarity values between pairs of terms. This class will be use to write matrices read by PrecomputedMatrixTermSimilarity.

5.55.2 Constructor & Destructor Documentation

5.55.2.1 TermSimilarityWriter::TermSimilarityWriter (GoGraph * *goGraph,* AnnotationData * *annoData*) [inline]

Parameterized constructor.

A simple parameterized constructor. This class take an instance of the GO Graph.

5.55.3 Member Function Documentation

5.55.3.1 void TermSimilarityWriter::writeSimilarityMatrix (TermSimilarityInterface * *termSim,* std::string *fileName,* long *ontology_code*) [inline]

A method to write a term similarity matrix.

Calculates and writes the similarity matrix to file. Calculates the similarity between all pairs of terms. The complexity is O(N^2) * O(Term Pair Calculation Cost). O(Term Pair Calculation Cost) is usually near constant time, but some method will be extremely slow and require other methods

5.55.3.2 void TermSimilarityWriter::writeSimilarityMatrixBP (TermSimilarityInterface * *termSim,* std::string *fileName*) [inline]

A method to write a term similarity matrix for Biological Process.

Calcualtes and writes the biological process term similarity matrix

5.55.3.3 void TermSimilarityWriter::writeSimilarityMatrixCC (TermSimilarityInterface * *termSim*, std::string *fileName*) [inline]

A method to write a term similarity matrix for Cellular Component.

Calcualtes and writes the cellular component term similarity matrix

5.55.3.4 void TermSimilarityWriter::writeSimilarityMatrixMF (TermSimilarityInterface * *termSim*, std::string *fileName*) [inline]

A method to write a term similarity matrix for Molecular Function.

Calcualtes and writes the molecular function term similarity matrix

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

- ggtk/TermSimilarityWriter.hpp

5.56 GoGraph::VertexProps Struct Reference

A Vertex Property object.

`#include <ggtk/GoGraph.hpp>`

Public Member Functions

- ~VertexProps ()

Public Attributes

- std::string termId
- GO::Onto ontology

5.56.1 Detailed Description

A Vertex Property object.

This struct represent the data needed by each vertex. Boost provides constant time access to these members by querying them using the vertex and graph objects (graphVar[vertex].termId ... etc.).

5.56.2 Constructor & Destructor Documentation

5.56.2.1 GoGraph::VertexProps::~VertexProps ( ) [inline]

Desctructor

5.56.3 Member Data Documentation

5.56.3.1 GO::Onto GoGraph::VertexProps::ontology

The ontology of the GO term, BP, MF, CC

5.56.3.2 std::string GoGraph::VertexProps::termId

The term id of the go term, the GO accession, GO:########.

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

- ggtk/GoGraph.hpp

Index