

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

Generated by Doxygen 1.8.11

Contents

1	Namespace Index	1
1.1	Namespace List	1
2	Hierarchical Index	3
2.1	Class Hierarchy	3
3	Class Index	5
3.1	Class List	5
4	Namespace Documentation	9
4.1	Accumulators Namespace Reference	9
4.1.1	Detailed Description	10
4.1.2	Typedef Documentation	10
4.1.2.1	CovarianceAccumulator	10
4.1.2.2	MaxAccumulator	10
4.1.2.3	MeanAccumulator	10
4.1.2.4	MinAccumulator	10
4.1.2.5	SimpleAccumulator	11
4.1.2.6	VarianceAccumulator	11
4.1.3	Function Documentation	11
4.1.3.1	extractCovariance(const CovarianceAccumulator &acc)	11
4.1.3.2	extractMax(const MaxAccumulator &acc)	11
4.1.3.3	extractMax(const SimpleAccumulator &acc)	11
4.1.3.4	extractMean(const MeanAccumulator &acc)	11
4.1.3.5	extractMean(const SimpleAccumulator &acc)	11

4.1.3.6	extractMin(const MinAccumulator &acc)	12
4.1.3.7	extractMin(const SimpleAccumulator &acc)	12
4.1.3.8	extractSD(const VarianceAccumulator &acc)	12
4.1.3.9	extractVariance(const VarianceAccumulator &acc)	12
4.2	AppUtilities Namespace Reference	12
4.2.1	Detailed Description	13
4.2.2	Function Documentation	13
4.2.2.1	checkParams(const std::map< std::string, std::string > &paramMap, const std::vector< std::string > &params, std::string &message)	13
4.2.2.2	paramList(std::string param_str)	13
4.2.2.3	parseParamFile(const std::string fname, std::map< std::string, std::string > &paramMap)	13
4.2.2.4	parseSimpleFile(const std::string fname)	13
4.2.2.5	setToVec(const boost::unordered_set< std::string > &uset)	13
4.2.2.6	splitTake(const std::string &name, const char &sep, const size_t &n)	14
4.2.2.7	vecToSet(const std::vector< std::string > &vec)	14
4.3	EnrichmentTools Namespace Reference	14
4.3.1	Detailed Description	14
4.3.2	Function Documentation	14
4.3.2.1	enrichmentSignificance(GoGraph *go, AnnotationData *data, boost::unordered_set< std::string > &genes, const std::string &term)	14
4.3.2.2	getDescendantGenes(GoGraph *go, AnnotationData *data, const std::string &term)	15
4.3.2.3	oneSidedRawPvalue_hyper(size_t sample, size_t success, size_t population, size_t test_value)	15
4.4	GO Namespace Reference	15
4.4.1	Detailed Description	16
4.4.2	Enumeration Type Documentation	16
4.4.2.1	EvidenceCode	16
4.4.2.2	Onto	16
4.4.2.3	Relationship	16
4.4.3	Function Documentation	17
4.4.3.1	evidenceStringToCode(std::string code)	17
4.4.3.2	evidenceToString(const EvidenceCode &evidence)	17
4.4.3.3	ontologyStringToCode(std::string code)	17
4.4.3.4	ontologyToString(const Onto &o)	17
4.4.3.5	relationshipStringToCode(std::string code)	17
4.4.3.6	relationshipToString(const Relationship &relationship)	17
4.5	SetUtilities Namespace Reference	17
4.5.1	Detailed Description	17

5	Class Documentation	19
5.1	AllowedRelationshipOboGoParser Class Reference	19
5.1.1	Detailed Description	19
5.1.2	Constructor & Destructor Documentation	20
5.1.2.1	AllowedRelationshipOboGoParser(RelationshipPolicyInterface *policy)	20
5.1.3	Member Function Documentation	20
5.1.3.1	clone()	20
5.1.3.2	isFileGood(const std::string &filename)	20
5.1.3.3	parseGoFile(std::string filename)	20
5.1.3.4	setPolicy(RelationshipPolicyInterface *policy)	20
5.1.3.5	splitWith(const std::string &instr, const std::string &splitStr, std::string &attr, std::string &value)	20
5.2	AllowedRelationshipXmlGoParser Class Reference	21
5.2.1	Detailed Description	21
5.2.2	Constructor & Destructor Documentation	21
5.2.2.1	AllowedRelationshipXmlGoParser(RelationshipPolicyInterface *policy)	21
5.2.3	Member Function Documentation	22
5.2.3.1	clone()	22
5.2.3.2	isFileGood(const std::string &filename)	22
5.2.3.3	parseGoFile(std::string filename)	22
5.2.3.4	setPolicy(RelationshipPolicyInterface *policy)	22
5.3	AllowedSetEvidencePolicy Class Reference	22
5.3.1	Detailed Description	23
5.3.2	Constructor & Destructor Documentation	23
5.3.2.1	AllowedSetEvidencePolicy()	23
5.3.2.2	AllowedSetEvidencePolicy(std::vector< GO::EvidenceCode > evidenceCodes)	23
5.3.3	Member Function Documentation	23
5.3.3.1	addEvidence(GO::EvidenceCode evidenceCode)	23
5.3.3.2	addEvidence(const std::string &stringCode)	24
5.3.3.3	isAllowed(GO::EvidenceCode evidenceCode)	24
5.3.3.4	isEmpty()	24

5.4	AllowedSetRelationshipPolicy Class Reference	24
5.4.1	Detailed Description	25
5.4.2	Constructor & Destructor Documentation	25
5.4.2.1	AllowedSetRelationshipPolicy()	25
5.4.2.2	AllowedSetRelationshipPolicy(std::vector< GO::Relationship > relationships)	25
5.4.3	Member Function Documentation	25
5.4.3.1	addRelationship(GO::Relationship relationship)	25
5.4.3.2	addRelationship(const std::string &relString)	25
5.4.3.3	isAllowed(GO::Relationship relationship)	25
5.4.3.4	isEmpty()	26
5.5	AllPairsAverageSetSimilarity Class Reference	26
5.5.1	Detailed Description	26
5.5.2	Constructor & Destructor Documentation	26
5.5.2.1	AllPairsAverageSetSimilarity(TermSimilarityInterface *simMeasure)	26
5.5.3	Member Function Documentation	27
5.5.3.1	calculateSimilarity(const boost::unordered_set< std::string > &termsA, const boost::unordered_set< std::string > &termsB)	27
5.6	AllPairsMaxSetSimilarity Class Reference	27
5.6.1	Detailed Description	27
5.6.2	Constructor & Destructor Documentation	28
5.6.2.1	AllPairsMaxSetSimilarity(TermSimilarityInterface *simMeasure)	28
5.6.3	Member Function Documentation	28
5.6.3.1	calculateSimilarity(const boost::unordered_set< std::string > &termsA, const boost::unordered_set< std::string > &termsB)	28
5.7	AncestorMeanSharedInformation Class Reference	28
5.7.1	Detailed Description	29
5.7.2	Constructor & Destructor Documentation	29
5.7.2.1	AncestorMeanSharedInformation(GoGraph *goGraph, TermInformation↔ ContentMap &icMap)	29
5.7.3	Member Function Documentation	29
5.7.3.1	hasTerm(const std::string &term)	29
5.7.3.2	isSameOntology(const std::string &termA, const std::string &termB)	29

5.7.3.3	<code>maxInformationContent(const std::string &term)</code>	29
5.7.3.4	<code>sharedInformation(const std::string &termA, const std::string &termB)</code>	30
5.7.3.5	<code>sharedInformation(const std::string &term)</code>	30
5.8	AnnotationData Class Reference	30
5.8.1	Detailed Description	31
5.8.2	Constructor & Destructor Documentation	32
5.8.2.1	<code>AnnotationData()</code>	32
5.8.2.2	<code>~AnnotationData()</code>	32
5.8.3	Member Function Documentation	32
5.8.3.1	<code>addAssociation(const std::string &gene, const std::string &goTerm, const std::string &evidenceCode)</code>	32
5.8.3.2	<code>addGenesForGoTerm(const std::string &goTerm, boost::unordered_set< std::string > &geneSet)</code>	32
5.8.3.3	<code>getAllGenes()</code>	32
5.8.3.4	<code>getAllGoTerms()</code>	32
5.8.3.5	<code>getGenesEvidenceForGoTerm(const std::string &goTerm)</code>	33
5.8.3.6	<code>getGenesForGoTerm(const std::string &goTerm)</code>	33
5.8.3.7	<code>getGoTermsEvidenceForGene(const std::string &gene)</code>	33
5.8.3.8	<code>getGoTermsForGene(const std::string &gene)</code>	33
5.8.3.9	<code>getGoTermsForGeneBP(const std::string &gene, GoGraph *G)</code>	33
5.8.3.10	<code>getGoTermsForGeneByOntology(const std::string &gene, GO::Onto filter, Ontology, GoGraph *G)</code>	33
5.8.3.11	<code>getGoTermsForGeneCC(const std::string &gene, GoGraph *G)</code>	33
5.8.3.12	<code>getGoTermsForGeneMF(const std::string &gene, GoGraph *G)</code>	34
5.8.3.13	<code>getNumAnnotationsForGene(const std::string &gene)</code>	34
5.8.3.14	<code>getNumAnnotationsForGoTerm(const std::string &goTerm)</code>	34
5.8.3.15	<code>getNumGenes()</code>	34
5.8.3.16	<code>getNumGoTerms()</code>	34
5.8.3.17	<code>getOntologyTerms(GoGraph *graph, GO::Onto ontology)</code>	34
5.8.3.18	<code>hasGene(const std::string &gene)</code>	34
5.8.3.19	<code>hasGoTerm(const std::string &goTerm)</code>	35
5.8.4	Member Data Documentation	35

5.8.4.1	_genes	35
5.8.4.2	_geneToGos	35
5.8.4.3	_geneToGosEvidence	35
5.8.4.4	_goTerms	35
5.8.4.5	_goToGenes	35
5.8.4.6	_goToGenesEvidence	35
5.8.4.7	_stringToGene	36
5.8.4.8	_stringToGo	36
5.9	AnnotationParserFactory Class Reference	36
5.9.1	Detailed Description	36
5.9.2	Constructor & Destructor Documentation	37
5.9.2.1	AnnotationParserFactory()	37
5.9.2.2	~AnnotationParserFactory()	37
5.9.3	Member Function Documentation	37
5.9.3.1	addParser(std::string name, AnnotationParserInterface *parser)	37
5.9.3.2	getParser(std::string name)	37
5.10	AnnotationParserInterface Class Reference	37
5.10.1	Detailed Description	38
5.10.2	Member Function Documentation	38
5.10.2.1	clone()=0	38
5.10.2.2	isFileGood(const std::string &fileName)=0	38
5.10.2.3	parseAnnotationFile(std::string fileName)=0	38
5.11	BestMatchAverageSetSimilarity Class Reference	39
5.11.1	Detailed Description	39
5.11.2	Constructor & Destructor Documentation	39
5.11.2.1	BestMatchAverageSetSimilarity(TermSimilarityInterface *simMeasure)	39
5.11.3	Member Function Documentation	39
5.11.3.1	calculateSimilarity(const boost::unordered_set< std::string > &termsA, const boost::unordered_set< std::string > &termsB)	39
5.12	CoutoGraSMAAdjustedSharedInformation Class Reference	40
5.12.1	Detailed Description	40

5.12.2	Constructor & Destructor Documentation	41
5.12.2.1	CoutoGraSMAadjustedSharedInformation(GoGraph *goGraph, TermInformation↵ ContentMap &icMap)	41
5.12.3	Member Function Documentation	41
5.12.3.1	getCommonDisjointAncestors(const std::string &termC1, const std::string &termC2)	41
5.12.3.2	getNumPaths(const std::string &termA, const std::string &termB)	41
5.12.3.3	hasTerm(const std::string &term)	41
5.12.3.4	isDisjoint(const std::string &termC, const std::string &termA1, const std::string &termA2)	41
5.12.3.5	isSameOntology(const std::string &termA, const std::string &termB)	41
5.12.3.6	maxInformationContent(const std::string &term)	42
5.12.3.7	sharedInformation(const std::string &termA, const std::string &termB)	42
5.12.3.8	sharedInformation(const std::string &term)	42
5.13	CoutoGraSMSharedInformation Class Reference	42
5.13.1	Detailed Description	43
5.13.2	Constructor & Destructor Documentation	43
5.13.2.1	CoutoGraSMSharedInformation(GoGraph *goGraph, TermInformationContent↵ Map &icMap)	43
5.13.3	Member Function Documentation	43
5.13.3.1	getCommonDisjointAncestors(const std::string &termC1, const std::string &termC2)	43
5.13.3.2	getNumPaths(const std::string &termA, const std::string &termB)	44
5.13.3.3	hasTerm(const std::string &term)	44
5.13.3.4	isDisjoint(const std::string &termC, const std::string &termA1, const std::string &termA2)	44
5.13.3.5	isSameOntology(const std::string &termA, const std::string &termB)	44
5.13.3.6	maxInformationContent(const std::string &term)	44
5.13.3.7	sharedInformation(const std::string &termA, const std::string &termB)	44
5.13.3.8	sharedInformation(const std::string &term)	45
5.14	TermProbabilityMap::dfs_cumulative_annotations_visitor Class Reference	45
5.14.1	Detailed Description	46
5.14.2	Member Function Documentation	46

5.14.2.1	<code>finish_vertex(Vertex u, const Graph &g)</code>	46
5.15	<code>DisallowedSetEvidencePolicy</code> Class Reference	46
5.15.1	Detailed Description	47
5.15.2	Constructor & Destructor Documentation	47
5.15.2.1	<code>DisallowedSetEvidencePolicy()</code>	47
5.15.2.2	<code>DisallowedSetEvidencePolicy(std::vector< GO::EvidenceCode > evidenceCodes)</code>	47
5.15.3	Member Function Documentation	47
5.15.3.1	<code>addEvidence(GO::EvidenceCode evidenceCode)</code>	47
5.15.3.2	<code>addEvidence(const std::string &stringCode)</code>	47
5.15.3.3	<code>isAllowed(GO::EvidenceCode evidenceCode)</code>	47
5.15.3.4	<code>isEmpty()</code>	48
5.16	<code>GoGraph::EdgeProps</code> Struct Reference	48
5.16.1	Detailed Description	48
5.16.2	Constructor & Destructor Documentation	48
5.16.2.1	<code>~EdgeProps()</code>	48
5.16.3	Member Data Documentation	48
5.16.3.1	<code>relType</code>	48
5.17	<code>EntrezGene2GoAnnotationParser</code> Class Reference	49
5.17.1	Detailed Description	49
5.17.2	Constructor & Destructor Documentation	49
5.17.2.1	<code>EntrezGene2GoAnnotationParser(EvidencePolicyInterface *policy)</code>	49
5.17.2.2	<code>EntrezGene2GoAnnotationParser()</code>	49
5.17.3	Member Function Documentation	50
5.17.3.1	<code>clone()</code>	50
5.17.3.2	<code>isFileGood(const std::string &fileName)</code>	50
5.17.3.3	<code>parseAnnotationFile(std::string filename)</code>	50
5.18	<code>EvidencePolicyInterface</code> Class Reference	50
5.18.1	Detailed Description	51
5.18.2	Member Function Documentation	51
5.18.2.1	<code>isAllowed(GO::EvidenceCode evidenceCode)=0</code>	51

5.19	ExclusivelyInheritedSharedInformation Class Reference	51
5.19.1	Detailed Description	52
5.19.2	Constructor & Destructor Documentation	52
5.19.2.1	ExclusivelyInheritedSharedInformation(GoGraph *goGraph, TermInformation↵ContentMap &icMap)	52
5.19.3	Member Function Documentation	52
5.19.3.1	getCommonDisjointAncestors(const std::string &termC1, const std::string &termC2)	52
5.19.3.2	hasTerm(const std::string &term)	52
5.19.3.3	isSameOntology(const std::string &termA, const std::string &termB)	53
5.19.3.4	maxInformationContent(const std::string &term)	53
5.19.3.5	sharedInformation(const std::string &termA, const std::string &termB)	53
5.19.3.6	sharedInformation(const std::string &term)	53
5.20	ExperimentalEvidencePolicy Class Reference	53
5.20.1	Detailed Description	54
5.20.2	Constructor & Destructor Documentation	54
5.20.2.1	ExperimentalEvidencePolicy()	54
5.21	FrontierSharedInformation Class Reference	54
5.21.1	Detailed Description	55
5.21.2	Constructor & Destructor Documentation	55
5.21.2.1	FrontierSharedInformation(GoGraph *goGraph, TermInformationContentMap &icMap)	55
5.21.3	Member Function Documentation	55
5.21.3.1	getCommonDisjointAncestors(const std::string &termC1, const std::string &termC2)	55
5.21.3.2	hasTerm(const std::string &term)	55
5.21.3.3	isSameOntology(const std::string &termA, const std::string &termB)	55
5.21.3.4	maxInformationContent(const std::string &term)	55
5.21.3.5	sharedInformation(const std::string &termA, const std::string &termB)	56
5.21.3.6	sharedInformation(const std::string &term)	56
5.22	GafAnnotationParser Class Reference	56
5.22.1	Detailed Description	57

5.22.2	Constructor & Destructor Documentation	57
5.22.2.1	GafAnnotationParser()	57
5.22.2.2	GafAnnotationParser(EvidencePolicyInterface *policy)	57
5.23	GenomicRegion Class Reference	57
5.23.1	Detailed Description	58
5.23.2	Constructor & Destructor Documentation	58
5.23.2.1	GenomicRegion(const std::string chrom, const size_t start, const size_t end, const size_t id, std::string name, std::string desc)	58
5.23.2.2	GenomicRegion(const size_t id, const size_t start, const size_t end)	58
5.23.2.3	GenomicRegion()	58
5.23.3	Member Function Documentation	58
5.23.3.1	getChrom()	58
5.23.3.2	getDesc()	58
5.23.3.3	getName()	58
5.23.4	Member Data Documentation	59
5.23.4.1	_chrom	59
5.23.4.2	_desc	59
5.23.4.3	_name	59
5.24	GentlemanUISimilarity Class Reference	59
5.24.1	Detailed Description	59
5.24.2	Constructor & Destructor Documentation	60
5.24.2.1	GentlemanUISimilarity(TermSimilarityInterface *simMeasure)	60
5.24.3	Member Function Documentation	60
5.24.3.1	calculateSimilarity(boost::unordered_set< std::string > termsA, boost↵ ::unordered_set< std::string > termsB)	60
5.25	GoaAnnotationParser Class Reference	60
5.25.1	Detailed Description	61
5.25.2	Constructor & Destructor Documentation	61
5.25.2.1	GoaAnnotationParser(EvidencePolicyInterface *policy)	61
5.25.2.2	GoaAnnotationParser()	61
5.25.3	Member Function Documentation	61

5.25.3.1	clone()	61
5.25.3.2	isFileGood(const std::string &fileName)	61
5.25.3.3	parseAnnotationFile(std::string filename)	61
5.26	GoGraph Class Reference	62
5.26.1	Detailed Description	64
5.26.2	Member Typedef Documentation	64
5.26.2.1	EdgeIndexMap	64
5.26.2.2	GoEdge	64
5.26.2.3	GoVertex	65
5.26.2.4	GoVertexIterator	65
5.26.2.5	Graph	65
5.26.2.6	Graph_t	65
5.26.2.7	InEdgeIterator	65
5.26.2.8	OutEdgeIterator	65
5.26.2.9	VertexIndexMap	65
5.26.3	Constructor & Destructor Documentation	66
5.26.3.1	~GoGraph()	66
5.26.4	Member Function Documentation	66
5.26.4.1	filterSetForBP(const std::vector< std::string > &inSet)	66
5.26.4.2	filterSetForBP(const boost::unordered_set< std::string > &inSet)	66
5.26.4.3	filterSetForCC(const std::vector< std::string > &inSet)	66
5.26.4.4	filterSetForCC(const boost::unordered_set< std::string > &inSet)	66
5.26.4.5	filterSetForMF(const std::vector< std::string > &inSet)	66
5.26.4.6	filterSetForMF(const boost::unordered_set< std::string > &inSet)	67
5.26.4.7	filterSetForOntology(const boost::unordered_set< std::string > &inSet, GO::Onto onto)	67
5.26.4.8	filterSetForOntology(const std::vector< std::string > &inSet, GO::Onto onto)	67
5.26.4.9	getAllTerms()	67
5.26.4.10	getAllTermsBP()	67
5.26.4.11	getAllTermsCC()	67
5.26.4.12	getAllTermsMF()	67

5.26.4.13	<code>getAncestorTerms(const std::string &term)</code>	68
5.26.4.14	<code>getChildTerms(const std::string &term)</code>	68
5.26.4.15	<code>getDescendantTerms(const std::string &term)</code>	68
5.26.4.16	<code>getGraph()</code>	68
5.26.4.17	<code>getInducedSubgraph(const std::string &termId)</code>	68
5.26.4.18	<code>getInducedSubgraph2(const std::string &termId)</code>	68
5.26.4.19	<code>getNumComponents()</code>	68
5.26.4.20	<code>getNumEdges()</code>	69
5.26.4.21	<code>getNumVertices()</code>	69
5.26.4.22	<code>getOntologyTerms(GO::Onto ontology)</code>	69
5.26.4.23	<code>getParentTerms(const std::string &term)</code>	69
5.26.4.24	<code>getRoot()</code>	69
5.26.4.25	<code>getTermDescription(std::string term)</code>	69
5.26.4.26	<code>getTermDescriptionByIndex(std::size_t index)</code>	69
5.26.4.27	<code>getTermIndex(const std::string &term)</code>	69
5.26.4.28	<code>getTermName(std::string term)</code>	70
5.26.4.29	<code>getTermNameByIndex(std::size_t index)</code>	70
5.26.4.30	<code>getTermOntology(const std::string &term)</code>	70
5.26.4.31	<code>getTermOntologyByIndex(std::size_t index)</code>	70
5.26.4.32	<code>getTermOntologyByVertex(GoVertex vertex)</code>	70
5.26.4.33	<code>getTermStringIdByIndex(std::size_t index)</code>	70
5.26.4.34	<code>getVertexByIndex(std::size_t index)</code>	70
5.26.4.35	<code>getVertexByName(const std::string &term)</code>	70
5.26.4.36	<code>getVertexIndex(GoVertex vertex)</code>	71
5.26.4.37	<code>hasTerm(const std::string &term)</code>	71
5.26.4.38	<code>initMaps()</code>	71
5.26.4.39	<code>insertRelationship(const std::string &termParent, const std::string &termChild, const std::string &relationship)</code>	71
5.26.4.40	<code>insertTerm(const std::string &termId, const std::string &name, const std::string &description, const std::string &ontology)</code>	71
5.27	GoParserFactory Class Reference	71

5.27.1 Detailed Description	72
5.27.2 Constructor & Destructor Documentation	72
5.27.2.1 GoParserFactory()	72
5.27.2.2 ~GoParserFactory()	72
5.27.3 Member Function Documentation	72
5.27.3.1 addParser(std::string name, GoParserInterface *parser)	72
5.27.3.2 getParser(std::string name)	73
5.28 GoParserInterface Class Reference	73
5.28.1 Detailed Description	73
5.28.2 Member Function Documentation	73
5.28.2.1 clone()=0	73
5.28.2.2 isFileGood(const std::string &filename)=0	74
5.28.2.3 parseGoFile(std::string fileName)=0	74
5.29 JaccardSetSimilarity Class Reference	74
5.29.1 Detailed Description	74
5.29.2 Constructor & Destructor Documentation	75
5.29.2.1 JaccardSetSimilarity()	75
5.29.3 Member Function Documentation	75
5.29.3.1 calculateSimilarity(const boost::unordered_set< std::string > &termsA, const boost::unordered_set< std::string > &termsB)	75
5.30 JiangConrathSimilarity Class Reference	75
5.30.1 Detailed Description	76
5.30.2 Constructor & Destructor Documentation	76
5.30.2.1 JiangConrathSimilarity(GoGraph *goGraph, TermInformationContentMap &icMap)	76
5.30.3 Member Function Documentation	76
5.30.3.1 calculateNormalizedTermSimilarity(std::string goTermA, std::string goTermB)	76
5.30.3.2 calculateTermSimilarity(std::string goTermA, std::string goTermB)	76
5.30.3.3 getMICA(boost::unordered_set< std::string > &ancestorsA, boost::unordered_set< std::string > &ancestorsB)	76
5.31 LinSimilarity Class Reference	77
5.31.1 Detailed Description	77

5.31.2	Constructor & Destructor Documentation	77
5.31.2.1	LinSimilarity(GoGraph *goGraph, TermInformationContentMap &icMap)	77
5.31.3	Member Function Documentation	78
5.31.3.1	calculateNormalizedTermSimilarity(std::string goTermA, std::string goTermB)	78
5.31.3.2	calculateTermSimilarity(std::string goTermA, std::string goTermB)	78
5.31.3.3	getMICA(boost::unordered_set< std::string > &ancestorsA, boost::unordered_set< std::string > &ancestorsB)	78
5.32	MgiAnnotationParser Class Reference	78
5.32.1	Detailed Description	79
5.32.2	Constructor & Destructor Documentation	79
5.32.2.1	MgiAnnotationParser()	79
5.32.2.2	MgiAnnotationParser(EvidencePolicyInterface *policy)	79
5.33	MICASharedInformation Class Reference	79
5.33.1	Detailed Description	80
5.33.2	Constructor & Destructor Documentation	80
5.33.2.1	MICASharedInformation(GoGraph *goGraph, TermInformationContentMap &icMap)	80
5.33.3	Member Function Documentation	80
5.33.3.1	hasTerm(const std::string &term)	80
5.33.3.2	isSameOntology(const std::string &termA, const std::string &termB)	81
5.33.3.3	maxInformationContent(const std::string &term)	81
5.33.3.4	sharedInformation(const std::string &termA, const std::string &termB)	81
5.33.3.5	sharedInformation(const std::string &term)	81
5.34	ModularJiangConrath Class Reference	81
5.34.1	Detailed Description	82
5.34.2	Constructor & Destructor Documentation	82
5.34.2.1	ModularJiangConrath(SharedInformationInterface *sharedInformationCalculator)	82
5.34.3	Member Function Documentation	82
5.34.3.1	calculateNormalizedTermSimilarity(std::string goTermA, std::string goTermB)	82
5.34.3.2	calculateTermSimilarity(std::string goTermA, std::string goTermB)	83
5.34.3.3	setSharedInformationCalculator(SharedInformationInterface *newSharedInformationCalculator)	83

5.35 ModularLin Class Reference	83
5.35.1 Detailed Description	84
5.35.2 Constructor & Destructor Documentation	84
5.35.2.1 ModularLin(SharedInformationInterface *sharedInformationCalculator)	84
5.35.3 Member Function Documentation	84
5.35.3.1 calculateNormalizedTermSimilarity(std::string goTermA, std::string goTermB)	84
5.35.3.2 calculateTermSimilarity(std::string goTermA, std::string goTermB)	84
5.35.3.3 setSharedInformationCalculator(SharedInformationInterface *newSharedInformationCalculator)	84
5.36 ModularResnik Class Reference	85
5.36.1 Detailed Description	85
5.36.2 Constructor & Destructor Documentation	85
5.36.2.1 ModularResnik(SharedInformationInterface *sharedInformationCalculator)	85
5.36.3 Member Function Documentation	86
5.36.3.1 calculateNormalizedTermSimilarity(std::string goTermA, std::string goTermB)	86
5.36.3.2 calculateTermSimilarity(std::string goTermA, std::string goTermB)	86
5.36.3.3 setSharedInformationCalculator(SharedInformationInterface *newSharedInformationCalculator)	86
5.37 NCList Class Reference	86
5.37.1 Detailed Description	87
5.37.2 Constructor & Destructor Documentation	87
5.37.2.1 NCList(std::vector< GenomicRegion > ®ions)	87
5.37.3 Member Function Documentation	87
5.37.3.1 getFeatures()	87
5.37.3.2 getFeaturesAt(const size_t index)	87
5.37.3.3 getFeaturesInRange(const GenomicRegion r)	88
5.37.3.4 getFeaturesInRange(const std::pair< size_t, size_t > p)	88
5.37.3.5 getFeaturesInRange(const size_t start, const size_t end)	88
5.37.3.6 getNodes()	88
5.37.3.7 getOverlapIndex(const GenomicRegion r)	88
5.37.3.8 getOverlapIndex(const std::pair< size_t, size_t > p)	88

5.37.3.9	<code>getOverlapIndex(const size_t start, const size_t end)</code>	88
5.37.3.10	<code>getOverlapIndex(const size_t point)</code>	88
5.37.3.11	<code>getRegionAt(size_t i)</code>	88
5.37.3.12	<code>numFeatures()</code>	88
5.37.3.13	<code>size()</code>	89
5.37.4	Member Data Documentation	89
5.37.4.1	<code>_items</code>	89
5.38	NList::NCNode Class Reference	89
5.38.1	Detailed Description	89
5.38.2	Constructor & Destructor Documentation	89
5.38.2.1	<code>NCNode(GenomicRegion region, std::vector< GenomicRegion > &containments)</code>	89
5.38.3	Member Function Documentation	90
5.38.3.1	<code>contains(const GenomicRegion g)</code>	90
5.38.3.2	<code>contains(const std::pair< size_t, size_t > p)</code>	90
5.38.3.3	<code>contains(const size_t start, const size_t end)</code>	90
5.38.3.4	<code>contains(const size_t point)</code>	90
5.38.3.5	<code>getRegion()</code>	90
5.38.3.6	<code>overlaps(const GenomicRegion g)</code>	90
5.38.3.7	<code>overlaps(const std::pair< size_t, size_t > p)</code>	90
5.38.3.8	<code>overlaps(const size_t start, const size_t end)</code>	90
5.38.4	Member Data Documentation	90
5.38.4.1	<code>_containments</code>	90
5.38.4.2	<code>_region</code>	91
5.39	PekarStaabSimilarity Class Reference	91
5.39.1	Detailed Description	91
5.39.2	Constructor & Destructor Documentation	92
5.39.2.1	<code>PekarStaabSimilarity(GoGraph *goGraph, TermDepthMap &icMap)</code>	92
5.39.3	Member Function Documentation	92
5.39.3.1	<code>calculateNormalizedTermSimilarity(std::string goTermA, std::string goTermB)</code>	92
5.39.3.2	<code>calculateTermSimilarity(std::string goTermA, std::string goTermB)</code>	92

5.39.3.3	<code>getLCA(boost::unordered_set< std::string > &ancestorsA, boost::unordered_set< std::string > &ancestorsB)</code>	92
5.40	PrecomputedMatrixTermSimilarity Class Reference	92
5.40.1	Detailed Description	93
5.40.2	Constructor & Destructor Documentation	93
5.40.2.1	<code>PrecomputedMatrixTermSimilarity(std::string matrix_file)</code>	93
5.40.3	Member Function Documentation	93
5.40.3.1	<code>calculateNormalizedTermSimilarity(std::string goTermA, std::string goTermB)</code>	93
5.40.3.2	<code>calculateTermSimilarity(std::string goTermA, std::string goTermB)</code>	93
5.40.3.3	<code>projectTermSet(const std::vector< std::string > &terms)</code>	94
5.41	RapidXmlGoParser Class Reference	94
5.41.1	Detailed Description	94
5.41.2	Member Function Documentation	94
5.41.2.1	<code>clone()</code>	94
5.41.2.2	<code>isFileGood(const std::string &filename)</code>	95
5.41.2.3	<code>parseGoFile(std::string filename)</code>	95
5.42	RelationshipPolicyInterface Class Reference	95
5.42.1	Detailed Description	95
5.42.2	Member Function Documentation	96
5.42.2.1	<code>isAllowed(GO::Relationship relationship)=0</code>	96
5.43	RelevanceSimilarity Class Reference	96
5.43.1	Detailed Description	96
5.43.2	Constructor & Destructor Documentation	97
5.43.2.1	<code>RelevanceSimilarity(GoGraph *goGraph, TermInformationContentMap &icMap)</code>	97
5.43.3	Member Function Documentation	97
5.43.3.1	<code>calculateNormalizedTermSimilarity(std::string goTermA, std::string goTermB)</code>	97
5.43.3.2	<code>calculateTermSimilarity(std::string goTermA, std::string goTermB)</code>	97
5.43.3.3	<code>getMICA(boost::unordered_set< std::string > &ancestorsA, boost::unordered_set< std::string > &ancestorsB)</code>	97
5.44	ResnikSimilarity Class Reference	97
5.44.1	Detailed Description	98

5.44.2	Constructor & Destructor Documentation	98
5.44.2.1	ResnikSimilarity(GoGraph *goGraph, TermInformationContentMap &icMap) . . .	98
5.44.3	Member Function Documentation	98
5.44.3.1	calculateNormalizedTermSimilarity(std::string goTermA, std::string goTermB) . .	98
5.44.3.2	calculateTermSimilarity(std::string goTermA, std::string goTermB)	99
5.44.3.3	getMICA(boost::unordered_set< std::string > &ancestorsA, boost::unordered← _set< std::string > &ancestorsB)	99
5.45	SharedInformationInterface Class Reference	99
5.45.1	Detailed Description	99
5.45.2	Member Function Documentation	100
5.45.2.1	hasTerm(const std::string &term)=0	100
5.45.2.2	isSameOntology(const std::string &termA, const std::string &termB)=0	100
5.45.2.3	maxInformationContent(const std::string &term)=0	100
5.45.2.4	sharedInformation(const std::string &termA, const std::string &termB)=0	100
5.45.2.5	sharedInformation(const std::string &term)=0	101
5.46	SimpleRegion Class Reference	101
5.46.1	Detailed Description	102
5.46.2	Constructor & Destructor Documentation	102
5.46.2.1	SimpleRegion()	102
5.46.2.2	SimpleRegion(const size_t id, const size_t start, const size_t end)	102
5.46.3	Member Function Documentation	102
5.46.3.1	contains(const SimpleRegion &rhs)	102
5.46.3.2	contains(const std::pair< size_t, size_t > &rhs)	102
5.46.3.3	contains(const size_t rhs_start, const size_t rhs_end)	102
5.46.3.4	contains(const size_t point)	102
5.46.3.5	distance(const SimpleRegion &rhs)	102
5.46.3.6	getEnd()	102
5.46.3.7	getId()	103
5.46.3.8	getStart()	103
5.46.3.9	midpoint() const	103
5.46.3.10	overlaps(const SimpleRegion &rhs)	103

5.46.3.11 overlaps(const std::pair< size_t, size_t > &rhs)	103
5.46.3.12 overlaps(const size_t rhs_start, const size_t rhs_end)	103
5.46.4 Member Data Documentation	103
5.46.4.1 _end	103
5.46.4.2 _id	103
5.46.4.3 _start	103
5.47 StandardOboGoParser Class Reference	104
5.47.1 Detailed Description	104
5.47.2 Constructor & Destructor Documentation	104
5.47.2.1 StandardOboGoParser()	104
5.47.3 Member Function Documentation	104
5.47.3.1 clone()	104
5.47.3.2 isFileGood(const std::string &filename)	105
5.47.3.3 parseGoFile(std::string filename)	105
5.48 StandardRelationshipPolicy Class Reference	105
5.48.1 Detailed Description	105
5.48.2 Constructor & Destructor Documentation	106
5.48.2.1 StandardRelationshipPolicy()	106
5.48.2.2 StandardRelationshipPolicy(std::vector< GO::Relationship > relationships)	106
5.48.3 Member Function Documentation	106
5.48.3.1 isAllowed(GO::Relationship relationship)	106
5.49 StandardXmlGoParser Class Reference	106
5.49.1 Detailed Description	107
5.49.2 Constructor & Destructor Documentation	107
5.49.2.1 StandardXmlGoParser()	107
5.49.3 Member Function Documentation	107
5.49.3.1 clone()	107
5.49.3.2 isFileGood(const std::string &filename)	107
5.49.3.3 parseGoFile(std::string filename)	108
5.50 TermDepthMap Class Reference	108

5.50.1	Detailed Description	108
5.50.2	Constructor & Destructor Documentation	109
5.50.2.1	TermDepthMap(GoGraph *graph)	109
5.50.2.2	TermDepthMap()	109
5.50.2.3	~TermDepthMap()	109
5.50.3	Member Function Documentation	109
5.50.3.1	getKeys()	109
5.50.3.2	getValue(std::string termId)	109
5.50.3.3	getValues()	109
5.50.3.4	hasTerm(std::string testTerm)	109
5.50.3.5	operator[](std::string termId)	110
5.50.4	Member Data Documentation	110
5.50.4.1	_depths	110
5.50.4.2	_nameToIndex	110
5.51	TermInformationContentMap Class Reference	110
5.51.1	Detailed Description	111
5.51.2	Constructor & Destructor Documentation	111
5.51.2.1	TermInformationContentMap(GoGraph *graph, AnnotationData *annoData)	111
5.51.2.2	TermInformationContentMap()	111
5.51.3	Member Function Documentation	111
5.51.3.1	badIdValue()	111
5.52	TermProbabilityMap Class Reference	112
5.52.1	Detailed Description	113
5.52.2	Constructor & Destructor Documentation	113
5.52.2.1	TermProbabilityMap(GoGraph *graph, AnnotationData *annoData)	113
5.52.2.2	TermProbabilityMap()	113
5.52.2.3	~TermProbabilityMap()	113
5.52.3	Member Function Documentation	114
5.52.3.1	badIdValue()	114
5.52.3.2	getKeys()	114

5.52.3.3	getMinBP()	114
5.52.3.4	getMinCC()	114
5.52.3.5	getMinMF()	114
5.52.3.6	getValue(std::string termId)	114
5.52.3.7	getValues()	114
5.52.3.8	hasTerm(const std::string &testTerm)	115
5.52.3.9	operator[](std::string termId)	115
5.52.4	Member Data Documentation	115
5.52.4.1	_bp_normalization_min_1anno	115
5.52.4.2	_bp_normalization_min_minAnno	115
5.52.4.3	_cc_normalization_min_1anno	115
5.52.4.4	_cc_normalization_min_minAnno	115
5.52.4.5	_isSingleAnnoMin	115
5.52.4.6	_mf_normalization_min_1anno	116
5.52.4.7	_mf_normalization_min_minAnno	116
5.52.4.8	_nameToIndex	116
5.52.4.9	_probabilities	116
5.53	TermSetSimilarityInterface Class Reference	116
5.53.1	Detailed Description	117
5.53.2	Member Function Documentation	117
5.53.2.1	calculateSimilarity(const boost::unordered_set< std::string > &termsA, const boost::unordered_set< std::string > &termsB)=0	117
5.54	TermSimilarityInterface Class Reference	117
5.54.1	Detailed Description	118
5.54.2	Member Function Documentation	118
5.54.2.1	calculateNormalizedTermSimilarity(std::string goTermA, std::string goTermB)=0	118
5.54.2.2	calculateTermSimilarity(std::string goTermA, std::string goTermB)=0	118
5.55	TermSimilarityWriter Class Reference	118
5.55.1	Detailed Description	119
5.55.2	Constructor & Destructor Documentation	119
5.55.2.1	TermSimilarityWriter(GoGraph *goGraph, AnnotationData *annoData)	119
5.55.3	Member Function Documentation	119
5.55.3.1	writeSimilarityMatrix(TermSimilarityInterface *termSim, std::string fileName, long ontology_code)	119
5.55.3.2	writeSimilarityMatrixBP(TermSimilarityInterface *termSim, std::string fileName)	119
5.55.3.3	writeSimilarityMatrixCC(TermSimilarityInterface *termSim, std::string fileName)	120
5.55.3.4	writeSimilarityMatrixMF(TermSimilarityInterface *termSim, std::string fileName)	120
5.56	GoGraph::VertexProps Struct Reference	120
5.56.1	Detailed Description	120
5.56.2	Constructor & Destructor Documentation	120
5.56.2.1	~VertexProps()	120
5.56.3	Member Data Documentation	121
5.56.3.1	ontology	121
5.56.3.2	termId	121

Chapter 1

Namespace Index

1.1 Namespace List

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

Accumulators	
The Accumulators namespace provides min, max, and average accumulators to the broader code base	9
AppUtilities	
The AppUtilities namespace provides utility functions that facilitate application creation and integration	12
EnrichmentTools	
The EnrichmentTools namespace provides simple functions for calculating GO term enrichment	14
GO	
GO namespaces	15
SetUtilities	
The SetUtilities namespace provides useful operations on generic boost::unordered_set . . .	17

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

AnnotationData	30
AnnotationParserFactory	36
AnnotationParserInterface	37
EntrezGene2GoAnnotationParser	49
GoaAnnotationParser	60
GafAnnotationParser	56
MgiAnnotationParser	78
default_dfs_visitor	
TermProbabilityMap::dfs_cumulative_annotations_visitor	45
GoGraph::EdgeProps	48
EvidencePolicyInterface	50
AllowedSetEvidencePolicy	22
ExperimentalEvidencePolicy	53
DisallowedSetEvidencePolicy	46
GoGraph	62
GoParserFactory	71
GoParserInterface	73
AllowedRelationshipOboGoParser	19
AllowedRelationshipXmlGoParser	21
RapidXmlGoParser	94
StandardOboGoParser	104
StandardXmlGoParser	106
NCList	86
NCList::NCNode	89
RelationshipPolicyInterface	95
AllowedSetRelationshipPolicy	24
StandardRelationshipPolicy	105
SharedInformationInterface	99
AncestorMeanSharedInformation	28
CoutoGraSMAadjustedSharedInformation	40
CoutoGraSMSharedInformation	42
ExclusivelyInheritedSharedInformation	51
FrontierSharedInformation	54
MICASharedInformation	79

SimpleRegion	101
GenomicRegion	57
TermDepthMap	108
TermProbabilityMap	112
TermInformationContentMap	110
TermSetSimilarityInterface	116
AllPairsAverageSetSimilarity	26
AllPairsMaxSetSimilarity	27
BestMatchAverageSetSimilarity	39
GentlemanUISimilarity	59
JaccardSetSimilarity	74
TermSimilarityInterface	117
JiangConrathSimilarity	75
LinSimilarity	77
ModularJiangConrath	81
ModularLin	83
ModularResnik	85
PekarStaabSimilarity	91
PrecomputedMatrixTermSimilarity	92
RelevanceSimilarity	96
ResnikSimilarity	97
TermSimilarityWriter	118
GoGraph::VertexProps	120

Chapter 3

Class Index

3.1 Class List

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

AllowedRelationshipOboGoParser	
A class to parse only a specified set of relationships	19
AllowedRelationshipXmlGoParser	
A class to parse only a specified set of relationships	21
AllowedSetEvidencePolicy	
A class to allow only a set of evidence codes for annotations	22
AllowedSetRelationshipPolicy	
A class to allow only a set of relationships	24
AllPairsAverageSetSimilarity	
A class to calculate the average similarity between all pairs of go terms for 2 sets	26
AllPairsMaxSetSimilarity	
A class to calculate the max similarity between all pairs of go terms for 2 sets	27
AncestorMeanSharedInformation	
A class to calculate shared infromation as the average information conent of all common ances- tors	28
AnnotationData	
A class for storing information about genes annotated with go terms	30
AnnotationParserFactory	
A class to return an instance of AnnotationParserInterface at runtime based on an argument .	36
AnnotationParserInterface	
An interface class to define annotation parsers	37
BestMatchAverageSetSimilarity	
A class to calculate the best match average similarity between go terms for 2 sets	39
CoutoGraSMAAdjustedSharedInformation	
A class to calculate shared infromation accross disjoint common ancetors using an adjusted algorithm	40
CoutoGraSMSharedInformation	
A class to calculate shared infromation accross disjoint common ancetors using the exact algo- rithm as written in the paper	42
TermProbabilityMap::dfs_cumulative_annotations_visitor	
Depth first search boost visitor	45
DisallowedSetEvidencePolicy	
A class to allow only a set of evidence codes for annotations	46
GoGraph::EdgeProps	
An Edge Property object	48

EntrezGene2GoAnnotationParser	
A class to parse an Entrez gene2go annotation file	49
EvidencePolicyInterface	
An interface to check evidence codes for GO annotations	50
ExclusivelyInheritedSharedInformation	
A class to calculate shared infomation in linear time after Zhang and Lai	51
ExperimentalEvidencePolicy	
A class to allow experimental evidence codes for annotations	53
FrontierSharedInformation	
A class to calculate shared infomation across disjoint common ancestors in linear time	54
GafAnnotationParser	
A class to parse a GO Annotation File (GAF, Format 2.0)	56
GenomicRegion	57
GentlemanUISimilarity	
A class to calculate Gentleman's UI similarity between go terms for 2 sets	59
GoaAnnotationParser	
A class to parse a Uniprot Gene Ontolog Annotation (GOA) file	60
GoGraph	
This class holds the Gene Ontology directed acyclic graph	62
GoParserFactory	
A class to return an instance of GoParserInterface at runtime based on an argument	71
GoParserInterface	
An interface class to define go graph parsers	73
JaccardSetSimilarity	
A class to calculate jaccard similarity between 2 sets	74
JiangConrathSimilarity	
A class to calculate Jiang Conrath similarity between 2 terms	75
LinSimilarity	
A class to calculate Lin similarity between 2 terms	77
MgiAnnotationParser	
A class to parse an Mouse Genome Informatics go annotation file	78
MICASharedInformation	
A class to calculate shared infomation as the most informative common ancestor (MICA)	79
ModularJiangConrath	
A class to calculate Jiang Conrath similarity between 2 terms	81
ModularLin	
A class to calculate Lin similarity between 2 terms	83
ModularResnik	
A class to calculate resnik similarity between 2 terms using a shared information interface	85
NCList	
A container class for quickly finding intersections of genomic intervals	86
NCList::NCNode	89
PekarStaabSimilarity	
A class to calculate PekarStaab similarity between 2 terms	91
PrecomputedMatrixTermSimilarity	
A class to calculate similarity between go terms for 2 sets using a precomuted term similarity matrix	92
RapidXmlGoParser	
This class parses a go XML file using RapidXML library	94
RelationshipPolicyInterface	
An interface to check relationships between GO terms	95
RelevanceSimilarity	
A class to calculate Relevance similarity between 2 terms	96
ResnikSimilarity	
A class to calculate resnik similarity between 2 terms	97
SharedInformationInterface	
An interface class to define shared information calculations	99
SimpleRegion	101

StandardOboGoParser	
A class to parse only is_a or part_of relationships	104
StandardRelationshipPolicy	
A class to allow only a set of relationships	105
StandardXmlGoParser	
A class to parse only is_a or part_of relationships	106
TermDepthMap	
A class to calculate the depth of a GO term in the ontology	108
TermInformationContentMap	
A class to calculate the information content of a GO term	110
TermProbabilityMap	
A class to calculate the probability of a GO term	112
TermSetSimilarityInterface	
An interface class for comparing semantic similarity of sets of GO terms	116
TermSimilarityInterface	
An interface class for comparing semantic similarity of GO terms	117
TermSimilarityWriter	
A class write a term similarity matrix to file. Companion to PrecomputedMatrixTermSimilarity	118
GoGraph::VertexProps	
A Vertex Property object	120

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 overloaded helper helper function to extract the min.*
- double `extractMax` (const `SimpleAccumulator` &acc)
- An overloaded helper helper function to extract the max.*
- double `extractMean` (const `SimpleAccumulator` &acc)
- An overloaded 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 extractors 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 overloaded 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 overloaded 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 > ¶mMap)
A method for parsing a parameter file.
- bool [checkParams](#) (const std::map< std::string, std::string > ¶mMap, const std::vector< std::string > ¶ms, 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 applications using this tool kit. The main use for this namespace is to construct and check the validity of parameters used as input to [GO](#) applications and to provide certain container conversion (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 into 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 parameters.

This method provides an easy method for inputting a list of mandatory params to be checked with `checkParams`

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 returns 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 calculating [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 hypergeometric 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 calculating [GO](#) term enrichment.

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

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 hypergeometric 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 hypergeometric 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 }

Evidence 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

Evidence Code enum type.

This enum defines a type for evidence codes. Defined at <http://www.geneontology.org/GO.%E2%9C%93evidence.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 operations on generic boost::unordered_set.

4.5.1 Detailed Description

The [SetUtilities](#) namespace provides useful operations on generic boost::unordered_set.

[SetUtilities](#) provides useful operations on boost::unordered_set. set_intersection is optimized to traverse the smaller set and test existence in the larger set.

Chapter 5

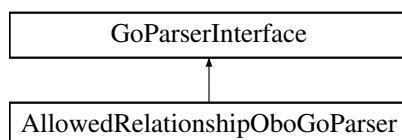
Class Documentation

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 is 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

creates 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:

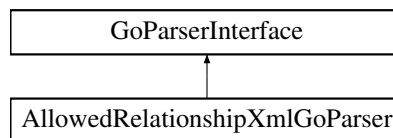
- ggtk/AllowedRelationshipOboGoParser.hpp

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 is 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
creates 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:

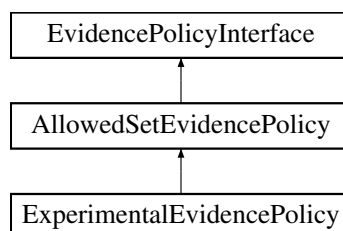
- `ggtk/AllowedRelationshipXmlGoParser.hpp`

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 evidence 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 restrict 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.

Creates 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:

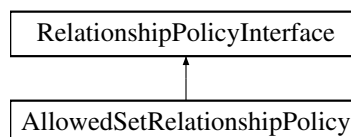
- `ggtk/AllowedSetEvidencePolicy.hpp`

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 relatioship 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 restrict 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.

Creates 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 relationship 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:

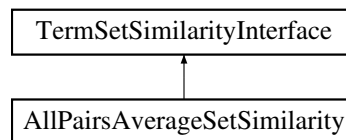
- `ggtk/AllowedSetRelationshipPolicy.hpp`

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 between 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 member.

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:

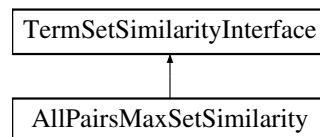
- `ggtk/AllPairsAverageSetSimilarity.hpp`

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:

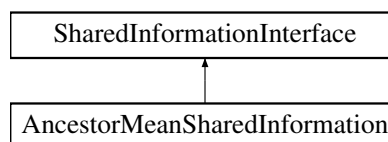
- [ggtk/AllPairsMaxSetSimilarity.hpp](#)

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 information as the average information content of all common ancestors.

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

This shared information method is used as 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 information 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 provides a mechanism for returning a term's information 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 destructor
- 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 ontology.
- 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 destructor

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 database

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 database

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 ontology.

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:

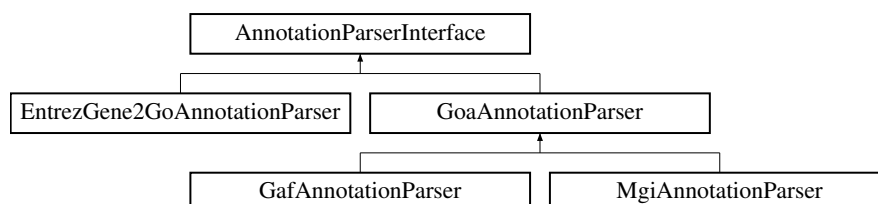
- ggk/AnnotationParserFactory.hpp

5.10 AnnotationParserInterface Class Reference

An interface class to define annotation parsers.

```
#include <ggk/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:

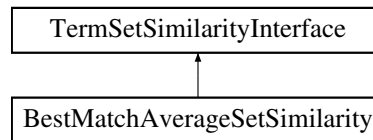
- ggk/AnnotationParserInterface.hpp

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 between 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 member.

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:

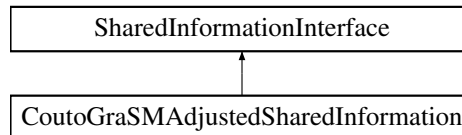
- ggtk/BestMatchAverageSetSimilarity.hpp

5.12 CoutoGraSMAadjustedSharedInformation Class Reference

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

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

Inheritance diagram for CoutoGraSMAadjustedSharedInformation:



Public Member Functions

- [CoutoGraSMAadjustedSharedInformation](#) ([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 information 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 information accross disjoint common ancetors using an adjusted algorithm.

This class calculates shared information accross disjoint common ancetors. This is a modifaicon of the original algorithm provided by Couto. The adjustment changes the constrain 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 `CoutoGraSMAadjustedSharedInformation::CoutoGraSMAadjustedSharedInformation (GoGraph * goGraph, TermInformationContentMap & icMap)` `[inline]`

Constructor.

Creates the [CoutoGraSMAadjustedSharedInformation](#) class

5.12.3 Member Function Documentation

5.12.3.1 `boost::unordered_set<std::string> CoutoGraSMAadjustedSharedInformation::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 CoutoGraSMAadjustedSharedInformation::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 CoutoGraSMAadjustedSharedInformation::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 CoutoGraSMAadjustedSharedInformation::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 CoutoGraSMAadjustedSharedInformation::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 CoutoGraSMAadjustedSharedInformation::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 CoutoGraSMAadjustedSharedInformation::sharedInformation (const std::string & termA, const std::string & termB) [inline], [virtual]`

Shared infomation between two conecepts.

A method for calculating the shared infromation between two concepts.

Implements [SharedInformationInterface](#).

5.12.3.8 `double CoutoGraSMAadjustedSharedInformation::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:

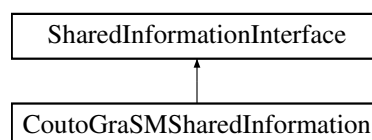
- `ggtk/CoutoGraSMAadjustedSharedInformation.hpp`

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 information across disjoint common ancestors using the exact algorithm as written in the paper.

This class calculates shared information across disjoint common ancestors.

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 as a substitute 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 CoutoGraSMSSharedInformation::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 CoutoGraSMSSharedInformation::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 CoutoGraSMSSharedInformation::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 CoutoGraSMSSharedInformation::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 CoutoGraSMSSharedInformation::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 CoutoGraSMSSharedInformation::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 CoutoGraSMSSharedInformation::sharedInformation (const std::string & term) [inline],[virtual]`

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

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

Implements [SharedInformationInterface](#).

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

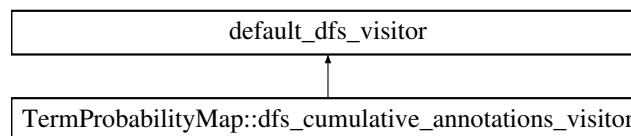
- ggtk/CoutoGraSMSSharedInformation.hpp

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 TermProbabilityMap 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 annotations 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 cumulative 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:

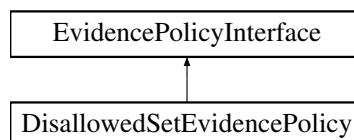
- ggk/TermProbabilityMap.hpp

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 evidence 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 restrict 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.

Creates 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 evidence 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]`

Destructor

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:

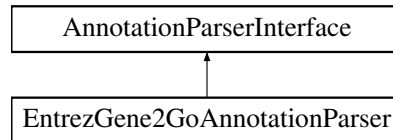
- `ggtk/GoGraph.hpp`

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 annotations 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:

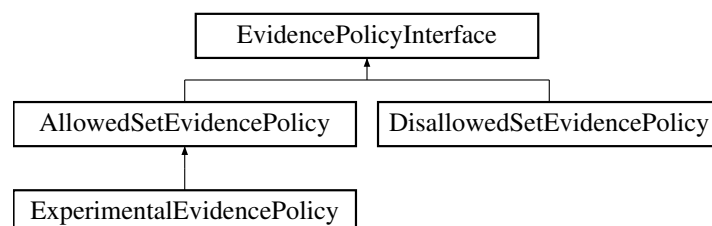
- `ggtk/EntrezGene2GoAnnotationParser.hpp`

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 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 implement an `isAllowed` method to enforce the evidence policy.

Implemented in [AllowedSetEvidencePolicy](#), and [DisallowedSetEvidencePolicy](#).

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

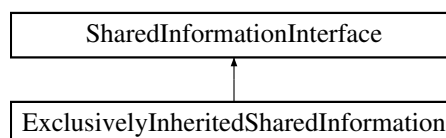
- `ggtk/EvidencePolicyInterface.hpp`

5.19 ExclusivelyInheritedSharedInformation Class Reference

A class to calculate shared information 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 information 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 information. *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 [CoutoGraSMSSharedInformation](#) 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 prvides a mechanism for returing a term's infromation content.

Implements [SharedInformationInterface](#).

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

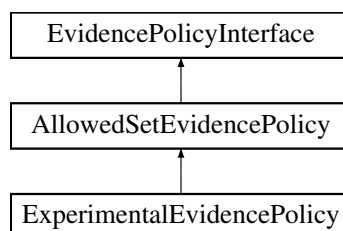
- `ggtk/ExclusivelyInheritedSharedInformation.hpp`

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 [AllowedSetEvidencePolicy](#) 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:

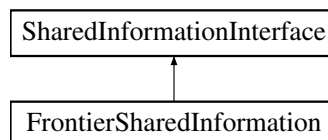
- ggtk/ExperimentalEvidencePolicy.hpp

5.21 FrontierSharedInformation Class Reference

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

This class calculates shared information 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 prvides a mechanism for returing a term's infromation content.

Implements [SharedInformationInterface](#).

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

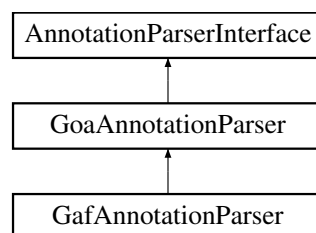
- `ggtk/FrontierSharedInformation.hpp`

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

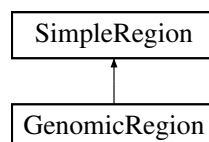
The documentation for this class was generated from the following file:

- 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:

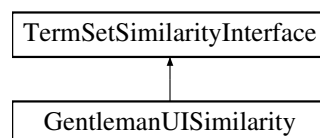
- ggtk/SimpleRegion.hpp

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 member.

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:

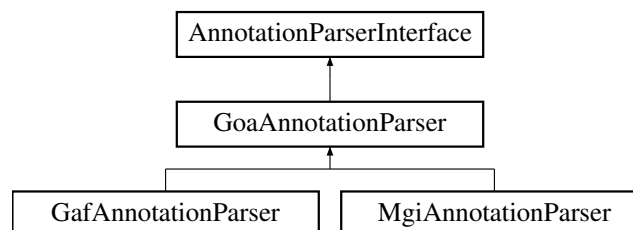
- ggtk/GentlemanSimUISetSimilarity.hpp

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 and 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 input 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:

- ggk/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 descendant 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::InEdgelterator

An in edge iterator.

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

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

An out edge iterator.

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

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 descendant terms for a given term.

This method takes a term and returns a list of descendant 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 contains 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 ontology 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 ontology 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 ontology 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 relationship 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:

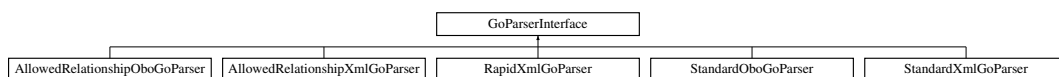
- ggtk/GoParserFactory.hpp

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:

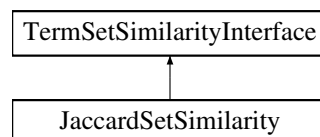
- `ggtk/GoParserInterface.hpp`

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:

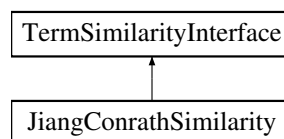
- ggtk/JaccardSetSimilarity.hpp

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.

$\text{distance} = \text{IC}(\text{termA}) + \text{IC}(\text{termB}) - 2 * \text{IC}(\text{MICA})$ $\text{maxDistance} = 2 * \text{IC}(\text{single_annotaiothn})$ $\text{similarity} = 1 - \text{distance} / \text{maxDistance}$ (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:

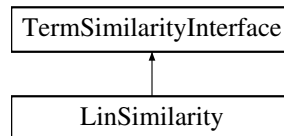
- `ggtk/JiangConrathSimilarity.hpp`

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 International Conference on Machine Learning. San Francisco, 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:

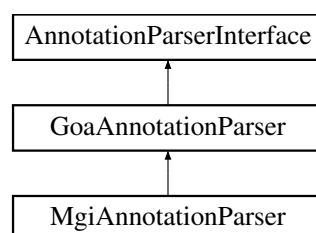
- `ggtk/LinSimilarity.hpp`

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:

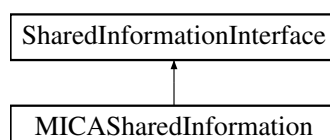
- ggtk/MgiAnnotationParser.hpp

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 et 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 information 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 provides a mechanism for returning a term's information content.

Implements [SharedInformationInterface](#).

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

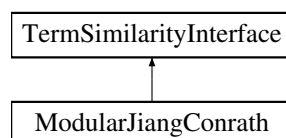
- ggtk/MICASharedInformation.hpp

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](#) *newSharedInformationCalculator)
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.

$$\text{distance} = \text{IC}(\text{termA}) + \text{IC}(\text{termB}) - 2 * \text{IC}(\text{MICA})$$

$$\text{maxDistance} = 2 * \text{IC}(\text{single annotaiotn})$$

$$\text{similarity} = 1 - \text{distance} / \text{maxDistance} \text{ (see Lord et al.)}$$

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:

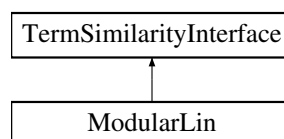
- `ggtk/ModularJiangConrath.hpp`

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 International Conference on Machine Learning. San Francisco, 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 * newSharedInformationCalculator)` `[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:

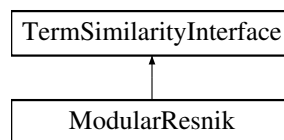
- `ggtk/ModularLin.hpp`

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](#) *newSharedInformationCalculator)
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 *
newSharedInformationCalculator)` `[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](#) > ®ions)
- 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 prerepresents 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 intervals 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

Containment |-----| |-----|

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:

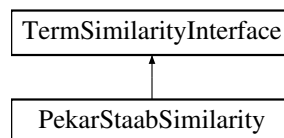
- ggtk/NCList.hpp

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) $\text{GraphDist}(\text{LCA}, \text{root}) / (\text{GraphDist}(\text{a}, \text{LCA}) + \text{GraphDist}(\text{b}, \text{LCA}) + \text{GraphDist}(\text{LCA}, \text{root}))$

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:

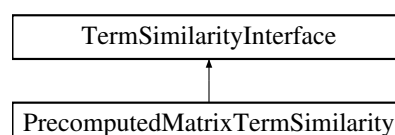
- ggk/PekarStaabSimilarity.hpp

5.40 PrecomputedMatrixTermSimilarity Class Reference

A class to calculate similarity between go terms for 2 sets using a precomputed 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 precomputed 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:

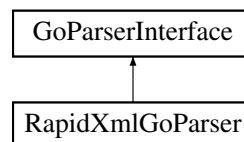
- `ggtk/PrecomputedMatrixTermSimilarity.hpp`

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.

creates 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:

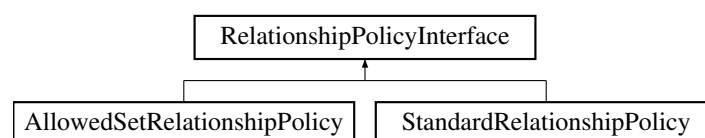
- ggtk/RapidXmlGoParser.hpp

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:

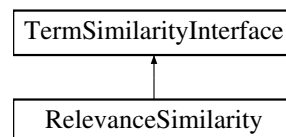
- `ggtk/RelationshipPolicyInterface.hpp`

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:

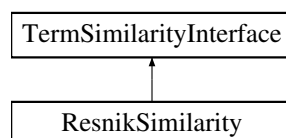
- `ggtk/RelevanceSimilarity.hpp`

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:

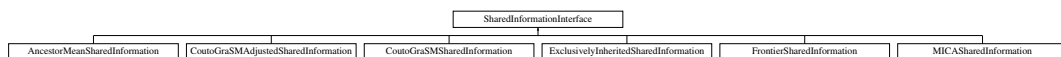
- `ggtk/ResnikSimilarity.hpp`

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 provides a mechanism for returning a term's information content.

Implemented in [CoutoGraSMSSharedInformation](#), [CoutoGraSMAAdjustedSharedInformation](#), [FrontierSharedInformation](#), [ExclusivelyInheritedSharedInformation](#), [MICASharedInformation](#), and [AncestorMeanSharedInformation](#).

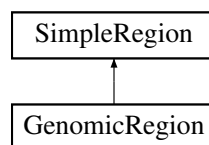
The documentation for this class was generated from the following file:

- 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:

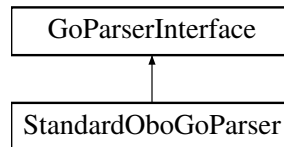
- ggtk/SimpleRegion.hpp

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

creates 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:

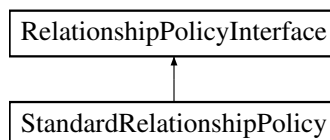
- ggtk/StandardOboGoParser.hpp

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 relationship 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 restrict 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.

Creates 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 relationship 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:

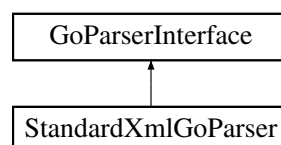
- `ggtk/StandardRelationshipPolicy.hpp`

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

creates 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 probabilities 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 constructor 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:

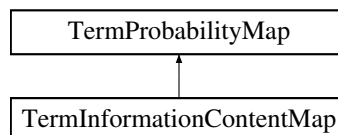
- ggtk/TermDepthMap.hpp

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 constructor is allowed to ensure these objects are created with valid parameters. This constructor relies on the [TermProbabilityMap](#).

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:

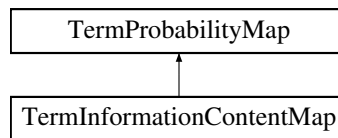
- `ggtk/TermInformationContentMap.hpp`

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 probabilities 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 certainty. 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 probability for the bp ontology

5.52.3.4 `double TermProbabilityMap::getMinCC () [inline]`

Get the specific minimum probability for CC.

This function returns the minimum probability for the cc ontology

5.52.3.5 `double TermProbabilityMap::getMinMF () [inline]`

Get the specific minimum probability for MF.

This function returns the minimum probability 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 probabilities 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 divided 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 divided 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 divided 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 divided 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:

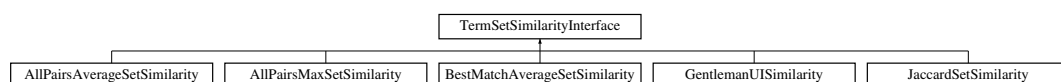
- `ggtk/TermProbabilityMap.hpp`

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:

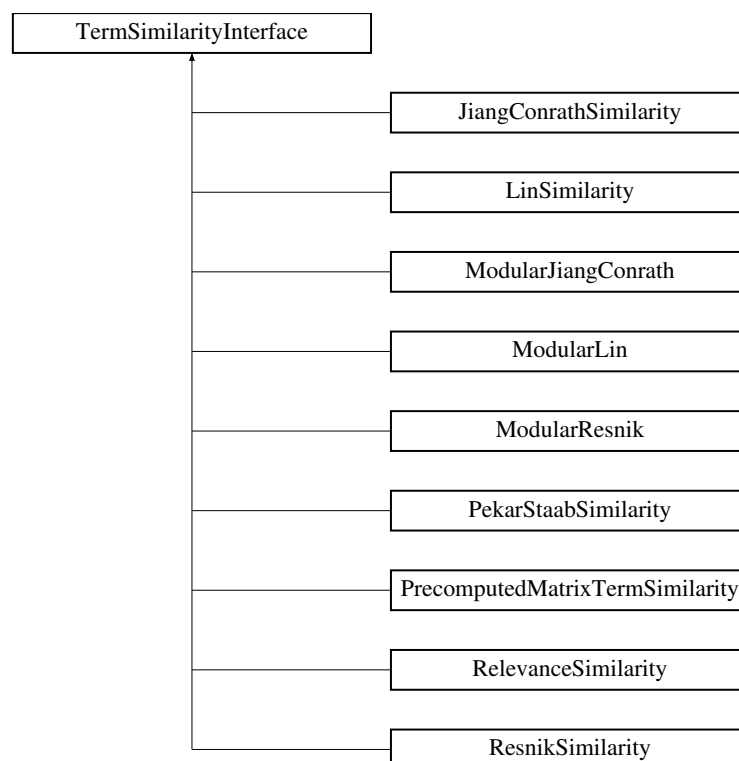
- `ggtk/TermSetSimilarityInterface.hpp`

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 normalized. 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](#), [ModularJiangConrath](#), [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 $O(N^2) * O(\text{Term Pair Calculation Cost})$. $O(\text{Term Pair Calculation Cost})$ 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.

Calcuates 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.

Calculates 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.

Calculates 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]`

Destructor

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

- [_bp_normalization_min_1anno](#)
 - [TermProbabilityMap, 115](#)
 - [_bp_normalization_min_minAnno](#)
 - [TermProbabilityMap, 115](#)
 - [_cc_normalization_min_1anno](#)
 - [TermProbabilityMap, 115](#)
 - [_cc_normalization_min_minAnno](#)
 - [TermProbabilityMap, 115](#)
 - [_chrom](#)
 - [GenomicRegion, 59](#)
 - [_containments](#)
 - [NCList::NCNode, 90](#)
 - [_depths](#)
 - [TermDepthMap, 110](#)
 - [_desc](#)
 - [GenomicRegion, 59](#)
 - [_end](#)
 - [SimpleRegion, 103](#)
 - [_geneToGos](#)
 - [AnnotationData, 35](#)
 - [_geneToGosEvidence](#)
 - [AnnotationData, 35](#)
 - [_genes](#)
 - [AnnotationData, 35](#)
 - [_goTerms](#)
 - [AnnotationData, 35](#)
 - [_goToGenes](#)
 - [AnnotationData, 35](#)
 - [_goToGenesEvidence](#)
 - [AnnotationData, 35](#)
 - [_id](#)
 - [SimpleRegion, 103](#)
 - [_isSingleAnnoMin](#)
 - [TermProbabilityMap, 115](#)
 - [_items](#)
 - [NCList, 89](#)
 - [_mf_normalization_min_1anno](#)
 - [TermProbabilityMap, 115](#)
 - [_mf_normalization_min_minAnno](#)
 - [TermProbabilityMap, 116](#)
 - [_name](#)
 - [GenomicRegion, 59](#)
 - [_nameToIndex](#)
 - [TermDepthMap, 110](#)
 - [TermProbabilityMap, 116](#)
 - [_probabilities](#)
 - [TermProbabilityMap, 116](#)
 - [_region](#)
 - [NCList::NCNode, 90](#)
- [_start](#)
 - [SimpleRegion, 103](#)
 - [_stringToGene](#)
 - [AnnotationData, 35](#)
 - [_stringToGo](#)
 - [AnnotationData, 36](#)
 - [~AnnotationData](#)
 - [AnnotationData, 32](#)
 - [~AnnotationParserFactory](#)
 - [AnnotationParserFactory, 37](#)
 - [~EdgeProps](#)
 - [GoGraph::EdgeProps, 48](#)
 - [~GoGraph](#)
 - [GoGraph, 66](#)
 - [~GoParserFactory](#)
 - [GoParserFactory, 72](#)
 - [~TermDepthMap](#)
 - [TermDepthMap, 109](#)
 - [~TermProbabilityMap](#)
 - [TermProbabilityMap, 113](#)
 - [~VertexProps](#)
 - [GoGraph::VertexProps, 120](#)
- [Accumulators, 9](#)
 - [CovarianceAccumulator, 10](#)
 - [extractCovariance, 11](#)
 - [extractMax, 11](#)
 - [extractMean, 11](#)
 - [extractMin, 11, 12](#)
 - [extractSD, 12](#)
 - [extractVariance, 12](#)
 - [MaxAccumulator, 10](#)
 - [MeanAccumulator, 10](#)
 - [MinAccumulator, 10](#)
 - [SimpleAccumulator, 10](#)
 - [VarianceAccumulator, 11](#)
- [addAssociation](#)
 - [AnnotationData, 32](#)
- [addEvidence](#)
 - [AllowedSetEvidencePolicy, 23](#)
 - [DisallowedSetEvidencePolicy, 47](#)
- [addGenesForGoTerm](#)
 - [AnnotationData, 32](#)
- [addParser](#)
 - [AnnotationParserFactory, 37](#)
 - [GoParserFactory, 72](#)
- [addRelationship](#)
 - [AllowedSetRelationshipPolicy, 25](#)
- [AllPairsAverageSetSimilarity, 26](#)
 - [AllPairsAverageSetSimilarity, 26](#)

- calculateSimilarity, 27
- AllPairsMaxSetSimilarity, 27
 - AllPairsMaxSetSimilarity, 28
 - calculateSimilarity, 28
- AllowedRelationshipOboGoParser, 19
 - AllowedRelationshipOboGoParser, 20
 - clone, 20
 - isFileGood, 20
 - parseGoFile, 20
 - setPolicy, 20
 - splitWith, 20
- AllowedRelationshipXmlGoParser, 21
 - AllowedRelationshipXmlGoParser, 21
 - clone, 22
 - isFileGood, 22
 - parseGoFile, 22
 - setPolicy, 22
- AllowedSetEvidencePolicy, 22
 - addEvidence, 23
 - AllowedSetEvidencePolicy, 23
 - isAllowed, 24
 - isEmpty, 24
- AllowedSetRelationshipPolicy, 24
 - addRelationship, 25
 - AllowedSetRelationshipPolicy, 25
 - isAllowed, 25
 - isEmpty, 25
- AncestorMeanSharedInformation, 28
 - AncestorMeanSharedInformation, 29
 - hasTerm, 29
 - isSameOntology, 29
 - maxInformationContent, 29
 - sharedInformation, 29, 30
- AnnotationData, 30
 - _geneToGos, 35
 - _geneToGosEvidence, 35
 - _genes, 35
 - _goTerms, 35
 - _goToGenes, 35
 - _goToGenesEvidence, 35
 - _stringToGene, 35
 - _stringToGo, 36
 - ~AnnotationData, 32
 - addAssociation, 32
 - addGenesForGoTerm, 32
 - AnnotationData, 32
 - getAllGenes, 32
 - getAllGoTerms, 32
 - getGenesEvidenceForGoTerm, 32
 - getGenesForGoTerm, 33
 - getGoTermsEvidenceForGene, 33
 - getGoTermsForGene, 33
 - getGoTermsForGeneBP, 33
 - getGoTermsForGeneByOntology, 33
 - getGoTermsForGeneCC, 33
 - getGoTermsForGeneMF, 33
 - getNumAnnotationsForGene, 34
 - getNumAnnotationsForGoTerm, 34
 - getNumGenes, 34
 - getNumGoTerms, 34
 - getOntologyTerms, 34
 - hasGene, 34
 - hasGoTerm, 34
- AnnotationParserFactory, 36
 - ~AnnotationParserFactory, 37
 - addParser, 37
 - AnnotationParserFactory, 37
 - getParser, 37
- AnnotationParserInterface, 37
 - clone, 38
 - isFileGood, 38
 - parseAnnotationFile, 38
- AppUtilities, 12
 - checkParams, 13
 - paramList, 13
 - parseParamFile, 13
 - parseSimpleFile, 13
 - setToVec, 13
 - splitTake, 13
 - vecToSet, 14
- badIdValue
 - TermInformationContentMap, 111
 - TermProbabilityMap, 114
- BestMatchAverageSetSimilarity, 39
 - BestMatchAverageSetSimilarity, 39
 - calculateSimilarity, 39
- calculateNormalizedTermSimilarity
 - JiangConrathSimilarity, 76
 - LinSimilarity, 78
 - ModularJiangConrath, 82
 - ModularLin, 84
 - ModularResnik, 86
 - PekarStaabSimilarity, 92
 - PrecomputedMatrixTermSimilarity, 93
 - RelevanceSimilarity, 97
 - ResnikSimilarity, 98
 - TermSimilarityInterface, 118
- calculateSimilarity
 - AllPairsAverageSetSimilarity, 27
 - AllPairsMaxSetSimilarity, 28
 - BestMatchAverageSetSimilarity, 39
 - GentlemanUISimilarity, 60
 - JaccardSetSimilarity, 75
 - TermSetSimilarityInterface, 117
- calculateTermSimilarity
 - JiangConrathSimilarity, 76
 - LinSimilarity, 78
 - ModularJiangConrath, 82
 - ModularLin, 84
 - ModularResnik, 86
 - PekarStaabSimilarity, 92
 - PrecomputedMatrixTermSimilarity, 93
 - RelevanceSimilarity, 97
 - ResnikSimilarity, 98
 - TermSimilarityInterface, 118

- checkParams
 - AppUtilities, 13
- clone
 - AllowedRelationshipOboGoParser, 20
 - AllowedRelationshipXmlGoParser, 22
 - AnnotationParserInterface, 38
 - EntrezGene2GoAnnotationParser, 50
 - GoParserInterface, 73
 - GoaAnnotationParser, 61
 - RapidXmlGoParser, 94
 - StandardOboGoParser, 104
 - StandardXmlGoParser, 107
- contains
 - NCList::NCNode, 90
 - SimpleRegion, 102
- CoutoGraSMAAdjustedSharedInformation, 40
 - CoutoGraSMAAdjustedSharedInformation, 41
 - getCommonDisjointAncestors, 41
 - getNumPaths, 41
 - hasTerm, 41
 - isDisjoint, 41
 - isSameOntology, 41
 - maxInformationContent, 41
 - sharedInformation, 42
- CoutoGraSMSharedInformation, 42
 - CoutoGraSMSharedInformation, 43
 - getCommonDisjointAncestors, 43
 - getNumPaths, 43
 - hasTerm, 44
 - isDisjoint, 44
 - isSameOntology, 44
 - maxInformationContent, 44
 - sharedInformation, 44
- CovarianceAccumulator
 - Accumulators, 10
- DisallowedSetEvidencePolicy, 46
 - addEvidence, 47
 - DisallowedSetEvidencePolicy, 47
 - isAllowed, 47
 - isEmpty, 47
- distance
 - SimpleRegion, 102
- EdgeIndexMap
 - GoGraph, 64
- enrichmentSignificance
 - EnrichmentTools, 14
- EnrichmentTools, 14
 - enrichmentSignificance, 14
 - getDescendantGenes, 14
 - oneSidedRawPvalue_hyper, 15
- EntrezGene2GoAnnotationParser, 49
 - clone, 50
 - EntrezGene2GoAnnotationParser, 49
 - isFileGood, 50
 - parseAnnotationFile, 50
- EvidenceCode
 - GO, 16
- EvidencePolicyInterface, 50
 - isAllowed, 51
- evidenceStringToCode
 - GO, 17
- evidenceToString
 - GO, 17
- ExclusivelyInheritedSharedInformation, 51
 - ExclusivelyInheritedSharedInformation, 52
 - getCommonDisjointAncestors, 52
 - hasTerm, 52
 - isSameOntology, 52
 - maxInformationContent, 53
 - sharedInformation, 53
- ExperimentalEvidencePolicy, 53
 - ExperimentalEvidencePolicy, 54
- extractCovariance
 - Accumulators, 11
- extractMax
 - Accumulators, 11
- extractMean
 - Accumulators, 11
- extractMin
 - Accumulators, 11, 12
- extractSD
 - Accumulators, 12
- extractVariance
 - Accumulators, 12
- filterSetForBP
 - GoGraph, 66
- filterSetForCC
 - GoGraph, 66
- filterSetForMF
 - GoGraph, 66
- filterSetForOntology
 - GoGraph, 67
- finish_vertex
 - TermProbabilityMap::dfs_cumulative_annotations←_visitor, 46
- FrontierSharedInformation, 54
 - FrontierSharedInformation, 55
 - getCommonDisjointAncestors, 55
 - hasTerm, 55
 - isSameOntology, 55
 - maxInformationContent, 55
 - sharedInformation, 55, 56
- GafAnnotationParser, 56
 - GafAnnotationParser, 57
- GenomicRegion, 57
 - _chrom, 59
 - _desc, 59
 - _name, 59
 - GenomicRegion, 58
 - getChrom, 58
 - getDesc, 58
 - getName, 58
- GentlemanUISimilarity, 59
 - calculateSimilarity, 60

- GentlemanUISimilarity, 60
- getAllGenes
 - AnnotationData, 32
- getAllGoTerms
 - AnnotationData, 32
- getAllTerms
 - GoGraph, 67
- getAllTermsBP
 - GoGraph, 67
- getAllTermsCC
 - GoGraph, 67
- getAllTermsMF
 - GoGraph, 67
- getAncestorTerms
 - GoGraph, 67
- getChildTerms
 - GoGraph, 68
- getChrom
 - GenomicRegion, 58
- getCommonDisjointAncestors
 - CoutoGraSMAadjustedSharedInformation, 41
 - CoutoGraSMSharedInformation, 43
 - ExclusivelyInheritedSharedInformation, 52
 - FrontierSharedInformation, 55
- getDesc
 - GenomicRegion, 58
- getDescendantGenes
 - EnrichmentTools, 14
- getDescendantTerms
 - GoGraph, 68
- getEnd
 - SimpleRegion, 102
- getFeatures
 - NCList, 87
- getFeaturesAt
 - NCList, 87
- getFeaturesInRange
 - NCList, 87, 88
- getGenesEvidenceForGoTerm
 - AnnotationData, 32
- getGenesForGoTerm
 - AnnotationData, 33
- getGoTermsEvidenceForGene
 - AnnotationData, 33
- getGoTermsForGene
 - AnnotationData, 33
- getGoTermsForGeneBP
 - AnnotationData, 33
- getGoTermsForGeneByOntology
 - AnnotationData, 33
- getGoTermsForGeneCC
 - AnnotationData, 33
- getGoTermsForGeneMF
 - AnnotationData, 33
- getGraph
 - GoGraph, 68
- getId
 - SimpleRegion, 102
- getInducedSubgraph
 - GoGraph, 68
- getInducedSubgraph2
 - GoGraph, 68
- getKeys
 - TermDepthMap, 109
 - TermProbabilityMap, 114
- getLCA
 - PekarStaabSimilarity, 92
- getMICA
 - JiangConrathSimilarity, 76
 - LinSimilarity, 78
 - RelevanceSimilarity, 97
 - ResnikSimilarity, 99
- getMinBP
 - TermProbabilityMap, 114
- getMinCC
 - TermProbabilityMap, 114
- getMinMF
 - TermProbabilityMap, 114
- getName
 - GenomicRegion, 58
- getNodes
 - NCList, 88
- getNumAnnotationsForGene
 - AnnotationData, 34
- getNumAnnotationsForGoTerm
 - AnnotationData, 34
- getNumComponents
 - GoGraph, 68
- getNumEdges
 - GoGraph, 68
- getNumGenes
 - AnnotationData, 34
- getNumGoTerms
 - AnnotationData, 34
- getNumPaths
 - CoutoGraSMAadjustedSharedInformation, 41
 - CoutoGraSMSharedInformation, 43
- getNumVertices
 - GoGraph, 69
- getOntologyTerms
 - AnnotationData, 34
 - GoGraph, 69
- getOverlapIndex
 - NCList, 88
- getParentTerms
 - GoGraph, 69
- getParser
 - AnnotationParserFactory, 37
 - GoParserFactory, 72
- getRegion
 - NCList::NCNode, 90
- getRegionAt
 - NCList, 88
- getRoot
 - GoGraph, 69
- getStart

- SimpleRegion, 103
- getTermDescription
 - GoGraph, 69
- getTermDescriptionByIndex
 - GoGraph, 69
- getTermIndex
 - GoGraph, 69
- getTermName
 - GoGraph, 69
- getTermNameByIndex
 - GoGraph, 70
- getTermOntology
 - GoGraph, 70
- getTermOntologyByIndex
 - GoGraph, 70
- getTermOntologyByVertex
 - GoGraph, 70
- getTermStringIdByIndex
 - GoGraph, 70
- getValue
 - TermDepthMap, 109
 - TermProbabilityMap, 114
- getValues
 - TermDepthMap, 109
 - TermProbabilityMap, 114
- getVertexByIndex
 - GoGraph, 70
- getVertexByName
 - GoGraph, 70
- getVertexIndex
 - GoGraph, 70
- GO, 15
 - EvidenceCode, 16
 - evidenceStringToCode, 17
 - evidenceToString, 17
 - Onto, 16
 - ontologyStringToCode, 17
 - ontologyToString, 17
 - Relationship, 16
 - relationshipStringToCode, 17
 - relationshipToString, 17
- GoEdge
 - GoGraph, 64
- GoGraph, 62
 - ~GoGraph, 66
 - EdgeIndexMap, 64
 - filterSetForBP, 66
 - filterSetForCC, 66
 - filterSetForMF, 66
 - filterSetForOntology, 67
 - getAllTerms, 67
 - getAllTermsBP, 67
 - getAllTermsCC, 67
 - getAllTermsMF, 67
 - getAncestorTerms, 67
 - getChildTerms, 68
 - getDescendantTerms, 68
 - getGraph, 68
 - getInducedSubgraph, 68
 - getInducedSubgraph2, 68
 - getNumComponents, 68
 - getNumEdges, 68
 - getNumVertices, 69
 - getOntologyTerms, 69
 - getParentTerms, 69
 - getRoot, 69
 - getTermDescription, 69
 - getTermDescriptionByIndex, 69
 - getTermIndex, 69
 - getTermName, 69
 - getTermNameByIndex, 70
 - getTermOntology, 70
 - getTermOntologyByIndex, 70
 - getTermOntologyByVertex, 70
 - getTermStringIdByIndex, 70
 - getVertexByIndex, 70
 - getVertexByName, 70
 - getVertexIndex, 70
 - GoEdge, 64
 - GoVertex, 64
 - GoVertexIterator, 65
 - Graph, 65
 - Graph_t, 65
 - hasTerm, 71
 - InEdgeIterator, 65
 - initMaps, 71
 - insertRelationship, 71
 - insertTerm, 71
 - OutEdgeIterator, 65
 - VertexIndexMap, 65
- GoGraph::EdgeProps, 48
 - ~EdgeProps, 48
 - relType, 48
- GoGraph::VertexProps, 120
 - ~VertexProps, 120
 - ontology, 121
 - termId, 121
- GoParserFactory, 71
 - ~GoParserFactory, 72
 - addParser, 72
 - getParser, 72
 - GoParserFactory, 72
- GoParserInterface, 73
 - clone, 73
 - isFileGood, 73
 - parseGoFile, 74
- GoVertex
 - GoGraph, 64
- GoVertexIterator
 - GoGraph, 65
- GoaAnnotationParser, 60
 - clone, 61
 - GoaAnnotationParser, 61
 - isFileGood, 61
 - parseAnnotationFile, 61
- Graph

- GoGraph, 65
- Graph_t
 - GoGraph, 65
- hasGene
 - AnnotationData, 34
- hasGoTerm
 - AnnotationData, 34
- hasTerm
 - AncestorMeanSharedInformation, 29
 - CoutoGraSMAAdjustedSharedInformation, 41
 - CoutoGraSMSharedInformation, 44
 - ExclusivelyInheritedSharedInformation, 52
 - FrontierSharedInformation, 55
 - GoGraph, 71
 - MICASharedInformation, 80
 - SharedInformationInterface, 100
 - TermDepthMap, 109
 - TermProbabilityMap, 114
- InEdgeIterator
 - GoGraph, 65
- initMaps
 - GoGraph, 71
- insertRelationship
 - GoGraph, 71
- insertTerm
 - GoGraph, 71
- isAllowed
 - AllowedSetEvidencePolicy, 24
 - AllowedSetRelationshipPolicy, 25
 - DisallowedSetEvidencePolicy, 47
 - EvidencePolicyInterface, 51
 - RelationshipPolicyInterface, 96
 - StandardRelationshipPolicy, 106
- isDisjoint
 - CoutoGraSMAAdjustedSharedInformation, 41
 - CoutoGraSMSharedInformation, 44
- isEmpty
 - AllowedSetEvidencePolicy, 24
 - AllowedSetRelationshipPolicy, 25
 - DisallowedSetEvidencePolicy, 47
- isFileGood
 - AllowedRelationshipOboGoParser, 20
 - AllowedRelationshipXmlGoParser, 22
 - AnnotationParserInterface, 38
 - EntrezGene2GoAnnotationParser, 50
 - GoParserInterface, 73
 - GoaAnnotationParser, 61
 - RapidXmlGoParser, 94
 - StandardOboGoParser, 104
 - StandardXmlGoParser, 107
- isSameOntology
 - AncestorMeanSharedInformation, 29
 - CoutoGraSMAAdjustedSharedInformation, 41
 - CoutoGraSMSharedInformation, 44
 - ExclusivelyInheritedSharedInformation, 52
 - FrontierSharedInformation, 55
 - MICASharedInformation, 80
- SharedInformationInterface, 100
- JaccardSetSimilarity, 74
 - calculateSimilarity, 75
 - JaccardSetSimilarity, 75
- JiangConrathSimilarity, 75
 - calculateNormalizedTermSimilarity, 76
 - calculateTermSimilarity, 76
 - getMICA, 76
 - JiangConrathSimilarity, 76
- LinSimilarity, 77
 - calculateNormalizedTermSimilarity, 78
 - calculateTermSimilarity, 78
 - getMICA, 78
 - LinSimilarity, 77
- MICASharedInformation, 79
 - hasTerm, 80
 - isSameOntology, 80
 - MICASharedInformation, 80
 - maxInformationContent, 81
 - sharedInformation, 81
- MaxAccumulator
 - Accumulators, 10
- maxInformationContent
 - AncestorMeanSharedInformation, 29
 - CoutoGraSMAAdjustedSharedInformation, 41
 - CoutoGraSMSharedInformation, 44
 - ExclusivelyInheritedSharedInformation, 53
 - FrontierSharedInformation, 55
 - MICASharedInformation, 81
 - SharedInformationInterface, 100
- MeanAccumulator
 - Accumulators, 10
- MgiAnnotationParser, 78
 - MgiAnnotationParser, 79
- midpoint
 - SimpleRegion, 103
- MinAccumulator
 - Accumulators, 10
- ModularJiangConrath, 81
 - calculateNormalizedTermSimilarity, 82
 - calculateTermSimilarity, 82
 - ModularJiangConrath, 82
 - setSharedInformationCalculator, 83
- ModularLin, 83
 - calculateNormalizedTermSimilarity, 84
 - calculateTermSimilarity, 84
 - ModularLin, 84
 - setSharedInformationCalculator, 84
- ModularResnik, 85
 - calculateNormalizedTermSimilarity, 86
 - calculateTermSimilarity, 86
 - ModularResnik, 85
 - setSharedInformationCalculator, 86
- NCList, 86
 - _items, 89

- getFeatures, 87
- getFeaturesAt, 87
- getFeaturesInRange, 87, 88
- getNodes, 88
- getOverlapIndex, 88
- getRegionAt, 88
- NCList, 87
- numFeatures, 88
- size, 88
- NCList::NCNode, 89
 - _containments, 90
 - _region, 90
 - contains, 90
 - getRegion, 90
 - NCNode, 89
 - overlaps, 90
- NCNode
 - NCList::NCNode, 89
- numFeatures
 - NCList, 88
- oneSidedRawPvalue_hyper
 - EnrichmentTools, 15
- Onto
 - GO, 16
- ontology
 - GoGraph::VertexProps, 121
- ontologyStringToCode
 - GO, 17
- ontologyToString
 - GO, 17
- operator[]
 - TermDepthMap, 109
 - TermProbabilityMap, 115
- OutEdgeIterator
 - GoGraph, 65
- overlaps
 - NCList::NCNode, 90
 - SimpleRegion, 103
- paramList
 - AppUtilities, 13
- parseAnnotationFile
 - AnnotationParserInterface, 38
 - EntrezGene2GoAnnotationParser, 50
 - GoaAnnotationParser, 61
- parseGoFile
 - AllowedRelationshipOboGoParser, 20
 - AllowedRelationshipXmlGoParser, 22
 - GoParserInterface, 74
 - RapidXmlGoParser, 95
 - StandardOboGoParser, 105
 - StandardXmlGoParser, 107
- parseParamFile
 - AppUtilities, 13
- parseSimpleFile
 - AppUtilities, 13
- PekarStaabSimilarity, 91
 - calculateNormalizedTermSimilarity, 92
 - calculateTermSimilarity, 92
 - getLCA, 92
 - PekarStaabSimilarity, 92
- PrecomputedMatrixTermSimilarity, 92
 - calculateNormalizedTermSimilarity, 93
 - calculateTermSimilarity, 93
 - PrecomputedMatrixTermSimilarity, 93
 - projectTermSet, 93
- projectTermSet
 - PrecomputedMatrixTermSimilarity, 93
- RapidXmlGoParser, 94
 - clone, 94
 - isFileGood, 94
 - parseGoFile, 95
- relType
 - GoGraph::EdgeProps, 48
- Relationship
 - GO, 16
- RelationshipPolicyInterface, 95
 - isAllowed, 96
- relationshipStringToCode
 - GO, 17
- relationshipToString
 - GO, 17
- RelevanceSimilarity, 96
 - calculateNormalizedTermSimilarity, 97
 - calculateTermSimilarity, 97
 - getMICA, 97
 - RelevanceSimilarity, 97
- ResnikSimilarity, 97
 - calculateNormalizedTermSimilarity, 98
 - calculateTermSimilarity, 98
 - getMICA, 99
 - ResnikSimilarity, 98
- setPolicy
 - AllowedRelationshipOboGoParser, 20
 - AllowedRelationshipXmlGoParser, 22
- setSharedInformationCalculator
 - ModularJiangConrath, 83
 - ModularLin, 84
 - ModularResnik, 86
- setToVec
 - AppUtilities, 13
- SetUtilities, 17
- sharedInformation
 - AncestorMeanSharedInformation, 29, 30
 - CoutoGraSMAAdjustedSharedInformation, 42
 - CoutoGraSMSharedInformation, 44
 - ExclusivelyInheritedSharedInformation, 53
 - FrontierSharedInformation, 55, 56
 - MICASharedInformation, 81
 - SharedInformationInterface, 100
- SharedInformationInterface, 99
 - hasTerm, 100
 - isSameOntology, 100
 - maxInformationContent, 100
 - sharedInformation, 100

- SimpleAccumulator
 - Accumulators, 10
- SimpleRegion, 101
 - _end, 103
 - _id, 103
 - _start, 103
 - contains, 102
 - distance, 102
 - getEnd, 102
 - getId, 102
 - getStart, 103
 - midpoint, 103
 - overlaps, 103
 - SimpleRegion, 102
- size
 - NCList, 88
- splitTake
 - AppUtilities, 13
- splitWith
 - AllowedRelationshipOboGoParser, 20
- StandardOboGoParser, 104
 - clone, 104
 - isFileGood, 104
 - parseGoFile, 105
 - StandardOboGoParser, 104
- StandardRelationshipPolicy, 105
 - isAllowed, 106
 - StandardRelationshipPolicy, 106
- StandardXmlGoParser, 106
 - clone, 107
 - isFileGood, 107
 - parseGoFile, 107
 - StandardXmlGoParser, 107
- TermDepthMap, 108
 - _depths, 110
 - _nameToIndex, 110
 - ~TermDepthMap, 109
 - getKeys, 109
 - getValue, 109
 - getValues, 109
 - hasTerm, 109
 - operator[], 109
 - TermDepthMap, 109
- termId
 - GoGraph::VertexProps, 121
- TermInformationContentMap, 110
 - badIdValue, 111
 - TermInformationContentMap, 111
- TermProbabilityMap, 112
 - _bp_normalization_min_1anno, 115
 - _bp_normalization_min_minAnno, 115
 - _cc_normalization_min_1anno, 115
 - _cc_normalization_min_minAnno, 115
 - _isSingleAnnoMin, 115
 - _mf_normalization_min_1anno, 115
 - _mf_normalization_min_minAnno, 116
 - _nameToIndex, 116
 - _probabilities, 116
 - ~TermProbabilityMap, 113
 - badIdValue, 114
 - getKeys, 114
 - getMinBP, 114
 - getMinCC, 114
 - getMinMF, 114
 - getValue, 114
 - getValues, 114
 - hasTerm, 114
 - operator[], 115
 - TermProbabilityMap, 113
- TermProbabilityMap::dfs_cumulative_annotations_↔
 - visitor, 45
 - finish_vertex, 46
- TermSetSimilarityInterface, 116
 - calculateSimilarity, 117
- TermSimilarityInterface, 117
 - calculateNormalizedTermSimilarity, 118
 - calculateTermSimilarity, 118
- TermSimilarityWriter, 118
 - TermSimilarityWriter, 119
 - writeSimilarityMatrix, 119
 - writeSimilarityMatrixBP, 119
 - writeSimilarityMatrixCC, 119
 - writeSimilarityMatrixMF, 120
- VarianceAccumulator
 - Accumulators, 11
- vecToSet
 - AppUtilities, 14
- VertexIndexMap
 - GoGraph, 65
- writeSimilarityMatrix
 - TermSimilarityWriter, 119
- writeSimilarityMatrixBP
 - TermSimilarityWriter, 119
- writeSimilarityMatrixCC
 - TermSimilarityWriter, 119
- writeSimilarityMatrixMF
 - TermSimilarityWriter, 120