Package 'rBiopaxParser'

August 22, 2012

Type Package

Title Parses BioPax level 2 files and represents them in R

Version 0.14

Date 2012-08-22

Author Frank Kramer

Maintainer Frank Kramer <dev@frankkramer.de>

Description Parses BioPAX files (at the moment only BioPAX level 2 is supported, Level 3 is comming soon) and represents them in R.

License GPL (>= 2)

Imports XML, graph

Suggests Rgraphviz, RCurl

URL https://github.com/frankkramer/rBiopaxParser

BuildVignettes false

Collate 'biopax2Classes.R' 'downloadBiopaxData.R' 'helperFunctions.R' 'modifyBiopax.R' 'parseBiopax.R' 'rBiopaxParser-package.R' 'selectBiopax.R' 'visualizeBiopax.R' 'writeBiopax.R'

R topics documented:

`rBiopaxParser-package`

Parses BioPax level 2 files and represents them in R

Description

Parses BioPax level 2 files and represents them in R

Details

rBiopaxParser is a...

Package:	rBiopaxParser
Type:	Package
Version:	0.09
Date:	2012-07-02
License:	GPL (>= 2)

Author(s)

Frank Kramer `<dev@frankkramer.de>`

Examples

```
## Not run: readBiopax(file="biopaxmodel.owl)"
```

`addBiochemicalReaction`

This function adds a new biochemical reaction to the biopax model.

Description

This function adds a new biochemical reaction of class biochemicalReaction to the biopax model. This is a convenience function, internally the function addBiopaxInstance is called with properties LEFT and RIGHT set.

Usage

```
addBiochemicalReaction(biopax, LEFT = c(), RIGHT = c(),
  ID = NULL)
```

Arguments

biopax	A biopax model
LEFT	vector of strings. IDs of the physicalEntityParticipant instances that are on the left side of this reaction.
RIGHT	vector of strings. IDs of the physicalEntityParticipant instances that are on the right side of this reaction.
ID	string. ID for the control. If NULL a new ID is generated with prefix "biochemicalReaction".

Value

Returns the biopax model with the added pathway.

Author(s)

fkramer

---

addBiopaxInstance *This function adds a new instance an existing biopax model.*

Description

This function adds a new instance an existing biopax model. "properties" is a named list of vectors, with the vector name as the name of the property and every entry of the vector a property value. Please note: case sensitivity! In Biopax Level 2 all properties are written in all capital letters.

Usage

```
addBiopaxInstance(biopax, instancetype, instanceid,
  properties = list(NAME = c()), verbose = TRUE)
```

Arguments

biopax	A biopax model
instancetype	string. Class name
instanceid	string. ID of the instance
properties	named list of properties.
verbose	logical. Be verbose about what was added.

Value

Returns the supplied biopax model with the new instance added.

Author(s)

Frank Kramer

---

addBiopaxInstances *This function adds new instances to an existing biopax model.*

Description

This function adds new instances (supplied as a compatible data.frame) to an existing biopax model.

Usage

```
addBiopaxInstances(biopax, newInstancesDF)
```

Arguments

biopax	A biopax model
newInstancesDF	data.frame. Compatible with internal Biopax Level 2 implementation.

Value

Returns the supplied biopax model with the new instances added.

Author(s)

Frank Kramer

addControl *This function adds a new control to the biopax model.*

Description

This function adds a new interaction of class control to the biopax model. This is a convenience function to add controls, internally the function addBiopaxInstance is called with properties CONTROL-TYPE, CONTROLLER and CONTROLLED set.

Usage

```
addControl(biopax,
  CONTROL_TYPE = c("ACTIVATION", "INHIBITION"),
  CONTROLLER = "", CONTROLLED = c(), ID = NULL)
```

Arguments

biopax	A biopax model
CONTROL_TYPE	string. Specifies wether this is an activating or inhibiting control.
CONTROLLER	string. ID of the physicalEntityParticipant instance that is the controller of this interaction.
CONTROLLED	vector of strings. IDs of the interaction and/or pathway instances that are being controlled.
ID	string. ID for the control. If NULL a new ID is generated with prefix "control".

Value

Returns the biopax model with the added pathway.

Author(s)

fkramer

addhash *Adds a hash in front of a string*

Description

Adds a hash in front of a string

Usage

```
addhash(x)
```

Arguments

x	A string to be preceeded by a hash

Value

The supplied string with a hash "#" pasted in front of it.

Author(s)

Frank Kramer #

`addns` *Add a namespace tag to the supplied classname string*

Description

This function takes the input classname, checks if it already has a namespace, and if not pastes the namespace tag with a dividing ":" in front of it.

Usage

```
addns(classname, namespace = "bp")
```

Arguments

classname	A string containing a classname
namespace	A string containing a namespace

Value

If the classname is not preceeded by a namespace yet, the supplied namespace is pasted in front of it and returned.

Author(s)

Frank Kramer #

`addPathway` *This function adds a new pathway to the biopax model.*

Description

This function adds a new pathway + its PATHWAY-COMPONENTS (references to interaction/pathways/pathwaySteps)

Usage

```
addPathway(biopax, NAME, PATHWAY_COMPONENTS = c(),
  ID = NULL, ORGANISM = NULL, COMMENT = NULL)
```

Arguments

`biopax`	A biopax model
`NAME`	string. Name of the pathway
`PATHWAY_COMPONENTS`	character vector. IDs of the pathway components. This must be IDs of instances of type interaction/pathway/pathwayStep (or their subclasses).
`ID`	string. ID for the pathway. If NULL a new ID is generated with prefix "pathway".
`ORGANISM`	string. Organism property of the pathway. optional.
`COMMENT`	string. An optional comment

Value

Returns the biopax model with the added pathway.

Author(s)

fkramer

`addPathwayComponents` *This function adds pathway components to an existing pathway*

Description

This function adds pathway components to an existing pathway. Property PATHWAY-COMPONENTS are references to IDs of interaction/pathways/pathwaySteps (or subclasses of those)

Usage

```
addPathwayComponents(biopax, ID,
  PATHWAY_COMPONENTS = c())
```

Arguments

biopax	A biopax model
ID	string. ID for the pathway
PATHWAY_COMPONENTS	character vector. IDs of the pathway components. This must be IDs of instances of type interaction/pathway/pathwayStep (or their subclasses).

Value

Returns the biopax model with the pathway components added to the pathway

Author(s)

fkramer

---

addPhysicalEntity *This function adds a new physical entity.*

Description

This function adds a new physical entity of chosen class to the biopax model. This is a convenience function to add physical entities, internally the function addBiopaxInstance is called with properties NAME and ORGANISM set.

Usage

```
addPhysicalEntity(biopax,
  class = c("dna", "rna", "protein", "smallMolecule", "complex")[1],
  NAME, ID = NULL, ORGANISM = NULL, COMMENT = NULL)
```

Arguments

biopax	A biopax model
class	string. Class of the physical entity to add, choose from c("dna","rna","protein","smallMolecule","com
NAME	string. Name of the new physical entity
ID	string. ID for the physical entity. If NULL a new ID is generated with prefix "physicalEntity".
ORGANISM	string. Organism property of the molecule. optional.
COMMENT	string. An optional comment

Value

Returns the biopax model with the added physical entity.

Author(s)

fkramer

`addPhysicalEntityParticipant`
This function adds a new physical entity participant.

Description

This function adds a new physical entity participant instance, which is a placeholder for physicalEntity class instances in interactions. This is a convenience function to add physicalEntityParticipant instances, internally the function addBiopaxInstance is called.

Usage

```
addPhysicalEntityParticipant(biopax,
  referencedPhysicalEntityID, ID = NULL)
```

Arguments

`biopax`	A biopax model
`referencedPhysicalEntityID`	string. ID the new physicalEntity instance to reference here.
`ID`	string. ID for the physical entity participant. If NULL a new ID is generated with prefix "physicalEntityParticipant".

Value

Returns the biopax model with the added physicalEntityParticipant.

Author(s)

fkramer

`addPropertiesToBiopaxInstance`
This function adds new properties to an existing biopax instance.

Description

This function adds new properties to an existing biopax instance.

Usage

```
addPropertiesToBiopaxInstance(biopax, instanceid,
  properties)
```

Arguments

biopax	A biopax model
instanceid	string. ID of the instance
properties	named list of properties.

Value

Returns the supplied biopax model with new properties added to this instance.

Author(s)

Frank Kramer

---

calcGraphOverlap *This function calculates the overlap of 2 graphs*

Description

This function calculates the overlap of supplied graph1 wtih graph2. Layout and weights of graph1 are kept.

Usage

```
calcGraphOverlap(graph1, graph2)
```

Arguments

graph1	graphNEL
graph2	graphNEL

Value

Returns a list containing the compared graphs and edge- and node-wise overlap between them.

Author(s)

Frank Kramer

checkValidity *This function checks the supplied biopax model for validity.*

Description

This function checks the supplied biopax model for validity, concerning classes, properties, etc. Not yet implemented. Called internally by writeBiopax.

Usage

```
checkValidity(biopax)
```

Arguments

biopax	A biopax model

Value

logical. Returns TRUE is the biopax model is valid Biopax Level 2, or FALSE otherwise.

Author(s)

Frank Kramer

CLASS_INHERITANCE *CLASS_INHERITANCE*

Description

Class inheritance relationships in Biopax Level 2.

Format

A data frame with 46 rows and 2 columns

Details

A data.frame listing all direct superclasses for every Biopax Level 2 class. The variables are as follows:

- class. Name of the class
- superclass. Name of the superclass

CLASS_PROPERTIES *CLASS_PROPERTIES*

Description

Class properties in Biopax Level 2.

Format

A data frame with 106 rows and 4 columns

Details

A data.frame listing all direct properties for every Biopax Level 2 class. Together with CLASS_INHERITANCE this allows to list all properties, including the inherited ones, of every class.

The variables are as follows:

- class. Name of the class
- property. Name of the superclass
- property_type.Type of the property, value or reference
- cardinality. Maximum allowed cardinality of a property. Many properties may only be singular.

diffGraphs *This function returns the different nodes and edges between graph1 and graph2.*

Description

This function returns the different nodes and edges between graph1 and graph2. Layout options of graph1 are kept. Coloring currently not implemented.

Usage

```
diffGraphs(graph1, graph2, colorNodes = TRUE,
  colors = c("#B3E2CD", "#FDCDAC"))
```

Arguments

graph1	graphNEL
graph2	graphNEL
colorNodes	logical
colors	character vector of colors. If colorNodes==TRUE these colors are used for graph1 and graph2 respectivley.

Value

Return the diff between the graphs.

Author(s)

Frank Kramer

`downloadBiopaxData` *This function downloads Biopax data from online databases*

Description

This function has an internal list of download links for some online databases. It will retrieve the selected model from the selected database using RCurl. The downloaded file is (if needed) unzipped and ready to be used as input for rBiopaxParser::readBiopax. This function requires package RCurl to run. Automatically unzipping the downloaded BioPAX file requires Linux. You can easily skip this step by downloading the exported file yourself and continuing with readBiopax.

Usage

```
downloadBiopaxData(database = "NCI",
  model = c("pid", "biocarta", "reactome"),
  outputfile = "", version = "biopax2")
```

Arguments

`database`	string. Select which database you want to download from. Currently only NCI links have been stored.
`model`	string. Select which model/file you want to download. Currently NCI versions of the Pathway Interaction Database, Biocarta and Reactome are linked.
`version`	string. Select which Biopax Version you want to download.
`outputfile`	string. The file name to save the downloaded data in. If left empty the URL file name will be used. The unzipped file name can be different from this. Check the screen output of gunzip.

Value

none. Check output for the name of the unzipped biopax .owl file.

Author(s)

fkramer

`generateNewUniqueID` *This function generates a new unique id for a biopax model*

Description

This function generates a new unique id for a biopax model. Pass it an startin g point like "pathway" or "protein" to get a niceer looking id.

Usage

```
generateNewUniqueID(biopax, id = "")
```

Arguments

`biopax`	A biopax model
`id`	string. This is used as a prefix for the id.

Value

Returns an unused unique ID.

Author(s)

fkramer

`getBiopaxIDsByName` *Returns all ids of instances with a certain name.*

Description

Returns all ids of instances with a certain name.

Usage

```
getBiopaxIDsByName(biopax, name)
```

Arguments

`biopax`	A biopax model
`name`	string. Name of the instance

Value

Returns a vector containing all instance ids with the supplied name.

Author(s)

Frank Kramer

`getBiopaxInstancesByID`
Returns all instances with a certain ID.

Description

Returns all instances with a certain ID.

Usage

```
getBiopaxInstancesByID(biopax, id)
```

Arguments

`biopax`	A biopax model
`id`	string. ID of the instance

Value

Returns a data.frame containing all instances with the supplied ID.

Author(s)

Frank Kramer

`getBiopaxInstancesByName`
Returns all instances with a certain name.

Description

Returns all instances with a certain name.

Usage

```
getBiopaxInstancesByName(biopax, name)
```

Arguments

`biopax`	A biopax model
`name`	string. Name of the instance

Value

Returns a data.frame containing all instances with the supplied name.

Author(s)

Frank Kramer

`getBiopaxInstancesByType`
Returns all instances of a certain class.

Description

Returns all instances of a certain class.

Usage

```
getBiopaxInstancesByType(biopax, type)
```

Arguments

`biopax`	A biopax model
`type`	string. Class of the instance

Value

Returns a data.frame containing all instances of the supplied class.

Author(s)

Frank Kramer

`getBiopaxInstancesList`
Returns all list of all instances of a certain class, or all instances.

Description

Returns a character vector of IDs of all instances of a certain class, or all instances if type=="all".

Usage

```
getBiopaxInstancesList(biopax, type = "all")
```

Arguments

`biopax`	A biopax model
`type`	string. Class of the instances

Value

Returns a character vector containing all instances of the supplied class.

Author(s)

Frank Kramer

`getClassProperties` *This function returns the properties of the supplied biopax class.*

Description

This function returns the superclasses of the supplied biopax class. It always considers inhertance. Every class inhertis the properties of its super classes.

Usage

```
getClassProperties(classname)
```

Arguments

`classname`	A string containing a class name

Value

Returns character vector containing the superclasses of the supplied class

Author(s)

Frank Kramer

`getComplexComponentList` *This function lists all components of a given complex.*

Description

This function returns a vector of all component IDs of the supplied complex.

Usage

```
getComplexComponentList(biopax, id)
```

Arguments

`biopax`	A biopax model
`id`	string. A complex ID

Value

Returns a character vector with IDs

Author(s)

Frank Kramer

`getInstanceClass` *This function returns the class name of the instance.*

Description

This function returns the class name of the instance.

Usage

```
getInstanceClass(biopax, instanceid)
```

Arguments

`biopax`	A biopax model
`instanceid`	string

Value

Returns the class name of the biopax instance.

Author(s)

fkramer

`getInstanceProperty` *This function returns all properties of the specified type for an instance.*

Description

This function returns all properties of the specified type for an instance.

Usage

```
getInstanceProperty(biopax, instanceid,
  property = "NAME")
```

Arguments

`biopax`	A biopax model
`instanceid`	string
`property`	string. Attention: All properties in Biopax Level 2 are all upper case.

Value

Returns a character vector with all properties of the selected type for this instance. Returns NULL if no property of this type is found.

Author(s)

fkramer

`getNeighborhood` *This function returns the neighborhood of a physicalEntity*

Description

This function searches the supplied biopax for interactions that are connected to the molecule or within 'depth' number of steps from it.

Usage

```
getNeighborhood(biopax, instanceid, depth = 1,
  onlyInPathways = c())
```

Arguments

biopax	A biopax model
instanceid	string. ID of a physicalEntity (dna, rna, protein, complex, smallMolecule)
depth	integer. Search depth, this specifies how far out from the specified molecule the neighborhood should be streched.
onlyInPathways	character vector of pathway IDs. Search only in these pathways for neighbors.

Value

Returns ids of interactions within 'depth' number of steps of the specified physicalEntity

Author(s)

fkramer

---

getPathway *This function returns a data.frame containing the pathway.*

Description

This function returns a data.frame containing the pathway instance pointed at by id.

Usage

```
getPathway(biopax, id)
```

Arguments

biopax	A biopax model
id	string. ID of the pathway instance

Value

Returns a data.frame containing the pathway instance.

Author(s)

Frank Kramer

`getPathwayComponentList` *This function lists all pathway components of a given pathway.*

Description

This function returns a (unique) vector of pathway component IDs of the supplied pathway.

Usage

```
getPathwayComponentList(biopax, id)
```

Arguments

`biopax`	A biopax model
`id`	string. A pathway ID

Value

Returns a character vector with IDs

Author(s)

Frank Kramer

`getPathwayComponents` *This function returns a data.frame containing all pathway components.*

Description

This function returns a data.frame containing all instances referenced in a pathways PATHWAY-COMPONENTS property.

Usage

```
getPathwayComponents(biopax, id)
```

Arguments

`biopax`	A biopax model
`id`	string. Pathway ID

Value

Returns a data.frame containing all pathway components.

Author(s)

Frank Kramer

`getPathwayList` *This function returns a list of all pathway ids.*

Description

This function returns a vector of all pathway ids.

Usage

```
getPathwayList(biopax)
```

Arguments

`biopax`	A biopax model

Value

Returns a character vector containing the names of all pathways.

Author(s)

Frank Kramer

`getPathways` *This function returns a data.frame containing all pathways.*

Description

This function returns a data.frame containing the all pathway instances.

Usage

```
getPathways(biopax)
```

Arguments

`biopax`	A biopax model

Value

Returns a data.frame containing all pathway instances.

Author(s)

Frank Kramer

`getReferencedIDs` *This function returns a vector of ids of all referenced instances of the supplied instance.*

Description

This function takes an id and a biopax model as input. The id of every instance that is referenced is returned. If recursive == TRUE this function recurses through all referenced IDs of the referenced instances and so on. "onlyFollowProperties" limits the recursivness to only certain properties, for example follow only complexes or physicalEntities.

Usage

```
getReferencedIDs(biopax, id, recursive = TRUE,
  onlyFollowProperties = c())
```

Arguments

`biopax`	A biopax model
`id`	string. ID of the instance
`recursive`	logical
`onlyFollowProperties`	character vector

Value

Returns a character vector of IDs referenced by the supplied id in the supplied biopax model.

Author(s)

Frank Kramer

`getReferencedInstances` *This function returns a data.frame containing all referenced instances of the supplied instance.*

Description

This function takes an id and a biopax model as input. A data.frame containing all instances referenced by the supplied instance is returned. If recursive == TRUE this function recurses through all referenced IDs of the referenced instances and so on. "onlyFollowProperties" limits the recursivness to only certain properties, for example follow only complexes or physicalEntities.

Usage

```
getReferencedInstances(biopax, id, recursive = TRUE,
  onlyFollowProperties = c())
```

Arguments

`biopax`	A biopax model
`id`	string
`recursive`	logical
`onlyFollowProperties`	character vector

Value

Returns a data.frame containing all referenced instances.

Author(s)

Frank Kramer

`getReferencingIDs` *This function returns a vector of ids of all instances that reference the supplied instanceid.*

Description

This function takes an id and a biopax model as input. The id of every instance that references the supplied id is returned. If recursive == TRUE this function recurses through all referencing IDs of the referencing instances and so on. "onlyFollowProperties" limits the recursivness to only certain properties, for example follow only complexes or physicalEntities.

Usage

```
getReferencingIDs(biopax, id, recursive = TRUE,
  onlyFollowProperties = c())
```

Arguments

biopax	A biopax model
id	string. ID of the instance
recursive	logical
onlyFollowProperties	character vector

Value

Returns a character vector of IDs referencing the supplied id in the supplied biopax model.

Author(s)

Frank Kramer

---

getReferencingInstances

This function returns a data.frame containing all instances referencing the supplied instance.

Description

This function takes an id and a biopax model as input. A data.frame containing all instances referencing the supplied instance is returned. If recursive == TRUE this function recurses through all referencing IDs of the referencing instances and so on. "onlyFollowProperties" limits the recursivness to only certain properties, for example follow only complexes or physicalEntities.

Usage

```
getReferencingInstances(biopax, id, recursive = TRUE,
  onlyFollowProperties = c())
```

Arguments

biopax	A biopax model
id	string
recursive	logical
onlyFollowProperties	character vector

Value

Returns a data.frame containing all referencing instances.

Author(s)

Frank Kramer

`getSubClasses` *This function returns the subclasses of the supplied biopax class.*

Description

This function returns the subclasses of the supplied biopax class.

Usage

```
getSubClasses(classname)
```

Arguments

`classname`	A string containing a class name

Value

Returns character vector containing the subclasses of the supplied class

Author(s)

Frank Kramer

`getSuperClasses` *This function returns the superclasses of the supplied biopax class.*

Description

This function returns the superclasses of the supplied biopax class.

Usage

```
getSuperClasses(classname)
```

Arguments

`classname`	A string containing a class name

Value

Returns character vector containing the superclasses of the supplied class

Author(s)

Frank Kramer

`getXrefAnnotations` *This function returns the annotations of the supplied IDs*

Description

This function returns the annotations of the supplied IDs in a data.frame.

Usage

```
getXrefAnnotations(biopax, id, splitComplexes = FALSE,
  followPhysicalEntityParticipants = TRUE)
```

Arguments

`biopax`	A biopax model
`id`	vector of strings. IDs of instances to get annotations
`splitComplexes`	logical. If TRUE complexes are split up into their components and the annotation of the components is added.
`followPhysicalEntityParticipants`	logical. If TRUE physicalEntityParticipants are resolved to their corresponding physicalEntities and their annotation is added.

Value

Returns data.frame with annotations

Author(s)

fkramer

hasProperty *Checks if instances in the biopax data.frame have a given property*

Description

Checks if instances in the biopax data.frame have a given property

Usage

```
hasProperty(df, property)
```

Arguments

df	A data.frame with biopax instances
property	A string containing the name of the property to check for

Value

Returns TRUE for every row in the data.frame with contains the supplied property

Author(s)

Frank Kramer

internal_checkArguments *This function checks the supplied arguments if they abid to the given restrictions*

Description

This function checks the supplied arguments if they abid to the given restrictions

Usage

```
internal_checkArguments(args = c(),
  allowedValues = list(), allowNULL = FALSE,
  allowNA = FALSE, allowEmptyString = TRUE,
  allowInf = TRUE)
```

Arguments

`args`	The vector of arguments to check
`allowedValues`	A named list of values the argument of a this name is allowed to have
`allowNULL`	Logical, allow NULL or not
`allowNA`	Logical, allow NA or not
`allowEmptyString`	Logical, allow empty strings or not
`allowInf`	Logical, allow values of +/- infinity or not

Value

Returns 1 if all checks completed successfully, returns error message otherwise.

Author(s)

Frank Kramer #

`internal_generateXMLfromBiopax`

This function generates the xmlTree from the supplied biopax model.

Description

This function is used internally by writeBiopax. It can also be called directly with a fitting dataframe in list(df=data.frame()), but this will probably break things.

Usage

```
internal_generateXMLfromBiopax(biopax,
  namespaces = namespaces)
```

Arguments

`biopax`	A biopax model
`namespaces`	A list of namespaces to use for the generated XML/RDF file

Value

Returns the xmlTree generated from the supplied biopax model.

Author(s)

Frank Kramer #

`internal_getBiopaxModelAsDataFrame`
This internal function parses the Biopax XML of the supplied biopax model and returns it in the data.frame format.

Description

This internal function parses the Biopax XML of the supplied biopax model and returns it in the data.frame format.

Usage

```
internal_getBiopaxModelAsDataFrame(biopax,
  verbose = TRUE)
```

Arguments

`biopax`	A biopax model
`verbose`	logical

Value

Returns the parsed biopax model in the internal data.frame format.

Author(s)

Frank Kramer #

`internal_NrOfXMLNodes`
This function in an internal function to count the Number of nodes and child nodes of an XMLNode.

Description

This function in an internal function to count the Number of nodes and child nodes of an XMLNode.

Usage

```
internal_NrOfXMLNodes(myXMLNode)
```

Arguments

`myXMLNode`	XMLNode to analyze

Value

This function returns the number of Nodes and child Nodes an XMLNode has.

Author(s)

Frank Kramer #

`internal_propertyListToDF` *Internal function to build a data.frame from the list of properties for a new instance*

Description

Internal function to build a data.frame from the list of properties for a new instance

Usage

```
internal_propertyListToDF(instancetype, instanceid,
  properties, namespace_rdf = "rdf")
```

Arguments

`instancetype`	string. Class name
`instanceid`	string. ID of the instance
`properties`	named list of properties.
`namespace_rdf`	string. This defines the rdf namespace to use.

Value

Returns a data.frame with the new properties for the given instance

Author(s)

Frank Kramer #

`internal_resolvePhysicalEntityParticipant` *This function resolves physicalEntityParticipantIDs to their corresponding physicalEntityIDs*

Description

This function resolves physicalEntityParticipantIDs to their corresponding physicalEntityIDs. Every physicalEntityParticipant corresponds exactly to one physicalEntity.

Usage

```
internal_resolvePhysicalEntityParticipant(biopax,
  physicalEntityId)
```

Arguments

`biopax`	A biopax model
`physicalEntityId`	string. IDs of physicalEntityParticipants to be resolved

Value

Returns ids of physicalEntity corresponding to the specified physicalEntityParticipantIDs

Author(s)

fkramer

`internal_XMLInstance2DF` *This function in an internal function that parses a Biopax Level 2 XMLNode.*

Description

This function in an internal function that parses a Biopax Level 2 XMLNode.

Usage

```
internal_XMLInstance2DF(myXMLNode, namespace_rdf)
```

Arguments

`myXMLNode`	XMLNode
`namespace_rdf`	String specifying the namespace to use for rdf:resource and rdf:datatype

Value

Returns the matrix generated by parsing the XMLNode

Author(s)

Frank Kramer #

`intersectGraphs` *This function returns a graph computed by the insection of supplied graph1 and graph2.*

Description

This function returns a graph computed by the insection of supplied graph1 and graph2. Layout and weights of graph1 are kept.

Usage

```
intersectGraphs(graph1, graph2)
```

Arguments

`graph1`	graphNEL
`graph2`	graphNEL

Value

Returns the intersection of graph1 and graph2.

Author(s)

Frank Kramer

`isOfClass` *Checks if instances in the biopax data.frame are of the given class*

Description

This function checks if instances in the supplied biopax data.frame are of a given class. If considerInheritance is set to TRUE it also checks if instances are of a given class or any of its inherited classes.

Usage

```
isOfClass(df, class, considerInheritance = FALSE)
```

Arguments

df	A data.frame with biopax instances
class	A string containing the class name to check for
considerInheritance	Logical value indicating wether to consider inheritance or not

Value

Returns TRUE for every row in the data.frame which is of the supplied class

Author(s)

Frank Kramer

---

isOfNamespace *Check if a classname is preceeded by a certain namespace tag like in "namespace:classname"*

Description

This function checks if the supplied input string starts with a supplied namespace tag

Usage

```
isOfNamespace(classname, namespace = "bp")
```

Arguments

classname	A string containing the classname to check
namespace	A string giving the namespace to check for

Value

This function returns TRUE if the supplied classname string is preceeded with the supplied namespace string, and FALSE if not.

Author(s)

Frank Kramer #

`layoutRegulatoryGraph` *This function generates a (more or less) beautiful layout for a regulatory graph.*

Description

This function generates a (more or less) beautiful layout for a regulatory graph. Call this after you generated a graph with pathway2RegulatoryGraph. Since beauty is always in the eye of the beholder consider this a starting point for making your graphs even nicer. Rgraphviz with dot layout is used. Edges are green/red with normal/tee arrowheads for activations/inhibitions. If you want to specifically paint subgraphs in different colors use lists of vectors with node names for parameter subgraphs and vector of color names for subgraphs.color for your choice of color. The output can be further tweaked by setting layout options using nodeRenderInfo(mygraph) <- list() ... See the Rgraphviz and Graphviz documentations.

Usage

```
layoutRegulatoryGraph(mygraph, label = "",
  node.height = 2, node.width = 2, node.fontsize = 20,
  node.labelfontsize = 20, node.fixedsize = FALSE,
  edge.weights = c("green", "black", "red"),
  edge.arrowheads = c("normal", "tee"),
  subgraphs = list(),
  subgraphs.colors = c("#B3E2CD", "#FDCDAC", "#F4CAE4", "#E6F5C9", "#FFF2AE"))
```

Arguments

`mygraph`	graphNEL
`label`	Label of the graph
`node.height`	Height of the nodes
`node.width`	Width of the nodes
`node.fontsize`	Sets the fontsize of nodes.
`node.labelfontsize`	Set labelfontsize of nodes
`node.fixedsize`	logical. If font size is fixed or variable in regards to the nodes.
`edge.weights`	vector. which colors to use for weighted edges
`edge.arrowheads`	vector. which arrowheads to use for weighted edges
`subgraphs`	A list of character vectors with node names defining the sub graphs.
`subgraphs.colors`	vector. which colors to use for subgraphs

Value

Returns the supplied graph in a layouted form with several parameters set for regulatory graph plotting.

Author(s)

Frank Kramer

`listBiopaxNodes` *This function returns a summary statistic for the biopax model.*

Description

This function returns a summary statistic for the biopax model. It checks if the summary has already been generated with summary.biopax and generates it if necessary. Have a look at the biopax$summary entry for more statistics. This function can take a while with really big Biopax files like NCIs Pathway Interaction Database or Reactome.

Usage

```
listBiopaxNodes(biopax)
```

Arguments

`biopax`	A biopax model

Value

This function returns a summary statistic for the biopax model.

Author(s)

Frank Kramer

`mergePathways` *This function merges two given pathways*

Description

This function merges two given pathways and appends it to the supplied biopax model. The user has to specify a new name for the pathways and can supply ID, ORGANISM and COMMENT properties for the new pathway. If no ID is supplied, a new unique ID is generated. If no organism property is supplied the organism property of the first pathway is re-used. If ORGANISM is NULL the property is not set. Optionally a comment can be added to the pathway.

Usage

```
mergePathways(biopax, pwid1, pwid2, NAME, ID = NULL,
  ORGANISM = "", COMMENT = NULL)
```

Arguments

biopax	A biopax model
pwid1	string. ID of first pathway to merge
pwid2	string. ID of second pathway to merge
NAME	string. Name of the new merged pathway
ID	string. ID for the pathway. If NULL a new ID is generated with prefix "pathway".
ORGANISM	string. Organism property of the pathway. By default uses the same organism as the first supplied pathway. If NULL no organism property is set.
COMMENT	string. An optional comment

Value

A biopax model with the merged pathway added.

Author(s)

fkramer

---

pathway2AdjacancyMatrix

This function generates an adjacency matrix from the activations/inhibitions of a pathway in a biopax model.

Description

This function internally first calls pathway2RegulatoryGraph, then converts the regulatory graph to an adjacency matrix. See pathway2RegulatoryGraph for more details.

Usage

```
pathway2AdjacancyMatrix(biopax, pwid,
  expandSubpathways = TRUE, splitComplexMolecules = TRUE,
  useIDasNodenames = FALSE, verbose = TRUE)
```

Arguments

biopax	A biopax model
pwid	string
expandSubpathways	logical. If TRUE subpathways are expanded into this graph, otherwise only this very pathway is used.
splitComplexMolecules	logical. If TRUE every complex is split up into its components. This leads to splitting a single node with name of the complex into several nodes with names of the components, these components all have identical edges.
useIDasNodenames	logical. If TRUE nodes of the graph are named by their molecule IDs instead of using the NAME property. This can help with badly annotated/formatted databases.
verbose	logical

Value

Returns the adjacency matrix representing the regulatory graph of the supplied pathway.

Author(s)

Frank Kramer

---

pathway2RegulatoryGraph

This function generates the regulatory graph from the activations/inhibitions of a pathway in a biopax model.

Description

This functions builds a graph from the pathway components of the supplied pathway. Only instances of class 'control' are considered, this leads a functinal graph with all edges either representing activations or inhibitions. No transports, no translocation, etc. If desired complexes can be split up into several nodes, this can sometimes lead to a more complex and cluttered graph. There can not be multiple edges between 2 nodes. Whenever duplicated edges are generated (especially by splitting up complexes) a warning is thrown.

Usage

```
pathway2RegulatoryGraph(biopax, pwid,
  expandSubpathways = TRUE, splitComplexMolecules = TRUE,
  useIDasNodenames = FALSE, verbose = TRUE)
```

Arguments

biopax	A biopax model
pwid	string
expandSubpathways	logical. If TRUE subpathways are expanded into this graph, otherwise only this very pathway is used.
splitComplexMolecules	logical. If TRUE every complex is split up into its components. This leads to splitting a single node with name of the complex into several nodes with names of the components, these components all have identical edges.
useIDasNodenames	logical. If TRUE nodes of the graph are named by their molecule IDs instead of using the NAME property. This can help with badly annotated/formatted databases.
verbose	logical

Value

Returns the representing the regulatory graph of the supplied pathway in a node-edge-list graph.

Author(s)

Frank Kramer

---

plotRegulatoryGraph

This function layouts a regulatory graph and plots it using Rgraphviz.

Description

This function takes a regulatory graph as generated by pathway2regulatoryGraph and plots it using standard layout options of layoutRegulatoryGraph. This function is a wrapper for layoutRegulatoryGraph with standard parameters. Subgraphs can be painted with different colors. This can be done by passing parameter subgraph a list of character vectors with node names.

Usage

```
plotRegulatoryGraph(mygraph, subgraphs = list(),
  layoutGraph = TRUE)
```

Arguments

mygraph	graphNEL, regulatory graph
subgraphs	list of character vectors with node names
layoutGraph	logical. If FALSE the graph is not laid out again but send directly to Rgraphviz::renderGraph.

Value

none

Author(s)

Frank Kramer

`readBiopax` *This function reads in a Biopax .owl file*

Description

This function reads in a Biopax .owl file and generates the internal data.frame format used in this package. This function can take a while with really big Biopax files like NCIs Pathway Interaction Database or Reactome. In almost every case this is your starting point. Returns a biopax model, which is a list with named elements:

biopaxxml The XML which was read in from file.

summary The generated summary statistic.

df The data.frame representing the biopax in R

ns_rdf RDF Namespace

ns_owl OWL Namespace

ns_bp Biopax Namespace

file File name

Usage

```
readBiopax(file, verbose = TRUE, generateSummary = TRUE,
  generateDF = TRUE)
```

Arguments

`file`	string. File name
`verbose`	logical. Output messages about how parsing is going and so on.
`generateSummary`	logical. Generates and attaches the summary for the Biopax file to biopax$summary
`generateDF`	logical. If this is set to FALSE no data.frame is generated, only the file is read in to biopax$biopaxxml

Value

A biopax model

Author(s)

Frank Kramer

removeInstance *This function removes an instance*

Description

This function removes an instance from an existing biopax model.

Usage

```
removeInstance(biopax, instanceid)
```

Arguments

biopax	A biopax model
instanceid	string. ID of the instance

Value

Returns the supplied biopax model with the instance removed from it.

Author(s)

Frank Kramer

removeProperties *This function removes a property*

Description

This function removes a property fram an existing biopax instance.

Usage

```
removeProperties(biopax, instanceid, properties)
```

Arguments

biopax	A biopax model
instanceid	string. ID of the instance
properties	character vector. listing the properties to remove.

Value

Returns the supplied biopax model with properties removed from this instance.

Author(s)

Frank Kramer

`splitComplex` *This functions splits up a complex into its components.*

Description

This function looks up the supplied Complex ID and returns the names of all its components.

Usage

```
splitComplex(biopax, complexid, recursive = TRUE)
```

Arguments

`biopax`	A biopax model
`complexid`	string ID of an complex
`recursive`	logical

Value

Returns a character vector with the names of all subcomponents.

Author(s)

Frank Kramer

`striphash` *Strips a hash in front of a string*

Description

Strips a hash in front of a string

Usage

```
striphash(x)
```

Arguments

`x`	A string to be stripped off a preceeeding hash

Value

The supplied string with a hash "#" stripped off front.

Author(s)

Frank Kramer #

stripns *Strips a namespace tag off a supplied classname string*

Description

Strips a namespace tag off a supplied classname string

Usage

```
stripns(classname)
```

Arguments

classname	A string containing a classname preceeded by a namespace tag

Value

The classname with the namespace tag stripped off it.

Author(s)

Frank Kramer #

summary_biopax *This function generates a summary statistics for the biopax model.*

Description

This function generates a summary statistics for the biopax model. This function can take a while with really big Biopax files like NCIs Pathway Interaction Database or Reactome. This function is called internally by readBiopax if generateSummary == TRUE, so just check biopax$summary if it has already been generated.

Usage

```
summary_biopax(object, verbose = TRUE)
```

Arguments

object	A biopax model
verbose	logical

Value

Returns the summary for the supplied biopax model.

Author(s)

Frank Kramer

`transitiveClosure` *This function generates the transitive closure of the supplied graph.*

Description

This function generates the transitive closure of the supplied graph. In short: if A->B->C then an edge A->C is added. Edge weights are conserved if possible (in a hopefully smart way).

Usage

```
transitiveClosure(mygraph)
```

Arguments

`mygraph`	graphNEL

Value

Returns the transitive closure of the supplied graph.

Author(s)

Frank Kramer

`transitiveReduction`
This function generates the transitive reduction of the supplied graph.

Description

This function generates the transitive reduction of the supplied graph. In short: if A->B->C AND A->C then edge A->C is removed. Edge weights are conserved if possible (in a hopefully smart way).

Usage

```
transitiveReduction(mygraph)
```

Arguments

`mygraph`	graphNEL

Value

Returns the transitive reduction of the supplied graph.

Author(s)

Frank Kramer

---

unfactorize *Replace factors/levels in a data.frame and use plain strings instead*

Description

This function takes a data.frame as argument and returns it with strings instead of factors.

Usage

```
unfactorize(df)
```

Arguments

df	any data.frame with factor levels in at least one column

Value

The data.frame is returned using strings instead of factors.

Author(s)

Frank Kramer #

---

uniteGraphs *This function unites two graphs.*

Description

This function unites the two supplied graphs. Layout parameters from graph1 are used. If colorNodes==TRUE the returned graph has different colors for overlapping nodes and nodes individual for each graph.

Usage

```
uniteGraphs(graph1, graph2, colorNodes = TRUE,
  colors = c("#B3E2CD", "#FDCDAC", "#F4CAE4"))
```

Arguments

graph1	graphNEL
graph2	graphNEL
colorNodes	logical
colors	colors character vector of colors. If colorNodes==TRUE these colors are used for graph1 and graph2 respectivley.

Value

Return a graph generated by uniting the two supplied graphs

Author(s)

Frank Kramer

---

writeBiopax *This function writes out a biopax model.*

Description

This function writes out a biopax model, as generated by readBiopax, to either a file or returns the xmlTree if file is omitted.

Usage

```
writeBiopax(biopax, file = "", verbose = TRUE,
  overwrite = FALSE,
  namespaces = list(rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#", bp =
```

Arguments

biopax	A biopax model as generated by readBiopax
file	A string giving a file name.
verbose	logical
overwrite	logical, if TRUE an already existing file will be overwritten, otherwise an error is thrown
namespaces	A list of namespaces to use for the generated XML/RDF file

Value

Returns the xmlTree object generated from the biopax model. If a filename is supplied the XML is written to this file.

Author(s)

Frank Kramer