Go /home/sam/Documents/Version4 ▼

SWI-Prolog 6.6.6

Search

[✓ All] Application Manual Name [✓ Summary] Help

reactome_utility_module.pl -- Reactome Utility Predicates

- This module has some useful predicates for interacting with the
- owl:rdf ontology file for Reactome
- @author Sam Neaves

reactome_string_id(*+R:reactome_string, -Id:reactome_id*) is **semidet**

Reactome_String is a string of the form: 'http://www.reactome.org/biopax/47/48887#Protein1'

reactome_name(*?Rid:reactome_id, ?Name:atom*) is **semidet**

Use with single quotes works in both directions.

reactome_name_fast(*+Rid:reactome_id, -Name:atom*) is **semidet**

Use with single quotes only works in direction of id to name.

protein_reactome_id_to_uniprot_id(*?Reactome_Id:protein_id, ?UniprotId:uniprot_id*) is **semidet**

Use with single quotes `protein_reactome_Id_to_Uniprot_Id2('Protein7',UniprotId)` works in both directions.

reactome_string_type(*+Reactome_string:reactome_string, -ReactomeType:reactome_type*) is **semidet**

Reactome_String is a string (atom) of the form: http://www.biopax.org/release/biopax-level3.owl#BiochemicalReaction

component_type(*+Component:reactome_id, -Type:Reactome_type*) is **semidet**

This is works in one direction only and returns protein sets as proteins.

See also

- component_type_slow/2 and type_c/2

component_type_slow(*?Component:reactome_id, ?Type:reactome_type*) is **semidet**

This works in both directions but can be slow. It returns protein sets as proteins.

See also

- component_type/2 and type_c/2

type_c(*+Component:reactome_id, -Type:Reactome_type*) is **semidet**

This three place predicate definition is simple checking of proteins, protein_sets and complexes, They work in one direction. The order of these three predicates is important for speed. The _c means calculated. Rdf files that have been preprocced will have a type/2 predicate to improve speed. Unlike component_type/2 and component_type_slow/2 these return the correct type for protein sets

To be done

- add defs for small molecules, dna, rna and physical entities maybe at the momnent the parent_child and descedant relations depend on this not being the case

all_reactions(*-Reactions:list*) is **det**

finds all the *Reactions* as a list in the ontology

input_to_reaction_c(*+Reaction_Id:reaction_id, -Input:reactome_id*) is **nondet**

_c is for calculated, some ontologies should be preprocessed for speed. On back tracking finds all inputs to a given reaction. Fails if reaction has no input or if reaction does not exist.

To be done

- What to do with 'Physical entities'? Should this be changed to only return, proteins, complexs and protein_sets?

inputs_to_reaction_list(*+Reaction_Id:reaction_id, -Inputs:list*) is **semidet**

uses input_to_reaction_c/2 rather than a preprocessed input_to_reaction/2 so is not very fast

output_to_reaction_c(*+Reaction_Id:reaction_id, -Output_Id:reactome_id*) is **nondet**

_c is for calculated, some ontologies should be preprocessed for speed. On back tracking finds all outputs for a given reaction. Fails if reaction has no output or if reaction does not exist @tbd What to do with 'Physical entities'? Should this be changed to only return, proteins, complexs and protein_sets?

outputs_to_reaction_list(*+Reaction_Id:reaction_id, -Outputs:list*) is **semidet**

uses output_to_reaction_c so is not very fast

component_of_complex(*+Complex_Id:complex_id, -Component_Id:reactome_id*) is **nondet**

Back tracks to find all components of a complex

complex_c(*+Complex_Id:complex_id, -List_of_Children_Of_Complex:list*) is **semidet**

_c is for calculated the ontology would normally be preproccessed to give a complex/3

component_of_protein_set(*+Protein_Set_Id:protein_set_id, -Protein_Id:protein_id*) is **nondet**

Is *Protein_Set_Id* a protein set? If so find members of that set on backtracking.

protein_set_c(*+Protein_Set_Id:protein_set_id, -List_Of_Children_Of_Protein_Set:list*) is **semidet**

_c means calcualted. The ontology would normally be preprocessed to provide protein_set/3

child_component(*+Parent:reactome_id, -Child:reactome_id*) is **nondet**

Uses complex/2 and protein_set/2 which would be asserted on preprocessing, otherwise you would need complex_c/2 and protein_set_c/2 uses child and parent as a metaphor for membership of a protein set or componet of a protein complex

descendant_component(*+Parent:reactome_id, -Child:reactome_id*) is **nondet**

On backtracking finds all descendants of a parent component. Uses parent child and descendant as a metaphor for the relations of set membership of a protein set or component of a protein complex. Uses type/2 for speed which should be preprocessed otherwise see type_c/2

descendant_complex_or_set(*+Parent:reactome_id, -Descendant:reactome_id*) is **semidet**

Finds on backtracking all descendents of a component that will also have descendants as they are themselves complexes or protein sets.

child_type_protein(*+Parent:reactome_id, +Child:protein_id*) is **semidet**

Is it true that *Child* is a simple protein not a protien complex or set and is a *Child* of *Parent*. Uses type/2 which would be preprocessed otherwise use type_c/2

all_children_proteins*(+Parent:reactome_id)* is **semidet**

Is it true that a component such as a complex or protein set has 'children' all of which are proteins. i.e none of its children are also protein sets or complexes. type/2 would be preprocessed otherwise see type_c/2

See also

- type_c/2