BioPAX – Biological Pathways Exchange Language

Level 3, Release Version 1 Documentation

BioPAX Release, July 2010.

The BioPAX data exchange format is the joint work of the BioPAX workgroup and Level 3 builds on the work of Level 2 and Level 1.

BioPAX Level 3 input from: Mirit Aladjem, Ozgun Babur, Gary D. Bader, Michael Blinov, Burk Braun, Michelle Carrillo, Michael P. Cary, Kei-Hoi Cheung, Julio Collado-Vides, Dan Corwin, Emek Demir, Peter D'Eustachio, Ken Fukuda, Marc Gillespie, Li Gong, Gopal Gopinathrao, Nan Guo, Peter Hornbeck, Michael Hucka, Olivier Hubaut, Geeta Joshi-Tope, Peter Karp, Shiva Krupa, Christian Lemer, Joanne Luciano, Irma Martinez-Flores, Zheng Li, David Merberg, Huaiyu Mi, Ion Moraru, Nicolas Le Novere, Elgar Pichler, Suzanne Paley, Monica Penaloza-Spinola, Victoria Petri, Elgar Pichler, Alex Pico, Harsha Rajasimha, Ranjani Ramakrishnan, Dean Ravenscroft, Jonathan Rees, Liya Ren, Oliver Ruebenacker, Alan Ruttenberg, Matthias Samwald, Chris Sander, Frank Schacherer, Carl Schaefer, James Schaff, Nigam Shah, Andrea Splendiani, Paul Thomas, Imre Vastrik, Ryan Whaley, Edgar Wingender, Guanming Wu, Jeremy Zucker

BioPAX Level 2 input from: Mirit Aladjem, Gary D. Bader, Ewan Birney, Michael P. Cary, Dan Corwin, Kam Dahlquist, Emek Demir, Peter D'Eustachio, Ken Fukuda, Frank Gibbons, Marc Gillespie, Michael Hucka, Geeta Joshi-Tope, David Kane, Peter Karp, Christian Lemer, Joanne Luciano, Elgar Pichler, Eric Neumann, Suzanne Paley, Harsha Rajasimha, Jonathan Rees, Alan Ruttenberg, Andrey Rzhetsky, Chris Sander, Frank Schacherer, Andrea Splendiani, Lincoln Stein, Imre Vastrik, Edgar Wingender, Guanming Wu, Jeremy Zucker

BioPAX Level 1 input from: Gary D. Bader, Eric Brauner, Michael P. Cary, Emek Demir, Andrew Finney, Ken Fukuda, Robert Goldberg, Susumu Goto, Chris Hogue, Michael Hucka, Peter Karp, Minoru Kanehisa, Stan Letovksy, Joanne Luciano, Debbie Marks, Natalia Maltsev, Elizabeth Marland, Peter Murray-Rust, Eric Neumann, Suzanne Paley, John Pick, Aviv Regev, Andrey Rzhetsky, Chris Sander, Vincent Schachter, Imran Shah, Mustafa Syed, Jeremy Zucker

Thanks to the many additional people who contributed to discussions

on the various BioPAX mailing lists and at BioPAX meetings.

This document was edited by Nadia Anwar, Gary Bader, Emek Demir, Sylva Donaldson and Igor Rodchenkov. Also edited for Level 1 and Level 2 by Michael P. Cary.

Abstract

There are over 300 Internet-accessible databases that store biological pathway data. Biologists often need to use information from many of these to support their research, but since each has its own representation conventions and data access methods, integrating data from multiple databases is very difficult. A widely-adopted biological pathway data exchange format will help make data collection and integration easier.

BioPAX (Biological Pathway Exchange - http://www.biopax.org) enables the integration of diverse pathway resources by defining an open file format specification for the exchange of biological pathway data. By utilizing the BioPAX format, the problem of data integration reduces to a semantic mapping between the data models of each resource and the data model defined by BioPAX. Widespread adoption of BioPAX for data exchange will increase access to and uniformity of pathway data from varied sources, thus increasing the efficiency of computational pathway research.

This document describes BioPAX Level 3, which expands the scope of BioPAX to include states of physical entities, generic physical entities, gene regulation and genetic interactions. BioPAX Level 3 supports the representation of the bulk of pathway data in publicly available databases.

Scope of this document

This BioPAX documentation is targeted at computational biologists with an interest in biological pathway data. For an overview of BioPAX, read the introduction (section 1). It is expected that readers are familiar with one or more pathway databases and have a basic understanding of both bioinformatics and molecular and cellular biology. This background information is available in a number of textbooks[1].

This document provides an overview the BioPAX Level 3 ontology. This includes descriptions of the BioPAX ontology classes, sample use cases and best practice recommendations. This document does not provide a full definition of the BioPAX Level 3 ontology, which is given by the BioPAX Level 3 OWL file, located at:

http://www.biopax.org/release/biopax-level3.owl

New Features in BioPAX Level 3

The major change in BioPAX Level 3 is that the representation of physical entities (e.g. proteins) has been redesigned to support physical entities in diverse states, and generic physical entities. This has required the removal of some utility classes and the addition of some new ones. Support for new features required backwards incompatible changes compared to the BioPAX Level 1 and 2 formats, however, the majority of the classes and properties are unchanged.

Better support for physical entities in diverse states

A protein, as recorded in a sequence database like UniProt, is now represented as a **ProteinReference**, which stores the protein sequence, name, external references, and potential sequence features (this is similar in meaning to the class 'protein' in BioPAX Level 1 and 2). The actual protein chemical species post-translationally modified, bound in a complex or present in a specific cellular compartment, that participates in an interaction is now represented as the class **Protein** (this is similar in meaning to the class **physicalEntityParticipant** in BioPAX Level 1 and 2, except that **stoichiometry** is part of **Conversion** in Level 3 and there is no need to duplicate proteins, as was done with **physicalEntityParticipant**s in Level 1 and 2). This new design makes it easier to create different forms of a protein while not duplicating information common to all forms (e.g. protein sequence) and explicitly linking all forms of a protein together (through the shared **ProteinReference**). Representation of sequence features and **stoichiometry** were significantly changed. Other physical entities: DNA, RNA and small molecule, have similarly been redesigned. Only complex (now **Complex**) has not been changed, since it is composed of other physical entities that have been redesigned.

The **physicalEntityParticipant** class has been removed, as it is no longer needed with the new design. This makes BioPAX easier to use, interactions now reference their participants directly, not through an intermediate **physicalEntityParticipant** class.

Support for generic physical entities

Generic physical entities are often used in pathway databases e.g. alcohols, nucleotides (dNTPs), and the Wnt protein family (there are many different Wnt genes and proteins in some genomes). Different types of these physical entity groupings can be used, such as homology groups or groups of small molecules that share the same chemical functional group. These can now be represented using the **EntityReference** class, instances of which can contain multiple member **EntityReference**s of the same type (via the *memberEntityReference* property). Generic features, such as binding

sites or post-translational modifications across molecules, are also supported using the **EntityFeature** class and its *memberFeature* property.

Support for gene regulation networks

Gene regulation networks, involving regulators of gene expression (e.g. transcription factors, microRNAs) and their targets can now be represented. The new **TemplateReaction** class captures polymerization of macromolecule polymers from a DNA or RNA template. It stores the template, product and the regulatory region common to all types of template reaction being described (e.g. promoter for transcription, 3'UTR for translation). A new control class, **TemplateReactionRegulation**, involving an expression regulator physical entity (e.g. transcription factor), controls a **TemplateReaction**.

Support for genetic interactions

Genetic interactions, such as epistasis or synthetic lethality, are important for mapping pathways from organisms like yeast, worm, fly and mouse. This information is increasingly available in pathway and interaction databases. To capture these interactions, there is now a **GeneticInteraction** class, which contains a set of genes and a phenotype (expressed using PATO or another phenotype controlled vocabulary). Controlled vocabulary terms to support genetic interactions have also been added to the PSI-MI controlled vocabulary. The **Gene** class has also been added to support genetic interactions.

Support for degradation

Degradation of physical entities, such as proteins, is important in many regulatory pathways. A new **Degradation** class, a sub-class of **Conversion**, has been added to capture this event. The left side of the interaction contains the degradation substrate and the right side is empty, signifying that the degradation products are not tracked within BioPAX and return to an unspecified molecule pool in the cell.

Major changes from BioPAX Level 2

Warning! The semantics of the **physicalEntity** classes have changed, but their names have not (except for the first letter, which is now in upper case). For example, **Protein** now refers to a protein (as a pool of molecules) in a state, whereas it used to refer to the base definition of the protein, as would be found in a protein sequence database. This base definition is now a sub-class of **UtilityClass**, called **EntityReference**.

The **PathwayStep** class has been moved to a new property in pathway to make pathways easier to create (you only need to create pathway step instances if you want to order parts of the pathway). Also, there is a new **BiochemicalPathwayStep** class, a subclass of **PathwayStep**, to make ordering of the biochemical processes easier.

L2 **physicalInteraction**, which stores molecular interactions from e.g. proteomics experiments, has been moved to be a child of the **Interaction** class named **MolecularInteraction**. This recognizes that it is a different type of interaction than control and conversion, which were previously children of the **physicalInteraction**.

All controlled vocabulary references now have their own class. E.g. **BioSource** references **TissueVocabulary**. This makes use of external controlled vocabularies easier. Also, the **openControlledVocabulary** class has been renamed **ControlledVocabulary**.

The **confidence** class has been renamed to **Score** to make it more general and suitable for describing genetic interactions.

Cardinality restrictions that documented required and optional properties are now specified. Documentation has been added that states which functional properties are required vs. optional.

By popular demand, all class names have been changed to the standard CamelCase and all property names to mixedCase.

Pathway Representation Abstraction Supported in BioPAX Level 3

Different pathway representation abstractions are in common use for different types of pathway information. Each abstraction is tailored to make representation of the specific type of pathway data easier. Multiple representation abstractions are supported in BioPAX Level 3. Understanding each abstraction and which classes it uses is the best way to understand how to use BioPAX.

Metabolic pathways

Metabolic pathways mostly involve biochemical reactions where protein enzymes convert small molecule reactants to small molecule products. While there are many exceptions to this general statement, the majority of metabolic pathway data in databases is covered. BioPAX Level 1 introduced support for this pathway data type.

Molecular interactions

Molecular interactions typically present in proteomics and functional genomics databases involve mainly pairwise (e.g. from yeast two-hybrid) and set (e.g. from affinity purifications) interactions between proteins (protein-protein interactions), DNA (protein-DNA interactions) and, sometimes, other molecules. Description of experimental details, such as the experiment type, is important for this pathway data type. The molecular interactions are typically known at a low level of detail i.e. we only know the molecules, but often not the binding sites or other details. BioPAX Level 2 introduced support for this pathway data type, adapted from PSI-MI (ref).

Signaling Pathways

Signaling pathways mostly involve cascades of chemical modifications on protein and other molecule to implement information transfer across the cell. An important difference between these pathways and metabolic or proteomics data is the central role of molecular states, such as protein post-translational modifications, and generic entities, such as the class of Wnt genes. Improved support of this pathway data type was introduced in BioPAX Level 3.

Gene Regulatory Networks

Gene regulatory networks are composed of regulator-target relationships involved in regulation of gene expression, such as relationships between transcription factors and the genes they regulate. Support for this pathway data type was introduced in BioPAX Level 3.

Genetic Interactions

A genetic interaction takes place when the action of one gene is

modified by one or more genes that assort independently. Genetic interactions are used extensively to map pathways in biology. Support for this pathway data type was introduced in BioPAX Level 3.

Key definitions

BioPAX **ontology**: The abstract representation of biological pathway concepts and their relationships developed by the BioPAX workgroup. This is also called the object model.

BioPAX **format**: The file format implementation of the BioPAX ontology that defines the syntax of representation for data. The BioPAX format is currently implemented only in OWL, but other implementations, such as XML Schema may be developed in the future.

OWL: Web Ontology Language. OWL is an XML-based language defined by the World Wide Web Consortium (see http://www.w3.org/TR/owl-guide/). OWL can be used to both define an ontology and to store instance data that adheres to that ontology. It is intended that the BioPAX ontology is used to validate that a set of instances provided by a user follows all BioPAX defined syntax and semantic rules. It is recommended that the BioPAX ontology be imported from its location on the biopax.org website, although it may also be defined directly within an instance data document.

BioPAX **workgroup**: Community group designing the BioPAX ontology and format.

Status of this document

This document is the final BioPAX Level 3 documentation. Comments may be sent to http://groups.google.com/group/biopax-discuss; This is an new mailing list as of 2010. Archives of the old mailing list are available here: http://www.biopax.org/mailman/private/biopax-discuss/ N.B., a subscription to the old list is still required to see these archives.

Discussion of certain topics is also on the BioPAX wiki at http://biopaxwiki.org

This document and the BioPAX Level 3 OWL file will be updated over time, based on community input. The documentation for the latest version of BioPAX Level 3 can always be found at: http://www.biopax.org/release/biopax-level3-documentation.pdf

BioPAX Namespace

The following URI is defined to be the BioPAX Level 3 namespace:

http://www.biopax.org/release/biopax-level3.owl#

This namespace (URI) will always be used to refer to the most recently released version of BioPAX; different URIs will be used for major versions of BioPAX Levels. You will often find this namespace used in conjunction with a prefix, usually **bp** in owl documents.

Document Conventions

In general, BioPAX property terms start with a lowercase character in the owl file, and in this document a property is italicized if appears as a single name in the text.

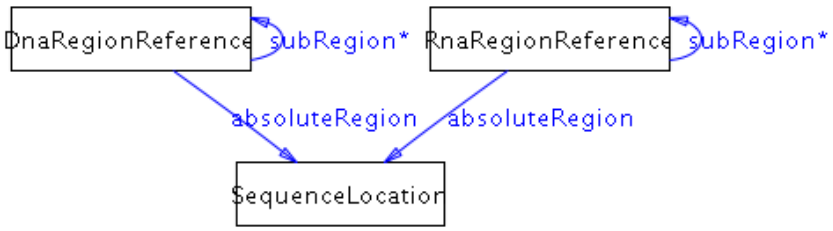
In the property lists, object properties are italicized, data type properties are not.

In general, class names start with an uppercase character in an owl file (it is now always true for the BioPAX Level3 classes), and they are highlighted in bold in the text of this document.

Throughout this document the structure "object:ClassName" refers to the object properties of a class.

The object properties diagrams display classes in boxes, and blue arcs between classes are labeled with the object property term.

e.g.



The class diagrams show the class with all its properties in a box linked with OWL Properties and objectProperties to other classes.
e.g.

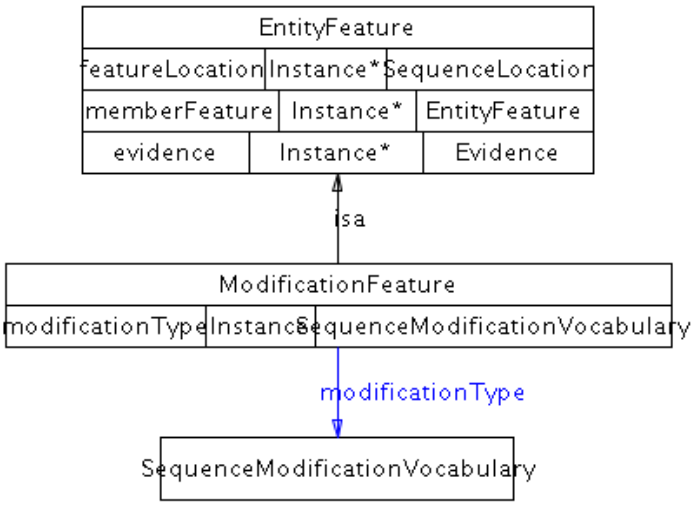


Table of Contents

1 Introduction

BioPAX (Biological Pathway Exchange) aims to facilitate the integration and exchange of data maintained in biological pathway databases. Traditionally, integrating data from a number of databases, diverse in form and content, has been a challenge in the field of Bioinformatics (. One solution is to define a mutually agreed upon file format as a standard way of representing a given type of data in a community. An example of such a standard is the DDBJ/EMBL/GenBank flat-file format, used to represent nucleic acid sequence data.

Currently, there is no other file format standard broadly applicable to biological pathway data, despite the presence of this data in over 300 different internet accessible databases. While previous work has been done to standardize specific types of pathway related data, such as the successful PSI-MI (format developed by the protein-protein interaction database community, there is no format capable of representing all of the most frequently used pathway data types (metabolic, signaling, gene regulation, molecular interaction and genetic interaction). **The goal of the BioPAX project is to provide a data exchange format for pathway data that will represent the key elements of the data models from a wide range of popular pathway databases.** To achieve this goal, the BioPAX ontology was designed to support the data models of a number of existing pathway databases, such as BioCyc (, BIND (, PATIKA (, Reactome (, aMAZE (, KEGG (, INOH (, NCI/Nature PID, PANTHER Pathways(and others. When designing the BioPAX ontology, the BioPAX workgroup endeavored to balance the many different representational needs of these and other biological pathway databases.

Because pathway data are complex and can be represented at many levels of detail, the BioPAX group is using a leveled development approach, similar to that of SBML(. While the overall framework of the BioPAX ontology, its root class structure, has been designed with the entire pathway data space in mind, representation of specific types of pathway data is the focus of individual levels. **BioPAX Level 1 was designed to represent metabolic pathway data.** Representing other types of pathway data with BioPAX Level 1 is possible but may not be optimal. **BioPAX Level 2 expanded the scope of Level 1 to include representation of molecular binding interactions and hierarchical pathways**. **BioPAX Level 3 adds support for representation of signal transduction pathways, gene regulatory networks and genetic interactions. It is intended that users use the latest version of BioPAX; i.e. Level 3 supersedes all the previous levels.**

The BioPAX language is used to model biological pathways that are typically represented in databases, the literature and textbooks. Other aspects of Biological pathways such as visualization, simulation and quantitative aspects are not considered in BioPAX. Other standardization efforts, such as SBGN, SBML and CellML and a growing software toolset supporting the remainder of the biological process space cover these aspects. Detailed information with regard to compatible pathway models can be found in Appendix C. giving more detail about mapping and exchange between BioPAX and other formats.

How to Participate

Since a data exchange format is only useful if it is widely adopted, the BioPAX project aims to promote the use of the BioPAX format by as many data sources and consumers as possible. This is achieved through community outreach at conferences and workshops and active participation in the project by data providers and consumers.

We encourage participation in BioPAX! You can help by promoting use of the format, encouraging participation by others, contributing to BioPAX discussions on mailing lists, reviewing BioPAX documents, providing data in the BioPAX format, developing software tools that support the BioPAX format, providing sponsorship for BioPAX activities, participating directly in its design.

BioPAX participation is currently on a volunteer basis and members have typically paid their own expenses. The US Department of Energy (DOE), Japan's JST and the NIH have provided funding to support workshops. BioPAX development is also supported by Award Number P41HG004118 from the National Human Genome Research Institute.

More details are available on www.biopax.org.

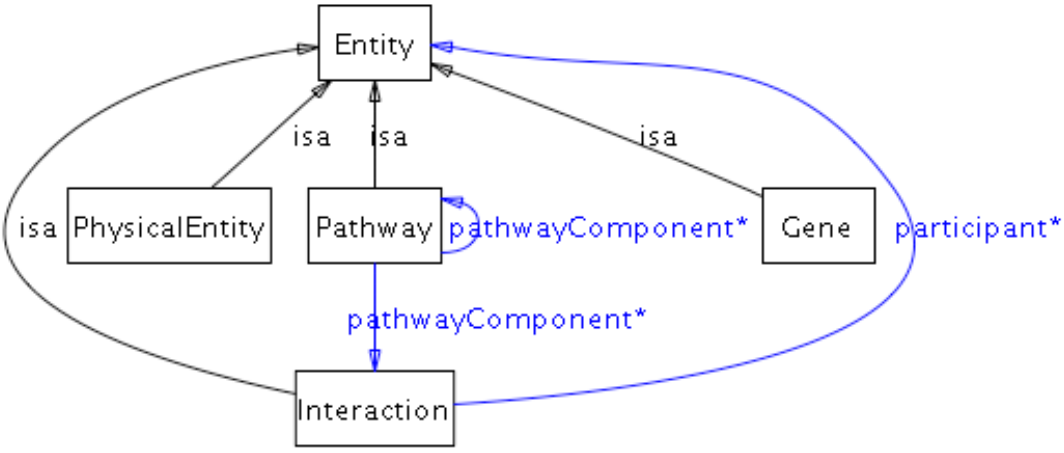
2 BioPAX Ontology Class Structure

This section provides an overview of the BioPAX Level 3 class structure. Formal definitions are found in the BioPAX Level 3 OWL document (http://www.biopax.org/release/biopax-level3.owl). Text definitions of classes are provided along with synonyms, comments and examples, where possible, to help the reader understand the definition and intended use of each class. If a value is not specified in a property, it is considered unknown. The most specific class available should be used. Effort is made to write clear and concise documentation, however, exact semantics may not always be captured in the class and property documentation. Questions should be directed to the biopax-discuss (http://groups.google.com/group/biopax-discuss) mailing list.

Interspersed throughout this section are object property diagrams that show the class and relationships (object properties) between it and other classes in the BioPAX ontology. These were created using the Ontoviz plugin for Protégé. An asterisk (*) next to a property indicates that multiple values are allowed. The sub-property feature of OWL is similar to the subclass feature. A property may have sub-properties. Any class that contains a parent property also contains the sub-properties. Also, any value of a sub-property will also be a value for the property. For example, the name property has sub-properties for different types of names, such as *displayName*. Whenever name is present in a class, *displayName* is also present. Any *displayName* values will also be in the name property.

Top level entity class

The BioPAX ontology defines 5 basic classes: the root level **Entity** class and its four subclasses: **Pathway, Interaction, PhysicalEntity** and **Gene**.



A pathway is a set of interactions or pathways. An interaction is a set of entities. This means that at the top level of BioPAX, any entities can interact. Examples include a protein-protein molecular interaction and small molecule inhibiting a pathway. Specific types of interactions are defined as children of the Interaction class. Specific types of pathways are not defined as classes.

Analogies of the root BioPAX ontology structure (first and second level classes) to other conceptual areas.				
	Linguistic	Graph representation	Pathway shorthand	Top Level Ontology
PhysicalEntity or **Gene**	Noun (Subject or Object)	Node	A, B, C	Continuant
relationship (object property)	Verb	Edge	→, →	Mediating
Interaction	Phrase/Sentence	Hyperedge, with node labels within context of hyperedge.	A→B, B→C	Occurrent
Pathway	Paragraph	Graph	A→B→C	Occurrent

Table 1 BioPAX root classes in context.

Entity (ontology root class)

Definition: A discrete biological unit used when describing pathways.
Comment: This is the root class for all biological classes in the

ontology, which include pathways, interactions, physical entities and genes.
Synonyms: thing, object, bioentity.
Properties: availability*,* comment*, dataSource, evidence,* name*, xref*

availability - Describes the availability of this data (e.g. a copyright statement). The availability statement applies to the instance it is attached to and all children instances. For example, the availability statement on a **Pathway** instance applies to the pathway and all elements of the pathway (proteins, evidence, xrefs).

comment - Comment on the data in the container class. This property should be used instead of the OWL documentation elements (rdfs:comment), as OWL metadata properties are not required to be recognized by BioPAX tools.

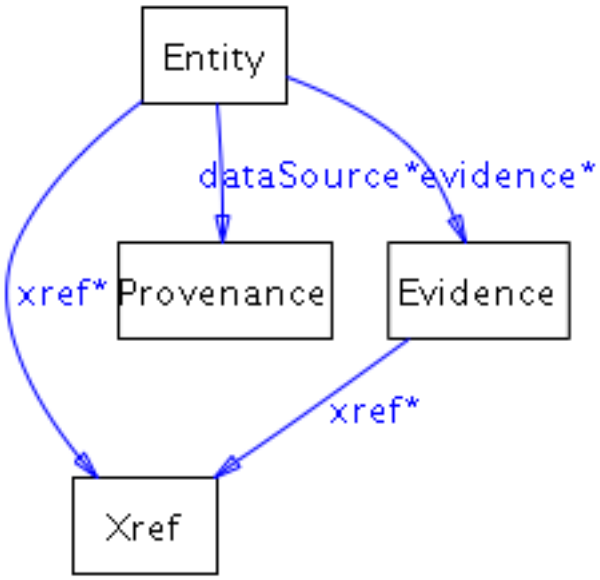
dataSource - (0 or 1 Provenanceobject:Provenance) A description of the source of this data, e.g. a database or person name. This property should be used to describe the source of the data by database groups that export their data to the BioPAX format or by systems that are integrating data from multiple sources. Similar to the availability property, the data source applies to the instance it is attached to and all children instances. This property reports the last data source, not all data sources that the data has passed through from creation. Further described in the **Provenance** class documentation.

evidence - (0 or 1 object:Evidence) Scientific evidence supporting the existence of the entity. Further described in the **Evidence** class documentation.

name - One or more names of this entity. This will automatically include values of the *displayName* and *standardName* properties, as they are child properties of the name property. *displayName* values are short names suitable for display in a graphic. Standard names are names that follow a standard nomenclature, like systematic yeast ORF names (e.g. YJL034W).

xref - Values of this property define external cross-references from this entity to entities in external databases. Further described in the Xref class documentation.

Object Properties Diagram:



Entity subclasses

Pathway

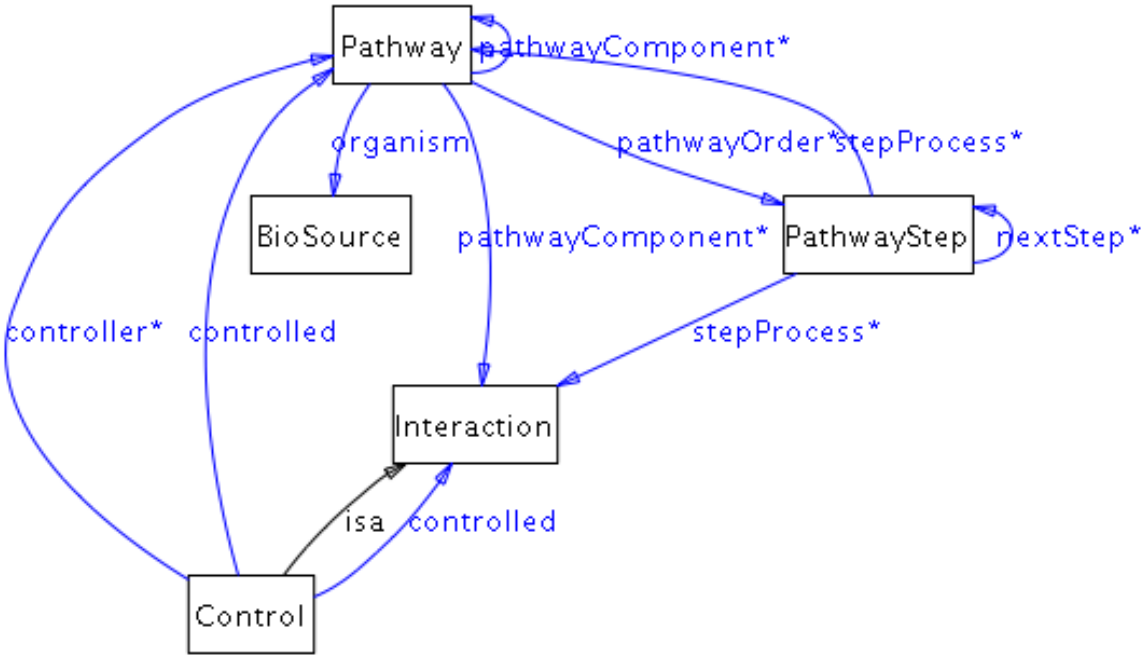
Definition: A set or series of interactions, often forming a network, which biologists have found useful to group together for organizational, historic, biophysical or other reasons. Pathways can also contain sub-pathways.
Comment: It is possible to define a pathway without specifying the interactions within the pathway. In this case, the pathway instance could consist simply of a name and could be treated as a 'black box'. In BioPAX, a pathway is defined using interactions and/or pathway instances. This provides sufficient flexibility to support two main representation conventions, the typical biochemical pathway (a process containing events), composed of a set of interactions, possibly ordered by **PathwayStep**s and possibly defined using sub-pathways, and the typical molecular interaction network (containing relationships that are not events), composed of a set of interactions not involving pathway steps or sub-pathways. Often, molecular interaction datasets do not contain any notion of a pathway, but instead simply store a collection of binary or higher order interactions between molecules. For these datasets, instances of the pathway class are not necessary, though may be used to store sub-networks of the overall interaction network that are part of a pathway.
Synonyms: network
Examples: glycolysis, the EGFR signaling pathway, the cell cycle
Parent class: Entity
Properties: *organism*, *pathwayComponent*, *pathwayOrder*, availability, comment, *dataSource*, *evidence*, name, *xref*

organism - (0 or 1 object:BioSource) An organism, e.g. 'Homo sapiens'. This is the organism that the pathway is found in. A pathway may not have an organism associated with it, for instance, reference pathways from KEGG. Multi-organism pathways must specify organism on the component physical entities (within interactions) and pathways, instead of using this property. This avoids ambiguity between a pathway that spans multiple organisms versus a generic one present in both organisms.

pathwayComponent – (0 or more object:Interaction or object:Pathway) The set of interactions or sub-pathways in this pathway. The *pathwayComponent* property may be left empty, in which case the pathway would simply have a name and could be treated as a black box (doing this defines a black box pathway).

pathwayOrder - (0 or more object:PathwayStep) The ordering of components (interactions and pathways) in the context of this pathway. This is useful to specify circular or branched pathways, or pathway order when component biochemical reactions are reversible, but are directed in the context of this pathway. Each instance of the **PathwayStep** class, referenced by *pathwayOrder*, defines: 1) a set of interactions that together define a particular step in the pathway for example, a **Catalysis** instance and the **Conversion** that it catalyzes; 2) an order relationship to one or more other pathway steps (via the *nextStep* property). If this property is used, it is still necessary to specify pathway components in the *pathwayComponent* property (even though all pathway components would also be listed in the set of *pathwayOrder* properties). A **PathwayStep** should not be listed in the *nextStep* property of another **PathwayStep** if the intersection of the entities in the *participant* properties of their interactions is empty. Typically, at least one product of the conversion in each preceding pathway step should participate either as a *controller* or as a substrate to the conversion interaction of a pathway step. Holes in the pathway are allowed, for instance if intermediate steps are not known. The *nextStep* property is meant only to represent pathway topology, not order of events. Temporal ordering information should only be inferred from the direction of each interaction within the pathway (see section on **conversionDirection** in BiochemicalReaction).

Object Properties Diagram:



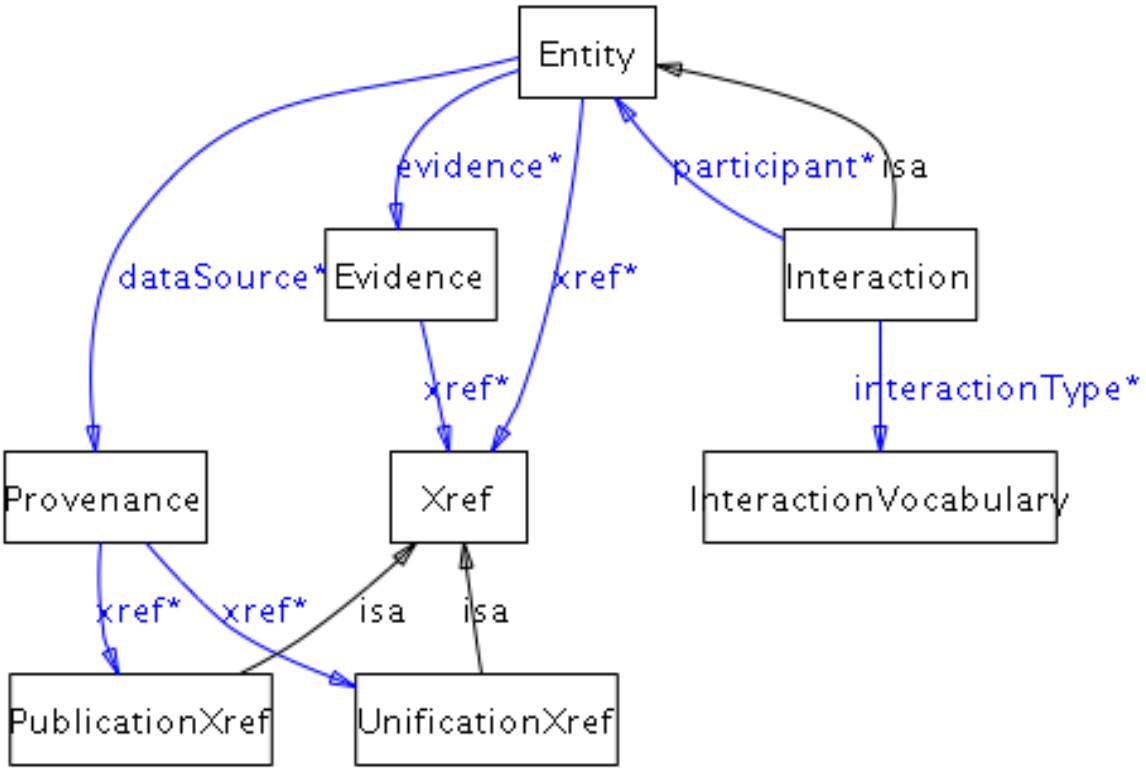
Interaction

Definition: A biological relationship between two or more entities. An interaction is defined by the entities it relates.
Comment: In BioPAX, interactions are atomic from a database modeling perspective, i.e. interactions can not be decomposed into sub-interactions. When representing non-atomic continuants with explicit sub events the pathway class should be used instead. Interactions are not necessarily temporally atomic, for example genetic interactions cover a large span of time. Interactions as a formal concept is a continuant, it retains its identity regardless of time, or any differences in specific states or properties.
Usage note: Interaction is a highly abstract class and in almost all cases it is more appropriate to use one of the subclasses of interaction. It is partially possible to define generic reactions by using generic participants. A more comprehensive method is planned for BioPAX L4 for covering all generic cases like oxidization of a generic alcohol.
Synonyms: relationship, link
Examples: protein-protein interaction, biochemical reaction, enzyme catalysis
Subclasses: **Control**, **Conversion**, **GeneticInteraction**, **MolecularInteraction**, **TemplateReaction**
Parent class: Entity
Properties: *interactionType, participant*, availability, comment, *dataSource*, *evidence*, name, *xref*

interactionType - (0 or more object:InteractionVocabulary) Controlled vocabulary annotating the interaction type for example, "phosphorylation reaction". This annotation is meant to be human readable and may not be suitable for computing tasks, like reasoning, that require formal vocabulary systems. For instance, this information would be useful for display on a web page or for querying a database. The PSI-MI interaction type controlled vocabulary should be used. This is browsable at: http://www.ebi.ac.uk/ontology-lookup/browse.do?ontName=MI&termId=MI%3A0190&termName=interaction%20type

participant - (0 or more object:**Entity**) The entities that participate in this interaction. For example, in a biochemical reaction, the participants are the union of the reactants and the products of the reaction. Multiple sub-properties of participant are defined, such as *left* and *right* used in the **BiochemicalReaction** class and *controller* and *controlled*, used in the **Control** class. Any value of the sub-properties is automatically values of the participant property.

Object Properties Diagram:



PhysicalEntity

Definition: A pool of entities, where each entity has a physical structure. An instance of this class does not represent a specific

molecular instance of an entity in a cell. The pool can be a set of identical molecules or a grouping of molecular pools, in which case it is generic. The pool represents a state of a physical entity. Constant aspects of the set of states (e.g. protein sequence) are stored in a subclass of **EntityReference**, which is referenced from the (subclass of) **PhysicalEntity**. The variable aspects that define the state are stored in the features of the physical entity. Generic physical entities can be defined for specific types of physical entities using the **EntityReference** object referenced by the physical entity. For instance, generic proteins, like a protein family, can be defined.
Synonyms: part, interactor, object
Naming rationale: **PhysicalEntity** is a compromise between more specific and more general terms that encompass all of the subclasses of this class. Other names for this class in use by other groups: PSI-MI uses ‘interactor’, BIND uses ‘object’, BioCyc uses ‘chemicals’.
Examples: protein, small molecule, RNA, DNA region
Subclasses: **Complex, DNA, Protein, RNA, SmallMolecule, DnaRegion, RnaRegion**
Parent class: Entity
Properties: *cellularLocation, feature, memberPhysicalEntity, notFeature,* availability*,* comment*, dataSource, evidence,* name*, xref*

cellularLocation - (0 or 1 object:CellularLocationVocabulary) A cellular location, e.g. 'cytoplasm'. This should reference a term in the Gene Ontology Cellular Component ontology. A molecule in two different cellular compartments is in two different states (one for each compartment).

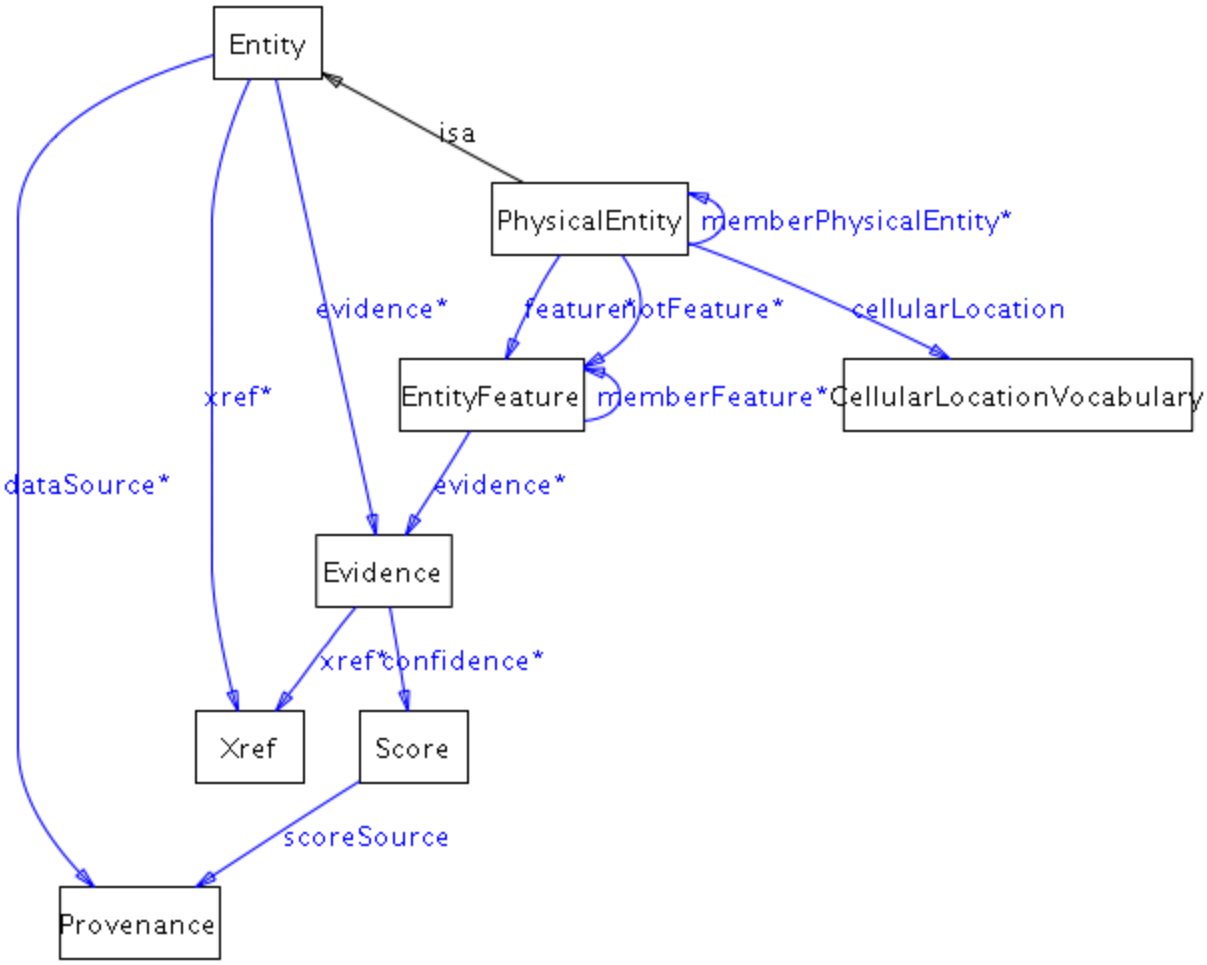
feature - (0 or more object:EntityFeature) Features of the owner physical entity. For example, binding sites, post-translational modifications on proteins, or sequence fragments. Only features relevant to the interaction are captured. A set of features on a **PhysicalEntity** helps define the state of that entity.

memberPhysicalEntity – (0 or more object:PhysicalEntity) **Note: Use of this property is not recommended. It is only defined to support legacy data in certain databases.** Used to define a generic physical entity that is a collection of other physical entities. In general, **EntityReference** class should be used to create generic groups of physical entities, however, there are some cases where this is not possible, in which case, this property must be used. For instance, when **entity reference** is used to define a generic physical entity with generic features, the generic features of the same type must be grouped. If you do not have grouping information for features of generic physical entities, you cannot use **entity reference** to define generic physical entities and must use the **memberPhysicalEntity**

property. Another example for using this property is to create generic complexes, which are currently not supported with the **EntityReference** scheme (there is no “ComplexReference” class).

notFeature - (0 or more object:EntityFeature) Features this physical entity is known to be lacking. Features that are not specified are not known to be absent. Only features known to be lacking that are relevant to the interaction should be captured.

Object Properties Diagram:



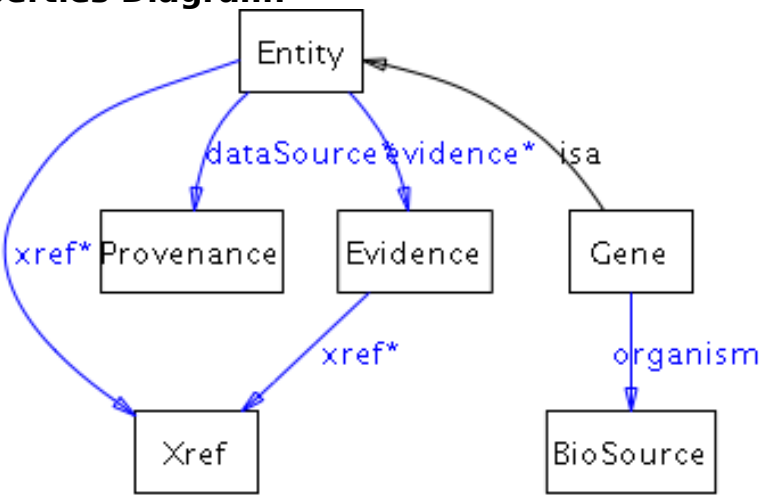
Gene

Definition: An entity that encodes information that can be inherited through replication. This is a generalization of the prokaryotic and eukaryotic notion of a gene. N.B. this is used only for genetic interactions (class **GeneticInteraction**), gene expression regulation makes use of **DNA**, **RNA**, **DnaRegion** and **RnaRegion** physical entities.
Comment: A gene is not a physical entity, but both genes and physical entities are continuants, as defined by most top level ontologies. **Gene** and **PhysicalEntity** classes are conceptually similar, though there is no continuant class in BioPAX to group them.

Examples: The BRCA1 gene
Parent class: Entity
Properties: *organism*, availability, comment, *dataSource*, *evidence*, name, *xref*

organism - (0 or 1 object:BioSource) An organism, e.g. 'Homo sapiens'. This is the organism that the gene is found in.

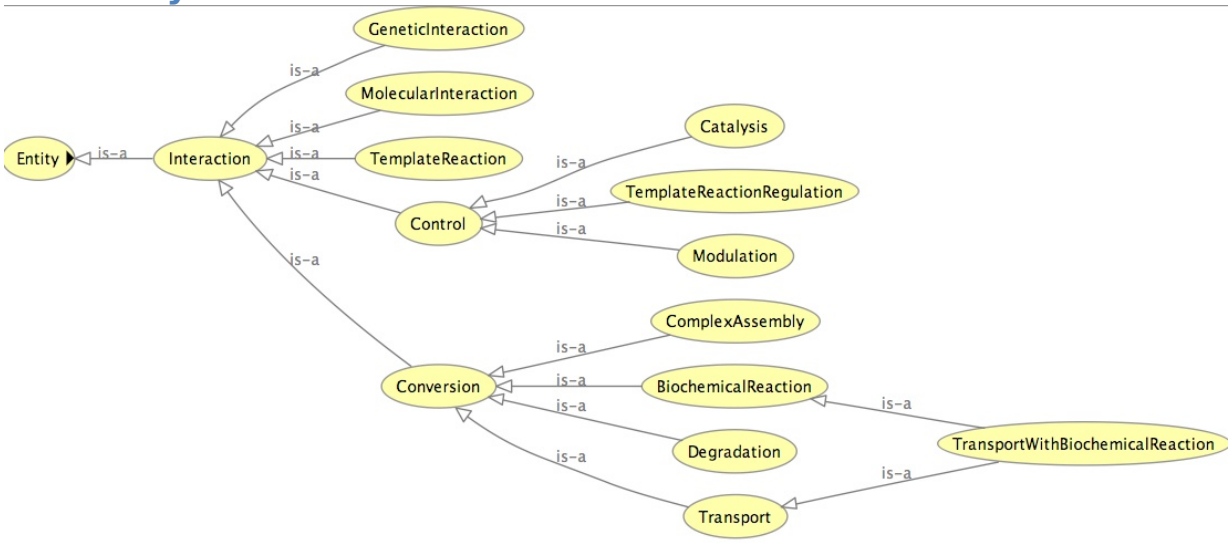
Object Properties Diagram:



Interaction subclasses

The interaction class has five subclasses: **Control**, **Conversion**, **GeneticInteraction**, **MolecularInteraction** and **TemplateReaction**.

Summary of Interaction Class Structure



Control

Definition: An interaction in which one entity regulates, modifies, or otherwise influences another. Two types of control interactions are defined: activation and inhibition.
Comment: The targets of control processes (i.e. values of the

controlled property) should be **Interactions** or **Pathways**, not physical entities. The physical entities are involved in processes, which are controlled. The physical entities are not themselves controlled. For example, a kinase activating a protein is a frequent event in signaling pathways and is usually represented in signaling diagrams using an 'activation' arrow from the kinase to the substrate. The problem with this is that the substrate may not be active in other contexts. For this reason, BioPAX does not support these types of control or activation flow networks. In BioPAX, this information should be captured as the kinase catalyzing (via an instance of the **Catalysis** class) a reaction in which the substrate is phosphorylated.
Synonyms: regulation, mediation
Examples: A small molecule that inhibits a pathway by an unknown mechanism controls the pathway.
Subclasses: **Catalysis, Modulation, TemplateReactionRegulation**
Parent Class: Interaction
Properties: *controlled, controller, controlType,* availability*,* comment*, dataSource, evidence, interactionType,* name*, participant, xref*

controlType – (0 or 1) Defines the nature of the control relationship between the CONTROLLER and the CONTROLLED entities.

The following terms are possible values:

ACTIVATION: General activation. Compounds that activate the specified enzyme activity by an unknown mechanism. The mechanism is defined as unknown, because either the mechanism has yet to be elucidated in the experimental literature, or the paper(s) curated thus far do not define the mechanism, and a full literature search has yet to be performed.

The following term is not used in the catalysis class:
INHIBITION: General inhibition. Compounds that inhibit the specified enzyme activity by an unknown mechanism. The mechanism is defined as unknown, because either the mechanism has yet to be elucidated in the experimental literature, or the paper(s) curated thus far do not define the mechanism, and a full literature search has yet to be performed.

The following terms can only be used in the modulation class (these definitions from EcoCyc):
INHIBITION-ALLOSTERIC
Allosteric inhibitors decrease the specified enzyme activity by binding reversibly to the enzyme and inducing a conformational change that decreases the affinity of the enzyme to its

substrates without affecting its VMAX. Allosteric inhibitors can be competitive or noncompetitive inhibitors, therefore, those inhibition categories can be used in conjunction with this category.

INHIBITION-COMPETITIVE
Competitive inhibitors are compounds that competitively inhibit the specified enzyme activity by binding reversibly to the enzyme and preventing the substrate from binding. Binding of the inhibitor and substrate are mutually exclusive because it is assumed that the inhibitor and substrate can both bind only to the free enzyme. A competitive inhibitor can either bind to the active site of the enzyme, directly excluding the substrate from binding there, or it can bind to another site on the enzyme, altering the conformation of the enzyme such that the substrate cannot bind to the active site.

INHIBITION-IRREVERSIBLE
Irreversible inhibitors are compounds that irreversibly inhibit the specified enzyme activity by binding to the enzyme and dissociating so slowly that it is considered irreversible. For example, alkylating agents, such as iodoacetamide, irreversibly inhibit the catalytic activity of some enzymes by modifying cysteine side chains.

INHIBITION-NONCOMPETITIVE
Noncompetitive inhibitors are compounds that noncompetitively inhibit the specified enzyme by binding reversibly to both the free enzyme and to the enzyme-substrate complex. The inhibitor and substrate may be bound to the enzyme simultaneously and do not exclude each other. However, only the enzyme-substrate complex (not the enzyme-substrate-inhibitor complex) is catalytically active.

INHIBITION-OTHER
Compounds that inhibit the specified enzyme activity by a mechanism that has been characterized, but that cannot be clearly classified as irreversible, competitive, noncompetitive, uncompetitive, or allosteric.

INHIBITION-UNCOMPETITIVE
Uncompetitive inhibitors are compounds that uncompetitively inhibit the specified enzyme activity by binding reversibly to the enzyme-substrate complex but not to the enzyme alone.

ACTIVATION-NONALLOSTERIC

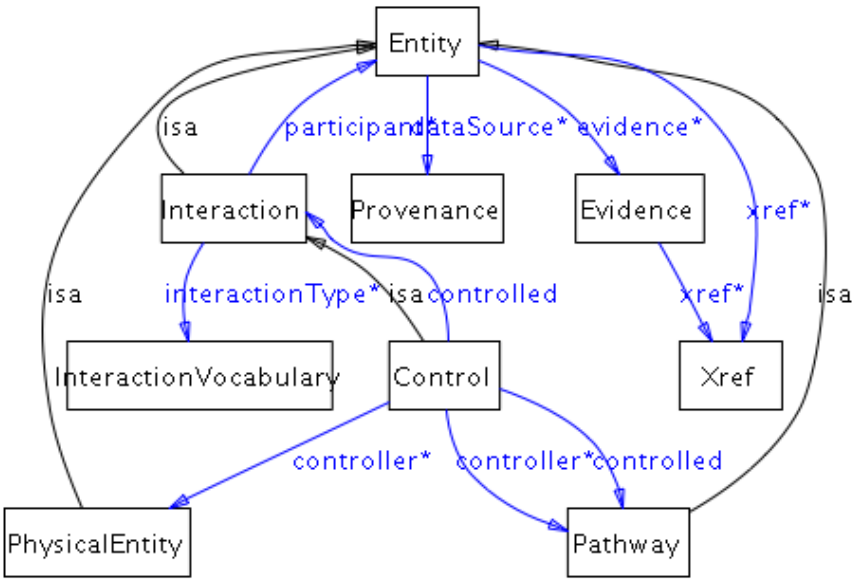
Nonallosteric activators increase the specified enzyme activity by means other than allosteric.

ACTIVATION-ALLOSTERIC
Allosteric activators increase the specified enzyme activity by binding reversibly to the enzyme and inducing a conformational change that increases the affinity of the enzyme to its substrates without affecting its VMAX.

controlled - (0 or 1 object:Interaction or object:Pathway) The entity that is controlled, e.g., in a biochemical reaction, the reaction is controlled by an enzyme. Interactions or pathways can be controlled by physical entities (e.g. small molecule activates a pathway, protein enzyme catalyzes a reaction). *controlled* is a sub-property of *participant*.

controller - (0 or more object:**PhysicalEntity** or object:Pathway) In for example, a biochemical reaction, an enzyme is the controlling entity of the reaction. Physical entities pathways can be controllers (e.g. protein enzyme catalyzes a reaction, a pathway activates a reaction, caspase cascade activates apoptosis). Multiple controllers are all required for the control to occur (AND relationship). OR relationships are defined using multiple control interaction instances. *controller* is a sub-property of *participant*.

Object Properties Diagram:



Conversion

Definition: An interaction in which one or more physical entities is

physically transformed into one or more other ones. A conversion is often paired with a relevant **Control** class to create a control-conversion pair. For example, a **BiochemicalReaction** conversion is often controlled by a **Catalysis** instance. **Degradation** is often catalyzed as well. A **Catalysis** would also control an active transport, but not a passive transport conversion. **ComplexAssembly** conversions are usually not controlled by other entities (i.e. they are spontaneous).

Comment: This class is designed to represent a simple, single-step transformation, as commonly found in textbook biochemical pathways. Multi-step transformations, such as the conversion of glucose to pyruvate in the glycolysis pathway, should be represented as pathways, if known.
Usage Note: Subclasses of conversion represent different types of transformation reflected by the properties of different **physicalEntity**. **BiochemicalReactions** will change the **ModificationFeatures** on a **PhysicalEntity** , Transport will change the **CellularLocation** and **ComplexAssembly** will change **BindingFeatures**. Generic Conversion class should only be used when the modification does not fit into one of these classes.
Examples: A biochemical reaction converts substrates to products, the process of complex assembly converts single molecules to a complex, transport converts entities in one compartment to the same entities in another compartment.
Subclasses: **BiochemicalReaction, ComplexAssembly, Degradation, Transport**
Parent Class: Interaction
Properties: conversionDirection, *left*, *right , participant*, *participantStoichiometry*, , spontaneous, availability, comment, *dataSource*, *evidence*, *interactionType*, name, *xref*

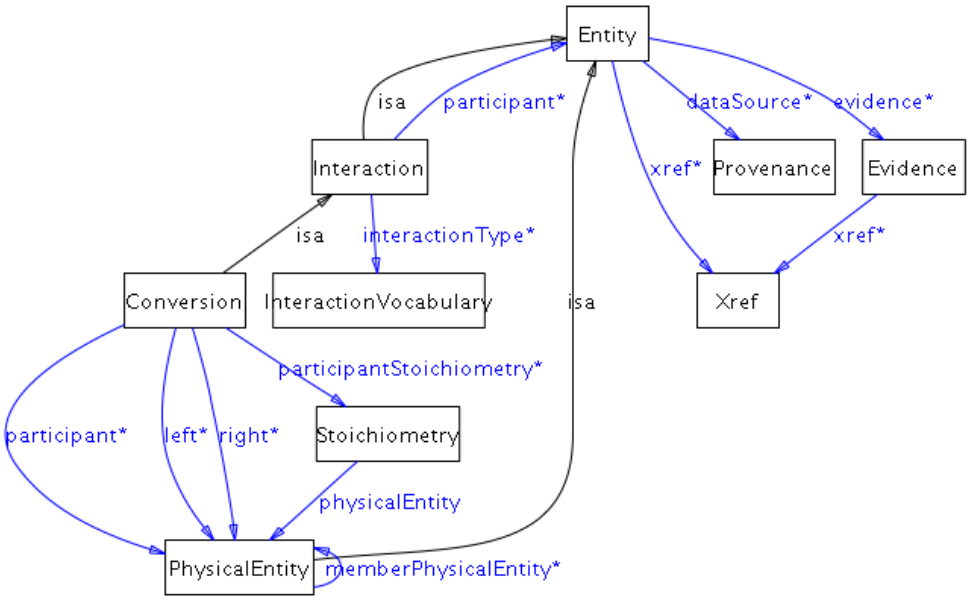
left - (0 or more object:PhysicalEntity) The participants on the left side of the conversion interaction. Since conversion interactions may proceed in either the left-to-right or right-to-left direction, occupants of the *left* property may be either reactants or products. *left* is a sub-property of *participant*.

right - (0 or more object:PhysicalEntity) The participants on the right side of the conversion interaction. Since conversion interactions may proceed in either the left-to-right or right-to-left direction, occupants of the *right* property may be either reactants or products. *right* is a sub-property of *participant*.

participantStoichiometry - (0 or more object:Stoichiometry) Stoichiometry of the left and right participants.

spontaneous - Specifies whether a conversion occurs spontaneously. If the spontaneity is not known, the *spontaneous* property should be left empty.

Object Properties Diagram:



GeneticInteraction

Definition: Genetic interactions between genes occur when two genetic perturbations (e.g. mutations) have a combined phenotypic effect not caused by either perturbation alone. A gene participant in a genetic interaction represents the gene that is perturbed. Genetic interactions are not physical interactions but logical (AND) relationships. Their physical manifestations can be complex and span an arbitarily long duration.
Comment: There is a large body of interaction data, mostly produced by high throughput systems, that does not satisfy the level of detail required to model them with ComplexAssembly class. Specifically, what is lacking is the stoichiometric information and completeness (closed-world) of participants required to model them as chemical processes. Nevertheless interaction data is extremely useful and can be captured in BioPAX using this class.
Example: A synthetic lethal interaction occurs when cell growth is possible without either gene A OR B, but not without both gene A AND B. If you knock out A and B together, the cell will die.
Parent class: **Interaction**
Properties: *interactionScore*, *interactionType*, *participant*, *phenotype*, availability, comment, *dataSource*, *evidence*, name, *xref*

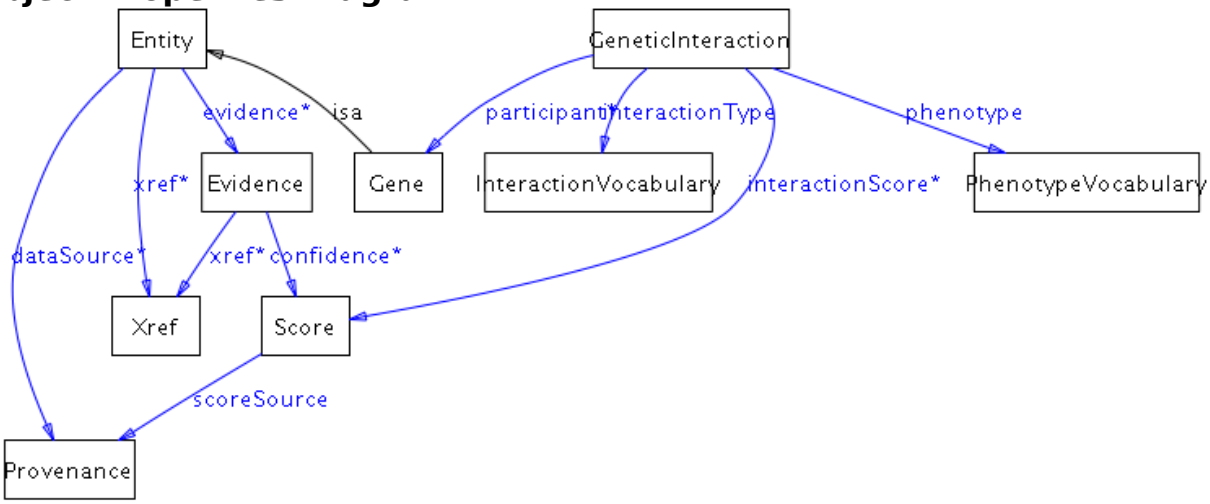
participant - (at least 2 object:Gene) Lists the entities that participate in this Genetic Interaction.

interactionType - (0 or 1 and only 1object:InteractionVocabulary*)*

interactionScore - (0 or more object:Score) The score of an interaction e.g. a genetic interaction score. This is useful for quantitative genetic interactions, usually defined based on an additive or multiplicative interaction model.

phenotype - (at least 1 object:PhenotypeVocabulary) The phenotype quality used to define this genetic interaction e.g. viability. Different genetic interactions in the same system can be revealed by using different phenotypes.

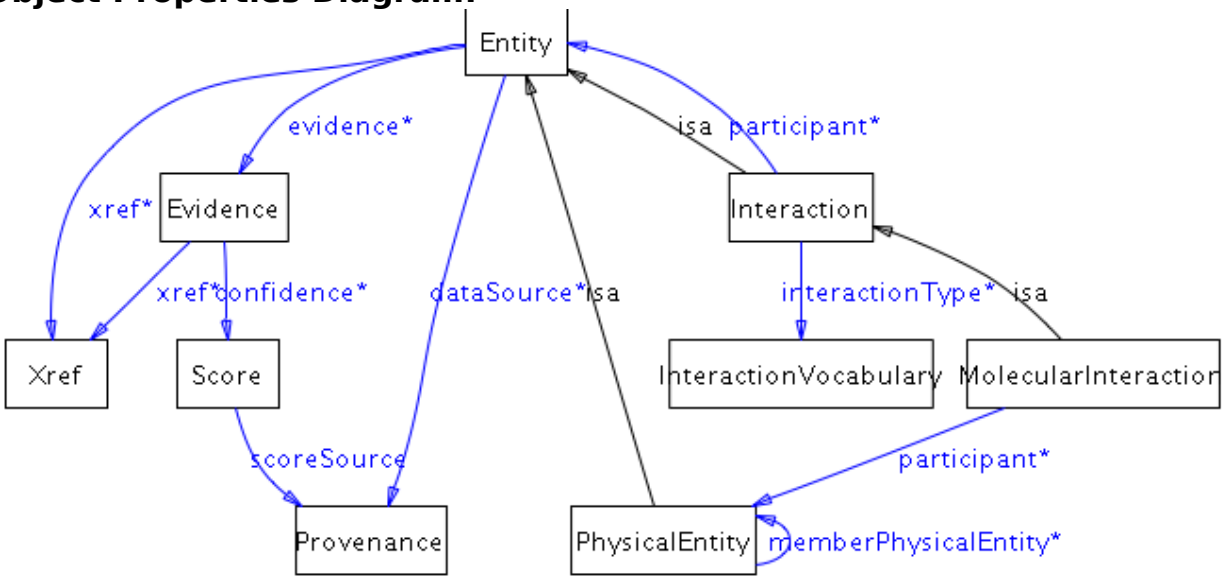
Object Properties Diagram:



MolecularInteraction

Definition: An interaction involving molecular contact between physical entities. The exact mechanism may not be known.
Comment: There is a large body of interaction data, mostly produced by high throughput systems, that does not satisfy the level of detail required to model them with **ComplexAssembly** class. Specifically what is lacking, is the stoichiometric information and completeness (closed-world) of participants required to model them as chemical processes. This class should be used by default for representing molecular interactions such as those defined by PSI-MI level 2.5. The participants in a molecular interaction should be listed in the *participant* object property. Note that this is one of the cases in which the *participant* property should be directly populated with instances (see comments on the *participant* property in the **Interaction** class description). If sufficient information on the nature of a molecular interaction is available, a more specific BioPAX interaction class should be used, such as **ComplexAssembly**.
Example: Two proteins observed to interact in a yeast-two-hybrid experiment where there is not enough experimental evidence to suggest that the proteins are forming a complex by themselves without any indirect involvement of other proteins. This is the case for

most large-scale yeast two-hybrid screens. Many protein-protein interaction databases contain examples of this type of interaction.
Parent class: *Interaction*
Properties: *participant*, availability, comment, *dataSource*, *evidence*, *interactionType*, name, *xref*
Object Properties Diagram:



TemplateReaction

Definition: An interaction where a macromolecule is polymerized from a template macromolecule.
Examples: DNA to RNA is transcription, RNA to protein is translation and DNA to protein is protein expression from DNA. Other examples are possible.
Comment: The template and product types determine the type of template reaction. For instance, a DNA template and a protein product is protein expression from DNA, while an RNA template and protein product is protein translation. This is a convenient abstraction of the actual template reaction process that omits all of the machinery common to most template reactions of that class. For instance, the transcription machinery is omitted for transcription. These complex processes can be described in BioPAX, using control and conversion classes, but are often not described for convenience. Also, the initiation region is part of the definition of the **TemplateReaction**. This means that, for example, two transcription events from the same gene, but with different promoters are encoded using two different **TemplateReactions**.
Usage Note: Regulation of **TemplateReaction**, e.g. via a transcription factor can be captured using **TemplateReactionRegulation**. **TemplateReaction** can also be indirect for example, it is not necessary to represent intermediary mRNA for describing expression of a protein. It was decided to not

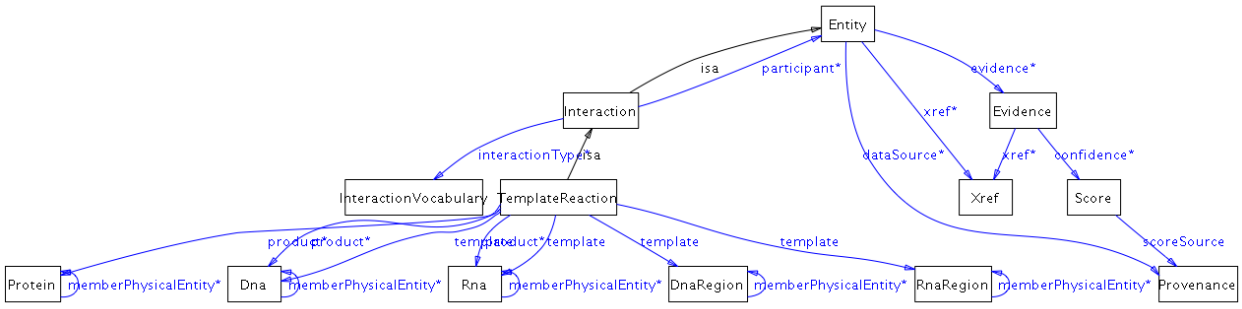
subclass **TemplateReaction** to subtypes such as transcription of translation for the sake of simplicity. If needed these subclasses can be added in the future.
Parent class: *Interaction*
Properties: *participant*, *product*, *template*, *templateDirection*, availability, comment, *dataSource*, *evidence*, *interactionType*, name, *xref*

Product - (0 or more object:DNARegion)
Definition: A region of DNA.
Comment: This is not a 'gene', it is a molecule of DNA, which could encode a gene or other things, like transcription factor binding sites.
Examples: a chromosome, a plasmid, chromosome 7 of Homo sapiens.
Parent class: *PhysicalEntity*
Properties: *entityReference*, *memberPhysicalEntity*, availability, *cellularLocation*, comment, *dataSource*, *evidence*, *feature*, name, *notFeature*, *xref*

Product - **DNARegion**The product of a template reaction, such as DNA, RNA or protein. *product* is a sub-property of *participant*.

template - (0 or 1 object:DNARegion or object:RNARegion or object:DNA or object:RNA) The template molecule that is used in this template reaction. Either DNA (e.g. eukaryotic or prokaryotic gene expression) or RNA (e.g. an RNA virus). *template* is a sub-property of *participant*.

Object Properties Diagram:



Control subclasses

Three types of control processes exist under the control class: **Catalysis**, **Modulation** and **TemplateReactionRegulation**.

Catalysis

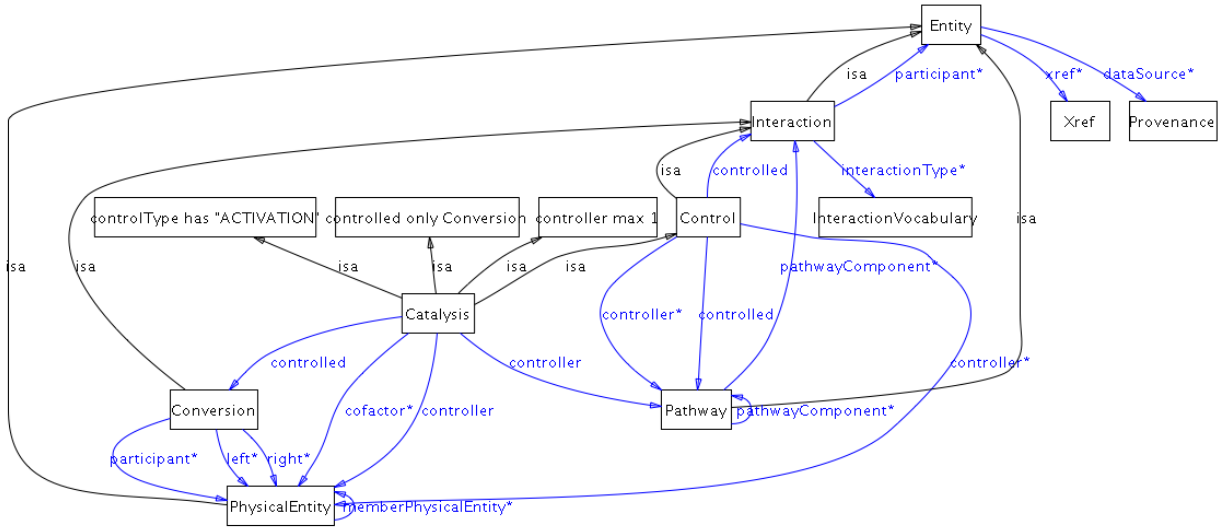
Definition: A control interaction in which a physical entity (a catalyst) increases the rate of a conversion interaction by lowering its activation energy. Instances of this class describe a pairing between a catalyzing entity and a catalyzed conversion.
Comment: A separate catalysis instance should be created for each different conversion that a **physicalEntity** may catalyze and for each different **physicalEntity** that may catalyze a conversion. For example, a bifunctional enzyme that catalyzes two different biochemical reactions would be linked to each of those biochemical reactions by two separate instances of the catalysis class. Also, catalysis reactions from multiple different organisms could be linked to the same generic biochemical reaction (a biochemical reaction is generic if it only includes small molecules, or involves generic physical entities). Generally, the enzyme catalyzing a conversion is known and the use of this class is obvious. In the cases where a catalyzed reaction is known to occur but the enzyme is not known, a catalysis instance should be created without a controller specified (i.e. the *controller* property should remain empty).

Synonyms: facilitation, acceleration.
Examples: The catalysis of a biochemical reaction by an enzyme, the enabling of a transport interaction by a membrane pore complex, and the facilitation of a complex assembly by a scaffold protein. Hexokinase -> (The “Glucose + ATP -> Glucose-6-phosphate +ADP” reaction). A plasma membrane Na+/K+ ATPase is an active transporter (antiport pump) using the energy of ATP to pump Na+ out of the cell and K+ in. Na+ from cytoplasm to extracellular space would be described in a transport instance. K+ from extracellular space to cytoplasm would be described in a transport instance. The ATPase pump would be stored in a catalysis instance controlling each of the above transport instances. A biochemical reaction that does not occur by itself under physiological conditions, but has been observed to occur in the presence of cell extract, likely via one or more unknown enzymes present in the extract, would be stored in the *controlled* property, with the *controller* property empty.
Parent class: *Control*
Properties: catalysisDirection, *cofactor*, *controlled*, *controller*, *controlType*, availability, comment, *dataSource*, *evidence*, *interactionType*, name, *participant*, *xref*
cofactor - (0 or more object:PhyicalEntity) Any cofactor(s) or coenzyme(s) required for catalysis of the conversion by the enzyme. *cofactor* is a sub-property of *participant*.

catalysisdirection - Specifies the reaction direction of the interaction catalyzed by this instance of the catalysis class. Possible values of this

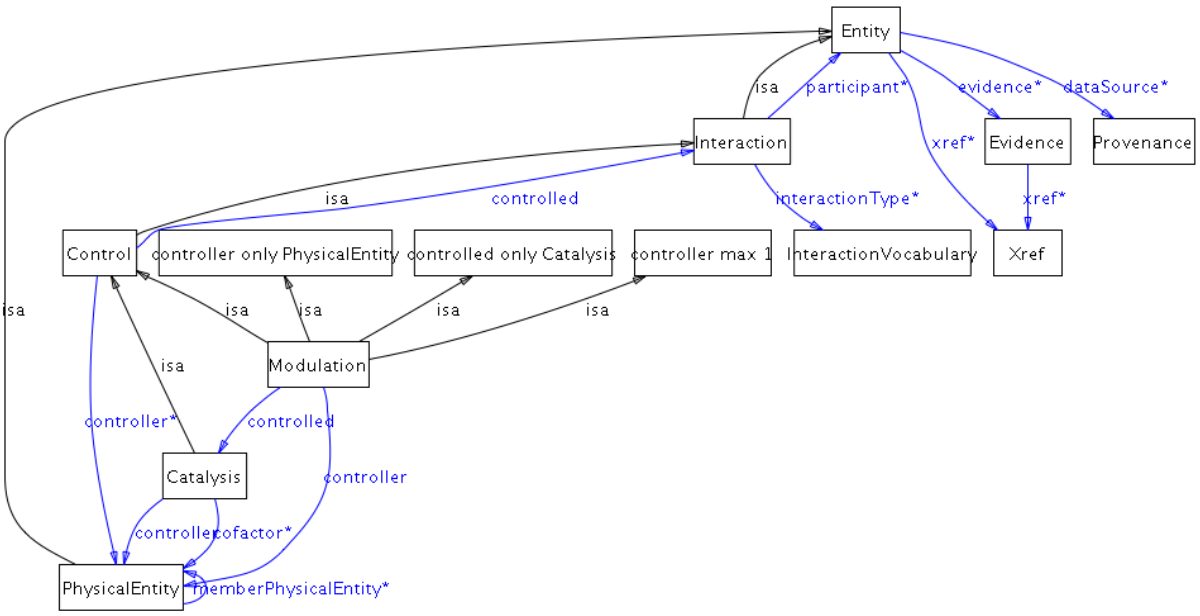
property are: REVERSIBLE: Interaction occurs in both directions in physiological settings. PHYSIOL-LEFT-TO-RIGHT PHYSIOL-RIGHT-TO-LEFT The interaction occurs in the specified direction in physiological settings, because of several possible factors including the energetics of the reaction, local concentrations of reactants and products, and the regulation or expression level of the enzyme. IRREVERSIBLE-LEFT-TO-RIGHT IRREVERSIBLE-RIGHT-TO-LEFT For all practical purposes, the interaction occurs only in the specified direction in physiological settings, because of chemical properties of the reaction. (This definition from EcoCyc)

Object Properties Diagram:



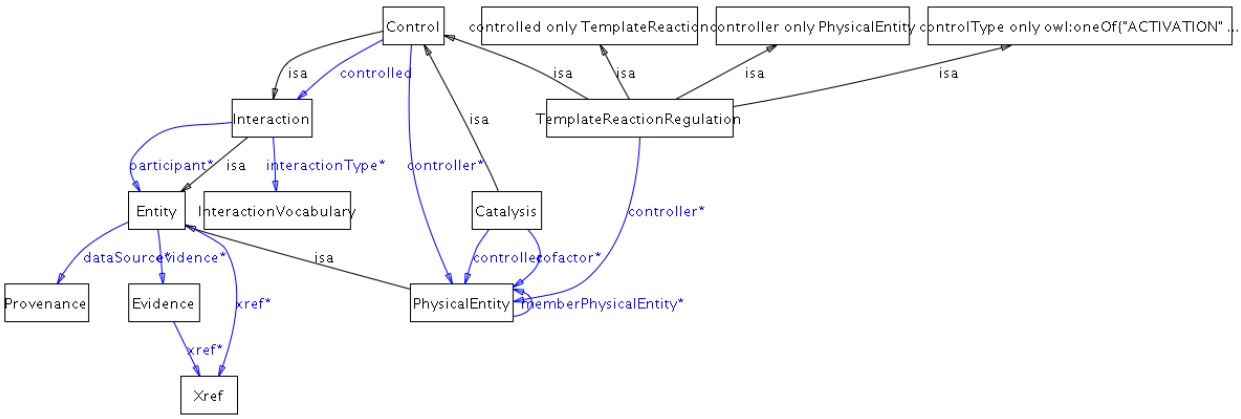
Modulation

Definition: A control interaction in which a physical entity modulates a catalysis interaction. Biologically, most modulation interactions describe an interaction in which a small molecule alters the ability of an enzyme to catalyze a specific reaction. Instances of this class describe a pairing between a modulating entity and a catalysis interaction.
Comment: A separate modulation instance should be created for each different catalysis instance that a physical entity may modulate and for each different physical entity that may modulate a catalysis instance. A typical modulation instance has a small molecule as the controller entity and a catalysis instance as the controlled entity.
Examples: Allosteric activation and competitive inhibition of an enzyme's ability to catalyze a specific reaction.
Parent class: *Control*
Properties: *controlled*, *controller*, availability, comment, *controlType*, *dataSource*, *evidence*, *interactionType*, name, *particpant*
Object Properties Diagram:



TemplateReactionRegulation

Definition: Regulation of a **TemplateReaction** by a controlling physical entity.
Examples: A transcription factor regulating transcription or gene expression. A microRNA inhibiting translation.
Comment: To represent the binding of the transcription factor to a regulatory element in the **TemplateReaction**, create a complex of the transcription factor and the regulatory element and set that as the controller. As with other control classes, multiple controllers means that they are all required for control (AND). If either of multiple controllers is required for control, they should be defined in separate **TemplateReactionRegulation** instances (OR). Control type can only be Activation or Inhibition. Controllers can only be physical entities. Controlled is always a **TemplateReaction**.
Parent class: *Control*
Properties: *controlled*, *controller*, controlType, availability, comment, *dataSource*, *evidence*, *interactionType*, name, *participant*, *xref*
Object Properties Diagram:



Conversion subclasses

Five types of conversion processes exist under the conversion class: **BiochemicalReaction**, **ComplexAssembly**, **Degradation**, **Transport** and **TransportWithBiochemicalReaction**.

BiochemicalReaction

Definition: A conversion interaction in which one or more entities (substrates) undergo covalent changes to become one or more other entities (products). The substrates of biochemical reactions are defined in terms of sums of species, thus do not have to be mass balanced. This is convention in biochemistry, and, in principle, all of the Enzyme Commission (EC) reactions should be biochemical reactions.
Examples: ATP + H_2O = ADP + P_i
Comment: In the example reaction above, ATP is considered to be an equilibrium mixture of several species, namely ATP4-, HATP3-, H2ATP2-, MgATP2-, MgHATP-, and Mg2ATP. Additional species may also need to be considered if other ions (e.g. Ca2+) that bind ATP are present. Similar considerations apply to ADP and to inorganic phosphate (Pi). When writing biochemical reactions, it is important not to attach charges to the biochemical reactants and not to include ions such as H+ and Mg2+ in the equation. The reaction is written in the direction specified by the EC nomenclature system, if applicable, regardless of the physiological direction(s) in which the reaction proceeds. Polymerization reactions involving large polymers whose structure is not explicitly captured should generally be represented as unbalanced reactions in which the monomer is consumed but the polymer remains unchanged, e.g. glycogen + glucose = glycogen.
Parent class: *Conversion*
Subclasses: *TransportWithBiochemicalReaction*
Properties: *deltaG*, deltaH, deltaS, eCNumber, *kEQ*, *participant*, availability, comment, conversionDirection, *dataSource*, *evidence*, *interactionType*, *left*, name, *participantStoichiometry*, *right*, spontaneous, *xref*

deltaG - (0 or more object:DeltaG) For biochemical reactions, this property refers to the standard transformed Gibbs energy change for a reaction written in terms of biochemical reactants (sums of species), delta-G'^{o}.

> delta-G'^{o} = -RT lnK' and delta-G'^{o} = delta-H'^{o} – T delta-S'^{o}
>
> delta-G'^{o} has units of kJ/mol. Like K', it is a function of temperature (T), ionic strength (I), pH, and pMg (pMg = -$\log_{10}[Mg^{2+}]$). Therefore, these quantities must be specified, and values for DELTA-G for biochemical reactions are represented as 5-tuples of the form (delta-G'^{o} T I pH pMg). This property may have multiple values, representing different measurements for delta-G'^{o} obtained under the different experimental conditions listed in the 5-tuple. (This definition from EcoCyc)

deltaH - For biochemical reactions, this property refers to the standard transformed enthalpy change for a reaction written in terms of biochemical reactants (sums of species), delta-H'^{o}. delta-G'^{o} = delta-H'^{o} - T delta-S'^{o} (This definition from EcoCyc)

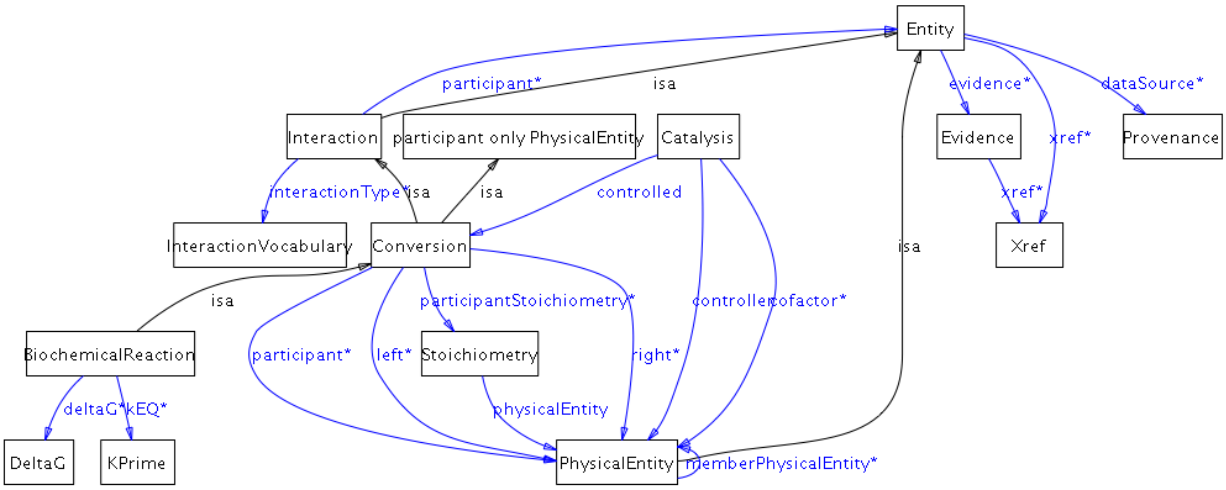
deltaS - For biochemical reactions, this property refers to the standard transformed entropy change for a reaction written in terms of biochemical reactants (sums of species), delta-S'^{o}. delta-G'^{o} = delta-H'^{o} - T delta-S'^{o} (This definition from EcoCyc)

eCNumber - The unique number assigned to a reaction by the Enzyme Commission of the International Union of Biochemistry and Molecular Biology. Note that not all biochemical reactions have EC numbers assigned to them.

kEQ – (0 or more object:kPrime) This quantity is dimensionless and is usually a single number. The measured equilibrium constant for a biochemical reaction, encoded by the property kEQ, is actually the apparent equilibrium constant, K'. Concentrations in the equilibrium constant equation refer to the total concentrations of all forms of particular biochemical reactants. For example, in the equilibrium constant equation for the biochemical reaction in which ATP is hydrolyzed to ADP and inorganic phosphate: K' = [ADP][P_i]/[ATP], The concentration of ATP refers to the total concentration of all of the following species: [ATP] = [ATP^{4-}] + [$HATP^{3-}$] + [H_2ATP^{2-}] + [$MgATP^{2-}$] + [$MgHATP^{-}$] + [Mg_2ATP]. The apparent equilibrium constant is formally dimensionless, and can be kept so by inclusion of as many of the terms (1 mol/dm^3) in the numerator or denominator as necessary. It is a function of temperature (T), ionic strength (I), pH, and pMg (pMg = -$\log_{10}[Mg^{2+}]$). Therefore, these quantities must be specified to be

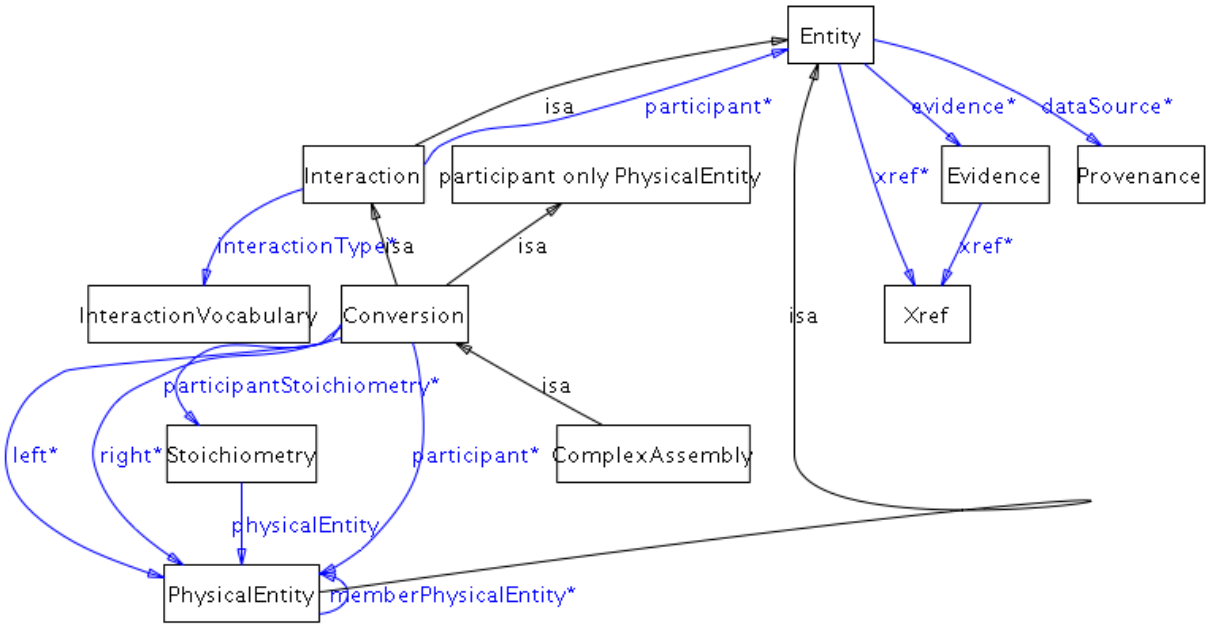
precise, and values for KEQ for biochemical reactions may be represented as 5-tuples of the form (K' T I pH pMg). This property may have multiple values, representing different measurements for K' obtained under the different experimental conditions listed in the 5-tuple. (This definition adapted from EcoCyc)

Object Properties Diagram:



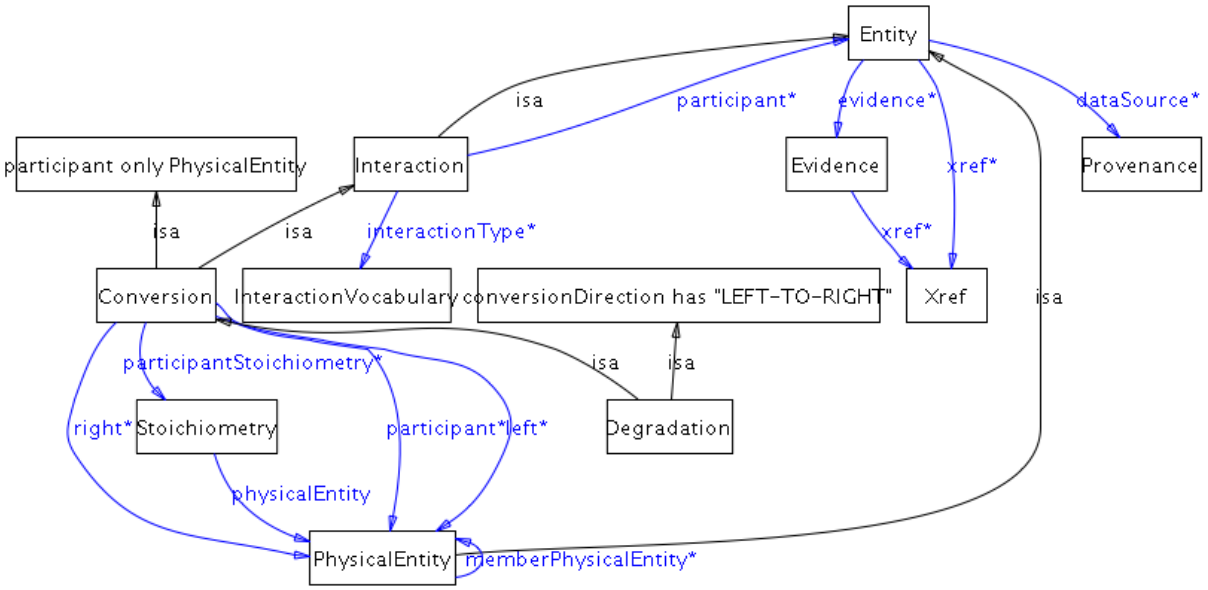
ComplexAssembly

Definition: A conversion interaction in which a set of physical entities, at least one being a macromolecule (e.g. protein, RNA, or DNA), aggregate to from a complex physicalEntity. One of the participants of a **ComplexAssembly** must be an instance of the class **Complex**. The modification of the physicalentities involved in the ComplexAssembly is captured via **BindingFeature** class.
Comment: This class is also used to represent complex disassembly. The assembly or disassembly of a complex is often a spontaneous process, in which case the direction of the **ComplexAssembly** (toward either assembly or disassembly) should be specified via the *spontaneous* property.
Synonyms: aggregation, complex formation
Examples: Assembly of the TFB2 and TFB3 proteins into the TFIIH complex, and assembly of the ribosome through aggregation of its subunits.
Note: The following are not examples of complex assembly: Covalent phosphorylation of a protein (this is a **BiochemicalReaction**); the TFIIH complex itself (this is an instance of the complex class, not the ComplexAssembly class).
Parent class: *Conversion*
Properties: *participant*, availability, comment, conversionDirection, *dataSource*, *evidence*, *interactionType*, *left*, name, *participantStiochiometry*, *right*, spontaneous, *xref*
Object Properties Diagram:



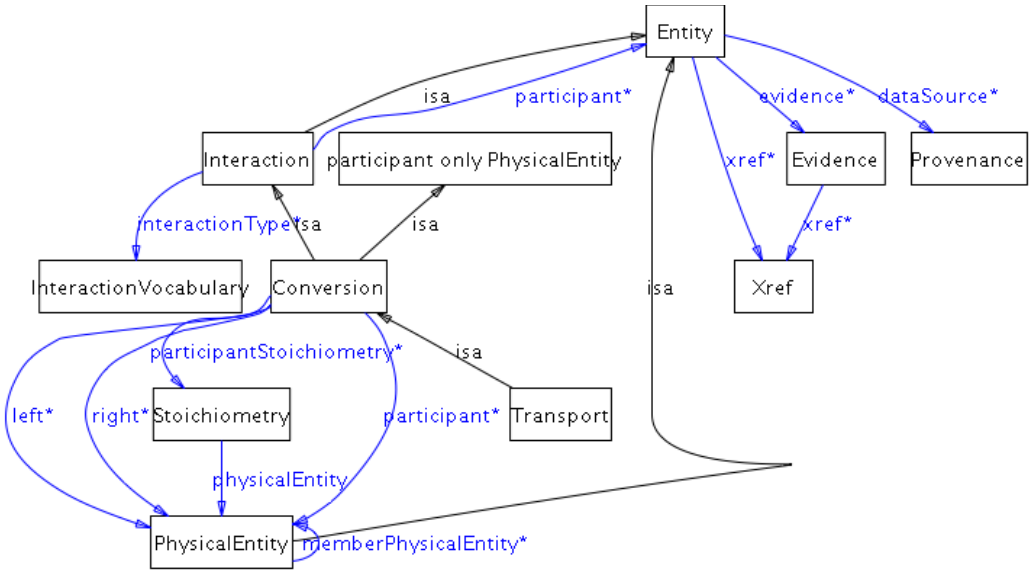
Degradation

Definition: The process of degrading a physical entity. The physical entity is converted to unspecified degraded components.
Comment: The right side of the conversion is generally not specified, indicating degraded components. Degradation spontaneous direction can only be LEFT-TO-RIGHT, NOT-SPONTANEOUS or unknown. It cannot be RIGHT-TO-LEFT. Also, any catalysis that references a degradation conversion cannot specify a RIGHT-TO-LEFT direction. Degradation usually contains nothing on the right side (we don't model degradation products). However, some cases involving complexes where part of the complex degrades, but the rest does not, like Mdm2-P53 and ubiquitin ligases that target a partner for degradation and these can have right side participants (any undegraded physical entities).
Parent class: *Conversion*
Properties: conversionDirection, *participant*, availability, comment, *dataSource*, *evidence*, *interactionType*, *left*, name, *participantStoichiometry*, *right*, spontaneous, *xref*
Object Properties Diagram:



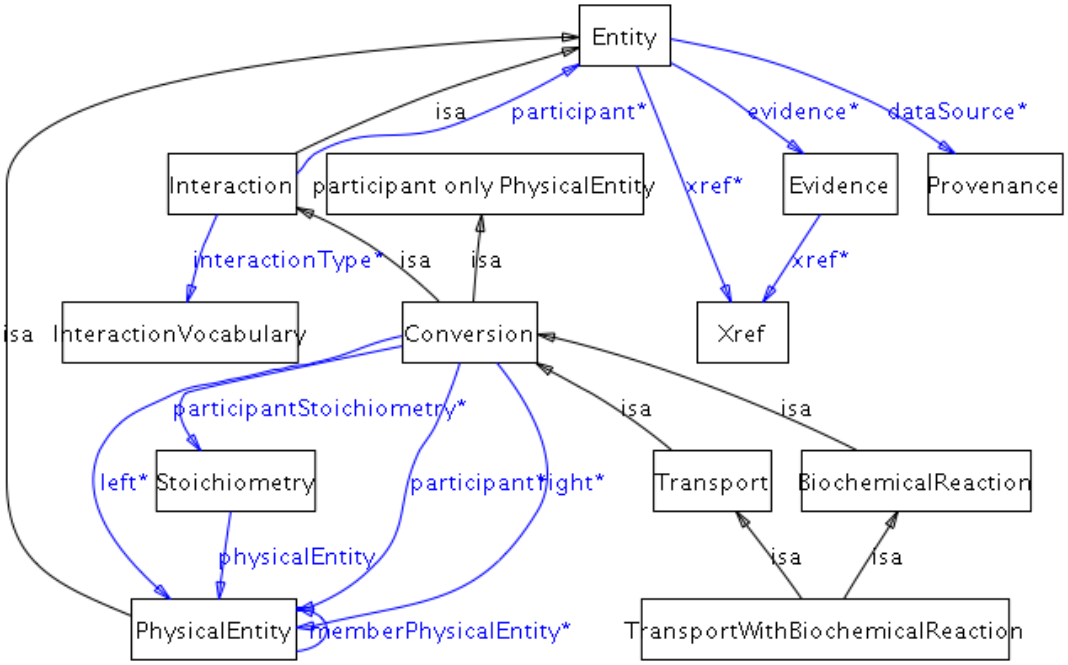
Transport

Definition: A conversion interaction in which a physical entity (or set of physical entities) changes location within or with respect to the cell.
Comment: A transport interaction does not include the transporter entity, even if one is required in order for the transport to occur. Instead, required transporters are linked to transport interactions via the catalysis class. Transport interactions do not involve chemical changes of the participant(s). These cases are handled by the TransportWithBiochemicalReaction class.
Synonyms: translocation.
Examples: The movement of Na+ into the cell through an open voltage-gated channel.
Parent class: *Conversion*
Subclasses: *TransportWithBiochemicalReaction*
Properties: *participant*, availability, comment, conversionDirection, *dataSource*, *evidence*, *interactionType*, *left*, name, *participantStoiciometry*, *right*, spontaneous, *xref*
Object Properties Diagram:

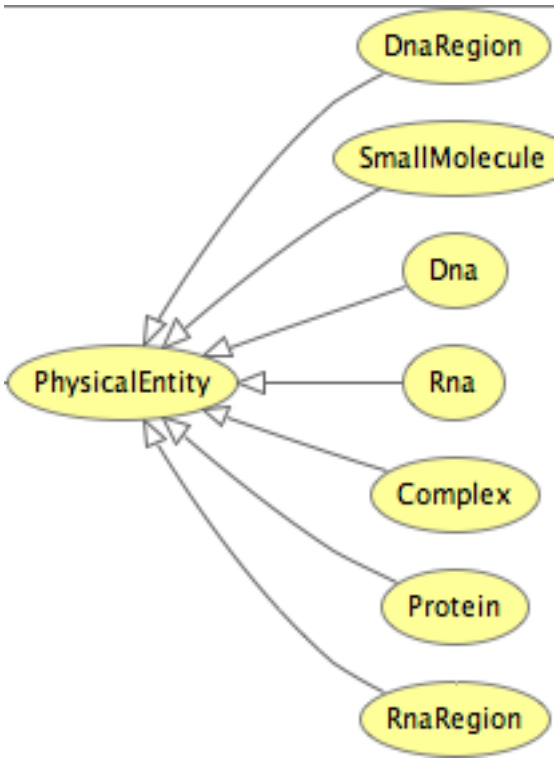


TransportWithBiochemicalReaction

Definition: A conversion interaction that is both a **BiochemicalReaction** and a Transport. In **TransportWithBiochemicalReaction** interactions, one or more of the substrates change both their location and their physical structure. Active transport reactions that use ATP as an energy source fall under this category, even if the only covalent change is the hydrolysis of ATP to ADP.
Comment: This class was added to support a large number of transport events in pathway databases that have a biochemical reaction during the transport process. It is not expected that other double inheritance subclasses will be added to the ontology at the same level as this class.
Examples: In the PEP-dependent phophotransferase system, transportation of sugar into an E. coli cell is accompanied by the sugar's phosphorylation as it crosses the plasma membrane.
Parent class: *BiochemicalReaction*
Properties: *participant*, availability, comment, conversionDirection, *dataSource*, *deltaG*, deltaH, deltaS, eCNumber, *evidence*, *interactionType*, *kEQ*, *left*, name, *partipantStoichiometry*, *right*, spontaneous, *xref*
Object Properties Diagram:



PhysicalEntity subclasses



Warning! The semantics of the physicalEntity classes have changed in BioPAX Level 3 compared to Level 2, but their names have not. For example, protein now refers to a protein in a specific state, whereas it

used to refer to the base definition of the protein, as would be found in a protein sequence database. This base definition is now a utility class, called EntityReference.

Complex

Definition: A physical entity whose structure is comprised of other physical entities bound to each other non-covalently, at least one of which is a macromolecule (e.g. protein, DNA, or RNA). Complexes must be stable enough to function as a biological unit; in general, the temporary association of an enzyme with its substrate(s) should not be considered or represented as a complex. A complex is the physical product of an interaction (ComplexAssembly) and is not itself considered an interaction. Aspects of the state of the complex, including cellular location and features, are defined here, using properties inherited from PhysicalEntity.
Comment: In general, complexes should not be defined recursively so that smaller complexes exist within larger complexes, i.e. a complex should not be a *component* of another complex (to avoid errors in interpretation - see comments on the *component* property below). Instead, the subunits should be a simple list. The boundaries on the size of complexes described by this class are not defined here, although elements of the cell as large and dynamic as, e.g., a mitochondrion would typically not be described using this class. The strength of binding among subunits is not currently described.
Examples: Ribosome, RNA polymerase II. Other examples of this class include complexes of multiple protein monomers and complexes of proteins and small molecules.
Parent class: *PhysicalEntity*
Properties: *component*, *componentStoichiometry*, *memberPhysicalEntity*, availability, *cellularLocation*, comment, *dataSource*, *evidence*, *feature*, name, *notfeature*, *xref*

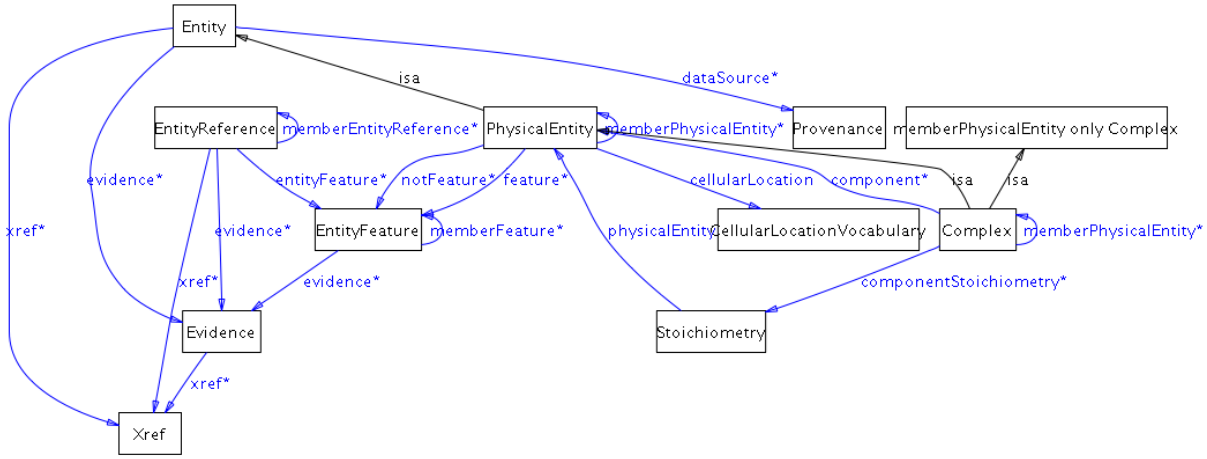
component - (0 or more object:PhysicalEntity) Defines the PhysicalEntity subunits of this complex. This property should not contain other complexes, i.e. it should always be a flat representation of the complex. For example, if two protein complexes join to form a single larger complex via a complex assembly interaction, the *component* of the new complex should be the individual proteins of the smaller complexes, not the two smaller complexes themselves. Exceptions are black-box complexes (i.e. complexes in which the *component* property is empty), which may be used as *component*'s of other complexes because their constituent parts are unknown. The reason for keeping complexes flat is to signify that there is no information stored in the way complexes are nested, such as assembly order. Otherwise, the complex assembly order may be implicitly encoded and interpreted by some users. This interpretation would lead

to errors if the assembly order were not encoded, for instance by creating hierarchical complexes randomly.

componentStoichiometry - (0 or more object:Stoichiometry) The stoichiometry of components in a complex.

memberPhysicalEntity – (0 or more object:Complex) Used to create generic complexes. Stores a set of Complex instances that define the generic complex. If this property is used, the referring complex is a generic complex.

Object Properties Diagram:



DNA

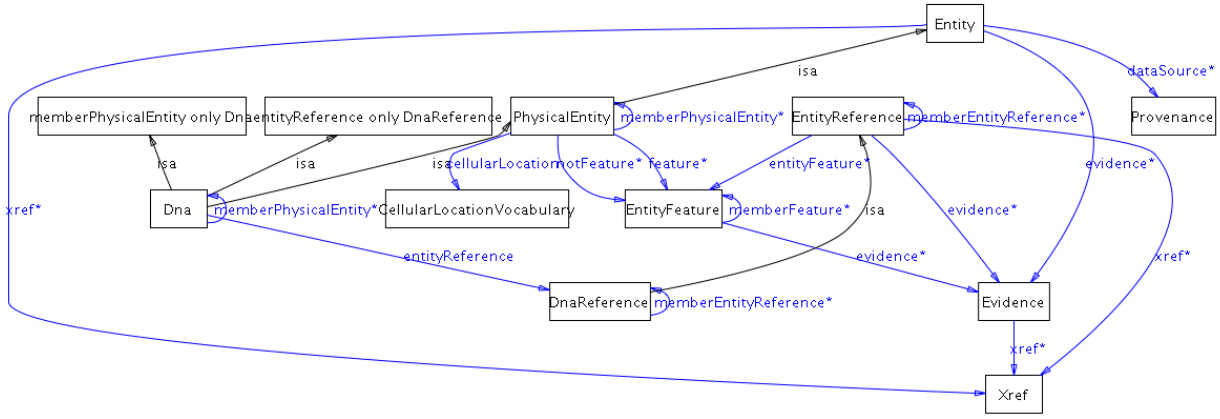
Definition: A physical entity consisting of a sequence of deoxyribonucleotide monophosphates; a deoxyribonucleic acid.
Comment: This is not a 'gene', since gene is a genetic concept, not a physical entity. The concept of a gene may be added later in BioPAX. Please note that the PhysicalEntity:DNARegion should be used when making reference to a more specific region on a DNA molecule.
Usage Note: DNA should be used for pools of individual DNA molecules. For describing subregions on those molecules use DNARegion.
Examples: a chromosome, a plasmid, viral genome. A specific example is chromosome 7 of Homo sapiens..
Parent class: *PhysicalEntity*
Properties: *entityReference*, *memberPhysicalEntity*, availability, *cellularLocation*, comment, *dataSource*, *evidence*, *feature*, name, *notFeature*, *xref*

entityReference - (0 or more object:DNAReference) The entity reference stores the non-changing aspects of this DNA molecule, such as the name, sequence, organism, database cross-references or

evidence.

memberPhysicalEntity – (0 or more object:DNA) Used to define a generic DNA molecule that is a collection of other DNA molecules. Important note: In general, the EntityReference class should be used to create generic DNA molecules (via its memberEntityReference property), however there are some cases where this is not possible, in which case, this property must be used. For instance, when EntityReference is used to define a generic physical entity with generic features, the generic features of the same type must be grouped. If you do not have grouping information for features of generic physical entities, you cannot use EntityReference to define generic physical entities and must use the memberPhysicalEntity property.

Object Properties Diagram:



DNARegion

Definition: A region of DNA.
Comment: This is not a ‘gene’, it is a molecule of DNA, which could encode a gene or other things, like transcription factor binding sites.
Examples: a chromosome, a plasmid, chromosome 7 of Homo sapiens.
Usage Note: DNARegion is not a pool of independent molecules but a subregion on these molecules. As such, every DNARegion has a defining DNA molecule.
Parent class: *PhysicalEntity*
Properties: *entityReference*, *memberPhysicalEntity*, availability, *cellularLocation*, comment, *dataSource*, *evidence*, *feature*, name, *notFeature*, *xref*

Protein

Definition: A physical entity consisting of a sequence of amino acids; a protein monomer; a single polypeptide chain. Aspects of the state of the protein, including cellular location and features, are defined here, using properties inherited from PhysicalEntity.

Examples: The epidermal growth factor receptor (EGFR) protein.
Parent class: *PhysicalEntity*
Properties: *entityReference*, *memberPhysicalEntity*, availability, *cellularLocation*, comment, *dataSource*, *evidence*, *feature*, name, *notFeature*, *xref*

entityReference - (0 or more object:ProteinReference) The entity reference stores the non-changing aspects of this protein, such as the name, sequence, organism, database cross-references or evidence.

memberPhysicalEntity – (0 or more object:Protein) Used to define a generic protein that is a collection of other proteins. Important note: In general, the EntityReference class should be used to create generic proteins (via its memberEntityReference property), however there are some cases where this is not possible, in which case, this property must be used. For instance, when EntityReference is used to define a generic physical entity with generic features, the generic features of the same type must be grouped. If you do not have grouping information for features of generic physical entities, you cannot use EntityReference to define generic physical entities and must use the memberPhysicalEntity property.

Object Properties Diagram: see DNA

RNA

Definition: A physical entity consisting of a sequence of ribonucleotide monophosphates; a ribonucleic acid. Aspects of the state of the RNA molecule, including cellular location and features, are defined here, using properties inherited from PhysicalEntity.
Examples: messengerRNA, microRNA, ribosomalRNA. A specific example is the let-7 microRNA.
Usage Note: RNA should be used for pools of individual RNA molecules. For describing subregions on those molecules use RNARegion.
Parent class: *PhysicalEntity*
Properties: *entityReference*, *memberPhysicalEntity*, availability, *cellularLocation*, comment, *dataSource*, *evidence*, *feature*, name, *notFeature*, *xref*

entityReference - (0 or more object:RNAReference) The entity reference stores the non-changing aspects of this RNA molecule, such as the name, sequence, organism, database cross-references or evidence.

memberPhysicalEntity – (0 or more object:RNA) Used to define a

generic RNA molecule that is a collection of other RNA molecules. Important note: In general, the EntityReference class should be used to create generic RNA molecules (via its memberEntityReference property), however there are some cases where this is not possible, in which case, this property must be used. For instance, when EntityReference is used to define a generic physical entity with generic features, the generic features of the same type must be grouped. If you do not have grouping information for features of generic physical entities, you cannot use EntityReference to define generic physical entities and must use the memberPhysicalEntity property.

Object Properties Diagram: see DNA

RNARegion

Definition: A region of RNA.
Comment: This is not a 'gene', it is a molecule of RNA, which could encode a gene or other things, like transcription factor binding sites.
Usage Note: RNARegion is not a pool of independent molecules but a subregion on these molecules. As such, every RNARegion has a defining RNA molecule.
Parent class: *PhysicalEntity*
Properties: *entityReference*, *memberPhysicalEntity*, availability, *cellularLocation*, comment, *dataSource*, *evidence*, *feature*, name, *notFeature*, *xref*

SmallMolecule

Definition: A small bioactive molecule. Small is not precisely defined, but includes all metabolites and most drugs and does not include large polymers, including complex carbohydrates. Aspects of the state of the small molecule, including cellular location and binding features, are defined here, using properties inherited from PhysicalEntity.
Comment: A number of small molecule databases are available to cross-reference from this class, such as PubChem. Complex carbohydrates are not currently modeled in BioPAX or most pathway databases, due to the lack of a publicly available complex carbohydrate database.
Examples: glucose, penicillin, phosphatidylinositol
Parent class: *PhysicalEntity*
Properties: *entityReference*, *feature*, *memberPhysicalEntity*, *notFeature*, availability, *cellularLocation*, comment, *dataSource*, *evidence*, name, *xref*

entityReference - (0 or more object:RNARegionReference) The entity reference stores the non-changing aspects of this small molecule, such

as the name, structure, database cross-references or evidence.

memberPhysicalEntity – (0 or more object:SmallMolecule) Used to define a generic small molecule that is a collection of other small molecules. Important note: In general, the EntityReference class should be used to create generic small molecules (via its memberEntityReference property), however there are some cases where this is not possible, in which case, this property must be used. For instance, when EntityReference is used to define a generic physical entity with generic features, the generic features of the same type must be grouped. If you do not have grouping information for features of generic physical entities, you cannot use EntityReference to define generic physical entities and must use the memberPhysicalEntity property.

feature - (0 or more object:BindingFeature) Features of the owner small molecule. Small molecule features are limited to binding sites.

notFeature - (0 or more object:BindingFeature) Features this small molecule is known to be lacking. Features that are not specified here, are not known to be absent. Only features known to be lacking that are relevant to the interaction are captured. Small molecule features are limited to binding sites.

Object Properties Diagram: see DNA

Utility classes

A number of properties in the ontology accept instances of utility classes as values. Utility classes are created when simple properties are insufficient to describe an aspect of an entity. This is a placeholder for classes, used for annotating the "Entity" and its subclasses. Mostly, these are not an "Entity" themselves. Examples include references to external databases, controlled vocabularies, evidence and provenance.
Utility subclasses
There are 15 direct subclasses of UtilityClass: **BioSource, ChemicalStructure, ControlledVocabulary, DeltaG, EntityFeature, EntityReference, Evidence, ExperimentalForm, kPrime, PathwayStep, Provenance, Score, SequenceLocation, Stoichiometry, and Xref.**

BioSource

Definition: The biological source of an entity (e.g. protein, RNA or DNA). Some entities are considered source-neutral (e.g. small molecules), and the biological source of others can be deduced from their constituents (e.g. complex, pathway), if specified.

Examples: HeLa cells, human, and mouse liver tissue.
Parent class: *Utility*
Properties: *cellType*, name, *Xref*, *tissue*, comment

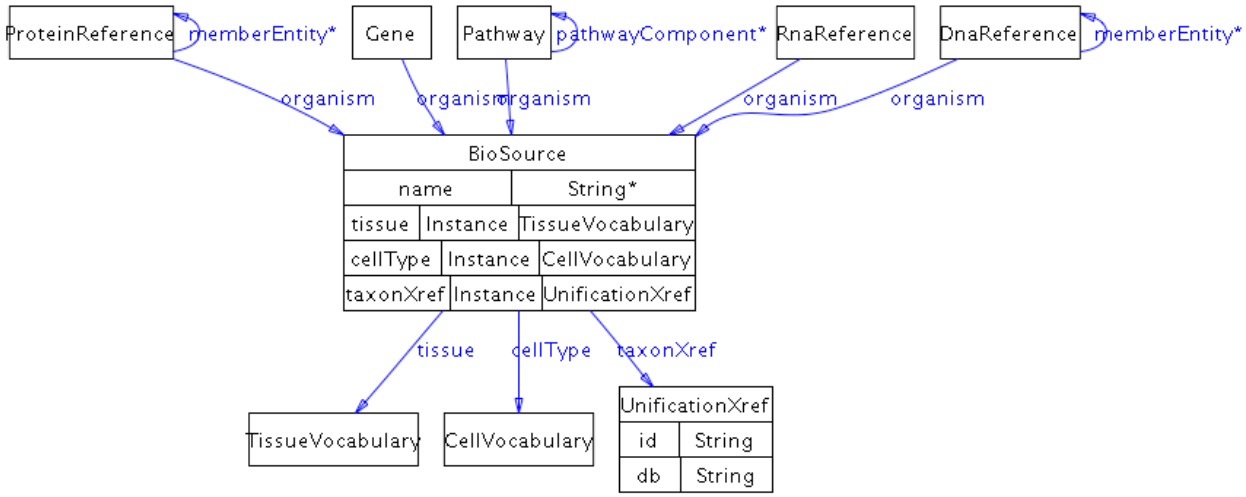
cellType - (0 or more object:CellVocabulary) A cell type, e.g. 'HeLa'. This should reference a term in the Cell Type Ontology (CL). Homepage at http://obofoundry.org/cgi-bin/detail.cgi?cell. Browse at http://www.ebi.ac.uk/ontology-lookup/browse.do?ontName=CL

name - (0 or 1) names of this entity. This will automatically include values of the displayName and standardName properties, as they are child properties of the name property. DisplayName values are short names suitable for display in a graphic. Standard names are names that follow a standard nomenclature, like systematic yeast ORF names (e.g. YJL034W).

Xref - (0 or 1, and only 1 object:UnificationXref) An xref to an organism taxonomy database, preferably NCBI taxon.

tissue - (0 or 1 object:TissueVocabulary) An external controlled vocabulary of tissue types. A reference to the BRENDA (BTO). Homepage at http://www.brenda-enzymes.info/. Browse at http://www.ebi.ac.uk/ontology-lookup/browse.do?ontName=BTO

Class Diagram:



ChemicalStructure

Definition: Used to describes a small molecule structure. Structure information is stored in the property *structureData*, in one of three formats: the CML format[13] (see URL www.xml-cml.org), the SMILES format[14] (see URL www.daylight.com/dayhtml/smiles/) or the InChI format (http://www.iupac.org/inchi/). The *structureFormat* property

specifies which format is used. The structureFormat property specifies which format is used.
Comment: By virtue of the expressivity of CML, an instance of this class can also provide additional information about a small molecule, such as its chemical formula, names, and synonyms, if CML is used as the structure format.
Examples: The following SMILES string describes the structure of glucose-6-phosphate:

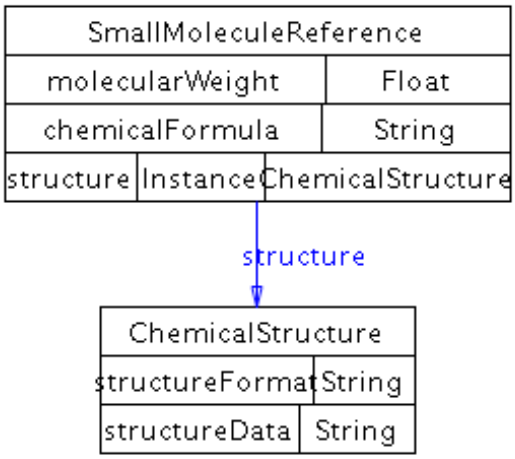
'C(OP(=O)(O)O)[CH]1([CH](O)[CH](O)[CH](O)[CH](O)O1)'.

The structure of ATP as an InChI string: InChI=1/C10H16N5O13P3/c11-8-5-9(13-2-12-8)15(3-14-5)10-7(17)6(16)4(26-10)1-25-30(21,22)28-31(23,24)27-29(18,19)20/h2-4,6-7,10,16-17H,1H2,(H,21,22)(H,23,24)(H2,11,12,13)(H2,18,19,20)/t4-,6-,7-,10-/m1/s1/f/h18-19,21,23H,11H2
Parent class: *UtilityClass*
Properties: structureData, structureFormat, comment

structureData - A string of data defining chemical structure or other information, in either the CML, SMILES or InChI format, as specified in property Structure-Format. If, for example, the CML format is used, then the value of this property is a string containing the XML encoding of the CML data.

structureFormat - The format is used to define chemical structure data.

Class Diagram:



ControlledVocabulary

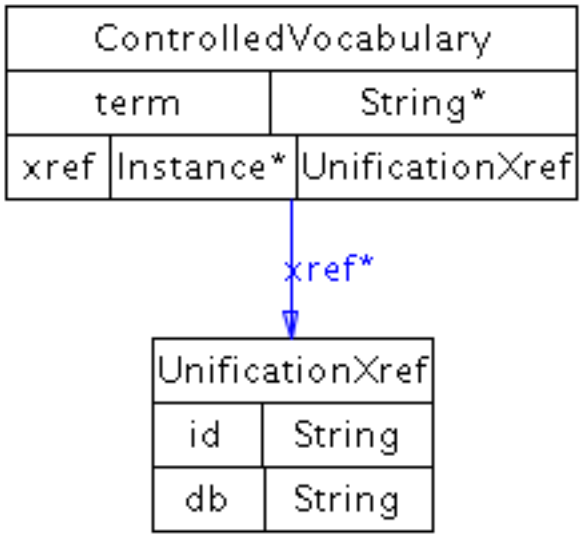
Definition: Used to reference terms from external controlled vocabularies (CVs) from the ontology. To support consistency and compatibility, open, freely available CVs should be used whenever

possible, such as the Gene Ontology (GO)[15] or other open biological CVs listed on the OBO website (http://obo.sourceforge.net/). See the section on controlled vocabularies in Section 4 for more information.
Comment: The ID property in unification xrefs to GO and other OBO ontologies should include the ontology letter code in the ID property (e.g. ID="GO:0005634" instead of ID="0005634").
Subclasses: *CellularLocationVocabulary, CellVocabulary, EntityReferenceTypeVocabulary, EvidenceCodeVocabulary, ExperimentalFormVocabulary, InteractionVocabulary, PhenotypeVocabulary, RelationshipTypeVocabulary, SequenceModificationVocabulary, SequenceRegionVocabulary, TissueVocabulary*
Parent class: *UtilityClass*
Properties: term, *xref*, comment

term - The external controlled vocabulary term.

xref - (0 or more object:UnificationXref) External cross-reference to the term in the ontology.

Class Diagram:



DeltaG

Definition: For biochemical reactions, this property refers to the standard transformed Gibbs energy change for a reaction written in terms of biochemical reactants (sums of species), delta-G'o.

delta-G'o = -RT lnK'
and

delta-G'o = delta-H'o - T delta-S'o

delta-G'o has units of kJ/mol. Like K', it is a function of temperature (T), ionic strength (I), pH, and pMg (pMg = -log10[Mg2+]). Therefore, these quantities must be specified, and values for delta G for biochemical reactions are represented as 5-tuples of the form (delta-G'o T I pH pMg). This property may have multiple values, representing different measurements for delta-G'o obtained under the different experimental conditions listed in the 5-tuple.

(This definition from EcoCyc)
Usage Note: Delta-G is represented as a 5-tuple of delta-G'⁰, temperature, ionic strength , pH, and pMg . A conversion in BioPAX may have multiple Delta-G values, representing different measurements for delta-G'⁰ obtained under the different experimental conditions.
Parent class: *UtilityClass*
Properties: deltaGPrime0, ionicStrength, pH, pMg, temperature, comment

deltaGPrimeO - For biochemical reactions, this property refers to the standard transformed Gibbs energy change for a reaction written in terms of biochemical reactants (sums of species), delta-G'o.

delta-G'o = -RT lnK'
and
delta-G'o = delta-H'o - T delta-S'o

delta-G'o has units of kJ/mol. Like K', it is a function of temperature (T), ionic strength (I), pH, and pMg (pMg = -log10[Mg2+]). Therefore, these quantities must be specified, and values for DELTA-G for biochemical reactions are represented as 5-tuples of the form (delta-G'o T I pH pMg).

(This definition from EcoCyc)

ionicStrength - The ionic strength is defined as half of the total sum of the concentration (ci) of every ionic species (i) in the solution times the square of its charge (zi). For example, the ionic strength of a 0.1 M solution of CaCl2 is 0.5 x (0.1 x 22 + 0.2 x 12) = 0.3 M
(Definition from http://www.lsbu.ac.uk/biology/enztech/ph.html)

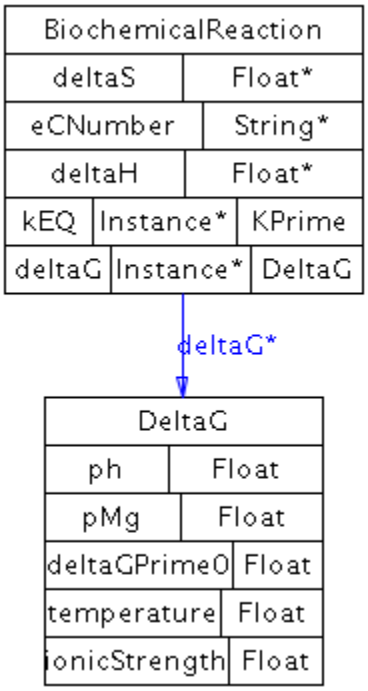
ph - a measure of acidity and alkalinity of a solution that is a number on a scale on which a value of 7 represents neutrality and lower numbers indicate increasing acidity and higher numbers increasing alkalinity and on which each unit of change represents a tenfold

change in acidity or alkalinity and that is the negative logarithm of the effective hydrogen-ion concentration or hydrogen-ion activity in gram equivalents per liter of the solution. (Definition from Merriam-Webster Dictionary)

pMg - A measure of the concentration of magnesium (Mg) in solution. (pMg = -log10[Mg2+])

temperature - Temperature in Celsius.

Class Diagram:



EntityFeature

Definition: A feature or aspect of a physical entity that can be changed while the entity still retains its biological identity. Used in physical entities to further define the state of the entity and differentiate states from each other. Entity features can be generic across physical entities that are in a generic grouping. This allows generic features to be defined on a generic physical entity. This can be used directly if no subclass covers the feature that differentiates states.
Examples: Conformation, open/closed state of a channel or repeated subunits in a complex that bind different states of another physical entity. Subclasses cover more specific examples.
Usage Note: Subclasses of entity feature describe most common biological instances and should be preferred whenever possible. One common use case for instantiating entity feature is, for describing

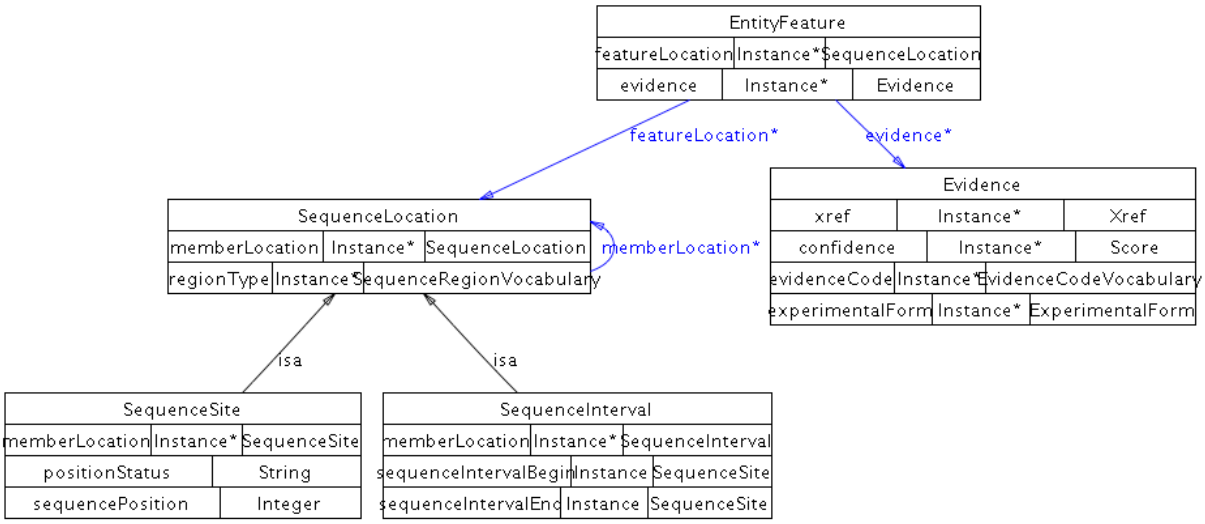
active/inactive states of proteins where more specific feature information is not available.
Subclasses: *BindingFeature, FragmentFeature, ModificationFeature*
Parent class: *UtilityClass*
Properties: *evidence*, *featureLocation*, *featureLocationType*, *memberFeature*, comment

evidence - (0 ore more object:Evidence) Scientific evidence supporting the existence of the feature.

featureLocation - (0 or more object:SequenceLocation) Location of the feature on the sequence of the interactor. One feature may have more than one location, used e.g. for features which involve sequence positions close in the folded, three-dimensional state of a protein, but non-continuous along the sequence.

memberFeature – (0 or more object:EntityFeature) Used to define a generic feature, which is an entity feature that belongs to a generic physical entity, as defined, for example, by a homology grouping. If used, the referencing feature is generic. Lists features that are part of the generic feature. Example: a homologous phosphorylation site across a protein family.

Class Diagram:



EntityReference

Definition: An entity reference is a grouping of several physical entities across different contexts and molecular states, that share common physical properties and often named and treated as a single entity with multiple states by biologists. This is similar to the definition of a protein in a protein sequence database, like UniProt and there

should only be one EntityReference defined per UniProt ID.

Entity reference instances store the information common to a set of molecules in various states, including database cross-references. For instance, the P53 protein can be phosphorylated in multiple different ways. Each separate P53 protein (pool) in a phosphorylation state is represented as a different Protein (child of PhysicalEntity) and all things common to all P53 proteins, including all possible phosphorylation sites, the sequence common to all of them and common references to protein databases containing more information about generic P53 are stored in an EntityReference. The entity reference is important because it explicitly links multiple physical entities representing different states of the same molecule, which would otherwise be difficult to recognize as related. Also, EntityReference allows the accurate creation of unification links to source databases for generic entities, for instance linking P53 to the UniProt P53 protein record (since UniProt stores generic proteins, not any specific individual protein state). Databases that store proteins in states, like Reactome or the Nature Molecule Pages databases.

Comment: Many protein, small molecule and gene databases follow EntityReference semantics, thus EntityReference is an important element for interoperability with those databases. Biologists often group different pools of molecules in different contexts under the same name or database identifier. For example cytoplasmic and extracellular calcium have different effects on the cell's behavior, but they are still called calcium. This grouping has three semantic implications:

1. Members of different pools share many physical and biochemical properties. This includes their chemical structure, sequence, organism and set of molecules they react with. They will also share a lot of secondary information such as their names, functional groupings, annotation terms and database identifiers.

2. A small number of transitions separate these pools. In other words molecules frequently are transformed from one physical entity to another by enzymatic reaction or other mechanism, where all the transformed states share the majority of their definition, that is they belong to the same EntityReference. For example, an extracellular calcium ion can be cytoplasmic, and p53 protein can be phosphorylated. However no calcium ion virtually becomes sodium, or no p53 becomes Mdm2. In the former it is the sheer energy barrier of a nuclear reaction, in the latter sheer statistical improbability of synthesizing the same sequence without a template. If one thinks about the biochemical network as molecules transforming into each other, and remove edges that correspond to transcription, translation,

degradation and covalent modification of small molecules, each remaining component is a reference entity.

3. Some of the pools in the same group can overlap. P53-p@ser15 can overlap with p53-p@thr18. Most of the experiments in molecular biology will only investigate one feature, rarely multiple, and almost never all possible combinations. So almost all statements that refer to the state of the molecule in scientific literature refer to a pool that can overlap with other pools. However no overlap is possible between molecules of different EntityReference's.
Subclasses: DnaReference, ProteinReference, RnaReference, SmallMoleculeReference
Parent class: *UtilityClass*
Properties: *entityFeature*, *entityReferenceType*, *evidence*, *memberEntityReference*, name ,*xref*

entityFeature - (0 or more object:entityFeature) Variable feature observed for this entity, such as protein post-translational modification, DNA methylation site, or non-covalent binding partners. This acts as a database of features in use by the physical entities that reference this EntityReference instance. Note that features are also referenced in PhysicalEntity, in the feature and notFeature properties. In that case, only features that defined the physical entity are referenced. This introduces some redundancy, as features are referenced multiple times. However, this property is required to relate generic feature definitions to generic physical entities, which are respectively defined using the memberFeature property of EntityFeature and the memberEntityReference of EntityReference.

evidence - (0 or more object:Evidence) Scientific evidence supporting the existence of this EntityReference.

entityReferenceType – (0 or more object:entityReferenceTypeVocabulary) A controlled vocabulary term that describes the type of grouping, such as homology or functional group.

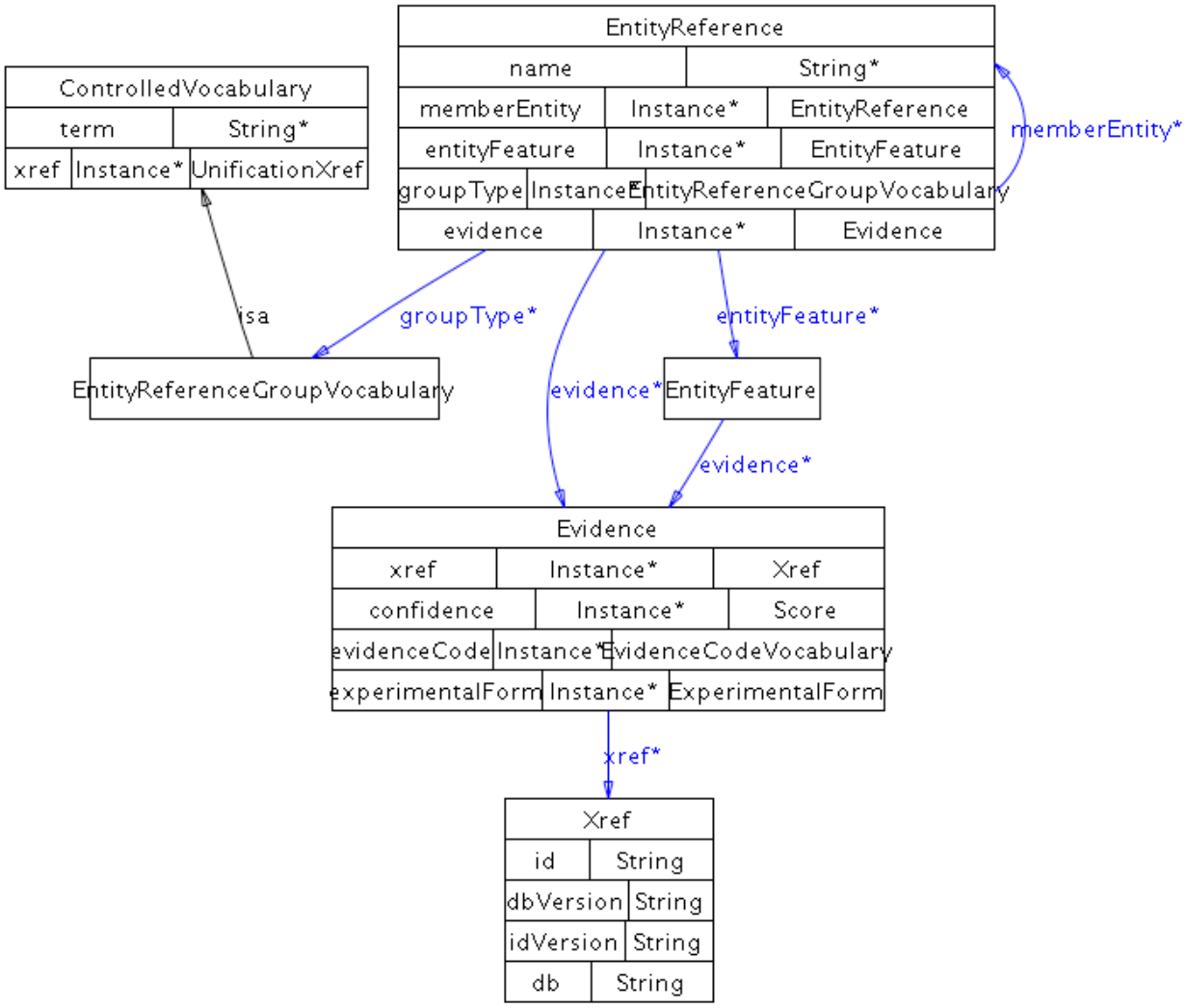
memberEntityReference – (0 or more object:EntityReference) If used, defines a generic EntityReference. The set of values defines the generic EntityReference. For example, a homology grouping, which is a set of physical entities that are homologous, like a PFAM protein family.

name - One or more names of this entity. This will automatically include values of the displayName and standardName properties, as they are child properties of the name property. DisplayName values are short names suitable for display in a graphic. Standard names are

names that follow a standard nomenclature, like systematic yeast ORF names (e.g. YJL034W).

xref - (0 or more object:Xref) External cross-references from this EntityReference to entities in external databases.

Class Diagram:



Evidence

Definition: The scientific support for a particular assertion, such as the existence of an interaction or pathway. At least one of *confidence*, *evidenceCode*, or *experimentalForm* must be instantiated. *xref* may reference a publication describing the experimental evidence using a **PublicationXref** or may reference a description of the experiment in an experimental description database using a **UnificationXref** (if the referenced experiment is the same) or **RelationshipXref** (if it is not identical, but similar in some way e.g. similar in protocol). Evidence is

meant to provide more information than just an xref to the source paper.
Examples: A molecular binding assay used to detect a protein-protein interaction, yeast two-hybrid.
Parent class: *UtilityClass*
Properties: *confidence*, *evidenceCode*, *experimentalForm*, *xref*, comment

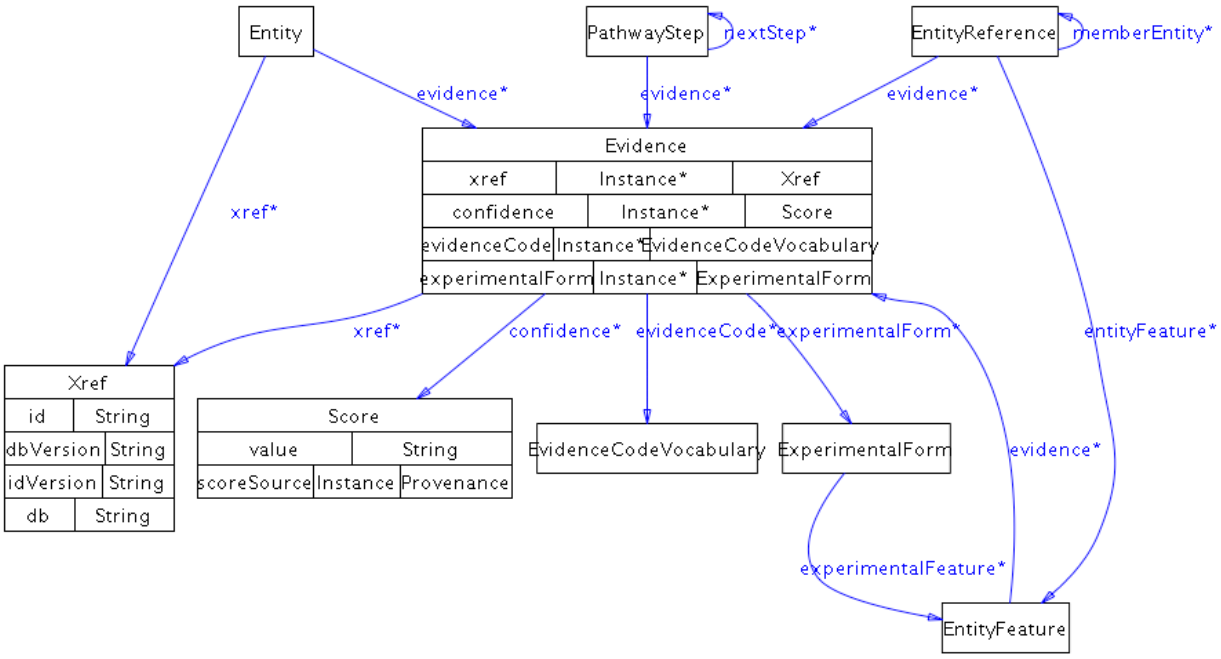
confidence - (0 or more object:Score) Confidence in the containing instance, usually a statistical measure.

evidenceCode - (0 or more object:EvidenceCodeVocabulary) References a term in an external controlled vocabulary, such as the PSI-MI experimental method types or BioCyc evidence codes, that describes the nature of the support. See the section on controlled vocabularies in Section 4 for more information.

experimentalForm - (0 or more object:ExperimentalForm) The experimental forms associated with an evidence instance.

xref - (0 or more object:Xref) External cross-reference to evidence in an external database.

Class Diagram:



ExperimentalForm

Definition: The form of a physical entity in a particular experiment, as it may be modified for purposes of experimental design. This is not the

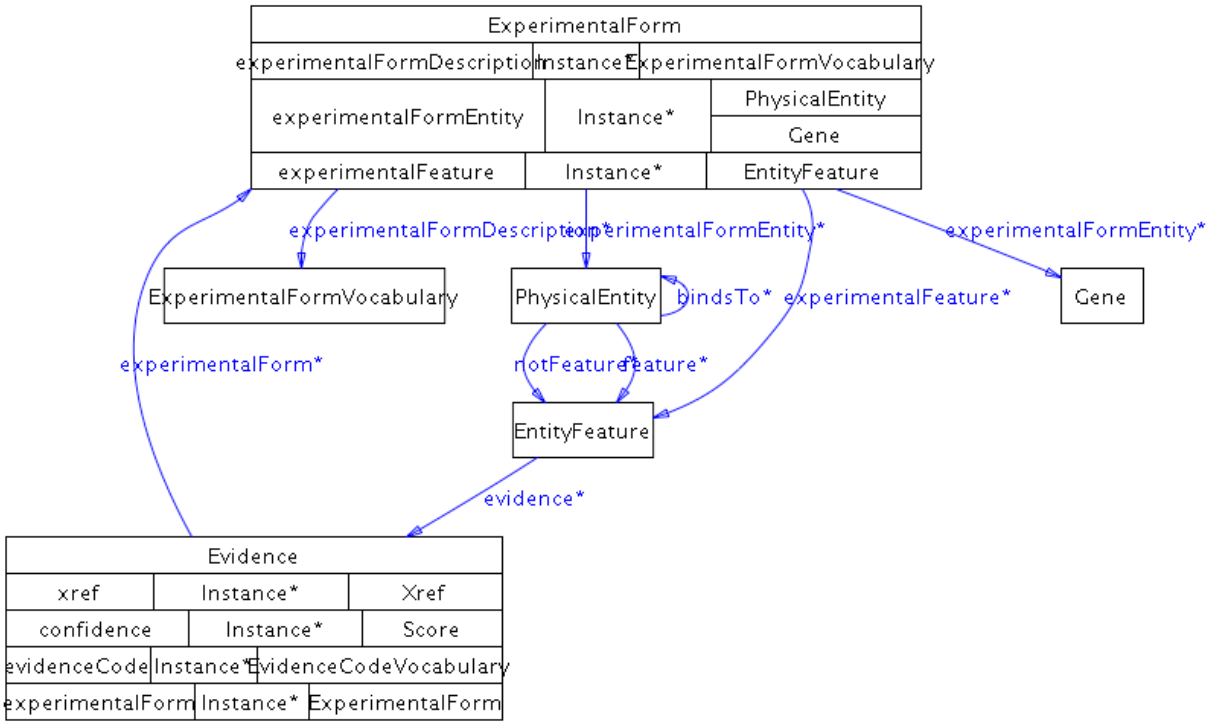
same as biologically relevant modifications captured in physical entity or entity reference features.
Examples: A His-tagged protein in a binding assay. A protein can be tagged by multiple tags, so can have more than one experimental form type terms. In the case of a gene, this could be a knock-out or mutation.
Parent class: *UtilityClass*
Properties: *experimentalFeature*, *experimentalFormDescription*, *experimentalFormEntity*, comment

experimentalFormDescription- (0 or more object:ExperimentalFormVocabulary) Descriptor of this experimental form from a controlled vocabulary. A reference to the PSI Molecular Interaction ontology (MI) participant identification method (e.g. mass spectrometry), experimental role (e.g. bait, prey), experimental preparation (e.g. expression level) type. Homepage at http://www.psidev.info/. Browse at http://www.ebi.ac.uk/ontology-lookup/browse.do?ontName=MI&termId=MI%3A0002&termName=participant%20identification%20method See the section on controlled vocabularies in Section 4 for more information.

experimentalFormEntity - (0 or more object:PhysicalEntity or object:Gene) The Gene or PhysicalEntity that has the experimental form being described.

experimentalFeature - (0 or more object:EntityFeature) A feature of the experimental form of the participant of the interaction, such as a protein tag. It is not expected to occur in vivo or be necessary for the interaction.

Class Diagram:



kPrime

Definition: The apparent equilibrium constant, K', and associated values. Concentrations in the equilibrium constant equation refer to the total concentrations of all forms of particular biochemical reactants. For example, in the equilibrium constant equation for the biochemical reaction in which ATP is hydrolyzed to ADP and inorganic phosphate:

K' = [ADP][Pi]/[ATP],

The concentration of ATP refers to the total concentration of all of the following species:

[ATP] = [ATP4-] + [HATP3-] + [H2ATP2-] + [MgATP2-] + [MgHATP-] + [Mg2ATP].

The apparent equilibrium constant is formally dimensionless, and can be kept so by inclusion of as many of the terms (1 mol/dm3) in the numerator or denominator as necessary. It is a function of temperature (T), ionic strength (I), pH, and pMg (pMg = -log10[Mg2+]). Therefore, these quantities must be specified to be precise, and values for KEQ for biochemical reactions may be represented as 5-tuples of the form (K' T I pH pMg). This property may have multiple values, representing different measurements for K' obtained under the

different experimental conditions listed in the 5-tuple. (This definition adapted from EcoCyc)

See http://www.chem.qmul.ac.uk/iubmb/thermod/ for a thermodynamics tutorial.
Parent class: *UtilityClass*
Properties: ionicStrength, kPrime, ph, pMg, temperature, comment

ionicStrength - The ionic strength is defined as half of the total sum of the concentration (ci) of every ionic species (i) in the solution times the square of its charge (zi). For example, the ionic strength of a 0.1 M solution of CaCl2 is 0.5 x (0.1 x 22 + 0.2 x 12) = 0.3 M
(Definition from http://www.lsbu.ac.uk/biology/enztech/ph.html)

kPrime - The apparent equilibrium constant K'. Concentrations in the equilibrium constant equation refer to the total concentrations of all forms of particular biochemical reactants. For example, in the equilibrium constant equation for the biochemical reaction in which ATP is hydrolyzed to ADP and inorganic phosphate:

K' = [ADP][Pi]/[ATP],

The concentration of ATP refers to the total concentration of all of the following species:

[ATP] = [ATP4-] + [HATP3-] + [H2ATP2-] + [MgATP2-] + [MgHATP-] + [Mg2ATP].

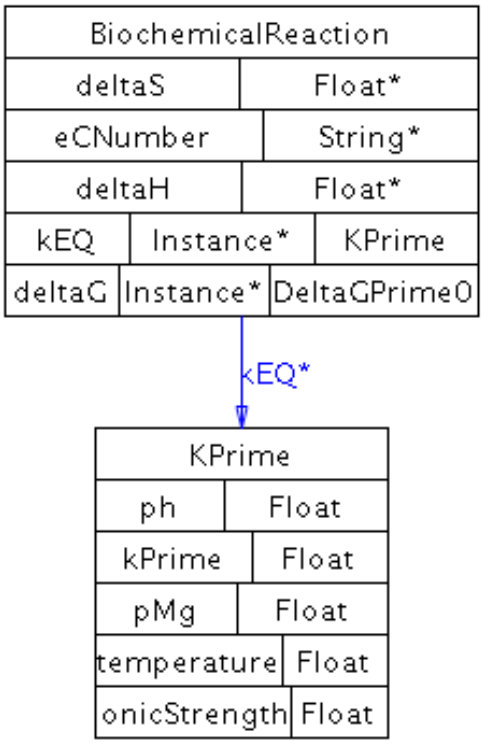
The apparent equilibrium constant is formally dimensionless, and can be kept so by inclusion of as many of the terms (1 mol/dm3) in the numerator or denominator as necessary. It is a function of temperature (T), ionic strength (I), pH, and pMg (pMg = -log10[Mg2+]). (Definition from EcoCyc)

ph - a measure of acidity and alkalinity of a solution that is a number on a scale on which a value of 7 represents neutrality and lower numbers indicate increasing acidity and higher numbers increasing alkalinity and on which each unit of change represents a tenfold change in acidity or alkalinity and that is the negative logarithm of the effective hydrogen-ion concentration or hydrogen-ion activity in gram equivalents per liter of the solution. (Definition from Merriam-Webster Dictionary)

pMg - A measure of the concentration of magnesium (Mg) in solution. (pMg = -log10[Mg2+])

temperature - Temperature in Celsius

Class Diagram:



PathwayStep

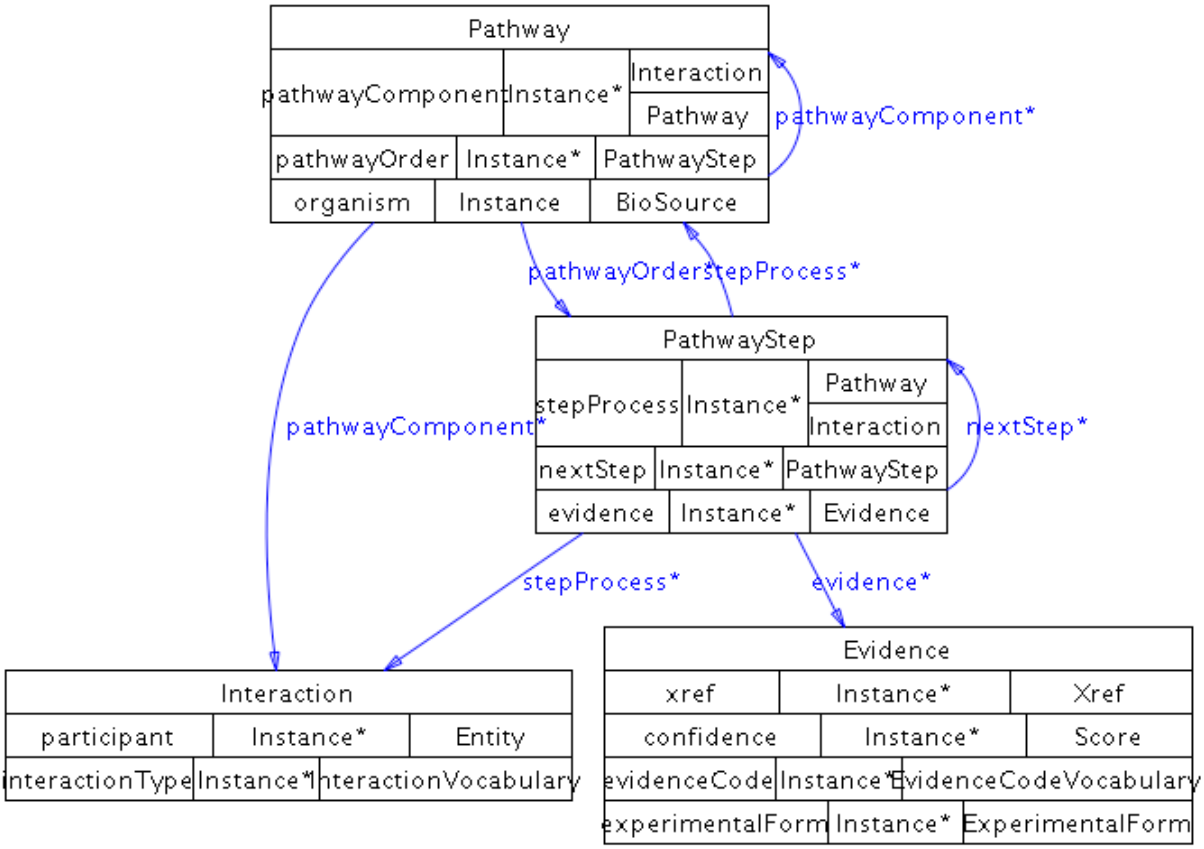
Definition: A step in an ordered pathway.
Comment: Some pathways can have a temporal order. For example, if the pathway boundaries are based on a perturbation phenotype link, the pathway might start with the perturbing agent and end at gene expression leading to the observed changes. Pathway steps can represent directed compound graphs.
Usage Note: Multiple interactions may occur in a pathway step, each should be listed in the stepProcess property. Order relationships between pathway steps may be established with the nextStep slot. If the reaction contained in the step is a reversible biochemical reaction but physiologically has a direction in the context of this pathway, use the subclass BiochemicalPathwayStep.
Example: A metabolic pathway may contain a pathway step composed of one biochemical reaction (BR1) and one catalysis (CAT1) instance, where CAT1 describes the catalysis of BR1. The M phase of the cell cycle, defined as a pathway, precedes the G1 phase, also defined as a pathway.
Subclasses: BiochemicalPathwayStep
Parent class: *UtiltiyClass*
Properties: *evidence*, *nextStep*, *stepProcess*, comment

evidence - (0 or more object:Evidence) Scientific evidence supporting this pathwayStep.

nextStep - (0 or more object:PathwayStep) The next step(s) of the pathway. Contains zero or more pathwayStep instances. If there is no next step, this property is empty.

stepProcess - (0 or more object:Pathway or object:Interaction) An interaction or a pathway that is part of this pathway step.

Class Diagram:



Provenance

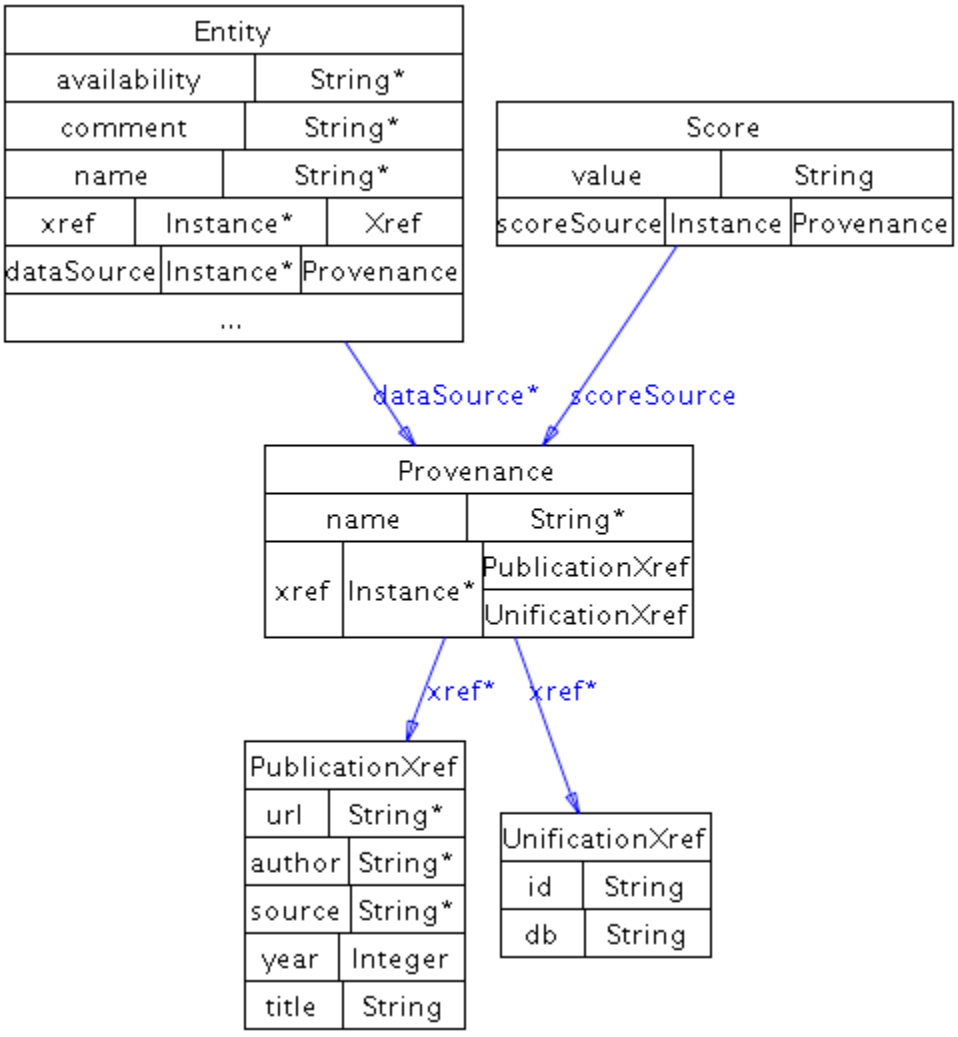
Definition: The direct source of this data. This does not store the trail of sources from the generation of the data to this point, only the last known source, such as a database. The *xref* property may contain a **PublicationXref** referencing a publication describing the data source (e.g. a database publication). A **UnificationXref** may be used e.g. when pointing to an entry in a database of databases describing this database.
Examples: A database or person name.

Parent class: *UtilityClass*
Properties: name, *xref*, comment

name - One or more names of this entity. This will automatically include values of the displayName and standardName properties, as they are child properties of the name property. DisplayName values are short names suitable for display in a graphic. Standard names are names that follow a standard nomenclature, like systematic yeast ORF names (e.g. YJL034W).

xref - (0 or more object:PublicationXref or object:UnificationXref) Cross-references from this provenance entity to entities in external databases.

Class Diagram:



Score

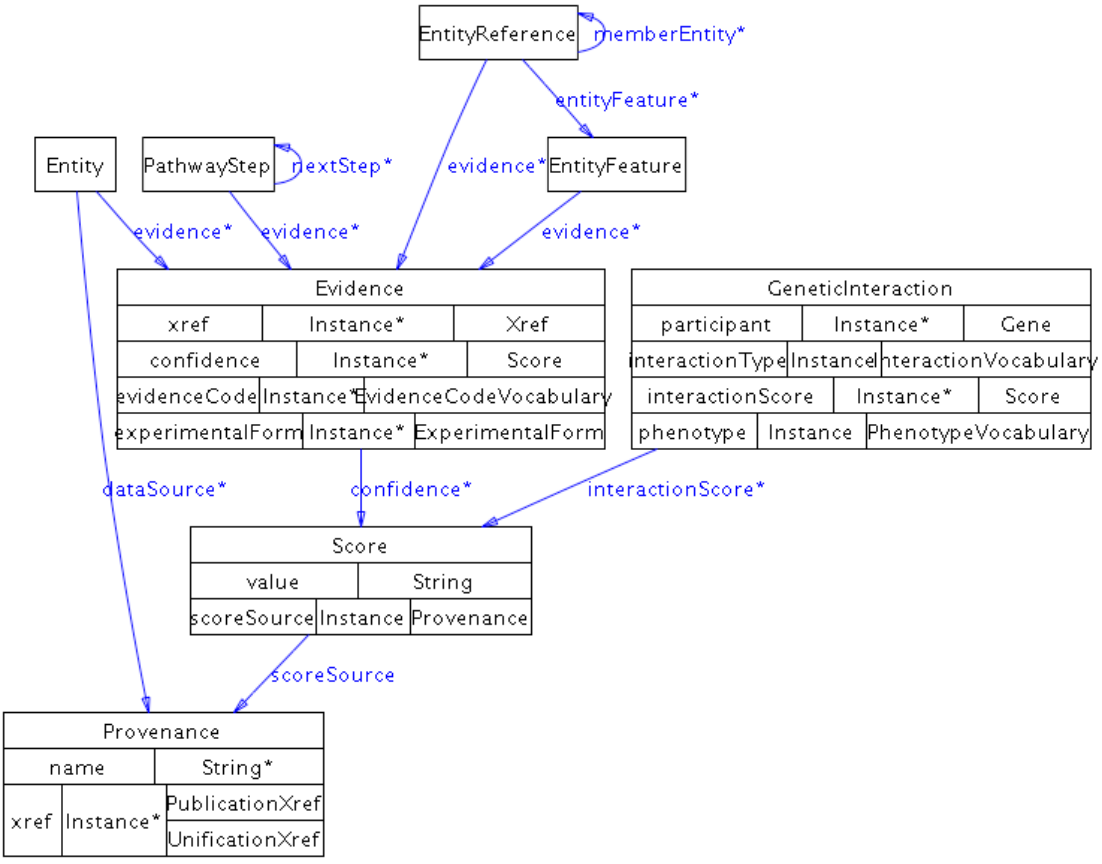
Definition: A score associated with a publication reference describing

how the score was determined, the name of the method and a comment briefly describing the method.
Usage Note: The xref must contain at least one publication that describes the method used to determine the score value. There is currently no standard way of describing values, so any string is valid. Examples: The statistical significance of a result, e.g. "p<0.05".", a genetic interaction score.
Parent class: *UtiltiyClass*
Properties: *scoreSource*, value, comment

scoreSource – (0 or 1 object:Provenance) The name and source of the score, such as a publication describing the scoring system.

value - The value of the score measure.

Class Diagram:

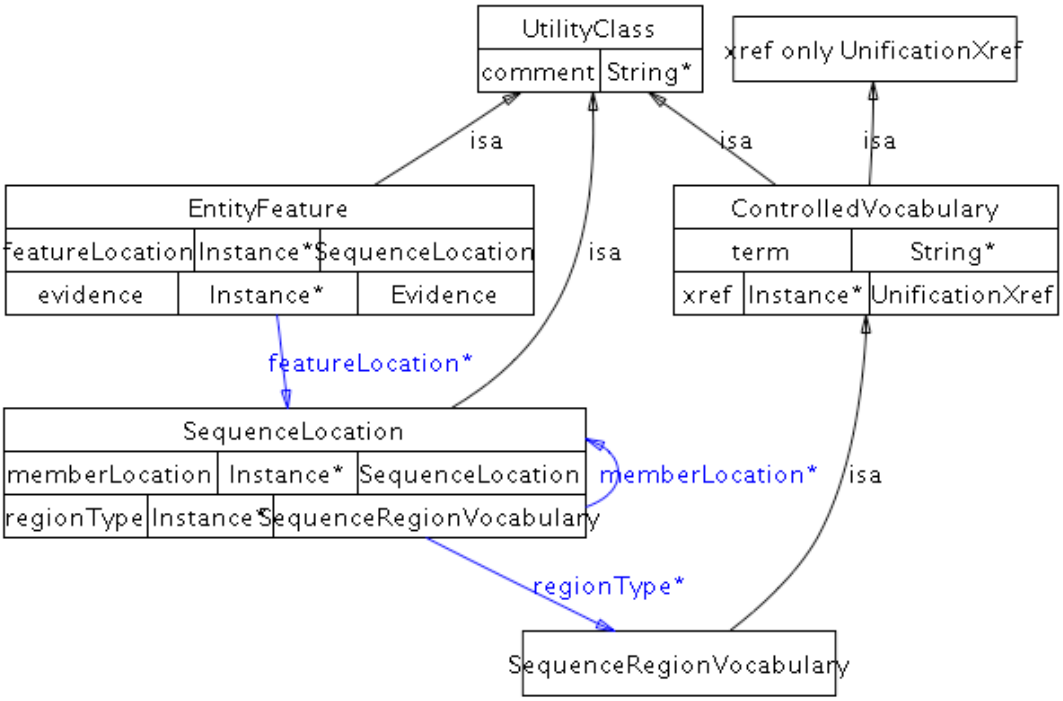


SequenceLocation

Definition: A location on a nucleotide or amino acid sequence.
Usage Note: For most purposes it is more appropriate to use subclasses of this class. Direct instances of SequenceLocation can be

used for uknown locations that can not be classified neither as an interval nor a site.
Subclasses: SequenceInterval, SequenceSite
Parent class: *UtilityClass*
Properties: comment

Class Diagram:



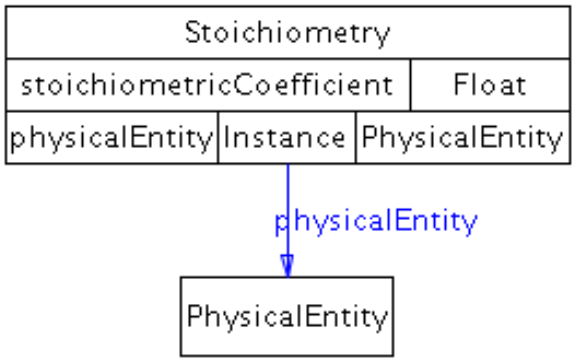
Stoichiometry

Definition: Stoichiometric coefficient of a physical entity in the context of a **Conversion** or **Complex**. This class is an n-ary specifier for left and right properties of a biochemical reaction, i.e. it specifies the stoichiometry externally to the left and right property data structures, using the dangling n-ary design pattern.
Usage Note: For each participating element there must be 0 or 1 stoichiometry element. A non-existing stoichiometric element is treated as unknown.
This is an n-ary bridge for left, right and component properties. Relative stoichiometries (e.g n, n+1) often used for describing polymerization is not supported.
Parent class: *UtilityClass*
Properties: *physicalEntity*, stoichiometricCoefficient, comment

physicalEntity - (0 or 1 object:PhysicalEntity) The physical entity annotated with the stoichiometry attribute from the corresponding Stoichiometry instance.

stoichiometricCoefficient - Each value of this property represents the stoichiometric coefficient for one of the entities in an interaction or complex. For a given interaction, the stoichiometry should always be used where possible instead of representing the number of participants with separate instances of each participant. If there are three ATP molecules, one ATP molecule should be represented as a participant and the stoichiometry should be set to 3.

Class Diagram:



Xref

Definition: A reference from an instance of a class in this ontology to an object in an external resource.
Usage Note: For most cases one of the subclasses of xref should be used.
Subclasses: *PublicationXref*, *RelationshipXref*, *UnificationXref*
Parent class: *UtilityClass*
Properties: db, dbVersion, id, idVersion, comment

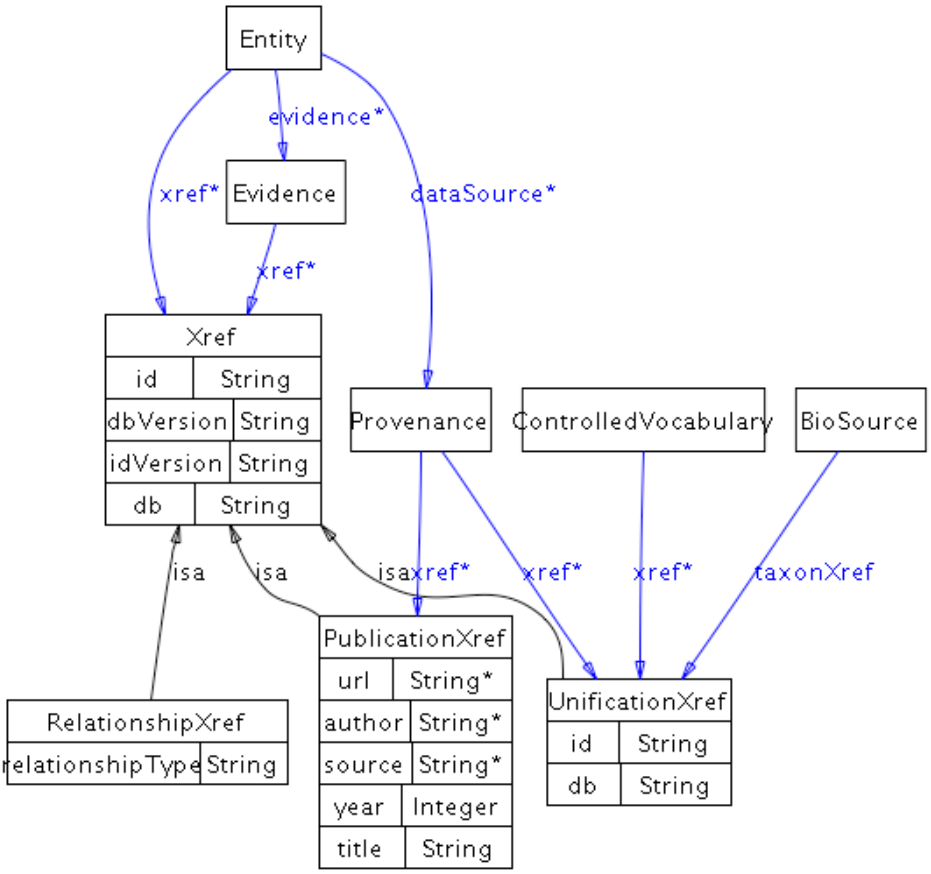
db - The name of the external database to which this xref refers.

dbVersion - The version of the external database in which this xref was last known to be valid. Resources may have recommendations for referencing dataset versions. For instance, the Gene Ontology recommends listing the date the GO terms were downloaded.

id - The primary identifier in the external database of the object to which this xref refers.

idVersion - The version number of the identifier (ID). E.g. The RefSeq accession number NM_005228.3 should be split into NM_005228 as the ID and 3 as the ID-VERSION.

Object Properties Diagram:



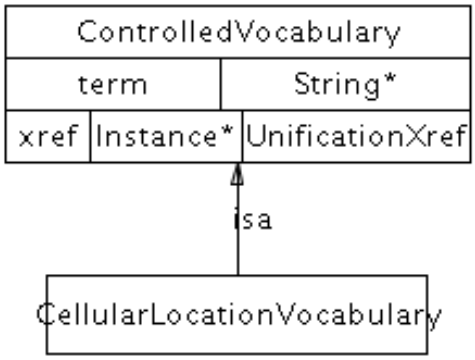
ControlledVocabulary subclasses

All ControlledVocabulary subclasses inherit a term and an xref property from ControlledVocabulary.

CellularLocationVocabulary

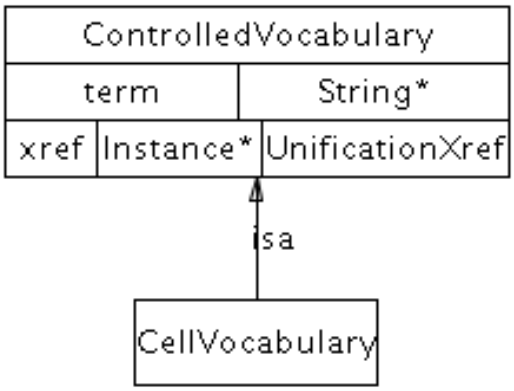
Definition: A reference to the Gene Ontology Cellular Component (GO CC) ontology. Homepage at http://www.geneontology.org. Browse at http://www.ebi.ac.uk/ontology-lookup/browse.do?ontName=GO
Comment: The location referred to by this property should be as specific as is known. If an interaction is known to occur in multiple locations, separate interactions must be created for each different location. Note: If a location is unknown then no term should be specified. Do not use the GO term for 'cellular component unknown' (GO:0008372). If the location of a participant in a complex is unspecified, it may be assumed to be the same location as that of the complex. In case of conflicting information, the location of the most outer layer of any nesting should be considered correct. Cellular location describes a specific location of a physical entity as it would be used in e.g. a transport reaction. It does not describe all of the possible locations that the physical entity could possibly be in the cell, as would be listed in all known GO cellular component annotations for the

protein.
Parent class: *ControlledVacabulary*
Properties: *xref*, term comment
Class Diagram:



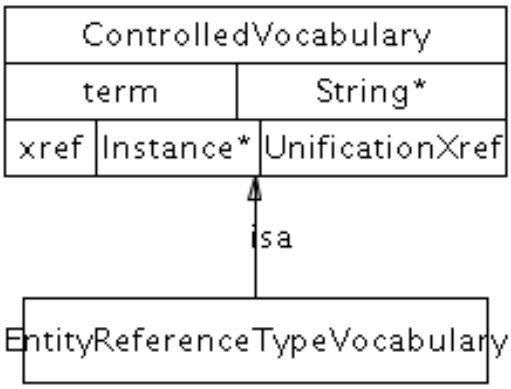
CellVocabulary

Definition: A reference to the Cell Type Ontology (CL). Homepage at http://obofoundry.org/cgi-bin/detail.cgi?cell. Browse at http://www.ebi.ac.uk/ontology-lookup/browse.do?ontName=CL
Parent class: *ControlledVacabulary*
Properties: *xref*, term comment
Class Diagram:



EntityReferenceTypeVocabulary

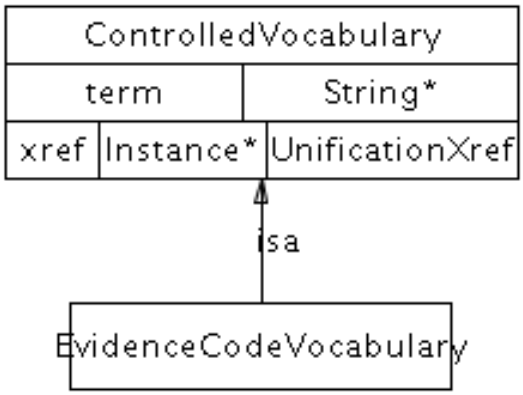
Definition: A reference to a term from an entity reference group ontology. As of the writing of this documentation, there is no standard ontology of these terms, though a common type is ‘homology’.
Parent class: *ControlledVacabulary*
Properties: *xref*, term comment
Class Diagram:



EvidenceCodeVocabulary

Definition: A reference to the PSI Molecular Interaction ontology (MI) experimental method types, including "interaction detection method", "participant identification method", "feature detection method". Homepage at http://www.psidev.info/. Browse at http://www.ebi.ac.uk/ontology-lookup/browse.do?ontName=MI

Terms from the Pathway Tools Evidence Ontology may also be used. Homepage http://brg.ai.sri.com/evidence-ontology/
Parent class: *ControlledVacabulary*
Properties: *xref*, term comment
Class Diagram:



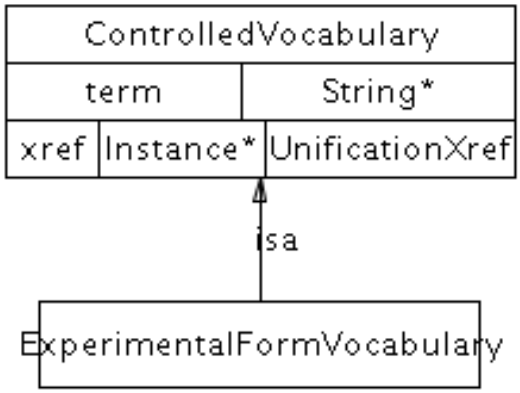
ExperimentalFormVocabulary

Definition: A reference to the PSI Molecular Interaction ontology (MI) participant identification method (e.g. mass spectrometry), experimental role (e.g. bait, prey), experimental preparation (e.g. expression level) type. Homepage at http://www.psidev.info/. Browse at http://www.ebi.ac.uk/ontology-lookup/browse.do?ontName=MI&termId=MI%3A0002&termName=participant%20identification%20method

http://www.ebi.ac.uk/ontology-lookup/browse.do?

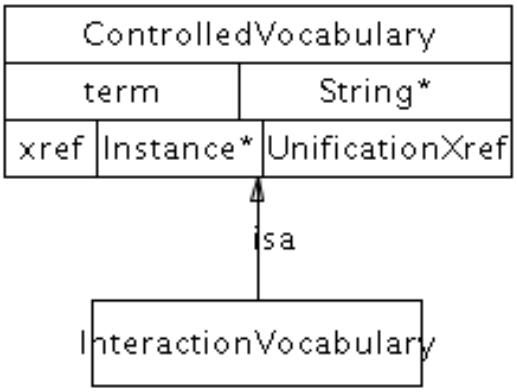
ontName=MI&termId=MI%3A0495&termName=experimental%20role

http://www.ebi.ac.uk/ontology-lookup/browse.do?
ontName=MI&termId=MI%3A0346&termName=experimental
%20preparation
Parent class: *ControlledVacabulary*
Properties: *xref*, term comment
Class Diagram:



InteractionVocabulary

Definition: A reference to the PSI Molecular Interaction ontology (MI) interaction type. Homepage at http://www.psidev.info/. Browse at http://www.ebi.ac.uk/ontology-lookup/browse.do?
ontName=MI&termId=MI%3A0190&termName=interaction%20type
Parent class: *ControlledVacabulary*
Properties: *xref*, term comment
Class Diagram:



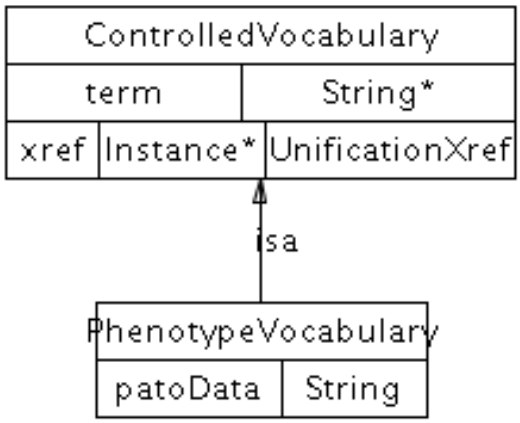
PhenotypeVocabulary

Definition: The phenotype measured in the experiment e.g. growth rate or viability of a cell. This is only the type, not the value e.g. for a synthetic lethal interaction, the phenotype is viability, specified by ID: PATO:0000169, "viability", not the value (specified by ID: PATO:0000718, "lethal (sensu genetics)". A single term in a phenotype

controlled vocabulary can be referenced using the xref, or the PhenoXML describing the PATO EQ model phenotype description can be stored as a string in patoData.
Parent class: proData, *xref*, comment, term
Properties:

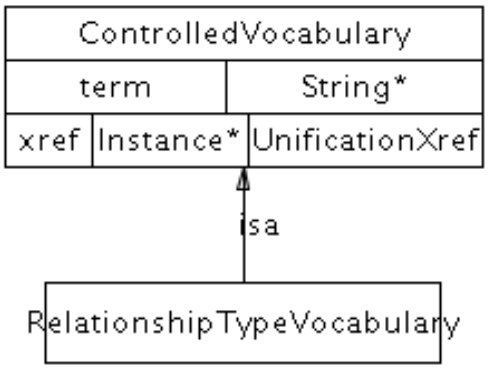
patoData - The phenotype data from PATO, formatted as PhenoXML (defined at http://www.fruitfly.org/~cjm/obd/formats.html)

Class Diagram:



RelationshipTypeVocabulary

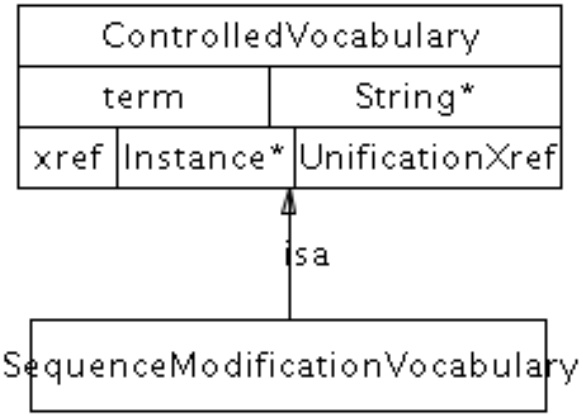
Definition: Vocabulary for defining relationship Xref types. A reference to the PSI Molecular Interaction ontology (MI) Cross Reference type. Homepage at http://www.psidev.info/. Browse at http://www.ebi.ac.uk/ontology-lookup/browse.do?ontName=MI&termId=MI%3A0353&termName=cross-reference%20type
Parent class: *ControlledVacabulary*
Properties: *xref*, term comment
Class Diagram:



SequenceModificationVocabulary

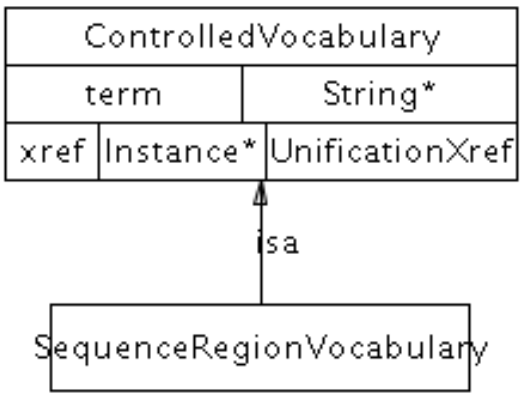
Definition: A reference to the PSI Molecular Interaction ontology (MI) of covalent sequence modifications. Homepage at http://www.psidev.info/. Browse at http://www.ebi.ac.uk/ontology-lookup/browse.do?ontName=MI&termId=MI%3A0252&termName=biological%20feature. Only children terms that are covalent modifications at specific positions can be used.

Object Properties Diagram:



SequenceRegionVocabulary

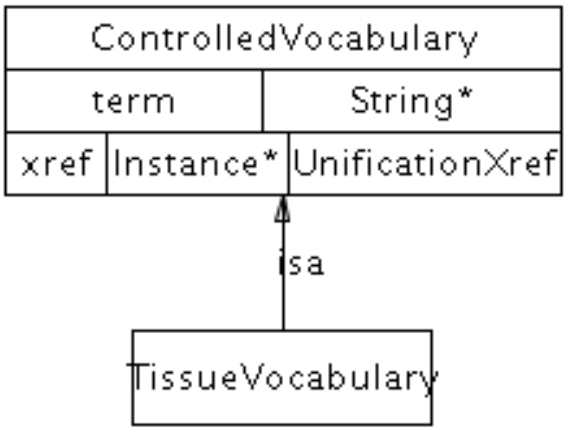
Definition: A reference to the Sequence Ontology (SO). Homepage at http://www.sequenceontology.org/. Browse at http://www.ebi.ac.uk/ontology-lookup/browse.do?ontName=SO
Parent class: *ControlledVacabulary*
Properties: *xref*, term comment
Class Diagram:



TissueVocabulary

Definition: A reference to the BRENDA (BTO). Homepage at http://www.brenda-enzymes.info/. Browse at http://www.ebi.ac.uk/ontology-lookup/browse.do?ontName=BTO
Parent class: *ControlledVacabulary*

Properties: *xref*, term comment
Class Diagram:



EntityFeature subclasses

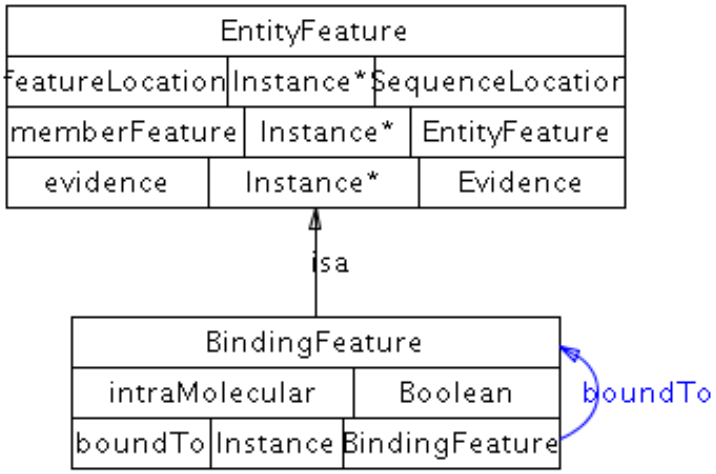
BindingFeature

Definition: An entity feature that represent the bound state of a physical entity. A pair of binding features represents a bond.
Comment: A physical entity in a molecular complex is considered as a new state of an entity as it is structurally and functionally different. Binding features provide facilities for describing these states. Similar to other features, a molecule can have bound and not-bound states.
Usage Note: Typically, binding features are present in pairs, each describing the binding characteristic for one of the interacting physical entities. One exception is using a binding feature with no paired feature to describe any potential binding. For example, an unbound receptor can be described by using a "not-feature" property with an unpaired binding feature as its value. *featureLocationType* and *featureLocation* allows annotating the binding location.
IntraMolecular property should be set to "true" if the bond links two parts of the same molecule. A pair of binding features are still used where they are owned by the same physical entity.
If the binding is due to the covalent interactions, for example in the case of lipoproteins, **CovalentBindingFeature** subclass should be used instead of this class.
Parent class: *EntityFeature*
Properties: *bindsTo*, intraMolecular, comment, *evidence*, *featureLocation*, *featureLocationType*, *memberFeature*

bindsTo – (0 or more object:BindingFeature) The corresponding feature that defines a binding relationship between two physicalEntity instances. They do not have to be in the same complex, but if they are, then this can be used to the complex topology. This property is symmetric.

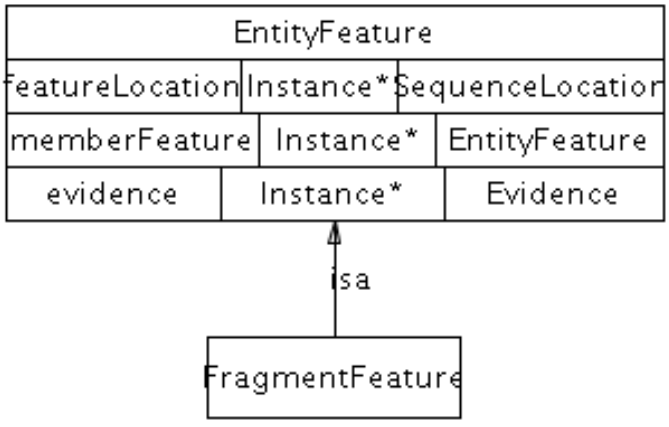
intramolecular – true if this is an intramolecular binding event, false otherwise.

Class Diagram:



FragmentFeature

Definition: Represents the results of a cleavage event for a biological sequence, protein, DNA or RNA.
Examples: A protein with a single cleavage site that converts the protein into two fragments, a protein with two cleavage sites that removes an internal sequence, such as an intein, or a cleavage of a circular sequence, such as a plasmid..
Parent class: *EntityFeature*
Properties: comment, *evidence*, *featureLocation*, *featureLocationType*, *memberFeature*
Class Diagram:



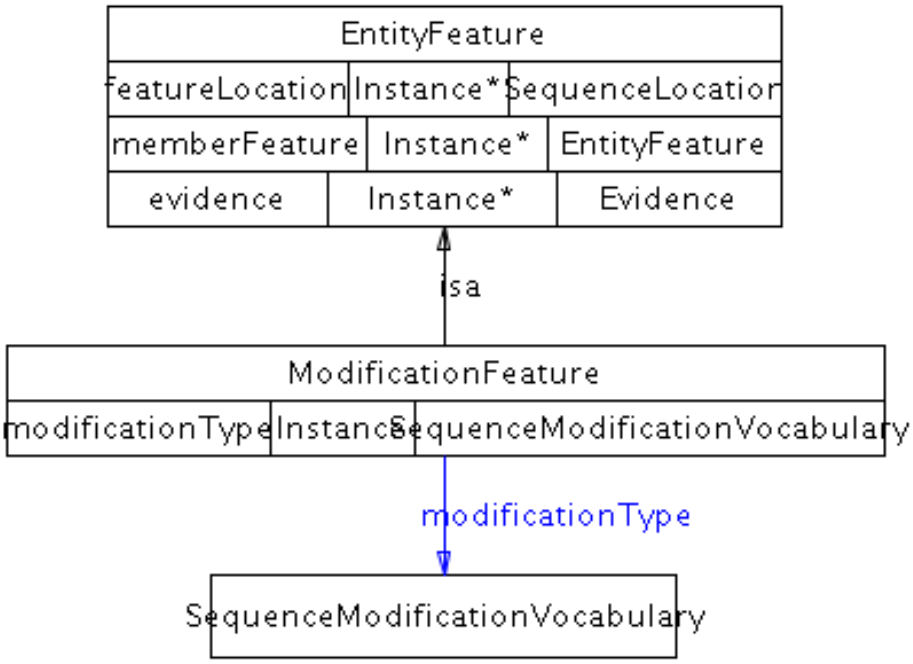
ModificationFeature

Definition: A covalently modified feature on a sequence, relevant to an interaction, such as a post-translational modification. The difference

between this class and BindingFeature is that this is covalent and BindingFeature is non-covalent.
Examples: A phosphorylation feature on a protein that enables the binding of an SH2 domain.
Usage Note: The added groups should be simple and stateless, such as phosphate or methyl groups and are captured by the modificationType controlled vocabulary. In other cases, such as covalently linked proteins, use CovalentBindingFeature instead.
Parent class: *EntityFeature*
Properties: *modificationType*, comment, *evidence*, *featureLocation*, *featureLocationType*, *memberFeature*

modificationType - (0 or 1 object:SequenceModificationVocabulary) Description and classification of the feature.

Class Diagram:



EntityReference subclasses

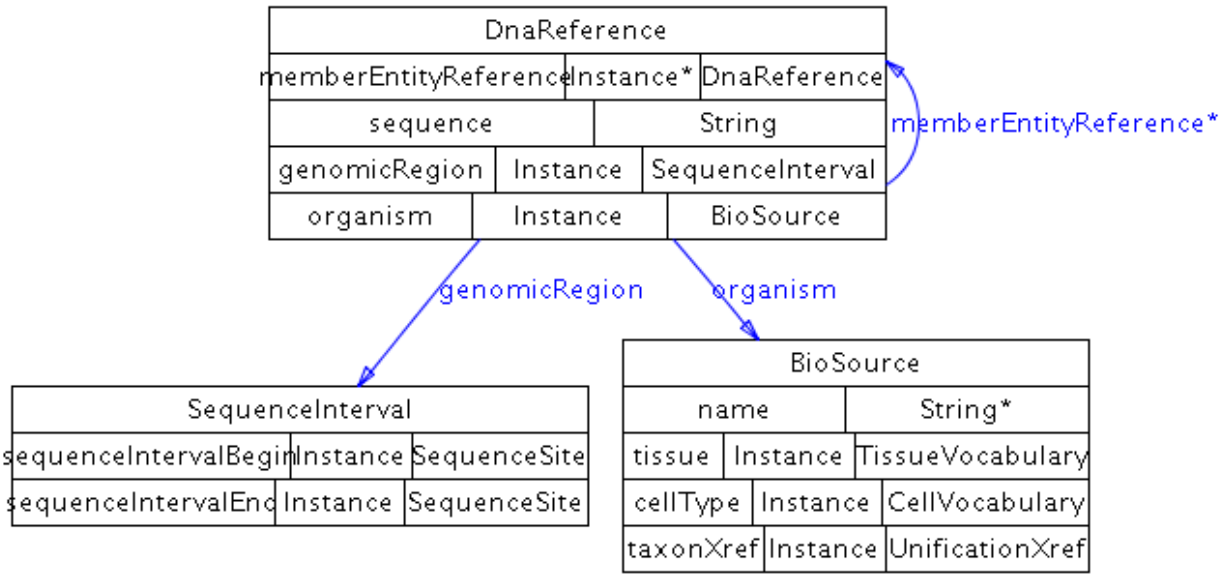
DNAReference
Definition: Used to store shared information about a set of related DNA molecules, such as name, sequence, genomic location, organism and database references. Conceptually, it is a grouping of several DNA molecules across different contexts and molecular states that share common physical properties and are often named and treated by biologists as a single entity with multiple states. Members of the grouping can differ, for example, in cellular location, sequence features, SNPs, other mutations and bound partners.

Comments: This is not a reference gene. A gene can possibly span multiple DNA molecules, sometimes far across a chromosome if regulatory regions are included. Further, a gene is not necessarily encoded by DNA (it could be encoded by RNA).
Parent class: *EntityReference*
Properties: *memberEntityReference*, *organism*, sequence, comment, *entityFeature*, *entityReferenceType*, *evidence*, name, *xref*

organism - (0 or 1 object:BioSource) An organism, e.g. 'Homo sapiens'. This is the organism that the DNA molecule is found in. Sequence-based entities (DNA, protein, RNA) may contain an xref to a sequence database that contains organism information, in which case the information should be consistent with the value for this property.

sequence - Polymer sequence in uppercase letters. For DNA, usually A,C,G,T letters representing the nucleosides of adenine, cytosine, guanine and thymine, respectively.

Class Diagram:



DNARegionReference

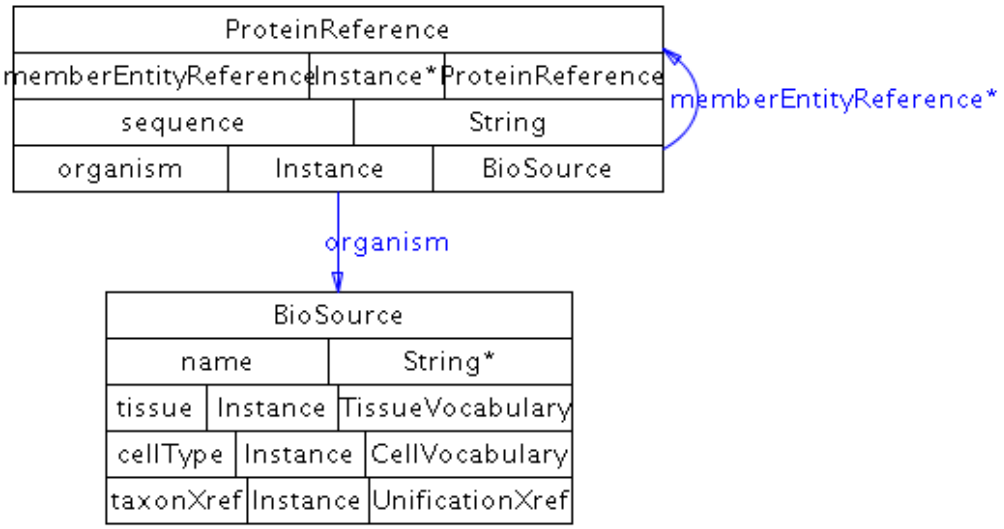
Definition: A DNARegion reference is a grouping of several DNARegion entities that are common in sequence and genomic position. Members can differ in cellular location, sequence features, SNPs, mutations and bound partners
Comments: This is not a reference gene, see DNAReference.
Parent class: *EntityReference*
Properties: *absoluteRegion*, *containerEntityReference*, *memberEntityReference*, *regionType*, subRegion, sequence, comment,

entityFeature, *entityReferenceType*, *evidence*, name, *xref*

ProteinReference

Definition: Used to store shared information about a set of related protein molecules encoded by the same gene, such as name, sequence, organism and database references. Conceptually, it is a grouping of several protein molecules across different contexts and molecular states that share common physical properties and are often named and treated by biologists as a single entity with multiple states. Members of the grouping can differ, for example, in cellular location, sequence features and bound partners. Conformational states (such as open and closed) are not covered.
Parent class: *EntityReference*
Properties: *memberEntityReference*, *organism*, sequence, comment, *entityFeature*, *entityReferenceType*, *evidence*, name, *xref*

Class Diagram:

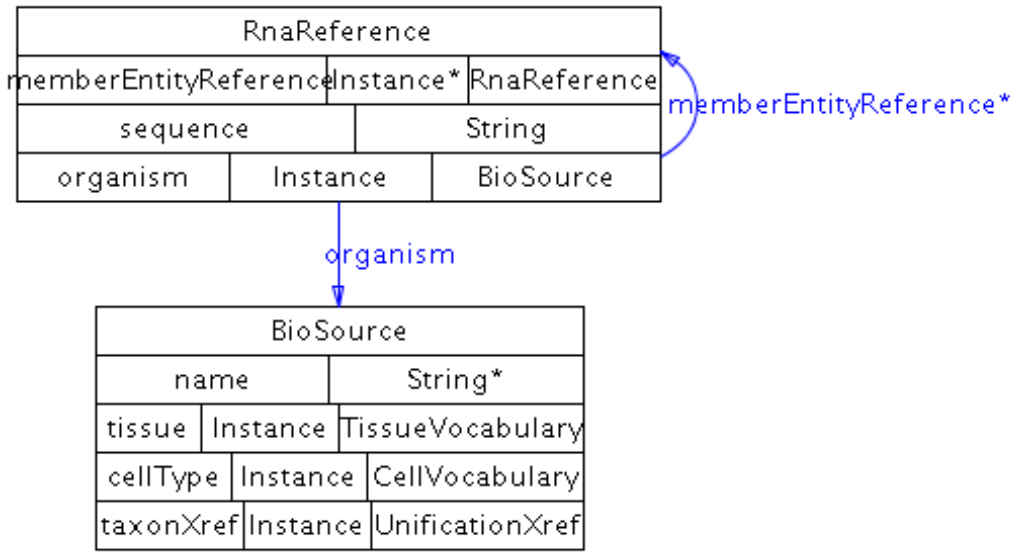


RNAReference

Definition: Used to store shared information about a set of related RNA molecules usually encoded by the same gene, such as name, sequence, organism and database references. Conceptually, it is a grouping of several RNA molecules across different contexts and molecular states that share common physical properties and are often named and treated by biologists as a single entity with multiple states. Members of the grouping can differ, for example, in cellular location, sequence features and bound partners. Currently conformational states (such as hairpin) are not covered.
Parent class: *EntityReference*
Properties: *memberEntityReference*, *organism*, sequence, comment,

entityFeature, *entityReferenceType*, *evidence*, name, *xref*

Class Diagram:



RNARegionReference

Definition: Used to store shared information about a set of related RNA entityRefernces. The *containerEntityReference* objectProperty connects to common properties, organism and any common features in the RNAReference.
Comments: This is not a reference gene, see RNAReference.
Parent class: *EntityReference*
Properties: *absoluteRegion*, *containerEntityReference*, *memberEntityReference*, *regionType*, subRegion, sequence, comment, *entityFeature*, *entityReferenceType*, *evidence*, name, *xref*

SmallMoleculeReference

Used to store shared information about a set of related small molecules, such as name, chemical structure, and database references. Covalent modifications of small molecules are not considered as state changes, as is the case for e.g. proteins. Instead, covalent modifications create different molecules.
Parent class: *EntityReference*
Properties: chemicalFormula, *memberEntityReference*, molecularWeight, *structure*, comment, *entityFeature*, *entityReferenceType*, *evidence*, name, *xref*

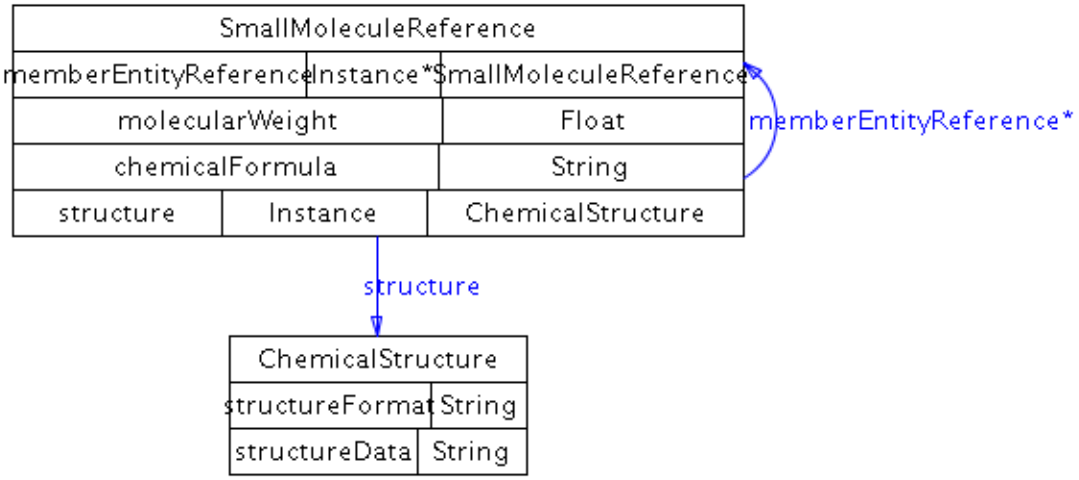
chemicalFormula – The chemical formula of the small molecule. Note: chemical formula can also be stored in the *structure* property (in CML). In case of disagreement between the value of this property and that in

the CML file, the CML value takes precedence.

molecularWeight – Defines the molecular weight of the molecule, in Daltons.

structure - (0 or 1 object:ChemicalStructure) Defines the chemical structure and other information about this molecule, using an instance of class ChemicalStructure.

Object Properties Diagram:



PathwayStep subclasses

BiochemicalPathwayStep

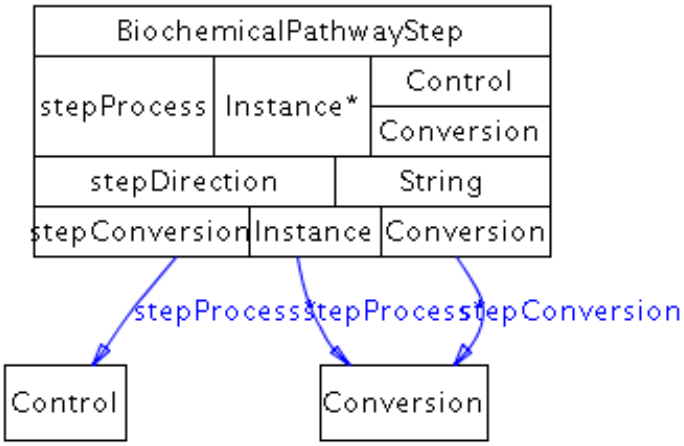
Definition: Imposes ordering on a step in a biochemical pathway, if not clear from the reaction itself. A biochemical reaction can be reversible by itself, but can be physiologically directed in the context of a pathway, for instance due to flux of reactants and products. Only one conversion interaction can be ordered at a time, but multiple catalysis or modulation instances can be part of one step. This is intended for use when the direction of the reaction in the pathway is not clear and should not be used otherwise.
Comment: If 'direction' of the Catalysis instance contained in the step is "PHYSIOL-LEFT-TO-RIGHT", then stepDirection of BiochemicalPathwayStep is blank (unknown, unspecified) or LEFT-TO-RIGHT. If stepDirection of BiochemicalPathwayStep is not empty then 'direction' of the Catalysis instance is either blank, "REVERSIBLE" or "PHYSIOL-LEFT-TO-RIGHT".
Parent class: *PathwayStep*
Properties: *stepConversion*, stepDirection, *stepProcess*, comment, *evidence*, *nextStep*

stepConversion - (0 or 1 object:Conversion) The central conversion process that takes place at this step of the biochemical pathway. This is a sub-property of *stepProcess*, which means any value of this property is automatically a value of the *stepProcess* property.

stepDirection - (REVERSIBLE, RIGHT-TO-LEFT, LEFT-TO-RIGHT) Direction of the conversion in this particular pathway context, specifically the conversion in *stepConversion*. This orders the direction of the conversion instance (usually a biochemical reaction) in the *stepConversion* property. This should not be left blank and should not conflict with the direction in the conversion in the *stepConversion* property.

stepProcess – (0 or more object:Control) Control or conversion instances that are part of this pathway step. Only one conversion instance is part of this step, specified in the *stepConversion* property (which automatically becomes a value of this property), but multiple Control instances, such as catalysis or modulations can be included.

Class Diagram:



Sequence Location subclasses

For the purposes of the following BioPAX classes, sequences start at index 1 and sequence interval coordinates are inclusive.

SequenceInterval

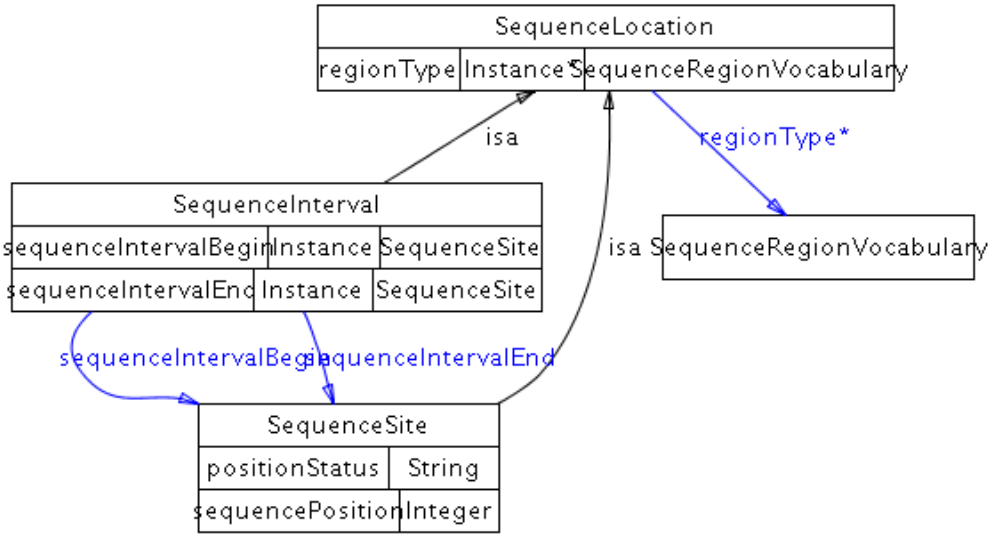
Definition: Describes an interval on a sequence. Interval is defined as an ordered pair of **SequenceSites**. All of the sequence from the begin site to the end site (inclusive) is described, not any subset.
Parent class: *SequenceLocation*
Properties: *sequenceIntervalBegin*, *sequenceIntervalEnd*, comment

sequenceIntervalBegin - (0 or 1 object:SequenceSite) The begin

position of a sequence interval.

sequenceIntervalEnd - (0 or 1 object:SequenceSite) The end position of a sequence interval.

Class Diagram:



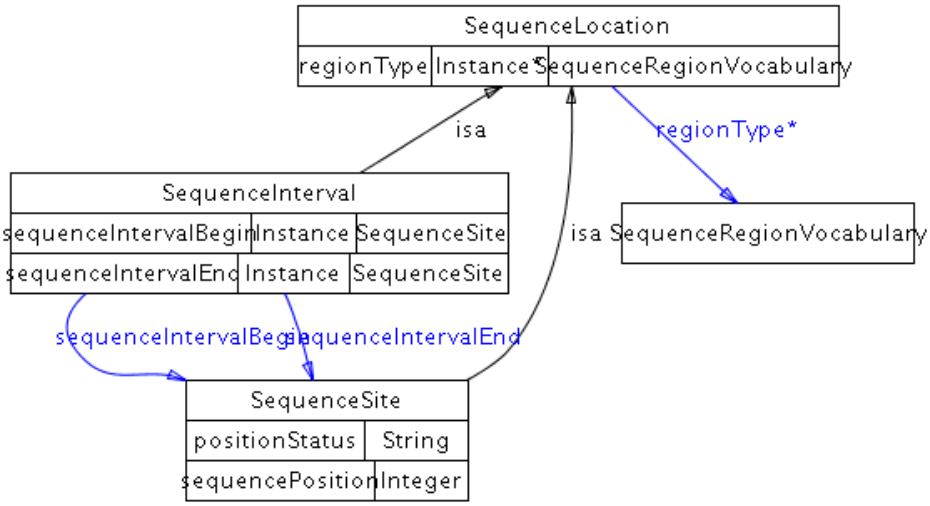
SequenceSite

Definition: Describes a site on a sequence, i.e. the position of a single nucleotide or amino acid.
Usage Note: A sequence site is always defined based on the reference sequence of the owning entity. For DNARegion and RNARegion it is relative to the region itself not the genome or full RNA molecule.
Parent class: *SequenceLocation*
Properties: positionStatus, sequencePosition, comment

positionStatus - The confidence status of the sequence position. This could be:
EQUAL: The sequence position is known to be at the *sequencePosition*.
GREATER-THAN: The site is greater than the *sequencePosition*.
LESS-THAN: The site is less than the *sequencePosition*.

sequencePosition - The integer listed gives the position. The first base or amino acid is position 1. In combination with the numeric value, the property ' *positionStatus*' allows to express fuzzy positions, e.g. 'less than 4'.

Class Diagram:



Xref subclasses

PublicationXref

Definition: An xref that defines a reference to a publication such as a journal article, book, web page, or software manual. The reference may or may not be in a database, although references to PubMed are preferred when possible. The publication should make a direct reference to the instance it is attached to.
Comment: Publication xrefs should make use of PubMed IDs wherever possible. The *db* property of an xref to an entry in PubMed should use the string “PubMed” and not “MEDLINE”.
Examples: PubMed:10234245
Parent class: *Xref*
Properties: author, source, title, url, year, comment, db, dbVersion, id, idVersion

The following properties may be used when the *db* and *id* fields cannot be used, such as when referencing a publication that is not in PubMed. The *url* property should not be used to reference publications that can be uniquely referenced using a *db*, *id* pair. The main reason for this is that it is expected that *db*, *id* pairs are more stable than URLs.

author - The authors of this publication, one per property value.

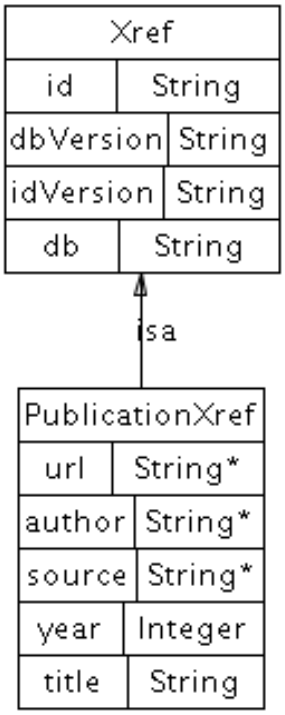
source - The source in which the reference was published, such as: a book title, or a journal title and volume and pages.

title - The title of the publication.

url - The URL at which the publication can be found, if it is available through the Web.

year - The year in which this publication was published.

Class Diagram:



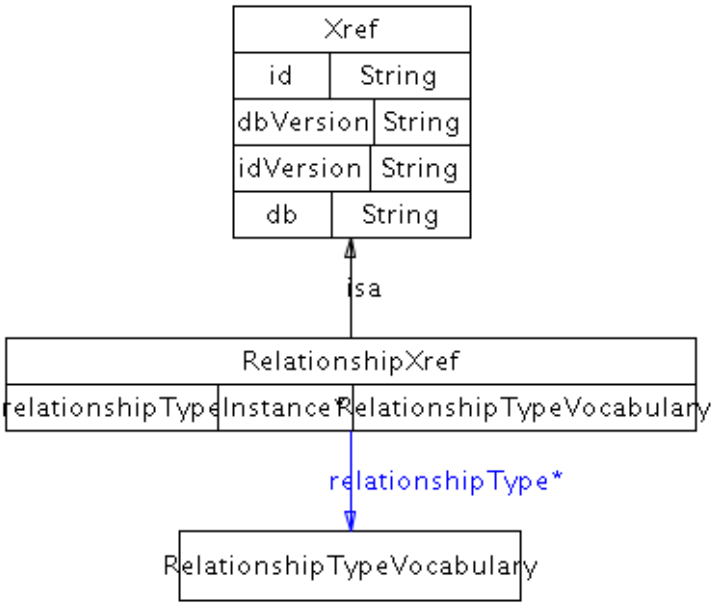
RelationshipXref

Definition: An xref that defines a reference to an entity in an external resource that does not have the same biological identity as the referring entity.
Examples: A link between a gene G in a BioPAX data collection, and the protein product P of that gene in an external database. This is not a unification xref because G and P are different biological entities (one is a gene and one is a protein). Another example is a relationship xref for a protein that refers to the Gene Ontology biological process, e.g. 'immune response,' that the protein is involved in.
Parent class: *Xref*
Properties: *relationshipType*, comment, db, dbVersion, id, idVersion

relationshipType - (0 or more object:RelationshipTypeVocabulary) This property names the type of relationship between the BioPAX object linked from, and the external object linked to, such as 'gene of this protein', or 'protein with similar sequence'. This should reference to the PSI Molecular Interaction ontology (MI) Cross Reference type. Homepage at http://www.psidev.info/. Browse at http://www.ebi.ac.uk/ontology-lookup/browse.do?ontName=MI&termId=MI%3A0353&termName=cross-reference%20type. See the section on controlled vocabularies in Section 4 for

more information.

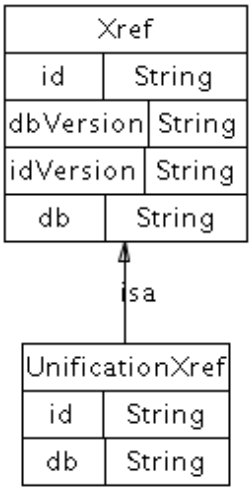
Class Diagram:



UnificationXref

Definition: A **unificationXref** defines a reference to an entity in an external resource that has the same biological identity as the referring entity[16]. For example, to link from a database record, C, describing a chemical compound in a BioPAX data collection to a record, C', describing the same chemical compound in an external database, use a **unificationXref** since records C and C' describe the same biological identity. Generally, unification xrefs should be used whenever possible, since they aid data integration, although there are cases where they might not be useful, such as application-to-application data exchange.
Comment: **unificationXref** in physical entities are essential for data integration, but are less important in interactions. This is because unification xrefs on the physical entities in an interaction can be used to compute the equivalence of two interactions of the same type. An xref in a protein pointing to a gene, e.g. in the Entrez Gene database[17], would not be a unification xref since the two entities do not have the same biological identity (one is a protein, the other is a gene). Instead, this link should be a captured as a relationship xref[16]. References to an external controlled vocabulary term within the **ControlledVocabulary** class should use a unification xref where possible (e.g. GO:0005737)
Examples: An xref in a protein instance pointing to an entry in the UniProt database, and an xref in an RNA instance pointing to the corresponding RNA sequence in the RefSeq database.
Parent class: *Xref*
Properties: db, id, dbVersion, idVersion, comment

Class Diagram:



Summary of BioPAX Class Structure

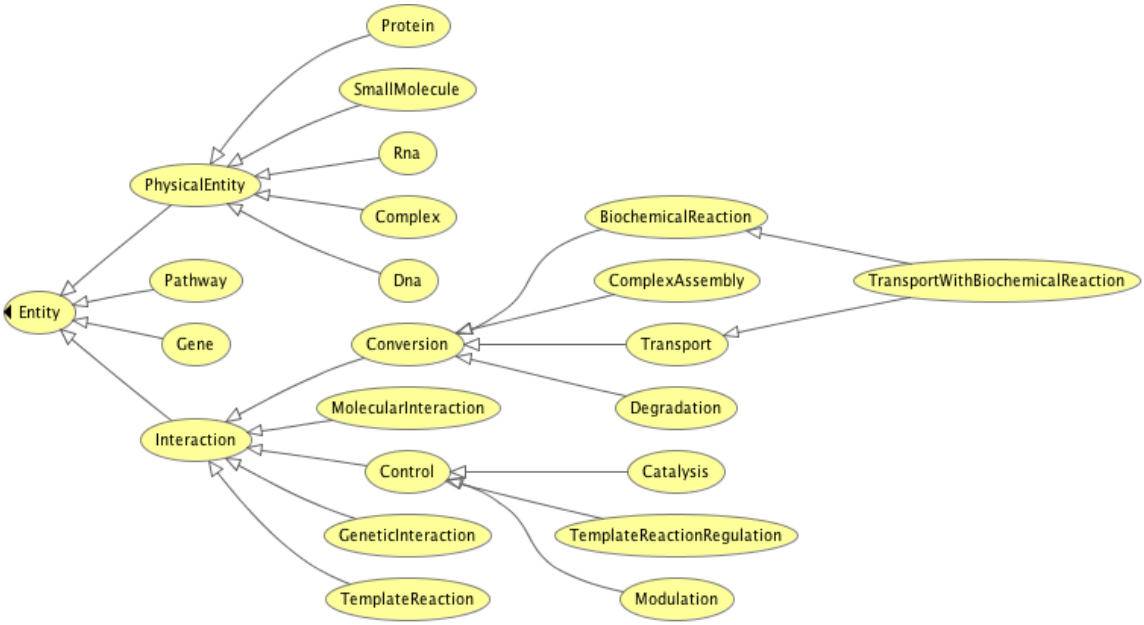


Figure 1 Entity Class Hierarchy

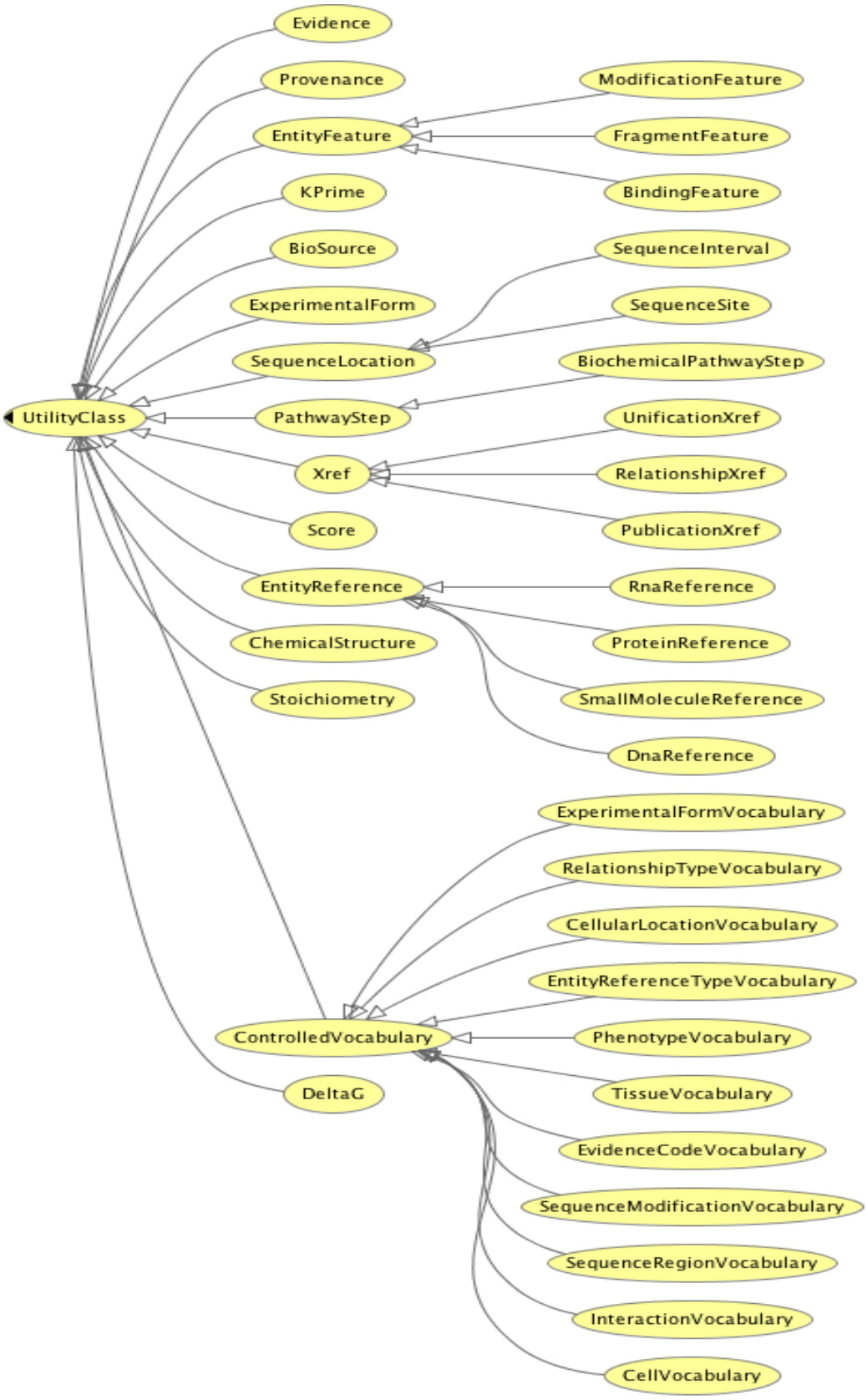


Figure 2 Utility Class Hierarchy

3 BioPAX Object Properties

This section provides an overview of the BioPAX Level 3 object properties. Formal definitions are found in the BioPAX Level 3 OWL document (http://www.biopax.org/release/biopax-level3.owl).

In OWL a property connects two individuals (i.e. they are binary relations between instances). There are dataType properties and object properties. A dataType property is a binary relation between an instance of a class and a literal value. For example, the property **name** of the **physicalEntity** class. An object property is the relation between instances of two classes. This section covers the object Properties defined in BioPAX Level 3 and extends the class descriptions and *objectProperty* diagrams given in the previous section. Interspersed in this section are class diagrams that show the classes that are related by owl object properties.

Object properties are described in alphabetical order.

Basic Definitions
The terms used in the descriptions that follow are defined here.

subProperty
When a property has subProperty characteristics the sub properties in the hierarchy describe more specific relationships between individuals.

Equivalent
Is used to state when two properties are equivalent, or synonymous. There are no examples in BioPAX, however owl:equivalentObjectProperty is widely used in data integration.

Disjoint
Is used to state when a set of properties are disjoint, i.e. they share no characteristics in common.

Range
Range limits the individuals that the property may have as its values.

Domain
Domain limits the individuals to which a property may be applied.

inverseFunctional
If a property is stated to be inverseFunctional then the inverse of the property has at most one value for the individual.

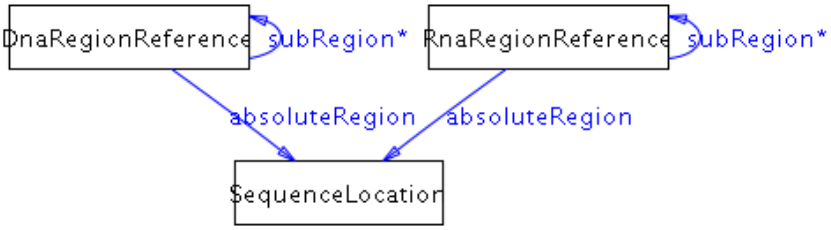
Functional
No more than one value for each individual (can be no values).

Transitive
Given the individuals x and y related by the property P (xPy) and the individuals y and z also related by the property P (yPz) , if the property P is transitive then the triple xPz can be inferred.

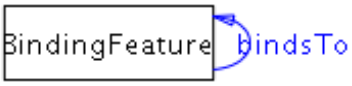
Symmetric
Given the individuals aSb, if S is a symmetric property then the triple bSa can be inferred.

Level 3 ObjectProperties

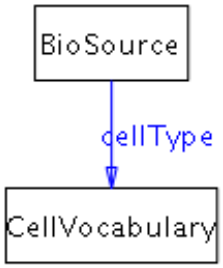
absoluteRegion
Definition: Absolute location as defined by the referenced sequence database record. E.g. an operon has a absolute region on the DNA molecule referenced by the UnificationXref
Domain*: DNARegionReference*, RNARegionReference
Range*: SequenceLocation*OWLProperty: Functional
Object Property Diagram:



bindsTo
Definition: A binding feature represents a "half" of the bond between two entities. This property points to another binding feature which represents the other half. The bond can be covalent or non-covalent.
Domain: BindingFeature
Range: BindingFeatureOWLProperty: Functional, InverseFunctional, Symmetric
Object Property Diagram:

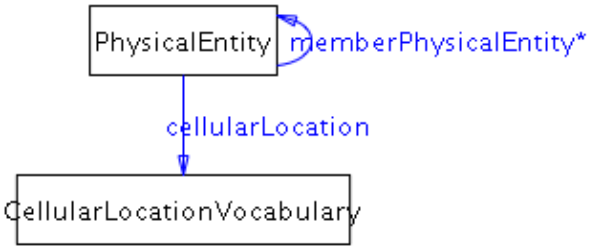


cellType
Definition: A cell type, e.g. 'HeLa'. This should reference a term in a controlled vocabulary of cell types. Best practice is to refer to OBO Cell Ontology. http://www.obofoundry.org/cgi-bin/detail.cgi?id=cell
Domain: BioSource
Range: *CellVocabulary***OWLProperty:** Functional
Object Property Diagram:



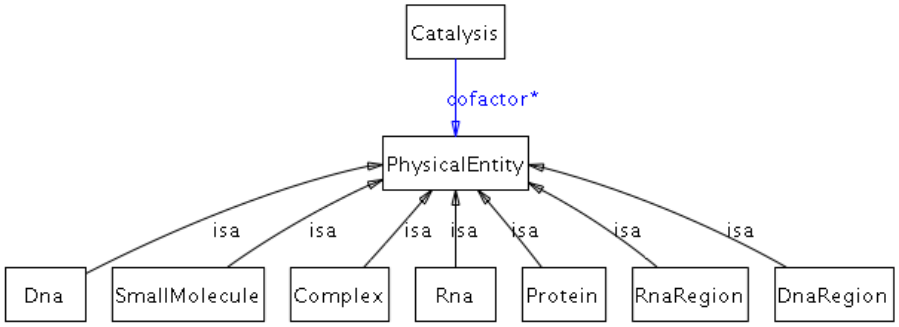
cellularLocation

Definition: A cellular location, e.g. 'cytoplasm'. This should reference a term in the Gene Ontology Cellular Component ontology. The location referred to by this property should be as specific as is known. If an interaction is known to occur in multiple locations, separate interactions (and physicalEntities) must be created for each different location. If the location of a participant in a complex is unspecified, it may be assumed to be the same location as that of the complex.
Domain: PhysicalEntity
Range: CellularLocationVocabularyOWLProperty: Functional
Object Property Diagram:



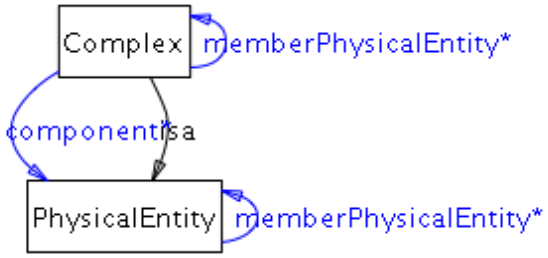
coFactor

Definition: Any cofactor(s) or coenzyme(s) required for catalysis of the conversion by the enzyme. COFACTOR is a sub-property of PARTICIPANTS
Domain: Catalysis
Range: PhysicalEntity
SuperProperty: *participant*
Object Property Diagram:



component

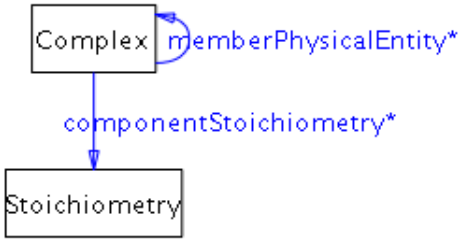
Definition: An instances that describes the composition of a complex.
Domain: Complex
Range: PhysicalEntityOWLProperty: InverseFunctional



Object Property Diagram:

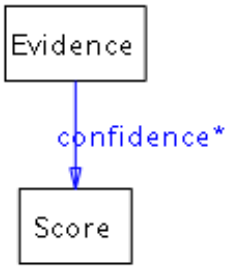
componentStoichiometry

Definition: The stoichiometry of components in a complex.
Domain: *Complex*
Range: StoichiometryObject Property Diagram:



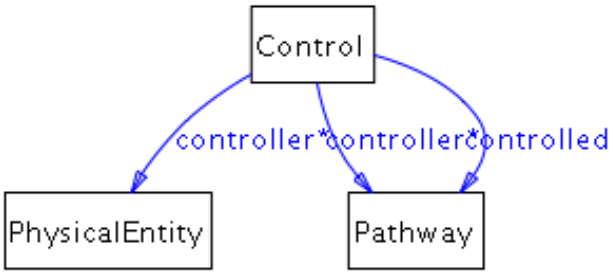
confidence

Definition: Confidence in the containing instance. Usually a statistical measure.
Domain: Evidence
Range: ScoreObject Property Diagram:



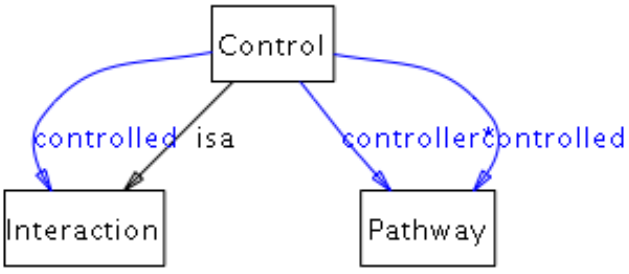
controller

Definition: The controlling entity, e.g., in a biochemical reaction, an enzyme is the controlling entity of the reaction. CONTROLLER is a sub-property of PARTICIPANTS.
Domain:Control
Range: PhysicalEntity, *Pathway*
SuperProperty: *participant*
Object Property Diagram:



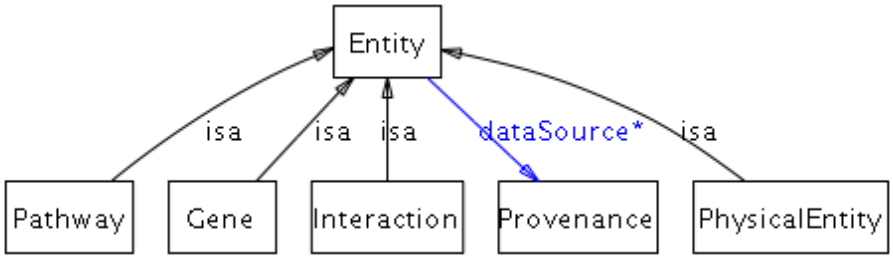
controlled

Definition: The entity that is controlled, e.g., in a biochemical reaction, the reaction is controlled by an enzyme. CONTROLLED is a sub-property of PARTICIPANTS.
Domain: Control
Range: Interaction, *Pathway*
SuperProperty: *participant*
Object Property Diagram:



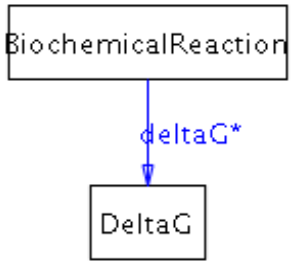
dataSource

Definition: Confidence in the containing instance. Usually a statistical measure.
Domain: Entity
Range: ProvenanceObject Property Diagram:



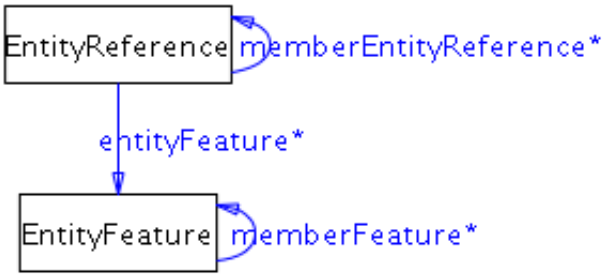
deltaG

Definition: For biochemical reactions, this property refers to the standard transformed Gibbs energy change for a reaction written in terms of biochemical reactants (sums of species), delta-G. Since Delta-G can change based on multiple factors including ionic strength and temperature a reaction can have multiple DeltaG values.
Domain: *BiochemicalReaction*
Range: DeltaGObject Property Diagram:



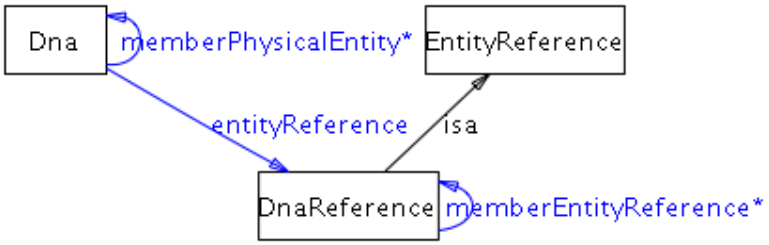
entityFeature

Definition: Variable features that are observed for this entity - such as known PTM or methylation sites and non-covalent bonds.
Domain: EntityReference
Range: EntityFeatureOWLProperty: InverseFunctional
Object Property Diagram:



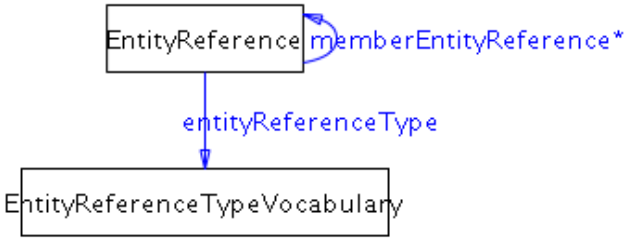
entityReference

Definition: Reference entity for this physical entity.
Domain: Protein, *DNA, RNA, SmallMolecule, DNARegion, RNARegion*
Range: EntityReferenceOWLProperty: Functional
Object Property Diagram:



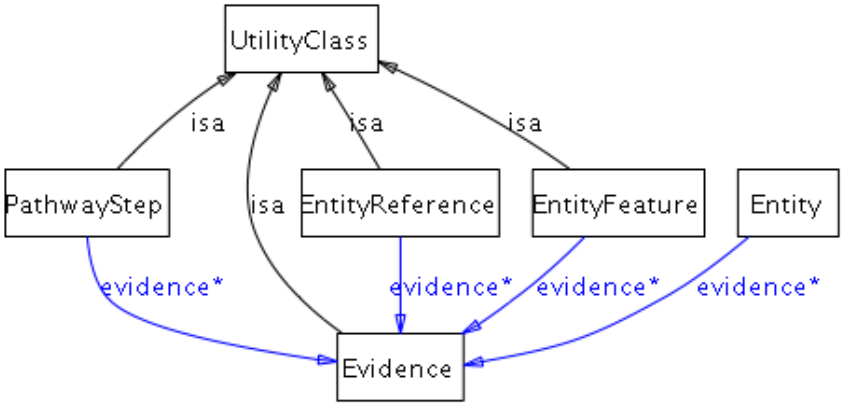
entityReferenceType

Definition: A controlled vocabulary term that is used to describe the type of grouping such as homology or functional group.
Domain: EntityReference
Range: EntityReferenceTypeVocabularyOWLProperty: Functional
Object Property Diagram:



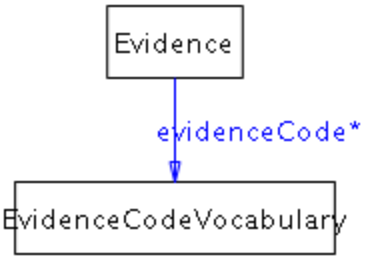
evidence

Definition: Scientific evidence supporting the existence of the entity as described.
Domain: PathwayStep, *EntityFeature*, *EntityReference*, Entity
Range: Evidence Object Property Diagram:



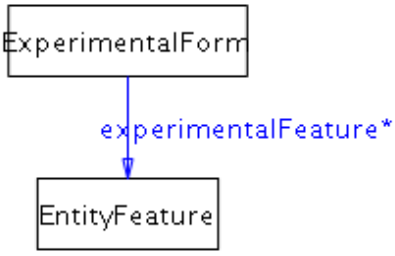
evidenceCode

Definition: A pointer to a term in an external controlled vocabulary, such as the GO, PSI-MI or BioCyc evidence codes, that describes the nature of the support, such as 'traceable author statement' or 'yeast two-hybrid'.
Domain: Evidence
Range:EvidenceCodeVocabularyObject Property Diagram:



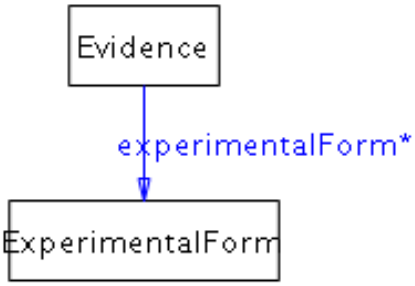
experimentalFeature

Definition: A feature of the experimental form of the participant of the interaction, such as a protein tag. It is not expected to occur in vivo or be necessary for the interaction.
Domain: ExperimentalForm
Range: EntityFeatureObject Property Diagram:



experimentalForm

Definition: A feature of the experimental form of the participant of the interaction, such as a protein tag. It is not expected to occur in vivo or be necessary for the interaction.
Domain: Evidence
Range: ExperimentalFormObject Property Diagram:

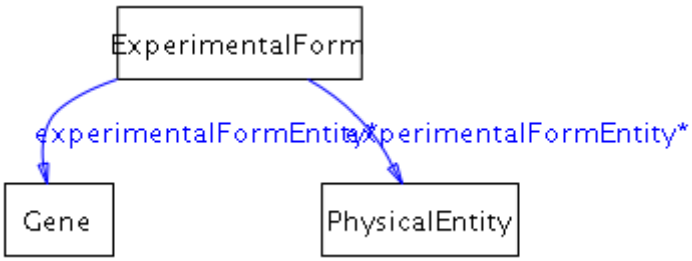


experimentalFormDescription

Definition: Descriptor of this experimental form from a controlled vocabulary.
Domain: ExperimentalForm
Range: ExperimentalFormVocabulary

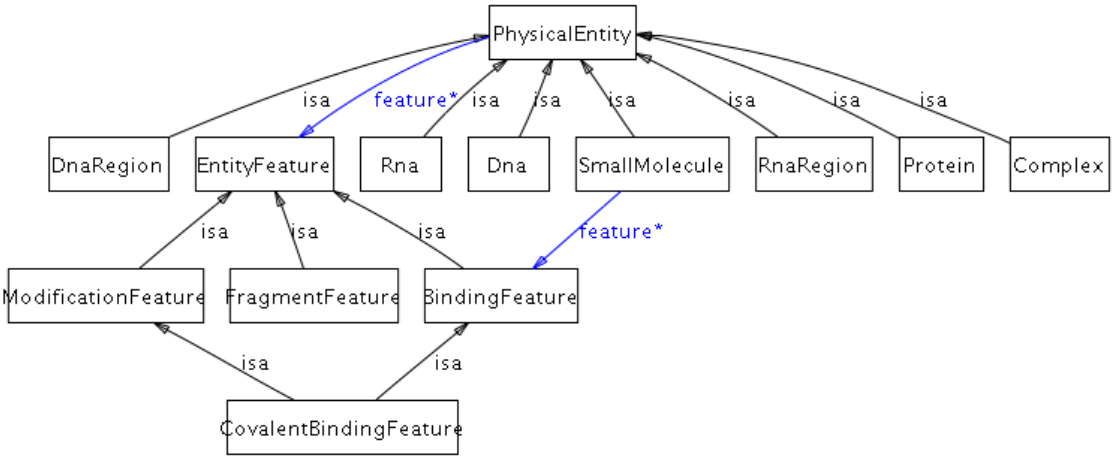
experimentalFormEntity

Definition: The gene or physical entity that this experimental form describes.
Domain: ExperimentalForm
Range: PhysicalEntity, *Gene***Object Property Diagram:**



feature

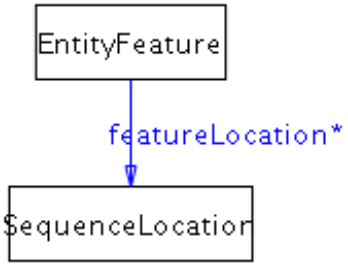
Definition: Sequence features of the owner physical entity.
Domain: PhysicalEntity
Range: *EntityFeature***Object Property Diagram:**



featureLocation

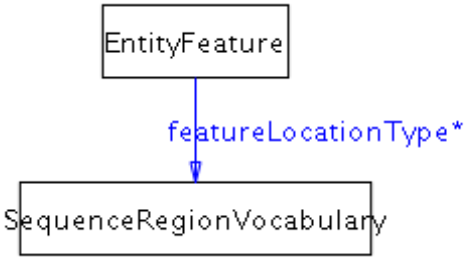
Definition: Location of the feature on the sequence of the interactor. One feature may have more than one location, used e.g. for features which involve sequence positions close in the folded, three-dimensional state of a protein, but non-continuous along the sequence.

Domain: EntityFeature
Range: *SequenceLocation***Object Property Diagram:**



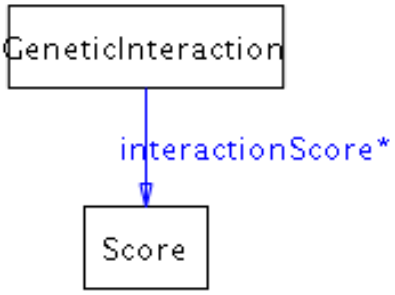
featureLocationType

Definition: Sequence features of the owner physical entity.
Domain: EntityFeature
Range: *SequenceRegionVocabulary***Object Property Diagram:**



interactionScore

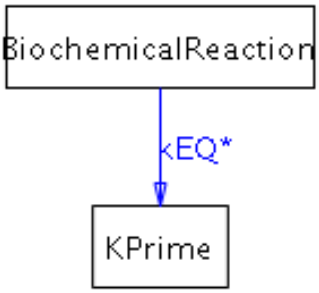
Definition: The score of an interaction e.g. a genetic interaction score.
Domain: GeneticInteraction
Range: *Score***Object Property Diagram:**



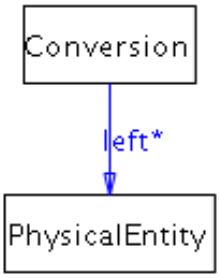
interactionType

Definition: The score of an interaction e.g. a genetic interaction score.
Domain: Interaction
Range: *InteractionVocabulary*

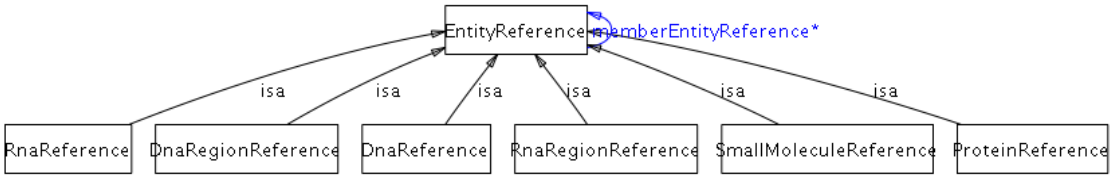
kEQ
Definition: The equilibrium constant for a biochemical reaction.
Domain: BiochemicalReaction
Range: kPrimeObject Property Diagram:



left
Definition: The participants on the left side of the conversion interaction. Since conversion interactions may proceed in either the left-to-right or right-to-left direction, occupants of the LEFT property may be either reactants or products. LEFT is a sub-property of PARTICIPANTS.
Domain: Conversion
Range: PhysicalEntity
SuperProperty: *participant*
Object Property Diagram:

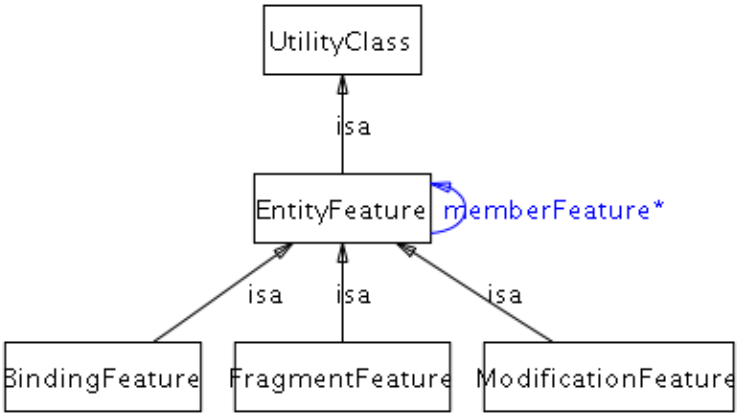


memberEntityReference
Definition: An entity reference that qualifies for the definition of this group. For example a member of a PFAM protein family.
Domain: EntityReference
Range: EntityReferenceOWLProperty: Transitive
Object Property Diagram:



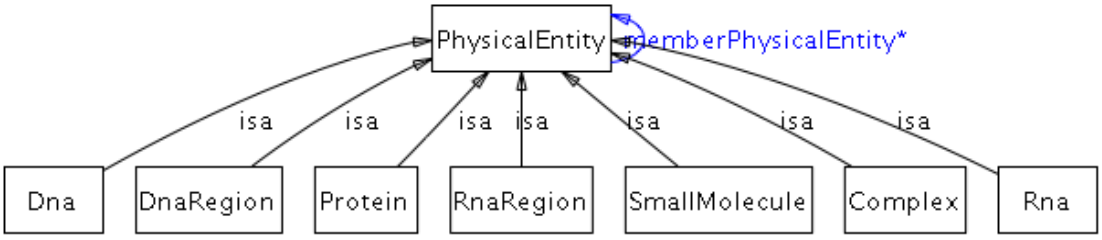
memberFeature

Definition: An entity reference that qualifies for the definition of this group. For example a member of a PFAM protein family.
Domain: EntityFeature
Range: EntityFeatureOWLProperty: Transitive
Object Property Diagram:



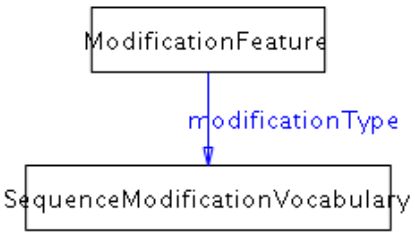
memberPhysicalEntity

Definition: This property stores the members of a generic physical entity.
Domain: PhysicalEntity
Range: PhysicalEntityOWLProperty: Transitive
Object Property Diagram:



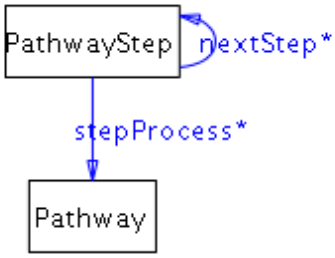
modificationType

Definition: Description and classification of the feature.
Domain: ModificationFeature
Range: SequenceModificationVocabulary**OWLProperty:** Functional
Object Property Diagram:



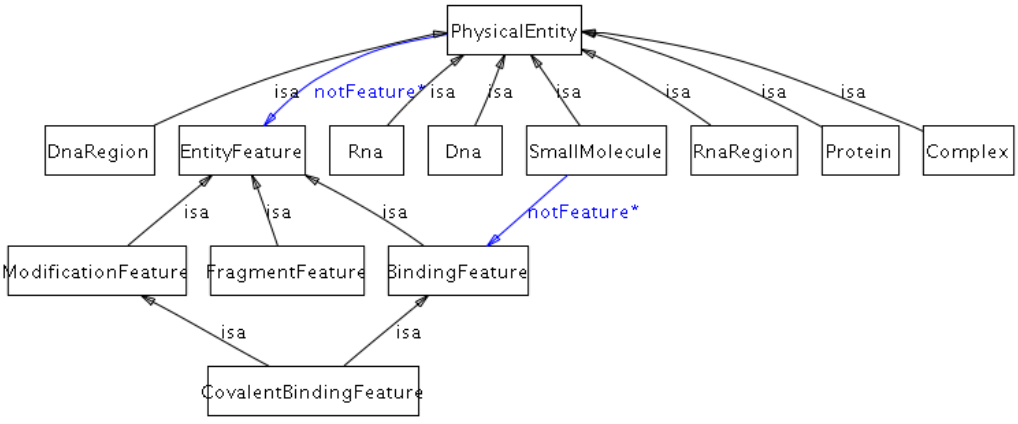
nextStep

Definition: The next step(s) of the pathway. Contains zero or more pathwayStep instances. If there is no next step, this property is empty. Multiple pathwayStep instances indicate pathway branching.
Domain: PathwayStep
Range: *PathwayStep***Object Property Diagram:**



notFeature

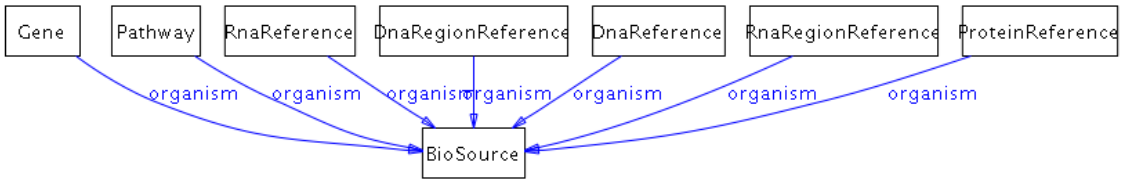
Definition: Sequence features where the owner physical entity has a feature it is known not to have.
Domain: PhysicalEntity
Range: EntityFeature
Object Property Diagram:



organism

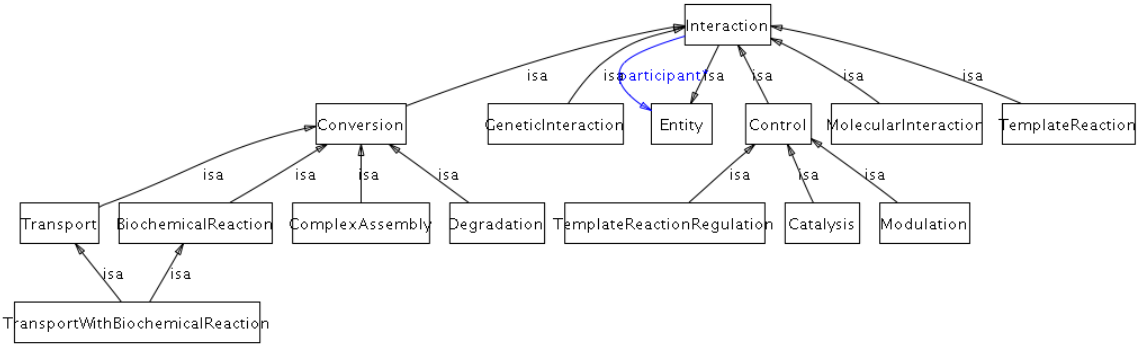
Definition: An organism, e.g. 'Homo sapiens'. This is the organism that the entity is found in. Pathways may not have an organism

associated with them, for instance, reference pathways from KEGG. Sequence-based entities (DNA, protein, RNA) may contain an xref to a sequence database that contains organism information, in which case the information should be consistent with the value for ORGANISM.
Domain: Pathway, *DNAReference, RNAReference, ProteinReference, Gene, DNARegionReference, RNARegionReference*
Range: *BioSource*
Object Property Diagram:



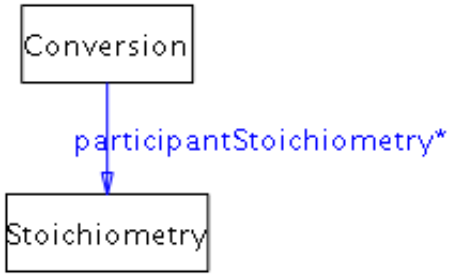
participant

Definition: This property lists the entities that participate in this interaction. For example, in a biochemical reaction, the participants are the union of the reactants and the products of the reaction. This property has a number of sub-properties, such as LEFT and RIGHT used in the biochemicalInteraction class. Any participant listed in a sub-property will automatically be assumed to also be in PARTICIPANTS by a number of software systems, including Protege, so this property should not contain any instances if there are instances contained in a sub-property.
Domain: Interaction
Range: Entity
SubProperties: left, product, right, controller, cofactor, template, controlled
Object Property Diagram:



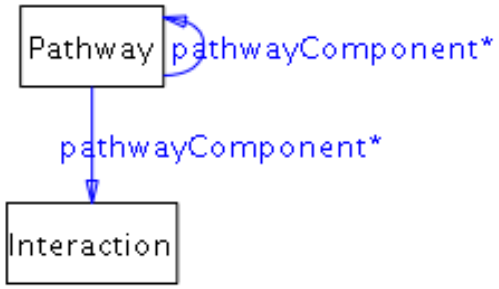
participantStoichiometry

Definition: Stoichiometry of the left and right participants.
Domain: Conversion
Range: StoichiometryObject Property Diagram:



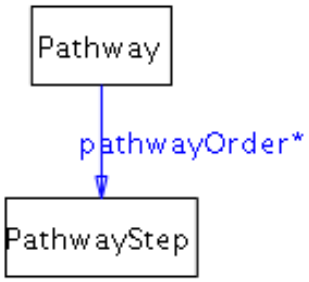
pathwayComponent

Definition: The set of interactions and/or pathwaySteps in this pathway/network. Each instance of the pathwayStep class defines: 1) a set of interactions that together define a particular step in the pathway, for example a catalysis instance and the conversion that it catalyzes; 2) an order relationship to one or more other pathway steps (via the NEXT-STEP property). Note: This ordering is not necessarily temporal - the order described may simply represent connectivity between adjacent steps. Temporal ordering information should only be inferred from the direction of each interaction.
Domain: Interaction
Range: *Interaction, Pathway***Object Property Diagram:**



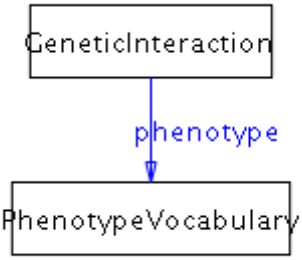
pathwayOrder

Definition:
Domain: Pathway
Range: PathwayStepObject Property Diagram:



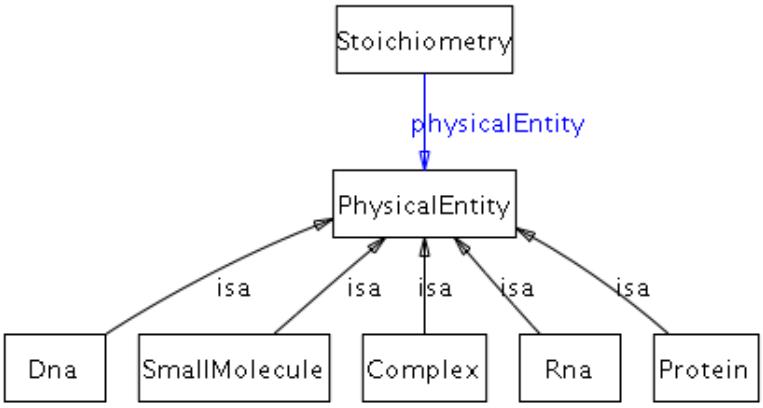
phenotype

Definition: The phenotype quality used to define this genetic interaction e.g. viability.
Domain: GeneticInteraction
Range: PhenotypeVocabularyOWLProperty: Functional
Object Property Diagram:



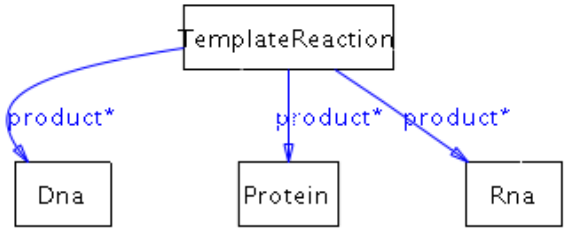
physicalEntity

Definition: The physical entity to be annotated with stoichiometry.
Domain: Stoichiometry
Range: ***PhysicalEntity*****OWLProperty:** Functional
Object Property Diagram:



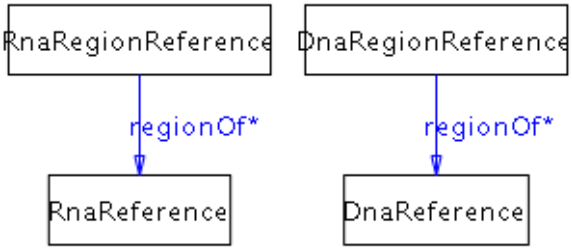
product

Definition: The product of a template reaction.
Domain: TemplateReaction
Range: DNA, *RNA, Protein*
SuperProperty: *participant*
Object Property Diagram:



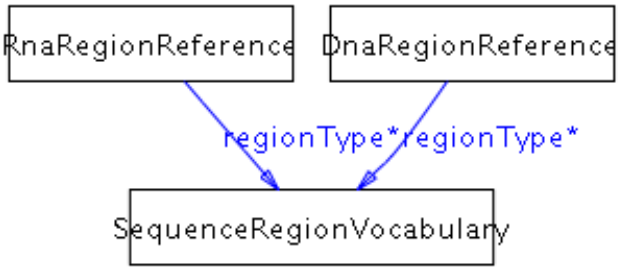
regionOf

Definition: Property pointer to the Parent EntityReference of a Region Reference.
Domain: DNARegionReference, *RNARegionReference*
Range: DNAReference, *RNAReference***Object Property Diagram:**



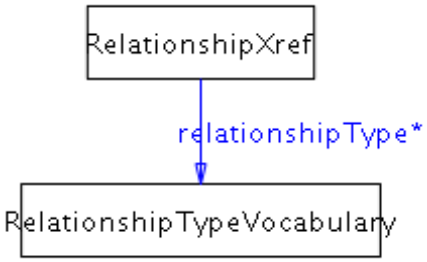
regionType

Definition: A controlled vocabulary term used to describe the type of sequenceRegion of the entity reference, for example: gene, insertion or binding site.
Domain: *RNARegionReference, DNARegionReference*
Range: SequenceRegionVocabulary Object Property Diagram:

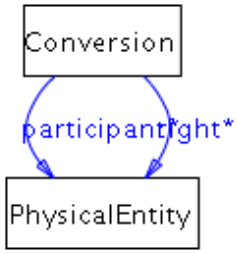


relationshipType

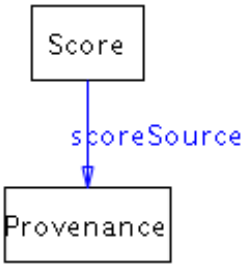
Definition: A controlled vocabulary term used to describe the type of relationship, for example: gene product, isoform, homolog**.**
Domain: RelationshipXref
Range: RelationshipTypeVocabulary Object Property Diagram:



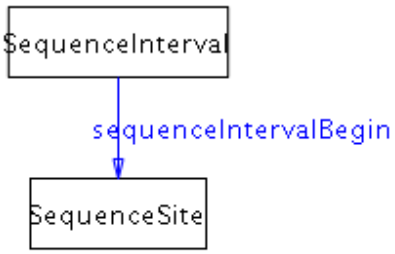
right
Definition: The participants on the right side of the conversion interaction. Since conversion interactions may proceed in either the left-to-right or right-to-left direction, occupants of the RIGHT property may be either reactants or products. RIGHT is a sub-property of PARTICIPANTS.
Domain: Conversion
Range: PhysicalEntity
SuperProperty: *participant*
Object Property Diagram:



scoreSource
Definition: score source pointer.
Domain: Score
Range: ProvenanceOWLProperty: Functional
Object Property Diagram:



sequenceIntervalBegin
Definition: The begin (start) position of a sequence interval.
Domain: SequenceIntervalError: Reference source not foundError: Reference source not found
Range: SequenceSiteOWLProperty: Functional
Object Property Diagram:

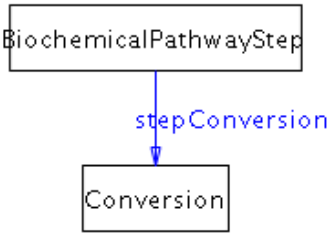


sequenceIntervalEnd

Definition: The end position of a sequence interval.
Domain: SequenceIntervalError: Reference source not foundError: Reference source not found
Range: SequenceSiteOWLProperty: Functional
Object Property Diagram: (as sequenceIntervalBegin)

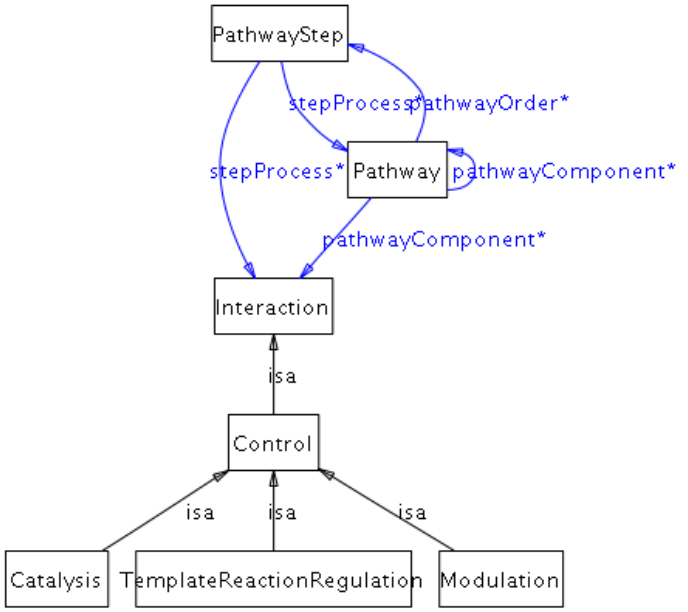
stepConversion

Definition: The central process that take place at this step of the biochemical pathway.
Domain: BiochemicalPathwayStep
Range: ConversionOWLProperty: Functional
SuperProperty: *stepProcess*
Object Property Diagram:



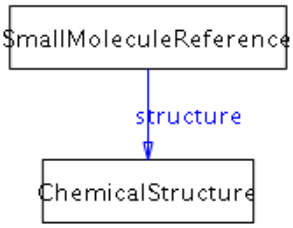
stepProcess

Definition: An interaction or a pathway that are a part of a pathway step.
Domain: PathwayStep
Range: Pathway,*Interaction***Object Property Diagram:**



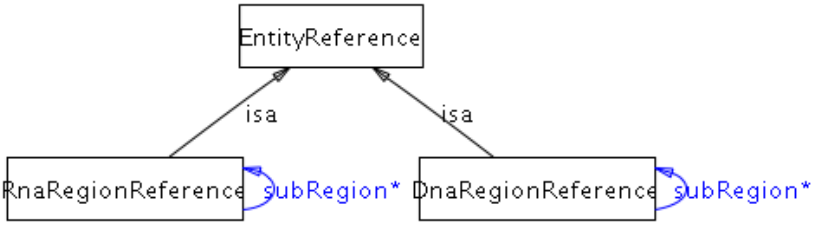
structure

Definition: Defines the chemical structure and other information about this molecule, using an instance of class chemicalStructure.
Domain: SmallMoleculeReference
Range: ChemicalStructureOWLProperty: Functional
Object Property Diagram:



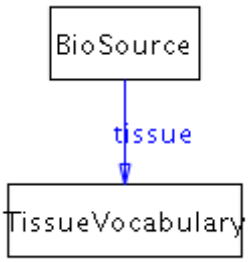
subRegion

Definition: The sub region of a region. The sub region must be wholly part of the region, not outside of it.
Domain: DNARegionReference, *RNARegionReference*
Range: EntityReferenceOWLProperty: Transitive
Object Property Diagram:



tissue

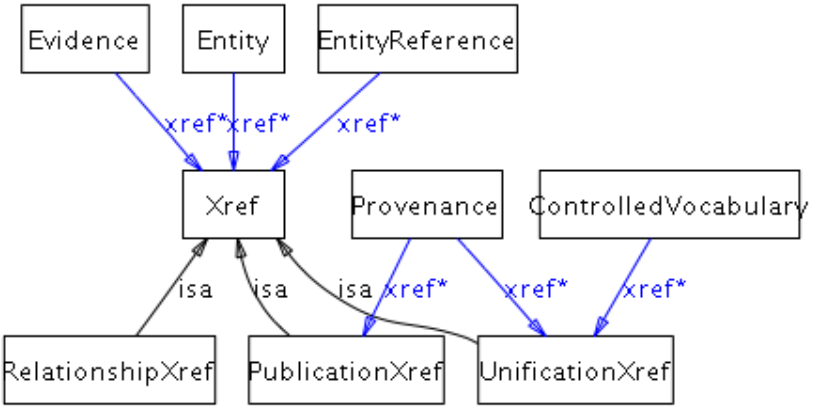
Definition: An external controlled vocabulary of tissue types.
Domain: BioSource
Range: TissueVocabularyOWLProperty: Functional
Object Property Diagram:



EntityReference

xref

Definition: Values of this property define external cross-references from this entity to entities in external databases.
Domain: ControlledVocabulary, Entity, *Provenance*, *Evidence*, *EntityReference*
Range: XrefObject Property Diagram:



4 Data Implementations

Database	Type	URL	Format	License	Statistics
BIND [33]	Protein interactions	http://tap.med.utoronto.ca/~bind/	PSI-MI Level 1	Free to all	>85,000 interactions
BioCyc databases [67,68]	Metabolic and signaling	http://biocyc.org	BioPAX Level 3	Free to all	~500 mostly computationally predicted pathway databases
BioGRID [37,69]	Protein-protein and genetic interactions	http://www.thebiogrid.org/	PSI-MI Level 1 and 2.5	Free to all	>265,000 interactions
BioModels [70]	Metabolic and signaling	http://biomodels.net/	SBML, BioPAX Level 2	Free to all	>450 pathways, >240 curated pathways, >40,000 interactions
Cancer Cell Map	**Signaling Pathways**	**http://cancer.cellmap.org**	**BioPAX Level 2**	**Free to all**	**Pathways: 10 Interactions: 2,104 Physical Entities: 899**
DIP [34,71]	Protein-protein interactions	http://dip.doe-mbi.ucla.edu/	PSI-MI Level 1	Free for Academics	>57,000 interactions
Ecocyc [14]	**Metabolic and Signaling Pathways**	**http://ecocyc.org/**	**BioPAX, Level 3**	**Free to all**	**Pathways: 246 Regulatory interactions: 5,000 Metabolic reactions: 1400 Physical Entities: 3,606**
HPRD [72]	Protein-protein interactions	http://hprd.org/	PSI-MI Level 2.5	Free for Academics	>38,000 interactions
IMID	Signaling	http://www.sbcny.org/data.htm	BioPAX Level 2	Free to all	>2000 interactions
INOH	Signaling	http://www.inoh.org/	BioPAX Level 2	Free to all	>60 pathways
IntAct [73]	Protein-protein interactions	http://www.ebi.ac.uk/intact	PSI-MI Level 1 and 2.5	Free to all	>200,000 interactions
KEGG Pathway [16]	Metabolic	http://www.genome.jp/kegg/	BioPAX Level 1	Free for Academics	>330 reference pathways
MetaCyc [15]	Metabolic and signaling	http://metacyc.org/	BioPAX Level 3	Free to all	>1399 curated pathways, >8,100 reactions
MINT [74]	Protein-protein interactions	http://mint.bio.uniroma2.it/mint	PSI-MI Level 1 and 2.5	Free to all	>80,000 interactions
MIPS MPact[75]	Protein-protein interactions	http://mips.gsf.de/genre/proj/mpact/	PSI-MI Level 1 and 2.5	Free to all	>12,000 interactions
NCI/Nature Pathway	Signaling	http://pid.nci.nih.gov/	BioPAX Level 2	Free to all	>400 curated pathways >12800 interactions

Interaction Database [9]					
NetPath	Signaling	http://netpath.org/	BioPAX Level 2	Free to all	20 large curated pathways
OPHID [76]	Protein-protein interaction	http://ophid.utoronto.ca	PSI-MI Level 1	Free for Academics	>424,000 interactions
Pathway Commons	Pathways and interactions	http://www.pathwaycommons.org	**BioPAX Level 2**	**Free to all**	>1,400 collected pathways >421,000 interactions
Reactome [77]	**Metabolic and Signaling Pathways**	**http://reactome.org/**	**BioPAX, Level 2**	**Free to all**	**>50 curated pathways**
Rhea	**Metabolic Reactions**	**http://www.ebi.ac.uk/rhea**	**BioPAX, Level 2**	**Free to all**	**>11,000 reactions**

Table 2 Databases supporting BioPAX

5 Software Implementations

BioPAX Validator

Pathway data are prone to different kinds of “error” due to for example, various conversion issues, OWL "Open World" semantics, external references to biological databases and ontologies amongst others. The BioPAX validator is a new open source Java application and API created to check the BioPAX constraints, data consistency, and implementations of the community best practices. It provides both a console (terminal) and web interface (which is optional but the recommended setup) and is configurable and extensible. Syntactic and semantic validation rules, through generic Java beans, make use of an in-memory BioPAX model. More rules can easily be developed and included in the validator using the API. The validator rules can check across several entities, entities that overlap in their subjects, or call other rules. The validator can be run in two-modes: Fail-fast validation and post-validation. In the fail-fast mode specific validation rules are triggered as the model is being edited and and if the edit is invalid, they immediately fail (hence the name fail-fast). This is especially useful for BioPAX editing software as the edit can be rolled back and exception can be propagated to the GUI to notify the user. In the post-validation scenario a model is checked completely to find all the errors using all the rules. Post validation is typically used once the model is complete and offers two advantages in this scenario over fast validation. First fail-fast stops at the first error whereas post-validation will try to find all possible problems. Second, some rule violations related to "completeness" can not be detected with the fail-fast method. For example if a reaction has no participants post-validation will complain whereas fail-fast validation will not. Post-validation, however, can be expensive and not-suitable for checking the model in an editing scenario. The validator can be set to different reporting options per rule such as exception, log error or log warning.

The validator uses Java, Spring Framework, and popular open source libraries. It integrates the following domain components: Paxtools, Ontology Manager (access to controlled vocabulary terms), and MIRIAM (to check external database identifiers). Although the first release is to check the BioPAX Level 3 data, it was designed to import data from all the BioPAX versions and check all the universal and level-specific rules that are available at runtime. Syntax (and unclassified) errors are reported mainly via the interception of the corresponding events in Paxtools (in RDF parser and model builder). Semantic rules, all the classes under the org.biopax.validator.rules package, apply either

immediately whenever new data import begins, after the model is built, or after/before any object is modified.

Example

```
<!-- Problem 1. Using illegal class (from the Level2)-->
<bp:unificationXref rdf:ID="xrefL2">
<bp:db
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">KEGG</bp:db>

</bp:unificationXref>
<!-- Problem 2. Using non-existing property (must be bp:id) -->
<bp:UnificationXref rdf:ID="xrefL3">
</bp:comment>
<bp:ID
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">C00002</bp:ID>

<!-- Problem 3. Using 'NIL' as a data property value -->
<bp:db
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">NIL</bp:db>
</bp:UnificationXref>

<!-- missing ID (RDF error) -->
<bp:Protein/>
```

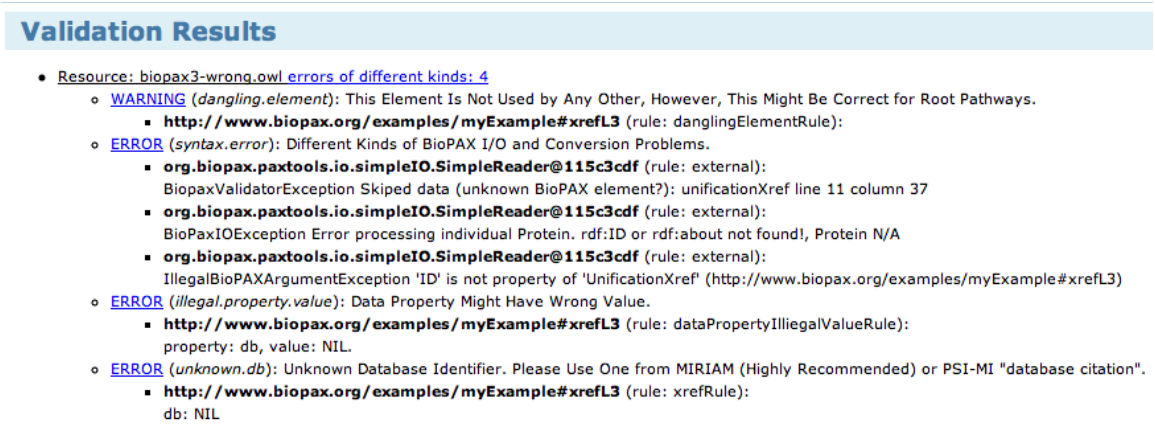
With the above example (adding the typical BioPAX header, and the </rdf:RDF> at the end of the document), Protégé will not complain at all, WonderWeb OWL Ontology Validator (WOV) reports:

1. Untyped Data Property: http://www.biopax.org/release/biopax-level3.owl#ID
2. Untyped Class: http://www.biopax.org/release/biopax-level3.owl#unificationXref

And the Validator reports the following (shortened):

1. WARNING (dangling.element): This Element Is Not Used by Any Other; http://www.biopax.org/examples/myExample#xrefL3
2. ERROR (syntax.error): Different Kinds of BioPAX I/O and Conversion Problems.
   - Skipped data (unknown BioPAX element?): unificationXref line 11 column 37
   - Error processing individual Protein. rdf:ID or rdf:about not found
   - 'ID' is not property of 'UnificationXref' (http://www.biopax.org/examples/myExample#xrefL3)
3. ERROR (illegal.property.value): Data Property Might Have Wrong Value. http://www.biopax.org/examples/myExample#xrefL3 property: db, value: NIL.
4. ERROR (unknown.db): Unknown Database Identifier. Please Use One from MIRIAM (Highly Recommended) or PSI-MI "database citation". http://www.biopax.org/examples/myExample#xrefL3 db: NIL

Screenshot 1. BioPAX Validation Result.



Validation Results

- Resource: biopax3-wrong.owl errors of different kinds: 4
  - WARNING (*dangling.element*): This Element Is Not Used by Any Other, However, This Might Be Correct for Root Pathways.
    - **http://www.biopax.org/examples/myExample#xrefL3** (rule: danglingElementRule):
  - ERROR (*syntax.error*): Different Kinds of BioPAX I/O and Conversion Problems.
    - **org.biopax.paxtools.io.simpleIO.SimpleReader@115c3cdf** (rule: external):
      BiopaxValidatorException Skiped data (unknown BioPAX element?): unificationXref line 11 column 37
    - **org.biopax.paxtools.io.simpleIO.SimpleReader@115c3cdf** (rule: external):
      BioPaxIOException Error processing individual Protein. rdf:ID or rdf:about not found!, Protein N/A
    - **org.biopax.paxtools.io.simpleIO.SimpleReader@115c3cdf** (rule: external):
      IllegalBioPAXArgumentException 'ID' is not property of 'UnificationXref' (http://www.biopax.org/examples/myExample#xrefL3)
  - ERROR (*illegal.property.value*): Data Property Might Have Wrong Value.
    - **http://www.biopax.org/examples/myExample#xrefL3** (rule: dataPropertyIlliegalValueRule):
      property: db, value: NIL.
  - ERROR (*unknown.db*): Unknown Database Identifier. Please Use One from MIRIAM (Highly Recommended) or PSI-MI "database citation".
    - **http://www.biopax.org/examples/myExample#xrefL3** (rule: xrefRule):
      db: NIL

Useful Links

- http://sourceforge.net/projects/biopax (BioPAX, Validator, Paxtools, wiki, etc.)
- http://www.pathwaycommons.org (Pathway Commons)
-

http://psidev.svn.sourceforge.net/viewvc/psidev/psi/tools/ontology-manager (Ontology Manager)

- http://www.ebi.ac.uk/miriam (MIRIAM)

Paxtools

Paxtools is a library for accessing and manipulating BioPAX. Software tools that use BioPAX, such as exporters, importers, analysis algorithms or editors can use Paxtools as their core BioPAX API. Paxtools supports BioPAX Level 2, and can also automatically convert Level 1 models to Level 2. In the development builds, the latest BioPAX Level 3 version is fully supported. Using Paxtools, a user can read a BioPAX model from a file or create a new model in memory. A model is a container for BioPAX elements, and has methods for querying and retrieving its contents. In a BioPAX model, elements are represented as plain java beans, with getter and setter methods for their BioPAX properties. A user can create and add new BioPAX elements to the model, remove existing elements or modify their properties. It is also possible for users to traverse the corresponding BioPAX graph to retrieve interesting subgraphs.
Paxtools provides a domain object model (DOM) with strong typing for domains and ranges of the properties and also has basic validation

capabilities, for example, it checks whether cardinality constraints are met. Some of BioPAX rules that were documented but could not represented formally in BioPAX OWL are also validated against. Paxtools either fails fast or warns if these rules are violated. Additionally, Paxtools checks for documented best practices and logs potentially problematic usages.
Paxtools comes with a state-of-the-art persistence and searching layer using Java Persistence API (JPA) and Lucene. This enables storing and querying particularly large models and concurrent editing of the same model by multiple users. Lucene allows efficient fulltext querying.
Paxtools has an experimental support for integrating two pathways from different sources by identifying interactions that are similar. It is often difficult to make an exact matching between entities in several cases- for example the same state of the protein can be represented as the active state in one pathway database and doubly phosphorylated in another one. An iterative two-step process fuzzy-matching entities followed by finding interactions that have similar participants however provides quite reasonable pathway alignments. This facility can be used for cross-validating pathways from different sources or as a first step for integrating overlapping pathways.
Paxtools has also several other useful functionalities such as reflection based access, traversal methods, and programmatic access to Pathway Commons web service API.

Software	Type	URL	Format	License	Language
BiNoM [78]	**Editor/Converter**	**http://bioinfo-out.curie.fr/projects/binom/**	**BioPAX Level 1 and 2**	**Free to all (open source)**	**Java**
BioPAX validator	**Validator**	**http://www.ohsucancer.com/biopaxvalidator/index.html**	**BioPAX Level 1 and 2**	**Free to all (open source)**	**Java**
BioPAX validator	**Validator**	**http://www.biopax.org/biopax-validator/**	**BioPAX Level 3**	**Free to all (open source)**	**Java**
BioUML	**Editor/Simulator**	**http://www.biouml.org/**	**BioPAX Level 2**	**Free to all (open source)**	**Java**
Biowarehouse	**Biological data warehouse software**	**http://biowarehouse.ai.sri.com/**	**BioPAX Level 1 and 2**	**Free to all (open source)**	**C and Java**
ChiBE	**Editor**	**http://www.bilkent.edu.tr/~bcbi/chibe.html**	**Level 3**	**Free to all (open source)**	**Java**
cPath [47]	**Pathway database software**	**http://cbio.mskcc.org/dev_site/cpath/**	**BioPAX Level 1 and 2**	**Free to all (open source)**	**Java**
Cytoscape [20]	**Visualization and analysis**	**http://cytoscape.org**	**BioPAX Level 1, 2, 3**	**Free to all (open source)**	**Java**

ExPlain Analysis System	**Pathway analysis**	**http://www.biobase-international.com/pages/index.php?id=286**	**BioPAX Level 1 and 2**	**Commercial**	
GeneSpring GX	**Pathway analysis**	**http://www.agilent.com/chem/genespring**	**BioPAX Level 1 and 2**	**Commercial**	**Java**
Pathway Tools [21]	**Pathway prediction, editing, visualization, network analysis, gene expression analysis**	**http://bioinformatics.ai.sri.com/ptools/**	**BioPAX Level 3**	**Free for Academics**	**Lisp**
PATIKA [3]	**Visualization**	**http://web.patika.org**	**BioPAX Level 1 and 2**	**Free to all**	**Java**
Paxtools	**BioPAX input/export library**	**http://www.biopax.org/paxtools/**	**BioPAX Level 1,2,3**	**Free to all (open source)**	**Java**
PSI-MI to BioPAX converter	**BioPAX translator**		**BioPAX Level 2,3**	**Free to all (open source)**	**Java**
QPACA [79]	**Gene expression analysis**	**https://cabig.nci.nih.gov/tools/QPACA**	**BioPAX Level 1 and 2**	**Free to all**	**Java**
SBML tp BioPAX converter	**BioPAX translator**	**http://www.ebi.ac.uk/compneur-srv/sbml/convertors/SBMLtoBioPax.html**	**BioPAX Level 2**	**Free to all (open source)**	**Java**
SHARKview [80]	**Pathway visualizer**	**http://www.bioinformatics.leeds.ac.uk/shark/**	**BioPAX Level 1 and 2**	**Free to all**	**Java**
The Gateway to Biological Pathways	**Pathway query web service**	**http://jlab.calumet.purdue.edu/theGateway/**	**BioPAX Level 1 and 2**	**Free to all**	**Java**
VisANT [22,81]	**Visualization and analysis**	**http://visant.bu.edu/**	**BioPAX Level 1 and 2**	**Free to all**	**Java**

Table 3 Software supporting BioPAX

6 Worked Examples

A set of BioPAX level 3 examples are available at the sourceforge CVS repository.
http://biopax.cvs.sourceforge.net/viewvc/biopax/biopax/examples/

A short description of each file is given in this section.

Short metablic pathway

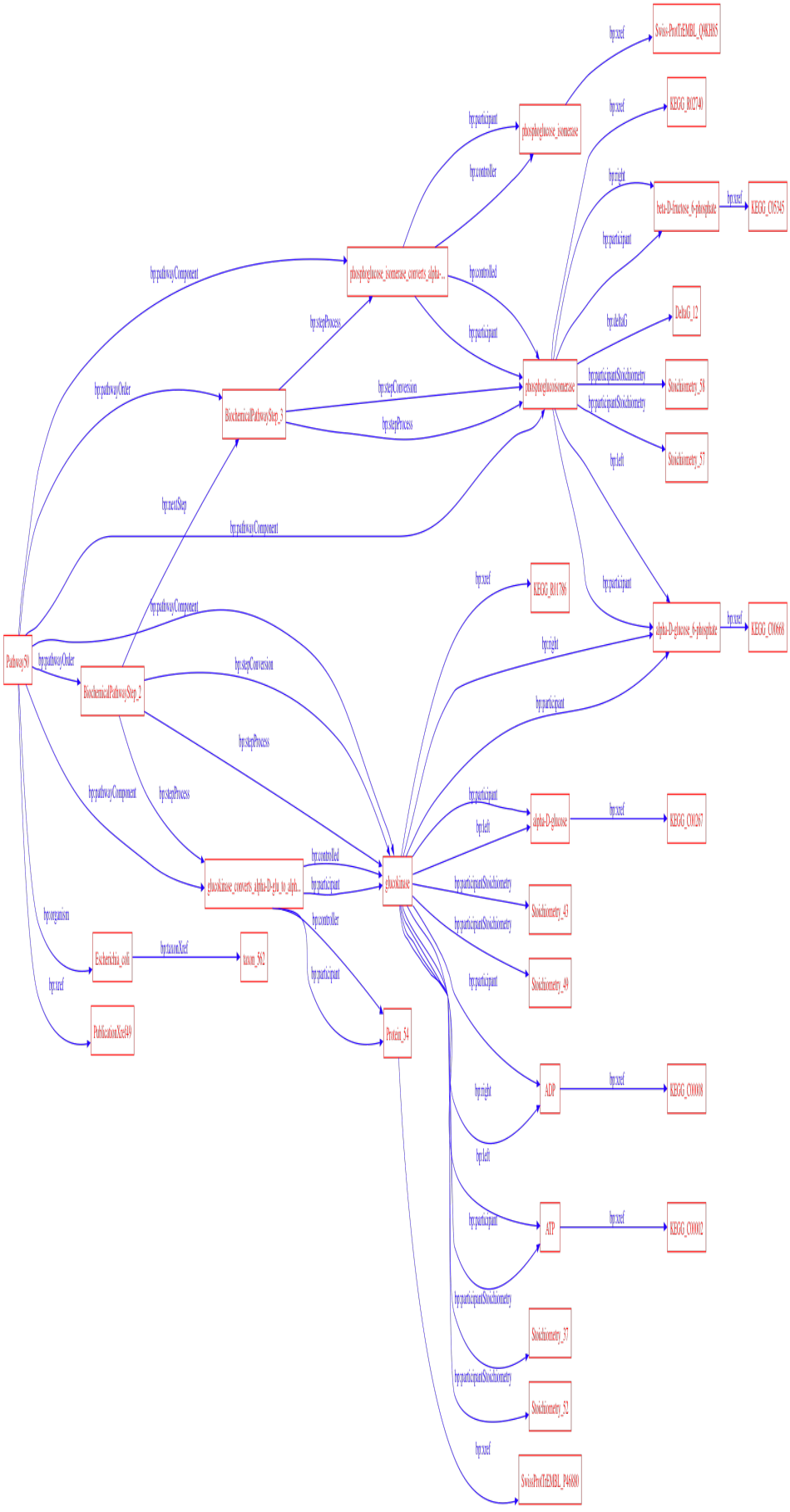
File:
http://biopax.cvs.sourceforge.net/viewvc/biopax/biopax/examples/biopax3-short-metabolic-pathway.owl?view=log
Description: This is a short example pathway showing the first two steps of glycolysis. It is a good overview of how classes and properties should be used in BioPAX3. This example illustrates the pairing of the conversion and control classes, where a BiochemicalReaction conversion is controlled by a Catalysis control.
Difference from level2: The use of EntityReferences as the main source of information about physical entities (such as protein sequence, or small molecule structure).
Classes used:
Entity: Catalysis, BiochemicalReaction, Pathway, Protein, SmallMolecule

UtilityClass: BioSource, ChemicalStructure, CellularLocationVocabulary, DeltaG, ProteinReference, SmallMoleculeReference, BiochemicalPathwayStep, Provenance, Stoichiometry, PublicationXref, UnificationXref

Figure 3 BioPAX classes used in a short metabolic pathway example.

Phosphorylation reaction

File:



http://biopax.cvs.sourceforge.net/viewvc/biopax/biopax/examples/biopax3-phosphorylation-reaction.owl?view=log
Description: This example is a single biochemical reaction (phosphorylation) from the Reactome database. This is a good illustration of the use of EntityReferences, as the ProteinReference contains information about the protein, and there are two separate Protein entities, one phosphorylated, and one not phosphorylated. Both of these Protein entities have the same ProteinReference. The details about the phosphorylation are recorded in the ModificationFeature UtilityClass, as opposed to the sequenceFeature utilityClass in BioPAX2. The addition of the EntityFeature UtilityClass allows for more specific information about post-translational modification to be recorded in BioPAX format.
Difference from level2: The MolecularInteraction class is new, as are the ControlledVocabulary categories EvidenceCodeVocabulary and RelationshipTypeVocabulary. In addition, MINT confidence values are now documented in the Score UtilityClass.

Protein-Protein Interaction

File:
http://biopax.cvs.sourceforge.net/viewvc/biopax/biopax/examples/biopax3-protein-interaction.owl?view=log
Description: This short example of protein-protein interaction data makes use of the MolecularInteraction class. This entity allows the interaction of molecules without a requirement of control or conversion. These interactions often require associated experimental evidence to interpret reliably. The ControlledVocabulary category EvidenceCodeVocabulary expands upon the information supporting the protein protein interaction, and RelationshipTypeVocabulary defines relationship Xref types in reference to the PSI Molecular Interaction ontology.
Classes used:
Entity: *MolecularInteraction, Protein*

UtilityClass: *BioSource, EvidenceCodeVocabulary, RelationshipTypeVocabulary, BindingFeature, ProteinReference, Evidence, Provenance, Score, SequenceInterval, SequenceSite, PublicationXref, RelationshipXref, UnificationXref*

Genetic Interaction

File:
http://biopax.cvs.sourceforge.net/viewvc/biopax/biopax/examples/biopax3-genetic-interaction.owl?view=log
Description: This short example of genetic interaction data makes use of the GeneticInteraction class. This entity allows the

documentation of interactions between genes that occur when two mutations have a combined effect not caused by either perturbation alone. This is a logical interaction.

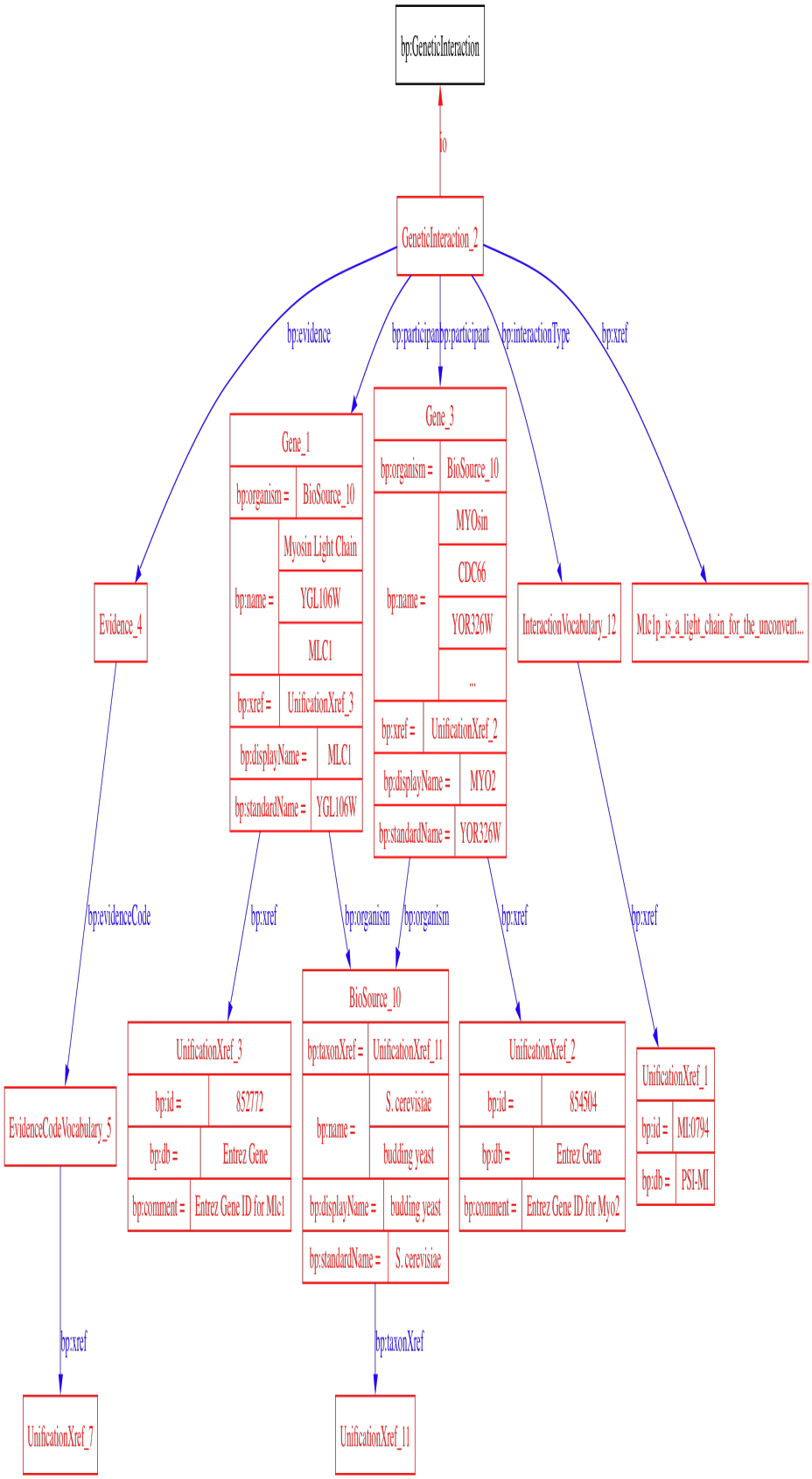
Classes used:

Entity: *Gene, GeneticInteraction*

UtilityClass: *BioSource, EvidenceCodeVocabulary, InteractionVocabulary, Evidence, PublicationXref, UnificationXref*

Figure 4 classes used in a GeneticInteraction

Figure 5


bp:GeneticInteraction
is
GeneticInteraction_2
bp:evidence
bp:participan
bp:participant
bp:interactionType
bp:xref
Gene_1
bp:organism =
BioSource_10
bp:name =
Myosin Light Chain
YGL106W
MLC1
bp:xref =
UnificationXref_3
bp:displayName =
MLC1
bp:standardName =
YGL106W
Gene_3
bp:organism =
BioSource_10
bp:name =
MYOsin
CDC66
YOR326W
...
bp:xref =
UnificationXref_2
bp:displayName =
MYO2
bp:standardName =
YOR326W
Evidence_4
InteractionVocabulary_12
Mlc1p_is_a_light_chain_for_the_unconvent...
bp:evidenceCode
bp:xref
bp:organism
bp:organism
bp:xref
bp:xref
EvidenceCodeVocabulary_5
UnificationXref_3
bp:id =
852772
bp:db =
Entrez Gene
bp:comment =
Entrez Gene ID for Mlc1
BioSource_10
bp:taxonXref =
UnificationXref_11
bp:name =
S. cerevisiae
budding yeast
bp:displayName =
budding yeast
bp:standardName =
S. cerevisiae
UnificationXref_2
bp:id =
854504
bp:db =
Entrez Gene
bp:comment =
Entrez Gene ID for Myo2
UnificationXref_1
bp:id =
MI:0794
bp:db =
PSI-MI
bp:xref
bp:taxonXref
UnificationXref_7
UnificationXref_11

Biochemical Reaction

File:
http://biopax.cvs.sourceforge.net/viewvc/biopax/biopax/examples/biopax3-insulin-maturation.owl?view=log
Description: This example of the insulin maturation pathway uses the EntityFeature of CovalentBindingFeature to document cysteine bond formation. The ControlledVocabulary category SequenceModificationVocabulary specifies the half-cysteine bonds, and the CovalentBindingFeature indicates which residues are involved in this bond formation.
Classes used:
Entity: *BiochemicalReaction, Protein*

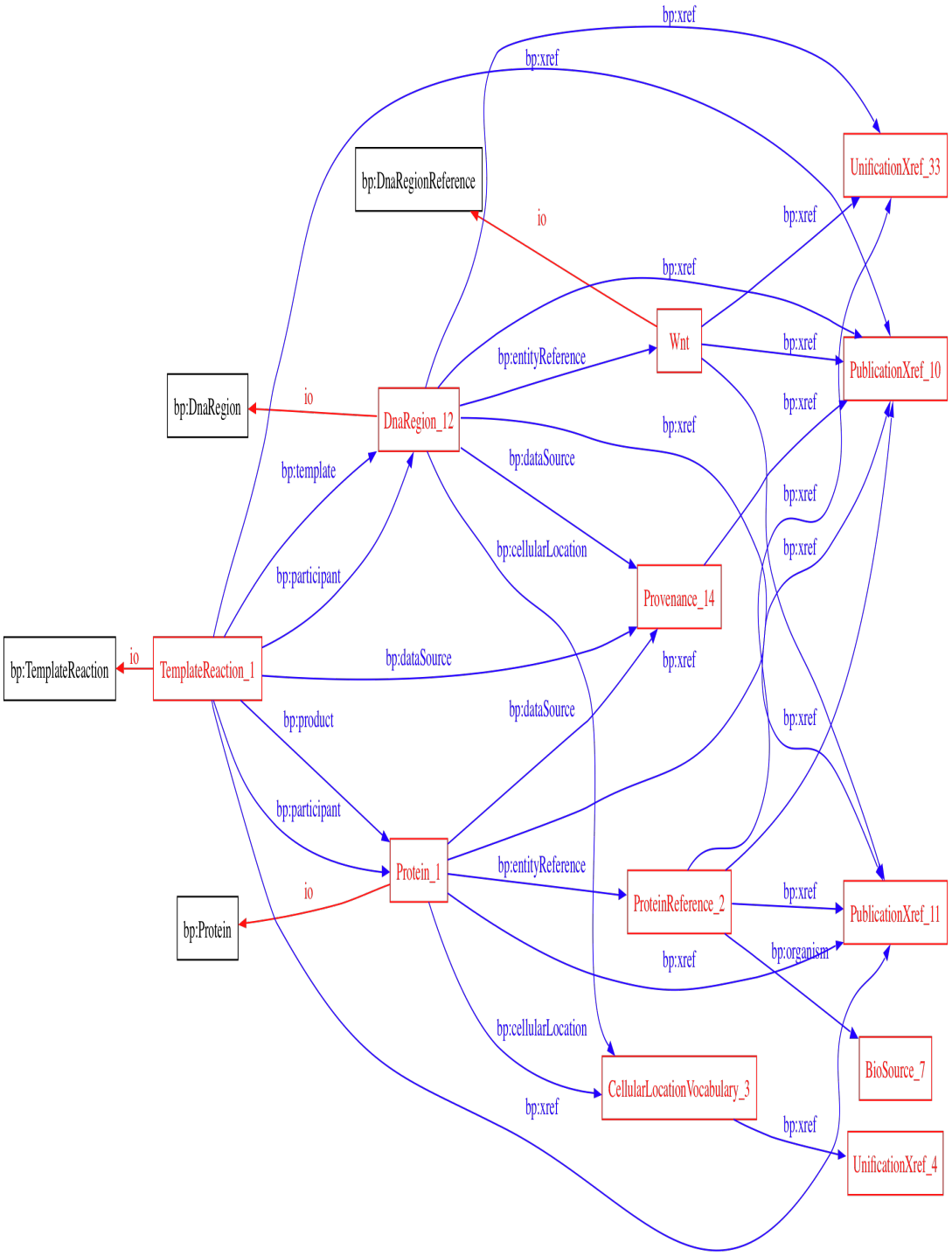
UtilityClass: *BioSource, InteractionVocabulary, SequenceModificationVocabulary, SequenceRegionVocabulary, FragmentFeature, CovalentBindingFeature, ProteinReference, SequenceInterval, SequenceSite, PublicationXref, UnificationXref*

Template Reaction

File:
http://biopax.cvs.sourceforge.net/viewvc/biopax/biopax/examples/biopax3-template-reaction.owl?view=log
Description: This example uses the TemplateReaction and TemplateReactionRegulation Interaction classes in a subset of the Davidson BioTapestry transcriptional map. Expression in the endoderm of sea urchin at 15 hours is documented here. TemplateReaction is used in this case to document protein production from a DNA template (DnaRegion). The production of an mRNA intermediate is assumed but not included in this example. TemplateReactionRegulation is only used in cases where the controlling element (transcription factor) is known. The ControlledVocabulary class InteractionVocabulary is used to document the physical association of the transcription factor with the DnaRegion. Promoter locations can also be included, if known, by creating a feature in the DnaRegion.
Classes used:
Entity: *TemplateReactionRegulation, TemplateReaction, Complex, DnaRegion, Protein*

UtilityClass: *BioSource, CellularLocationVocabulary, InteractionVocabulary, DnaRegionReference, ProteinReference, Provenance, PublicationXref, UnificationXref*

Figure 6 Classes used in a TemplateReaction


bp:xref
bp:xref
UnificationXref_33
bp:DnaRegionReference
io
bp:xref
bp:xref
Wnt
bp:xref
PublicationXref_10
bp:entityReference
bp:DnaRegion
io
DnaRegion_12
bp:xref
bp:xref
bp:dataSource
bp:template
bp:xref
bp:cellularLocation
bp:participant
bp:xref
Provenance_14
bp:TemplateReaction
io
TemplateReaction_1
bp:dataSource
bp:xref
bp:product
bp:dataSource
bp:xref
bp:participant
Protein_1
bp:entityReference
ProteinReference_2
bp:xref
PublicationXref_11
io
bp:Protein
bp:xref
bp:organism
bp:cellularLocation
BioSource_7
CellularLocationVocabulary_3
bp:xref
bp:xref
UnificationXref_4

7 Best Practices

While the BioPAX ontology imposes many logical constraints so that data encoded make sense for the use cases envisioned, some parts of the ontology have the potential for encoding data in multiple ways or there may be multiple options for treating data. This section recommends best practices in the use of the ontology that helps normalize data for exchange between groups. It supplements recommendations made in the class and property definitions provided in Chapters, 2 BioPAX Ontology Class Structure and Level 3 ObjectProperties. It is expected that major data providers follow these recommendations to ensure compatibility of their data with other BioPAX data.

Users of BioPAX who are not exchanging data between groups, e.g. using BioPAX as an internal data model for their software, might find alternate representations to the ones recommended here more useful for their purposes.

Referencing External Objects

BioPAX is focused on representing interactions and pathways, but also links to many different types of information. It is important to maintain these links, since they are useful for data integration and for further defining biological objects referenced by BioPAX documents. Biological objects in external databases, such as proteins and small molecules, should be referenced via instances of the Xref class hierarchy, and external controlled vocabulary terms, such as those defined by the Gene Ontology Consortium or the PSI-MI initiative, should be referenced via instances of the ControlledVocabulary class hierarchy. BioPAX does not currently support a general mechanism to use RDF IDs to seamlessly point to external biological objects and controlled vocabulary terms on the semantic web.

Using Xrefs

External references (Xrefs) relate elements within a BioPAX document to external data. Xrefs are more than just identifiers, as they contain the name of the data source the identifiers are part of and potentially version information as well. They exist to uniquely point to a record in an external data source (e.g. a bioinformatics database). For example, a pointer from a protein instance in BioPAX to a record in a database describing the protein would be established with an Xref. Note: Xrefs are NOT related to RDF IDs (see below).

Within any Xref, database names (in the DB property) should be from a controlled vocabulary to avoid data integration problems that arise when different people use different spellings of database names. The

PSI-MI makes available a database name controlled vocabulary (see ‘Using external controlled vocabulary terms’ section for more information). If it is not possible to use this controlled vocabulary for a specific database, be careful to use the database name exactly as spelled on the database website (e.g. Use “Swiss-Prot” instead of swissprot, SWP or other frequent spellings). Also, suggest that the database name you want to use be added to the PSI-MI vocabulary. Similarly, the ID property should use the primary unique identifier of the target object, e.g. “P54352” instead of “HUMAN_P53”. Software should be able to use Xref information to construct a web hyperlink to the database record being pointed to.

Xrefs to database accession numbers that contain version information should keep the version information separate from the identifier (ID), e.g. the accession number “CAA61361.2” should be stored as “id=CAA61361”, and “idVersion=2”. This is to enable computer software to easily identify the accession or the version without having to be aware of all possible ways of encoding the version in the accession number. If you are unsure how to encode the ID, think about how the encoded information could be used to build a web hyperlink to the referenced database record.

Entity References Vs Unification xrefs

Careful thought should go into the use of EntityReferences vs Unification Xref objectProperties of physicalEntities. Although it is possible to create both PhysicalEntities and EntityReferences pointing to the same Xref, this should be avoided. Instead, given a physicalEntity is "a pool of molecules" that are involved in some interaction in the cell, this physicalEntity should point to a generic form of the molecule as a memberEntityReference property. The EntityReference should in turn have a unificationXref to a database entry for such a molecule.

Importance of unification xrefs

Abundant use of unification xrefs, where possible, is highly recommended, especially in physicalEntity instances. These xrefs allow a user to understand that two independent instances from different BioPAX documents are actually the same entity (as long as they share one or more unification xrefs).

When exporting data from a database with primary keys, those keys should generally be encoded as unification xrefs. For example, if a database contains biochemical reactions with IDs for both the reactions and the small molecules that participate in those reactions, unification xrefs containing these IDs should appear in the corresponding BioPAX instances generated by the database. In general, the original data

record from which an instance was generated should be pointed to via a unification xref. The exception to this rule occurs when the native class of the data is not completely synonymous with the BioPAX class to which it is mapped. In these cases, the resulting BioPAX instances should point back to the original data records via relationship xrefs.

Caution: Complications with unification xrefs can arise when the database that is being pointed to contains redundant information or contains more than one type of record. If a database contains redundant information, such as GenBank or Chemical Abstracts Service (CAS), it is possible to reference the same physical entity in the same database, but use IDs of different redundant records. In this case, unification xrefs can not be guaranteed to be useful in determining if two physical entities are the same across multiple BioPAX documents. More information about database record relationships will be required, such as which records are synonymous. Also, if a database contains different types of records, such as mRNA and protein records in GenBank or chemical structures with and without R groups in CAS, then it may be impossible to determine the type of record referenced, which may lead to errors from unification xrefs that point to molecules of a different type than the referencing physical entity. Care in creating unification xrefs should be taken when linking to these types of databases so that the link is unambiguous.

External database identifiers

Use of standard database names and identifiers is recommended. For instance, UniProt or RefSeq IDs should be used for proteins. A list of database names is available in the PSI-MI controlled vocabulary at http://www.ebi.ac.uk/ontology-lookup/browse.do?ontName=MI&termId=MI%3A0444&termName=database%20citation. Also, the MIRIAM resource maintains a list of database names, and additional information about each database, such as a regular expression that can be used to validate the ID (see http://www.ebi.ac.uk/compneur-srv/miriam-main/).

Using external controlled vocabulary terms

A number of properties in the BioPAX ontology reference the ControlledVocabulary class. Some of these are referred to as “mission-critical” because the information they provide is very important to most users of pathway data and to enable software to make simplifying assumptions about which vocabularies to expect. These are: CellularLocation, EvidenceCode, ExperimentalFormDescription, SequenceModificationVocabulary and SequenceRegionVocabulary. It is required that the following external controlled vocabularies (CVs) be used for these mission-critical properties:

CellularLocation: Gene Ontology Cellular Component
EvidenceCode: BioCyc Evidence Ontology, PSI-MI interaction detection CV (Note: the PSI-MI CV will likely be extended to contain all BioCyc evidence codes. When this occurs, the PSI-MI CV will be preferred.)
ExperimentalFormDescription: PSI-MI participant identification method (e.g. mass spectrometry), experimental role (e.g. bait, prey), experimental preparation (e.g. expression level) type CVs
SequenceModificationVocabulary: PSI-MI Feature Type CV, biological feature child terms
SequenceRegionVocabulary: Sequence Ontology
db property (in xref): PSI-MI Database Name CV, where possible

Note: evidenceCode allows multiple CV terms to be included. Because it is mission critical, at least one term should be from the above recommended CV.

Several other properties, cellType, interactionType, relationshipType and tissue, also make use of the ControlledVocabulary class. These non-mission-critical properties serve simply to provide additional annotation. The following CVs are suggested for these properties:

cellType: http://obo.sourceforge.net/cgi-bin/detail.cgi?cell
interactionType: PSI-MI Interaction Type CV
relationshipType: PSI-MI Cross Reference Type CV
tissue: BRENDA Tissue Ontology (BTO)

Note: The PSI-MI Level 2.5 CVs may be found at: http://psidev.cvs.sourceforge.net/psidev/psi/mi/rel25/data/ and can be browsed at http://www.ebi.ac.uk/ontology-lookup/browse.do?ontName=MI

Developers should consult the latest versions for CVs to ensure current use.

Reusing utility class instances

Utility classes store structured bits of information in the context of the main ontology classes (‘entity’ and its subclasses). As such, they are not guaranteed to make sense out of the context of the classes they are used in. Utility classes should be reused carefully to avoid making improper statements out of context. For example, consider a BiochemicalPathwayStep instance that was used by multiple pathways. If new information became available for one of those pathways, addition of this additional information to the BiochemicalPathwayStep instance could invalidate it for all of the other pathways that refer to it. Due to these potential problems, it should not be assumed that utility class instances will be re-used in a BioPAX file. Software

implementations must be aware of this if instance equality is important, so that equality statements are made based on all content of utility class instances.
Control ‘controller’ and ‘controlled’ property conventions
Instances of Control can have multiple controller’s and controlled’s. Moreover, one Control instance can control another Control instance. The semantics of the use of these properties are as follows:

Multiple separate controls controlling a conversion means that they control in parallel (e.g. different enzymes catalyzing the same reaction). Generally, their effect on the rate of the reaction is cumulative.

A control with multiple controllers indicates a dependency between these controllers, typically meaning that both are required for the reaction to occur (e.g. a catalysis with an enzyme and a cofactor as controllers). Any further chaining of controls also implies dependency, for example allosteric inhibition of the aforementioned enzyme by a small molecule.

Here is a pseudo-BioPAX representation of the examples above:
rxn1 is a BiochemicalReaction

cat1 is a Catalysis
cat2 is a Catalysis

mod1 is a Modulation

enzyme1 is a Protein
enzyme2 is a Protein

cofactor1 is a SmallMolecule
drug1 is a SmallMolecule

cat1 has controlled rxn1
cat2 has controlled rxn1 (Both cat1 and cat2 can catalyze rxn1, independently)

cat1 has controller enzyme1

cat2 has controller enzyme2
cat2 has cofactor cofactor1 (both enzyme2 and cofactor1 is required for cat2 to occur)

mod1 has controlled cat2
mod1 has control-type INHIBITION_ALLOSTERIC

mod1 has controller drug1 (drug1 should NOT be present for cat2 to occur)

This structure is similar to disjunctive normal form (DNF) in Boolean logic. We could write this as: (enzyme1) OR (enzyme2 AND cofactor1 AND NOT drug1)

Conversion direction

Multiple places exist in BioPAX for providing information on the direction in which a conversion interaction proceeds. The direction property of the catalysis instance, if specified, should override all other sources of direction information. If the conversion is not catalyzed, or the direction property is empty, the spontaneous property of the conversion should be used as the source of direction information. If a conversion is spontaneous, then it will occur in the specified direction without any catalyst (although, in the cell, the reverse may happen by unknown processes). If no values for direction or spontaneous are specified, it may be possible to infer direction given the thermodynamic constants in the biochemical reaction, if specified and if assumptions about the conditions in the cell are made. It may be possible to infer direction using other computational techniques, such as flux-balance analysis. The topology information from any PathwayStep instances in the pathway class should not be used for direction information, however, the stepDirection property of BiochemicalPathwayStep can be used to infer direction of a pathway step, in the context of a pathway (this is needed when the reaction in the step is reversible, but proceeds in one direction in a pathway context e.g. due to the law of mass action). Do not assume that the default direction of conversions is in either the left-to-right or right-to-left directions.

Conventions for ‘left’ and ‘right’ properties of conversion

As stated above, substrates and products of a conversion may be placed in either the left or the right properties as these are not used to determine the direction of a conversion. However, in order to ease readability, it is preferable that users adhere to the same conventions for the contents of these properties. We therefore recommend the following, in order of precedence:

1. If the conversion has an Enzyme Commission (EC) number or a Transport Commission (TC) number, store the participants in the left and right properties such that they mirror the EC/TC reaction.
2. For complex assemblies, store the subunits in the left property and the complex in the right property.
3. For transport instances, store the outermost participants (relative to the interior of the cell or organelle) in the left property and the innermost participants in the right property.

4. If none of the above are applicable, store the participants from left-to-right in the order that the conversion occurs or is suspected to occur in the pathway.

Technical note: OWL and RDF Conventions

A typical set of BioPAX data consists of many instances of various BioPAX classes. Each of these instances must be given an RDF ID that is unique within the document to be a valid OWL/RDF document. These IDs are used to reference instances from other parts of the OWL document. When combined with a globally unique document namespace, these IDs form a URI that can provide a globally unique identifier for each BioPAX instance.

RDF IDs

In an OWL document, such as BioPAX, each instance of a class must have an RDF ID. This comes from the Resource Descriptor Framework standard (http://www.w3.org/RDF/). These IDs must be unique and are used to reference instances within a document. An RDF ID exists within a namespace, which can be explicitly appended before the RDF ID. If not explicit, the RDF ID exists in the default namespace of the document. Like anchors in HTML, a pointer to an RDF ID defined elsewhere in the document is denoted with a hash mark (“#”) in front of the RDF ID.

For Example:

```
<protein rdf:ID="protein76">
<XREF rdf:resource="#xref1146"/>
</protein>
```

It is recommended that RDF IDs do not encode any semantics and be composed of the class name followed by a unique positive integer (e.g. “protein76”) or some other naming convention that guarantees unique names within the file. Some applications that use OWL, such as Protégé, and some examples of OWL from the main OWL website, use human readable names for the RDF IDs. As long as these names are unique, a BioPAX document will be valid, but the use of human readable names as RDF IDs might encourage people to rely on information stored in them and is thus not recommended. RDF IDs may not persist after certain data processing operations, such as integrating data from two separate BioPAX files. For compatibility with the semantic web, standard URIs should be used, where available. For instance, the UniProt database maintains URIs for proteins and these could be used as RDF IDs in BioPAX (this is in addition to normal use of Xrefs).

Note that in the Protégé tool, the RDF ID of an instance is referred to as its Name. This should not be confused with the BioPAX name

property, which provides the human readable name for biological entities. Protégé can be configured to display the value of the name property (or another field value) instead of the RDF ID. Use the Display Slot pull-down menu in the Individuals tab to select the value to display.

rdf:ID, rdf:about and rdf:resource can cause some confusion. We recommend the usage exemplified below:
1. RDF:ID Vs RDF:about
Use rdf:ID URIs to uniquely identify a resource in your document and use RDF:about URIs to identify a resource that is not specific to the document. e.g.

```
<bp:xref>
<bp:RelationshipXref rdf:ID="RelationshipXref_1">
<bp:comment>hERG</bp:comment>
<bp:id rdf:about="http://www.genenames.org/data/hgnc_data.php?
match=KCNH2">KCNH2</bp:id>
<bp:db>HUGO Gene Nomenclature Committee </bp:db>
</bp:RelationshipXref>
</bp:xref>
```

2. RDF:Resources
When referencing something within the document, put the URI in an rdf:resource attribute in the BioPAX element. e.g.

```
<bp:Gene rdf:ID="Gene_23">
<bp:xref rdf:resource="#RelationshipXref_1"/>
</bp:Gene>
```

BioPAX Serialization RDF/XML recommendation

A note on supported syntax, BioPAX is edited and maintained using the Protege Ontology Editor which can serialize BioPAX in a number of forms, Turtle, N3, NTriple and RDF/XML. Any serialization of OWL that is supported by the Jena API can be used, however, for best performance we recommend RDF/XML for the serialization of BioPAX files.

Document namespace

OWL XML documents require a default namespace. The creator of the BioPAX document should create a namespace and encode it in the BioPAX document. The namespace and the RDF ID may be used together to reference instances in a document from an external document (explicit use of namespace). This reference mechanism is part of the basis of the planned Semantic Web (http://www.w3.org/2001/sw/). If a BioPAX document is going to be on the Semantic Web, it should have a unique namespace. Since there is no namespace naming authority, it is not possible to guarantee unique namespaces across the internet, but following these recommendations will reduce the chances of naming collisions.

Technically, any string without spaces is allowed (see namespace rules) as a namespace. Operationally, a URL (or more generally a URI) should be used. This does not have to be a ‘real’ URL that resolves to a web page, but it should be related to the organization of the creator and a registered domain name owned by the organization is useful to include e.g. “http://biocyc.org/ontology/biopax/#”.

Use of the xmlns and xml:base attributes to specify the namespace for any BioPAX documents created is recommended. The BioPAX ontology definition should be imported and the BioPAX namespace should be defined using the ‘bp’ string (if it does not conflict with other existing namespaces called ‘bp’) e.g. xmlns:bp=http://www.biopax.org/release/biopax-level3.owl, so that elements in the file appear like this: <bp:pathway></bp:pathway>.

A typical header of an OWL XML document that uses the BioPAX ontology will look like this:

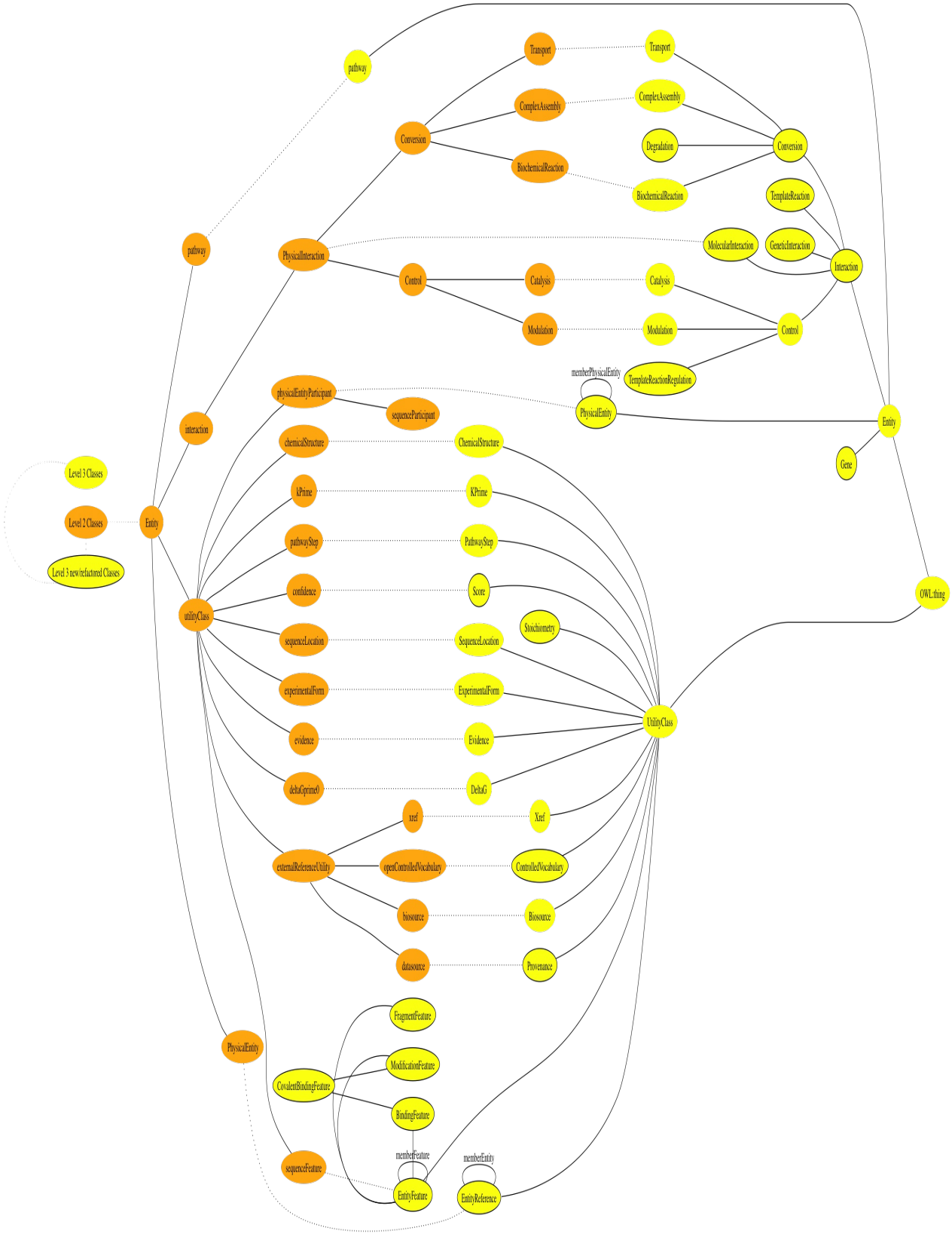
```
<?xml version="1.0" encoding="UTF-8" ?>
<rdf:RDF
 xmlns=" http://www.myorganization.org/ontology#"
 xmlns:owl="http://www.w3.org/2002/07/owl#"
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
 xmlns:xs=“http://www.w3.org/2001/XMLSchema#“
 xmlns:bp="http://www.biopax.org/release/biopax-level3.owl#"
 xml:base="http://www.myorganization.org/ontology"
>
<owl:Ontology rdf:about="">
 <owl:imports rdf:resource="http://www.biopax.org/release/biopax-
level3.owl"/>
</owl:Ontology>
```

Where “http://www.myorganization.org/ontology#” defines the namespace for this document.

OWL XML documents that mix BioPAX definitions with those from other ontologies or extend BioPAX will have different ways of using namespaces, but that is outside the scope of this document and will likely not be valid BioPAX Level 3 documents. It is good practice to specify the character encoding in the XML header, in this case UTF-8. If you have international characters in your BioPAX document, be sure to specify the correct character encoding. See http://www.w3.org/International/O-charset.html for more information.

Level2 to Level3 mappings

A converter for BioPAX Levels 1 and 2 to BioPAX level3 is available in



paxtools . This is still being tested and we welcome feedback on the conversion. As a general rule we would recommend native mapping of data from source to Level 3 as mapping from Level 2 may be lossy. If you are familiar with level 2, the figure below gives an outline of how Level 2 classes map to Level 3. Also a mapping to other data models is shown in Table 4.

BioPAX L2	BioPAX L3	Reactome
Physical Entity	EntityReference	Reference Entity
Participant(Property)	Participant (Property)	I/O
PEP	PhysicalEntity	Physical Entity
Interaction	Interaction	Event
SequenceFeature	EntityFeature	Instance Edit
Pathway	Pathway	Pathway
BioPAX L2	BioPAX L3	PID/NCI
Physical Entity	EntityReference	BioMolecule
Participant(Property)	Participant (Property)	I/O
PEP	PhysicalEntity	State
Interaction	Interaction	Biological Event
SequenceFeature	EntityFeature	PTM
Pathway	Pathway	Biological Process
BioPAX L2	BioPAX L3	INOH
Physical Entity	EntityReference	Reference Entity
Participant(Property)	Participant (Property)	I/O/ Controler
PEP	PhysicalEntity	State
Interaction	Interaction	Process or Event
SequenceFeature	EntityFeature	Biological Attribute
Pathway	Pathway	Pathway
BioPAX L2	BioPAX L3	PATIKA
Physical Entity	EntityReference	Bioentity
Participant(Property)	Participant (Property)	Substrate/Product
PEP	PhysicalEntity	State
Interaction	Interaction	Transition
SequenceFeature	EntityFeature	Bioentity Variable
Pathway	Pathway	Abstraction
BioPAX L2	BioPAX L3	aMAZE
Physical Entity	EntityReference	Physical Entity
Participant(Property)	Participant (Property)	I/O
PEP	PhysicalEntity	Bioentity, BioParticipant
Interaction	Interaction	BioEvent
SequenceFeature	EntityFeature	State
Pathway	Pathway	Pathway
BioPAX L2	BioPAX L3	SBML 2
Physical Entity	EntityReference	-
Participant(Property)	Participant (Property)	Simple Species Reference
PEP	PhysicalEntity	Species
Interaction	Interaction	Reaction
SequenceFeature	EntityFeature	-
Pathway	Pathway	
BioPAX L2	BioPAX L3	SBGN – PD
Physical Entity	EntityReference	-
Participant(Property)	Participant (Property)	Substrate/Product

PEP	PhysicalEntity	EPN
Interaction	Interaction	Process Node
SequenceFeature	EntityFeature	EntityVariable
Pathway	Pathway	Submap

Table 4 This table gives a mapping for BioPAX level 2 and BIoPAX level 3 classes.

8 Use Case Outlines

These use-cases were considered during the design of BioPAX. Other use-cases may be suggested via the http://groups.google.com/group/biopax-discuss mailing list.

Data Sharing Between Databases

One of the primary intended functions of the BioPAX format is to facilitate data exchange between existing biological pathway databases. In order for this to happen, databases must develop the ability to write-to and read-from the BioPAX format. Typically, this will require the creation of in-house software. While a number of freely available software packages may help make this task easier (e.g. Jena, an open source Java API for RDF; see http://jena.sourceforge.net/index.html or the Protégé OWL API; see http://protege.stanford.edu/plugins/owl/), development of data translation software may nonetheless require a fair amount of programming time for each individual database. This can be significantly reduced using the PaxTools Java library for reading, writing and validating BioPAX files.

The typical data transaction, i.e. passing a set of data from one database to another, consists of a number of steps:

1) Convert a set of data into the BioPAX format. This step involves mapping the native data model to the BioPAX data model (i.e. the BioPAX ontology) and then creating a BioPAX OWL file that contains instances of the mapped classes. This step will almost always require developing software to perform the mapping.
2) Transfer the BioPAX file. There are many mechanisms by which this could be accomplished, e.g. the data provider could make the file available for download from an FTP or HTTP server.
3) Convert the BioPAX file into the native format of the receiving database (the reverse of step 1). Again, this will likely require new software to perform the data conversion.
4) Merge data sets and remove redundancies. Often, many instances in a BioPAX file may already exist in the target database (Note: these are only detectable if the redundant instances share one or more unification Xrefs or if entire instances are compared). These instances should be merged

with the existing data (if they contain additional information not present in the database) or removed from the data set being imported (if not) to prevent redundant entries from being created. Also, any pointers to such instances must be redirected to the existing database objects.

As more datasets become available in the BioPAX format, software utilities will be developed (by members of the BioPAX group and others) to ease data sharing. For example, a utility to integrate the data from two different BioPAX files would be useful. With such a utility, users could integrate new BioPAX data with their own by first outputting their data into BioPAX format, then running the utility to combine it with the new data, then translating the combined data set back into their own format. Thus, the need for system-specific data integration software (step 4 above) would be reduced.

BioPAX as a Knowledge-Base (KB) Model

The BioPAX ontology is readily usable as the data model for a pathway knowledge-base (KB) using a tool like Protégé (http://protege.stanford.edu). Building a new KB with the BioPAX ontology would save time and resources since it would eliminate the need to create a data schema from scratch and it would reduce the translation requirement for exporting and importing data to/from the BioPAX format (some custom semantic mapping and ID mapping might still be required to import data from another database).

Of course, some users may wish to extend the BioPAX ontology to suit their own needs. For example, more specific classes may be added by the user. Also, many KBs use “inverse properties” – properties that are the reciprocal of other relationship properties – in order to speed up queries and facilitate browsing. Since such properties provide redundant information, they were left out of the BioPAX ontology. See the HOW-TO section for more information on creating a BioPAX KB. Note that instances adhering to an altered BioPAX ontology are not compatible with the official BioPAX standard unless converted back to standard BioPAX.

Pathway Data Warehouse

An initial motivation for creating the BioPAX standard was that it was seen as a logical first step toward creating a central public repository for biological pathway data, a resource strongly desired by many members of the pathway community. If many databases provide access to their data in the BioPAX format, it should be relatively simple to aggregate this data in a central repository, like BioWarehouse (http://biowarehouse.ai.sri.com/) or Pathway Commons

(http://www.pathwaycommons.org).

Pathway Analysis Software

Another intended function of BioPAX is to speed development time of software that makes use of pathway data to answer biological questions. Currently, in order for pathway software to access pathway data from multiple sources it must either be programmed to interpret each different format, or the data from each source must be translated into a format that the software supports. This can require significant development time and as a consequence most pathway analyses are run on only a few datasets, limiting utility.

The presence of a standard format and object model for pathway data should alleviate this problem. With the lower barrier to data access, pathway software will be easier to develop and apply. Also, additional software that might not be practical without an agreed upon standard, e.g. a sophisticated pathway visualization tool, may be more likely to be developed if BioPAX becomes widely adopted.

Pathway Analysis Software Example: Molecular profiling analysis

Genomics and proteomics technologies, such as gene expression microarrays and mass spectrometers, are being used to generate large datasets of molecules present at a specific place and time in an organism (molecular profiling), among other types of data. Molecular profiling experiments are often compared across two or more conditions (e.g. normal tissue and cancerous tissue). The result of this comparison is often a large list of genes that are differentially present in the tissue of interest. It is interesting and useful to analyze these lists of genes in the context of pathways to get a better sense of which cellular processes are affected. For instance, one could look for pathways that are statistically over-represented in the list of differentially expressed genes. The result is a list of pathways that are active or inactive in the condition of interest compared to a control. The list of pathways is often much shorter than the list of input genes, thus is easier to comprehend. BioPAX documents describing pathways could be supported by tools that perform pathway-based analysis.

Visualizing Pathway Diagrams

Pathway diagrams are useful for examining pathway data. A number of formats are available for these images, but only few available viewing tools link components in the image to underlying data. A mapping of BioPAX to a symbol library for pathway diagrams such as the Systems Biology Graphical Notation (SBGN, http://sbgn.org), described below could be the basis for a general BioPAX pathway diagram tool. Tools that support visualizing BioPAX pathways include PATIKAWeb (http://web.patika.org) and Cytoscape (http://cytoscape.org).

Visualizing BioPAX models with SBGN - PD

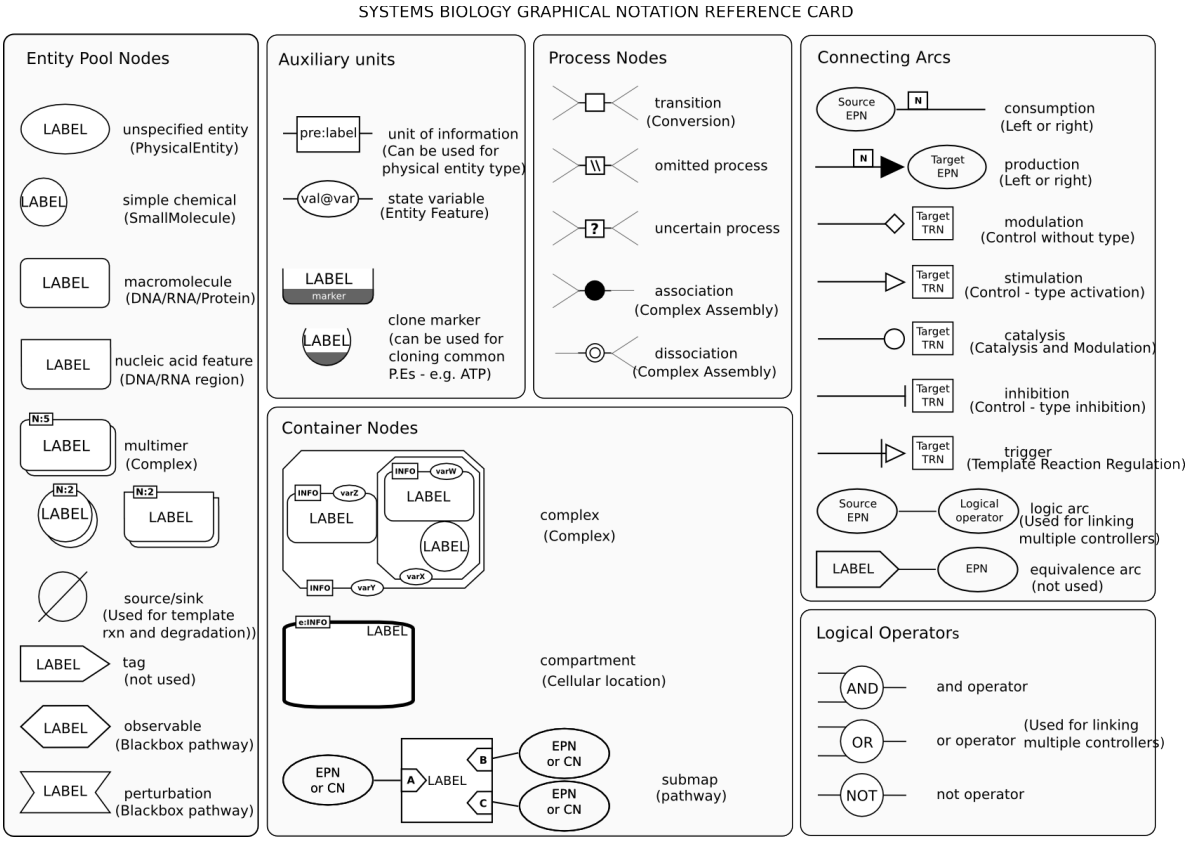
SBGN (Systems Biology Graph Notation) offers a standard graphical notation for visualizing cellular processes and interactions. SBGN is made up of three orthogonal languages, representing different views of biological systems, Process Description(PD), Entity Relationship(ER) and Activity Flow(AF). Each language defines a comprehensive set of symbols with precise semantics, together with detailed syntactic rules to enable maps are to be interpreted.

PD corresponds more closely to BioPAX than ER and AF and is therefore the recommended language for visualizing BioPAX models. Most PD symbols map one-to-one with BioPAX classes. Visualization with ER requires a non-trivial transformation which is still not completely specified. AF visualization requires a lossy transformation similar to the ones provided in PathwayCommons Sif rules.

This chapter provides a quick start guide for visualizing BioPAX Models with SBGN-PD. Several figures are copied from the SBGN documentation and publication.

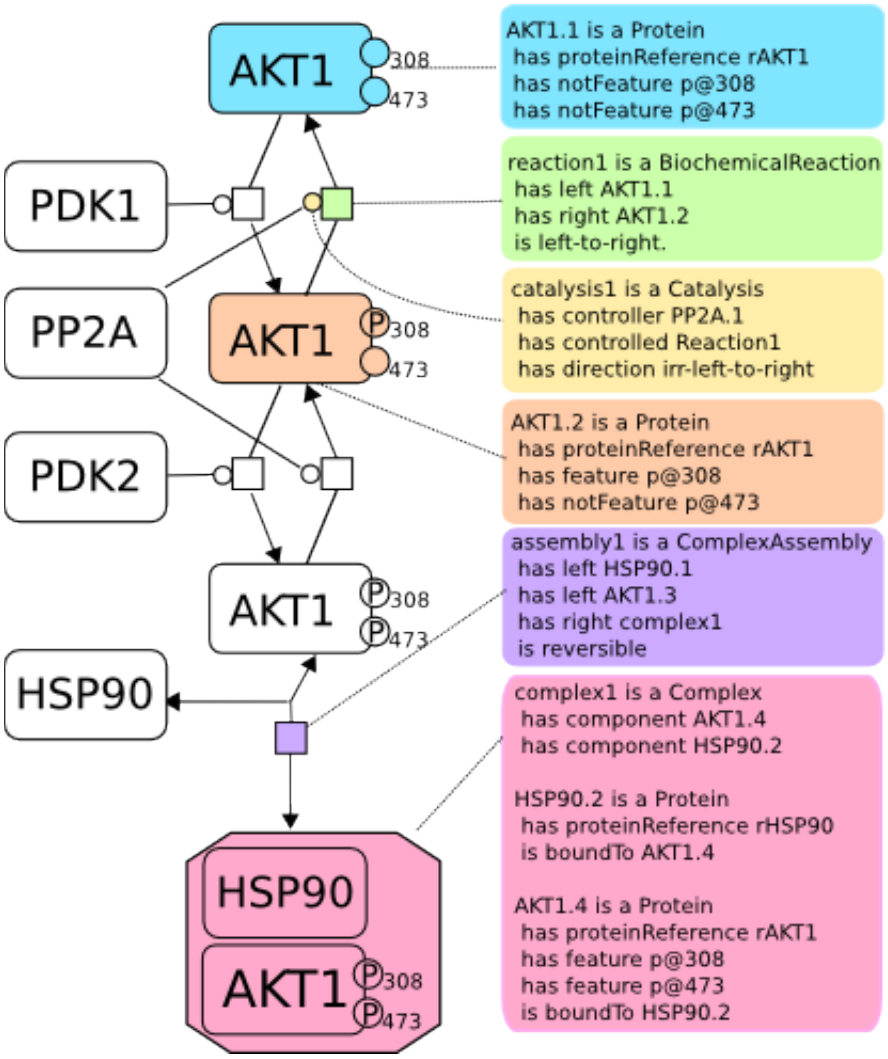
Terms and Concepts

The figure below shows the main SBGN glyphs. Generally, SBGN **Entity Pool Nodes** (EPN) map to **Physical Entities** in BioPAX and **Process** nodes map to **Interactions**. Currently, gene and genetic interactions diverge significantly from this pool/process logic and these BioPAX



classes can not be represented appropriately in SBGN.

The following figure illustrates how BioPAX representation of AKT phosphorylation can be visualized in SBGN.

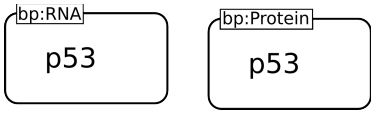


Patterns for common representation issues and best practices

Although BioPAX and SBGN share significant overlap, there are still several mismatches in representation. BioPAX and SBGN groups are working actively to resolve these issues. This chapter lists known issues or commonly asked questions and offers solutions when possible.
Please make sure that you check SBGN PD FAQ(http://sbgn.org/Documents/PD_L1_FAQ) first as it explains many commonly occurring SBGN issues. This section covers BioPAX specific issues that are not covered there.

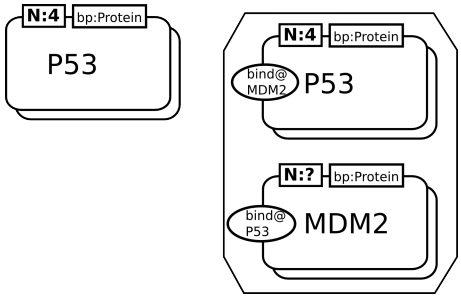
Choosing a good label for glyphs.
Depending on application needs, either the short-name of the entity or the owner entity reference is generally recommended. If this is not present a good fallback option might be to use the standard name**.**

A single EPN type maps to multiple physical entity classes in BioPAX. In SBGN, *macromolecule*, a type of EPN, maps to multiple BioPAX classes, namely *DNA*, *RNA* and *Protein*. Similarly, *nucleic acid feature* maps to *DNARegion* and *RNARegion*. In these cases, each specific BioPAX subclass should be specified using the *unit of information* glyph with prefix bp. For example:

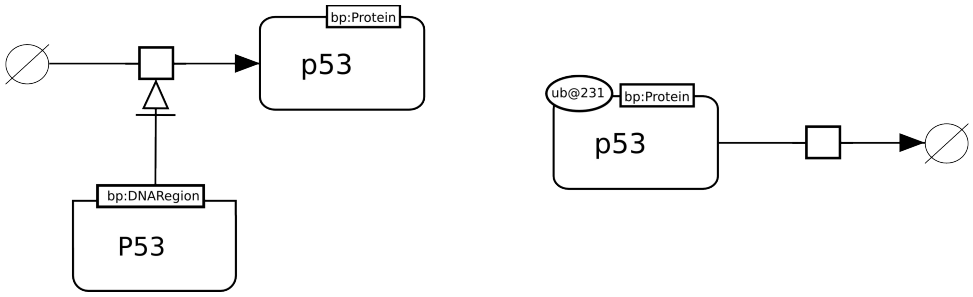


BioPAX Complex can be either represented with SBGN macromolecule or SBGN complex. SBGN *multimer* should always be used for representing *components* in a *complex* with *stoichiometries* bigger than one. If the *complex* is homomeric then it is ok to not use the container node SBGN *complex*.

Representing BioPAX Binding. Binding features are represented similar to other features. Unfortunately binding partners can not currently be represented explicitly in SBGN. As a best practice, the type of feature should be “bind” and value should be the name of the bound entity's label. For example:



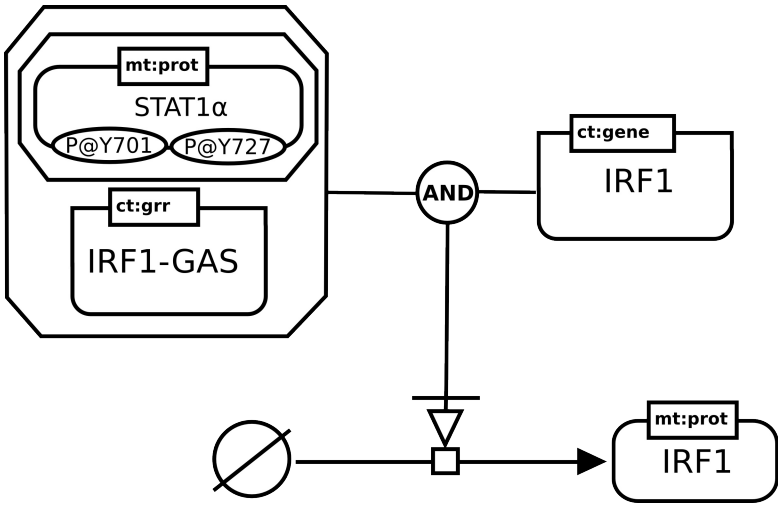
BioPAX TemplateReaction and Degradation are not represented in SBGN. Although these two interactions are not explicitly specified in SBGN, it is possible to capture most of their logic using source/sink EPN. For example:



Commonly occurring currency molecules such as ATP are difficult to layout/visualize because of their high connectivity. One common trick employed from the earliest days of metabolic diagrams is to duplicate such nodes in the graph. SBGN has a special clone marker for supporting this use case. Recommended best practice is to duplicate these nodes for each reaction they take part in.

Drawing pathway individuals. In BioPAX pathway class is overloaded and maps to three different SBGN PD elements. If it is a blackbox pathway for representing "starting" or "end" points of a signaling event, it should be represented with the perturbation/observable nodes. If it is used for complexity management, it should be represented with a submap node.

Drawing a control on a control, or multiple controllers. This is not directly supported in SBGN but logical operators can be used to represent most of the logic. For example:



BioPAX Generics. Generics, both protein families and generic features are currently not covered by SBGN. We are actively working to resolve these issues.

Pathway Modeling

Mathematical modeling to understand the dynamics of a pathway system is a frequent use of pathway information. Qualitative modeling requires information about components in the pathway and their connections, as well as some qualitative knowledge of rates (e.g. fast, slow) and concentrations of the components (e.g. high, medium, low). Quantitative modeling additionally requires such things as measured rate constants, stoichiometry and initial concentrations in order to quantitatively predict pathway behavior. Many tools are available for this type of modeling, and the SBML (http://sbml.org) and CellML (http://www.cellml.org) standards are available to describe the models, which many tools support. While BioPAX does not contain enough information to describe a pathway model as well as SBML and CellML, there are two envisioned pathway modeling related use cases:

Using BioPAX as metadata for SBML and CellML

SBML and CellML, as model representation languages, focus on representing the structure, parameters and mathematical description of a pathway model. BioPAX focuses on molecule and interaction classification schemes and database cross-referencing for pathway components. BioPAX and SBML or CellML could be linked together when a user wants both a full model description and information about types of pathway components and database links. A hybrid XML document containing BioPAX and SBML or CellML elements that are tied together using the CellML metadata standards could be created that fills this need.

Pathway analysis using logical inference

One advantage of representing BioPAX pathway data in OWL format is the availability of logical inference tools that support OWL. These tools are useful for analyzing pathways. For example, given a metabolic network model for an organism in BioPAX format, a known minimal nutrient media for that organism and the set of compounds essential for growth under one set of living conditions, then a transitive closure computation of the minimal nutrient set can be used to verify if the metabolic network model of the organism is sufficient to explain growth. If any essential compound is not reachable through the network from the minimal nutrient list, then the network model is incomplete.

9 Glossary

Some of the following definitions may be specific to BioPAX.

Biological pathway: A pathway is a series of molecular interactions and reactions (or other biological relationships), often forming a network. For molecular pathways, the start and end points are often defined by observation of a detectable phenotype after stimulation or perturbation, such as observing gene expression after stimulating the cell with a peptide growth hormone.

Class: Used in knowledge representation to represent a category of things. A specific member of a class is called an instance or individual.

Data exchange format: Any data format, usually electronic, used to exchange data.

Instance: A particular member of a class. Known as 'individual' in OWL.

n-ary specifier: A design pattern used in BioPAX to attach additional information to instances in particular properties. For examples, the Stoichiometry class represents an n-ary specifier that attaches stoichiometry information to physical entities in biochemical reactions and complexes.

Ontology: A system for describing knowledge, a conceptualization of a domain of interest usually made up of any or all of the following: concepts (classes), relations, attributes, constraints, objects, values. http://www-ksl.stanford.edu/kst/what-is-an-ontology.html

OWL: Web ontology language, a proposed W3C standard, is an extension of RDF to support ontologies. It provides semantics for classes and subclasses, instances, and relationships. http://www.w3.org/TR/owl-features/

Property: A 'field' or 'member' of a data structure. Known as a 'slot' in many knowledge representation systems.

Protégé: Protégé ontology and knowledge base editor. A software tool to build an ontology and manage instances of classes defined in that ontology. http://protege.stanford.edu/

RDF: Resource Description Framework, a proposed W3C standard, allows description of basic relationships between objects (subject-predicate-object semantics). OWL is built on top of RDF.

http://www.w3.org/TR/rdf-primer/

10 References

1) Alberts, B. (2002). Molecular biology of the cell. New York, Garland Science.

2) Alfarano, C. a. A., CE and Anthony, K. and Bahroos, N. and Bajec, M. and Bantoft, K. and Betel, D. and Bobechko, B. and Boutilier, K. and Burgess, E. and others (2005). "The biomolecular interaction network database and related tools 2005 update." Nucleic acids research **33**(Database Issue): D418.

3) Baxevanis, A. D. a. O., B.F.F. (2001). Bioinformatics: a practical guide to the analysis of genes and proteins. New York, John Wiley and sons.

4) Demir, E. a. B., O. and Dogrusoz, U. and Gursoy, A. and Nisanci, G. and Cetin-Atalay, R. and Ozturk, M. (2002). "PATIKA: an integrated visual environment for collaborative construction and analysis of cellular pathways." Bioinformatics **18**(7): 996.

5) Edwards, J. a. C., M. and Palsson, B. (2002). "Metabolic modelling of microbes: the flux-balance approach." Environmental Microbiology **4**(3): 133.

6) Harris, M. a. C., J. and Ireland, A. and Lomax, J. and Ashburner, M. and Foulger, R. and Eilbeck, K. and Lewis, S. and Marshall, B. and Mungall, C. and other (2004). "The Gene Ontology (GO) database and informatics resource." Nucleic acids research **32**(Database issue): D258.

7) Hermjakob, H. a. M.-P., L. and Bader, G. and Wojcik, J. and Salwinski, L. and Ceol, A. and Moore, S. and Orchard, S. and Sarkans, U. and Von Mering, C. and others (2004). "The HUPO PSI's molecular interaction format—a community standard for the representation of protein interaction data." Nature Biotechnology **22**(2): 177-138.

8) Hucka, M. a. F., A. and Sauro, HM and Bolouri, H. and Doyle, JC and Kitano, H. and others (2003). "The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models." Bioinformatics **19**(4).

9) Joshi-Tope, G. a. G., M. and Vastrik, I. and D'Eustachio, P. and Schmidt, E. and De Bono, B. and Jassal, B. and Gopinath, GR and Wu, GR and Matthews, L. and others (2005). "Reactome: a knowledgebase of biological pathways." Nucleic acids research **33**(Database Issue): D428.

10) Kanehisa, M. a. G., S. and Kawashima, S. and Okuno, Y. and Hattori, M. (2004). "The KEGG resource for deciphering the genome." Nucleic acids research **32**(Database Issue).

11) Karp, P. D. a. R., M. and Saier, M. and Paulsen, I.T. and Collado-Vides, J. and Paley, S.M. and Pellegrini-Toole, A. and Bonavides, C. and Gama-Castro, S. (1996). "Database links are a foundation for interoperability." Trends in Biotechnology **14**(8): 273-279.

12) Karp, P. D. a. R., M. and Saier, M. and Paulsen, I.T. and Collado-Vides, J. and Paley, S.M. and Pellegrini-Toole, A. and Bonavides, C. and Gama-Castro, S. (2002). "The ecocyc database." Nucleic acids research **30**(1): 56.

13) Krieger, C. J. a. Z., P. and Mueller, L.A. and Wang, A. and Paley, S. and Arnaud, M. and Pick, J. and Rhee, S.Y. and Karp, P.D. (2004). "MetaCyc: a multiorganism database of metabolic pathways and enzymes." Nucleic Acids Research **32**(Database Issue): D348.

14) Lemer, C. a. A., E. and Couche, F. and Fays, F. and Santolaria, X. and Janky, R. and Deville, Y. and Richelle, J. and Wodak, S.J. (2004). "The aMAZE LightBench: a web interface to a relational database of cellular processes." Nucleic Acids Research **32**(Database Issue): D443.

15) Mi, H. a. L.-U., B. and Loo, R. and Kejariwal, A. and Vandergriff, J. and Rabkin, S. and Guo, N. and Muruganujan, A. and Doremieux, O. and Campbell, M.J. and others (2005). "The PANTHER database of protein families, subfamilies, functions and pathways." Nucleic acids research **33**(Database Issue).

16) Murray-Rust, P. a. R., HS (2003). "Chemical markup, XML, and the World Wide Web. 4. CML schema." Journal of chemical information and computer sciences **43**(3).

17) Overbeek, R. a. L., N. and Pusch, G.D. and D'Souza, M. and others (2000). "WIT: integrated system for high-throughput genome sequence analysis and metabolic reconstruction." Nucleic acids research **28**(1): 123.

18) Ruttenberg, A. a. R., J. and Luciano, J. (2005). Experience using OWL DL for the exchange of biological pathway information. Proc. of the First OWL Experiences and Directions Workshop. Galway, Ireland,.

19) Stein, L. (2002). "Creating a bioinformatics nation." Nature **417**: 119-120.

20) Weininger, D. (1970). "SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules." Proc. Edinburgh Math. SOC **17**: 1-14.

21) Wheeler, D. L. a. B., T. and Benson, D.A. and Bryant, S.H. and Canese, K. and Chetvernin, V. and Church, D.M. and DiCuccio, M. and Edgar, R. and Federhen, S. and others (2006). "Database resources of the national center for biotechnology information." Nucleic acids research.

22) Yamamoto, S. a. K., T. and Ono, N. and Yamagata, Y. and Takagi, T. and Fukuda, K. (2003). "INOH: A Textual Knowledge Based Pathway Database." GENOME INFORMATICS SERIES: 679--680.

23)

24)

25)

Further Reading

- www.biopax.org
- www.biopaxwiki.org
-

A selection of research articles making use of BioPAX

- Roldán-García, M. d. M. (2009). KA-SB: from data integration to large scale reasoning BMC Bioinformatics 1-14.
- Sahoo, S., O. Bodenreider, et al. (2008). An ontology-driven semantic mashup of gene and biological pathway information: Application to the domain of nicotine dependence. Journal of Biomedical Informatics. **41:** 752-765.
- Anna Bauer-Mehren, L. I. F., Michael Rautschka and Ferran Sanz (2009). "From SNPs to pathways: integration of functional effect of sequence variations on models of cell signalling pathways " BMC Bioinformatics **10(Suppl 8):S6**
- Lister, A. L. a. L., P. and Pocock, M. and Wipat, A. (2009). "Annotation of SBML Models Through Rule-Based Semantic Integration." Nature Precedings.
- Matthew E. Holford, H. R., Hongyu Zhao,Kenneth K. Kidd, and Kei-Hoi Cheung (2009). "Semantic Web-Based Integration of Cancer Pathways and Allele Frequency Data." Cancer Informatics **8**(19-30).
- Ethan Cerami , E. D., Nikolaus Schultz, Barry S. Taylor, Chris Sander (2010). "Automated Network Analysis Identifies Core

Pathways in Glioblastoma." Plos One **5**(2).

Citing BioPAX

- Demir et. al. (2010) BioPAX - A Community Standard for Pathway Data Sharing. (accepted Nature Biotechnology)

11 Appendices

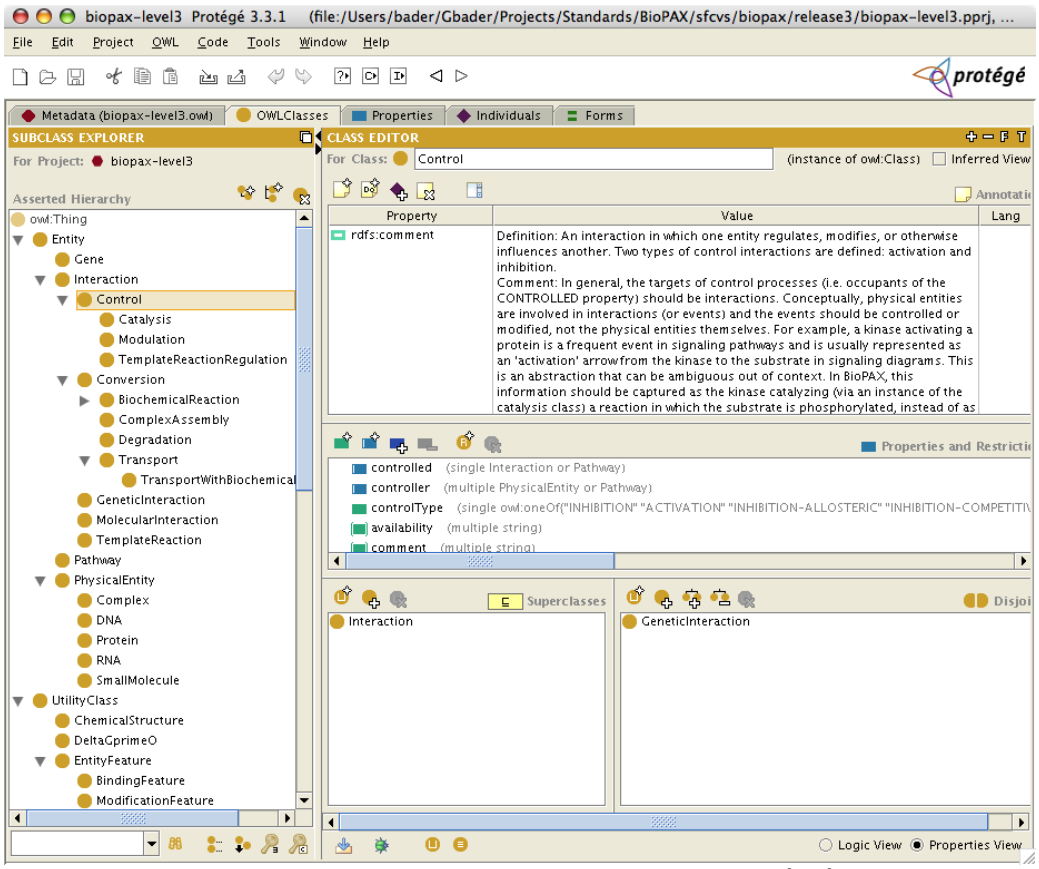
Appendix A - How To

Creating a knowledge-base using BioPAX and Protégé

Protégé is an open-source ontology and knowledge-base editor from Stanford University. It can be used to view and edit the BioPAX ontology and to create a database of instances of BioPAX classes. Download Protégé from http://protege.stanford.edu/ Downloading the current stable release and not the beta release is recommended. Make sure your Protégé version comes with OWL support. Follow the instructions for installing Protégé. These instructions were written for Protégé 3.3.1.

To import from a local copy of the BioPAX OWL file:

1. Load the BioPAX OWL file via the "File → Open…" menu item. In the resulting dialog box, select "Supported File (*.pprj, *.owl)" in the Files of Type drop down menu and browse to the BioPAX OWL file on the local computer disk drive. Select the file and press "OK". Protégé will load the BioPAX ontology.
2. Upon loading, the BioPAX ontology will be visible OWLClasses Tab. Ensure the properties view is selected, instead of the logic view.
3. Use the Individuals tab to create instances.

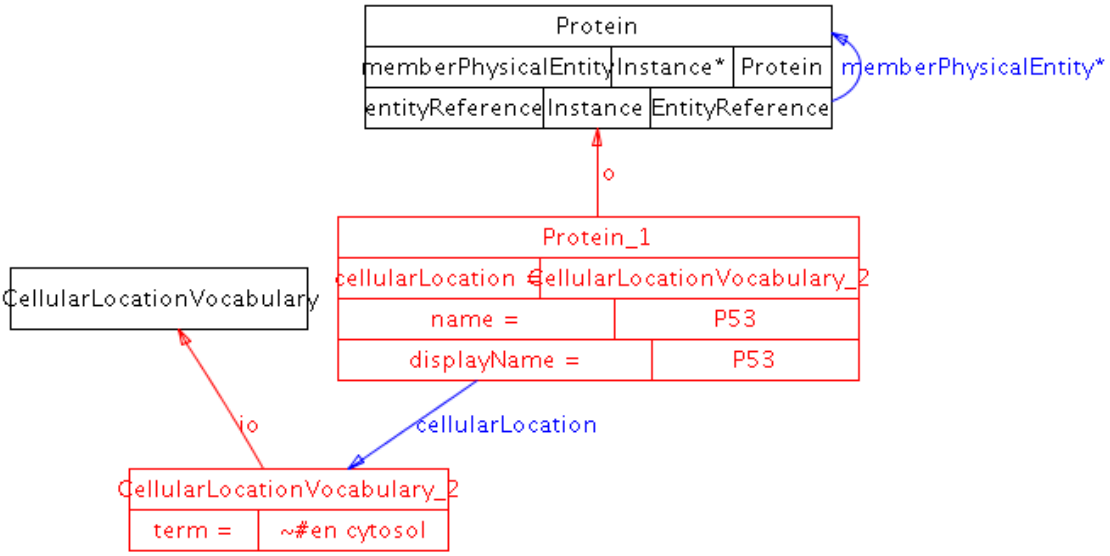


Note: This method of importing BioPAX into Protégé does not prevent inadvertently made changes to the imported BioPAX classes; changing the ontology is not recommended if the instance data are meant to be shared.

Protégé can be used as a full-fledged customizable database and data entry system, although it requires programming effort. For example, Reactome (http://www.reactome.org) uses Protégé as its backend system. If used this way, it may be desirable to modify the BioPAX ontology and create inverse properties for convenience. These properties should be removed in shared data files in order to make them compliant with the BioPAX standard.

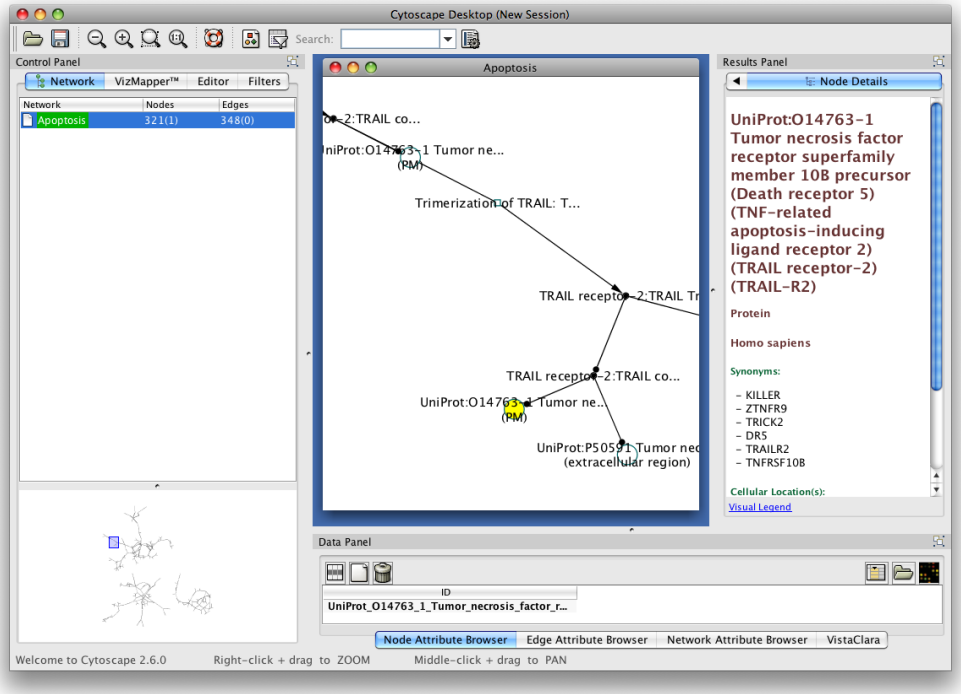
Viewing Classes and Instances Graphically in Protégé

Instances can be graphically viewed with a number of Protégé plugins that ship with the 'Full' Protégé download. For instance, the Ontoviz plugin enables a highly customized view of classes, their properties and instances in an OWL file, while OWLViz displays visually appealing class diagrams. Ontoviz will provide a structural view of the instances and classes, like the following example:



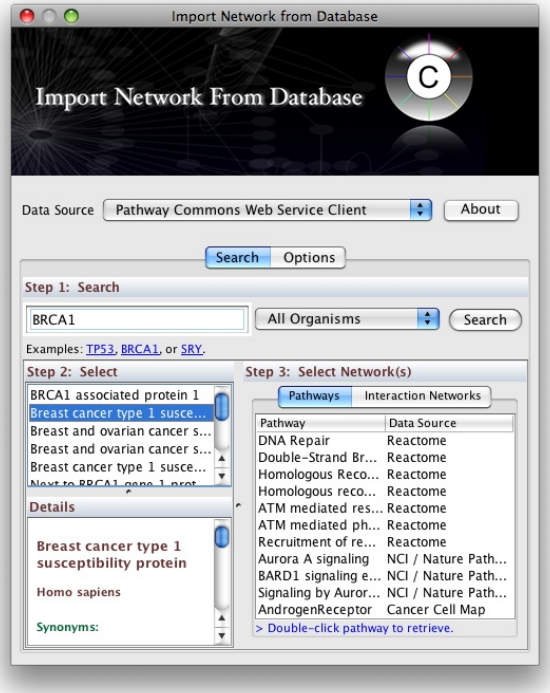
Viewing Pathways Graphically in Cytoscape

BioPAX OWL files containing pathway data (instances) can be viewed by Cytoscape (www.cytoscape.org) by importing multiple file types. The resulting view is interactive and clicking on nodes will provide more details about them, such as protein information, and links back to originating databases.



Pathways can also be imported by Pathway Commons

(www.pathwaycommons.org), which is being developed as a convenient single point of access for publicly available pathway information. Access Pathway Commons data through File→ Import→ Network from web services… . A dialog box like the following will appear, providing query features for Pathway Commons.



Appendix B - FAQ

Q. Why is there no ComplexReference class in BioPAX Level 3?
A. A ComplexReference class would be useful to define generic complexes. However, this is not required because a complex is defined by its components and those components can be defined as generic. Also, there is a problem with simply introducing a ComplexReference class in that it can't handle the case where a complex has multiple subunits of A, but some are 'A' and some are 'A with features' (e.g. A-phosphorylated).

Q. Can BioPAX encode control flow or activity flow networks e.g. a activates b inhibits c?
A. No, as this is ill defined, since the same entity can be an activator or an inactivator in different contexts. However, you can encode this as A controls a conversion from B to B active.

Q. How should cleavage sites and cleavage products of proteins, DNA and RNA be represented?
A. Cleavage sites should be represented as a binding site of a cleavage enzyme. The cleavage products should be represented using the FragmentFeature as a feature of a physical entity.

Q. How should I represent the stoichiometry of homodimers? Should I represent 2 x A or A + A?
A. It should be represented using only one A and specifying a stoichiometry of 2 using the participantStoichiometry property of Conversion or the componentStoichiometry of Complex.

Q. How should I represent EntityReferences that are a mix of concrete and generic entities?
A. Each EntityReference has an entityReferenceType property that can be used to specify whether the EntityReference is concrete or generic (and which type of generic entity it is). The generic EntityReference would be specified e.g. as a homology group, and the EntityReferences that make up the generic EntityReference would all be specified as concrete.

Q. How do I represent a disulfide bond?
A. Create a CovalentBindingFeature for both Cysteine residues that will be
connected by the disulfide bond. Define the modification type using the
CovalentBindingFeature->modificationType property and use the PSI-MI
controlled vocabulary modification feature term 'half cystine', which

has a synonym 'half disulfide bond' (Term: MI:0832 <http://www.ebi.ac.uk/ontology-lookup/?termId=MI%3A0832>)

Q. How do I represent binding interactions within molecules (intramolecular binding)?
A. You use a BindingFeature to define a binding site, but set the intramolecular flag to true.

Q. How do I represent small molecule binding to a protein?
A. You can add BindingFeatures to both molecules. The BindingFeature on the protein may have a featureLocation describing the binding site on the protein, however, the BindingFeature attached to the small molecule will not have a featureLocation. Currently BioPAX cannot represent submolecular binding sites on small molecules.

Q. Why aren't ExperimentalForm features (e.g. His-tagged protein) not defined using PhysicalEntity features?
A. ExperimentalForm features are used just to detect the interaction, but should not be considered part of the interaction. These are separated to avoid mixing the modeled version of the physical entity with the underlying raw data.

Q. How is the BioPAX ontology different than the Gene Ontology?
A. The Gene Ontology (GO) is a controlled vocabulary of gene function and BioPAX is similar to a data model for representing pathways. GO only contains terms and their relationships. Each term is a class and there are over 25,000 terms. There are no properties that further describe each term. Also, no instances are created for terms. On the other hand, BioPAX classes have many properties and are meant to be instantiated with values for most properties. This is more like the traditional object or data model used in databases. Also, BioPAX has less than 100 classes.

Q. How is the BioPAX ontology different than the Systems Biology Markup Language (SBML) or CellML? see also Q.What does BioPAX not do?
A. SBML and CellML represent mathematical descriptions of pathways while BioPAX focuses on semantic details of discrete pathway models. All of these groups coordinate to work towards compatibility of these standards that are designed for different uses.

Q. How is the BioPAX ontology different than the Proteomics Standards Initiative-Molecular Interaction (PSI-MI) format?
A. The PSI-MI represents molecular and genetic interactions, while BioPAX represents pathway information and molecular and genetic

interactions. The molecular and genetic interaction descriptions in BioPAX were designed based on that in the PSI-MI, in collaboration with the PSI. BioPAX makes heavy use of the PSI-MI controlled vocabularies.

Q. What does BioPAX Level 3 not cover that would be useful to cover in the future?

A. A number of things, including:

- Additional types of physical entities, like cells and photons
- Input/output of pathways to help use black box pathways (Nature PID request)
- Generic interactions. Some cases are covered by using generic physical entities.
- Better support for multi-cellular pathways e.g. host-pathogen, cell-cell
- Additional physical entity states: conformational change, open/closed states of channels (INOH request)
- Evidence on relationships e.g. add evidence to intra and inter molecular interactions between BindingFeature, and for branching pathwaySteps (INOH request)
- Submolecular sites on small molecules
- Support for approximate stoichiometry or infinite stoichiometry (INOH request)
- Support for polymerization reactions and stoichiometry of participants in these (INOH request)
- Support for modifications of generic complexes when you don’t know the modification on a subunit
- Addition of evidence to cellular location. This would likely require moving cellular location to be a type of EntityFeature. (INOH request)

Q: I'm not sure why you have introduced a Degradation class. Isn't degradation ideally a Biochemical reaction?

A: Degradation is a different than biochemical reaction because

- We do not care about the products
- Products can be combinatorially many
- It is non-stoichiometric
- It does not conserve mass.
-

Q: Is it valid to represent a transport reaction where an entity on the left has no location (or unkown) and the entity on the right has a location?

A: As a best practice we recommend not to model that which is unknown, a transport is only valid if both participants in the interaction have a known location which is different.

Q: Is a conversion that results in a covalently attached complex a ComplexAssembly or a BiochemicalReaction?
A: It is a BiochemicalReaction,since it involves a covalent bond.

Q: What does BioPAX not do?
Dynamic and quantitative aspects of biological processes, including temporal aspects of feedback loops and calcium waves are not covered in BioPAX. The SBML and CellML mathematical modeling languages and a growing software toolset supporting biological process simulation cover these aspects. Detailed information about experimental evidence supporting a pathway map is useful for recognizing the relative levels of support for different pathway aspects. This information is only included in BioPAX for molecular interactions, because that was already defined by the Proteomics Standards Initiative Molecular Interactions (PSI-MI) language and it was reused.

BioPAX does not aim to standardize how pathways should be visualized, however, work is coordinated with the Systems Biology Graphical Notation (SBGN, http://sbgn.org) community to ensure that SBGN can be used to visualize BioPAX pathways. Currently, most BioPAX concepts can be visualized using SBGN process description (PD) and SBGN activity flow (AF) diagrams and a mapping of BioPAX to SBGN entity relationship (ER) diagrams is in section

BioPAX development is coordinated with the other standardization efforts to ensure complementarity and compatibility.

Appendix C - Design Principles

Flexible: Biological pathway data are organized and represented in various ways depending on the type of data and its intended use. BioPAX must support the most frequently used representations to be widely accepted. Of course, there is a trade-off that must be considered: increased flexibility may increase data integration overhead. For example, the issue of semantic mapping between different representation styles must be dealt with when users wish to integrate BioPAX data sets that use different representations. Therefore, BioPAX should strike a reasonable balance between flexibility and rigidity by allowing multiple preferred representations and providing best practice recommendations to encourage consistent data representation. Through this community process, increased data sharing will be enabled.

Extensible: Biological pathway data are available in various forms and at varied levels of detail. BioPAX aims to initially support the most frequently used types of pathway data and levels of detail and to progressively broaden support for additional pathway data types and finer detail through a leveled approach. The root class structure of BioPAX was designed to be extensible for this reason. Many parts of the BioPAX ontology, such as internal controlled vocabularies and many of the intermediate level classes, may be extended in future BioPAX levels. All efforts will be made to keep future levels backwards compatible. Note: this worked well between BioPAX Level 1 and 2, but Level 3 broke backwards compatibility to implement important new features.

Encapsulation: Pathway data depends on many primary databases of physical entities (e.g. proteins, small molecules, etc.). Many pathway data sets reference physical entities using database identifiers. Because of the varied nature of the physical entity databases, resolving these identifiers in a general way can be difficult, especially for the naïve user. Frequently used data about the physical entities (e.g. sequence for proteins, structure for small molecules) is optionally present (encapsulated) in the BioPAX format for convenience.

Compatible: BioPAX uses existing standards for encoding biological pathway information to avoid “re-inventing the wheel”. Specifically, pointers to the Gene Ontology (GO), and instances of Chemical Markup Language (CML) and the SMILES format are used in various properties in the ontology. Also, compatibility with other pathway standards, such as SBML, CellML, and PSI-MI has influenced the design of many BioPAX features.

Computable: BioPAX stores data in a format that supports many different types of computational analysis. Values are strongly typed and the class structure is clearly defined. A wide range of computational tasks, from simple reading and parsing of a BioPAX file to logical inference based on the data, are supported. The OWL version of the BioPAX ontology is written in the OWL-DL sublanguage and is thus intended to be interpretable by description logic software such as RACER (http://www.sts.tu-harburg.de/~r.f.moeller/racer/). However, please, see Appendix E - BioPAX Non-Conformance with OWL Semantics regarding our use of OWL.Error: Reference source not found regarding our use of OWL.

Appendix D - Level and Version number conventions

BioPAX level numbers indicate the relative scope of the ontology. BioPAX Level 1 focuses on metabolic pathway data; Level 2 expand this scope to include molecular binding interactions; Level 3 adds support for signal transduction pathways, gene regulatory networks and genetic interactions. BioPAX level numbers are always whole numbers (e.g. Level 1, version 1.0).

In addition to the level numbers, BioPAX version numbers indicate the relative stage of development of each level. Version numbers are a composite of two individual integers: the major version number and the minor version number separated by a decimal point to form the composite version number (e.g. Level 1, version 1.1). The major version number appears before the first decimal point and is only incremented when an update is likely to affect existing data. Releases in which the major version is 0 are early draft releases of their respective levels (e.g. Level 1, version 0.5).

All versions of BioPAX are available in the following directory on the BioPAX website:
http://www.biopax.org/Downloads/

Major versions of each level of BioPAX are always available in this directory:
http://www.biopax.org/release/

Appendix E - BioPAX Non-Conformance with OWL Semantics

BioPAX Level 3 use of OWL does not fully conform to the OWL semantics. There is discussion in the community about the biological semantics of specific classes. BioPAX Level 3 does not fully respect OWL semantics because, until recently, they were not fully understood by the BioPAX community[19]. OWL is used similar to a UML style class hierarchy definition language with a standard XML serialization. OWL follows an "open world assumption", but BioPAX Level 3 in some cases assumes a "closed world", similar to most data schema definition languages. In particular:

- There are a number of cases where open world semantics conflict with the intended semantics of BioPAX entries. For example, in the case of metabolic reactions it is often (but not always) intended that the list of participants in the reaction is considered to be complete. However closing axioms are not added to make the OWL representation say so. On the other hand it would not be correct to say that the current BioPAX operates under closed-world semantics. One counterexample would be the representation of reactions where the catalyst is not known. Another would be assuming that a BioPAX file contains all reactions of a certain class - if a particular reaction is not found in a BioPAX file, it should not be assumed that it does not exist.
- BioPAX often uses domain and range where the meaning intended should be expressed using restrictions.
- Aspects of the meaning of some terms, such as Pathway, are embedded in the comments for the class, rather than being made explicit in the definitions of the classes. In some cases, this is due to limitations of OWL expressivity.
- Cardinality restrictions are often used as a means to suggest which properties are required or optional, rather than being considered definitional.

- BioPAX Level 3 assumes that there are no user-defined classes and that exchange files only contain instances.
- In order to ensure correct parsing of BioPAX files, and to support future levels, it is recommended that OWL-parsing tools, such as Jena, are used instead of non OWL-aware XML parsing tools such as SAX and DOM
-
-

Appendix F - Change Logs

Document:
5/28/10

- All ObjectProperty and Class Diagrams have been re-created
- New ObjectPoperties chapter
- updated FAQ
- updated References
- updated Best Practices
- updated Data Implementation
- updated Class Comments and Definitions
- New SBGN-BioPAX mapping
- New Validator Section
- New Worked Examples

OWL:
A complete change log is available from the BioPAX SourceForge CVS system at
http://biopax.cvs.sourceforge.net/biopax
Specifically:
http://biopax.cvs.sourceforge.net/biopax/Paxtools/src/org/biopax/paxtools/model/biopaxlevel3.owl?view=log
http://biopax.cvs.sourceforge.net/biopax/biopax/release3/biopax-level3.owl?view=log