SBML Level 3 Package Proposal: Annot

Dagmar Waltemath[1,*], Neil Swainston[2,*], Allyson Lister[3,*], Frank Bergmann[4], Ron Henkel[1], Stefan Hoops[5], Michael Hucka[6], Nick Juty[7], Sarah Keating[7], Christian Knuepfer[8], Falko Krause[9], Camille Laibe[7], Wolfram Liebermeister[9], Catherine Lloyd[10], Goksel Misirli[3], Marvin Schulz[9], Morgan Taschuk[3], Nicolas Le Novère[7]

[1]Database and Information Systems, University of Rostock, 18051 Rostock, MV, Germany

[2]Manchester Centre for Integrative Systems Biology, University of Manchester, Manchester, UK

[3]School of Computing Science, Newcastle University, Newcastle-Upon-Tyne, UK

[4]Department of Bioengineering, University of Washington, Seattle, WA 98195, USA

[5]Virginia Bioinformatics Institute, Virginia Tech, Washington St. 0477, Blacksburg, VA 24061, USA

[6]Control and Dynamical Systems, California Institute of Technology, Pasadena, CA 91125, USA

[7]European Bioinformatics Institute, Wellcome Trust Genome Campus, Hinxton, Cambridge, UK

[8]Institute of Computer Science, Friedrich-Schiller-University Jena, Germany

[9]Institut für Biologie, Theoretische Biophysik, Humboldt-Universität zu Berlin, Berlin, Germany

[10]Auckland Bioengineering Institute, University of Auckland, Auckland 1010, New Zealand

[*]These authors contributed equally to this work.

Proposal title

SBML Level 3 Annotation Package. (Keyword: annot).

Proposal tracking number

Number 3009839 in the SBML issue tracking system.

Version information

Version number and date of public release

This is version 1 of the Annotation package proposal. It reflects the results of the Annotation package meeting, 19–21 May 2010.

URL or this version of the proposal

Annot package proposal version 1 (2011-02).

URL for the previous version of this proposal

None.

Introduction and motivation

Annotations encode meta-information in SBML models. SBML allows users to annotate any SBML component that extends `SBase` (SBML L3 V1 Core specification, p. 15). The *Annotation* concept provides a container for optional software-generated, computer-readable content not meant to be shown to humans. The current syntax for encoding of information inside the annotation element, hereafter referred to as *Core annotation* recommends the use of a defined subset of RDF as described in the SBML L3 V1 Core specification, section 6. The Core annotation format allows the expression of relationships between SBML elements and resources referred to by values of `rdf:resource` attributes. The BioModels.net relation

qualifiers (predicates) (http://biomodels.net/qualifiers/) define the nature of the relationship (SBML L3 V1 Core specification, p. 87).

However, as annotations are independent from the model syntax and are not required for successful simulation of the models, it is proposed that it would be more suitable to define annotations in its own package. It is proposed to retain Core annotations in the SBML Level 3 Core, but to develop a Level 3 extension package to extend the possibilities of Core annotations and therefore support a richer set of meta-information that are currently not expressible. In future Levels, the original Core annotations may be completely replaced by this package.

Background

The package builds on the description of the Core annotation as currently described in the SBML L3 V1 Core specification, section 6. A short description of the Core annotation standard follows after the introduction to RDF.

Introduction to RDF

The Resource Description Framework (RDF) is a language for representing information about resources, in particular for representing metadata about web resources in the World Wide Web. The RDF Primer generalises the concept of a "web resource" to represent information about things that can be identified on the web, even when they cannot be directly retrieved on the web. RDF-encoded information can be processed by applications. The common framework provided by RDF to express the information in a standardised way leverages the loss-less exchange of information between different applications. RDF builds upon ideas from knowledge representation, artificial intelligence, and data management.

RDF Statements

The basic concept of RDF is the identification of things using Uniform Resource Identifiers (URIs). The resources are described by properties with particular property values. The specific terminology used in RDF is (see RDF Primer, section 2.1):

- **subject**: The part that identifies the thing the statement is about is called the subject.
- **predicate**: The part that identifies the property of the subject that the statement specifies is called the predicate.
- **object**: The part that identifies the value of a property is called the object.

Because of the generality characteristic of URIs, they are used in RDF to identify subjects, predicates and objects in statements. RDF statements effectively take the form of triples, allowing statements to be written in the form:

- **subject** has **predicate** whose value is **object**.

The RDF primer extends the concept of URIs to URI references, which are defined as:

- **`URIref`**: A URI reference (or `URIref`) is a URI, together with an optional fragment identifier at the end. The fragment is separated by the # character.

RDF URIs can be used to encode different kinds of information, including kinds of things, individuals, properties of things, or values of properties.

RDF refers to a resource as:

- **`resource`**: A `resource` is defined as anything that is identifiable by a URI reference (`URIref`).

Objects in RDF may either be `URIref`s, or constant values (literals). Subject and predicate cannot both be literals. Using `URIref`s as subject, predicate and object in statements supports the development and use of shared vocabularies on the web. One advantage of using `URIref`s for statement definitions is that an `URIref` allows for the more precise identification of a thing than using a sole string (e.g. `http://www.ex.org/staffif/1111` identifying a person more precisely than the string "Eric Miller").

RDF notations

RDF allows the encoded information to be modelled in different ways. One way is the representation of the information as a graph of nodes and arcs. An RDF graph is formed based on the idea that *the things being described have properties which have values, and that resources can be described by making statements [...] that specify those properties and values* (RDF Primer, section 2.1). The nodes in the graph represent the **subject** and **object** of a statement. The arc represents the **predicate**. It is directed from **subject** node to **object** node. Ellipses in the RDF graph represent `URIref`s, while boxes represent literals. A sample RDF graph is shown in Figure 1.

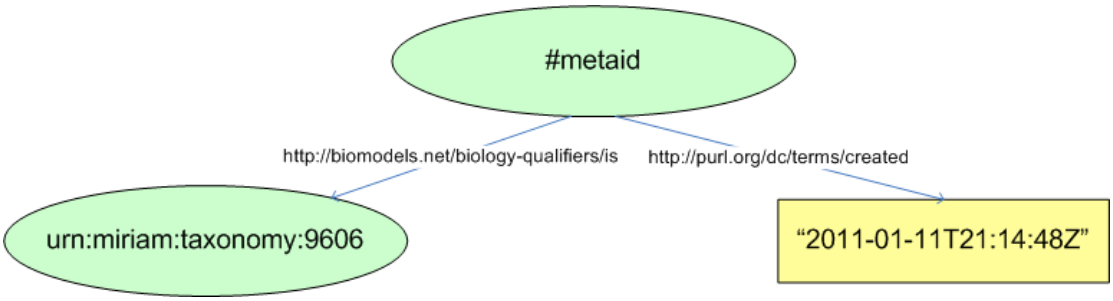

Figure 1: An example RDF graph utilising both a `URIref` and a literal.

A second way to represent RDF statements is the use of the triplet notation. It offers an alternative to the graph representation; e.g. if a graph gets too inconvenient to be drawn. Each statement of the graph is written as a single triple, consisting of the **subject**, **predicate** and **object** (in that order). A triple describes a single arc in the graph, with the **subject** being the arc's beginning and the **object** being the arc's ending. `URIref`s are put in angle brackets (`<...>`), while literals are put in quotes (`"..."`). Examples of such notation, as RDF triples, are:

Subject	Predicate	Object
`<#metaid>`	`<http://biomodels.net/biology-qualifiers/is>`	`<urn:miriam:taxonomy:9606>`
`<#metaid>`	`<http://purl.org/dc/terms/create>`	`"2011-01-11T21:14:48Z"`

Furthermore, XML can be used to represent statements in a machine readable way. The syntax for writing RDF in XML is called RDF/XML (see RDF/XML Syntax Specification). The description of a statement is enclosed in an `rdf:RDF` XML element. The statement itself is enclosed in an `rdf:Description` element; being regarded a description about the **subject** of the statement. The **subject** is referred to in the `rdf:about` attribute inside the `rdf:Description` element. The property element representing the **predicate** and **object** of the statement is nested within the containing `rdf:Description` element. The nesting indicates the application of the property on the given subject. More details on the RDF/XML syntax are given in the RDF Syntax Specification.

An example of RDF/XML representation, marking up the two statements above, is:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dcterms="http://purl.org/dc/terms/"
xmlns:bqbiol="http://biomodels.net/biology-qualifiers/">
  <rdf:Description rdf:about="#metaid_recon1_1">
    <bqbiol:is rdf:resource="urn:miriam:taxonomy:9606"/>
    <dcterms:created>2011-01-11T21:14:48Z</dcterms:created>
  </rdf:Description>
</rdf:RDF>
```

Figure 2: An example of RDF/XML utilising both a `URIref` and a literal.

SBML Core annotation standard

According to the current SBML Core annotation standard, RDF/XML is used to present the RDF statements (see Figure 2, taken from the SBML L3 V1 Core Specification, p.86).

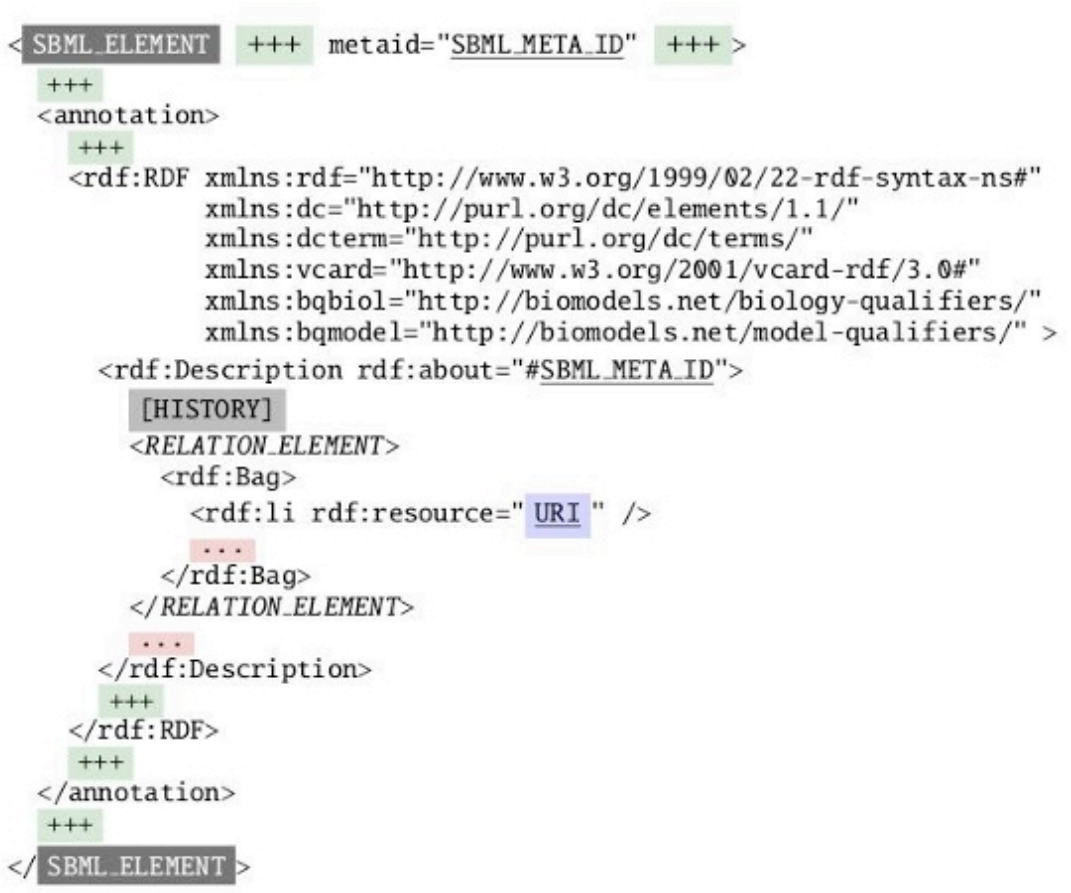

Figure 3: SBML L3 V1 Core annotation standard.

The current Core annotation schema, while written in RDF/XML, supports only a limited subset of RDF/XML. The above syntax must be followed, including the use of the mandatory `rdf:Bag` container, and the specification of the **subject** as a URI in the `rdf:li rdf:resource` attribute.

The URI link to an external resource must be perennial. To uniquely identify a controlled vocabulary term or object, the Minimum Information Required in the Annotation of Models (MIRIAM) standard is used[1]. A referenced MIRIAM URI maps to a physical web source, i.e. a URL. The connection between the addressed third-party knowledge and the annotated element is established using any of the model or biological qualifiers listed on http://www.biomodels.net/qualifiers/. If an annotation follows the proposed scheme, it is considered an SBML MIRIAM annotation. The SBML history element enables the tracking of changes as it allows the storage of the annotation creators and modification dates.

Problems with Core annotation

Statements about attributes

The Core annotation specification reuses the RDF approach of providing `rdf:Description` elements for SBML XML elements, such as species or compartment.

However, there currently does not exist a mechanism to annotate SBML attributes. See, for example, the following SBML code snippet:

```
<species metaid="metaid_0000042" id="Y" name="Intravesicular
```

[1] Le Novère N, Finney A, Hucka M, Bhalla U, Campagne F, Collado-Vides J, Crampin E, Halstead M, Klipp E, Mendes P, Nielsen P, Sauro H, Shapiro B, Snoep JL, Spence HD, Wanner BL. Minimum Information Requested In the Annotation of biochemical Models (MIRIAM). *Nature Biotechnology* 2005, **23**, 1509-1515.

```
Calcium" compartment="intravesicular" initialConcentration="0.36">
     <annotation>
       <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
ns#" xmlns:bqbiol=http://biomodels.net/biology-qualifiers/
xmlns:bqmodel="http://biomodels.net/model-qualifiers/">
         <rdf:Description rdf:about="#metaid_0000042">
           <bqbiol:isDescribedBy>
             <rdf:Bag>
               <rdf:li rdf:resource="urn:miriam:pubmed:12343565"/>
             </rdf:Bag>
           </bqbiol:isDescribedBy>
         </rdf:Description>
       </rdf:RDF>
     </annotation>
  </species>
```

Figure 4: Example of Core annotation applied to a species.

Using the current SBML annotation approach, it is not possible to annotate an attribute of an SBML element, such as the initial concentration of a species. The PubMed annotation in the example states that the `species` element as a whole is described by a particular PubMed reference (PubMed ID 12343565), while it was the intention to annotate the `species` attribute `initialConcentration`, effectively stating that the justification for the given initial concentration is described by the PubMed document with ID 12343565.

Statements about statements

With the current scheme all annotations of an SBML element are at the same level. They all relate to the element itself, but cannot be related to another statement. The ability to provide "statements about statements" is missing from Core annotation.

A simple use case is the request to annotate an annotation with the information that "this statement was added by...". A further use case would be annotations that involve non-binary relationships, such as "protein X is modified by modifier Y in position Z".

Relations between statements

In the Core annotation, it is currently not possible to define the relation between different annotations of a particular element. Apart from some conventions mentioned in the Core specification (see SBML L3 V1 Core Specification, p. 86) there is no fine-granular way of providing information on the annotation relations in a formal and specified manner.

The Core annotation standard syntactically limits the annotation of model constituents to:

```
<rdf:RDF>
 <rdf:Description rdf:about="#SBML_META_ID">
  <RELATION_ELEMENT>
   <rdf:Bag>
    <rdf:li resource="URI_1"/>
    <rdf:li resource="URI_2"/>
   </rdf:Bag>
  </RELATION_ELEMENT>
</rdf:RDF>
```

Figure 5: Current Core annotation syntax, illustrating the dependency on `rdf:Bag`.

RDF provides four different concepts to encode grouped statements, including the three Containers `rdf:Bag`, `rdf:Seq` and `rdf:Alt`, and the Collection `rdf:List` (see RDF Primer, sections 4.1 and 4.2):

- `rdf:Bag` represents an open group of resources or literals […] where there is no significance in the order of the members.
- `rdf:Seq` represents an open group of resources or literals [...] where the order of the members is significant.

- `rdf:Alt` represents an open group of resources or literals that are alternatives (typically for a single value of a property).
- `rdf:List` represents a closed group of resources or literates that consists only of the specified members.

The current Core annotation is restrictive as it does not allow the use of other containers than `rdf:Bag`, which only groups a set of statements, without implying any further semantics on the meaning of that group. Therefore, considering the above example, there is no way of currently determining what, if anything, the relationship is between `URI_1` and `URI_2`.

Examples of the ambiguity that can be caused by this limitation are highlighted in the following two examples. In the first example, the container `rdf:Bag` is used to define the relationship between two alternative annotations for glucose. In the second example, `rdf:Bag` is used to define the relationship between two components of a complex.

The first example effectively demonstrates an implied "or" relationship between two alternative means of annotating glucose (with a ChEBI term or a KEGG Compound term):

```
<species id=”glc" metaid=”meta_glc” name=“Glucose”>
 <annotation>
  <rdf:RDF>
   <rdf:Description rdf:about=”#meta_glc”>
    <bqbiol:is>
     <rdf:Bag>
      <rdf:li rdf:resource="urn:miriam:obo.chebi:CHEBI%3417234"/>
      <rdf:li rdf:resource="urn:miriam:kegg.compound:C00234"/>
     </rdf:Bag>
    </bqbiol:is>
   </rdf:Description>
  </rdf:RDF>
 </annotation>
</species>
```

Figure 6: Current Core annotation of species representing the simple molecule glucose.

`glc` is *either* `urn:miriam:obo.chebi:CHEBI:17234` **or** `urn:miriam:kegg.compound:C0023`, but not both.

The second example demonstrates an implied "and" relationship between two components of a complex (represented by a UniProt term for the protein, and a ChEBI term for the ligand):

```
<species id="Ca_calmodulin" metaid="cacam”>
 <annotation>
  <rdf:RDF>
   <rdf:Description rdf:about="#cacam”>
    <bqbiol:hasPart>
     <rdf:Bag>
      <rdf:li rdf:resource="urn:miriam:uniprot:P62158"/>
      <rdf:li rdf:resource="urn:miriam:kegg.compound:C00076"/>
     </rdf:Bag>
    </bqbiol:hasPart>
   </rdf:Description>
  </rdf:RDF>
 </annotation>
</species>
```

Figure 7: Current Core annotation of species representing the complex calcium calmodulin.

`Ca_calmodulin` has parts `urn:miriam:uniprot:P62158` **and** `urn:miriam:kegg.compound:C00076`.

The problem is that the relationship is implied: it is not made explicit by the container (`rdf:Bag`) used to define the relationship.

Furthermore, no clear definition of the different or similar meanings between using a list of

references in one `rdf:Bag` element as opposed to using a single `rdf:Bag` element for each reference is given. Consider the following two examples:

```
<rdf:RDF …>
 <rdf:Description rdf:about="#metaid_0000001">
  <bqbiol:is>
   <rdf:Bag>
    <rdf:li resource="x"/>
    <rdf:li resource="y"/>
   </rdf:Bag>
  </bqbiol:is>
 </rdf:Description>
</rdf:RDF>
```

```
<rdf:RDF …>
 <rdf:Description rdf:about="#metaid_0000001">
  <bqbiol:is>
   <rdf:Bag>
    <rdf:li resource="x"/>
   </rdf:Bag>
  </bqbiol:is>
  <bqbiol:is>
   <rdf:Bag>
    <rdf:li resource="y"/>
   </rdf:Bag>
  </bqbiol:is>
 </rdf:Description>
</rdf:RDF>
```

Figure 8: Current Core annotation examples indicating ambiguity between uses of different syntax to represent annotation with multiple resources.

Negative statements

The current Core annotation scheme does not allow for the definition of negative statements. That is, to make statements along the lines of "protein X is NOT phosphorylated".

Predicates and qualifiers

To satisfy RDF, predicates should be nouns, representing properties of the subject, rather than verbs as they are in the Core annotation. RDF triples should follow the pattern, "SUBJECT has PREDICATE whose value is OBJECT". Core annotations result in nonsensical RDF triples such as "SPECIES has IS_DESCRIBED_BY whose value is PUBMED:12345". It is proposed that the existing Biomodels.net predicates be updated, such that, taking the example above, "IS_DESCRIBED_BY" is replaced by "DESCRIPTION".

Doing so would allow the set of predicates (properties), and relationships between them, to be defined formally in an RDF schema (see http://www.w3.org/TR/rdf-primer/#rdfschema).

The Annot Package proposal

The following section summarises the proposals to be incorporated in the Annot package.

The examples enclosed within will use the proposed new predicates / qualifiers, as specified in the Appendix of this document.

Namespace and integration with SBML L3

The standard namespace for the Annot package is

`http://www.sbml.org/sbml/level3/version1/annot/version1`

A new version of the Annot package will be released with each new version of the Core package in order to comply with the new version of the Core (following the SBML L3 package

mechanism description).

In order to use the Annot package for SBML L3 models, the Annot namespace must be added to the `<sbml>` element namespace declarations:

```
<sbml xmlns="http://www.sbml.org/sbml/level3/version1/core" level="3"
version="1"
xmlns:annot="http://www.sbml.org/sbml/level3/version1/annot/version1"
...>
      ...
</sbml>
```

An SBML model can always be fully understood mathematically without understanding either the Core annotation or the Annot package extension annotation. Therefore, the use of the Annot package is optional. This can be defined by adding the XML attribute `annot:required` to the `sbml` element, and setting its value to `false`:

```
<sbml xmlns="http://www.sbml.org/sbml/level3/version1/core" level="3"
version="1"
xmlns:annot="http://www.sbml.org/sbml/level3/version1/annot/version1"
annot:required="false" ...>
      ...
</sbml>
```

Solutions

Statements about attributes

Sometimes, it is not only necessary to annotate an SBML element, but a more fine-grained annotation of a particular attribute of an element is needed.

The use of **XPath** (see http://www.w3schools.com/xpath/) to refer to a piece of XML inside the document is proposed. XPath is a standard technology for referencing elements and attributes inside an XML document, and it offers a well-defined scheme to do so. Furthermore, a great number of tools exist to evaluate XPath expressions.

Therefore, the `xpath` namespace is proposed, which allows the specification of any local object in the `rdf:about`:

```
rdf:about="xpath:XPathToTheObject"
```

One should use the element's id to refer to it, as in:

```
xpath://species[id='0001']/@initialConcentration
```

The following example shows an attribute annotation using the XPath notation.

```
 <species metaid="metaid_0000042" id="Y" name="Intravesicular
Calcium" compartment="intravesicular" initialConcentration="0.36">
    <annotation>
      <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
ns#" xmlns:bqbiol="http://biomodels.net/biology-qualifiers/">
        <rdf:Description
rdf:about="xpath://species[id='Y']/@initialConcentration">
          <bqbiol:description>
            <rdf:Bag>
              <rdf:li rdf:resource="urn:miriam:pubmed:12343565"/>
            </rdf:Bag>
          </bqbiol:description>
        </rdf:Description>
      </rdf:RDF>
    </annotation>
 </species>
```

Figure 9: Example of proposed Annot annotation for annotation of attributes.

The recommended way of providing the XPath statement is to:

- Avoid addressing attributes and elements by their (ordering) number.
- Use the abbreviated syntax to identify an XML element in the model by its `id`, and then refer to the particular attribute.

It would be error-prone to use the XPath concept of addressing attributes and elements by their indices, as SBML does not support elemental ordering. As such, expressions along the lines of the example below are **not** recommended for use in the Annot package. Instead, the XPath statement should be specified by a reference to its `id`, as in the example given above.

✗ `//species[7]/@initialConcentration`

✓ `//species[id='Y']/@initialConcentration`

Secondly, whenever possible, instead of providing the full paths to elements or attributes, the abbreviated syntax should be used, which first selects all elements of the given element name from the SBML model, and then limits the result set depending on the given `id`. In XPath, a double forward slash (`//`) selects from all descendants of the context node as well as the context node itself. At the beginning of an XPath expression, it selects from all descendants of the root node. For example, the following XPath expression selects all `species` elements in the document:

```
//species
```

It is suggested that this syntax be used in the Annot package, given its simplicity in comparison to the more verbose syntax, which would entire the full path to be specified:

```
/sbml/model/listOfSpecies/species
```

Statements about statements

RDF Reification

RDF Reification, the standard method of making statements about statements, as described in the RDF Primer, section 4.3, will be utilised. This approach allows statements to be assigned to other statements that have an `rdf:ID` assigned. Subsequent statements refer to this statement by specifying the `rdf:ID` in the `rdf:about` attribute of the `rdf:Description` attribute.

The following example demonstrates Reification being used to make a statement about a statement:

```
<species id="abc" metaid="meta_abc">
 <annotation>
  <rdf:RDF>
   <rdf:Description rdf:about="#meta_abc">
    <bqbiol:description rdf:ID="statement1">
     <rdf:Bag>
      <rdf:li rdf:resource="urn:miriam:pubmed:15387819"/>
     </rdf:Bag>
    </bqbiol:description>
   </rdf:Description>
   <rdf:Description rdf:about="#statement1">
    <dc:creator>John Smith</dc:creator>
   </rdf:Description>
  </rdf:RDF>
 </annotation>
</species>
```

Figure 10: Example of RDF Reification.

By adding an `rdf:ID` to the first statement (which states that the `species` has description PubMed document 15387819), a second statement can be specified about this first statement, which specifies that the first statement has a specified creator. Effectively the

second statement defines that the first statement has creator John Smith.

Person's meta-annotations

Core annotations are limited regarding specifying information on the different people involved in the model building, publishing, curating and maintaining process. In the Annot package, the use the `dc:creator` from Dublin Core to provide meta-information about persons is again proposed. However, it is proposed that such an annotation can be applied to both the `model` itself and any of the `model` sub-elements.

It is assumed that such annotations are inherited from parent nodes when a given node is not annotated with a `dc:creator`. For example, if a `model` element is annotated with a `dc:creator` but none of its sub-elements are, it is assumed that all sub-elements have been created by the `model` creator.

Non-binary relations

Related to this is the support for capturing non-binary relationships through the utilisation of blank nodes.

This example captures the statement "Hexokinase 2 is modified by phosphoserine in position 158", by specifying a blank node (`node1`) as the **object** of the modification **predicate**, and utilising this blank node as the **subject** of two subsequent statements. Note that phosphoserine is represented by the MIRIAM URN `urn:miriam:obo.psi-mod:MOD%3A00046`.

```
<species id=”x" metaid=”meta_x” name=“Hexokinase 2”>
 <annotation>
  <rdf:RDF>
   <rdf:Description rdf:about=”#meta_x”>
    <bqbiol:modification rdf:nodeID=”node1"/>
   </rdf:Description>
   <rdf:Description rdf:nodeID=”node1”>
    <bqbiol:modifier rdf:resource=”urn:miriam:obo.psi-
mod:MOD%3A00046"/>
    <bqbiol:position rdf:datatype="xsd:integer">158</bqbiol:position>
   </rdf:Description>
  </rdf:RDF>
 </annotation>
</species>
```

Figure 11: Example of the use of blank nodes.

Essentially, this specifies the following three triples:

`meta_x`	`bqbiol:modification`	`node1`
`node1`	`bqbiol:modifier`	`urn:miriam:obo.psi-mod:MOD%3A00046`
`node1`	`bqbiol:position`	`158`

Relations between statements

To enable a more detailed description of relations between statements, it is proposed to extend the current SBML annotation scheme to support all RDF Collections and Containers (`rdf:Bag`, `rdf:Seq`, `rdf:Alt`, and `rdf:List`).

The Core annotations specify that an `rdf:Bag` must be used. This, however, is unnecessary for single objects that can be specified more simply following the example syntax below:

```
<species id=”glc" metaid=”meta_glc” name=“Glucose”>
 <annotation>
  <rdf:RDF>
   <rdf:Description rdf:about=”#meta_glc”>
    <bqbiol:identity
rdf:resource="urn:miriam:obo.chebi:CHEBI%3417234"/>
   </rdf:Description>
```

```
  </rdf:RDF>
 </annotation>
</species>
```

Figure 12: Example of valid RDF/XML that does not use either a Collection or Container.

In addition to supporting all RDF Collections and Containers, the use of no Collections and Containers will be supported. Considering all RDF Collections and Containers, and taking the previous examples (see Problems with Core annotation: Relations between statements), the existing, Core annotation implied "or" relationship between two alternative means of annotating glucose (with a ChEBI term or a KEGG Compound term) can be made explicit by using the `rdf:Alt` collection:

```
<species id=”glc" metaid=”meta_glc” name=“Glucose”>
 <annotation>
  <rdf:RDF>
   <rdf:Description rdf:about=”#meta_glc”>
    <bqbiol:identity>
     <rdf:Alt>
      <rdf:li rdf:resource="urn:miriam:obo.chebi:CHEBI%3417234"/>
      <rdf:li rdf:resource="urn:miriam:kegg.compound:C00234"/>
     </rdf:Alt>
    </bqbiol:identity>
   </rdf:Description>
  </rdf:RDF>
 </annotation>
</species>
```

Figure 13: Example of using the Container `rdf:Alt` to represent an "or" relationship between resources.

Similarly, the existing, Core annotation implied "and" relationship between two components of a complex (represented by a UniProt term for the protein, and a ChEBI term for the ligand) can be made explicit by utilising the `rdf:List` collection to specify a closed set:

```
<species id="Ca_calmodulin" metaid="cacam”>
 <annotation>
  <rdf:RDF>
   <rdf:Description rdf:about="#cacam”>
    <bqbiol:part>
     <rdf:List>
      <rdf:li rdf:resource="urn:miriam:uniprot:P62158"/>
      <rdf:li rdf:resource="urn:miriam:kegg.compound:C00076"/>
     </rdf:List>
    </bqbiol:part>
   </rdf:Description>
  </rdf:RDF>
 </annotation>
</species>
```

Figure 14: Example of using the Collection `rdf:List` to represent an "and" relationship between resources.

Distinction between L3 Core and L3 Annot package annotations

To distinguish SBML Level 3 Core annotations from annotations provided through the Annot package, a new element `<annot:annotation>` from the `annot` namespace as a sibling of the current `<annotation>` element is proposed. This will allow L3 Annot package annotations, i.e. the ones in the scope of this draft proposal, and existing Core annotations to be distinguished.

The following example shows the `annot:annotation` element as a sibling of the current SBML annotation element:

```
<annotation>
      [CORE ANNOTATION]
```

```
</annotation>
<annot:annotation>
      [ANNOT PACKAGE ANNOTATION]
</annot:annotation>
```

The approach chosen here has the advantage of this approach is that it avoids further overloading of the already much used Core `annotation` element. It also allows a cleaner distinction between the Core and Annot package annotations.

The recommended practice for model annotation is the use of the Annot package, as it is less restricted in its syntax, and complies with RDF recommendations.

Cross-references and cross-element annotations

Self-references

In order to realise self-references, i.e. to refer to an element in the same document, use of the existing RDF standard will be supported:

```
<rdf:li rdf:resource="#metaID">
```

Non-URI references

The referencing of non-URI references to existing models (such as the example below), such as web addresses, URLs, or local directories, is NOT supported by this proposal.

```
<rdf:li rdf:resource="file://../models/BM02#_986127"/>
```

Negative statements

No suitable solution has been proposed for specifying negative statements. This issue will be addressed in a subsequent iteration of the Annot package.

Predicates and qualifiers

It is recognised that the current set of predicates (Biomodels.net qualifiers, http://www.ebi.ac.uk/miriam/main/qualifiers/) should be extended the RDF specification that predicates should be nouns, representing properties of the subject, rather than verbs as they are in the Core annotation (see http://www.w3.org/TR/rdf-primer/#rdfschema).

This process will be delegated to the developers of Biomodels.net. An initial mapping of existing (verb) predicates to new (noun) predicates is available in the Appendix.

Package dependencies

This package does not depend on any other SBML Level 3 package.

Prototype implementations

No prototype implementation exists as yet.

Translation to SBML Level 2

Translation of Annot package annotations back to SBML Level 2 annotations will not be supported.

Hints

Use of the Annot package

There is no way to legislate how other packages make use of the Annotation structures coming from this package. Individual packages determine how best to make use of Annotation structures.

Use of old and new annotations

Duplicating semantic information (in both Core annotation and the Annot package annotation) is technically possible, but it is considered bad practice and not recommended. Instead, it is recommended that, if Annot package annotation is to be used, these annotations should replace any existing Core annotations within the model.

Appendix

Predicates and qualifiers

It is recognised that the current set of predicates (Biomodels.net qualifiers, http://www.ebi.ac.uk/miriam/main/qualifiers/) should be extended the RDF specification that predicates should be nouns, representing properties of the subject, rather than verbs as they are in the Core annotation (see http://www.w3.org/TR/rdf-primer/#rdfschema).

This process will be delegated to the developers of Biomodels.net.

It is intended that the new predicates will coexist with the existing predicates, with the recommendation that the new set be used in preference to the existing set.

The following describes the mapping between old and new predicates. Where multiple options exist, these indicate candidate predicates that will be decided by the Biomodels.net community.

`bqmodel:is`	`bqmodel:identity`
`bqmodel:isDerivedFrom`	`bqmodel:progenitor, bqmodel:antecedent, bqmodel:ancestor, bqmodel:basis, bqmodel:base, bqmodel:foundation, bqmodel:origin`
`bqmodel:isDescribedBy`	`bqmodel:description`
`bqbiol:hasPart`	`bqbiol:part`
`bqbiol:hasProperty`	`bqbiol:property`
`bqbiol:hasVersion`	`bqbiol:version`
`bqbiol:is`	`bqbiol:identity`
`bqbiol:isDescribedBy`	`bqbiol:description`
`bqbiol:isHomologTo`	`bqbiol:homolog`
`bqbiol:isEncodedBy`	`bqbiol:encoder`
`bqbiol:encodes`	`bqbiol:encodement`
`bqbiol:isPartOf`	`bqbiol:encompassment, bqbiol:assembly, bqbiol:partship, bqbiol:parthood, bqbiol:whole, bqbiol:meronym`
`bqbiol:isPropertyOf`	`bqbiol:bearer, bqbiol:carrier`
`bqbiol:isVersionOf`	`bqbiol:consociate, bqbiol:cohort, bqbiol:superclass, bqbiol:hyponym`
`bqbiol:occursIn`	Physical containment: `bqbiol:encompassment, bqbiol:containment`
`bqbiol:occursIn`	Taxonomic instantiation: `bqbiol:instantiation`