
A Collection of Possible Features for Version 2 of SBML Level 2

Andrew Finney
afinney@cds.caltech.edu

October 6, 2004

Contents

1	Introduction	1
1.1	Terms of Reference	1
2	Diagram Layout	2
3	Use of Id Attribute	2
3.1	On SimpleSpeciesReference	2
3.2	General definition of id Attribute	2
4	Species Type	2
5	Controlled Vocabularies	3
5.1	Acknowledgement	3
5.2	Requirements	3
5.3	Implementation	4
5.4	Discussion	7
6	Assertions	7
6.1	Requirements	7
6.2	Implementation	7
6.3	Example	8
7	Variable and Flux Bounds	9
7.1	Requirements	9
7.2	Implementation	9
8	Dependent Variable Attribute on AlgebraicRule	10
9	Nested Unit Definitions	10
	References	10

1 Introduction

This document describes a number of possible extensions to SBML Level 2 that could together form SBML Level 2 Version 2. These extensions are independent of each other and any selection of these features could be incorporated into SBML Level 2 Version 2.

It is not the primary intention of the author to allocate these features to a specific version or level of SBML. The features described in this document could be introduced in SBML Level 3 for example.

1.1 Terms of Reference

The proposals described here do not describe any agreed standard nor any consensus as to what a standard should be. The proposals described here will only adopted as a new standard given the agreement of the SBML Forum. These proposals are made by the author only and are intended to provoke discussion.

2 Diagram Layout

SBML Level 2 Version 2 should include the diagram layout proposal described in (Gauges et al., 2004).

issue:

Should the diagram layout structures remain in their own namespace?

3 Use of Id Attribute

3.1 On SimpleSpeciesReference

A requirement of the Diagram Proposal is that `SimpleSpeciesReference` structures are uniquely identified in a model. To support this the `SimpleSpeciesReference` has new `Id` attribute. The value of this attribute is unique to the whole of the containing `Model`. The UML diagram for the new form of `SimpleSpeciesReference` is shown in Figure 1.

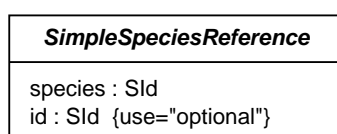


Figure 1: The UML diagram for the new form of `SimpleSpeciesReference`

3.2 General definition of id Attribute

It has been proposed that we add the `id` attribute to the `SBase` class as an optional attribute. The motivation being to reduce the change required to add new features to SBML. Whilst this works at a syntactic level the following issues remain:

- `id` on some subtypes of `SBase` require `id` to be mandatory.
- the scope of the `id` value varies depending on the element in which it is declared. For some classes that are derived from `SBase` we may not wish to define a scope for `id` values up front. In this case if the field is available it may get used with arbitrary scoping rules.

These issues apply both to Level 2 Version 1 and to proposals extend SBML beyond that version. Therefore its not recommended that `id` is made a field of `SBase`.

In the general case annotations requiring an `id` for an element which is unique to the whole XML document should use the optional `metaid` that is located on `SBase`.

4 Species Type

The feature described in this section was first proposed as part of the Multi-component species proposal (Finney, 2004).

In SBML Levels 1 and 2 there is no standard way to relate two or more `Species` representing the same chemical entity in different compartments. To overcome this limitation, under this proposal, we would introduce into a model a new list of `SpeciesType` elements. Each `SpeciesType` element represents a type of chemical entity independent of where the entity is located. A `SpeciesType` will derived from `SBase` and will have only one mandatory field `Id`. Values of this attribute will have scope across the all of the containing `model` structure.

`Species` structures will have an optional `speciesType SId` field which will refer to the type of the given `Species`. The role of the `Species` is then to locate a given chemical entity type, the `SpeciesType`, in a

specific compartment. A `Species` structure which does not have a `speciesType` value simply implies the existence of a hidden `SpeciesType` that is unique to that `Species` alone. The `SpeciesType` class and the new form of `Species` is shown in Figure 2

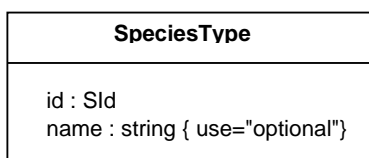


Figure 2: The UML diagram for the new form of `Species` and the `SpeciesType`

Its not clear whether we wish to create reactions which are generalized to all compartments i.e. contain `SimpleSpeciesReference` structures that refer to `SpeciesType` structures as opposed to `Species` structures.

5 Controlled Vocabularies

5.1 Acknowledgement

The requirement for a generic mechanism for controlled vocabularies was first identified, and an implementation outlined, by Akaza Research whilst consulting for Science's STKE (Sachs, 2003).

5.2 Requirements

Developers of a wide range of applications of SBML have requested that SBML be extended to include fields which contain terms from defined controlled vocabularies (CVs). Examples of these fields include:

- a *role* field on `SimpleSpeciesReference` - allowing a `Modifier` to be labelled as a catalyst for instance
- a *type* fields on `KineticLaw` - allowing a `KineticLaw` to be labelled as a hill equation for instance
- a *type* field on `AlgebraicRule` - allowing a `AlgebraicLaw` to be labelled as a conversation law for instance.

In many of these cases these fields are already being implemented in a SBML `annotation` elements but with limited interoperability between applications. The purpose of this proposed extension is to allow provide a mechanism for representing all the above examples of uses of controlled vocabularies and many others that developers will construct in the future.

These CVs that are used in SBML have the following aspects in common:

- a given term assignment does not typically affect the mathematical interpretation of a model. (A few applications may use CVs to make explicit mathematical properties of a model rather requiring the full parsing and interpretation of the SBML formulae - which should remain.)
- the CVs are typically application domain specific
- the CVs appear to be incomplete or obviously extensible and thus likely to change more rapidly than the core SBML standard.
- the CVs are not complex ontologies in fact they are not likely to be more complex than terms linked to form a directed acyclic graph (DAG) and will often consist of a list of alternative terms.
- the CVs are designed for use within an arbitrarily defined group for example in the whole community, the BioSpice consortium or with in a single laboratory.

Given these features it seems that its possible to develop a generic solution for these vocabularies. Such a solution will have the following features and requirements:

- allow the CVs to be developed independently of SBML with a clear boundary between the CVs and core SBML
- allow the relationship between an SBML object and CV terms to be defined in a generic fashion.
- allow users in model capture tools to select terms from a CV definition and associate them with a selected SBML object. The CV definition is located at a URL and the tool is not aware in advance the CV's existence. The CV format is sufficiently simple to allow for the straightforward implementation of this use case.
- allow the GO CV to be used with SBML

5.3 Implementation

The design described here allows for the definition of subject-verb-object statements where the subjects can be SBML structures (this is conceptually identical to RDF). An example might be 'species S is a protein' where 'S' is the subject, 'is a' is the verb and 'protein' is the object. The verbs and objects are always globally unique CV terms. In this approach the CV is encoded in the Gene Ontology (GO)XML Format (Gene Ontology Consortium, 2004). (The CV can be any CV in this format not just GO.) Both the GO XML format and the proposed mechanism for referring to the CV terms use RDF (Lassila and Swick, 1999; Manola and Miller, 2004).

5.3.1 SBML Element Extension

In this scheme the links to the actual CV data for the CV terms used in the document are attached to the top level SBML element. These links are contained in a `listOfControlledVocabularies` element. Each link in this list consists of a `ControlledVocabulary` structure each of which has a simple `xlink:href` attribute which contains the URL of controlled vocabulary data and a `xlink:type` attribute which always has the value `simple`. The XLink attributes are defined in (DeRose et al., 2001). It is not essential that for a parsing program load and parse these files for the parser to be able to interpret the use of CV terms. Its also not essential (but its recommended) that the set of files listed contain definitions of all the terms and verbs used in the rest of the SBML document. The UML diagram for the new form of SBML is shown in Figure 3.

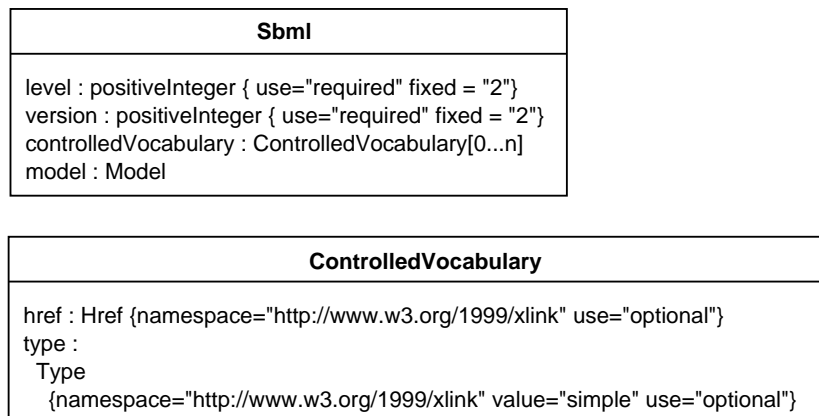


Figure 3: The UML diagram for the new form of sbml

A simple example of the use of the SBML extension would be:

```

<sbml
  level="2" version="2"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns="http://www.sbml.org/sbml/level2">
  
```

```

<listOfControlledVocabularies>
  <controlledVocabularies
    xlink:type="simple"
    xlink:href="http://www.gsk.com/SBML-resources/speciesCV.xml"/>
  </listOfControlledVocabularies>
  ...
</sbml>

```

5.3.2 SBase Extension

SBase is extended to contain a **statements** element which consists of a sequence of statements. Each member of the list makes a 'statement' about the given SBase structure. The SBase structure is always the subject of these statements. For example a SBML **Species** structure may contain 2 statements one stating that the 'type' of the species is 'protein' and the other stating the 'implementation' of the species is 'stochastic'. Each statement is interpreted as a sentence of the form subject-predicate-object where the subject is the SBase object, the predicate is a verb and the object is a CV term. For example a statement can say a given subject 'is a protein' where 'is a' is the predicate and 'protein' is the term.

The **statements** element contains an RDF **RDF** element which contains a sequence of RDF **Description** elements. Each **Description** element contains a SBML namespace **statement** element which in turn contains another nested RDF **Description** element. The inner **Description** element contains SBML namespace **term** and **predicate** elements.

The **term** element should contain a **rdf:resource** attribute which is the URI of a term. The **predicate** element should contain a **rdf:resource** attribute which is the URI of a predicate. Its is important to note that these are *not* URLs and won't typically resolve to web objects.

A SBase structure containing a **statements** element must also contain a **metaId** attribute. The outer **Description** elements must have a **RDF about** attribute which should contain a string of the form '#<X>' where X is the value of the **metaId** field on the SBase. (Although this linkage is redundant it ensures that this scheme complies with the RDF standard.)

The new form of SBase is shown in Figure reffig:sbase.

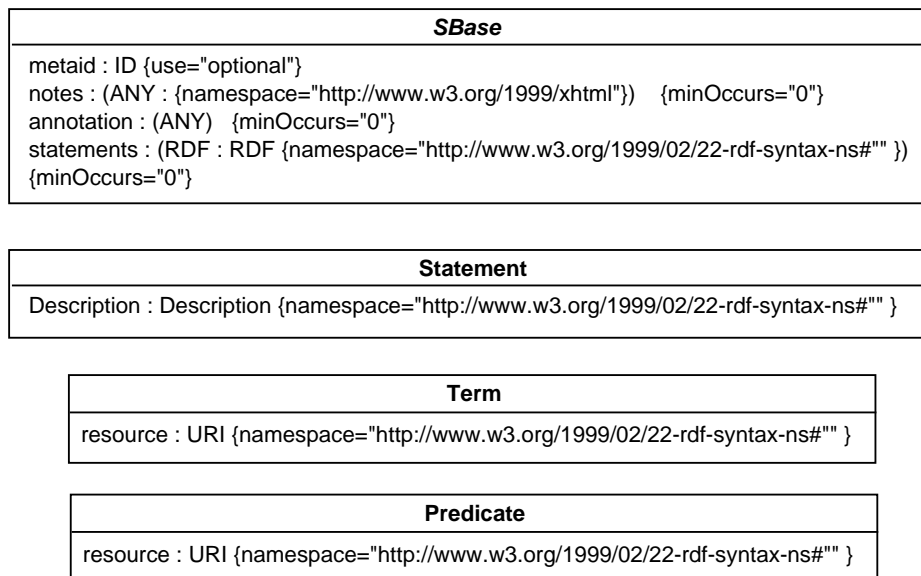


Figure 4: The UML diagram for the new form of SBase and the new classes statement, predicate and term

The following SBML fragment shows an example of the use of the SBase extension to make statements about a species.

```

...
<species metaid="x11" id="S" compartment="C">
  <statements>
    <rdf:RDF xmlns:rdf="http://www.w3c.org/1999/02/22-rdf-syntax-ns#">
      <rdf:Description rdf:about="#x11">
        <statement>
          <rdf:Description>
            <term
rdf:resource="http://biospice.org/BIOSPICE:0003673"/>
            <predicate
rdf:resource="http://biospice.org/BIOSPICE:0005454"/>
            </rdf:Description>
          </statement>
        </rdf:Description>
      <rdf:Description rdf:about="#x11">
        <statement>
          <rdf:Description>
            <term
rdf:resource="http://www.gsk.com/GSK:0000010"/>
            <predicate
rdf:resource="http://www.gsk.com/GSK:0000001"/>
            </rdf:Description>
          </statement>
        </rdf:Description>
      </rdf:RDF>
    </statements>
  </species>
...

```

The definition of the predicate and term from the first description element is given in Section ??.

5.3.3 CV Syntax

A controlled vocabulary is encoded in the GO RDF/XML format (Gene Ontology Consortium, 2004). Both `predicate` and `term` elements can be documented in this format. In the SBML scheme the `go:association` and `go:dbxref` elements and their content can be ignored. The following is an example of a file that defines the CV terms ‘protein’ and ‘polymer’ as well as the predicate ‘is a’. The file defines that a ‘protein’ is a ‘polymer’.

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE go:go>
<go:go
  xmlns:go="http://www.geneontology.org/dtds/go.dtd#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  <rdf:RDF>
    <go:term rdf:about="http://biospice.org/BIOSPICE:0003673">
      <go:accession>BIOSPICE:0003673</go:accession>
      <go:name>Protein</go:name>
      <go:definition>a polymer of amino acids</go:definition>
      <go:isa
        rdf:resource="http://biospice.org/BIOSPICE:0003674" />
      </go:term>
    <go:term rdf:about="http://biospice.org/BIOSPICE:0003674">
      <go:accession>BIOSPICE:0003674</go:accession>
      <go:name>Polymer</go:name>
      <go:definition>
        a compound made up of a linked series of repeated monomers
      </go:definition>
    </go:term>
    <go:term rdf:about="http://biospice.org/BIOSPICE:0005454">
      <go:accession>BIOSPICE:0003674</go:accession>
      <go:name>isa</go:name>
      <go:definition>
        the subject is a subtype of the object
      </go:definition>
    </go:term>
  </rdf:RDF>
</go:go>

```

With this definition of terms and predicates the example in Section 5.3.2 defines that ‘species S is a protein’.

5.4 Discussion

The benefits of the approach described for the implementation of CVs are:

- allows the SBML community to reuse and adapt tools developed to support GO
- allows the SBML to reuse and adapt tools developed to support RDF
- allows the use of the GO ontology
- sufficiently constrained to ensure its possible to straightforwardly implement tools that support the proposal
- additional CV definition formats can be used in the future without changing the SBML format or semantics

Some of issues with this approach are:

- since GO XML/RDF is not directly support by GO tools those tools will require adaption
- perhaps the approach is too complex for simple use where a pure XML approach might be more appropriate
- GO XML/RDF is limited to ‘isa’ and ‘partof’ predicates between CV terms.
- avoiding the use of OWL (McGuinness and van Harmelen, 2004) means that future tools may not be immediately useable

6 Assertions

6.1 Requirements

Some developers have expressed an interest in including within SBML mathematical statements that define assumptions and/or invariants that must remain true for a model to produce correct results. The aim being to define constraints that enable the detection of internal inconsistencies in a model and/or external perturbations of variables and parameters which render a model invalid.

These statements are simply math expressions that are either true or false given some subset of variables and constant defined in the model. Assertions are not structured to facilitate particular types of analysis in fact they are not meant to form the core of any given analysis but instead identify error conditions. Assertions must be clearly separated from other structures that are used directly in analysis.

An example of the use of assertions would be to define quantitatively the assumption in a rate law that the product concentration is much lower than that of the enzyme. If the product concentration becomes large enough to render the rate law invalid during a simulation the simulator can notify the user. To facilitate this type of use of assertions a error string should be optional associated with the assertion.

6.2 Implementation

A new subtype of `Rule`, `Assertion` is introduced. In addition to the inherited `math` field `Assertion` has an optional list of `Message` structures where each message is in a different language. These messages are to be displayed to the user when the expression in the `math` field becomes false.

The `Message` structure has content of type `String` and an attribute `lang` with type `String`. The content of the `lang` attribute should follow the IETF standard for language labels (Alvestrand, 2001). The UML diagram for the `Assertion` and `Message` classes are shown in Figure 5.

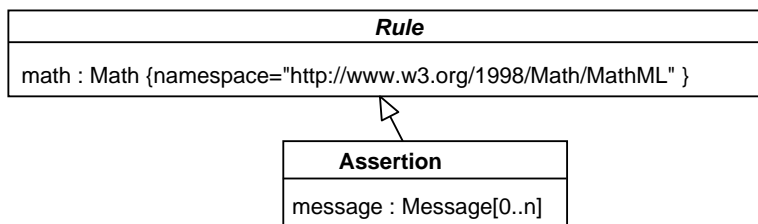


Figure 5: The UML diagram for the new form of Assertion and Message and the new classes statement, predicate and term

6.3 Example

The following hypothetical example shows a simple SBML fragment where the assumptions of a rate law are made explicit. In this example the rate law

$$\frac{V_{max} [S]}{K_m + [S]}$$

for reaction



operates under the assumption

$$[P] < 1$$

```

...
<listOfReactions>
  <reaction id="catalysis">
    <listOfReactants>
      <speciesReference species="S"/>
    </listOfReactants>
    <listOfProducts>
      <speciesReference species="P"/>
    </listOfProducts>
    <kineticLaw>
      <math xmlns="http://www.w3.org/1998/MathMathML">
        <apply>
          <divide/>
            <apply>
              <times/>
                <ci> V_max </ci>
                <ci> S </ci>
              </apply>
            <apply>
              <plus/>
                <ci> K_m </ci>
                <ci> S </ci>
              </apply>
            </apply>
          </math>
        <listOfParameters>
          <parameter id="K_m"/>
          <parameter id="V_max"/>
        </listOfParameters>
      </kineticLaw>
    </reaction>
  ...
</listOfReactions>
...
<listOfRules>
  <assertion>
    <math xmlns="http://www.w3.org/1998/MathMathML">
      <apply>

```

```

        <lt/>
        <ci> P </ci>
        <cn> 1 </cn>
    </apply>
</math>
<listOfMessages>
    <message lang="en">
        The product concentration, P, is too high for
        the model assumptions to hold.
    </message>
</listOfMessages>
</assertion>
...
</listOfRules>
...

```

7 Variable and Flux Bounds

7.1 Requirements

It is clear that there are non-spatial analyses of biochemical reaction networks which can be driven by mathematical structures that are not yet part of the SBML standard. These analyses can use the reaction network contained in a SBML model as a starting point yet they require additional information as input. Some analyses of biochemical reaction networks operate on bounds of various properties of a network rather than absolute values or expressions. Bounds need to be placed on initial values, variables, parameters and reaction rates.

7.2 Implementation

7.2.1 Bound structure

Under this proposal an empty abstract base class for rules, `RuleBase`, is introduced and the list of `Rule` structures on `Model` is modified to contain an arbitrary collection of `RuleBase` structures. A new abstract `Bound` class is then derived from `RuleBase` as is the existing class `Rule`.

The `Bound` has the following fields:

- `symbol` a SID field which contains a reference to the variable or initial condition that is constrained by the `Bound` structure.
- `lower` and `upper` double fields which define the exclusive lower and upper bounds on the value of the symbol referenced by the `symbol` field. These bounds values have the same units as the symbols when the symbol occurs in a MathML expression anywhere else in the model.

There are two concrete subclasses of `Bound`:

- `VariableBound` where the structure constrains the given symbol at all times and
- `InitialBound` where the structure only constrains the given symbol at $t = 0$

The UML diagram for the `Bound`, `VariableBound` and `InitialBound` classes are shown in Figure 6.

7.2.2 Flux symbol

To enable the use of these structures for flux balance analysis a symbol to represent a reaction flux is required. Under this proposal the flux of a reaction is represented by a symbol which refers to the reaction. This symbol can be used in any MathML expression to represent the reaction flux however it cannot be determined a `rule` structure. An attribvariable field of an `AssignmentRule` or `RateRule` cannot contain a reaction flux symbol.

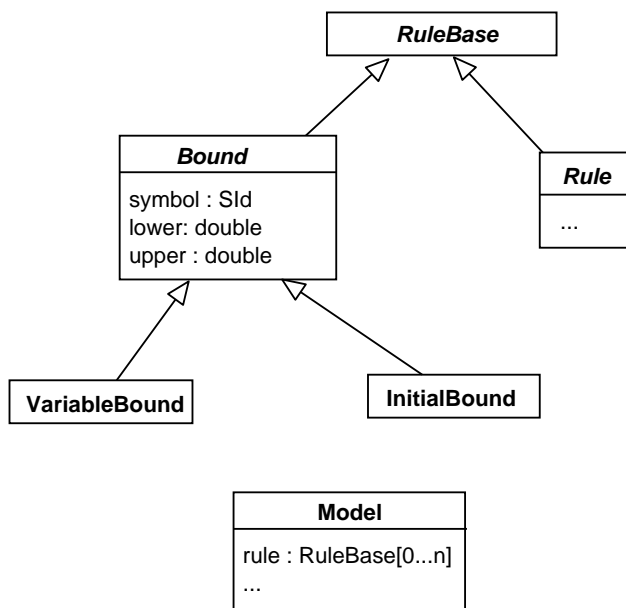


Figure 6: The UML diagram for the new form of Assertion and Message and the new classes statement, predicate and term

7.2.3 Example

example goes here

8 Dependent Variable Attribute on AlgebraicRule

It can be important for reproducing a given numerical solution to specify explicitly the dependent variable on an algebraic rule. Given more than one algebraic rule it is often ambiguous which variable is the dependent variable of a rule. A parsing program can decide which variable is the dependent variable of each rule however different decisions can unfortunately give different results. Therefore to ensure precise reproduction of results we propose the introduction of an attribute that makes the dependent variable of an algebraic rule explicit.

It is proposed that a new optional `Sid` field `dependentVariable` is added to the `AlgebraicRule` which makes explicit the variable whose value should be calculated using the rule. The field will contain a variable symbol defined by a `Species`, `Compartment` or `Parameter` structure. The field cannot refer to a constant symbol. The use of this field should be avoided in best practice.

9 Nested Unit Definitions

In SBML it is not possible to create nested unit definitions: all unit definitions have to be composed from fundamental units. To overcome this limitation it is proposed that the `kind` field of `Unit` structures be able to refer to unit definitions as well as the `unitkind` enumeration.

References

- Alvstrand, H. T. (2001). RFC 3066 - tags for the identification of languages. Available via the World Wide Web at <http://www.isi.edu/in-notes/rfc3066.txt>.
- DeRose, S., Maler, E., and Orchard, D. (2001). XML Linking Language (XLink) Version 1.0 W3C Recommendation. Available via the World Wide Web at <http://www.w3.org/TR/xlink/>.

- Finney, A. (2004). Systems Biology Markup Language (SBML) Level 3 Proposal: Multi-component Species Features. Available via the World Wide Web at <http://www.cds.caltech.edu/~afinney/multi-component-species.pdf>.
- Gauges, R., Rost, U., Sahle, S., and Wegner, K. (2004). Including layout information in SBML files version 2.1. Available via the World Wide Web at <http://projects.embl.org/bcb/sbml/level1/level2/20040630/SBMLLayoutExtension-20040630.pdf>.
- Gene Ontology Consortium (2004). GO file format guide - XML. Available via the World Wide Web at <http://www.geneontology.org/GO.format.html#XML>.
- Lassila, O. and Swick, R. (1999). Resource description framework (RDF) model and syntax specification. Available via the World Wide Web at <http://www.w3.org/TR/REC-rdf-syntax/>.
- Manola, F. and Miller, E. (2004). Rdf primer. Available via the World Wide Web at <http://www.w3.org/TR/rdf-primer/>.
- McGuinness, D. L. and van Harmelen, F. (2004). OWL web ontology language overview. Available via the World Wide Web at <http://www.w3.org/TR/owl-features/>.
- Sachs, S. (2003). Expressing qualitative signal transduction pathways in SBML. Available via the World Wide Web at <http://sbml.org/workshops/eighth/stke.pdf>.