Systems Biology Markup Language (SBML) Level 2 Proposal: Miscellaneous Features

Andrew Finney, Victoria Gor, Eric Mjolsness, Hamid Bolouri

afinney@cds.caltech.edu, gor@aig.jpl.nasa.gov,
Eric.D.Mjolsness@jpl.nasa.gov, hbolouri@cds.caltech.edu

April 19, 2002

Contents

1 Introduction

This document describes proposed features for inclusion in Systems Biology Markup Language (SBML) Level 2. These features do not address specific requirements that will enable SBML Level 2 to represent biochemical structures and processes but instead could potentially form a useful foundation for the features proposed elsewhere for inclusion in SBML Level 2.

This document is not a definition of SBML Level 2 or part of it. This document simply presents various features which could be incorporated into SBML Level 2 as the Systems Biology community wishes. This document is intended for detailed review by that community and to provoke alternative proposals. Throughout this document issues that the authors believe will require further discussion have been highlighted.

The features proposed here are:

- **Packages** : a mechanism that enables a model to document what SBML Level 2 features are used within the model. see section 3.
- **Conditional Expressions** : conditional operators in formulas. see section 4.
- **Functions** : a mechanism for defining new functions for use in formulas. see section 5.

Section 7 includes a complete model which uses all the features described in this proposal.

For brevity the text of this document is with reference to SBML Level 1 (Hucka et al., 2001) i.e. features are described in terms of changes to SBML Level 1. This document uses UML diagrams in the same way except that new features are shown in red.

All types proposed in this document will be derived from the `SBase` type.

2 Minimal Header for SBML Level 2 documents

We suggest that the minimal header, that SBML level 2 documents will have, is:

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level2" version="1" level="2">
```

3 Packages

It is unlikely that systems parsing SBML Level 2 will have functional internal representations of all the features proposed for Level 2, for example a given simulator might have a representation of modularity but not of geometric structures. To enable a parsing system to check whether a given model is within its scope it's proposed that an SBML document has an optional list of `Package` structures that define the features that are used by the document. The proposed new UML definition of the SBML type is shown in figure 1.

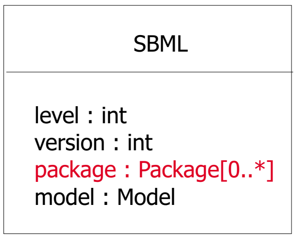

Figure 1: *The definition of the SBML type*

In this proposal an `SBML` structure can optionally contain a list of `Features`. The proposed UML definition of the `Feature` type is shown in figure 2.

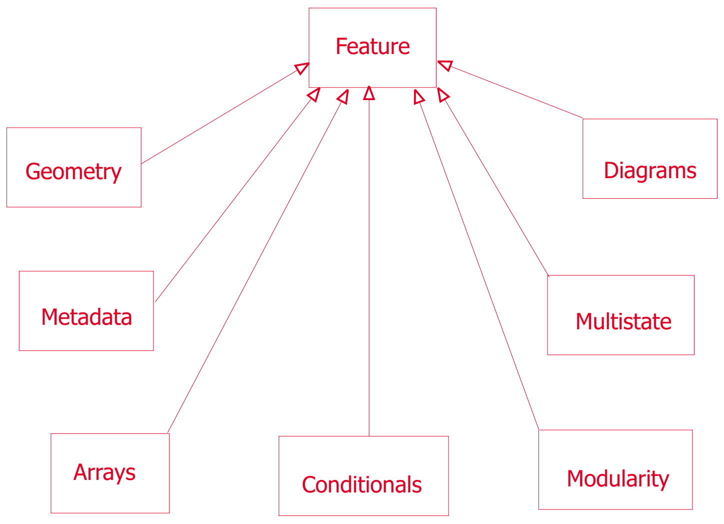

Figure 2: *The definition of the Feature type*

The set of `Feature` subtypes is very tentative. It is proposed that for each of the `Feature` subtypes there will be defined a set of SBML elements, numeric expression functions and numeric expression operators. (These sets will be defined later). Each of these sets constitutes a *package*. A *package* is used if the document contains any of the elements or operators in the *package*.

The following proposed list describes the expected scope of each of the *packages* corresponding to each `Package` types:

- **Geometry**: features needed to support 2D and 3D spatial models
- **Diagrams**: features to store diagrams of models
- **Metadata**: features to annotate models, to facilitate (amongst other things), the linking of model entities to data external to the models and the systematic storage and retrieval of models.
- **Arrays**: features enabling the inclusion of arrays of structures/entities in models. These features would allow a model to be assembled from many copies of identical parts. These features enable the representation of patterns of connection amongst elements of arrays.
- **Multistate**: features enabling the representation of biochemical entities with a hierarchy of possible states and applying reactions to those entities with a defined range of states. This group is also exploring possible schemes for representing entities, or complexes, which are graphs of component entities.
- **Modularity**: features enabling the construction of models from a hierarchy of submodels. This will allow a given reaction or process to be modeled at multiple levels of detail. These features will allow the construction of models from reusable components.
- **Conditionals**: incorporates conditional operators into numeric expressions.

In this proposal a valid level 2 document would use a subset of the *packages* that correspond to the `Package` elements in the document. The `Package` lists should not contain more than one element of a each `Package` type. An empty `Package` list corresponds to the list containing all possible `Package` types i.e. the document can contain any Level 2 operators or elements. Ideally a document should contain the smallest non empty valid list of `Package` structures.

The above list assumes that all systems will be able to support the functions feature described in section 5.

It is expected that the `modularity` *package* will allow more than one `Model` structure to be contained in an SBML Level 2 document. The `Package` list is a attached to the `SBML` structure to ensure that information on the features used isn't distributed through the document.

3.1 Example

The following example of an SBML document header, complying with this proposal, indicates that the document uses the `arrays` and `conditionals` features.

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level2" version="1" level="2">
    <listOfFeatures>
        <arrays/>
        <conditionals/>
    </listOfFeatures>
    <model name="cell">
    ...
    </model>
</sbml>
```

3.2 Issues

Some issues that need to be addressed:

- this concept might be better addressed using separate namespaces for the different *packages* however this won't cover the use of operators or functions. The idea behind this concept is to enable an event driven parse to halt parsing as soon as the `sbml` open tag has been parsed. As different namespaces can be added on an ad hoc basis throughout a document, an event driven parser will have to load the entire document to find all the namespaces and thus features used in the document.
- the classification of features will require review at the end of the development of SBML Level 2.
- should the `Conditionals` type be split into a `DynamicConditionals` type and a `StaticConditionals` type? The distinction being between those conditional expressions where the condition expression could potentially change during any simulation of the model (`DynamicConditionals`) and those that do not (`StaticConditionals`). This information may be an important criteria by which a simulator will judge whether it can simulate a model.
- `DynamicConditionals` can't be mapped to SBML Level 1.

4 Conditional expressions

This section proposes constructs, for describing discontinuities via numeric expressions or formula strings.

4.1 Operators

Table 1 shows a possible extended set of operators that could be available in SBML Level 2. New operators are shown in red. As in SBML Level 1 these operators work as defined in C except that the types are always `double`. This means that 0 is interpreted as false and all non zero numbers are interpreted as true. The operators `&&` and `||` short circuit the evaluation of their second operand depending on the value of the first operand.

4.2 `switch` function

In addition to the new operators described above it is proposed that an additional 'built-in' function `switch`, is introduced. This has the form:

```
switch(key, match1, result1, ... matchn, resultn, defaultResult);
```

Tokens	Operation	Class	Precedence	Associates
name	symbol reference	operand	10	n/a
(*expression*)	expression grouping	operand	10	n/a
$f(\ldots)$	function call	prefix	10	left
!	logical not	unary	9	right
$-$	negation	unary	9	right
ˆ	power	binary	8	left
$*$	multiplication	binary	7	left
/	division	binary	7	left
$+$	addition	binary	6	left
$-$	subtraction	binary	6	left
`<`	less than	binary	5	left
`>`	greater than	binary	5	left
`>=`	greater than or equal	binary	5	left
`<=`	less than or equal	binary	5	left
`==`	equality	binary	4	left
`!=`	inequality	binary	4	left
`&&`	logical and	binary	3	left
`||`	logical or	binary	2	left
`? :`	conditional	ternary	1	right

Table 1: *A table of the expression operators available in SBML, operators proposed in this document are shown in red. In the* **Class** *column, "operand" implies the construct is an operand, "prefix" implies the operation is applied to the following arguments, "unary" implies there is one argument, and "binary" implies there are two arguments. The values in the* **Precedence** *column show how the order of different types of operation are determined. For example, the expression* $a * b + c$ *is evaluated as* $(a * b) + c$ *because the* $*$ *operator has higher precedence. The* **Associates** *column shows how the order of similar precedence operations is determined; for example,* $a - b + c$ *is evaluated as* $(a - b) + c$ *because the* $+$ *and* $-$ *operators are left-associative. The precedence and associativity rules are taken from the C programming language (Harbison and Steele, 1995), except for the symbol ˆ, which is used in C for a different purpose.*

This function compares `key` to the values `match1...matchn` and returns the `result` value immediately following the matching `match` argument. If none of the `match` arguments are equal to `key` then `switch` returns the `defaultResult` argument.

The call

```
switch(k, m, r, d)
```

is equivalent to

```
(k == m) ? r : d
```

The call

```
switch(k, m1, r1, m2, r2, d)
```

is equivalent to

```
(k == m1) ? r1 : ((k == m2) ? r2 : d)
```

and so on.

4.3 Issues

Given that the function `switch` and the operators `<=`, `>=` and `!=` are redundant should they be incorporated into SBML Level 2?

5 Functions

So as to avoid the continuous revision of SBML, as additional built-in functions are required, a mechanism is required to allow the arbitrary definition of new functions in SBML Level 2. This section proposes such a mechanism.

The proposed UML definition of the `Model` type is shown in figure 3. (In SBML Level 2 the model type will probably contain additional optional substructures which are not shown). A `Model` can optionally contain a list of `Functions`. Figure 4 shows the proposed form of the `Function` type in UML form.

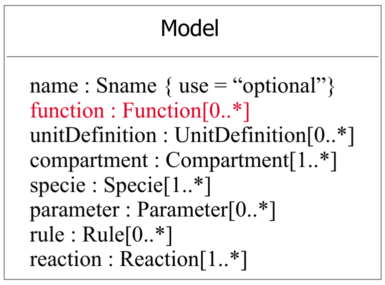
Model
name : Sname { use = “optional”}
function : Function[0..*]
unitDefinition : UnitDefinition[0..*]
compartment : Compartment[1..*]
specie : Specie[1..*]
parameter : Parameter[0..*]
rule : Rule[0..*]
reaction : Reaction[1..*]

Figure 3: *The definition of the Model type*

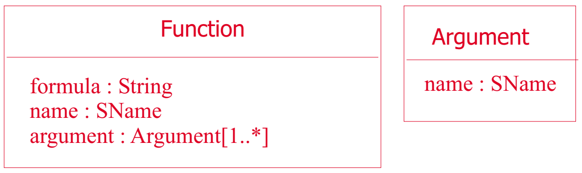

Figure 4: *The definition of the Function type*

A proposed `Function` structure consists of the function’s name (of `SName` type), a formula string and a list of `Argument` structures. An `Argument` structure simply consists of the argument’s name.

A proposed `Function` structure defines a new function that can be used in any formula that follows the `Function` structure. The only symbols that can be used in the formula string of a `Function` structure are the following:

- **Arguments:** any of the names given in the `Argument` list.
- **Previously defined functions:** any of the names given in preceding `Function` structures.
- **Built-in functions:** any of the names of built-in functions.

These restrictions mean that functions can’t be recursive or mutually recursive thus ensuring that function ‘calls’ can be expanded in place i.e. models using functions can be mapped to SBML Level 1.

A `Function` formula simply returns a `double` value. All the arguments to a function are of type `double`. These functions share the same namespace as other model-level components, this means that, for example a function can’t have the same name as a specie or a reserved name.

5.1 Example

The following is a simple example of a `Function` structure in a `Model` structure:

```
<model name="cell">
    <listOfFunctions>
        <function name="pow3" formula="x^3">
            <listOfArguments>
                <argument name="x"/>
            </listOfArguments>
        </function>
    </listOfFunctions>
    ...
</model>
```

5.2 Issues

- Is the scope of symbols in formulas in functions too restrictive?
- Do we need to be able to handle objects other than `double` values?

6 Models with no reactions

It is proposed that in SBML Level 2 a model can contain no reactions. A `specieConcentrationRule` structures can be defined to have the same effect as reactions.

7 Complete Example

This section contains a complete SBML document which includes all the proposed features described here. This model is a variation on the example model described in Hucka et al. (2001).

Consider the following hypothetical branched system:

$$\begin{array}{ccc} X_0 & \xrightarrow{k_1 pow3(X_0)} & S_1 \\ S_1 & \xrightarrow{\text{if } S_1 < 1 \text{ then } k_a S_1 \text{ else } k_b S_2} & X_1 \\ S_1 & \xrightarrow{k_3 S_1} & X_2 \end{array}$$

The following is a XML document that encodes the model shown above:

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level2" version="1" level="2">
    <listOfFeatures>
        <conditionals/>
    </listOfFeatures>
    <model name="Branch">
        <notes>
            <body xmlns="http://www.w3.org/1999/xhtml">
                <p>Simple branch system.</p>
                <p>The reaction looks like this:</p>
                <p>reaction-1:   X0 -> S1; k1* pow3(X0);</p>
                <p>reaction-2:   S1 -> X1; S1 < 1 ? k2a*S1 : k2b*S2 ;</p>
                <p>reaction-3:   S1 -> X2; k3*S1;</p>
                <p>where pow3(x) is x^3
            </body>
        </notes>
        <listOfFunctions>
            <function name="pow3" formula="x^3">
                <listOfArguments>
                    <argument name="x"/>
                </listOfArguments>
            </function>
        </listOfFunctions>
        <listOfCompartments>
            <compartment name="compartmentOne" volume="1"/>
        </listOfCompartments>
        <listOfSpecies>
            <specie name="S1" initialAmount="0" compartment="compartmentOne"
                    boundaryCondition="false"/>
            <specie name="X0" initialAmount="0" compartment="compartmentOne"
                    boundaryCondition="true"/>
            <specie name="X1" initialAmount="0" compartment="compartmentOne"
                    boundaryCondition="true"/>
            <specie name="X2" initialAmount="0" compartment="compartmentOne"
                    boundaryCondition="true"/>
        </listOfSpecies>
        <listOfReactions>
            <reaction name="reaction_1" reversible="false">
                <listOfReactants>
                    <specieReference specie="X0" stoichiometry="1"/>
                </listOfReactants>
                <listOfProducts>
                    <specieReference specie="S1" stoichiometry="1"/>
                </listOfProducts>
                <kineticLaw formula="k1 * pow3(X0)">
                    <listOfParameters>
                        <parameter name="k1" value="0"/>
                    </listOfParameters>
                </kineticLaw>
            </reaction>
            <reaction name="reaction_2" reversible="false">
                <listOfReactants>
                    <specieReference specie="S1" stoichiometry="1"/>
                </listOfReactants>
                <listOfProducts>
                    <specieReference specie="X1" stoichiometry="1"/>
                </listOfProducts>
                <kineticLaw formula="S1 < 1 ? ka*S1 : kb*S2">
                    <listOfParameters>
                        <parameter name="ka" value="0"/>
                        <parameter name="kb" value="1"/>
                    </listOfParameters>
                </kineticLaw>
            </reaction>
            <reaction name="reaction_3" reversible="false">
                <listOfReactants>
                    <specieReference specie="S1" stoichiometry="1"/>
                </listOfReactants>
                <listOfProducts>
                    <specieReference specie="X2" stoichiometry="1"/>
                </listOfProducts>
                <kineticLaw formula="k3 * S1">
                    <listOfParameters>
                        <parameter name="k3" value="0"/>
                    </listOfParameters>
                </kineticLaw>
            </reaction>
        </listOfReactions>
    </model>
</sbml>
```

References

Harbison, S. P. and Steele, G. L. (1995). *C: A Reference Manual.* Prentice-Hall.

Hucka, M., Finney, A., Sauro, H. M., and Bolouri, H. (2001). Systems Biology Markup Language (SBML) Level 1: Structures and facilities for basic model definitions. Available via the World Wide Web at `http://www.cds.caltech.edu/erato`.