An XML-Based Model Description Language for Systems Biology Simulations

Andrew Finney, Herbert Sauro, Michael Hucka, Hamid Bolouri
{afinney,hsauro,mhucka,hbolouri}@cds.caltech.edu
ERATO Kitano Systems Biology Project
Control and Dynamical Systems 107-81
California Institute of Technology, Pasadena, CA 91125

This document is based on discussions with the authors of
BioSpice (Arkin), DBSolve (Goryanin), E-Cell (Tomita, Nakayama, Takahashi), Gepasi (Mendez),
Jarnac (Sauro), StochSim (Bray, Firth & Shimizu), Virtual Cell (Loew & Schaff),
and the ERATO Kitano Systems Biology Project group.

Version of September 12, 2000

Contents

1 Introduction

We present a first attempt at specifying a common, model-based description language for systems biology simulation software. We call this this the Systems Biology Markup Language (SBML). The overall goal is to develop an **open standard** that will enable simulation software to communicate and exchange models, ultimately leading to the ability for researchers to run simulations and analyses across multiple software packages.

SBML is the result of merging the most obvious modeling-language features of BioSpice, DBSolve, E-Cell, Gepasi, Jarnac, StochSim, and Virtual Cell. The description language is encoded in XML, the Extensible Markup Language (Bosak and Bray, 1999; Bray, Paoli and Sperberg-McQueen, 1998). The XML encoding of the description language can define a file format; however, at this time, we are focusing on using the XML-based description language as an interchange format for use in communications between programs.

The primary purpose of this document is to serve as a basis for discussion and further development of a more comprehensive language specification. The final outcome of this process will be an XML Schema which can be used to communicate model descriptions between simulation packages. Appendix B contains the current version of this schema. As XML Schemas are difficult to read and absorb by human readers, we define the proposed data structures using a succinct graphical notation based on a subset of UML, the Unified Modeling Language (Eriksson, 1998; Oestereich, 1999). Our notation is explained in *A Notation for Describing Data Representations Intended for XML Encoding* (Hucka, 2000), available online at `ftp://ftp.cds.caltech.edu/pub/caltech-erato/notation/`. For the sake of clarity, we ask readers to use this notation when contributing to discussions about the specification. To facilitate discussions, a web/FTP site and a group mailing list have been set up for the participating groups. Please see the web site at `http://www.cds.caltech.edu/erato/` for details.

A few assumptions about the form of the language are not necessarily self-evident. In particular,

- We assume that, in order for each simulator program to read and write the model description language, the program will require native interface code or a stream/file converter. This interface code will have to translate the simulator's internal data structures to and from the model description language, as well as smooth any out small differences in conceptual organization between the simulator's specific internal representation and the model description language.
- A zero value is not the same as an empty attribute value. The model description language uses a number of structures that, in a given object instance, may be empty, depending on whether a given simulator needs them. Programs that interpret objects expressed in the language must be designed to pay attention to this distinction.

The XML definition of SBML follows a certain naming convention (Hucka, 2000): the names of object attributes begin with a lowercase letter and the names of object classes or types begin with an uppercase letter. In this document, we also follow the convention of writing keywords (names of types, attributes, elements, etc.) in a typewriter-style font; for example, `Compartment` is a class name and `compartment` is an element name.

Appendix A contains several examples of models encoded in XML using SBML. We use portions of these models as illustrations throughout the rest of this document.

2 Overview

The representation language is organized around five categories of information: model, compartment, geometry, specie and reaction. Not all of these will be needed by every simulation package; rather, the intent is to cover the range of data structures needed by the collection of all of the simulators examined so far.

Figure 1 depicts the highest level of organization of the objects in SBML. It shows that all classes are derived either directly or indirectly from the class `Identified`. `Identified` contains attributes `id` and `simpleNotes` and elements `notes` and `annotation`. The attribute `id` has type `ID`, so that `Identified`

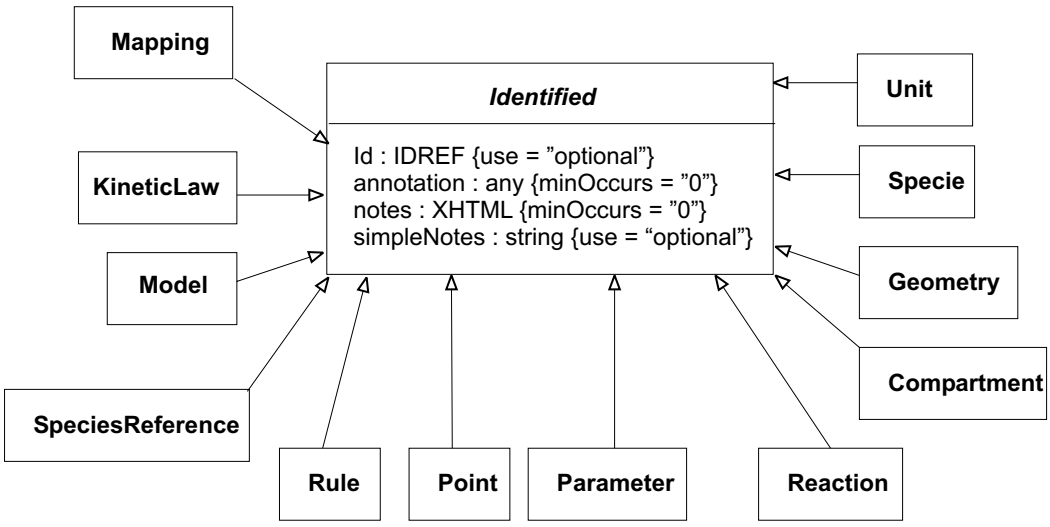

Figure 1: *A diagram of the highest level of the object class hierarchy. (The notation used in the figures in this document is summarized in Appendix E).*

objects can be referred to using the XML `ID/IDREF` mechanism (Biron and Malhotra, 2000). The element `notes` can contain XHTML content. The `simpleNotes` is of type `string`. Both `notes` and `simpleNotes` are intended to store information meant to be presented to humans. Finally, the element `annotation` can contain arbitrary elements and is intended to store information not necessarily intended for human viewing. All SBML classes are derived from `Identified`.

In SBML, several classes have attributes of the type `Name`. The `Name` type is a XML simple type derived from the XML `string` type, and is defined to have following pattern (expressed here in conventional Backus-Naur Form [BNF]):

```
Letter ::= 'a'..'z','A'..'Z'
Digit ::= '0'..'9'
Name ::= Letter | '_' { Letter | '_' | Digit}
```

In the following diagrams, we follow the UML convention of omitting the display of the attributes on subclasses derived from `Identified`, although it should be understood that these attributes are always available.

3 Elements of the Systems Biology Markup Language (SBML)

In the sections that follow, we discuss each of the major classes that are derived from `Named` or `Anonymous` in the representational hierarchy.

3.1 Model

The `Model` class defines a grouping of components; that is, a model does not necessarily represent a single specific biological entity. There is only one element of type `Model` per instance of an SBML document or data stream, and it contains the list of compartments, species and reactions that define a given model. The UML-based definition of the class is shown in Figure 2.

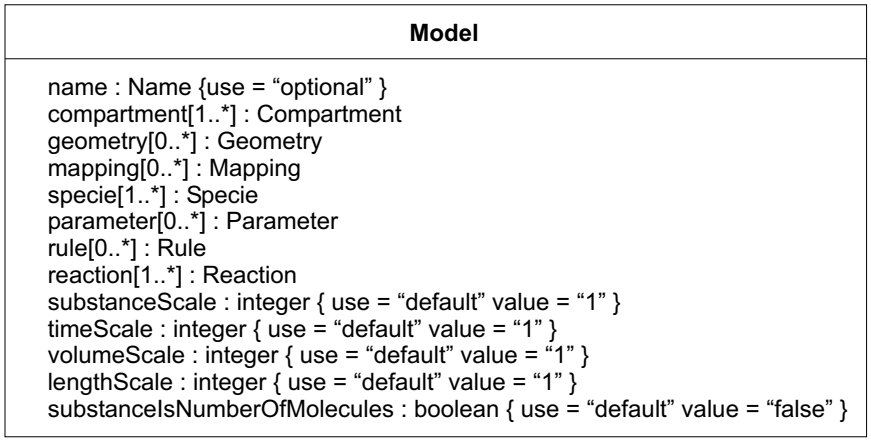

Figure 2: *A diagram of* `Model`*.*

The definition shows that there must be at least one specie, one reaction and one compartment in a model; otherwise, there would be little point in defining the model in the first place. There is no restriction on the total number of these elements; Appendix A.1 gives an example of a minimal model. In addition, a model can consist of zero or more `Geometry`, `Mapping`, `Parameter` and `Rule` elements. `Model` has an optional `name` attribute of type `Name`.

The following is a skeletal example:

```
<sbml version="1">
    <model>
        <listOfCompartments>
        </listOfCompartments>

        <listOfSpecies>
        </listOfSpecies>

        <listOfReactions>
        </listOfReactions>
    </model>
</sbml>
```

3.1.1 Units

`Model` and other classes have attributes for defining the units used. An example that uses this feature is given in Appendix A.3.

SMBL has several predefined quantity types: time, amount of substance, volume, charge and length. Apart from amount of substance and charge, these are all restricted to metric units. In all cases, apart from charge, the scale of the given value can be modified using a power-of-ten multiplier which is an optional integer attribute. Amount of substance can be expressed in number of molecules, depending on the state of the `substanceIsNumberOfMolecules` boolean attribute. Table 1 lists the built-in quantity types and their associated scale attributes.

All the scale attributes on `KineticLaw`, `Specie` and `Geometry` override those on `Model`. In various sections below, these units are combined to define the units involved in formulae.

3.2 Compartments

A `Compartment` represents a bounded container in which species are located. A diagram of the definition of `Compartment` is shown in Figure 3. A `Compartment` object has a `name` attribute of type `Name`.

Type	Attribute Name	Metric Unit	Default Scale	Applicable Elements
substance	`substanceScale`	Moles or no. of molecules	1	`Model`, `Specie`, `KineticLaw`
time	`timeScale`	Seconds	1	`Model`, `KineticLaw`
volume	`volumeScale`	litres	1	`Model`
length	`lengthScale`	metres	10^{-6} (μ)	`Model`, `Geometry`
charge		no. of electrons	1	

Table 1: *A table of the built-in quantities in SBML.*

The geometric characteristics of a compartment are described using a `Geometry` element and a `Mapping` element. The geometry information is effectively optional, because a `Mapping` element linking a `Compartment` element and a `Geometry` element may be absent. In such cases, the compartment simply serves as a topological structure.

For convenience, a compartment has the floating-point attribute `volume`, representing the total volume of the compartment in the units of volume defined on the `Model`. This enables concentrations of species to be calculated in the absence of geometry specifications. Any `Geometry` element associated with the compartment then overrides this value. The `volume` attribute is optional and defaults to a value of 1.

The following is an example of a `Compartment` element:

```
<compartment name="cell" volume="1"/>
```

Lists of compartments are itemized within the element `listOfCompartments` in a `Model` object, as in the following example:

```
<listOfCompartments>
  <compartment name="cytosol" volume="1"/>
  <compartment name="mitochondria" volume="0.3"/>
</listOfCompartments>
```

3.3 Geometry

As shown in the definition of `Model` in Figure 2, a `Model` object can contain zero or more `Geometry` elements. The `Geometry` class is intended to provide a means for specifying morphological characteristics of compartments in simulations. It is defined in Figure 4. `Geometry` has a `name` attribute of type `Name`.

The dimensionality of a compartment can be one, two or three dimensions, and a given compartment can have an overall physical size. These aspects are determined in an instance of a `Geometry` element by which of the three size attributes are given a value: either `length` (for 1-D), `surfaceArea` (for 2-D), or `volume` (for 3-D). Some examples of different dimensionalities are: DNA stretch (1-D); disks in photoreceptors (2-D); patch of membrane (2-D); cell nucleus (3-D); and dendritic spines (3-D). The size of a compartment may change over the course of a simulation. The 2-D geometrical information is a boundary specification that can take the form of a sequence of connected (x, y) coordinates (`Point` elements) in a global reference frame. (Each point is connected by a straight line to the next point in the sequence. The last point in the sequence is connected to the first. This set of lines forms the 2D boundary.)

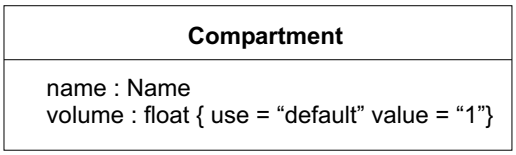

Figure 3: *The `Compartment` object class definition.*

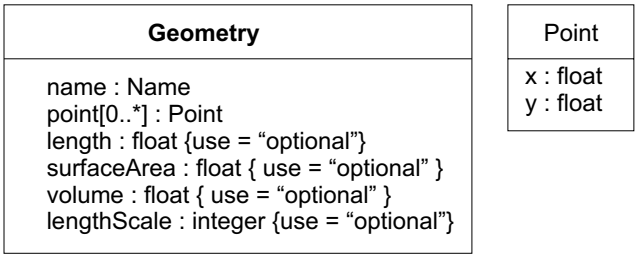

Figure 4: *The* `Geometry` *object class definition.*

All the values on a `Geometry` element and its associated elements are in length units as defined on the `Geometry` element by the `lengthScale` attribute. If these units are not defined on the `Geometry` element, the definition on the enclosing `Model` is used instead.

3.4 Mapping

A `Model` element can contain zero or more `Mapping` elements. A `Mapping` element maps a `Compartment` element to a `Geometry` element. Thus, `Mapping` has two name attributes, `compartment` and `geometry`. `Mapping` is defined in Figure 5.

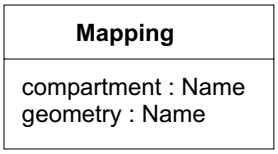

Figure 5: *The* `Mapping` *object class definition.*

3.5 Species

Species comprise all entities that take part in reactions. The `Specie` class is intended to represent these entities. These include simple ions (e.g., protons, calcium), simple molecules (e.g., glucose, ATP), and large molecules (e.g., RNA, polysaccharides, and proteins). Figure 6 presents the definition of `Specie`. `Species` has a `name` of type `Name`.

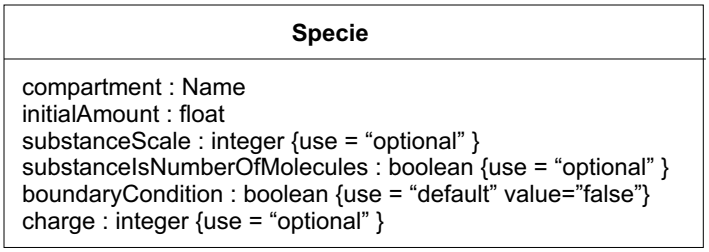

Figure 6: *The* `Specie` *object class definition.*

The attribute `initialAmount`, of type float, is used to set the initial amount of the specie. These are in the substance units as defined on `Specie` via the `substanceScale` and `substanceIsNumberOfMolecules` attributes in the way described in the definition of `Model` above. If these units are not defined on a `Specie` object, the definition on the `Model` element is used instead.

The boolean attribute `boundaryCondition` determines whether the amount of the specie is fixed or variable over the course of a simulation. `boundaryCondition` is optional and defaults to a value of false. The attribute

compartment, of type Name, is used to identify the compartment in which the specie belongs. The integer attribute charge indicates the charge on the species (in terms of electrons, not the SI unit Coulombs).

The following is an example of a minimal specie element inside a Model:

```
<specie name="s1" compartment="cell" initialAmount="4"/>
```

Lists of species are itemized within listOfSpecies inside a Model object, as in the following example:

```
<listOfSpecies>
  <specie name="Glucose" compartment="cell" initialAmount="4"/>
  <specie name="Glucose_6_P" compartment="cell" initialAmount="0.75"/>
  ...
</listOfSpecies>
```

3.6 Parameters

A Parameter element associates a symbol with a float value so that the symbol can be used in formulae in place of the value. Figure 7 gives the definition of this class. Parameter has a name attribute of type Name.

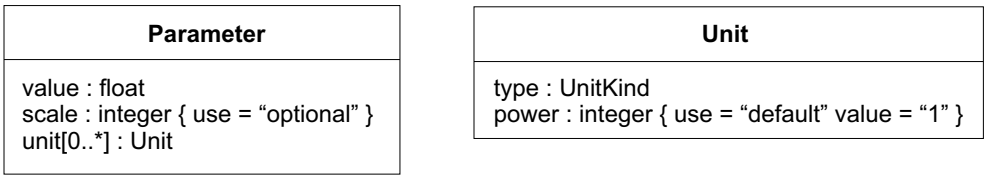

Figure 7: *The* Parameter *class definition.*

The symbol is set by the name attribute and the value is taken from the value attribute of the Parameter object. A parameter can be associated with either a model or a reaction. The parameter elements associated with the model define parameters that are *global* to the whole model; those that are associated with a reaction *overload* the global parameters. (See Section 3.9 for further details.)

The Parameter class has an optional integer attribute called scale. As for the built-in types, the scale attribute is a power of ten multiplier.

The unit of the parameter value is specified by the set of optional Unit elements contained within the parameter element. Each unit element has a attribute of type UnitKind. UnitKind is an enumeration type consisting of the following values: “mole”, “litre”, “second”, “metre”, “gram”, “ampere”, “kelvin”, “centigrade”, “candela”, “radian”, “streadian”, “hertz”, “newton”, “joule”, “calorie”, “watt”, “coulomb”, “volt”, “farad”, “ohm”, “weber”, “tesla”, “henry”, “lumen”, “lux”, “pascal”, “siemens”, “becquerel”, “gray”. The attribute type can only have one of the values listed. Finally, the Unit class also has an integer attribute, power, that represents an exponent modifier to the type value; its default value is 1.

The following is an example of a simple unit element:

```
<unit type="gram" power="3"/>
```

The following is a more complex example of a single Parameter element, showing how a symbol, Vm, is defined to be 3 $mM\ l^{-1}\ s^{-1}$:

```
<parameter name="Vm" value="3" scale="-3">
    <unit type="mole"/>
    <unit type="litre" power="-1"/>
    <unit type="second" power="-1"/>
</parameter>
```

Complex units can be created by placing several unit elements inside a Parameter class object. The use of derived units can help reduce the number of elements required.

Lists of parameters are itemized on the `listOfParameters` element within a `Model`. For example:

```
<listOfParameters>
  <specie name="Km1" value="2.3"/>
  <specie name="Km2" value="10.7"/>
  ...
</listOfParameters>
```

An example of a full model that uses parameters is presented in Appendix A.4.

3.7 Rules

A `Model` object can contain a list of `Rule` elements. Figure 8 shows the class hierarchy of `Rules` classes. The classes `CompartmentRule`, `SpeciesRule` and `ParameterRule` are subtypes of the `Rule` class. A `Rule` element represents a formula of the form `x = y`. A `Rule` has a string attribute `formula` that contains a text string representing a formula in place of `y`. `Rule` elements are evaluated in the order given in the XML stream/file, however there is no restriction on the order.

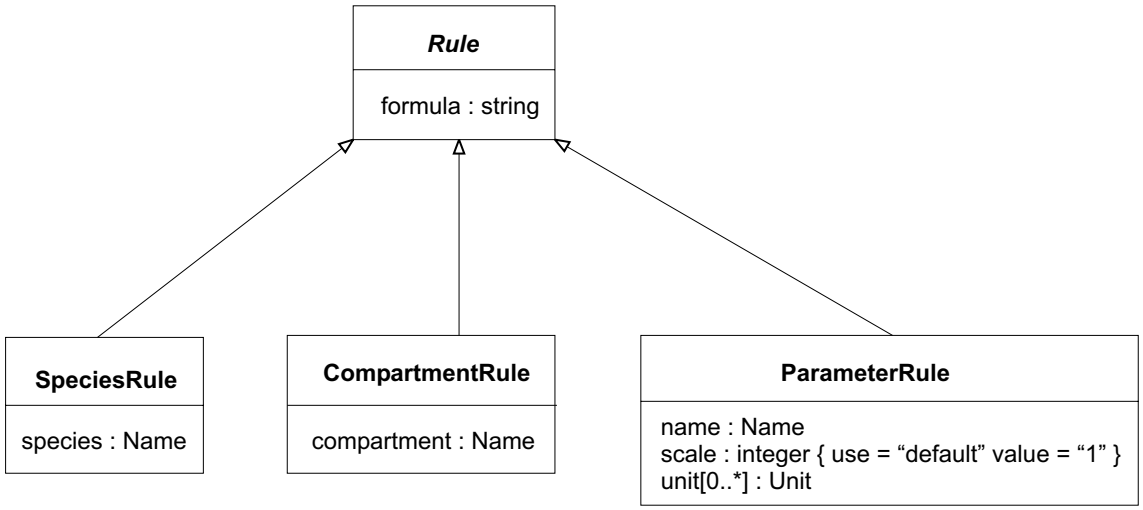

Figure 8: *The `Rule` class definition.*

When the simulation application reads the model specification, it will build a set of ordinary differential equations (ODE). These will be used by the simulator to perform various analyses on the model. The intent is that rule expressions are an integral part of the ODE expression list and must be evaluated by the simulator just before the ODE list.

One of the motivations for including rules like this is to be able to express the computation of the equilibrium concentrations of fast reactions and modeling pH. An example of an XML-encoded model that uses `Rule` elements is given in Appendix A.3.

3.7.1 CompartmentRule

The class `CompartmentRule` has an attribute, `compartment`, that has type `Name` and is used to store a compartment name. `CompartmentRule` inherits attribute `formula`, used to store a formula in volume units that are declared on the referenced compartment element. The effect of the rule is to set the referenced compartment volume to the volume returned by the formula.

3.7.2 SpeciesRule

The class `SpeciesRule` has an attribute, `species`, that has type `Name` and is used to store a species name. `SpeciesRule` inherits the attribute `formula`, which is used to store a formula in concentration

or *substance/volume* units. *Substance units* are those that are declared on the referenced `Specie` element, and *volume units* are those declared on the `compartment` element that contains the `Specie`. The effect of the rule is to set the referenced species concentration to the concentration returned by the formula.

3.7.3 ParameterRule

The class `ParameterRule` has attributes `name`, `scale` and `unit`. The `name` attribute has type `Name`. The `scale` and `unit` operate in the same way as in the `Parameter` class. The inherited attribute `formula` contains a formula in the units defined by the `scale` attribute and `unit` elements. The effect of the rule is to create a new parameter that can be used in subsequent formulae. This parameter has the value returned by the formula in the `formula` attribute.

The following is an example of a `listOfRules` element:

```
<listOfRules>
    <speciesRule species="s2" formula="k*t/(1+k)"/>
    <parameterRule name="t" formula="h*y"/>
    <compartmentRule compartment="cell" formula="z*t">
</listOfRules>
```

3.8 Formulae

Two classes, `KineticLaw`, and `Rule` have the string attribute `formula` that contains formulae. The attribute values are interpreted as expressions that evaluate to a floating-point value.

3.8.1 Operators

All operators in formulae return floating-point values. For boolean operators, 0 is interpreted as "false" and all other values are interpreted as "true". The operators available are shown in Table 3.8.1.

Simulators do not have to support the logical operators in the near future. The operators are listed here simply to reserve the name tokens for the given operation.

Tokens	Operator	Class	Precedence	Associates
names	names	primary	8	n/a
(*expression*)	sub expression	primary	8	n/a
$f(...)$	function call	prefix	8	left
not, !	logical not	unary	7	right
-	negation	unary	6	right
^	power	binary	5	left
*	multiplication	binary	4	left
/	division	binary	4	left
+	addition	binary	3	left
-	subtraction	binary	3	left
and, &&	logical and	binary	2	left
or, ||	logical or	binary	1	left
xor	logical exclusive or	binary	1	left

Table 2: *A table of the expression operators in SBML. In the* ***Class*** *column, "primary" 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` *a*`*b` *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.*

3.8.2 Functions

The function call operator consists of a function name, followed by an opening parenthesis token ('('), followed by a sequence of zero or more arguments separated by commas, followed by a closing parenthesis (')') token. Table 3 in Appendix C lists the basic math functions that are defined in SBML at this time. Table 4 in Appendix D lists all the built-in rate law functions.

3.8.3 Symbols

In formulae, the name tokens (other than function names) are the names of either parameters, parameter rules, compartments or species. For the purposes of this document, we call all of them *symbols*.

When a species name occurs in a formula, it represents the concentration (*substance/volume*) of the specie. The units of the volume are derived from the `volumeScale` attribute value on the `Model` element and the substance units defined for the `Specie` (see Section 3.5).

When a compartment name occurs in a formula, it represents the volume of the compartment. Again, the units of this value are derived from the `volumeScale` attribute value on the `Model` element.

3.9 Namespaces

The names of elements in SBML are constrained.

An element's name is given by the value of the `name` attribute on the element. By default, the name of an element is unique across all the elements of their element's class, but independent of all other element names. For example, reaction names are unique among all reactions in the model, but a reaction can have the same name as a compartment. This rule establishes a *global namespace* for each class.

Symbols declared outside a reaction element are treated slightly differently, in that they all have the same namespace. That is, a symbol is unique among all other symbols but is independent of all other element names.

The names of parameter elements contained in a `reaction` element are unique to just the reaction and override symbol names declared elsewhere. That is, parameter symbols can be defined in one of two names spaces, a local space confined to particular rate laws, and a global space. The advantage of this approach is the following. Some simulators currently use a local name space approach when declaring rate laws. This allows them to use the same symbol (but different instance) in many different rate laws, reducing the burden on the modeller when collating a set of parameter names. On the other hand, some simulators require the user to generate unique symbol names for every distinct parameter. To accomodate this approach, we introduced the global name space. In addition, the global namespace allows sets of rate equations and rules to share the same parameter symbol (single instance). For example, a particular enzyme might catalyze a number of different reactions, in which case it would be an advantage to specific a single parameter indicating the concentration of enzyme that would appear in all the effected rate laws.

3.10 Reactions

A `Reaction` represents some transformation, transport or binding process, typically a chemical reaction, that can change the amount of one or more species. `Reaction` is defined in Figure 9. In this framework, reactions are defined using lists of reactant species, products, and their stoichiometries, and by parameter values for separately-defined kinetic laws. `Reaction` has a boolean attribute, `reversible`, that has a value of "false" by default.

The examples in Appendix A clarify how the elements of the `Reaction` class are intended to be used.

3.10.1 SpeciesReference

An instance of `SpeciesReference` links a specie to a `Reaction`. This implies that the reaction will affect the amount of that specie. The attribute `specie`, of type `Name`, in the `SpecieReference` class in Figure 9 is

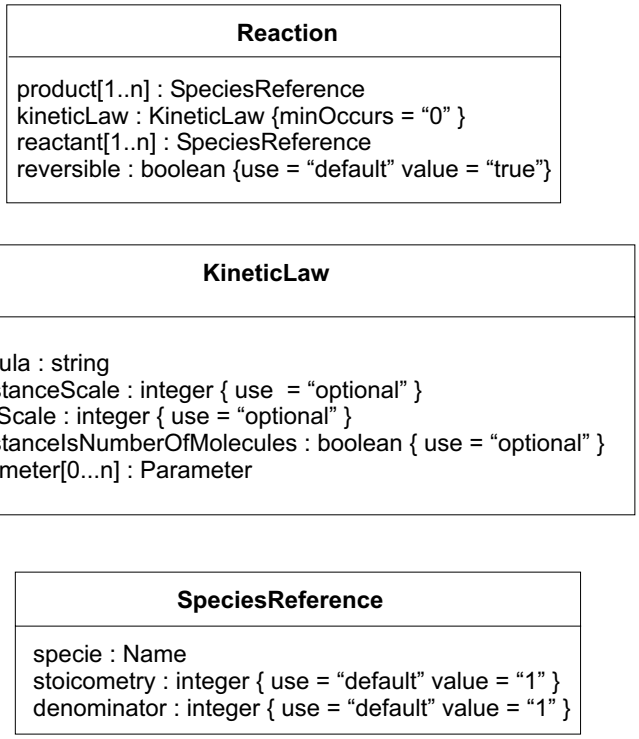

Figure 9: *The* `Reaction` *object class definition.*

intended to refer to the name of a specie in the species list of the `Model`. In other words, the species involved in a reaction are listed once in a `Model` and the `listOfReactants` and `listOfProducts` in reactions refer to the list of species. The instances of `SpeciesReference` are `specieReference` elements in the `listOfReactants` and `listOfProducts`.

The following is a simple example of a `specieReference` element:

```
<specieReference specie="X0" stoichiometry="1"/>
```

The effective stoichiometry value is created by dividing the `stoichiometry` attribute value by the `denominator` attribute value. The attribute `stoichiometry` is of type integer and is optional, defaulting to 1. The attribute `denominator` is of type integer and is optional, defaulting to 1. (This allows the user to employ rational arithmetic computations on the stoichiometry matrix. This helps to eliminate roundoff errors and other problems during the computation, especially for large matrices. These computations are particularly important when calculating things like elementary modes.) Note that the denominator attribute is **optional**.

If the reaction depends on the reactant binding order, such as in an ordered bi-bi reaction, then the order in which the substrate and products bind and leave the enzyme is given by the order of the reactants and products in their respective lists.

3.10.2 KineticLaw

A `kineticLaw` element describes the rate of the enclosing reaction. The `formula`, of type `string`, expresses the rate in *substance/time* units. The exact units for substance and time used can be given via the `substanceScale`, `timeScale` and `substanceIsNumberOfMolecules` attributes on `KineticLaw` in the same way described in the `Model` section above. These attributes are optional and the default values are taken from the `Model` element.

The `KineticLaw` element contained in a `Reaction` element is optional; however, in general there is no default element that can be substituted in place of a missing `kineticLaw` element. `KineticLaw` contains zero or more

`parameter` elements that define symbols that can be used in the formula string. These symbols overload those defined at the `Model` level.

The following is a simple example of a `KineticLaw` element.

```
<kineticLaw formula="k1*X0">
    <listOfParameters>
        <parameter name="k1" value="0"/>
    </listOfParameters>
</kineticLaw>
```

3.10.3 An Example of a Complete Reaction Element

The following is an example of a `reaction` element and defines the reaction

$J_1 : \ X_o \longrightarrow S_1; \ k_1 X_0$

```
<reaction name="J1">
    <listOfReactants>
        <specieReference specie="X0" stoichiometry="1"/>
    </listOfReactants>
    <listOfProducts>
        <specieReference specie="S1" stoichiometry="1"/>
    </listOfProducts>
    <kineticLaw formula="k1*X0">
        <listOfParameters>
            <parameter name="k1" value="0"/>
        </listOfParameters>
    </kineticLaw>
</reaction>
```

4 Versions of the Markup Language

The top level element `sbml` has the integer attribute `sbmlVersion` which indicates the version number of the SBML definition with which the XML stream/file complies.

5 Future Enhancements

We are currently considering the following features for future versions of SBML:

- *Literature References.* This feature will allow the models to be annotated with references to papers and authors.
- *Explicit references to ODEs.* A model should be able to explicitly specify ordinary differential equations alongside the rules and reactions. One application of this would be to allow the modelling of variable volume spaces.
- *Submodels.* This feature will allow the reuse of model libraries and the creation of several instances of the same model.
- *Indices.* This feature will allow sets of similar reactions to be defined that transform sets of species. A Formula string could then include 'sum' and 'product' functions.
- *DNA.* This feature will allow DNA to be explicitly modeled.
- *Diagrams.* This feature will allow elements to be annotated with data to enable the display of the model in a diagram.
- *Multiple state species.* This feature will allow species to have multiple states.

6 Your Comments

Please use the group email address (`sysbio@caltech.edu`) and web site `http://www.cds.caltech.edu/erato/` to send us your comments and suggestions.

Appendix

A Using the XML Encoding of SBML

In this section, we present an example of translating a model into the systems biology model description language defined in this document. Our approach to translating the object class definitions presented in the sections above is described in the companion document, *A Notation for Describing Model Representation Intended for XML Encoding* (Hucka, 2000). Appendix B gives the full listing of a preliminary version of an XML Schema corresponding to SBML, the Systems Biology Markup Language.

A.1 Minimal Model

The following is an example of a model that uses the minimum number of elements and attributes possible in SBML.

```
<sbml version="1">
    <model>
        <listOfCompartments>
            <compartment name="x"/>
        </listOfCompartments>
        <listOfSpecies>
            <specie name="y" compartment="x" initialAmount="1"/>
        </listOfSpecies>
        <listOfReactions>
            <reaction name="x">
                <listOfReactants>
                    <specieReference specie="y"/>
                </listOfReactants>
                <listOfProducts>
                    <specieReference specie="y"/>
                </listOfProducts>
            </reaction>
        </listOfReactions>
    </model>
</sbml>
```

A.2 A Simple Example Application of SBML

The following example is the main portion of an XML document that describes a simple branch system of the following form:

$$X0 \xrightarrow{k1 * X0} S1$$

$$S1 \xrightarrow{k2 * S1} X1$$

$$S1 \xrightarrow{k3 * S1} X2$$

The following is the XML encoding of the above reaction.

```
<sbml version="1">
    <model id="simplemodel" 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*X0;</p>
                <p>reaction-2:   S1 -> X1; k2*S1;</p>
                <p>reaction-3:   S1 -> X2; k3*S1;</p>
```

```
            </body>
        </notes>
        <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*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="k2*S1">
                    <listOfParameters>
                        <parameter name="k2" value="0"/>
                    </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>
```

The corresponding XML encoding shown above is quite straightforward. The outermost container is a tag, `smbl`, that identifies the contents as being systems biology markup language, an application of XML. The next-inner container is a single `model` element that serves as the highest-level object in the model. The model in this case contains a single `compartment` element.

The compartment has four species associated with it, and three reactions. The correspondence between these elements and the three reaction equations in the list above should be fairly obvious. The elements in

the `listOfReactants` and `listOfProducts` refer to the names of elements listed in the `listOfSpecies`.

The lone `compartment` has no value for `volume` because the value is irrelevant for this particular simple case example. Since it is an optional attribute, there is no mention of it in the XML object.

Note that the `notes annotation` elements of the specie and reaction elements above have been omitted because they are empty. The XML Schema definition in Appendix B is defined in such a way that these elements are optional.

A.3 Simple Use of Units Feature in a Model

The following model uses the units features of SBML. In this model, the `substanceScale` attribute on the `model` element has the value -3 that defines all qualities of substance to be defined in 10^{-3} mole units or milliMoles. This sets the default units in the model; elements can override this scale locally. The `volumeScale` and `timeScale` attributes are not set, ensuring that volume is in litres and time is in seconds. Thus, by default in this model, kinetic law formulae define rates in milliMoles per second and the specie symbols in them represent concentration values in milliMoles per litre. All the `specie` elements set the initial amount of the given specie to 1 milliMole.

The parameters `Vm` and `Km` are defined to be in milliMoles per Litre per Second and milliMolar respectively. This scale definition is entirely arbitrary and is not linked by SBML to the built in units described in the previous paragraph.

```
<sbml version="1">
    <model substanceScale="-3">
        <listOfCompartments>
            <compartment name="cell"/>
        </listOfCompartments>
        <listOfSpecies>
            <specie name="x0" compartment="cell" initialAmount="1"/>
            <specie name="x1" compartment="cell" initialAmount="1"/>
            <specie name="s1" compartment="cell" initialAmount="1"/>
            <specie name="s2" compartment="cell" initialAmount="1"/>
        </listOfSpecies>
        <listOfParameters>
            <parameter name="vm" value="2" scale="-3">
                <listOfUnits>
                    <unit type="mole"/>
                    <unit type="litre" power="-1"/>
                    <unit type="second" power="-1"/>
                </listOfUnits>
            </parameter>
            <parameter name="km" value="2" scale="-3">
                <listOfUnits>
                    <unit type="mole"/>
                </listOfUnits>
            </parameter>
        </listOfParameters>
        <listOfReactions>
            <reaction name="v1">
                <listOfReactants>
                    <specieReference specie="x0"/>
                </listOfReactants>
                <listOfProducts>
                    <specieReference specie="s1"/>
                </listOfProducts>
                <kineticLaw formula="(vm*s1)/(km+s1)"/>
            </reaction>
            <reaction name="v2">
                <listOfReactants>
                    <specieReference specie="s1"/>
                </listOfReactants>
                <listOfProducts>
                    <specieReference specie="s2"/>
                </listOfProducts>
```

```
            <kineticLaw formula="(vm*s2)/(km+s2)"/>
        </reaction>
        <reaction name="v3">
            <listOfReactants>
                <specieReference specie="s2"/>
            </listOfReactants>
            <listOfProducts>
                <specieReference specie="x1"/>
            </listOfProducts>
            <kineticLaw formula="(vm*s1)/(km+s1)"/>
        </reaction>
    </listOfReactions>
  </model>
</sbml>
```

A.4 A Simple Example Application Using Rules

The following model represents the pathway $X_0 \rightarrow S_1 \rightarrow S_2 \rightarrow X_1$, where $S_1 \rightarrow S_2$ is a fast reaction. The reaction $S_1 \rightarrow S_2$ is not modeled explicitly; instead, the effect of the reaction is encapsulated in rules.

```
<sbml version="1">
  <model>
    <listOfCompartments>
      <compartment name="cell" volume="1"/>
    </listOfCompartments>
    <listOfSpecies>
      <specie name="s1" compartment="cell" initialAmount="4"/>
      <specie name="s2" compartment="cell" initialAmount="2"/>
      <specie name="x0" compartment="cell" initialAmount="1"/>
      <specie name="x1" compartment="cell" initialAmount="0"/>
    </listOfSpecies>
    <listOfParameters>
      <parameter name="k1" value="1.2"/>
      <parameter name="k2" value="1000"/>
      <parameter name="k3" value="3000"/>
      <parameter name="k4" value="4.5"/>
    </listOfParameters>
    <listOfRules>
      <parameterRule formula="s1 + s2" name="t"/>
      <parameterRule formula="k3/k2" name="k" />
      <specieRule formula="k*t/(1+k)" species="s2"/>
      <specieRule formula="t-s2" species="s1"/>
    </listOfRules>
    <listOfReactions>
      <reaction name="j1">
        <listOfReactants>
          <specieReference specie="x0"/>
        </listOfReactants>
        <listOfProducts>
          <specieReference specie="s1"/>
        </listOfProducts>
        <kineticLaw formula="k1*x0"/>
      </reaction>
      <reaction name="j3">
        <listOfReactants>
          <specieReference specie="s2"/>
        </listOfReactants>
        <listOfProducts>
          <specieReference specie="x1"/>
        </listOfProducts>
        <kineticLaw formula="k4*s2"/>
      </reaction>
    </listOfReactions>
  </model>
</sbml>
```

B XML Schema for SBML

The following is a first draft of an XML Schema definition for the Systems Biology Markup Language. Example applications of this XML Schema are presented in Appendix A.

The schema below is well-formed and compiles with the XML Schema standard. However, the use of `unique` and `keyref` elements in this schema has not been fully validated. The other elements have been fully validated. In practice, the `unique` and `keyref` elements may either need to be commented out or will be ignored, because they are not supported by most XML parsers.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE xsd:schema PUBLIC "-//W3C//DTD XMLSCHEMA 19991216//EN" "" [
    <!ENTITY % p 'xsd:'>
    <!ENTITY % s ':xsd'>]>
<xsd:schema xmlns:xsd="http://www.w3.org/1999/XMLSchema">
    <xsd:annotation>
        <xsd:documentation>
          File name   : sbml.xsd
          Description : XML Schema for the Systems Biology Markup Language
          Organization: Caltech ERATO Kitano
          Version     : 1
          Modified    : 2000-09-12 10:58 PDT
        </xsd:documentation>
    </xsd:annotation>
    <!-- Name -->
    <xsd:simpleType base="xsd:string" name="Name">
        <xsd:pattern value="(_|[a-z]|[A-Z])(_|[a-z]|[A-Z]|[0-9])*"/>
    </xsd:simpleType>
    <!-- Identifed -->
    <xsd:complexType name="Identified" abstract="true">
        <xsd:element name="notes" maxOccurs="1" minOccurs="0">
            <xsd:complexType>
                <xsd:any namespace="http://www.w3.org/1999/xhtml"
                         maxOccurs="unbounded" minOccurs="1" processContents="skip"/>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="annotation" maxOccurs="1" minOccurs="0">
            <xsd:complexType>
                <xsd:any maxOccurs="*" minOccurs="1" processContents="skip"/>
            </xsd:complexType>
        </xsd:element>
        <xsd:attribute name="id" type="xsd:ID" use="optional"/>
        <xsd:attribute name="simpleNotes" type="xsd:string" use="optional"/>
    </xsd:complexType>
    <!-- listOfParameters -->
    <xsd:element name="listOfParameters">
        <xsd:complexType>
            <xsd:element name="parameter" type="Parameter" minOccurs="1" maxOccurs="unbounded"/>
        </xsd:complexType>
    </xsd:element>
    <!--listOfUnits-->
    <xsd:element name="listOfUnits">
        <xsd:complexType>
            <xsd:element name="unit" type="Unit" minOccurs="1" maxOccurs="unbounded"/>
        </xsd:complexType>
    </xsd:element>
    <!-- specieReference -->
    <xsd:element name="specieReference" type="SpecieReference"
                 minOccurs="1" maxOccurs="unbounded"/>
    <!-- Model -->
    <xsd:complexType name="Model" base="Identified" derivedBy="extension">
        <xsd:element name="listOfCompartments" minOccurs="1" maxOccurs="1">
            <xsd:complexType>
                    <xsd:element name="compartment" type="Compartment"
                                 minOccurs="1" maxOccurs="unbounded"/>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="listOfGeometries" minOccurs="0" maxOccurs="1">
```

```
        <xsd:complexType>
                <xsd:element name="geometry" type="Geometry"
                             minOccurs="1" maxOccurs="unbounded"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="listOfMappings" minOccurs="0" maxOccurs="1">
        <xsd:complexType>
                <xsd:element name="mapping" type="Mapping"
                             minOccurs="1" maxOccurs="unbounded"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="listOfSpecies" minOccurs="1" maxOccurs="1">
        <xsd:complexType>
                <xsd:element name="specie" type="Specie"
                             minOccurs="1" maxOccurs="unbounded"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element ref="listOfParameters" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="listOfRules" minOccurs="0" maxOccurs="1">
        <xsd:complexType>
            <xsd:choice maxOccurs="unbounded" minOccurs="1">
                <xsd:element name="compartmentRule" type="CompartmentRule"
                             minOccurs="0" maxOccurs="1"/>
                <xsd:element name="specieRule" type="SpecieRule"
                             minOccurs="0" maxOccurs="1"/>
                <xsd:element name="parameterRule" type="ParameterRule"
                             minOccurs="0" maxOccurs="1"/>
            </xsd:choice>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="listOfReactions" minOccurs="1" maxOccurs="1">
        <xsd:complexType>
            <xsd:element name="reaction" type="Reaction"
                         minOccurs="1" maxOccurs="unbounded">
                <xsd:unique name="parameters">
                     <xsd:selector>listOfParameters/parameter</xsd:selector>
                     <xsd:field>@name</xsd:field>
                </xsd:unique>
            </xsd:element>
        </xsd:complexType>
    </xsd:element>
    <xsd:attribute name="name" type="Name" use="optional"/>
    <xsd:attribute name="substanceScale" type="xsd:integer" use="default" value="0"/>
    <xsd:attribute name="timeScale" type="xsd:integer" use="default" value="0"/>
    <xsd:attribute name="volumeScale" type="xsd:integer" use="default" value="0"/>
    <xsd:attribute name="lengthScale" type="xsd:integer" use="default" value="-6"/>
    <xsd:attribute name="substanceIsNumberOfMolecules" type="xsd:boolean"
                   use="default" value="false"/>
</xsd:complexType>
<!-- Compartment -->
<xsd:complexType name="Compartment" base="Identified" derivedBy="extension">
    <xsd:attribute name="name" type="Name" use="required"/>
    <xsd:attribute name="volume" type="xsd:float" use="default" value="1"/>
</xsd:complexType>
<!-- Geometry -->
<xsd:complexType name="Geometry" base="Identified" derivedBy="extension">
    <xsd:element name="listOfPoints" minOccurs="0" maxOccurs="1">
        <xsd:complexType>
            <xsd:element name="point" type="Point" minOccurs="1" maxOccurs="unbounded"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:attribute name="name" type="Name" use="required"/>
    <xsd:attribute name="length" type="xsd:float" use="optional"/>
    <xsd:attribute name="surfaceArea" type="xsd:float" use="optional"/>
    <xsd:attribute name="volume" type="xsd:float" use="optional"/>
    <xsd:attribute name="lengthScale" type="xsd:integer" use="optional"/>
</xsd:complexType>
<xsd:complexType name="Point" base="Identified" derivedBy="extension">
    <xsd:attribute name="x" type="xsd:float" use="required"/>
```

```
    <xsd:attribute name="y" type="xsd:float" use="required"/>
</xsd:complexType>
<!--Mapping-->
<xsd:complexType name="Mapping" base="Identified" derivedBy="extension">
    <xsd:attribute name="compartment" type="Name" use="required"/>
    <xsd:attribute name="geometry" type="Name" use="required"/>
</xsd:complexType>
<!-- Specie -->
<xsd:complexType name="Specie" base="Identified" derivedBy="extension">
    <xsd:attribute name="name" type="Name" use="required"/>
    <xsd:attribute name="compartment" type="Name" use="required"/>
    <xsd:attribute name="initialAmount" type="xsd:float" use="required"/>
    <xsd:attributeGroup ref="substanceUnits"/>
    <xsd:attribute name="boundaryCondition" type="xsd:boolean" use="default" value="false"/>
    <xsd:attribute name="charge" type="xsd:integer" use="optional"/>
</xsd:complexType>
<!-- Parameter -->
<xsd:complexType name="Parameter" base="Identified" derivedBy="extension">
    <xsd:element ref="listOfUnits" minOccurs="0" maxOccurs="1"/>
    <xsd:attribute name="value" type="xsd:float" use="required"/>
    <xsd:attribute name="scale" type="xsd:integer" use="default" value="0"/>
    <xsd:attribute name="name" type="Name" use="required"/>
</xsd:complexType>
<xsd:simpleType base="xsd:string" name="UnitKind">
    <xsd:enumeration value="mole"/>
    <xsd:enumeration value="litre"/>
    <xsd:enumeration value="second"/>
    <xsd:enumeration value="metre"/>
    <xsd:enumeration value="gram"/>
    <xsd:enumeration value="ampere"/>
    <xsd:enumeration value="kelvin"/>
    <xsd:enumeration value="centigrade"/>
    <xsd:enumeration value="candela"/>
    <xsd:enumeration value="radian"/>
    <xsd:enumeration value="streadian"/>
    <xsd:enumeration value="hertz"/>
    <xsd:enumeration value="newton"/>
    <xsd:enumeration value="joule"/>
    <xsd:enumeration value="calorie"/>
    <xsd:enumeration value="watt"/>
    <xsd:enumeration value="coulomb"/>
    <xsd:enumeration value="volt"/>
    <xsd:enumeration value="farad"/>
    <xsd:enumeration value="ohm"/>
    <xsd:enumeration value="weber"/>
    <xsd:enumeration value="tesla"/>
    <xsd:enumeration value="henry"/>
    <xsd:enumeration value="lumen"/>
    <xsd:enumeration value="lux"/>
    <xsd:enumeration value="pascal"/>
    <xsd:enumeration value="siemens"/>
    <xsd:enumeration value="becquerel"/>
    <xsd:enumeration value="gray"/>
</xsd:simpleType>
<xsd:complexType name="Unit" base="Identified" derivedBy="extension">
    <xsd:attribute name="type" type="UnitKind" use="required"/>
    <xsd:attribute name="power" type="xsd:integer" use="default" value="1"/>
</xsd:complexType>
<!-- Rule -->
<xsd:complexType name="Rule" base="Identified" derivedBy="extension" abstract="true">
    <xsd:attribute name="formula" type="xsd:string" use="required"/>
</xsd:complexType>
<xsd:complexType name="CompartmentRule" base="Rule" derivedBy="extension">
    <xsd:attribute name="compartment" type="Name" use="required"/>
</xsd:complexType>
<xsd:complexType name="SpecieRule" base="Rule" derivedBy="extension">
    <xsd:attribute name="species" type="Name" use="required"/>
</xsd:complexType>
<xsd:complexType name="ParameterRule" base="Rule" derivedBy="extension">
```

```
    <xsd:element ref="listOfUnits" minOccurs="0" maxOccurs="1"/>
    <xsd:attribute name="name" type="Name" use="required"/>
    <xsd:attribute name="scale" type="xsd:integer" use="default" value="0"/>
</xsd:complexType>
<!-- Reaction -->
<xsd:complexType name="Reaction" base="Identified" derivedBy="extension">
    <xsd:element name="listOfReactants" minOccurs="1" maxOccurs="1">
        <xsd:complexType>
            <xsd:element ref="specieReference" minOccurs="1" maxOccurs="unbounded"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="listOfProducts" minOccurs="1" maxOccurs="1">
        <xsd:complexType>
            <xsd:element ref="specieReference" minOccurs="1" maxOccurs="unbounded"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="kineticLaw" type="KineticLaw" minOccurs="0" maxOccurs="1"/>
    <xsd:attribute name="name" type="Name" use="required"/>
    <xsd:attribute name="reversible" type="xsd:boolean" use="default" value="false"/>
</xsd:complexType>
<xsd:complexType name="KineticLaw" base="Identified" derivedBy="extension">
    <xsd:element ref="listOfParameters" minOccurs="0" maxOccurs="1"/>
    <xsd:attribute name="formula" type="xsd:string" use="required"/>
    <xsd:attributeGroup ref="substanceUnits"/>
    <xsd:attribute name="timeScale" type="xsd:integer" use="optional"/>
</xsd:complexType>
<xsd:complexType name="SpecieReference" base="Identified" derivedBy="extension">
    <xsd:attribute name="specie" type="xsd:string" use="required"/>
    <xsd:attribute name="stoichiometry" type="xsd:integer" use="default" value="1"/>
    <xsd:attribute name="denominator" type="xsd:integer" use="default" value="1"/>
</xsd:complexType>
<!-- substanceUnits -->
<xsd:attributeGroup name="substanceUnits">
    <xsd:attribute name="substanceScale" type="xsd:integer" use="optional"/>
    <xsd:attribute name="substanceIsNumberOfMolecules" type="xsd:boolean" use="optional"/>
</xsd:attributeGroup>
<!-- Top-level elements allowed in an sbml document. -->
<xsd:complexType name="sbmlDocument">
    <xsd:element name="model" minOccurs="1" maxOccurs="1" type="Model">
        <xsd:unique name="compartments">
                <xsd:selector>listOfComparments/compartment</xsd:selector>
                <xsd:field>@name</xsd:field>
        </xsd:unique>
        <xsd:unique name="geometries">
                <xsd:selector>listOfGeometries/geometry</xsd:selector>
                <xsd:field>@name</xsd:field>
        </xsd:unique>
        <xsd:unique name="species">
                <xsd:selector>listOfSpecies/specie</xsd:selector>
                <xsd:field>@name</xsd:field>
        </xsd:unique>
        <xsd:unique name="reactions">
                <xsd:selector>listOfReactions/reaction</xsd:selector>
                <xsd:field>@name</xsd:field>
        </xsd:unique>
        <xsd:unique name="globalSymbols">
                <xsd:selector>*/*[self::compartment or self::parameter
                     or self::species or self::parameterRule]</xsd:selector>
                <xsd:field>@name</xsd:field>
        </xsd:unique>
        <xsd:keyref name="mappingToCompartment" refer="compartments">
                <xsd:selector>listOfMappings/mapping</xsd:selector>
                <xsd:field>@compartment</xsd:field>
        </xsd:keyref>
        <xsd:keyref name="mappingToGeometry" refer="geometries">
                <xsd:selector>listOfMappings/mapping</xsd:selector>
                <xsd:field>@geometry</xsd:field>
        </xsd:keyref>
        <xsd:keyref name="specieToCompartment" refer="compartments">
```

```
                <xsd:selector>listOfSpecies/specie</xsd:selector>
                <xsd:field>@compartment</xsd:field>
            </xsd:keyref>
            <xsd:keyref name="specieReferenceToSpecie" refer="species">
                <xsd:selector>listOfReactions/reaction/*/specieReference</xsd:selector>
                <xsd:field>@specie</xsd:field>
            </xsd:keyref>
            <xsd:keyref name="specieRuleToSpecie" refer="species">
                <xsd:selector>llistOfRules/specieRule</xsd:selector>
                <xsd:field>@specie</xsd:field>
            </xsd:keyref>
            <xsd:keyref name="compartmentRuleToComparment" refer="compartment">
                <xsd:selector>llistOfRules/compartmentRule</xsd:selector>
                <xsd:field>@compartment</xsd:field>
            </xsd:keyref>
        </xsd:element>
        <xsd:attribute name="xmlns"/>
        <xsd:attribute name="version" type="xsd:positiveInteger" use="required"/>
    </xsd:complexType>
    <xsd:element name="sbml" type="sbml:sbmlDocument" minOccurs="1" maxOccurs="1"/>
    <!-- The end. -->
</xsd:schema>
```

C Simple Math Functions in SBML

Table 3 defines the simple mathematical functions available for use in formula expressions in SBML.

Name	Args.	Formula or Meaning	Argument Constraints	Result Constraints
abs	x	absolute value of x		
acos	x	arc cosine of x in radians	$-1.0 \leq x \leq 1.0$	$0 \leq acos(x) \leq \pi$
asin	x	arc sine of x in radians	$-1.0 \leq x \leq 1.0$	$-\pi/2 \leq acos(x) \leq \pi/2$
atan	x	arc tangent of x in radians	$-1.0 \leq x \leq 1.0$	$-\pi/2 \leq acos(x) \leq \pi/2$
ceil	x	smallest number not less than x whose value is an exact mathematical integer		
cos	x	cosine of x		
exp	x	e^x where e is the base of the natural logrithms		
floor	x	the largest number not greater than x whose value is an exact integer		
log	x	natural logarithm of x	$x > 0$	
log10	x	base 10 logarithm of x	$x > 0$	
pow	x, y	x^y		
sqr	x	x^2		
sqrt	x	$\sqrt{x}$	$x \geq 0$	$sqrt(x) \geq 0$
sin	x	sine of x		
tan	x	tangent of x		

Table 3: *A table of the simple math functions in SBML.*

D Rate law functions in SBML

Table 4 defines the rate law functions available in formula expressions in SBML. These were extracted from the Gepasi help file (3.21). Segel (1993) provides more information; Hofmeyr and Cornish-Bowden (1997) provide specific details on the reversible Hill equations. In all cases, $Km > 0$, $V_x \geq 0$, $S \geq 0$ and $P \geq 0$.

E Summary of Notation

The definitive explanation for the notation used in this document can be found in the companion notation document. Here we briefly summarize some of the main components of the notations used in describing SBML.

Within the definitions of the various object classes introduced in this document, the following types of expressions ar used many times:

```
name : float
name[0..*] : integer
name : (XHTML)
name : float {use = "optional" value = "0.0"}
```

The symbol `name` represents an attribute on an object. It indicates the name of a field in which data is stored in the object. The colon immediately after the name simply separates the name of the attribute from the *type* of data that it stores. The examples above show the types `float` and `integer` being used.

More complex specifications use square brackets (`[]`) just after the attribute name. This is used to indicate that the attribute contains a list of elements. Specifically, the notation `[0..*]` signifies a list containing zero or more elements; the notation `[1..*]` signifies a list containing at least one element; and so on.

An attribute whose type is shown in parentheses is implemented as as an XML element rather than an XML attribute. The parenthese indicate that the type refers to the type of the element value. Attributes that are lists are also implemented as XML elements.

Expressions in curly braces ({}) shown after an attribute type indicate additional constraints placed on the attribute. We express contraints using XML Schema language. In the examples above, the expression `use="optional" value="0.0"` indicates that the attribute is optional and that it has a default value of 0.0.

Name	Arguments	Meaning	Formula
massi	S_i, k	Irreversible Mass Action Kinetics	$v = k \prod_i S_i$
massr	S_i, P_j, k_1, k_2	Reversible Mass Action Kinetics	$v = k_1 \prod_i S_i - k_2 \prod_j P_j$
uui	S, Vm, Km	Irreversible Simple Michaelis-Menten	$v = \frac{VmS}{Km + S}$
uur	S, P, V_f, V_r, Kms, Kmp	Uni-Uni Reversible Simple Michaelis-Menten	$v = \frac{V_f \frac{S}{Kms} - V_r \frac{P}{Kmp}}{1 + \frac{S}{Kms} + \frac{P}{Kmp}}$
uuhr	S, P, V_f, $Km1$, $Km2$, Keq	Uni-Uni Reversible Simple Michaelis-Menten with Haldane adjustment	$v = \frac{\frac{V_f}{Km1}\left(S - \frac{P}{Keq}\right)}{1 + \frac{S}{Km1} + \frac{P}{Km2}}$
isouur	S, P, V_f, Kms, Kmp, K_{ii}, Keq	Iso Uni-Uni	$v = \frac{V_f\left(S - \frac{P}{Keq}\right)}{S\left(1 + \frac{P}{K_{ii}}\right) + Kms\left(1 + \frac{P}{Kmp}\right)}$
hilli	S, V, $S_{0.5}$, h	Hill Kinetics	$v = \frac{VS^h}{S_{0.5}^h + S^h}$
hillr	S, P, V_f, $S_{0.5}$, $P_{0.5}$, h, K_{eq}	Reversible Hill Kinetics	$v = \frac{\left(V_f \frac{S}{S_{0.5}}\right)\left(1 - \frac{P}{SK_{eq}}\right)\left(\frac{S}{S_{0.5}} + \frac{P}{P_{0.5}}\right)^{h-1}}{1 + \left(\frac{S}{S_{0.5}} + \frac{P}{P_{0.5}}\right)^h}$
hillmr	S, M, P, V_f, K_{eq}, k, h, α	Reversible Hill Kinetics with One Modifier	$v = \frac{\left(V_f \frac{S}{S_{0.5}}\right)\left(1 - \frac{P}{SK_{eq}}\right)\left(\frac{S}{S_{0.5}} + \frac{P}{P_{0.5}}\right)^{h-1}}{K_1 + K_2}$ where $K_1 = \left(\frac{S}{S_{0.5}} + \frac{P}{P_{0.5}}\right)^h, \quad K_2 = \frac{1 + \left(\frac{M}{M_{0.5}}\right)^h}{1 + \alpha\left(\frac{M}{M_{0.5}}\right)^h}$
hillmmr	S, P, M, V_f, K_{eq}, k, h, a, b, α_1, α_2, α_{12}	Reversible Hill Kinetics with Two Modifiers	$v = \frac{\left(V_f \frac{S}{S_{0.5}}\right)\left(1 - \frac{P}{SK_{eq}}\right)\left(\frac{S}{S_{0.5}} + \frac{P}{P_{0.5}}\right)^{h-1}}{K_1 + K_2}$ where $K_1 = \left(\frac{S}{S_{0.5}} + \frac{P}{P_{0.5}}\right)^h, \quad K_2 = \frac{1 + \left(\frac{Ma}{Ma_{0.5}}\right)^h + \left(\frac{Mb}{Mb_{0.5}}\right)^h}{\left[1 + \alpha_1\left(\frac{Ma}{Ma_{0.5}}\right)^h + \alpha_2\left(\frac{Mb}{Mb_{0.5}}\right)^h + \alpha_1\alpha_2\alpha_{12}\left(\frac{Ma}{Ma_{0.5}}\right)^h\left(\frac{Mb}{Mb_{0.5}}\right)^h\right]}$

Table 4: *Table of rate law functions in SBML.*

Name	Arguments	Meaning	Formula
usii	S, V, Km, K_i	Substrate Inhibition Kinetics (Irreversible)	$v = V\dfrac{S/Km}{1 + S/Km + S^2/K_i}$
usir	S, P, V_f, V_r, Kms, Kmp, K_i	Substrate Inhibition Kinetics (Reversible)	$v = \dfrac{V_f S/Kms + V_r P/Kmp}{1 + S/Kms + P/Kmp + S^2/K_i}$
uai	S, V, Ksa, Ksc	Substrate Activation	$v = \dfrac{V\left(S/Ksa\right)^2}{1 + S/Ksc + \left(S/Ksa\right)^2 + S/Ksa}$
ucii	S, V, Km, Ki	Competitive Inhibition (Irreversible)	$v = \dfrac{VS/Km}{1 + S/Km + I/Ki}$
ucir	S, P, V_f, V_r, Kms, Kmp, Ki	Competitive Inhibition (Reversible)	$v = \dfrac{V_f S/Kms - V_r P/Kmp}{1 + S/Kms + P/Kmp + I/Ki}$
unii	S, I, V, Km, Ki	Noncompetitive Inhibition (Irreversible)	$v = \dfrac{VS/Km}{1 + I/Ki + S/Km\,(1 + I/Ki)}$
unir	S, P, I, V_f, Kms, Kmp, Ki	Noncompetitive Inhibition (Reversible)	$v = \dfrac{V_f S/Kms - V_r P/Kmp}{1 + I/Ki + (S/Kms + P/Kmp)\,(1 + I/Ki)}$
uuci	S, I, V, Km, Ki	Uncompetitive Inhibition (Irreversible)	$v = \dfrac{VS/Km}{1 + S/Km\,(1 + I/Ki)}$
uucr	S, P, I, V_f, V_r, Kms, Kmp, Ki	Uncompetitive Inhibition (Reversible)	$v = \dfrac{V_f S/Kms - V_r P/Kmp}{1 + (S/Kms + P/Kmp)\,(1 + I/Ki)}$
umi	S, I, V, Km, Kis, Kic	Mixed Inhibition Kinetics (Irreversible)	$v = \dfrac{VS/Km}{1 + I/Kis + S/Km\,(1 + I/Kic)}$
umr	S, P, I, V_f, V_r, Kms, Kmp, Kis, Kic	Mixed Inhibition Kinetics (Reversible)	$v = \dfrac{V_f S/Kms - V_r P/Kmp}{1 + I/Kis + (S/Kms + P/Kmp)\,(1 + I/Kic)}$
uai	S, A_c, V, Km, Ka	Specific Activation Kinetics - irreversible	$v = \dfrac{VS/Km}{1 + S/Km + Ka/A_c}$
uar	S, P, A_c, V_f, V_r, Kms, Kmp, Ka	Specific Activation Kinetics (Reversible)	$v = \dfrac{V_f S/Kms - V_r P/Kmp}{1 + S/Kms + P/Kmp + Ka/A_c}$
ucti	S, A_c, V, Km, Ka	Catalytic Activation (Irreversible)	$v = \dfrac{VS/Km}{1 + Ka/A_c + S/Km\,(1 + Ka/A_c)}$

Table 4: *Table of rate law functions in SBML (continued).*

Name	Arguments	Meaning	Formula
uctr	S, P, A_c, V_f, V_r, Kms, Kmp, Ka	Catalytic Activation (Reversible)	$v = \frac{V_f S/Kms - V_r P/Kmp}{1 + Ka/A_c + (S/Kms + P/Kmp)(1 + Ka/A_c)}$
umai	S, A_c, V, Km, Kas, Kac	Mixed Activation Kinetics (Irreversible)	$v = \frac{VS/Km}{1 + Kas/A_c + S/Km\,(1 + Kac/A_c)}$
umar	S, P, A_c, V_f, V_r, Kms, Kmp, Kas, Kac	Mixed Activation Kinetics (Reversible)	$v = \frac{V_f S/Kms - V_r P/Kmp}{1 + Kas/A_c + (S/Kms + P/Kmp)(1 + Kac/A_c)}$
uhmi	S, M, V, Km, K_d, a, b	General Hyperbolic Modifier Kinetics (Irreversible)	$v = \frac{VS/Km\,(1 + bM/(aK_d))}{1 + M/K_d + S/Km\,(1 + M/(aK_d))}$
uhmr	S, P, M, V_f, V_r, Kms, Kmp, K_d, a, b	General Hyperbolic Modifier Kinetics (Reversible)	$v = \frac{(V_f S/Kms - V_r P/Kmp)(1 + bM/(aK_d))}{1 + M/K_d + (S/Kms + P/Kmp)(1 + M/(aK_d))}$
ualii	S, I, V, Ks, Ki, n, L	Allosteric inhibition (Irreversible)	$v = \frac{V\,(1 + S/Ks)^{n-1}}{L\,(1 + I/Ki)^n + (1 + S/Ks)^n}$
ordubr	A, P, Q, V_f, V_r, Kma, Kmq, Kmp, Kip, Keq	Ordered UniBi Kinetics	$v = \frac{V_f\,(A - PQ/Keq)}{\Big[Kma + A\,(1 + P/Kip) + V_f/(V_r Keq)\,(KmqP + KmpQ + PQ)\Big]}$
ordbur	A, B, P, V_f, V_r, Kma, Kmb, Kmp, Kia, Keq	Ordered BiUni Kinetics	$v = \frac{V_f\,(AB - P/Keq)}{\Big[AB + KmaB + KmbA + V_f/(V_r Keq)\,(Kmp + P\,(1 + A/Kia))\Big]}$
ordbbr	A, B, P, Q, V_f, Kma, Kmb, Kmp, Kia, Kib, Kip, Keq	Ordered BiBi Kinetics	$v = \frac{V_f\,(AB - PQ/Keq)}{AB\,(1 + P/Kip) + Kmb(A + Kia) + KmaB + K1}$ where $K1 = V_f/(V_r Keq)\,(KmqP\,(1 + A/Kia) + QK2)$, $K2 = Kmp\,(1 + KmaB/(KiaKmb) + P\,(1 + B/Kib))$
ppbr	A, B, P, Q, V_f, V_r, Kma, Kmb, Kmp, Kmq, Kia, Kiq, Keq	Ping Pong BiBi Kinetics	$v = \frac{V_f\,(AB - PQ/Keq)}{AB + KmbA + KmaB\,(1 + Q/Kiq) + K1}$ where $K1 = V_f/(V_r Keq)\,(KmqP\,(1 + A/Kia) + Q(Kmp + P))$

Table 4: *Table of rate law functions in SBML (continued).*

Symbol	Meaning
Si	Substrate
Pi	Product
ki	Rate Constant
$k1$	Forward Rate Constant
$k2$	Reverse Rate Constant
Vm	Forward Maximum Velocity
V_f	Forward Maximum Velocity
V_r	Reverse Maximum Velocity
K_m	Forward Michaelis-Menten Constant
Kms	Substrate Michaelis-Menten Constant
Kmp	Product Michaelis-Menten Constant
Keq	Equilibrium Constant
Kii	Isoinhibition Constant
$S_{0.05}$	Irreversible rate laws: Substrate concentration such that $v = V_f/2$ when $P = 0, M = 0$
$P_{0.05}$	Product concentration s.t. $v = -V_r/2$ when $P = M = 0$ (V_r is limiting rate of reverse reaction)
h	Hill Coefficient
M	Modifier
α	Factor accounting for effect of S and P on binding of M (if $M < 1$, M is inhibitor; if $M > 1$, M is activator)
$M_{0.5}$	Concentration of M that half-saturates its binding site when $S = 0$, $P = 0$
Ki	Inhibition constant for the substrate.
Ksc	Dissociation constant of substrate-active site
Ksa	Dissociation constant of substrate-activation site
I	Inhibitor
A_c	Activator
V	Forward Maximum Velocity
Kis	Specific (competitive) inhibition constant.
Kic	Catalytic (noncompetitive) inhibition constant.
Ka	Activation Constant
Kac	Catalytic Activation Constant
Kas	Specific Activation Constant
Kd	Dissociation constant of the elementary step $E + M = EM$.
a	Ratio of dissociation constant of elementary step $ES+M = ESM$ over that of $E+M = EM$.
b	Ratio of rate constant of elementary step $ESM \rightarrow EM + P$ over that of $ES \rightarrow E + P$.
Ks	Dissociation constant of the substrate from the active form of the enzyme
Ki	Dissociation constant of the inhibitor from the inactive form of the enzyme
L	Equilibrium constant between the active and inactive forms of the enzyme
n	No. binding sites for substrate & inhibitor (most times the number of monomers in enzyme)
A	First substrate in two subatrate reaction
B	Second substrate in two substrate reaction
P	First product in two product reaction
Q	Second product in two product reaction
Kma	Concentration of A such that $v = V_f/2$ (Michaelis constant) at zero P and zero Q
Kmb	Concentration of B such that $v = V_f/2$ (Michaelis constant) at saturating A and zero P
Kmp	Concentration of P such that $v = -V_r/2$ (Michaelis constant) at zero A and B
Kmq	Concentration of Q such that $v = -V_r/2$ (Michaelis constant) at zero A and saturating P
Kip	Product inhibition constant of P acting on the forward reaction
Kia	Product inhibition constant of A acting on the reverse reaction
Kib	Product inhibition constant of B acting on the reverse reaction.
Kia	Product inhibition constant of A acting on the reverse reaction (Ping-pong)
Kiq	Product inhibition constant of Q acting on the forward reaction.

Table 5: *Table of symbols used in Table 4.*

References

Biron, P. V., and Malhotra, A. (2000). XML Schema Part 2: Datatypes (W3C Working Draft 7 April 2000). Available via the World Wide Web at `http://www.w3.org/TR/xmlschema-2/`.

Bosak, J., and Bray, T. (1999). XML and the Second-Generation Web. Scientific American, May. Also available via the World Wide Web at `http://www.sciam.com/1999/0599issue/0599bosak.html`.

Bray, T., Paoli, J., and Sperberg-McQueen, C. M. (1998) Extensible Markup Language (XML) 1.0, W3C Recommendation 10-February-1998. Available via the World Wide Web at `http://www.w3.org/TR/1998/REC-xml-19980210`.

Eriksson, H.-E., and Penker, M. (1998). UML Toolkit. New York: John Wiley & Sons.

Hucka, M. (2000). A Notation for Describing Model Representations Intended for XML Encoding. Available via FTP at `ftp://ftp.cds.caltech.edu/pub/caltech-erato/notation/`.

Oestereich, B. (1999). Developing Software with UML: Object-Oriented Analysis and Design in Practice. Addison-Wesley.

St. Laurent, S. (2000). XML Elements of Style. New York: McGraw-Hill.

Fallside, D. C. (2000). XML Schema Part 0: Primer (W3C Working Draft, 7 April 2000). Available via the World Wide Web at `http://www.w3.org/TR/xmlschema-0/`.

Segel, I.H (1993). Enzyme Kinetics, Wiley Classics Library Edition

Hofmeyr and Cornish-Bowden (1997). Comput. Appl. Biosci. 13, 377 - 385