Systems Biology Markup Language (SBML) Level 1: Structures and Facilities for Basic Model Definitions

Michael Hucka, Andrew Finney, Herbert Sauro, Hamid Bolouri
{mhucka,afinney,hsauro,hbolouri}@cds.caltech.edu
Systems Biology Workbench Development Group
ERATO Kitano Systems Biology Project
Control and Dynamical Systems, MC 107-81
California Institute of Technology, Pasadena, CA 91125, USA
http://www.cds.caltech.edu/erato

Principal Investigators: John Doyle and Hiroaki Kitano

SBML Level 1, Version 1 (Final), 2 March 2001

Contents

1 Introduction

We present the **S**ystems **B**iology **M**arkup **L**anguage (SBML) Level 1, a description language for simulations in systems biology. SBML is oriented towards representing biochemical networks common in research on a number of topics, including cell signaling pathways, metabolic pathways, biochemical reactions, gene regulation, and many others. A recent conference (Kitano, 2001) highlights the range of topics that fall under the umbrella of *systems biology* and are in the domain of the description language defined here. Many contemporary research initiatives demonstrate the growing popularity of this kind of multidisciplinary work (e.g., Abbott, 1999; Gilman, 2000; Popel and Winslow, 1998; Smaglik, 2000a,b).

SBML Level 1 is the result of merging modeling-language features from the following simulation systems: *BioSpice* (Arkin, 2001), *DBSolve* (Goryanin, 2001; Goryanin et al., 1999), *E-Cell* (Tomita et al., 1999, 2001), *Gepasi* (Mendes, 1997, 2001), *Jarnac* (Sauro, 2000; Sauro and Fell, 1991), *StochSim* (Bray et al., 2001; Morton-Firth and Bray, 1998), and *Virtual Cell* (Schaff et al., 2000, 2001). SBML was developed with the help of the authors of these packages. As a result of being based on actual working simulation software, it is a practical and functional description language. Our goal in creating it has been to provide an open standard that will enable simulation software to exchange models, something that is currently impossible because there is no standard model exchange language. We expect SBML models to be encoded using XML, the eXtensible Markup Language (Bosak and Bray, 1999; Bray et al., 1998), and we include here an XML Schema that defines SBML Level 1.

1.1 Scope and Limitations

SBML Level 1 is meant to support non-spatial biochemical models and the kinds of operations that are possible in existing analysis/simulation tools. A number of potentially desirable features have been intentionally omitted from the language definition. Future software tools will undoubtedly require the evolution of SBML; we expect that subsequent releases of SBML (termed *levels*) will add additional structures and facilities currently missing from Level 1, once the simulation community gains experience with the current language definition. In Section 6.1, we discuss extensions that will likely be included in SBML Level 2.

The definition of the model description language presented here does not specify *how* programs should communicate or read/write SBML. We assume that for a simulation program to communicate a model encoded in SBML, the program will have to translate its internal data structures to and from SBML, use a suitable transmission medium and protocol, etc., but these issues are outside of the scope of this document.

1.2 Notational Conventions

SBML is intended to be a common XML-based format for encoding systems biology models in a simple form that software tools can use as an exchange format. However, for easier communication to human readers, we define SBML using a graphical notation based upon UML, the Unified Modeling Language (Eriksson and Penker, 1998; Oestereich, 1999). This UML-based definition in turn is used to define an XML Schema (Biron and Malhotra, 2000; Fallside, 2000; Thompson et al., 2000) for SBML. There are three main advantages to using UML as a basis for defining SBML data structures. First, compared to using other notations or a programming language, the UML visual representations are generally easier to grasp by readers who are not computer scientists. Second, the visual notation is implementation-neutral: the defined structures can be encoded in any concrete implementation language—not just XML, but C or Java as well. Third, UML is a de facto industry standard that is documented in many sources. Readers are therefore more likely to be familiar with it than other notations.

Our notation and our approach for mapping UML to XML Schemas is explained in a separate document (Hucka, 2000). A summary of the essential points is presented in Appendix A, and examples throughout this document illustrate the approach. We also follow certain naming and typographical conventions throughout this document. Specifically, the names of data structure attributes or fields begin with a lowercase letter, and the names of data structures and types begin with an uppercase letter. Keywords (names of types, XML elements, etc.) are written in a typewriter-style font; for example, `Compartment` is a type name and `compartment` is a field name. Likewise, literal XML examples are also written in a typewriter-style font.

2 Overview of SBML

The following is an example of a simple, hypothetical biochemical network that can be represented in SBML:

$$X_0 \quad \xrightarrow{k_1 X_0} \quad S_1$$

$$S_1 \quad \xrightarrow{k_2 S_1} \quad X_1$$

$$S_1 \quad \xrightarrow{k_3 S_1} \quad X_2$$
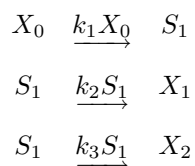
Broken down into its constituents, this model contains a number of components: reactant species, product species, reactions, rate laws, and parameters in the rate laws. To analyze or simulate this network, additional components must be made explicit, including compartments for the species, and units on the various quantities. The top level of an SBML model definition simply consists of lists of these components:

> *beginning of model definition*
> > *list of unit definitions (optional)*
> > *list of compartments*
> > *list of species*
> > *list of parameters (optional)*
> > *list of rules (optional)*
> > *list of reactions*
> *end of model definition*

The meaning of each component is as follows:

Unit definition: A name for a unit used in the expression of quantities in a model. Units may be supplied in a number of contexts in an SBML model, and it is convenient to have a facility for both setting default units and for allowing combinations of units to be given abbreviated names.

Compartment: A container of finite volume for substances. In SBML Level 1, a compartment is primarily a topological structure with a volume but no geometric qualities.

Specie: A substance or entity that takes part in a reaction. Some example species are ions such as Ca^{2++} and molecules such as glucose or ATP. The primary qualities associated with a specie in SBML Level 1 are its initial amount and the compartment in which it is located.

Reaction: A statement describing some transformation, transport or binding process that can change the amount of one or more species. For example, a reaction may describe how certain entities (reactants) are transformed into certain other entities (products). Reactions have associated rate laws describing how quickly they take place.

Parameter: A quantity that has a symbolic name. SBML Level 1 provides the ability to define parameters that are global to a model as well as parameters that are local to a single reaction.

Rule: In SBML, a mathematical expression that is added to the differential equations constructed from the set of reactions and can be used to set parameter values, establish constraints between quantities, etc.

A software package can read an SBML model description and translate it into its own internal format for model analysis. For example, a package might provide the ability to simulate the model, by constructing differential equations representing the network and then performing numerical time integration on the equations to explore the model's dynamic behavior.

SBML allows models of arbitrary complexity to be represented. Each type of component in a model is described using a specific type of data structure that organizes the relevant information. The data structures determine how the resulting model is encoded in XML.

In the sections that follow, the various constructs in SBML and their uses are described in detail. Section 3 first introduces a few basic structures that are used throughout SBML, then Section 4 provides details on each of the main components of SBML. Section 5 provides several complete examples of models encoded in XML using SBML.

3   Preliminary Definitions

This section covers certain constructs that are used repeatedly in the rest of SBML and are useful to discuss before diving into the details of the components provided in SBML.

3.1   Type SBase

Each of the main components composing an SBML model definition has a specific data type that is derived directly or indirectly from a single base type called SBase. This inheritance hierarchy is depicted in Figure 1.
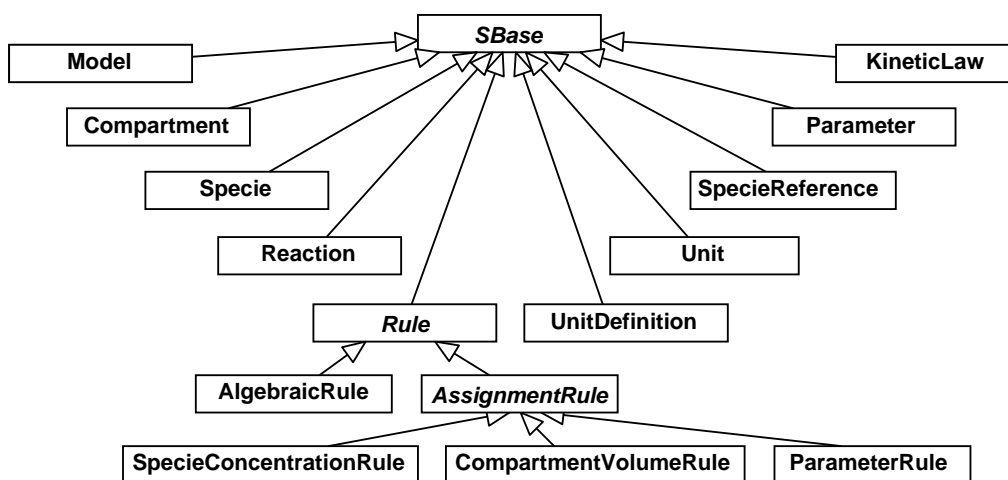
Figure 1: *A UML diagram of the inheritance hierarchy of major data types in SBML. Open arrows indicate inheritance, pointing from inheritors to their parents (Eriksson and Penker, 1998; Oestereich, 1999).*

The type SBase is designed to allow a modeler or a software package to attach information to each component in an SBML model. The definition of SBase is presented in Figure 2. SBase contains two fields, both of which are optional: notes and annotations. The field notes is a container for XHTML content. It is intended for recording optional user-visible annotations. Every data object derived directly or indirectly from type SBase can have a separate value for notes, allowing users considerable freedom for annotating their models. The second field, annotations, is provided for software-generated annotations. It is a container for arbitrary data (XML type any) and is intended to store information not intended for human viewing. As with the user-visible notes field, every data object can have its own annotations value.

In other type definitions presented below, we follow the UML convention of eliding the attributes derived from a parent type such as SBase. It should be kept in mind that these attributes are always available.

Figure 2: *The definition of SBase. Text enclosed in braces next to attribute types (i.e., {minOccurs="1"}) indicates constraints on the possible attribute values; we use XML Schema language to express constraints since we are primarily interested in the XML encoding of SBML.*

3.2 Guidelines for the Use of the `annotations` Field in `SBase`

The `annotations` field in the definition of `SBase` is formally unconstrained in order that software developers may attach any information they need to different components in an SBML model. However, it is important that this facility not be misused accidentally. In particular, it is critical that information essential to a model definition is *not* stored in `annotations`. Parameter values, functional dependencies between model components, etc., should not be recorded as annotations.

Here are examples of the kinds of data that may be appropriately stored in `annotations`: (a) Information about graphical layout of model components; (b) application-specific processing instructions that do not change the essence of a model; (c) bibliographic information pertaining to a given model; and (d) identification information for cross-referencing components in a model with items in a database. (We expect to introduce an explicit scheme for recording bibliographic information and making database references in SBML Level 2, at which time using annotations for these purposes will become unnecessary.)

Different applications may use XML Namespaces (Bray et al., 1999) to specify the intended vocabulary of a particular annotation. Here is an example of this kind of usage. Suppose that a particular application wants to annotate data structures in an SBML model definition with screen layout information and a time stamp. The application developers should choose a URI (*Universal Resource Identifier*; Harold and Means 2001; W3C 2000a) reference that uniquely identifies the vocabulary that the application will use for such annotations, and a prefix string to be used in the annotations. For illustration purposes, let us say the URI reference is "`http://www.mysim.org/ns`" and the prefix is `mysim`. An example of an annotation might then be as follows:

```
...
<annotations xmlns:mysim="http://www.mysim.org/ns">
    <mysim:nodecolors mysim:bgcolor="green" mysim:fgcolor="white"/>
    <mysim:timestamp>2000-12-18 18:31 PST</mysim:timestamp>
</annotations>
...
```

The namespace prefix `mysim` is used to qualify the XML elements `mysim:nodecolors` and `mysim:timestamp`; presumably these symbols have meaning to the application. This example places the XML Namespace information on `annotations` itself rather than on a higher-level enclosing construct or the enclosing document level, but other placements would be valid as well (Bray et al., 1999).

The use of XML Namespaces permits multiple applications to place annotations on XML elements of a model without risking interference or element name collisions. Annotations stored by different simulation packages can thus coexist in the same model definition. Although XML Namespace names ("`http://www.mysim.org/`" in the example above) must be URIs references, an XML Namespace name is *not required* to be directly usable in the sense of identifying an actual, retrieval document or resource on the Internet (Bray et al., 1999). The name is simply intended to enable unique identification of constructs, and using URIs is a common and simple way of creating a unique name string. For the convenience of the simulation tools developer community, we reserve certain namespace names for use with annotations in SBML. These reserved names are listed in Table 1.

`http://www.sbml.org/2001/ns/biospice`
`http://www.sbml.org/2001/ns/dbsolve`
`http://www.sbml.org/2001/ns/ecell`
`http://www.sbml.org/2001/ns/gepasi`
`http://www.sbml.org/2001/ns/jarnac`
`http://www.sbml.org/2001/ns/jdesigner`
`http://www.sbml.org/2001/ns/stochsim`
`http://www.sbml.org/2001/ns/vcell`

Table 1: *Reserved XML Namespace names in SBML Level 1.*

Note that the namespaces being referred to here are XML Namespaces specifically in the context of the `annotations` field on `SBase`. The namespace issue here is unrelated to the namespaces discussed in Section 3.4 below in the context of `SName` and symbols in SBML.

3.3   Type `SName`

The type `SName` is used in many places in SBML for expressing names of components in a model. `SName` is is a data type derived from the basic XML type `string`, but with restrictions about the types of characters permitted and the sequence in which they may appear. Its definition is shown in Figure 3.

```
letter ::= 'a'..'z','A'..'Z'
digit  ::= '0'..'9'
SName  ::= { '_' } letter { letter | '_' | digit }
```

Figure 3: *The definition of the type `SName`, expressed in conventional Backus-Naur Form (Naur, 1960). The meta symbols { and } signify "zero or more times" the items they enclose. Note that although XML permits the use of Unicode characters (Unicode Consortium, 1996), SBML limits the allowable characters in `SName` to plain ASCII text characters for compatibility with existing simulation software.*

The need to define a constrained data type for names stems from the fact that many existing simulation packages allow only a limited set of characters in symbol names. SBML codifies this limitation in the form of a lowest-common-denominator data type (`SName`), to prevent the creation of models with symbol names that might confuse some simulation software packages. This is important for facilitating model exchange between tools.

3.4   Component Names and Namespaces in SBML

A biochemical network model can contain a large number of named components representing different parts of a model. This leads to a problem in deciding the scope of a symbol: in what contexts does a given symbol X represent the same thing? The approaches used in existing simulation packages tend to fall into two categories that we may call global and local. The *global* approach places all symbols into a single global namespace, so that a symbol X represents the same thing wherever it appears in a given model definition. The *local* approach places symbols in different namespaces depending on the context, where the context may be, for example, individual rate laws. The latter approach means that a user may use the same symbol X in different rate laws and have each instance represent a different quantity.

The fact that different simulation programs may use different rules for name resolution poses a problem for the exchange of models between simulation tools. Without careful consideration, a model written out in SBML format by one program may be misinterpreted by another program. SBML must therefore include a specific set of rules for treating symbols and namespaces.

The namespace rules in SBML Level 1 are relatively straightforward and are intended to avoid this problem with a minimum of requirements on the implementation of software tools:

- All model-level component names (compartments, species, reactions, parameters, parameter rules, and units) reside in the same global namespace. This means, for example, that a reaction and a specie definition cannot both have the same name.

- Each reaction definition (see Section 4.7) establishes a private local namespace for parameter names. Within the definition of a given reaction, parameter names introduced in that reaction override (shadow) identical names in the global namespace.

- Certain names in SBML Level 1 are reserved or otherwise have special meaning. Table 2 on the following page lists these reserved names. They are comprised of predefined mathematical functions, certain operators (present and expected in the future), and rate law functions. In order to prevent name collisions, these reserved names cannot be used as names for any component of a model.

6

abs	cos	hillr	not	ppbr	tan	ucii	umar	usii	volume
acos	exp	isouur	or	sin	umai	ucir	umi	usir	xor
and	floor	log	ordbbr	sqr	umar	ucti	unii	uuci	
asin	hilli	log10	ordbur	sqrt	uai	uctr	unir	uucr	
atan	hillmmr	massi	ordubr	substance	ualii	uhmi	uuhr	uui	
ceil	hillmr	massr	pow	time	uar	uhmr	umr	uur	

Table 2: *The reserved names in SBML Level 1.*

The set of rules above can enable software packages using either local or global namespaces to exchange SBML model definitions. In particular, software environments using local namespaces internally should be able to accept SBML model definitions without needing to change component names. Environments using a global namespace internally can perform a simple manipulation of the names of elements within reaction definitions to avoid name collisions. (An example approach for the latter would be the following: when receiving an SBML-encoded model, prefix each name inside each reaction with a string constructed from the reaction's name; when writing an SBML-encoded model, strip off the prefix.)

The namespace rules described here provide a clean transition path to future levels of SBML, when submodels are introduced (Section 6.1). Submodels will provide the ability to compose one model from a collection of other models. This capability will have to be built on top of SBML Level 1's namespace organization. A straightforward approach to handling namespaces is to make each submodel's space be private. The rules governing namespaces within a submodel can simply be the Level 1 namespace rule described here, with each submodel having its own (to itself, global) namespace.

3.5 Formulas

Formulas in SBML Level 1 are expressed in text string form. They are used in the definitions of kinetic laws (Section 4.7.2) and in rules (Section 4.6). The formula strings are interpreted as expressions that evaluate to a floating-point value of type `double`. The formula strings may contain operators, function calls, and symbols. Table 3 presents the precedence rules for the different entities that may appear in formula strings. All operators in formulas return `double` values.

Tokens	Operation	Class	Precedence	Associates
name	symbol reference	operand	8	n/a
(*expression*)	expression grouping	operand	8	n/a
$f(...)$	function call	prefix	8	left
$-$	negation	unary	6	right
^	power	binary	5	left
$*$	multiplication	binary	4	left
/	division	binary	4	left
$+$	addition	binary	3	left
$-$	subtraction	binary	3	left

Table 3: *A table of the expression operators available in SBML. 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 ^, which is used in C for a different purpose.*

7

The function call syntax consists of a function name, followed by optional white space, followed by an opening parenthesis token ('('), followed by a sequence of zero or more arguments separated by commas, followed by a closing parenthesis (')') token. The function name must be chosen from one of the functions available in SBML. Table 6 in Appendix C lists the basic mathematical functions that are defined in SBML at this time, while Table 7 lists a large number of common rate law functions defined in SBML. The names of these predefined functions are reserved and make up the bulk of the list of names in Table 2 on the page before.

A program parsing a formula in an SBML model should assume that name tokens other than function names are names of parameters, parameter rules, compartments or species. When a specie name occurs in a formula, it represents the concentration (i.e., *substance/volume*) of the specie. When a compartment name occurs in a formula, it represents the volume of the compartment. The units of substance and volume are determined from the built-in `substance` and `volume` of Table 5 on page 10.
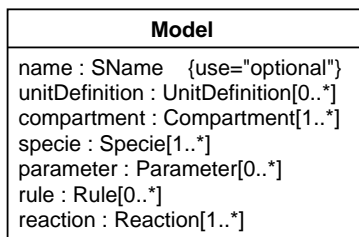
Readers may wonder why mathematical formulas in SBML are not expressed using MathML (W3C, 2000b), an XML-based mathematical formula language. Although using MathML would be more in the spirit of using XML and would in some ways be a more forward-looking choice, it would require simulation software to use fairly complex parsers to read and write the resulting SBML. Most contemporary systems biology simulation software simply represent mathematical formulas using text strings. To keep SBML Level 1 simple and compatible with known simulation software, we chose to represent formulas as strings. This does not preclude a later level of SBML from introducing the ability to use MathML as an extension.

4 SBML Components

In this section, we define each of the major data structures in SBML. To provide illustrations of their use, we give partial XML encodings of SBML model components, but we leave full XML examples to Section 5.

4.1 Models

The `Model` structure is the highest-level construct in an SBML data stream or document. It defines a grouping of components—the list of compartments, species, reactions, parameters, rules and unit definitions that define a given model. Only one component of type `Model` is allowed per instance of an SBML document or data stream, although it does not necessarily need to represent a single biological entity. The UML definition of the `Model` structure is shown in Figure 4.

Model
name : SName   {use="optional"}
unitDefinition : UnitDefinition[0..*]
compartment : Compartment[1..*]
specie : Specie[1..*]
parameter : Parameter[0..*]
rule : Rule[0..*]
reaction : Reaction[1..*]

Figure 4: *The definition of* `Model`. *Additional fields are inherited from* `SBase`.

A model must contain at least one `Specie`, one `Reaction` and one `Compartment` data element, but SBML does not impose restrictions on the total number of these. A `Model` data object may optionally have lists of `UnitDefinition`, `Parameter` and `Rule` components (they are optional because the lists in fields `unitDefinition`, `compartment`, and `rule` are permitted to have zero length). It may also have an optional `name` field that can be used to give the model a name. The name must be a text string conforming to the syntax permitted by the `SName` data type described in Section 3.3.

In the XML encoding of an SBML model, the lists of species, reactions, compartments, and optional unit definitions, parameters and rules, are translated into lists of XML elements that each have headings of the form `listOf_____s`, where the blank is replaced by the name of the component type (e.g., "`Reaction`"). The resulting XML data object has the form illustrated by the following skeletal model:

```
<model name="My_Model">
    <listOfUnitDefinitions>
        ...
    </listOfUnitDefinitions>
    <listOfCompartments>
        ...
    </listOfCompartments>
    <listOfSpecies>
        ...
    </listOfSpecies>
    <listOfParameters>
        ...
    </listOfParameters>
    <listOfRules>
        ...
    </listOfRules>
    <listOfReactions>
        ...
    </listOfReactions>
</model>
```

Readers may wonder about the motivations for the `listOf_____`s notation. A simpler approach to creating the lists of components would be to place them all directly at the top level under `<model> ... </model>`. We chose instead to group them within XML elements named after `listOf_____`s, because we believe this helps organize the components and makes visual reading of model definitions easier.

4.2 Unit Definitions

Units may be supplied in a number of contexts in an SBML model. A facility for defining units is convenient to have so that combinations of units can be given abbreviated names. This is the motivation behind the `UnitDefinition` data structure, whose definition is shown in Figure 5.
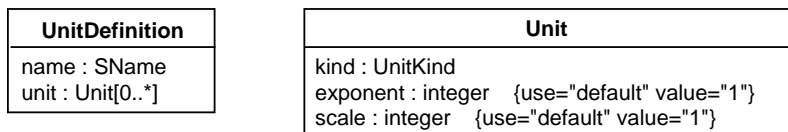
UnitDefinition
name : SName
unit : Unit[0..*]

Unit
kind : UnitKind
exponent : integer   {use="default" value="1"}
scale : integer   {use="default" value="1"}

Figure 5: *The definition of* `UnitDefinition`.

A unit definition consists of a `name` field of type `SName` and an optional list of structures of type `Unit`. The approach to defining units in SBML is compositional; for example, $meter\ second^{-2}$ is constructed by combining a `Unit`-type element representing $meter$ with a `Unit`-type element representing $second^{-2}$. The `Unit` data structure has a `kind` field whose value must be taken from `UnitKind`, an enumeration of SI units. The possible values of `UnitKind` are listed in Table 4 on the next page. The `exponent` field on `Unit` represents an exponent on the unit. Its default value is "1" (one). In the example just mentioned, $second^{-2}$ is obtained by using `kind="second"` and `exponent="-2"`. Finally, the `scale` field in `Unit` is an integer attribute that scales the unit. For example, a unit that has a `kind` value of "gram" and a `scale` value of "-3" signifies $10^{-3} * gram$, or milligrams.

Unit combinations are constructed by listing several `Unit` structures inside a `UnitDefinition`-type structure. The following example illustrates the definition of an abbreviation named "`mmls`" for the units $mmol\ l^{-1}\ s^{-1}$:

```
<listOfUnitDefinitions>
    <unitDefinition name="mmls">
        <listOfUnits>
            <unit kind="mole"   scale="-3"/>
            <unit kind="litre"  exponent="-1"/>
            <unit kind="second" exponent="-1"/>
        </listOfUnits>
    </unitDefinition>
</listOfUnitDefinitions>
```

ampere	farad	joule	lumen	ohm	steradian
becquerel	gram	katal	lux	pascal	tesla
candela	gray	kelvin	meter	radian	volt
celsius	henry	kilogram	metre	second	watt
coulomb	hertz	liter	mole	siemens	weber
dimensionless	item	litre	newton	sievert	

Table 4: *The possible values of* kind *in a* UnitKind *structure. All are names of base or derived SI units, except for "dimensionless" and "item", which are SBML additions important for handling certain common cases. "Dimensionless" is intended for cases where a quantity does not have units, and "item" is needed in certain contexts to express such things as "N items" (e.g., "100 molecules"). Strictly speaking, "celsius" should be capitalized; however, for simplicity, SBML requires that the values of* UnitKind *be treated in a case-insensitive manner by software reading and writing SBML. Also, note that the gram and liter/litre are not strictly part of SI (Taylor, 1995); however, they are so useful in SBML's areas of application that they are included in the* UnitKind *enumeration of unit names. (The standard SI unit of mass is in fact the kilogram, and volume is defined in terms of cubic meters.)*

Many of the components in a model refer to quantities that have associated units. SMBL Level 1 has three predefined quantity types: amount of substance, time, and volume. SBML defines default units and scales for these quantities. The defaults are summarized in Table 5.

Name	Allowable Units	Default Units	Default Scale
substance	moles or no. of molecules	moles	1
time	seconds	seconds	1
volume	liters	liters	1

Table 5: *SBML's built-in quantities and their default scale values. The names in the left-hand column are reserved. These names may be used wherever units may be supplied in a model component.*

Wherever unit specifications are permitted in a model (for example, for the volume in a compartment), the relevant built-in name from Table 5 may be used. Such usage signifies that the units to be used for the quantity should be the designated defaults. A model may change the default scales by reassigning the keywords "substance", "time", and "volume" in a unit definition. This takes advantage of the UnitDefinition structure's facility for defining scales on units. The following example changes the default units of volume to be milliliters:

```
<model>
    ...
    <listOfUnitDefinitions>
        <unitDefinition name="volume">
            <listOfUnits>
                <unit kind="liters" scale="-3"/>
            </listOfUnits>
        </unitDefinition>
    </listOfUnitDefinitions>
    ...
</model>
```

If the definition above appeared in a model, the volume scale on all components that did not explicitly use different units would be changed to milliliters.

The list of unit definitions in a Model-type structure is the only place where new units can be defined. The new unit names may be used anywhere in a model where unit specifications are permitted. The various components of a model, such as reaction parameters and rules, can only use the base units from Table 4, the global unit definitions in the model, or the three predefined keywords "substance", "time", and "volume".

4.3 Compartments

A `Compartment` represents a bounded container in which species are located. The definition of `Compartment` is shown in Figure 6.

<div align="center">

Compartment
name : SName
volume : double   {use="default" value="1"}
units : SName   {use="optional"}
outside : SName   {use="optional"}

</div>

Figure 6: *The definition of* `Compartment`*. Fields inherited from* `SBase` *are omitted here but are assumed.*

A `Compartment` data object has a `name` field of type `SName`. A compartment also has a floating-point field called `volume`, representing the total volume of the compartment in the default units of volume. (See Table 5 on the page before.) This enables concentrations of species to be calculated in the absence of geometry information. The `volume` field is optional and defaults to a value of "1" (one).

The units of volume may be explicitly set using the optional field `units` in `Compartment`; the named units must be either one of the base units from Table 4 on the preceding page, the built-in default named `volume`, or a new unit defined by a unit definition in the enclosing model. If absent, the units default to the value set by the built-in `volume` of Table 5.

The optional field `outside` of type `SName` can be used to express containment relationships between compartments. If present, the value of `outside` for a given compartment should be the name of the compartment enclosing it, or in other words, the compartment that is "outside" of it. This facility can be used to model cell membranes. For example, to express that a compartment named B has a membrane that is modeled as another compartment M, which in turn is located within another compartment A, one would write:

```
<model>
    ...
    <listOfCompartments>
        <compartment name="A"/>
        <compartment name="M" outside="A"/>
        <compartment name="B" outside="M"/>
    </listOfCompartments>
    ...
</model>
```

In the absence of a value for `outside`, compartment definitions in SBML Level 1 do not have any implied spatial relationships between each other. Thus, compartments may be adjacent to each other or have other spatial relationships. For many modeling applications, the transfer of substances described by the reactions in a model sufficiently express the relationships between the compartments. (SBML Level 1 currently does not provide for spatial characteristics aside from compartment volume and containment. As discussed in Section 6.1, we expect that SBML Level 2 will introduce the ability to define geometries and spatial qualities.)

In an XML data stream containing an SBML model, compartments are listed inside an XML element called `listOfCompartments` within a `Model`-type data structure. (See the discussion of `Model` in Section 4.1.) The following example illustrates two compartments in an abbreviated SBML example of a model definition:

```
<model>
    ...
    <listOfCompartments>
        <compartment name="cytosol" volume="2.5"/>
        <compartment name="mitochondria" volume="0.3"/>
    </listOfCompartments>
    ...
</model>
```

4.4   Species

The term *species* refers to entities that take part in reactions. These include simple ions (e.g., protons, calcium), simple molecules (e.g., glucose, ATP), and large molecules (e.g., RNA, polysaccharides, and proteins). The `Specie` data structure is intended to represent these entities. Its definition is shown in Figure 7.

Specie
name : SName compartment : SName initialAmount : double units : SName   {use="optional"} boundaryCondition : boolean   {use="default" value="false"} charge : integer   {use="optional"}

Figure 7: *The definition of `Specie`. As usual, fields inherited from `SBase` are omitted here but are assumed.*

`Specie` has a `name` field of type `SName`. The field `compartment`, also of type `SName`, is used to identify the compartment in which the specie is located. The field `initialAmount`, of type `double`, is used to set the initial amount of the specie in the named compartment. The units of the substance quantity may be explicitly set using the optional field `units`. The value assigned to `units` must be chosen from one of the following possibilities: one of base unit names from Table 4 on page 10, the name "`volume`", or a new unit name defined by a unit definition in the enclosing model. If absent, the units default to the value set by the built-in "`substance`" of Table 5 on page 10.

The optional boolean field `boundaryCondition` determines whether the amount of the specie is fixed or variable over the course of a simulation. `boundaryCondition` defaults to a value of "`false`", indicating that by default, the amount is not fixed. The optional field `charge` is an integer indicating the charge on the species (in terms of electrons, not the SI unit Coulombs). This may be useful when the specie involved is a charged ion such as calcium (Ca^{2++}).

The following example shows two specie definitions within an abbreviated SBML model definition. The example shows that species are listed under the heading `listOfSpecies` in the model:

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

Note that the compartment's name is used instead of an identifier, because simulation systems typically enable users to locate species using compartment names rather than machine-generated identifiers. Since a compartment's name must be unique among all the compartments in a model (see the discussion of namespaces in Section 3.4), there is no danger of ambiguity and no compelling reason to introduce identifiers on compartments.

4.5   Parameters

A `Parameter` structure is used to associate a symbol with a floating-point value so that the symbol can be used in formulas in place of the value. The definition of `Parameter` is shown in Figure 8 on the following page.

`Parameter` has a `name` field of type `SName`. This name represents the symbol. The field `value` determines the value (of type `double`) assigned to the symbol. The units of the parameter `value` are specified by the field `units`. The value assigned to `units` must be chosen from one of the following possibilities: one of base unit names from Table 4 on page 10; one of the three names "`substance`", "`time`", or "`volume`" (see Table 5); or the name of a new unit defined in the list of unit definitions in the enclosing `Model` structure.

Parameter
name : SName
value : double
units : SName   {use="optional"}

Figure 8: *The definition of* `Parameter`*.*

Parameters are used in two places in SBML: in lists of parameters defined at the top level in a `Model`-type structure, and within individual reaction definitions. Parameters defined at the top level are *global* to the whole model; parameters that are defined within a reaction are local to the particular reaction and (within that reaction) *override* any global parameters having the same names. (See Section 3.4 for further details.)

The following is an example of parameters defined at the `Model` level:

```
<model>
    ...
    <listOfSpecies>
        ...
    </listOfSpecies>
    <listOfParameters>
        <parameter name="Km1" value="2.3" units="second"/>
        <parameter name="Km2" value="10.7" units="second"/>
    </listOfParameters>
    <listOfReactions>
        ...
    </listOfReactions>
    ...
</model>
```

An example of a full model that uses parameters is presented in Section 5.3.

4.6   Rules

In SBML, *rules* provide a way to create constraints on variables for cases in which the constraints cannot be expressed using the reaction components (Section 4.7). There are two orthogonal dimensions by which rules can be described. First, there are three different possible functional forms, corresponding to the following three general cases (where x and y are variables, f is some arbitrary function, and P is a vector of parameters):

Case 1, left-hand side is zero: $0 = f(x, P)$
Case 2, left-hand side is a scalar: $x = f(P)$
Case 3, left-hand side is a rate-of-change: $dx/dt = f(x, P)$

The second dimension concerns the role of the variable x in the equations above: x can be the name of a compartment (to set its volume), the name of a specie (to set its concentration), or a parameter name (to set its value).

The approach taken to covering these cases in SBML is to define an abstract `Rule` structure that contains just one field, `formula`, to hold the right-hand side expression, then to derive subtypes of `Rule` that add fields to cover the various cases above. Figure 9 on the following page gives the definitions of `Rule` and the subtypes derived from it. The figure shows that `AlgebraicRule` is defined directly from `Rule`, whereas `CompartmentVolumeRule`, `SpecieConcentrationRule`, and `ParameterRule` are all derived from an intermediate abstract structure called `AssignmentRule`.

The `type` field introduced in `AssignmentRule` is an enumeration of type `RuleType` that determines whether a rule falls into category 2 or 3 in the list of cases above. In SBML Level 1, the enumeration has two possible values: "`scalar`" and "`rate`". The former means that the expression has a scalar value on the left-hand side [i.e., $x = f(x, P)$, as in case 2 in the list above]; the latter means that the expression has a rate of change differential on the left-hand side (as in case 3 in the list above). Future releases of SBML may add to the possible values of `RuleType`.
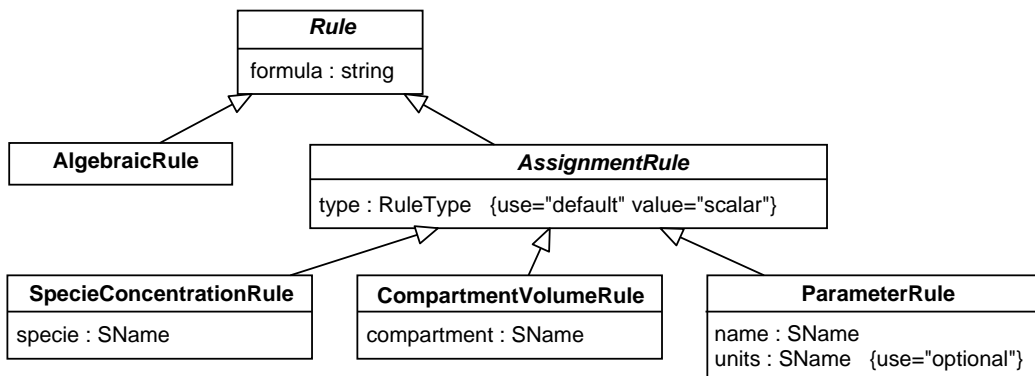
Figure 9: *The definition of `Rule` and derived types.*

4.6.1 `AlgebraicRule`

The rule type `AlgebraicRule` is used to express equations whose left-hand sides are zero. `AlgebraicRule` does not add any fields to the basic `Rule`; its role is simply to distinguish this case from the other cases.

4.6.2 `SpeciesConcentrationRule`

The `SpeciesConcentrationRule` structure adds one field, `specie`, to the basic `AssignmentRule` type. The field `specie` has type `SName` and is used to identify the specie affected by the rule. The effect of the rule depends on the value of `type`: if the value is "`scalar`", the rule sets the referenced specie's concentration to the value determined by the formula; if the value is "`rate`", the rule sets the rate of change of the specie's concentration to the value determined by the formula. The units are in terms of *substance/volume*, where the *substance* units are those that are declared on the referenced `Specie` element, and the *volume* units are those declared on the `compartment` element that contains the `Specie`.

4.6.3 `CompartmentVolumeRule`

The `CompartmentRule` structure adds one field, `compartment`, to the basic `AssignmentRule` type. The field `compartment` has type `SName` and is used to identify the compartment affected by the assignment. The effect of the rule depends on the value of `type`: if the type is "`scalar`", the rule sets the referenced compartment's volume to the volume determined by the formula; if the type is "`rate`", the rule sets the rate of change of the compartment's volume to the volume determined by the formula.

4.6.4 `ParameterRule`

The `ParameterRule` structure adds two fields, `name` and `units`, to the basic `AssignmentRule` type. The `name` attribute has type `SName` and identifies the parameter. The `units` field acts in the same way as in the case of the `Parameter` structure (Section 4.5). The value assigned to `units` must be chosen from one of the following possibilities: one of base unit names from table 4; one of the three names "`substance`", "`time`", or "`volume`" (see Table 5); or the name of a new unit defined in the list of unit definitions in the enclosing `model` structure. The effect of this rule is to give a value to a parameter that can be used in subsequent formulas. This parameter has the value returned by the expression in the `formula` attribute.

4.6.5 Guidelines for Evaluating Rules

Rules of type "`scalar`" should be evaluated first in the order given in the SBML model definition, followed by the rules of type "`rate`" in the order given. Following this, the "`rate`" type rules can be evaluated any number of times before the "`scalar`" type rules are re-evaluated.

4.6.6 Example of Rule Use

The following is an example use of rules:

```
<model>
    ...
    <listOfRules>
        <parameterRule name="k" formula="k3/k2"/>
        <specieConcentrationRule specie="s2" formula="k * t/(1 + k)"/>
        <compartmentVolumeRule compartment="A" formula="0.10 * t"/>
    </listOfRules>
    ...
</model>
```
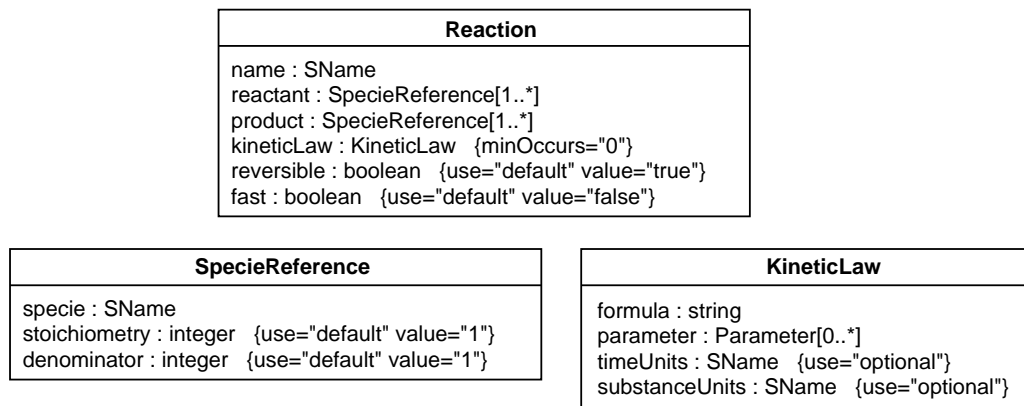
4.7 Reactions

A *reaction* represents some transformation, transport or binding process, typically a chemical reaction, that can change the amount of one or more species. The `Reaction` type is defined in Figure 10.

Reaction
name : SName
reactant : SpecieReference[1..*]
product : SpecieReference[1..*]
kineticLaw : KineticLaw   {minOccurs="0"}
reversible : boolean   {use="default" value="true"}
fast : boolean   {use="default" value="false"}

SpecieReference
specie : SName
stoichiometry : integer   {use="default" value="1"}
denominator : integer   {use="default" value="1"}

KineticLaw
formula : string
parameter : Parameter[0..*]
timeUnits : SName   {use="optional"}
substanceUnits : SName   {use="optional"}

Figure 10: *The definitions of Reaction, KineticLaw and SpecieReference.*

In SBML, reactions are defined using lists of reactant species, products, and their stoichiometries, and by parameter values for separately-defined kinetic laws. These various quantities are recorded in the fields `reactant`, `product`, and `kineticLaw`. Both `reactant` and `product` are references to species implemented using lists of `SpecieReference` structures (defined in Section 4.7.1 below). The `SpecieReference` structure contains fields for recording the names of species and their stoichiometries. `kineticLaw` is an optional field of type `KineticLaw` (defined in Section 4.7.2 below), used to provide a mathematical formula describing the rate of the reaction.

In addition to these fields, the `Reaction` structure also has a boolean field, `reversible`, that indicates whether the reaction is reversible. The field is optional, and if left unspecified in a model, it defaults to a value of "`true`". Information about reversibility is useful in certain kinds of structural analyses such as elementary mode analysis.

The field `fast` is another boolean attribute in the `Reaction` data structure; a value of "`true`" signifies that the given reaction is a "fast" one. This may be relevant when computing equilibrium concentrations of rapidly equilibrating reactions. Simulation/analysis packages may chose to use this information to reduce the number of ODEs required and thereby optimize such computations. The default value of `fast` is "`false`". (A simulator/analysis package that has no facilities for dealing with fast reactions can ignore this attribute. In theory, if the choice of which reactions are fast is correctly made, then a simulation performed with them should give the same results as a simulation performed without fast reactions. However, currently there appears to be no single unambiguous method for designating which reactions should be considered fast, and some users may designate a reaction as fast when in fact it is not. Caveat developer.)

4.7.1 `SpecieReference`

Each unique specie involved in a reaction is listed once in a model, in a list contained in the `specie` field of the `Model` data structure discussed in Section 4.1. Lists of products and reactants in `Reaction` type structures refer to those species. The connection between the products and reactants in a reaction definition and the specie names listed in the enclosing `Model` definition is achieved using the `SpecieReference` type data structure defined in Figure 10 on the preceding page.

The field `specie` of type `SName` in `SpecieReference` must refer to the name of a specie defined in the enclosing `Model`-type structure. The two fields `stoichiometry` and `denominator` together set the stoichiometry value for a specie in a reaction. Both are integers, and both have default values of "1" (one). The use of these separate terms allows a simulator to employ rational arithmetic computations on the stoichiometry matrix, potentially reducing round-off errors and other problems during computations. Such computations are particularly important when working with large matrices and calculating such things as elementary modes.

The following is a simple example of a specie reference in a list of reactants within a reaction named "J1":

```
<model>
    ...
    <listOfReactions>
        <reaction name="J1">
            <listOfReactants>
                <specieReference specie="X0" stoichiometry="2"/>
            </listOfReactants>
            ...
        </reaction>
        ...
    </listOfReactions>
    ...
</model>
```

4.7.2 `KineticLaw`

A `kineticLaw` structure describes the rate of the enclosing reaction. The use of a `KineticLaw` structure in a `Reaction` component is optional; however, in general there is no useful default that can be substituted in place of a missing kinetic law definition in a reaction.

The field `formula`, of type `string`, expresses the rate in *substance/time* units. (Section 3.5 discusses formulas.) The optional fields `substanceUnits` and `timeUnits` determine the units of substance and time. If not set, the units are taken from the defaults defined by the built-in "`substance`" and "`time`" of Table 5 on page 10.

A `KineticLaw` type structure can contain zero or more `Parameter` structures (Section 4.5) that define symbols that can be used in the `formula` string. As discussed in Section 3.4, reactions introduce local namespaces for parameter names. Within a `KineticLaw` structure inside a reaction definition, a parameter whose name is identical to a global parameter defined in the enclosing `Model`-type structure takes precedence over that global parameter.

The following is an example of a `Reaction` structure that defines the reaction $J_1 : X_0 \longrightarrow S_1; k_1 X_0$. It demonstrates the use of specie references and the `KineticLaw` structure:

```
<model>
    ...
    <listOfReactions>
        <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>
            </listOfReactions>
            ...
        </model>
```

5 Examples of Full Models Encoded in XML Using SBML

In this section, we present several examples of complete models encoded in XML using SBML Level 1. Our approach to translating the UML-based structure definitions presented in the previous sections is described elsewhere (Hucka, 2000). Appendix B gives the full listing of an XML Schema corresponding to SBML Level 1.

5.1 A Simple Example Application of SBML

Consider the following hypothetical branched system:

$$X_0 \quad \underset{\longrightarrow}{k_1 X_0} \quad S_1$$
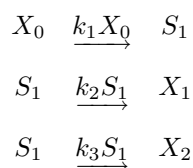
$$S_1 \quad \underset{\longrightarrow}{k_2 S_1} \quad X_1$$

$$S_1 \quad \underset{\longrightarrow}{k_3 S_1} \quad X_2$$

The following is the main portion of an XML document that encodes the model shown above:

```
<sbml level="1" version="1">
    <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*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 XML encoding shown above is quite straightforward. The outermost container is a tag, `<smbl>`, that identifies the contents as being <u>S</u>ystems <u>B</u>iology <u>M</u>arkup <u>L</u>anguage. The attributes `level` and `version` indicate that the content is formatted according to version 1 of the Level 1 definition of SBML. The `version` attribute is present in case SBML Level 1 must be revised in the future to correct errors.

The next-inner container is a single `<model>` element that serves as the highest-level object in the model. The model has a name, "Branch". The model contains one compartment, four species, and three reactions. The elements in the `<listOfReactants>` and `<listOfProducts>` in each reaction refer to the names of elements listed in the `<listOfSpecies>`. The correspondences between the various elements is explicitly stated by the `<specieReference>` elements.

The model includes a `<notes>` annotation that summarizes the model in text form, with formatting based on XHTML. This may be useful for a software package that is able to read such annotations and, for example, render them in HTML in a graphical user interface.

5.2   Simple Use of Units Feature in a Model

The following model uses the units features of SBML Level 1. In this model, the default value of `substance` is changed in the list of unit definitions to be mole units with a scale factor of -3, or millimoles. This sets the default substance units in the model; components can override this scale locally. The `volume` and `time` built-ins are left to their defaults, ensuring that volume is in liters and time is in seconds. The result is that, in this model, kinetic law formulas define rates in millimoles per second and the specie symbols in them represent concentration values in millimoles per liter. All the `specie` elements set the initial amount of every given specie to 1 millimole. The parameters `Vm` and `Km` are defined to be in millimoles per liter per second, and milliMolar, respectively.

```
        <sbml level="1" version="1">
            <model>
                <listOfUnitDefinitions>
                    <unitDefinition name="substance">
                        <listOfUnits>
                            <unit kind="mole" scale="-3"/>
                        </listOfUnits>
```

```
                </unitDefinition>
                <unitDefinition name="mls">
                    <listOfUnits>
                        <unit kind="mole"   scale="-3"/>
                        <unit kind="liter"  exponent="-1"/>
                        <unit kind="second" exponent="-1"/>
                    </listOfUnits>
                </unitDefinition>
            </listOfUnitDefinitions>
            <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" units="mls"/>
                <parameter name="km" value="2"/>
            </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>
```

5.3   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 level="1" 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 name="t" formula="s1 + s2"/>
        <parameterRule name="k" formula="k3/k2"/>
        <specieConcentrationRule specie="s2" formula="k * t/(1 + k)"/>
        <specieConcentrationRule specie="s1" formula="t - s2"/>
    </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>
```

6   Discussion

The volume of data now emerging from molecular biotechnology leave little doubt that extensive computer-based modeling, simulation and analysis will be critical to understanding and interpreting the data (Abbott, 1999; Gilman, 2000; Popel and Winslow, 1998; Smaglik, 2000a). This has lead to an explosion in the development of computer tools by many research groups across the world. The explosive rate of progress is exciting, but the rapid growth of the field is accompanied by problems and pressing needs.

One problem is that simulation models and results often cannot be directly compared, shared or re-used, because the tools developed by different groups often are not compatible with each other. As the field of systems biology matures, researchers increasingly need to communicate their results as computational models rather than box-and-arrow diagrams. They also need to reuse published and curated models as library components in order to succeed with large-scale efforts (e.g., the Alliance for Cellular Signaling; Gilman, 2000; Smaglik, 2000a). These needs require that models implemented in one software package be portable to other software packages, to maximize public understanding and to allow building up libraries of curated computational models.

We offer SBML to the systems biology community as a suggested format for exchanging models between simulation/analysis tools. SBML is an open model representation language oriented specifically towards representing biochemical network models. SBML Level 1 provides the most basic facilities that are necessary for expressing these kinds of models in terms of compartments, species, reactions, parameters, rules and units.

Our vision for SBML is to create an open standard that will enable simulation software to exchange models. SBML is not static; we continue to develop and experiment with it, and we interact with other groups who seek to develop similar markup languages. We plan on continuing to evolve SBML with the help of the systems biology community to make SBML increasingly more powerful, flexible and useful.

6.1 Future Enhancements to SBML: Level 2 and Beyond

As mentioned above, SBML Level 1 is intended to provide the most basic foundations for modeling biochemical networks. A number of significant capabilities are lacking from Level 1; these will be introduced in higher-level definitions of SBML. The following summarizes additional features that will likely be included in SBML Level 2:

- *Arrays*. This will enable the creation of arrays of components (species, reactions, compartments and submodels).

- *Connections*. This will be a mechanism for describing the connections between items in an array. For example, it should be possible to create a 2-D array of compartments and then a 3-D array of reactions which transport species between the compartments, where the third dimension is the connections between the compartments. Two possible ways of describing a connection scheme are: (1) sparse/explicit, simply listing the relative co-coordinates of connected objects for patterns of points; (2) algebraic, where a conditional equation describes whether two objects are connected.

- *Database Interoperability*. In order to store models in a database, it will be necessary to add additional header information that provides information about authors, version numbers, revision dates, etc.

- *Geometry*. We will develop a scheme for representing the 3-D structure of compartments.

- *Submodels*. This will enable a large model to be built up out of instances of other models. It will also allow the reuse of model components and the creation of several instances of the same model.

- *Component Identification*. This will enable components to be described using some stable universal identification scheme.

- *References*. This will enable literature/authors to be cited for any component.

- *Diagrams*. This feature will allow components to be annotated with data to enable the display of the model in a diagram. It will also enable multistate representations.

6.2 Relationships to Other Efforts

There are a number of ongoing efforts with similar goals as those of SBML. Many of them are oriented more specifically toward describing protein sequences, genes and related entities for database storage and search. These are generally not intended to be computational models, in the sense that they do not describe entities and behavioral rules in such a way that a simulation package could "run" the models.

The effort perhaps closest in spirit to SBML is CellML™ (Physiome Sciences, 2001). CellML is an XML-based markup language designed for storing and exchanging computer-based biological models. It includes facilities for representing model structure, mathematics and additional information for database storage and search. Models are described in terms of networks of connections between discrete components, where a component is a functional unit that may correspond to a physical compartment or simply a convenient modeling abstraction. Components contain variables and connections contain mappings between the variables of connected components. CellML provides facilities for grouping components and specifying the kinds of relationships that may exist between components. It also uses MathML (W3C, 2000b) for expressing mathematical relationships between components and provides the ability to use ECMAScript (formerly known as JavaScript) to define functions.

The constructs in CellML tend to be at a more abstract and general level than those in SBML Level 1, and describes the structure and underlying mathematics of cellular models in a very general way. By contrast, SBML is closer to the internal object model used in model analysis software. Because SBML Level 1 is being developed in the context of interacting with a number of existing simulation packages, it is a more concrete language than CellML and may be better suited to its purpose of enabling interoperability with existing simulation tools. However, CellML offers viable alternative ideas and the developers of SBML and CellML are actively engaged in ensuring that the two representations can be translated between each other.

6.3 Tracking the XML Schema Standard

One of the problems in attempting to define an XML Schema for SBML is that, at the time of this writing, the XML Schema specification (Biron and Malhotra, 2000; Thompson et al., 2000) has not actually been finalized. This has been another motivation for defining SBML in terms of abstract data structures in a UML-based notation rather than directly as an XML Schema.

The moving-target status of the XML Schema standard definition requires that we plan to update the Schema corresponding to SBML. The following is our planned approach for handling changes in the Schema standard:

1. The definition of SBML Level 1 in this document is independent of XML Schema. Therefore, the definition of SBML Level 1 expressed here can remain the same regardless of what happens to the exact form of XML Schema. Among other benefits, this allows developers to leave their programs' internal data structures unchanged in the face of possible revisions in the Schema standard.

2. In Appendix B, we provide an XML Schema corresponding to SBML Level 1 that has been created using the current definition of XML Schema from the W3C Organization (Biron and Malhotra, 2000; Thompson et al., 2000).

3. Whenever the definition of XML Schema is updated by the W3C in the future, we will issue a revised version of the XML Schema for SBML Level 1 that conform to the updated standard. We will leave the previous versions still available for reference. The updated XML Schemas for SBML Level 1 will be identical to the previous versions except where changes in XML Schema force a change in the definition of the Schema for SBML Level 1.

6.4 Availability

The SBML Level 1 definition, the XML Schema corresponding to SBML Level 1, and other related documents are openly available from the Caltech ERATO web site, `http://www.cds.caltech.edu/erato/`.

Acknowledgments

Appendix

A   Summary of Notation

The definitive explanation for the notation used in this document can be found in the companion notation document (Hucka, 2000). 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 are used many times:

```
field1 : float
field2 : integer[0..*]
field3 : (XHTML)
field4 : float {use = "default" value = "0.0"}
```

The symbols `field1`, `field2`, etc., represents fields in a data structure. The colon immediately after the name separates the name of the attribute from the type of data that it stores.

More complex specifications use square brackets (`[]`) just after a type name. This is used to indicate that the field 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. The approach used here to translate from a list form into XML is, first, create a subelement named `listOf_____s`, where the blank indicates the capitalized name of the field, and then put a list of elements named after the field as the content of the `listOf_____s` element.

A field whose type is shown in parentheses is implemented as as an XML subelement rather than an XML attribute. The parentheses indicate that the type refers to the type of the subelement value.

Expressions in curly braces (`{}`) shown after an attribute type indicate additional constraints placed on the field. We express constraints using XML Schema language. In the examples above, the expression `{use="default" value="0.0"}` indicates that the field `field4` is optional and that it has a default value of 0.0.

B   XML Schema for SBML

The following is an XML Schema definition for the Systems Biology Markup Language. Example applications of this XML Schema are presented in Section 5.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">
  <xsd:annotation>
    <xsd:documentation>
      File name : sbml.xsd
      Author : Daniel Lucio, James Schaff, Andrew Finney, Michael Hucka
      Description : XML Schema for the Systems Biology Markup Language
      Organization: Caltech ERATO Kitano
      Version : 1
      Modified : 2001-03-02 22:29 PST
    </xsd:documentation>
  </xsd:annotation>
  <!-- SName -->
  <xsd:simpleType name="SName">
    <xsd:annotation>
      <xsd:documentation>
         SName type used for expressing
         names of components in a model
      </xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="(_|[a-z]|[A-Z])(_|[a-z]|[A-Z]|[0-9])*"/>
    </xsd:restriction>
  </xsd:simpleType>
```

```
<!-- SBase -->
<xsd:complexType name="SBase" abstract="true">
  <xsd:annotation>
    <xsd:documentation>
      Designed to allow a modeler or a pacckage to attach
      information to each component.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element name="notes" minOccurs="0">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:any namespace="http://www.w3.org/1999/xhtml"
                   processContents="skip" maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="annotations" minOccurs="0">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:any processContents="skip" maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
<!-- LisOfParameter  -->
<xsd:element name="listOfParameters">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="parameter" type="Parameter"
                   maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<!-- Model -->
<xsd:complexType name="Model">
  <xsd:complexContent>
    <xsd:extension base="SBase">
      <xsd:sequence>
        <xsd:element name="listOfUnitDefinitions" minOccurs="0">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="unitDefinition" type="UnitDefinition"
                           maxOccurs="unbounded"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="listOfCompartments">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="compartment" type="Compartment" maxOccurs="unbounded"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="listOfSpecies">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="specie" type="Specie" maxOccurs="unbounded"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
        <xsd:element ref="listOfParameters" minOccurs="0"/>
        <xsd:element name="listOfRules" minOccurs="0">
          <xsd:complexType>
            <xsd:choice maxOccurs="unbounded">
              <xsd:element name="algebraicRule" type="AlgebraicRule" minOccurs="0"/>
              <xsd:element name="compartmentVolumeRule" type="CompartmentVolumeRule"
                           minOccurs="0"/>
```

```xsd
                <xsd:element name="specieConcentrationRule" type="SpecieConcentrationRule"
                             minOccurs="0"/>
                <xsd:element name="parameterRule" type="ParameterRule" minOccurs="0"/>
              </xsd:choice>
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="listOfReactions">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="reaction" type="Reaction" maxOccurs="unbounded"/>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
        </xsd:sequence>
        <xsd:attribute name="name" type="SName" use="optional"/>
      </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
<!-- UnitDefinition -->
<xsd:complexType name="UnitDefinition">
  <xsd:complexContent>
    <xsd:extension base="SBase">
      <xsd:sequence>
        <xsd:element name="listOfUnits" minOccurs="0">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="unit" type="Unit" maxOccurs="unbounded"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
      <xsd:attribute name="name" type="SName" use="required"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<!-- Compartment -->
<xsd:complexType name="Compartment">
  <xsd:complexContent>
    <xsd:extension base="SBase">
      <xsd:attribute name="name" type="SName" use="required"/>
      <xsd:attribute name="volume" type="xsd:double" use="default" value="1"/>
      <xsd:attribute name="units" type="SName" use="optional"/>
      <xsd:attribute name="outside" type="SName" use="optional"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<!-- Specie -->
<xsd:complexType name="Specie">
  <xsd:complexContent>
    <xsd:extension base="SBase">
      <xsd:attribute name="name" type="SName" use="required"/>
      <xsd:attribute name="compartment" type="SName" use="required"/>
      <xsd:attribute name="initialAmount" type="xsd:double" use="required"/>
      <xsd:attribute name="units" type="SName" use="optional"/>
      <xsd:attribute name="boundaryCondition" type="xsd:boolean"
                     use="default" value="false"/>
      <xsd:attribute name="charge" type="xsd:integer" use="optional"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<!-- Parameter -->
<xsd:complexType name="Parameter">
  <xsd:complexContent>
    <xsd:extension base="SBase">
      <xsd:attribute name="name" type="SName" use="required"/>
      <xsd:attribute name="value" type="xsd:double" use="required"/>
      <xsd:attribute name="units" type="SName" use="optional"/>
    </xsd:extension>
  </xsd:complexContent>
```

```xml
    </xsd:complexType>
    <!-- UnitKind -->
    <xsd:simpleType name="UnitKind">
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="ampere"/>
        <xsd:enumeration value="becquerel"/>
        <xsd:enumeration value="candela"/>
        <xsd:enumeration value="celsius"/>
        <xsd:enumeration value="coulomb"/>
        <xsd:enumeration value="dimensionless"/>
        <xsd:enumeration value="farad"/>
        <xsd:enumeration value="gram"/>
        <xsd:enumeration value="gray"/>
        <xsd:enumeration value="henry"/>
        <xsd:enumeration value="hertz"/>
        <xsd:enumeration value="item"/>
        <xsd:enumeration value="joule"/>
        <xsd:enumeration value="katal"/>
        <xsd:enumeration value="kelvin"/>
        <xsd:enumeration value="kilogram"/>
        <xsd:enumeration value="liter"/>
        <xsd:enumeration value="litre"/>
        <xsd:enumeration value="lumen"/>
        <xsd:enumeration value="lux"/>
        <xsd:enumeration value="meter"/>
        <xsd:enumeration value="metre"/>
        <xsd:enumeration value="mole"/>
        <xsd:enumeration value="newton"/>
        <xsd:enumeration value="ohm"/>
        <xsd:enumeration value="pascal"/>
        <xsd:enumeration value="radian"/>
        <xsd:enumeration value="second"/>
        <xsd:enumeration value="siemens"/>
        <xsd:enumeration value="sievert"/>
        <xsd:enumeration value="steradian"/>
        <xsd:enumeration value="tesla"/>
        <xsd:enumeration value="volt"/>
        <xsd:enumeration value="watt"/>
        <xsd:enumeration value="weber"/>
      </xsd:restriction>
    </xsd:simpleType>
    <!-- Unit -->
    <xsd:complexType name="Unit">
      <xsd:complexContent>
        <xsd:extension base="SBase">
          <xsd:attribute name="kind" type="UnitKind" use="required"/>
          <xsd:attribute name="exponent" type="xsd:integer" use="default" value="1"/>
          <xsd:attribute name="scale" type="xsd:integer" use="default" value="1"/>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
    <!-- Rule -->
    <xsd:simpleType name="RuleType">
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="scalar"/>
        <xsd:enumeration value="rate"/>
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType name="Rule" abstract="true">
      <xsd:complexContent>
        <xsd:extension base="SBase">
          <xsd:attribute name="formula" type="xsd:string" use="required"/>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="AlgebraicRule">
      <xsd:complexContent>
        <xsd:extension base="Rule"/>
      </xsd:complexContent>
```

```xsd
    </xsd:complexType>
    <xsd:complexType name="AssignmentRule" abstract="true">
      <xsd:complexContent>
        <xsd:extension base="Rule">
          <xsd:attribute name="type" type="RuleType" use="default" value="scalar"/>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="CompartmentVolumeRule">
      <xsd:complexContent>
        <xsd:extension base="AssignmentRule">
          <xsd:attribute name="compartment" type="SName" use="required"/>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="SpecieConcentrationRule">
      <xsd:complexContent>
        <xsd:extension base="AssignmentRule">
          <xsd:attribute name="specie" type="SName" use="required"/>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="ParameterRule">
      <xsd:complexContent>
        <xsd:extension base="AssignmentRule">
          <xsd:attribute name="name" type="SName" use="required"/>
          <xsd:attribute name="units" type="SName" use="optional"/>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
    <!-- Reaction -->
    <xsd:element name="specieReference">
      <xsd:complexType>
        <xsd:complexContent>
          <xsd:extension base="SBase">
            <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:extension>
        </xsd:complexContent>
      </xsd:complexType>
    </xsd:element>
    <xsd:complexType name="Reaction">
      <xsd:complexContent>
        <xsd:extension base="SBase">
          <xsd:sequence>
            <xsd:element name="listOfReactants">
              <xsd:complexType>
                <xsd:sequence>
                  <xsd:element ref="specieReference" maxOccurs="unbounded"/>
                </xsd:sequence>
              </xsd:complexType>
            </xsd:element>
            <xsd:element name="listOfProducts">
              <xsd:complexType>
                <xsd:sequence>
                  <xsd:element ref="specieReference" maxOccurs="unbounded"/>
                </xsd:sequence>
              </xsd:complexType>
            </xsd:element>
            <xsd:element name="kineticLaw" type="KineticLaw" minOccurs="0"/>
          </xsd:sequence>
          <xsd:attribute name="name" type="SName" use="required"/>
          <xsd:attribute name="reversible" type="xsd:boolean" use="default" value="true"/>
          <xsd:attribute name="fast" type="xsd:boolean" use="default" value="false"/>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="KineticLaw">
```

```xml
  <xsd:complexContent>
    <xsd:extension base="SBase">
      <xsd:sequence>
        <xsd:element ref="listOfParameters" minOccurs="0"/>
      </xsd:sequence>
      <xsd:attribute name="formula" type="xsd:string" use="required"/>
      <xsd:attribute name="timeUnits" type="SName" use="optional"/>
      <xsd:attribute name="substanceUnits" type="SName" use="optional"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<!-- Top-level elements allowed in an SBML document. -->
<xsd:complexType name="sbmlDocument">
  <xsd:sequence>
    <xsd:element name="model" type="Model"/>
  </xsd:sequence>
  <xsd:attribute name="xmlns"/>
  <xsd:attribute name="level" type="xsd:positiveInteger" use="required"/>
  <xsd:attribute name="version" type="xsd:positiveInteger" use="required"/>
</xsd:complexType>
<xsd:element name="sbml" type="sbmlDocument"/>
<!-- The end. -->
</xsd:schema>
```

C Predefined Functions in SBML

Table 6 lists the basic mathematical functions that are defined in SBML Level 1 at this time.

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	$-1.0 \leq x \leq 1.0$	$-\pi/2 \leq asin(x) \leq \pi/2$
atan	x	arc tangent of x in radians		$-\pi/2 \leq atan(x) \leq \pi/2$
ceil	x	smallest number not less than x whose value is an exact integer		
cos	x	cosine of x		
exp	x	e^x, where e is the base of the natural logarithm		
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	$x \neq n\frac{\pi}{2}$, for odd integer n	

Table 6: *Basic mathematical functions defined in SBML.*

Table 7 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.

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, V_m, K_m	Irreversible Simple Michaelis-Menten	$v = \dfrac{V_m S}{K_m + S}$
uur	S, P, V_f, V_r, K_{ms}, K_{mp}	Uni-Uni Reversible Simple Michaelis-Menten	$v = \dfrac{V_f \frac{S}{K_{ms}} - V_r \frac{P}{K_{mp}}}{1 + \frac{S}{K_{ms}} + \frac{P}{K_{mp}}}$
uuhr	S, P, V_f, K_{m1}, K_{m2}, K_{eq}	Uni-Uni Reversible Simple Michaelis-Menten with Haldane adjustment	$v = \dfrac{\frac{V_f}{K_{m1}} \left(S - \frac{P}{K_{eq}} \right)}{1 + \frac{S}{K_{m1}} + \frac{P}{K_{m2}}}$
isouur	S, P, V_f, K_{ms}, K_{mp}, K_{ii}, K_{eq}	Iso Uni-Uni	$v = \dfrac{V_f \left(S - \frac{P}{K_{eq}} \right)}{S \left(1 + \frac{P}{K_{ii}} \right) + K_{ms} \left(1 + \frac{P}{K_{mp}} \right)}$
hilli	S, V, $S_{0.5}$, h	Hill Kinetics	$v = \dfrac{V S^h}{S_{0.5}^h + S^h}$
hillr	S, P, V_f, $S_{0.5}$, $P_{0.5}$, h, K_{eq}	Reversible Hill Kinetics	$v = \dfrac{\left(V_f \frac{S}{S_{0.5}} \right) \left(1 - \frac{P}{S K_{eq}} \right) \left(\frac{S}{S_{0.5}} + \frac{P}{P_{0.5}} \right)^{h-1}}{1 + \left(\frac{S}{S_{0.5}} + \frac{P}{P_{0.5}} \right)^h}$
hillmr	S, M, P, V_f, K_{eq}, k, h, α	Reversible Hill Kinetics with One Modifier	$v = \dfrac{\left(V_f \frac{S}{S_{0.5}} \right) \left(1 - \frac{P}{S K_{eq}} \right) \left(\frac{S}{S_{0.5}} + \frac{P}{P_{0.5}} \right)^{h-1}}{K_1 + K_2}$ where $K_1 = \left(\dfrac{S}{S_{0.5}} + \dfrac{P}{P_{0.5}} \right)^h, \quad K_2 = \dfrac{1 + \left(\frac{M}{M_{0.5}} \right)^h}{1 + \alpha \left(\frac{M}{M_{0.5}} \right)^h}$
hillmmr	S, P, M, V_f, K_{eq}, k, h, a, b, α_1, α_2, α_{12}	Reversible Hill Kinetics with Two Modifiers	$v = \dfrac{\left(V_f \frac{S}{S_{0.5}} \right) \left(1 - \frac{P}{S K_{eq}} \right) \left(\frac{S}{S_{0.5}} + \frac{P}{P_{0.5}} \right)^{h-1}}{K_1 + K_2}$ where $$K_1 = \left(\frac{S}{S_{0.5}} + \frac{P}{P_{0.5}} \right)^h,$$ $$K_2 = \frac{1 + \left(\frac{Ma}{Ma_{0.5}} \right)^h + \left(\frac{Mb}{Mb_{0.5}} \right)^h}{\left[1 + \alpha_1 \left(\frac{Ma}{Ma_{0.5}} \right)^h + \alpha_2 \left(\frac{Mb}{Mb_{0.5}} \right)^h \right.}$$ $$\left. + \alpha_1 \alpha_2 \alpha_{12} \left(\frac{Ma}{Ma_{0.5}} \right)^h \left(\frac{Mb}{Mb_{0.5}} \right)^h \right]$$

Table 7: *Table of rate law functions in SBML. In all cases, $K_m > 0$, $V_x \geq 0$, $S \geq 0$ and $P \geq 0$.*

Name	Arguments	Meaning	Formula
usii	$S, V, K_m,$ K_i	Substrate Inhibition Kinetics (Irreversible)	$v = V \dfrac{S/K_m}{1 + S/K_m + S^2/K_i}$
usir	$S, P, V_f,$ $V_r, K_{ms},$ K_{mp}, K_i	Substrate Inhibition Kinetics (Reversible)	$v = \dfrac{V_f S/K_{ms} + V_r P/K_{mp}}{1 + S/K_{ms} + P/K_{mp} + S^2/K_i}$
uai	$S, V, K_{sa},$ K_{sc}	Substrate Activation	$v = \dfrac{V\,(S/K_{sa})^2}{1 + S/K_{sc} + (S/K_{sa})^2 + S/K_{sa}}$
ucii	$S, V, K_m,$ K_i	Competitive Inhibition (Irreversible)	$v = \dfrac{V S/K_m}{1 + S/K_m + I/K_i}$
ucir	$S, P, V_f,$ $V_r, K_{ms},$ K_{mp}, K_i	Competitive Inhibition (Reversible)	$v = \dfrac{V_f S/K_{ms} - V_r P/K_{mp}}{1 + S/K_{ms} + P/K_{mp} + I/K_i}$
unii	$S, I, V,$ K_m, K_i	Noncompetitive Inhibition (Irreversible)	$v = \dfrac{V S/K_m}{1 + I/K_i + S/K_m\,(1 + I/K_i)}$
unir	$S, P, I, V_f,$ $K_{ms}, K_{mp},$ K_i	Noncompetitive Inhibition (Reversible)	$v = \dfrac{V_f S/K_{ms} - V_r P/K_{mp}}{1 + I/K_i + (S/K_{ms} + P/K_{mp})\,(1 + I/K_i)}$
uuci	$S, I, V,$ K_m, K_i	Uncompetitive Inhibition (Irreversible)	$v = \dfrac{V S/K_m}{1 + S/K_m\,(1 + I/K_i)}$
uucr	$S, P, I, V_f,$ $V_r, K_{ms},$ K_{mp}, K_i	Uncompetitive Inhibition (Reversible)	$v = \dfrac{V_f S/K_{ms} - V_r P/K_{mp}}{1 + (S/K_{ms} + P/K_{mp})\,(1 + I/K_i)}$
umi	$S, I, V,$ $K_m, K_{is},$ K_{ic}	Mixed Inhibition Kinetics (Irreversible)	$v = \dfrac{V S/K_m}{1 + I/K_{is} + S/K_m\,(1 + I/K_{ic})}$
umr	$S, P, I,$ $V_f, V_r,$ $K_{ms}, K_{mp},$ K_{is}, K_{ic}	Mixed Inhibition Kinetics (Reversible)	$v = \dfrac{V_f S/K_{ms} - V_r P/K_{mp}}{1 + I/K_{is} + (S/K_{ms} + P/K_{mp})\,(1 + I/K_{ic})}$
uai	$S, A_c, V,$ K_m, K_a	Specific Activation Kinetics - irreversible	$v = \dfrac{V S/K_m}{1 + S/K_m + K_a/A_c}$
uar	$S, P, A_c,$ $V_f, V_r, K_{ms},$ K_{mp}, K_a	Specific Activation Kinetics (Reversible)	$v = \dfrac{V_f S/K_{ms} - V_r P/K_{mp}}{1 + S/K_{ms} + P/K_{mp} + K_a/A_c}$
ucti	$S, A_c, V,$ K_m, K_a	Catalytic Activation (Irreversible)	$v = \dfrac{V S/K_m}{1 + K_a/A_c + S/K_m\,(1 + K_a/A_c)}$

Table 7: *Table of rate law functions in SBML (continued). In all cases, $K_m > 0$, $V_x \geq 0$, $S \geq 0$ and $P \geq 0$.*

Name	Arguments	Meaning	Formula
uctr	S, P, A_c, V_f, V_r, K_{ms}, K_{mp}, K_a	Catalytic Activation (Reversible)	$v = \dfrac{V_f S/K_{ms} - V_r P/K_{mp}}{1 + K_a/A_c + (S/K_{ms} + P/K_{mp})\left(1 + K_a/A_c\right)}$
umai	S, A_c, V, K_m, Kas, Kac	Mixed Activation Kinetics (Irreversible)	$v = \dfrac{V S/K_m}{1 + Kas/A_c + S/K_m\left(1 + Kac/A_c\right)}$
umar	S, P, A_c, V_f, V_r, K_{ms}, K_{mp}, K_{as}, K_{ac}	Mixed Activation Kinetics (Reversible)	$v = \dfrac{V_f S/K_{ms} - V_r P/K_{mp}}{1 + K_{as}/A_c + (S/K_{ms} + P/K_{mp})\left(1 + K_{ac}/A_c\right)}$
uhmi	S, M, V, K_m, K_d, a, b	General Hyperbolic Modifier Kinetics (Irreversible)	$v = \dfrac{V S/K_m\left(1 + bM/(aK_d)\right)}{1 + M/K_d + S/K_m\left(1 + M/(aK_d)\right)}$
uhmr	S, P, M, V_f, V_r, K_{ms}, K_{mp}, K_d, a, b	General Hyperbolic Modifier Kinetics (Reversible)	$v = \dfrac{\left(V_f S/K_{ms} - V_r P/K_{mp}\right)\left(1 + bM/(aK_d)\right)}{1 + M/K_d + (S/K_{ms} + P/K_{mp})\left(1 + M/(aK_d)\right)}$
ualii	S, I, V, K_s, K_{ii}, n, L	Allosteric inhibition (Irreversible)	$v = \dfrac{V\left(1 + S/K_s\right)^{n-1}}{L\left(1 + I/K_{ii}\right)^n + \left(1 + S/K_s\right)^n}$
ordubr	A, P, Q, V_f, V_r, K_{ma}, K_{mq}, K_{mp}, K_{ip}, K_{eq}	Ordered Uni Bi Kinetics	$v = \dfrac{V_f\left(A - PQ/K_{eq}\right)}{\left[\begin{array}{l} K_{ma} + A\left(1 + P/K_{ip}\right) \\[4pt] + V_f/(V_r K_{eq})\left(K_{mq}P + K_{mp}Q + PQ\right)\end{array}\right]}$
ordbur	A, B, P, V_f, V_r, K_{ma}, Kmb, K_{mp}, K_{ia}, K_{eq}	Ordered Bi Uni Kinetics	$v = \dfrac{V_f\left(AB - P/K_{eq}\right)}{\left[\begin{array}{l} AB + K_{ma}B + KmbA \\[4pt] + V_f/(V_r K_{eq})\left(K_{mp} + P\left(1 + A/K_{ia}\right)\right)\end{array}\right]}$
ordbbr	A, B, P, Q, V_f, K_{ma}, K_{mb}, K_{mp}, K_{ia}, K_{ib}, K_{ip}, K_{eq}	Ordered Bi Bi Kinetics	$v = \dfrac{V_f\left(AB - PQ/K_{eq}\right)}{AB\left(1 + P/K_{ip}\right) + K_{mb}(A + K_{ia}) + K_{ma}B + K_1}$ where $K_1 = V_f/(V_r K_{eq})\left(K_{mq}P\left(1 + A/K_{ia}\right) + Qk_2\right),$ $K_2 = K_{mp}\left(1 + K_{ma}B/(K_{ia}K_{mb}) + P\left(1 + B/K_{ib}\right)\right)$
ppbr	A, B, P, Q, V_f, V_r, K_{ma}, K_{mb}, K_{mp}, K_{mq}, K_{ia}, K_{iq}, K_{eq}	Ping Pong Bi Bi Kinetics	$v = \dfrac{V_f\left(AB - PQ/K_{eq}\right)}{AB + K_{mb}A + K_{ma}B\left(1 + Q/K_{iq}\right) + K_1}$ where $K_1 = V_f/(V_r K_{eq})\left(K_{mq}P\left(1 + A/K_{ia}\right) + Q(K_{mp} + P)\right)$

Table 7: *Table of rate law functions in SBML (continued). In all cases, $K_m > 0$, $V_x \geq 0$, $S \geq 0$ and $P \geq 0$.*

Symbol	Meaning
α	Effect of S and P on binding of M (if $M < 1$, M is inhibitor; if $M > 1$, M is activator)
A	First substrate in two substrate reaction
A_c	Activator
B	Second substrate in two substrate reaction
I	Inhibitor
K_1	Forward Rate Constant
K_2	Reverse Rate Constant
K_a	Activation Constant
K_{ac}	Catalytic Activation Constant
K_{as}	Specific Activation Constant
K_d	Dissociation constant of the elementary step $E + M = EM$
K_{eq}	Equilibrium Constant
K_{ii}	Dissociation constant of the inhibitor from the inactive form of the enzyme
K_i	Inhibition constant for the substrate.
K_{ia}	Product inhibition constant of A acting on the reverse reaction
K_{ib}	Product inhibition constant of B acting on the reverse reaction
K_{ic}	Catalytic (noncompetitive) inhibition constant
K_{ip}	Product inhibition constant of P acting on the forward reaction
K_{iq}	Product inhibition constant of Q acting on the forward reaction
K_{is}	Specific (competitive) inhibition constant
K_m	Forward Michaelis-Menten Constant
K_{ma}	Concentration of A such that $v = V_f/2$ (Michaelis constant) at zero P and zero Q
K_{mb}	Concentration of B such that $v = V_f/2$ (Michaelis constant) at saturating A and zero P
K_{mp}	Concentration of P such that $v = -V_r/2$ (Michaelis constant) at zero A and B
K_{mp}	Product Michaelis-Menten Constant
K_{mq}	Concentration of Q such that $v = -V_r/2$ (Michaelis constant) at zero A and saturating P
K_{ms}	Substrate Michaelis-Menten Constant
K_s	Dissociation constant of the substrate from the active form of the enzyme
K_{sa}	Dissociation constant of substrate-activation site
K_{sc}	Dissociation constant of substrate-active site
L	Equilibrium constant between the active and inactive forms of the enzyme
M	Modifier
$M_{0.5}$	Concentration of M that half-saturates its binding site when $S = 0$, $P = 0$
P	First product in two product reaction
$P_{0.05}$	Product concent. s.t. $v = -V_r/2$ when $P = M = 0$ (V_r is limiting rate of reverse reaction)
Q	Second product in two product reaction
$S_{0.05}$	Irreversible rate laws: Substrate concentration such that $v = V_f/2$ when $P = 0, M = 0$
V	Forward Maximum Velocity
V_f	Forward Maximum Velocity
V_m	Forward Maximum Velocity
V_r	Reverse Maximum Velocity
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 $ES \rightarrow E + P$.
h	Hill Coefficient
n	No. binding sites for substrate & inhibitor (typically the number of monomers in the enzyme)

Table 8: *Table of symbols used in Table 7.*

References

Abbott, A. (1999). Alliance of US labs plans to build map of cell signalling pathways. *Nature*, 402:219–200.

Arkin, A. P. (2001). *Simulac* and *Deduce*. Available via the World Wide Web at `http://gobi.lbl.gov/~aparkin/Stuff/Software.html`.

Biron, P. V. and Malhotra, A. (2000). XML Schema part 2: Datatypes (W3C candidate recommendation 24 October 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*.

Bray, D., Firth, C., Le Novère, N., and Shimizu, T. (2001). *StochSim*. Available via the World Wide Web at `http://www.zoo.cam.ac.uk/comp-cell/StochSim.html`.

Bray, T., D. Hollander, D., and Layman, A. (1999). Namespaces in XML. World Wide Web Consortium 14-January-1999. Available via the World Wide Web at `http://www.w3.org/TR/1999/REC-xml-names-19990114/`.

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*. John Wiley & Sons, New York.

Fallside, D. C. (2000). XML Schema part 0: Primer (W3C candidate recommendation 24 October 2000). Available via the World Wide Web at `http://www.w3.org/TR/xmlschema-0/`.

Gilman, A. (2000). A letter to the signaling community. Alliance for Cellular Signaling, The University of Texas Southwestern Medical Center. Available via the World Wide Web at `http://afcs.swmed.edu/afcs/Letter_to_community.htm`.

Goryanin, I. (2001). *DBSolve*: Software for metabolic, enzymatic and receptor-ligand binding simulation. Available via the World Wide Web at `http://websites.ntl.com/-igor.goryanin/`.

Goryanin, I., Hodgman, T. C., and Selkov, E. (1999). Mathematical simulation and analysis of cellular metabolism and regulation. *Bioinformatics*, 15(9):749–758.

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

Harold, E. R. and Means, E. S. (2001). *XML in a Nutshell*. O'Reilly & Associates.

Hofmeyr, J. H. and Cornish-Bowden, A. (1997). The reversible Hill equation: How to incorporate cooperative enzymes into metabolic models. *Computer Applications in the Biosciences*, 13:377–385.

Hucka, M. (2000). SCHUCS: A notation for describing model representations intended for XML encoding. Available via the World Wide Web at `ftp://ftp.cds.caltech.edu/pub/caltech-erato/notation/`.

Kitano, H. (2001). *Foundations of Systems Biology*. MIT Press.

Mendes, P. (1997). Biochemistry by numbers: Simulation of biochemical pathways with Gepasi 3. *Trends in Biochemical Sciences*, 22:361–363.

Mendes, P. (2001). Gepasi 3.21. Available via the World Wide Web at `http://www.gepasi.org`.

Morton-Firth, C. J. and Bray, D. (1998). Predicting temporal fluctuations in an intracellular signalling pathway. *Journal of Theoretical Biology*, 192:117–128.

Naur, P. (1960). Revised report on the algorithmic language ALGOL 60. *Communications of the ACM*, 3(5):299–314.

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

Physiome Sciences, I. (2001). CellML™ home page. Available via the World Wide Web at `http://cellml.org/`.

Popel, A. and Winslow, R. L. (1998). A letter from the directors... Center for Computational Medicine & Biology, Johns Hopkins School of Medicine, Johns Hopkins University. Available via the World Wide Web at `http://www.bme.jhu.edu/ccmb/ccmbletter.html`.

Sauro, H. M. (2000). Jarnac: A system for interactive metabolic analysis. In Hofmeyr, J.-H. S., Rohwer, J. M., and Snoep, J. L., editors, *Animating the Cellular Map: Proceedings of the 9th International Meeting on BioThermoKinetics*. Stellenbosch University Press. ISBN 0-7972-0776-7.

Sauro, H. M. and Fell, D. A. (1991). SCAMP: A metabolic simulator and control analysis program. *Mathl. Comput. Modelling*, 15:15–28.

Schaff, J., Slepchenko, B., and Loew, L. M. (2000). Physiological modeling with the Virtual Cell framework. In Johnson, M. and Brand, L., editors, *Methods in Enzymology*, volume 321, pages 1–23. Academic Press, San Diego.

Schaff, J., Slepchenko, B., Morgan, F., Wagner, J., Resasco, D., Shin, D., Choi, Y. S., Loew, L., Carson, J., Cowan, A., Moraru, I., Watras, J., Teraski, M., and Fink, C. (2001). Virtual Cell. Available via the World Wide Web at `http://www.nrcam.uchc.edu`.

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

Smaglik, P. (2000a). For my next trick... *Nature*, 407:828–829.

Smaglik, P. (2000b). US grant glues 'virtual cell' together. *Nature*, 407(6806):819.

Taylor, B. N. (1995). Guide for the use of the international system of units (SI). NIST Special Publication 811, 1995 Edition. National Institute of Standards and Technology, United States Department of Commerce. Available via the World Wide Web at `http://physics.nist.gov/Pubs/SP811/contents.html`.

Thompson, H. S., Beech, D., Maloney, M., and Mendelsohn, N. (2000). XML Schema part 1: Structures (W3C candidate recommendation 24 October 2000). Available via the World Wide Web at `http://www.w3.org/TR/xmlschema-1/`.

Tomita, M., Hashimoto, K., Takahashi, K., Shimizu, T., Matsuzaki, Y., Miyoshi, F., Saito, K., Tanida, S., Yugi, K., Venter, J. C., and Hutchison, C. (1999). E-Cell: Software environment for whole cell simulation. *Bioinformatics*, 15(1):72–84.

Tomita, M., Nakayama, Y., Naito, Y., Shimizu, T., Hashimoto, K., Takahashi, K., Matsuzaki, Y., Yugi, K., Miyoshi, F., Saito, Y., Kuroki, A., Ishida, T., Iwata, T., Yoneda, M., Kita, M., Yamada, Y., Wang, E., Seno, S., Okayama, M., Kinoshita, A., Fujita, Y., Matsuo, R., Yanagihara, T., Watari, D., Ishinabe, S., and Miyamoto, S. (2001). E-Cell. Available via the World Wide Web at `http://www.e-cell.org/`.

Unicode Consortium (1996). *The Unicode Standard, Version 2.0*. Addison-Wesley Developers Press, Reading, Massachusetts.

W3C (2000a). Naming and addressing: URIs, URLs, ... Available via the World Wide Web at `http://www.w3.org/Addressing/`.

W3C (2000b). W3C's math home page. Available via the World Wide Web at `http://www.w3.org/Math/`.