Encoding neuronal models in SBML

Sarah M. Keating and Nicolas Le Novère

1 Systems Biology and the need to exchange models

Systems Biology of neurobiological systems needs to take into account the interactions between a very large number of physical entities, and the analysis of many parameters. As seen in the previous chapters, the quantitative relationships between entities, their interactions, are often described using mathematical models. It is therefore crucial to be able to encode those models in a standard way to foster their exchange and re-use. Additionally using a standard format permits further processing, for instance merging of models, and relating them to other types of information. Early modelers and software developers in systems biology quickly realized that if their efforts were to be of benefit to the wider community it must be possible to share and re-use the models. The best way to facilitate this, and to enable concurrent use of multiple software packages with different capabilities, was to agree a common format for describing the models. There are many ways to describe models in a standardized manner. One can use natural languages, graphical languages, sets of equations, logical relationships between elements etc. [A REF HERE WOULD BE COOL]. In this chapter, as in the chapter [CHAPTER-GLEESON], we focus on the representation of the variables representing physical entities, their relationships and the necessary parameters, encoded in a text file.

Sarah M Keating
EMBL-EBI, Cambridge, UK, e-mail: skeating@ebi.ac.uk

Nicolas Le Novère
EMBL-EBI, Cambridge, UK, e-mail: lenov@ebi.ac.uk

1.1 A bit of history

The need for a language to exchange models became manifest at the end of the last century, with efforts starting in the field of metabolic networks (Kell and Mendes, 2008, see history in) and physiology modeling (Hedley et al, 2001). A similar need was expressed during the first Workshop on Software Platforms for Systems Biology, held at the California Institute of Technology in early 2000. The Systems Biology Markup Language (SBML) (Hucka et al, 2003) is a machine-readable model definition language based on XML, the eXtensible Markup Language (Bray et al, 2000). An SBML document contains all the information pertaining to the structure of a model, including the list of symbols, both variables and constants, the list of mathematical relationships linking them, and all the numbers needed to instantiate simulations. SBML was originally viewed as being aimed towards models of molecular pathways (Strömbäck and Lambrix, 2005). However, its versatility means that SBML can be, and today is being, used in a variety of modeling contexts. For instance, BioModels Database (Le Novère et al, 2006) contains SBML representations of models including cell signaling (Goldbeter, 1991), metabolism (Curto et al, 1998), gene regulation (Elowitz and Leibler, 2000) and neuronal models, some of which are described in detail later in this chapter. In general, SBML enables the encoding of any mathematical model based on pools of entities and processes that modify them. This versatility is currently expanding, towards rule-based modeling, reaction-diffusion etc. Since its creation in 2000, SBML has continued to evolve as an international community effort, and has grown in terms of the levels of acceptance; to the point where, at the time of this writing, it is used by over 200 software packages worldwide and required as a format for model encoding by many journals.

1.2 Levels and Versions of SBML

SBML is being developed in stages, with specifications released at the end of each development stage. This approach, which effectively freezes SBML development at incremental points, allows users to work with stable standards and gain experience with the standard before further development. Future development can then benefit from the practical experiences of users and developers.

Major editions of SBML are termed *levels* and represent substantial changes to the composition and structure of the language. The latest level being developed is Level 3 (Hucka et al, 2009); representing a major evolution of the language through Level 2 (Hucka et al, 2008) from the introduction of Level 1 in the year 2001 (Hucka et al, 2001, 2003). SBML Level 3 is being developed as a modular language, with a central core comprising a self-sufficient model definition language, and extension packages layered on top of this core to provide additional, optional sets of features. Only the core will be described in this chapter.

The separate levels of SBML are intended to coexist. All of the constructs of Level 1, i.e. the elements and attributes of the SBML representation, can be mapped to Level 2; likewise, the majority of the constructs from Level 2 can be mapped to Level 3 Core[1]. In addition, a subset of Level 3 constructs can be mapped to Level 2, and a subset of Level 2 constructs can be mapped to Level 1. However, the levels remain distinct; a valid SBML Level 1 document is not a valid SBML Level 2 document, and so on.

Minor revisions of SBML are termed *versions*, and constitute changes within a Level to correct, adjust and refine language features. All examples used here will be from the latest stable version of SBML; that is, SBML Level 3 Version 1 Core. It should be noted that SBML Level 3 is a recent development and as yet not many software tools support it.

2 Structure of SBML

SBML is a structured language, with a strict syntax and very precise semantics. In this section we will present the most common constructs of SBML. However, a serious understanding of the language can only be achieved through the SBML specification document (Hucka et al, 2009).

An SBML document is essentially an XML document containing an **sbml** element which declares the *namespace*, *level* and *version* of SBML. The **sbml** element MUST contain a **model** element which itself consists of lists of one or more components. The SBML snippet shown illustrates an **sbml** element containing a **model** element.

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level3/version1/core"
      level="3" version="1">
  ...
  <model ...>  ... </model>
</sbml>
```

Table 1 lists all the components defined by SBML; with a brief description of the semantics of each.

Some components in SBML represent items that have a numerical value, that may be constant or may vary throughout a simulation. The constructs that represent possible variables are compartment, species and parameter. In all these cases, the *id* attribute of the component is used throughout the model to represent the numerical value of that component at the point in time specified by any simulation/analysis that is being undertaken. It is also possible to introduce variable stoichiometry into reactions but this is beyond the scope of the current text.

[1] The SpeciesType and CompartmentType contructs which appear in Level 2 Versions 2-4 were removed in Level 3 Core as it was considered they were better suited to an extension package.

Table 1 Components of an SBML **model** element

Component	Description
compartment	a container of finite size for well-stirred substances
species	a pool of undistinguishable entities
parameter	a quantity of whatever type is appropriate
reaction	a statement describing some transformation, transport or binding process that can change one or more species
rule	a mathematical expression that is added to the model equations
event	a set of mathematical formulas evaluated at specified moments in the time evolution of the system
initialAssignment	a mathematical formula to assign the initial value of a component
functionDefinition	a named mathematical function that can be used in place of repeated expressions
constraint	a mathematical formula for stating the assumptions under which the model is designed to operate
unitDefinition	a name for a unit used in the expression of quantities in a model

Other components represent mathematical constructs that define some level of interaction between the components that can be varied. These constructs include the reaction, rules and event components.

The remaining constructs: initialAssignment, functionDefinition, constraint and unitDefinition; provide methods of adding further information or mathematical detail to a model. It is however possible to construct a complete model without using these components, which will not be explored further here.

2.1 Compartment

The **compartment** component in SBML represents a container of finite size for well-stirred substances where the species defined in the model are located. Biologically speaking this may represent for instance a body fluid, a cell or a subcellular compartment, but SBML does not require the **compartment** to represent an actual structure, either inside or outside a biological system.

The **compartment** has attributes that specify its *spatialDimensions*, its *size* and the corresponding *units*, plus a *constant* attribute that determines whether the size can change or not during a simulation. The following SBML snippet represents a constant, 3D compartment with volume 2.3 litres.

```
<model>
  ...
  <listOfCompartments>
    <compartment id="cell" spatialDimensions="3"
                 size="2.3" units="litre" constant="true"/>
  </listOfCompartments>
  ...
```

```
</model>
```

The *id* attribute can be used elsewhere in the model to represent the numerical value of the size of the **compartment**.

2.2 Species

The **species** component in SBML does not represent a single molecule but rather a pool, that is an ensemble of indistinguishable entities, represented by its concentration or amount in a **compartment**. The environment is well-stirred and thus no concentration gradients need to be considered. [2] Regardless of whether species within models are specified using amount or concentration, the value of the attribute *id* of a **species** refers to concentration when it is used in a mathematical context, UNLESS the **species** has been declared as being in units of amount by either using the *hasOnlySubstanceUnits* attribute or locating the **species** in a **compartment** with *spatialDimensions* of zero.

It is also common for a species to exist on the boundary of the system being modeled; in which situation the quantity is unchanged by reactions it may be involved in. The *boundaryCondition* attribute implies that whilst the species may be a product or reactant within reactions, its quantity is not determined by those reactions.

The SBML snippet illustrates a variable species located in the compartment with id ‘cell’. It is not on the boundary, it is to be used as a concentration and possesses an initial amount of 4.6 moles.

```
<model>
  ...
  <listOfSpecies>
    <species id="s"
             compartment="cell"
             initialAmount="4.6"
             substanceUnits="mole"
             hasOnlySubstanceUnits="false"
             boundaryCondition="false"
             constant="false"/>
  </listOfSpecies>
  ...
</model>
```

The species specified above has an initial amount of 4.6 moles. However, the *hasOnlySubstanceUnits* attribute has a value of false, indicating that whenever the *id* of the species appears in the model it refers to concentration. Thus for any analysis, it may be necessary to convert between amount and concentration using

[2] Discussion are under way to propose one or more Level 3 package that will address this issue.

$$concentration = \frac{amount}{size}$$

where size refers to the *size* of the **compartment** in which the **species** is located.

It is possible to create models in SBML without the need to consider units and thus units have largely been ignored within this text. However, in the situation where a model uses species that have been located within a compartment whose size is not unity the issue of concentration and amount **must** be considered.

2.3 Parameter

The **parameter** component in SBML can represent anything with a numerical value that the modeler wishes to include. This may be the rate constant of a rate equation, the potential of a membrane or the current used to induce spiking in a neuronal model. Therefore, it is a component of particular importance for neurobiological models. The **parameter** has attributes that specify the *value*, *units* and whether the value is fixed.

The SBML snippet illustrates a variable parameter with value 3000 and a constant parameter with value 8000.

```
<model>
  ...
  <listOfParameters>
    <parameter id="p1"
               value="3000"
               constant="false"/>
    <parameter id="p2"
               value="8000"
               constant="true"/>
  </listOfParameters>
  ...
</model>
```

2.4 Reaction

A **reaction** in SBML represents any kind of process that can change the quantity of one of more **species**. It may be a mass action reaction, or involve transport, catalysis, or any process that changes the species involved (note that transport changes species because they are located in compartments). It is necessary to define the participating reactants and/or products. This is done using a **speciesReference** component that identifies the species from the model's **listOfSpecies** and assigns a *stoichiometry* value to that species role within the reaction. Species that merely influence a reaction, such as a catalyst, are listed as objects of type **modifierSpeciesReference**. This construct is similar to **speciesReference** without the *stoichiometry* attribute. Attributes for a **reaction** object allow the modeler to specify whether the reaction is *reversible* or *fast*. The mathematics describing the velocity of the reaction can be encoded in the **kineticLaw** component. SBML uses a subset of the MathML 2.0 standard (Ausbrooks et al, 2003) to encode mathematical formula directly within SBML components. **LocalParameter** objects can be included within a **kineticLaw**. These localParameters have constant values and are local to the enclosing kineticLaw; they cannot be used elsewhere in the model. Note that an SBML **kineticLaw** represents the extent of the reaction per unit of time, and not the rate of the reaction. In other words, the result is not a concentration per time, but a quantity per time. This is why the rate is multiplied by the volume in the **kineticLaw**.

The SBML snippet shows the description of the reaction

$$S0 \xrightarrow{k} S1$$

with a rate of

$$k \times S0 \times S2$$

where $S0$ and $S1$ are two species residing in a compartment V, and $S2$ is a catalyst.

```
<model>
  ...
  <listOfReactions>
    <reaction reversible="false"
              fast="false">
      <listOfReactants>
        <speciesReference species="S0"
                          stoichiometry="1"/>
      </listOfReactants>
      <listOfProducts>
        <speciesReference species="S1"
                          stoichiometry="1"/>
      </listOfProducts>
      <listOfModifiers>
        <modifierSpeciesReference species="S2"/>
      </listOfModifiers>
      <kineticLaw>
        <math xmlns="http://www.w3.org/1998/Math/MathML">
          <apply>
            <times/>
            <ci> k </ci>
            <ci> S0 </ci>
            <ci> S2 </ci>
            <ci> V </ci>
          </apply>
        </math>
        <listOfLocalParameters>
          <localParameter id="k" value="0.1"/>
        </listOfLocalParameters>
      </kineticLaw>
    </reaction>
  </listOfReactions>
  ...
</model>
```

2.5 Rule

In SBML, **rules** provide additional ways of defining values of variables in a model, their relationships, and the dynamical behaviors of those variables. For example, a model may wish to calculate the total concentration of a chemical that appears as a part of several compounds within the model or vary a parameter representing voltage. There are three subclasses of **rules**: **algebraicRules**, **assignmentRules** and **rateRules**. In the current discussion, we will only consider the latter two types, which have the following form:

$$Assignment: \ x = f(V)$$

$$Rate: \ \frac{dx}{dt} = f(W)$$

where

x	variable
f	some arbitrary function
V	vector of variables not including x
W	vector of variables that may include x

Rules included in an SBML model are considered to hold true at all times. Therefore they must be included in the set of equations that define the model for simulation or other purposes.

The SBML snippet here shows two rules that describe the following equations.

$$y = 2x + 1$$

$$\frac{dg}{dt} = g - 1$$

```
  <model>
    ...
    <listOfRules>
      <assignmentRule variable="y">
```

```
        <math xmlns="http://www.w3.org/1998/Math/MathML">
          <apply>
            <plus/>
            <apply>
              <times/>
              <cn> 2 /cn>
              <ci> x </ci>
            </apply>
            <cn> 1 </cn>
          </apply>
        </math>
      </assignmentRule>
      <rateRule variable="g">
        <math xmlns="http://www.w3.org/1998/Math/MathML">
          <apply>
            <minus/>
            <ci> g </ci>
            <cn> 1 </cn>
          </apply>
        </math>
      </rateRule>
    </listOfRules>
    ...
  </model>
```

2.6 Event

An **event** in SBML describes the time and form of an explicit discontinuous state change in the model. For example this could be a situation where the voltage is reset when it reaches a given threshold.

The description of an **event** involves a **trigger**; a mathematical statement that determines when the event is fired; and a **listOfEventAssignments** that determine the action to be executed.

SBML does provide a **delay** for describing delayed events, when there is a period of time between when an event is ‘fired’ and when the event is ‘executed’. There is also a **priority** component for assigning a priority to the **event**. Neither **delay** or **priority** are discussed here.

The SBML snippet[3] describes an **event** that resets the value of **parameter** Vthres to -60 when the value of a second **parameter** V exceeds a value of 30.

```
  <model>
    ...
    <listOfParameters>
      <parameter id="Vthres"
                 value="30"
```

[3] In order to show valid SBML a number of attributes are shown within the snippet. These are not referred to in the text as they represent a level of complexity beyond the scope of this text.

```
                 constant="false"/>
    </listOfParameters>
    ...
    <listOfEvents>
      <event useValuesFromTriggerTime="true">
        <trigger initialValue="true" persistent="true">
          <math xmlns="http://www.w3.org/1998/Math/MathML">
            <apply>
              <gt/>
              <ci> V </ci>
              <cn> 30 </cn>
            </apply>
          </math>
        </trigger>
        <listOfEventAssignments>
          <eventAssignment variable="Vthres">
            <math xmlns="http://www.w3.org/1998/Math/MathML">
              <cn> -60 </cn>
            </math>
          </eventAssignment>
        </listOfEventAssignments>
      </event>
    </listOfEvents>
    ...
  </model>
```

2.7 Metadata

In addition to the model semantics, that is the variables and their mathematical relationships, SBML provides two mechanisms to add a layer of biological semantics on top of each component of the model.

Firstly, an attribute *sboTerm* allows any element to be linked to a single term of the *Systems Biology Ontology* (http://www.ebi.ac.uk/sbo, (Le Novère et al, 2007)). SBO is a controlled vocabulary (an "ontology") tailored specifically for the kinds of problems being faced in Systems Biology, especially in the context of computational modeling. SBO is made up of different vocabularies covering quantitative parameters, modeling frameworks, type of mathematical expressions, biological interactions, types of entities etc. The *sboTerm* enables unambiguous identification of the type of concept being dealt with by the model.

Secondly, one can add biological information by linking an SBML component to external resources, either terms from controlled vocabularies or entries in biological databases. In order to precisely relate SBML components and annotations, the links are encoded using existing semantic web technologies such as the *Resource Description Framework* (Manola and Miller, 2004).

The following SBML snippet described a species that corresponds to a "protein complex" (SBO:0000297). It is made up of two parts, the protein calmodulin (described by the UniProt entry P62158) and the divalent calcium cation (ChEBI term CHEBI:29108).

```
<species id="Ca_calmodulin" metaid="cacam" sboTerm="SBO:0000297"
         compartment="C" hasOnlySubstanceUnits="false"
         boundaryCondition="false" constant="false">
  <annotation>
    <rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
              xmlns:bqbiol="http://biomodels.net/biology-qualifiers/">
      <rdf:Description rdf:about="#cacam">
        <bqbiol:hasPart>
          <rdf:Bag>
            <rdf:li rdf:resource="urn:miriam:uniprot:P62158"/>
            <rdf:li rdf:resource="urn:miriam:obo.chebi:CHEBI%3A29108"/>
          </rdf:Bag>
        </bqbiol:hasPart>
      </rdf:Description>
    </rdf:RDF>
  </annotation>
</species>
```

3 Creating neuronal models in SBML

The SBML structures discussed above provide a syntax sufficiently rich to encode many basic neuronal models. In particular, models involving metabolic networks, signaling pathways or gene regulatory networks are covered. In addition, “single compartment” electrical models can also be encoded in SBML. In this section, we give a couple of concrete examples. For encoding multi-compartment electrical models (“Rall models”), other representations such as NeuroML (Gleeson et al (2010); see also chapter [REFERENCE TO THE NEUROML CHAPTER]) are more suitable.

3.1 Integration of Dopamine and Glutamate Signals

The integration of different neurochemical signals (Cohen, 1992) is one of the fundamental basis of neuronal function and plasticity (Bhalla and Iyengar, 1999). Figure 1 depicts a simple integration of glutamate and dopamine signals by the phosphatase inhibitor DARPP-32. The model is derived from a more comprehensive one described by Fernandez et al (2006). It is possible to create a quantitative model of the reactions involved in the phosphorylation and dephosphorylation of DARPP-32, and study the dynamic behavior using calcium and cAMP as inputs to represent the response to glutamate and dopamine respectively. The complete model can be found in BioModels Database with the accession BIOMD0000000152.

3.1.1 Mathematical model of the biochemistry

Here we construct a reduced version of the model concentrating on the threonine 34 phosphorylation site of DARPP-32, the phosphorylation due to protein kinase A (PKA) and the dephosphorylation by calcineurin (PP2B). The resulting reactions are listed below in Table 2. The parameter values are modified slightly from the original model in order to retain the main behavior despite a drastic simplification.

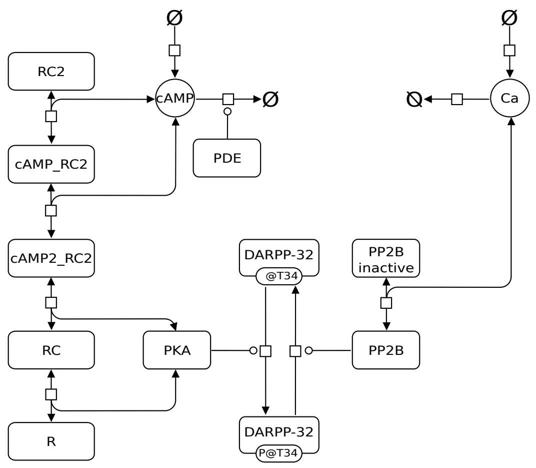

Fig. 1 Graphical representation of the model in the Process Description language of the Systems Biology Graphical Notation (Le Novère et al, 2009)

3.1.2 Encoding the model in SBML

The model consists of **species** that represent the chemicals being altered, **parameters** that define the rate constants, and **reactions** that determine how the species are being altered. All reactions are modeled using Mass-Action Law, and the enzymatic processes are decomposed in elementary steps. Consider the first equation listed:

$$D + PKA \rightleftharpoons D\_PKA \qquad (1)$$

with forward rate equation:

$$v_f = k1_f \times D \times PKA \qquad (2)$$

and backward rate equation:

$$v_b = k1_b \times D\_PKA \qquad (3)$$

Table 2 Reactions in the model

Type	Reactions
Phosphorylations	
	$D + PKA \rightleftharpoons D\_PKA$
	$D\_PKA \rightarrow D34 + PKA$
	$D34 + PP2B \rightleftharpoons D34\_PP2B$
	$D34\_PP2B \rightarrow D + PP2B$
PP2B activation	
	$PP2B_{inactive} + 2Ca \rightleftharpoons PP2B$
PKA activation	
	$R2C2 + cAMP \rightleftharpoons cAMP\_R2C2$
	$cAMP\_R2C2 + cAMP \rightleftharpoons cAMP2\_R2C2$
	$cAMP2\_R2C2 + cAMP \rightleftharpoons cAMP3\_R2C2$
	$cAMP3\_R2C2 + cAMP \rightleftharpoons cAMP4\_R2C2$
	$cAMP4\_R2 + PKA \rightleftharpoons cAMP4\_R2C$
	$cAMP4\_R2C + PKA \rightleftharpoons cAMP4\_R2C2$
cAMP degradation	
	$cAMP + PDE \rightleftharpoons cAMP\_PDE$
	$cAMP\_PDE \rightarrow PDE + \emptyset$
Calcium input/destroy	
	$\emptyset \rightarrow Ca$
	$Ca \rightarrow \emptyset$

The SBML model to encode this reaction must contain a **compartment**, in which the species are located and definitions for each of the **species**. Although biochemically speaking the reaction is reversible, since the rate equations for the forward and reverse reactions differ it is sometimes more convenient to define this as two irreversible SBML **reactions**. In the following, we use reversible reactions for reasons of compactness.

```
<listOfCompartments>
  <compartment id="Spine" size="1e-15" spatialDimension="3" constant="true"/>
```

```
</listOfCompartments>

<listOfSpecies>
  <species id="D" compartment="Spine"
           initialConcentration="4.98e-06" boundaryCondition="false"
           hasOnlySubstanceUnits="false" constant="false"/>
  <species id="PKA" compartment="Spine"
           initialConcentration="0" boundaryCondition="false"
           hasOnlySubstanceUnits="false" constant="false"/>
  <species id="D_PKA" compartment="Spine"
           initialConcentration="0" boundaryCondition="false"
           hasOnlySubstanceUnits="false" constant="false"/>
  ...
</listOfSpecies>

<listOfReactions>
  <reaction name="D_PKA_binding" reversible="true" fast="false">
    <listOfReactants>
      <speciesReference species="D" stoichiometry="1" constant="true"/>
      <speciesReference species="PKA" stoichiometry="1" constant="true"/>
    </listOfReactants>
    <listOfProducts>
      <speciesReference species="D_PKA" stoichiometry="1" constant="true"/>
    </listOfProducts>
    <kineticLaw>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <apply>
          <minus/>
          <apply>
            <times/>
            <ci> Spine </ci>
            <ci> kon </ci>
            <ci> D </ci>
            <ci> PKA </ci>
          </apply>
          <apply>
            <times/>
            <ci> Spine </ci>
            <ci> koff </ci>
            <ci> D_PKA </ci>
          </apply>
        </apply>
      </math>
      <listOfLocalParameters>
        <localParameter id="kon" value="5600000"/>
        <localParameter id="koff" value="10.8"/>
      </listOfLocalParameters>
    </kineticLaw>
  </reaction>
  ...
</listOfReactions>
```

This model includes two reactions intended to regulate the level of calcium, that is, the input and output of calcium from the system, represented as:

$$\emptyset \rightarrow Ca$$
$$Ca \rightarrow \emptyset$$

Obviously calcium does not just magically appear and disappear; however the input and output are not relevant to the system being modeled. If one wants to explicitly represent the fact that there are source and sink, one can make use of the *boundaryCondition* attribute on a **species** ‘Empty’, and define the reactions as follows. Labeling a species as being a *boundaryCondition* implies that it remains unaffected by any reactions it takes part in.

```
<listOfSpecies>
  <species id="Empty" compartment="Spine" boundaryCondition="true"
           hasOnlySubstanceUnits="false" constant="true"/>
  ...
</listOfSpecies>

<listOfReactions>
  <reaction name="Ca_in" reversible="false" fast="false">
    <listOfReactants>
      <speciesReference species="Empty" stoichiometry="1" constant="true"/>
    </listOfReactants>
    <listOfProducts>
      <speciesReference species="Ca" stoichiometry="1" constant="true"/>
    </listOfProducts>
    <kineticLaw>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <apply>
          <times/>
          <ci> Spine </ci>
          <ci> kin </ci>
        </apply>
      </math>
    </kineticLaw>
  </reaction>

  <reaction name="Ca_out" reversible="false" fast="false">
    <listOfReactants>
      <speciesReference species="Ca" stoichiometry="1" constant="true"/>
    </listOfReactants>
    <listOfProducts>
      <speciesReference species="Empty" stoichiometry="1" constant="true"/>
    </listOfProducts>
    <kineticLaw>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <apply>
          <times/>
          <ci> Spine </ci>
          <ci> kout </ci>
          <ci> Ca </ci>
        </apply>
      </math>
      <listOfLocalParameters>
        <localParameter id="kout" value="1.7"/>
      </listOfLocalParameters>
    </kineticLaw>
  </reaction>
  ...
</listOfReactions>
```

Since the value of the species “Empty” is neither used nor displayed, it could equally be ignored altogether. Indeed a reaction in SBML is only required to have at least one reactant or one product. The following example represents the regulation of calcium with one reversible reaction.

```
  <listOfReactions>
    <reaction name="Ca_reg" reversible="true" fast="false">
      <listOfProducts>
        <speciesReference species="Ca" stoichiometry="1" constant="true"/>
      </listOfProducts>
      <kineticLaw>
        <math xmlns="http://www.w3.org/1998/Math/MathML">
          <apply>
            <minus/>
            <apply>
              <times/>
              <ci> Spine </ci>
              <ci> kin </ci>
            </apply>
            <apply>
              <times/>
              <ci> Spine </ci>
              <ci> kout </ci>
              <ci> Ca </ci>
            </apply>
          </apply>
        </math>
      </kineticLaw>
    </reaction>
    ...
  </listOfReactions>
```

Note that in the reactions above, the parameter ‘kin’ is a global parameter, the value of which will affect the amount of calcium being added to the system. Since the purpose of this model is to study the effect of different amounts of calcium, the parameter ‘kin’ can be varied to simulate different situations. The model uses **events** to add a pulse of cAMP and calcium in the course of the simulation. For the cAMP, this is done by setting a high concentration at a given time, while for the calcium, we increase the input (conductance of calcium channels) at a particular point and return it to a lower value after a delay. The **events** to produce a spike of calcium are shown below. Note there is an assumption that ‘kin’ is initially low.

```
<listOfEvents>
  <event id="event_2" name="ca_on1" useValuesFromTriggerTime="true">
    <trigger initialValue="true" persistent="false">
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <apply>
          <geq/>
          <csymbol encoding="text"
            definitionURL="http://www.sbml.org/sbml/symbols/time">
            time </csymbol>
          <apply>
            <plus/>
            <ci> cAMP_delay </ci>
            <ci> cAMP_Ca_delay </ci>
          </apply>
        </apply>
      </math>
    </trigger>
    <listOfEventAssignments>
      <eventAssignment variable="kin">
        <math xmlns="http://www.w3.org/1998/Math/MathML">
          <ci> kon_high </ci>
        </math>
      </eventAssignment>
    </listOfEventAssignments>
  </event>

  <event id="event_4" name="ca_off" useValuesFromTriggerTime="true">
    <trigger initialValue="true" persistent="false">
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <apply>
          <geq/>
          <csymbol encoding="text"
            definitionURL="http://www.sbml.org/sbml/symbols/time">
            time </csymbol>
          <apply>
            <plus/>
            <ci> cAMP_delay </ci>
            <ci> cAMP_Ca_delay </ci>
            <ci> spike_duration </ci>
          </apply>
        </apply>
      </math>
    </trigger>
    <listOfEventAssignments>
      <eventAssignment variable="kin">
        <math xmlns="http://www.w3.org/1998/Math/MathML">
          <ci> kon_low </ci>
        </math>
      </eventAssignment>
    </listOfEventAssignments>
  </event>
  ...
</listOfEvents>
```

Since the chemical of interest in this model is DARPP-32 phosphorylated on Thr34, and that some of these molecules may be bound to other species, such as PP2B, the SBML adds an **assignmentRule** which calculates the total number of D34 molecules present.

```
<assignmentRule variable="parameter_1">
  <math xmlns="http://www.w3.org/1998/Math/MathML">
    <apply>
      <plus/>
      <apply>
        <times/>
        <ci> D34 </ci>
        <csymbol encoding="text"
          definitionURL="http://www.sbml.org/sbml/symbols/avogadro">
          Na </csymbol>
        <ci> Spine </ci>
      </apply>
      <apply>
        <times/>
        <ci> D34_PP2B </ci>
        <csymbol encoding="text"
          definitionURL="http://www.sbml.org/sbml/symbols/avogadro">
          Na </csymbol>
        <ci> Spine </ci>
      </apply>
    </apply>
  </math>
</assignmentRule>
```

3.1.3 Simulation of a time course

We can use any simulator supporting SBML's reactions, assignmentRules and events to simulate the model and obtain the time course of the different model's variables. The result obtained with COPASI (Hoops et al, 2006) is shown on Figure 2. The description of the simulation experiment, that is what to do with the model and how, is not encoded in SBML. Another language is under development to cover this part of the model life-cycle, the Simulation Experiment Description Markup Language (SED-ML) (Köhn and Le Novère, 2008).

3.2 Hodgkin-Huxley axon model

In the 1930s, Alan Hodgkin and Andrew Huxley started a series of experiments and modeling to elucidate the flow of electric current through an axonal membrane. This led to the formulation of the Hodgkin-Huxley model in 1952 (Hodgkin and Huxley, 1952), a model that had major influence on our understanding of neuronal function

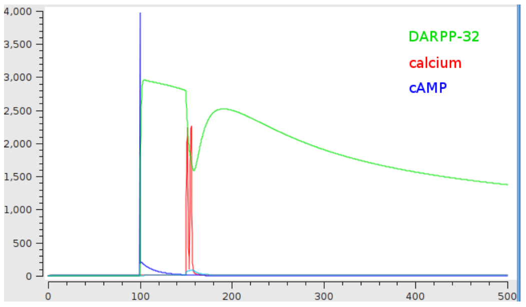

Fig. 2 Simulation of a model of glutamate and dopamine signal integration. The plot represent the temporal evolution of DARPP-32 phosphorylated on Thr34. After 100 sec, cAMP molecules are added, and after a further delay, two square increases of conductance cause calcium spikes.

and for which they received the Nobel Prize in 1963. The development of this model can also be seen as the birth of Systems Biology, since the authors described the emergence of a system's dynamic behavior using computational simulations of its components. The complete model can be found in BioModels Database with the accession BIOMD0000000020.

3.2.1 Mathematical model of the axon

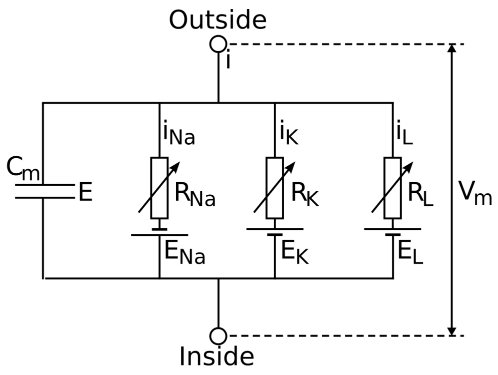

Fig. 3 In the Hodgkin-Huxley model, the membrane can be represented as an electrical circuit. Ionic current through the membrane can be divided into three components: potassium current (i_K), sodium current (i_{Na}), and a small leakage current (i_L) caused by other ions. The total current is calculated assuming that these components plus the capacity current are in parallel.

Using the equivalent electrical circuit to represent a patch of membrane (Figure 3) Hodgkin and Huxley derived four main equations that describe how the system, that is the probabilities of channel openings and the membrane voltage, varies with time.

$$\frac{dV}{dt} = \frac{I - (i_{Na} + i_K + i_L)}{C_m}$$
$$\frac{dm}{dt} = \alpha_m (1 - m) - \beta_m \cdot m$$

$$\frac{dh}{dt} = \alpha_h(1-h) - \beta_h \cdot h$$
$$\frac{dn}{dt} = \alpha_n(1-n) - \beta_n \cdot n \quad (4)$$

where

V	membrane depolarization voltage
I	**constant** applied current
i_{Na}	sodium current
i_K	potassium current
i_L	leakage current
C_m	**constant** membrane capacitance
m	sodium channel activation coefficient
h	sodium channel inactivation coefficient
n	potassium channel activation coefficient

and the α and β parameters are rate coefficients for the opening and closure of the gates, dependent on the instantaneous value of the membrane potential.

$$\alpha_m = \frac{0.1(V+25)}{\exp\left(\frac{V+25}{10}\right)-1}$$
$$\alpha_n = \frac{0.01(V+10)}{\exp\left(\frac{V+10}{10}\right)-1}$$
$$\alpha_h = 0.07\exp\left(\frac{V}{20}\right)$$
$$\beta_m = 4\exp\left(\frac{V}{18}\right)$$
$$\beta_n = 0.125\exp\left(\frac{V}{80}\right)$$
$$\beta_h = \frac{1}{\exp\frac{V+30}{10}+1} \quad (5)$$

The individual current values also depend on the difference between the equilibrium potential of each ion and the membrane potential, and the conductance of each channel (e.g. g_{Na} for the sodium channel).

$$i_{Na} = g_{Na} \cdot m^3 \cdot h \cdot (V - V_{Na})$$
$$i_K = g_K \cdot n^4 \cdot (V - V_K)$$
$$i_L = g_L \cdot (V - V_L) \quad (6)$$

3.2.2 Encoding the model in SBML

Since none of the variables under consideration in the model represent **species**, the SBML encoding of the model defines all these, and a number of other constants as parameters. Similarly the model does not contain any reactions, and all the equations are encoded using **rules**. For instance, the temporal evolution of the system is described with **rateRules**:

```
<rateRule variable="V">
  <math xmlns="http://www.w3.org/1998/Math/MathML">
    <apply>
      <divide/>
      <apply>
        <minus/>
        <ci> I </ci>
        <apply>
          <plus/>
          <ci> i_Na </ci>
          <ci> i_K </ci>
          <ci> i_L </ci>
        </apply>
      </apply>
      <ci> Cm </ci>
    </apply>
  </math>
</rateRule>

<rateRule variable="m">
  <math xmlns="http://www.w3.org/1998/Math/MathML">
    <apply>
      <minus/>
      <apply>
        <times/>
        <ci> alpha_m </ci>
        <apply>
          <minus/>
          <cn> 1 </cn>
          <ci> m </ci>
        </apply>
      </apply>
      <apply>
        <times/>
        <ci> beta_m </ci>
        <ci> m </ci>
      </apply>
    </apply>
  </math>
...
</rateRule>
```

The rate coefficients and the current, depending on the instantaneous voltage are encoded in SBML using **assignmentRules**. For instance:

```
<assignmentRule variable="alpha_n">
```

```
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      <apply>
        <divide/>
        <apply>
          <times/>
          <cn> 0.01 </cn>
          <apply>
            <plus/><ci> V </ci><cn> 10 </cn>
          </apply>
        </apply>
        <apply>
          <minus/>
          <apply>
            <exp/>
            <apply>
              <times/>
              <cn> 0.1 </cn>
              <apply>
                <plus/><ci> V </ci><cn> 10 </cn>
              </apply>
            </apply>
          </apply>
          <cn> 1 </cn>
        </apply>
      </apply>
    </math>
  </assignmentRule>
  ...
  <assignmentRule variable="i_Na">
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      <apply>
        <times/>
        <ci> g_Na </ci>
        <apply>
          <power/><ci> m </ci><cn> 3 </cn>
        </apply>
        <ci> h </ci>
        <apply>
          <minus/><ci> V </ci><ci> V_Na </ci>
        </apply>
      </apply>
    </math>
  </assignmentRule>
```

3.2.3 Simulation of an action potential

Any simulator supporting SBML's assignment and rate rules cab be used to simulate the behavior of the model. The results obtained with SBMLodeSolver (Machné et al, 2006) (see Figure 4 reproduce Figure 12 from the original paper.

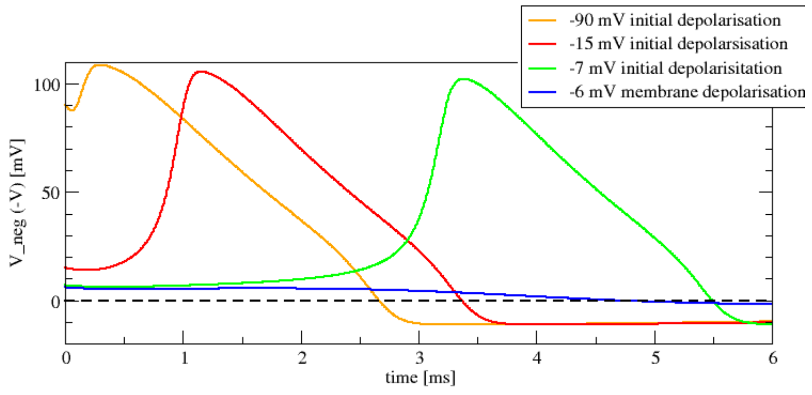

Fig. 4 Simulation output of Hodgkin-Huxley model using SBML ODESolver

3.3 Cortical spiking neurons

While sections 3.1 and 3.2 describe mechanistic models based on experimental measurements of components' properties, whether biochemical or electrophysiological, one can also describe the behavior of neurons by developing phenomenological models. This is particularly useful when one does not possess enough molecular or morphological details about the neuron of interest to reconstruct large-scale neuronal networks. Examples of such models are the spiking neurons, discussed in more details in chapter [REFERENCE TO CHAPTER BY NAUD AND GERSTNER]. Because one can encode any mathematical description in SBML rules, those models can be easily encoded.

In his paper of 2003 (Izhikevich, 2004a), Eugene Izhikevich proposed a simple model of spiking neuron able to reproduce the behavior of many spiking and bursting cortical neurons. In a further paper, he explored the response of this model to various inputs (Izhikevich, 2004b). Here we will construct an SBML version of the model with a choice of parameters that makes it a class 1 excitability neuron, that is a neuron firing at low frequency for lower inputs. The complete model can be found in BioModels Database with the accession BIOMD0000000141.

3.3.1 Model of a spiking neuron

Izhikevitch's model is a 2-dimensional FitzHughNagumo class model (FitzHugh, 1961) described with the following equations:

$$\frac{dv}{dt} = 0.04v^2 + 5v + 140 - u + I$$

$$\frac{du}{dt} = a(b-u) \tag{7}$$

the system being reset when $v \geq 30$ as:

$$\begin{aligned} v &= c \\ u &= u + d \end{aligned} \tag{8}$$

where

v	membrane potential
u	membrane recovery variable
I	input current
a	constant
b	constant
c	constant
d	constant

3.3.2 Encoding a model with conditional assignment

As with the previous example, none of the variables under consideration in the model represent **species** or **compartments**. The SBML encoding of the model defines all these, and a number of other constants as **parameters**. Similarly the model does not contain any reactions, and all the equations are encoded using **rules**. Note in some cases the parameters will be constant. Others must have the *constant* attribute set to ‘false’ as they will be controlled by other elements of the model, in this case through **rules** and **events**.

```
<listOfParameters>
  <parameter id="a"       value="0.02" constant="true"/>
  <parameter id="b"       value="-0.1" constant="true"/>
  <parameter id="c"       value="-55"  constant="true"/>
  <parameter id="d"       value="6"    constant="true"/>
  <parameter id="Vthresh" value="30"   constant="true"/>
  <parameter id="I"       value="0"    constant="false"/>
  <parameter id="flag"    value="0"    constant="false"/>
  <parameter id="v"       value="-60"  constant="false"/>
  <parameter id="u"       value="6"    constant="false"/>
</listOfParameters>
```

Equations 7 can be encoded using **rateRules** as follows.

```
<rateRule variable="v">
  <math xmlns="http://www.w3.org/1998/Math/MathML">
    <apply>
      <plus/>
      <apply>
        <minus/>
```

```
          <apply>
            <plus/>
            <apply>
              <times/>
              <cn> 0.04 </cn>
              <apply>
                <power/> <ci> v </ci> <cn> 2 </cn>
              </apply>
            </apply>
            <apply>
              <times/> <cn> 4.1 </cn> <ci> v </ci>
            </apply>
            <cn type="integer"> 108 </cn>
          </apply>
          <ci> u </ci>
        </apply>
        <ci> i </ci>
      </apply>
    </math>
  </rateRule>

  <rateRule variable="u">
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      <apply>
        <times/>
        <ci> a </ci>
        <apply>
          <minus/>
          <apply>
            <times/>
            <ci> b </ci>
            <ci> v </ci>
          </apply>
          <ci> u </ci>
        </apply>
      </apply>
    </math>
  </rateRule>
```

Equation 8 can be represented by using an **event**.

```
  <event useValuesFromTriggerTime="true">
    <trigger initialValue="true" persistent="false">
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <apply>
          <gt/>
          <ci> v </ci>
          <ci> Vthresh </ci>
        </apply>
      </math>
    </trigger>
    <listOfEventAssignments>
      <eventAssignment variable="v">
        <math xmlns="http://www.w3.org/1998/Math/MathML">
```

```
          <ci> c </ci>
        </math>
      </eventAssignment>
      <eventAssignment variable="u">
        <math xmlns="http://www.w3.org/1998/Math/MathML">
          <apply>
            <plus/>
            <ci> u </ci>
            <ci> d </ci>
          </apply>
        </math>
      </eventAssignment>
    </listOfEventAssignments>
  </event>
```

The Class 1 Excitable neurons can encode the strength of the input into their firing rate. In order to illustrate this, we want to encode an input current that is zero until a certain point and then increases steadily with time. Mathematically, this can be described as a discontinuous function:

$$I(t) = \begin{cases} 0 & t < 30 \\ 0.075(t-30) & t \geq 30 \end{cases} \tag{9}$$

To encode Equation 9, we can use of an **assignmentRule** containing a piecewise construct:

```
    <assignmentRule variable="I">
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <piecewise>
          <piece>
            <cn type="integer"> 0 </cn>
            <apply>
              <lt />
              <csymbol encoding="text"
                       definitionURL="http://www.sbml.org/sbml/symbols/time" />
              <cn type="integer"> 30 </cn>
            </apply>
          </piece>
          <otherwise>
            <apply>
              <times/>
              <cn> 0.075 </cn>
              <apply>
                <minus/>
                <csymbol encoding="text"
                         definitionURL="http://www.sbml.org/sbml/symbols/time" />
                <cn type="integer"> 30 </cn>
              </apply>
            </apply>
          </otherwise>
        </piecewise>
      </math>
    </assignmentRule>
```

3.3.3 Simulation of a firing pattern

We can use any simulator supporting SBML's piecewise assignmentRules to simulate the model and obtain the time course of the different model's variables. The result obtained with MathSBML (Shapiro et al, 2004) as shown in Figure 5 reproduces Figure 1G from the original paper.

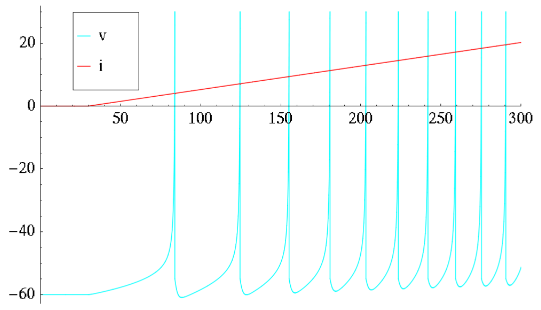

Fig. 5 Simulation of 300 ms of the class I excitable neuron of Izhikevich's model using MathSBML. The monotonic input is shown as well as the resulting increasing frequency of discharge.

3.4 Conclusion and perspectives

For the modeling of biological systems, SBML represented a breakthrough, by enabling for the first time interoperability between modeling and simulation tools. Since the number, size and complexity of computational models in biology has increased in line with the rise of Systems Biology, it is illusory to imagine that people could re-implement models they required, as was the case during the last century. SBML provided a means for researchers to consistently encode, exchange and reuse models. Its creation has revolutionized the modeling process.

A frequent misconception about SBML is that only models of biochemical reactions using chemical kinetic approaches can be encoded. The versatility of the language enables the encoding of a wide diversity of models, either based on processes affecting pools of entities (where the entities are not necessarily biomolecules), or variables described by differential or algebraic equations. The existence of discrete conditional events allows the introduction of perturbations, discontinuous behaviors and event-driven model elements.

Some types of models cannot currently be encoded in SBML, such as compartmental approximations of the cable theory and state-transition representations of ion channels. Other languages, like NeuroML, are more suitable for such a purpose. As this chapter is written, the NeuroML and SBML communities are working conjointly to make those languages interoperable, so that hybrid models containing biochemical and electrophysiological components may be exchanged. Such a cooperation may show the way for other collaborations, for instance with the nascent

NineML language developed under the guidance of the International Neuroinformatics Coordination Facility (INCF) for encoding large neuronal networks.

References

Ausbrooks R, Buswell S, Carlisle D, Dalmas S, Devitt S, Diaz A, Froumentin M, Hunter R, Ion P, Kohlhase M, Miner R, Poppelier N, Smith B, Soiffer N, Sutor R, Watt S (2003) Mathematical markup language (MathML) version 2.0 (second edition). Available via the World Wide Web at http://www.w3.org/TR/MathML2/

Bhalla US, Iyengar R (1999) Emergent properties of networks of biological signaling pathways. Science 283(5400):381–387

Bray T, Paoli J, Sperberg-McQueen CM, Maler E (2000) Extensible markup language (XML) 1.0 (second edition), W3C recommendation 6-October-2000. Available via the World Wide Web at http://www.w3.org/TR/1998/REC-xml-19980210

Cohen P (1992) Signal integration at the level of protein kinases, protein phosphatases and their substrates. Trends in Biochemical Sciences 17(10):408–413

Curto R, O VE, A S, M C (1998) Mathematical models of purine metabolism in man. Math Biosci 151(1):1–49

Elowitz MB, Leibler S (2000) A synthetic oscillatory network of transcriptional regulators. Nature 403(6767):335–338

Fernandez E, Schippa R, Girault J, Novère NL (2006) Darpp-32 is a robust integrator of dopamine and glutamate signals. PLoS Computational Biology 2(12):e176

FitzHugh R (1961) Impulses and physiological states in theoretical models of nerve membrane. Biophys J 1:445–466

Gleeson P, Crook S, Cannon R, Hines M, Billings G, Farinella M, Morse T, Davison A, Ray S, Bhalla U, Barnes S, Dimitrova Y, RA S (2010) Neuroml: A language for describing data driven models of neurons and networks with a high degree of biological detail. PLoS Comput Biol p in the press

Goldbeter A (1991) A minimal cascase model for the mitotic oscillator involving cyclin and cdc2 kinase. Proc Natl Acad Sci USA 88(20):9107–11

Hedley W, Nelson M, Bullivant P DP Nielsen (2001) A short introduction to cellml. Phys Trans R Soc Lond A 359(5):1073–1079

Hodgkin A, Huxley A (1952) A quantitative description of membrane current and its application to conduction and excitation in nerve. J Physiol 117(5):500–544

Hoops S, Sahle S, Gauges R, Lee C, Pahle J, Simus N, Singhal M, Xu L, Mendes P, Kummer U (2006) Copasi–a complex pathway simulator. Bioinformatics 22(24):3067–3074

Hucka M, Finney A, Sauro HM, Bolouri H (2001) Systems Biology Markup Language (SBML) Level 1: Structures and facilities for basic model definitions. Available via the World Wide Web at http://www.sbml.org/Documents/Specifications

Hucka M, Finney A, Sauro HM, Bolouri H, Doyle JC, Kitano H, Arkin AP, Bornstein BJ, Bray D, Cornish-Bowden A, Cuellar AA, Dronov S, Gilles ED, Ginkel M, Gor V, Goryanin II, Hedley WJ, Hodgman TC, Hofmeyr JH, Hunter PJ, Juty NS, Kasberger JL, Kremling A, Kummer U, Le Novère N, Loew LM, Lucio D, Mendes P, Minch E, Mjolsness ED, Nakayama Y, Nelson MR, Nielsen PF, Sakurada T, Schaff JC, Shapiro BE, Shimizu TS, Spence HD, Stelling J, Takahashi K, Tomita M, Wagner J, Wang J (2003) The Systems Biology Markup Language (SBML): A medium for representation and exchange of biochemical network models. Bioinformatics 19(4):524–531

Hucka M, Hoops S, Keating SM, Novère NL, Sahle S, Wilkinson DJ (2008) Systems Biology Markup Language (SBML) level 2: Structures and facilities for model definitions. Available via the World Wide Web at http://www.sbml.org/Documents/Specifications

Hucka M, Bergmann F, Hoops S, Keating SM, Sahle S, Wilkinson DJ (2009) The Systems Biology Markup Language (SBML) language specification for level 3 version 1 core. Available via the World Wide Web at http://www.sbml.org/Documents/Specifications

Izhikevich EM (2004a) Simple model of spiking neurons. IEEE Trans Neural Netw 14(6):1569–1573

Izhikevich EM (2004b) Which model to use for cortical spiking neurons? IEEE Trans Neural Netw 15(5):1063–1070

Kell D, Mendes P (2008) The markup is the model: Reasoning about systems biology models in the semantic web era. J Theor Biol 252:538–543

Köhn D, Le Novère N (2008) Sed-ml - an xml format for the implementation of the miase guidelines. In: Heiner M, Uhrmacher A (eds) Proceedings of the 6th conference on Computational Methods in Systems Biology, Lecture Notes in Bioinformatics, vol 5307, pp 176–190

Le Novère N, Bornstein B, Broicher A, Courtot M, Donizelli M, Dharuri H, Li L, Sauro H, Schilstra M, Shapiro B, Snoep JL, Hucka M (2006) BioModels Database: A free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. Nucleic Acids research 34:D689–D691

Le Novère N, Courtot M, Laibe C (2007) Adding semantics in kinetics models of biochemical pathways. In: Proceedings of the 2nd International Symposium on experimental standard conditions of enzyme characterizations, pp 137–153

Le Novère N, Hucka M, Mi H, Moodie S, Schreiber F, Sorokin A, Demir E, Wegner K, Aladjem MI, Wimalaratne SM, Bergman FT, Gauges R, Ghazal P, Kawaji H, Li L, Matsuoka Y, Villéger A, Boyd SE, Calzone L, Courtot M, Dogrusoz U, Freeman TC, Funahashi A, Ghosh S, Jouraku A, Kim S, Kolpakov F, Luna A, Sahle S, Schmidt E, Watterson S, Wu G, Goryanin I, Kell DB, Sander C, Sauro H, Snoep JL, Kohn K, Kitano H (2009) The systems biology graphical notation. Nature biotechnology 27(8):735–741

Machné R, Finney A, Müller S, Lu J, Widder S, Flamm C (2006) The sbml ode solver library: a native api for symbolic and fast numerical analysis of reaction networks. Bioinformatics 22(11):1406–1407

Manola F, Miller E (2004) RDF primer. Tech. rep., W3C, URL http://www.w3.org/TR/2004/REC-rdf-primer-20040210/

Shapiro BE, Hucka M, Finney A, Doyle J (2004) Mathsbml: a package for manipulating sbml-based biological models. Bioinformatics 20(16):2829–2831

Strömbäck L, Lambrix P (2005) Representations of molecular pathways: an evaluation of sbml, psi ml and biopax. Bioinformatics 21(24):4401–4407

Index