
System Biology Markup Language (SBML) Level 3 extension : Formal models (L3F)

Berenguier Duncan, Le Novère Nicolas
duncan@ebi.ac.uk, lenov@ebi.ac.uk

2008

Contents

I	Introduction	2
1	Why this document	2
2	Logical models	2
3	Petri networks	2
II	Specification proposal	3
4	Nomenclature	3
5	Encoding logical models	4
6	Encoding Petri networks	4
7	Relations tree	5
8	<species> extension	5
9	<listOfTransitions> and <transition>	5
10	<listOfInputs> and <input>	6
11	<listOfOutputs> and <output>	6
12	<listOfActivities> and <activity>	6
III	Examples	6
13	Encoding logicals models	6
13.1	Boolean model	7
13.2	Multivaluated models	8
14	Encoding Petri networks	8

Part I

Introduction

1 Why this document

Quantitative methods for modeling biologicals network require an in-depth knowledge of the biochemical reactions and their stoichiometry and kinetics parameters. For most biologicals networks, this knowledge is missing, therefore many qualitative methods are developed [3], to use different informations such as gene expression data coming from functional genomic experiments for example.

Actually, the number of people using theses kind of methods increase, consequently, the number of (in-compatible) software is increasing too. The aim of this document is to propose a possible package (extension) for the next version of SBML, SBML version 3. This package called L3F should re-use the basics elements of SBML such as species, compartment, SBase... and extend SBML with new elements to encode qualitative models.

For the moment, this proposal focus only on logical models and Petri networks.

2 Logical models

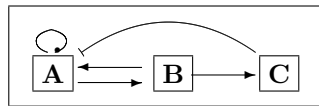


Figure 1: A simple Boolean network

Initiated in the 60's by Kauffman [6], Boolean models describe the different elements of the network with Boolean variables expressing if it is active (1) or inactive (0). Each variable follow a logical formula defined by the interactions with the other elements. Then an element could stimulate or inhibit another.

Figure 1 is showing a simple Boolean network, constituted of 3 genes, A, B and C. The gene A is activated by the gene B and inhibited by C, B is activated by A and C is activated by B. This network could be represented using the following formula : $A := B \wedge \neg C$, $B := A$, $C := B$

The Booleans networks were then generalized to multivaluate logical networks. Theses models allow variables to take more than two values, these values are called activities. The formula describing the value of the variable could be split into smaller logical formulae each describing one activity.

Admitting the node A has 3 possible activities 0, 1 and 2, and the nodes B and C are Boolean, the figure 1 could represent the following formulae :

$$A := \begin{cases} 2 & \text{if } A = 2 \wedge B = 1 \wedge C = 0 \\ 1 & \text{if } (A = 2 \wedge B = 1 \wedge C = 1) \vee (A = 0 \wedge B = 1 \wedge C = 0) \\ 0 & \text{otherwise} \end{cases}, \quad B := A = 1, \quad C := B = 1$$

3 Petri networks

Also initiated in the 60's, by Carl Adam Petri [7, 8], Petri networks, are used to describe discrete distributed systems. They have been successfully used to model biological systems [9, 1, 4, 5, 2].

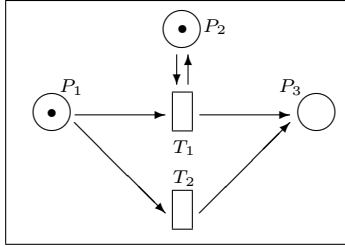


Figure 2: A simple Petri network

A basic Petri net is a directed bipartite graph. They are constituted of 2 kind of nodes, the places noted $P = P_1, P_2, \dots, P_n$, drawn with circles and the transitions noted $T = T_1, T_2, \dots, T_m$, drawn with rectangles. The directed arcs goes from a place to a transition or to a transition to a place, they are drawn with arrows. Each place could carry zero or one token, drawn with a small black filled circle.

The dynamic of the network are modeled with the transitions. The transitions are capable to transfer tokens between places. The transfer called firing occurs when all the places with arcs incoming on the transition are marked with a token and when the out-coming places is not marked, then the out-coming place become marked. They could be multiple out-coming places, then a place should be choose randomly or all the possible choose should be computed. At each iteration of the system, all the transitions try to fire.

There is many extensions to the basic Petri networks, generally called high level Petri networks : the place/transitions networks (P/T nets) allow to have multiples token per places and weight on the arcs, the stochastic Petri net (SPN) add a probability of firing to each transition. The timed Petri networks (TPN) add timers on arcs, places, transitions or tokens, the colored Petri networks (CPN) allow to have multiple type (color) of tokens ...

The figure 2 show a simple example of a basic Petri network. It involve 3 places P_1 , P_2 and P_3 , and 2 transitions T_1 and T_2 . P_1 and P_2 are marked with a token. As there is arcs between theses two places and T_1 , the transition could fired. If it does dire, the tokens from P_1 and P_2 are consumed, and two tokens are produces in P_2 and P_3 . As P_2 is consume and refilled, the arc from P_2 to T_1 is called a read arc. The second transition could fire too. If it does, the token from P_1 is consumed and a token is produce in P_3 . The simulation has then two possibilities, either firing T_1 or T_2 . The simulation should made a choice or firing both into two different simulations.

Part II

Specification proposal

4 Nomenclature

In this document, the xml elements are represented in bold, eg **listOfInputs**. Monospaced font is used for xml attributes, eg **species**. Italic style is used for constant values, eg *0*, *1*, *true*, *false* ...

The mathematical formulae taking place in **math** elements will be written in plain text instead of MathML. This intend to help the readability of the formulae and made the listings a lot more shorter.

5 Encoding logical models

Logical models associate a discrete activity level (0 or 1 in Booleans models) with **species**. A logical formula indicate the level a species can take depending on the species regulating it.

The element **transition** is used to define the set of species (input) regulating one species (output) and to specify the logical formula.

It could contains a **listOfInputs** indicating each regulating species and a **listOfOutputs** indicating the regulated species.

It could contains also a **listOfActivities** to encode the formula. The **activity** element contains a **level** attribute and a **math** element holding a logical formula. If this formula is evaluated to *true*, then the activity of the output species is set to the **activity**'s **level** value. If all the formula encoded in the **activity** elements are evaluated to *false* then, the activity is set to 0 .

In **input** elements, the **sboTerm** attribute should be used to indicates the type of control the input species have on the output species.

If you can not precise the type of control, for example because the control depend on the others species, use *control* (*SBO:0000168*).

If you can precise it, use *inhibition* (*SBO:0000169*), *stimulation* (*SBO:0000170*) or a *stimulation*'s child *necessary stimulation* (*SBO:0000171*) or *catalysis* (*SBO:0000172*). (see SBO tree : <http://www.ebi.ac.uk/sbo/>)

A **transition** could have no inputs, for example to represent an input of the network, eg. a species not regulated.

A **transition** could have several outputs, in this case the activities and the inputs are exactly the same (note the different outputs could have different initial activity).

6 Encoding Petri networks

A simple Petri net is constituted of places and transitions. Places are encoding using **species** while transitions are encoded by **transition**.

The arcs from a place to a **transition** are encoded using **listOfInputs**, and the arcs from a **transition** to a place are encoded using **listOfOutputs**.

The initial number of token in a place is encoded with the **species**' attribute **initialActivityLevel**.

To encode Places/Transitions networks, you can declare a number of token greater than one in the **species**' attribute **initialActivityLevel**.

The weight of inputs arcs is declared with the **input**'s **consume** attribute and the weight of outputs arcs with the **output**'s **produce** attribute.

To encode Stochastic Petri networks, you define a probability a **transition** has to fire. This probability is encoded with the **rate** attribute. It represent the inverse of the average firing time of the transition.

To encode Timed Transitions Petri Networks (TTPN), a delay is associated with a **transition**.

The firing is compose of three phases, first, tokens are consumed from the input places, second, the delay elapses, third, tokens are produced in the output places.

You can encode it using the **transition**'s **delay** attribute.

Inhibitory arcs should be represented as in logical models using the **sboTerm** *inhibition* (*SBO:0000169*).

In like manner, we should create a new ontology in SBO to represent a test arc.

7 Relations tree

Figure 3 show the relations of compositions between the L3F/SBML elements.

The elements **transition**, **activity**, **input** and **output** inherit from SBML element **SBase**. The elements **listOfTransitions**, **listOfActivities**, **listOfInputs** and **listOfOutputs** inherit from **ListOf**, which inherit from **SBase**.

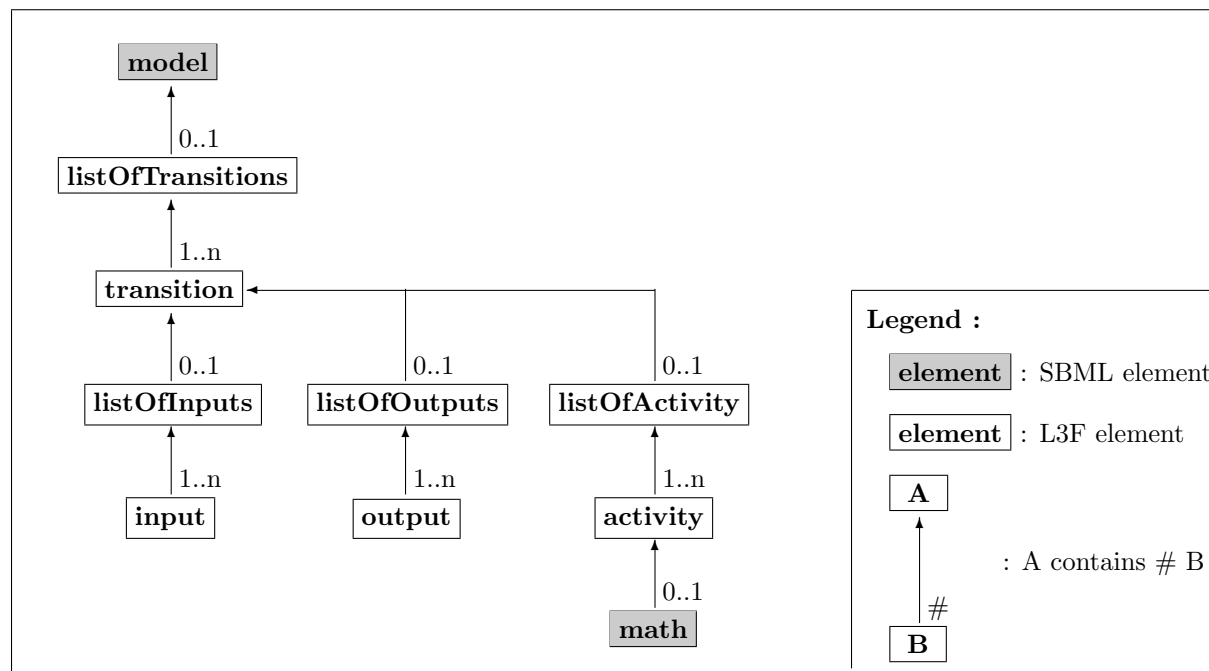


Figure 3: Composition of L3F elements

8 <species> extension

The SBML element **species** is extended with a new attribute **initialActivity**, indicating the node's initial level of activity or the node's number of tokens.

species	
initialActivity	: integer {use="optional", default="0"}

9 <listOfTransitions> and <transition>

The attribute **delay** is used in timed transitions Petri networks to represent the time a token must spend in the transition before being fired.

The attribute **rate** is used in stochastic Petri networks. It correspond to the inverse of the average firing time of the transition.

transition	
delay	: double {use="optional", default="0.0"}
rate	: double {use="optional", default="1.0"}

10 <listOfInputs> and <input>

The attribute **species** references a **species** using its ID.

The attribute **consume** indicates how many token should be removed from the inputs species after the transition's firing. In logical networks, this values should be set to 0 (default).

input	
species	: sIDREF {use="required" }
consume	: integer {use="optional", default="0" }

11 <listOfOutputs> and <output>

The attribute **species** references a **species** using its ID.

The attribute **produce** indicates how many token should be added from the outputs species after the transition's firing. In logical networks, this values should be set to 0 (default).

output	
species	: sIDREF {use="required" }
produce	: integer {use="optional", default="0" }

12 <listOfActivities> and <activity>

The attribute **level** defines the value the outputs transitions must take when the formula (contained by the math element) is evaluated to *true*.

activity	
level	: integer {use="required" }

Part III

Examples

13 Encoding logicals models

The model in figure 1 represent the topology of a simple model.

The listing 1 represents the minimal SBML code to encode the model. It declare a model with one **compartment** called *main*, then 3 **species** *A*, *B* and *C* with the attribute **initialActivityLevel**. The brackets [...] represent the place where the furthers listings will take place.

Listing 1: base declaration example

```
<?xml version="1.0" encoding="UTF-8"?>
```

```

<sbml xmlns="http://www.sbml.org/sbml/level3/version1" level="3" version="1">
  <model id="logical_model_example">
    <listOfCompartments>
      <compartment id="main" name="main" />
    </listOfCompartments>
    <listOfSpecies>
      <species id="A" name="A" compartment="main" initialActivityLevel="1" />
      <species id="B" name="B" compartment="main" initialActivityLevel="0" />
      <species id="C" name="C" compartment="main" initialActivityLevel="0" />
    </listOfSpecies>
    [...]
  </model>
</sbml>

```

13.1 Boolean model

This listing show how to encode the previous graph and the following formula : $A := B \wedge \neg C$, $B := A$, $C := B$

Listing 2: Boolean model

```

<listOfTranstitions>
  <transition>
    <listOfInputs>
      <input species="B" sboTerm="SBO:0000170" /><!-- stimulation -->
      <input species="C" sboTerm="SBO:0000169" /><!-- inhibition -->
    </listOfInputs>
    <listOfOutputs>
      <output species="A" />
    </listOfOutputs>
    <listOfActivities>
      <activity level="1">
        <math>B = 1 \text{ and } C = 0</math>
      </activity>
    </listOfActivities>
  </transition>
  <transition>
    <listOfInputs>
      <input species="A" sboTerm="SBO:0000170" /><!-- stimulation -->
    </listOfInputs>
    <listOfOutputs>
      <output species="B" />
    </listOfOutputs>
    <listOfActivities>
      <activity level="1">
        <math>A = 1</math>
      </activity>
    </listOfActivities>
  </transition>
  <transition>
    <listOfInputs>
      <input species="B" sboTerm="SBO:0000170" /><!-- stimulation -->
    </listOfInputs>
    <listOfOutputs>
      <output species="C" />
    </listOfOutputs>
    <listOfActivities>
      <activity level="1">
        <math>B = 1</math>
      </activity>
    </listOfActivities>
  </transition>
</listOfTranstitions>

```

13.2 Multivaluated models

This listing show how to encode the previous graph and the following formulae : $B := A = 1, C := B = 1,$

$$A := \begin{cases} 2 & \text{if } A = 2 \wedge B = 1 \wedge C = 0 \\ 1 & \text{if } (A = 2 \wedge B = 1 \wedge C = 1) \vee (A = 0 \wedge B = 1 \wedge C = 0) \\ 0 & \text{otherwise} \end{cases}$$

Listing 3: multivaluated model

```
<listOfTranstitions>
  <transition>
    <listOfInputs>
      <input species="A" sboTerm="SBO:0000170" /><!-- stimulation -->
      <input species="B" sboTerm="SBO:0000170" /><!-- stimulation -->
      <input species="C" sboTerm="SBO:0000169" /><!-- inhibition -->
    </listOfInputs>
    <listOfOutputs>
      <output species="A" />
    </listOfOutputs>
    <listOfActivities>
      <activity level="2">
        <math>A = 2 \text{ and } B = 1 \text{ and } C = 0</math>
      </activity>
      <activity level="1">
        <math>(A = 2 \text{ and } B = 1 \text{ and } C = 1) \text{ or } (A = 0 \text{ and } B = 1 \text{ and } C = 0)</math>
      </activity>
    </listOfActivities>
  </transition>
  <transition>
    <listOfInputs>
      <input species="A" sboTerm="SBO:0000170" /><!-- stimulation -->
    </listOfInputs>
    <listOfOutputs>
      <output species="B" />
    </listOfOutputs>
    <listOfActivities>
      <activity level="1">
        <math>A = 1</math>
      </activity>
    </listOfActivities>
  </transition>
  <transition>
    <listOfInputs>
      <input species="B" sboTerm="SBO:0000170" /><!-- stimulation -->
    </listOfInputs>
    <listOfOutputs>
      <output species="C" />
    </listOfOutputs>
    <listOfActivities>
      <activity level="1">
        <math>B = 1</math>
      </activity>
    </listOfActivities>
  </transition>
</listOfTranstitions>
```

14 Encoding Petri networks

The model in figure 2 is the representation of a simple Stochastic Petri network.

The listing 4 represents the minimal SBML code to encode the model. It declare a model with one **com-**

partment called *main*, then 3 species P_1 , P_2 and P_3 with the attribute `initialActivityLevel`.

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level3/version1" level="3" version="1">
  <model id="petri_net_model_example">
    <listOfCompartments>
      <compartment id="main" name="main" />
    </listOfCompartments>
    <listOfSpecies>
      <species id="P1" name="P1" compartment="main" initialActivityLevel="1" />
      <species id="P2" name="P2" compartment="main" initialActivityLevel="1" />
      <species id="P3" name="P3" compartment="main" initialActivityLevel="0" />
    </listOfSpecies>
    <listOfTransitions>
      <transition id="T1" name="T1" rate="0.87">
        <listOfInputs>
          <input species="P1" consume="1" />
          <input species="P2" consume="1" />
        </listOfInputs>
        <listOfOutputs>
          <output species="P2" produce="1" />
          <output species="P3" produce="1" />
        </listOfOutputs>
      </transition>
      <transition id="T2" name="T2" rate="0.23">
        <listOfInputs>
          <input species="P1" consume="1" />
        </listOfInputs>
        <listOfOutputs>
          <output species="P3" produce="1" />
        </listOfOutputs>
      </transition>
    </listOfTransitions>
  </model>
</sbml>
```

References

- [1] C. Chaouiya. Qualitative modelling of genetic networks: From logical regulatory graphs to standard petri nets. *LNCS*, 3099(137-156), 2004.
- [2] J-P Comet. Modeling multi-valued genetic regulatory networks using high-level petri nets. *LNCS*, 3536:208–227, June 2005.
- [3] Hidde de Jong. Modeling and simulation of genetic regulatory systems: A literature review. *Journal of Computational Biology*, 9(1):67–103, 2002.
- [4] E. Simão. Qualitative modelling of regulated metabolic pathways: Application to the tryptophan biosynthesis in e.coli. *Bioinformatics*, 21:190–196, 2005.
- [5] M. Heiner. Analysis and simulation of steady states in metabolic pathways with petri nets. *CPN'01*, pages 15–34, 2001.
- [6] S. A. Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biology*, 22(3):437–467, 1969.
- [7] C. A. Petri. Fundamentals of a theory of asynchronous information flow. *IFIP Congress*, pages 386–390, 1962.

- [8] C. A. Petri. Concepts of net theory. *MFCS*, pages 137–146, 1973.
- [9] Reddy V.N. Qualitative analysis of biochemical reaction systems. *Comput. Biol. Med.*, 26(1):9–24, Jan 1996.