Core SBML and its Formal Semantics

Joachim Niehren, Cédric Lhoussaine, Vaginay Ahténaïs

Core SBML and its Formal Semantics

Joachim Niehren[2,1], Cédric Lhoussaine[1], and Athénaïs Vaginay[3]

[1] BioComputing Team of CRIStAL Lab, Université de Lille
[2] Inria Center of the University of Lille
[3] Université de Lorraine, CNRS, CRAN & LORIA, F-54000 Nancy

Abstract. The systems biology markup language (SBML) permits to represent biological models mixing reaction networks, algebraic equations, differential equations, and events. Its main objective is to exchange biological models between various tools for simulation and analysis. The specification of SBML, however, lacks a formal semantics. This makes it often difficult to understand SMBL models and to design correct and general interfaces with SBML. In the present paper, we propose Core SBML, a novel language covering a large subset of SBML with clear formal semantics.We present a compiler of the delay-free fragment of SBML to Core SBML (without any formal correctness guarantees). We then show how to compile Core SBML further to BioCham while preserving the semantics. We implemented and applied our compilers to the more than 500 SBML models from the curated part of the BioModels database.

1 Introduction

The three most prominent formalisms to model the quantitative dynamical evolution of biological systems over time are differential equations, reaction networks [7], and hybrid networks [9]. If all parameters and initial values are known, then such models can be simulated numerically over time by some variant of the Euler's algorithm for deterministic simulation [4]. In practice, this is supported by various tools, including Copasi [10] and BioCham [3,6].

All the formalisms mentioned above can be integrated into a unified framework, as provided by the Systems Biology Markup Language (SBML) [14]. Each of these formalisms has variables for denoting real-valued functions. These values of these variables can be constrained in different manners, by chemical reactions, differential equations, algebraic equations, and events. For reaction networks, the *species* of chemical reactions are considered as variables for the species' concentrations. These must satisfy the ordinary differential equations (ODEs) inferred from the reaction network. These ODEs specify how the concentrations of the species evolve continuously over real time according to the kinetic laws of the reactions. In hybrid networks, the values of variables can be changed by assignments triggered by some events. An event is triggered at the first time points when some logical conditions becomes valid. In mathematical models, variables may be constrained by algebraic equations and differential equations respectively. When allowed to combine the four kinds of constraints for variables (reactions, events, algebraic equations, differential equations) and to use not only species but all kinds of variables in kinetic laws of reactions, then the integrated framework is obtained.

The concrete *syntax* of SBML is an instance of the Extendable Markup Language (XML) that is formally defined by an XML schema. However, the *semantics* of SBML models lacks a formal definition. In particular, its advanced concepts such as boundaryConditions, assignment rules, rate rules, and events with and delays are often difficult to interpret unambiguously. In such cases, most users (even experts) have to consult the equations that are inferred from by some tool (e.g. Copasi [**?**]) in order to understand the meaning of a given model. The lack of a formal semantics that is independent of any implementation also makes it difficult to design correct interfaces with large coverage between various tools in systems biology, while using SBML as an exchange format. It is also not clear how to specify fragments of SBML that can be compiled to other languages such as BioCham. Furthermore, the correctness of such compilers cannot even be stated formally, given that SBML doesn't come with a formal semantics.

In the present paper, we propose Core SBML, a novel language covering a large subset of SBML, while coming with a clear formal semantics. We base Core SBML on an abstract term syntax, enabling the definition of the formal semantics. We also provide a concrete XML syntax of Core SBML following its abstract syntax. As a consequence the concrete syntax of Core SBML becomes different from that of SBML. We define the XML syntax of Core SBML by an XML schema that is composed of a DTD and a Schematron [11] (rather than in RelaxNG [24] or XML Schema [22] as for SBML on level 3 and respectively level 2). All Core SBML networks that we produced were validated with respect to our XML schema.

Second, we present a compiler of a large fragment of SBML to Core SBML. Besides some tedious semantic aspects, a trick is needed to deal with r¡compartment volumes in full generality: we introduce algebraic variables for concentrations of species, and link them to variables for the amount of species via an algebraic equation. A formal correctness result cannot be expected, given that SBML lacks a formal semantics. We have implemented our compiler and applied it to all SBML models of the curated part of the BioModels database [16]. It turns out that the only aspects of SBML models that cannot be translated by our compiler are delays in differential equations and events. These occur in only 19 out of the 547 SBML networks, though. Adding delays to Core SBML is not difficult syntactically, but would require further efforts in semantics and implementation.

The third contribution is a compiler from Core SBML to BioCham. The abstract syntax of BioCham is a restriction of that of Core SBML. In particular, BioCham does neither support delays, nor in differential equation nor in events. BioCham's concrete syntax is not based on XML. Its conciseness increases the human readability but makes automatic processing more laborious. One concept missing in BioCham compared to Core SBML are differential variables, but these can be easily converted to species variables. Another difference is that BioCham does not permit to specify initial values of species by arithmetic expressions in dependence of the initial values of other species. Therefore, our compiler has to evaluate the arithmetic expressions for initial values in Core SBML networks statically. Finally, some arithmetic functions such as tanh are built in (Core) SBML, but need to be encoded explicitely in BioCham. In this way, our compiler from Core SBML to BioCham preserves the differential algebraic equations and the initial values up to logical equivalence. It also preserves the events. This implies

its correctness with respect to the semantics of both languages. It should be noted that BioCham has its own function for importing SBML models and converting them into reaction networks in BioCham's format. A major difference to our compiler is that BioCham's import ignores compartment volumes all over, which is correct only if all volumes are equal to 1.

We implemented both compilers in the XML Stylesheet Transformation Language (XSLT) [25] with the Saxon programming tool [13], and started testing it by comparing numerical simulations. We made all Core SBML and BioCham networks produced by our compilers from the SBML networks in the curated part of the BioModels database available at http://researchers.lille.inria.fr/niehren/Core-SBML.

Related Work. Reaction networks lay the foundation of all quantitative modeling frameworks in systems biology. They may be given different semantics besides the deterministic continuous based on ODEs [5]. The non-deterministic rewrite semantics [15] ignores the kinetic expressions, while the stochastic semantics uses them differently, for computing the probability of a reaction to happen. Reaction networks also have a Boolean semantics which abstracts from the rewrite semantics (rather than the ODE semantics [18]) . This Boolean semantics serves in BioCham for model verification.

Qualitative logical reasoning in systems biology is usually supported by Boolean networks [8,21,17,20]. Compilers from reaction networks to Boolean networks with correctness guarantees with respect to the ODE semantics were considered in [18] and without in [23]. Alternatively, reaction networks with partial kinetic information on reactions on inhibitors and activators were considered for quantitative reasoning [1,19,12]. SBML level 3 models can support some kinds of partial kinetic information too. In the present paper, however, we consider only models with complete kinetic information, as supported by SBML level 2 models. Neither do we consider models with missing kinetic parameters or missing initial concentrations.

Outline. After preliminaries on arithmetic expressions and algebraic differential equations in Section 2, we discuss SMBL informally in Section 3. The language of Core SBML is contributed in Section 4. A compiler of a large fragment of SBML to Core SBML is described in Section 5. The compiler to BioCham is discussed in Section 6. The appendix contains supplementary material, including the Core SBML and BioCham networks obtained by our compilers from three SBML models in the BioModels database.

Table of Contents

2 Preliminaries

Let $\mathbb{B} = \{0, 1\}$ be the set of Booleans, $\mathbb{N}$ the set of natural numbers including 0, $\mathbb{R}$ the set of real numbers, and $\mathbb{R}_+$ the set of positive real numbers including 0. Note that $\mathbb{B} \subseteq \mathbb{N} \subseteq \mathbb{R}_+ \subseteq \mathbb{R}$.

A partial function $g : S \hookrightarrow T$ is a binary relation $g \subseteq S \times T$ that is functional, i.e., for all $s \in S$ there exists at most one $t \in T$ such that $(s, t) \in g$. For any partial function g, we write $g(s) = t$ if and only if $(s, t) \in g$. The domain of a partial function is $dom(g) = \{s \in S \mid \exists t \in T.\ g(s) = t\}$. A total function $g : S \to T$ is a partial function $g : S \hookrightarrow T$ with $dom(g) = S$. The restriction of g to some $S' \subseteq dom(g)$ is written as $g_{|S'}$.

A substitution $g = [s_1/t_1, \ldots, s_n/t_n]$ is the finite function with domain $\{s_1, \ldots, s_n\}$ such that $g(s_i) = t_i$ for all $1 \leq i \leq n$. We also write $g' = g[s/t]$ for the function with $g'(s) = t$ and $g'(s') = g(s')$ for all $s' \neq s$. If $s_1, \ldots, s_n \in \mathbb{R}_+$ then we sometimes write the substitution g as a formal sum: $g = s_1 t_1 + \ldots + s_n t_n$. Such functions can be seen as chemical solutions that contain species t_i at concentration s_i for all $1 \leq i \leq n$ and all other species at concentration 0. If $n = 0$, then g is the empty chemical solution that we denote by $\mathbf{0}$.

2.1 Arithmetic and Boolean Expressions

We introduce arithmetic and boolean expressions in mutual recursion since boolean expression may occur in the conditionals of arithmetic expressions, while arithmetic expressions may occur in the comparisons of boolean expressions. In SBML and BioCham, arithmetic expressions are used in kinetic laws of reactions, while boolean expressions serve as conditions of events. But they are well-known from various programming languages, so clearly of general interest beyond systems biology.

We assume a set of variables $x \in \mathcal{V}$ and a collection $F = (F^{(n)})_{n \in \mathbb{N}}$ with sets of function symbols $f \in F^{(n)}$ each requiring n arguments. The set of arithmetic expressions $e \in \mathcal{E}_\mathcal{V}$ and boolean expressions $b \in \mathcal{B}_\mathcal{V}$ are the set of all terms with the following abstract syntax:

$$
\begin{array}{rcll}
e, e', e_1, \ldots, e_n \in \mathcal{E}_\mathcal{V} & ::= & x \mid k \mid & \text{where } x \in \mathcal{V},\ k \in \mathbb{R}, \\
& & \mid\ e + e' \mid e - e' \mid ee' \mid e/e' & \\
& & \mid\ f(e_1, \ldots, e_n) & f \in F^{(n)}, n \in \mathbb{N} \\
& & \mid\ \textbf{if } b \textbf{ then } e \textbf{ else } e' & \\
b, b' \in \mathcal{B}_\mathcal{V} & ::= & e \leq e' \mid \neg b \mid b \wedge b' &
\end{array}
$$

The set of free variables $fv(e)$ is the set of all those variables that occur in e. The operators of addition +, multiplication ee', and division / can be interpreted over the relational structure of the real numbers as usual. We will formally do it in Fig. 1. Note that division by zero is undefined. For any symbols $f \in F^{(n)}$ we assume a real-valued function $f^{\mathbb{R}^n \to \mathbb{R}} : \mathbb{R}^n \to \mathbb{R}$ defined elsewhere. This permits to have other real-valued functions such as exponentiation $\sqrt{.}$, $.^2$, exp, sin, cos, etc. For instance, we can define for any arithmetic expression v_1, v_2, J_1, J_2 another arithmetic expression as follows:

$$
goldbeter\_koshland(v_1, v_2, J_1, J_2) =_{\text{def}} \frac{2v_1 J_2}{v_2 - v_1 + J_1 v_2 + J_2 \ \ v_1 + \sqrt{(v_2 - v_1 + J_1 v_2 + J_2 v_1)^2 - 4(v_2 - v_1)\ v_1 J_2}}
$$

$$
\begin{array}{ll}
[\![x]\!]^\eta = \eta(x) & [\![e + e']\!]^\eta = [\![e]\!]^\eta +^{\mathbb{R}} [\![e']\!]^\eta \\
[\![k]\!]^\eta = k & [\![e - e']\!]^\eta = [\![e]\!]^\eta -^{\mathbb{R}} [\![e']\!]^\eta \\
[\![ee']\!]^\eta = [\![e]\!]^\eta *^{\mathbb{R}} [\![e']\!]^\eta & [\![e/e']\!]^\eta = \begin{cases} [\![e]\!]^\eta /^{\mathbb{R}} [\![e']\!]^\eta & \text{if } [\![e']\!]^\eta \neq 0 \\ \mathit{undef} & \text{otherwise} \end{cases}
\end{array}
$$

$$
[\![f(e_1, \ldots, e_n)]\!]^\eta = f^{\mathbb{R}^n \to \mathbb{R}}([\![e_1]\!]^\eta, \ldots, [\![e_n]\!]^\eta)
$$

$$
[\![\textbf{if } b \textbf{ then } e \textbf{ else } e']\!]^\eta = \begin{cases} [\![e]\!]^\eta & \text{if } [\![b]\!]^\eta = 1 \\ [\![e']\!]^\eta & \text{if } [\![b]\!]^\eta = 0 \\ \mathit{undef} & \text{otherwise} \end{cases}
$$

$$
[\![e \leq e']\!]^\eta = \begin{cases} 1 & \text{if } [\![e]\!]^\eta \leq^{\mathbb{R}} [\![e']\!]^\eta \\ 0 & \text{if } [\![e]\!]^\eta >^{\mathbb{R}} [\![e']\!]^\eta \\ \mathit{undef} & \text{otherwise} \end{cases}
$$

$$
[\![b \wedge b']\!]^\eta = [\![b]\!]^\eta \wedge^{\mathbb{B}} [\![b']\!]^\eta \qquad [\![\neg b]\!]^\eta = \neg^{\mathbb{B}}([\![b]\!]^\eta)
$$

Fig. 1: Interpretation of arithmetic and boolean expressions over the reals in environments $\eta : \mathcal{V} \hookrightarrow \mathbb{R}$.

Arithmetic expressions support conditionals **if** b **then** e **else** e' which use a boolean expression b as condition and arithmetic subexpressions e and e' as branches. The set of free variables of boolean expression $fv(b)$ is the set of those variables that occur in b. A boolean expression can test two arithmetic expressions for inequality $e \leq e'$ while being closed under the boolean operators of conjunction $b \wedge b'$ and negation $\neg b$. Equality and strict inequality tests can be defined as boolean expressions too:

$$
e \mathrel{\overset{\circ}{=}} e' =_{\text{def}} e \leq e' \wedge e' \leq e \qquad \text{and} \qquad e < e' =_{\text{def}} e \leq e' \wedge \neg e \mathrel{\overset{\circ}{=}} e'
$$

Furthermore, we note that disjunctions $b \vee b'$ can be defined as usual by $\neg(\neg b \wedge \neg b')$.

2.2 Interpretation over the Reals

Let $\eta : \mathcal{V} \hookrightarrow \mathbb{R}$ be a real-valued variable assignment. For any arithmetic expression e with $fv(e) \subseteq dom(\eta)$ we can define its interpretation $[\![e]\!]^\eta$ as a real number or undefined in Fig. 1

Undefineness comes from undefined operations such as the division by zero mentioned previously. When interpreting an expression, the value *undef* for undefinedness propagates recursively. In mutual recursion, we define in Fig. 1 for any boolean expression b with $fv(b) \subseteq dom(\eta)$ a boolean interpretation $[\![b]\!]^\eta$ which may also be undefined.

2.3 Interpretation as Real-Valued Functions over Time

Since dynamical systems evolve over time, we are interested in interpreting arithmetic expressions over real-valued functions over time. We now define such an interpretation.

Let $T \subseteq \mathbb{R}_+$ be a subset of time points. For any variable assignment $\alpha : \mathcal{V} \hookrightarrow (T \to \mathbb{R})$, we can interpret any arithmetic expression e with $fv(e) \subseteq dom(\alpha)$ to a real-valued function of type $[\![e]\!]^\alpha : T \to \mathbb{R}$ or remain undefined. The interpretation of boolean expressions b as functions $[\![e]\!]^\alpha : T \to \mathbb{B}$ from time points to Booleans works in analogy.

For this, we define any time point $t \in T$ the real-valued assignments $\alpha_t : \mathcal{V} \hookrightarrow \mathbb{R}$ such that $\alpha_t(x) = \alpha(x)(t)$. The interpretations over real functions then satisfies for all time points $t \in T$:

$$[\![e]\!]^{\alpha}(t) = [\![e]\!]^{\alpha_t} \qquad \text{and} \qquad [\![b]\!]^{\alpha}(t) = [\![b]\!]^{\alpha_t}.$$

Suppose that we are given a variable *time* whose interpretation $\alpha(time)$ is fixed as the identity function on $\mathbb{R}_+$, so that for all time points $t \in \mathbb{R}_+$:

$$\alpha(time)(t) = t$$

Arithmetic expressions can be used to define piecewise real-valued functions such as:

$$\textbf{if } 0 \leq time \wedge time \leq 2.5 \textbf{ then } 1.3\ time \textbf{ else } 0$$

In MathML as used by SBML, the same arithmetic expression can be defined.

The conditional **if** $0 \leq time \wedge time \leq 2.5$ **then** $1.3\ time$ **else** 0 can be defined as follows in MathML and thus SBML:

```
<piecewise>
  <piece>
    <apply>
      <mult/>
      <cn> 1.3 </cn>
      <csymbol definitionURL="http://www.sbml.org/sbml/symbols/time"/>
    </apply>
    <apply>
      <and/>
      <apply>
        <leq/>
        <cn> 0 </cn>
        <csymbol definitionURL="http://www.sbml.org/sbml/symbols/time"/>
      </apply>
      <apply>
        <leq/>
        <csymbol definitionURL="http://www.sbml.org/sbml/symbols/time"/>
        <cn> 2.5 </cn>
      </apply>
    </apply>
  </piece>
  <otherwise>
    <cn> 0 </cn>
  </otherwise>
</piecewise>
```

2.4 Differential Algebraic Equation Systems

We recall the syntax and semantics of systems of algebraic and differential equations. We use the standard framework of first-order logic. A differential algebraic system of

equations is a first-order formula with the following abstract syntax, where $x \in \mathcal{V}$ and $e \in \mathcal{E}_{\mathcal{V}}$:

$$S, S' \ ::= \ x \mathrel{\mathring{=}} e \mid \dot{x} \mathrel{\mathring{=}} e \mid S \wedge S' \mid \exists x.S$$

We can define inequations $S \geq 0$ by the equation system $\exists x.\ S \mathrel{\mathring{=}} xx$. An equation system is called algebraic if it contains no differential equation $\dot{x} \mathrel{\mathring{=}} e$. It is called an ordinary differential equation (ODE) if it contains no algebraic equation $x \mathrel{\mathring{=}} e$.

Let $T \subseteq \mathbb{R}_+$ and $\alpha : \mathcal{V} \hookrightarrow (T \to \mathbb{R})$. We call α a solution of the algebraic equation $x \mathrel{\mathring{=}} e$ if $\alpha(x) = [\![e]\!]^\alpha$. We call α a solution of a differential equation $\dot{x} \mathrel{\mathring{=}} e$ if $\dot{\alpha(x)} = [\![e]\!]^\alpha$. Note that the derivative of $\alpha(x)$ may be undefined, in which case the above equation does not hold. We call α a solution of $S \wedge S'$ if it is a solution of both S and S'. And finally, α is a solution of $\exists x.S$ if there exists a function $f : D \to \mathbb{R}$ such that $\alpha[x/f]$ is a solution of S.

For algebraic equations $x \mathrel{\mathring{=}} e$, we can define real-valued solutions $\eta : \mathcal{V} \hookrightarrow \mathbb{R}$ such that $\eta(x) = [\![e]\!]^\eta$. Note however, that the structure of real numbers cannot give interpretation to the derivative operator, so real-valued solutions do not make sense for differential equations.

2.5 Adding Delays

For getting delay differential equations, it is sufficient to extend the set of arithmetic expressions $\mathcal{E}_{\mathcal{V}}$ with a delay operator. The abstract syntax becomes:

$$e \in \mathcal{E}_{\mathcal{V}}(delay) ::= \ldots \mid delay_k(e) \qquad \text{where } k \in \mathbb{R}_+$$

The semantics over real-valued functions is such that $[\![delay_k(e)]\!]^\alpha(t) = [\![e]\!]^\alpha(t - k)$.

3 SBML

The systems biology markup language (SBML) is widely used to represent models of dynamical systems in systems biology. SBML models subsume chemical reaction networks, algebraic equations, differential equations, and events. While systems biology is the main application domain of SBML, the scope of dynamical systems and thus SMBL is way larger (economics, weather forcast, etc). The BioModels database [**?**] provides more than 500 curated SBML models for biological systems online, which were often introduced in research papers. The prime tool for the numerical simulation of SBML models is Copasi [**?**] but many other tools in systems biology have SBML interfaces.

In the present paper, we focus on complete SBML models that can be simulated numerically. Such models are available from SBML level 2 [**?**]. SBML levels correspond to upward-compatible specifications that add features and expressive power. Partial descriptions of complete models were added on level 3 of SBML [**?**]. These are out of the scope of the present paper.

Inside a level, successive versions supersed one another. Each version has an XML syntax that is defined by some schema for XML documents. An SBML model is an XML document that is valid for the schema of the considered version. The schemas of

the different versions of SBML on level 2 are defined in the language XML Schema [**?**], while the schemas of the versions of SBML on level 3 are defined in language RelaxNG [24].

We now present the concepts of the language SBML on level 2 informally, while focussing on their semantics. A formal semantic of SBML models, however, is missing. This is problematic since even experts often have difficulties to understand the precise meaning of an SBML model. The semantics is described by mixing algebraic and differential equations with events. Tools like Copasi [10] and SBML2LaTeX [**?**] permit to infer these equations from SBML models. These informal description of how this works may be insufficient to predict the precise equations manually.

SBML models describe the values of a finite set $\mathcal{V}$ of variables, such that each variable $x \in \mathcal{V}$ denotes some real valued function $[\![x]\!] : \mathbb{R}_+ \to \mathbb{R}$. The variables are constrained in different manners, by reactions, assignment rules, rate rules, and events.

Let e be an arithmetic expression in $\mathcal{E}_{\mathcal{V}}(delay)$. Assignment rules $x \stackrel{\circ}{=} e$ make a direct assignment to the variable values, while rate rules $\dot{x} \stackrel{\circ}{=} e$ make a direct assignment to the rate of change of the variable x, denoted $\dot{x}$. The simplest events have the form $b \Rightarrow x := e$ where $b \in \mathcal{B}_{\mathcal{V}}$, $x \in \mathcal{V}$ and $e \in \mathcal{E}_{\mathcal{V}}$. They change the value of a variable x to the current value of e at all earliest time point when condition b becomes true. But there are more complex forms of event with delays. Even events with priorities are pemitted, but these do not occur in the curated BioModels.

Variables may stand for various objects including kinetic parameters and compartment volumes. Some of the variables have a special role as they store the *amount* of the species of interest. In addition, for each such species variable $x \in \mathcal{V}$ there is a joined *concentration* variable denoted by $[x] \in \mathcal{V}$, whose semantics is defined implicitly, as the amount of the species divided by the volume c of the compartment in wich the species is located, so $[x] \stackrel{\circ}{=} x/c$. Reactions can only change the amount of species variables. They have the form:

$$k_1 x_1 + \ldots + k_n x_n \stackrel{e}{\to} k'_1 y_1 + \ldots + k'_m y_m$$

where $k_i, k'_i \in \mathbb{R}_+$, $x_i, y_i \in \mathcal{V}$, and $n, m \in \mathbb{N}$, while e is an arithmetic expression in $\mathcal{E}_{\mathcal{V}}(delay)$. Species on the left of the arrows are refered to as reactants and those on the right as products. Those species that appear in e but those amount is not changed by reaction are refered to as modifiers (of which some can be distinguished explicitely).

If the same variable is changed both by some rules and by some reactions, then the priority is given to the rules. Finally, all variables are given an initial values, either directly, or by the mean of an assignment rule.

In the present paper, SBML models are represented by graphically. The graphs are produced by converting SMBL to Core SBML and the applying BioComputing's Network-Graph tool http://researchers.lille.inria.fr/ niehren/BioComputing/Network-Graph/doc.html. The newest version can chose x-y-coordinates automatically. But for producing nicer graphs in the paper, we have improved the layout semi-automatically for the examples with yet another tool of the BioComputing group.

In the graphs of SBML models (see e.g. Fig. 2) circles represent species variables and gray boxes represent reactions. Variables identifiers for species amount and concentration are in red, as well as the reaction name. Other parameters are in black. The kinetic expression of a reaction is annotated to its box. Its modifiers are indicated by dash edges.

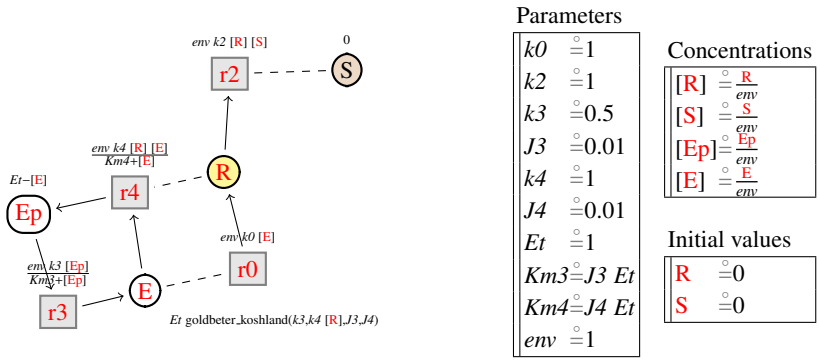

Fig. 2: A graphical illustration of the SBML model B309 of the BioModels database. The values of the species in the white circles are defined by algebraic equations $Ep \mathrel{\overset{\circ}{=}} Et - [E]$ and $E \mathrel{\overset{\circ}{=}} goldbeter\_koshland(k3, k4\ conc_R, J3, J4)$ from assignment rules. Therefore, the species Ep and E are to be ignored as reactants and products of reactions $r3$ and $r4$. Species S has a default rate 0 yielding the ODE $\dot{S} \mathrel{\overset{\circ}{=}} 0$. The kinetic law of reaction $r4$ is $\frac{env\ k3\ [Ep]}{Km3+[Ep]}$. The parameters there are defined by the algebraic equations on the right. The dashed lines indicate modifiers of a reaction.

Plain directed edges from a species to a reaction, and from a reaction to a specie encode the reactants and the product of the reaction, respectively. These edges are annotated with the stoichiometric coefficients of the reaction k_i, k_i' if these are different from 1.

The products and reactants of reactions are drawn in yellow circles. A white circles indicates that the value of the corresponding species is defined by some assignment rules. The expression that is assigned is annotated to the circle. Similarly, a brown circle indicate that the rate of the variable is directly dependent of a rate rule. Here, the rate is annotated to the circle.

The SMBL model B309 of the BioModels database (called BIOMOD00000309 there) is presented graphically in Fig. 2. This model is a toy example of homeostasis with negative feedback. It involves four reactions *r0*, *r2*, *r3* and *r4*, ten parameters and four species (*E*, *Ep*, *R* and *S*). All species belong to the same compartment *env* with fixed volume 1. Only the value of the species *R* is actually modified by reactions. All the other species are in fact change by rules: assignment rule for *Ep* and *E* and a rate rule for $\dot{S}$. These species are thus to be ignored as reactant and product of the reactions, even if they are listed such. Graphically, this is reflected by having a yellow circle for *R*, white circles for *E* and *Ep*, and a brown circle for *S*.

4 Core SBML

Reaction networks of Core SBML also contain reactions, equations and events. A main difference to SMBL is that each variable has a unique type, so that it cannot be constrained in contradictory manner a priori. So no priorities need to be formulated to rule out conflicting definitions.

In Core SBML it is optional to relate a species variable to some variable for its concentration and to another variable for its compartment volume. In SBML, in contrast, this is more mandatory. In case that the option is taken, the algebraic equation defining the concentration from the amount of the species and the volume must be stated explicitly in Core SBML. The consistency of this algebraic equation is then required by the schema of Core SBML networks.

4.1 Abstract Syntax and Semantics

We start the formal definition with a set $\mathcal{V} = \mathcal{S} \uplus \mathcal{V}_{\text{alg}} \uplus \mathcal{V}_{\text{diff}}$ with three disjoints sets of variables: species in $\mathcal{S}$, algebraic variables in $\mathcal{V}_{\text{alg}}$, and differential variables in $\mathcal{V}_{\text{diff}}$.

Definition 1. *Let $\mathcal{S} \subseteq \mathcal{V}$ be finite sets. A reaction of Core SBML with species in $\mathcal{S}$ and variables in $\mathcal{V}$ is a triple (R, P, e) where $R, P : \mathcal{S} \to \mathbb{R}_+$ and $e \in \mathcal{E}_{\mathcal{V}}$.*

We denote reactions as $R \xrightarrow{e} P$ instead of (R, P, e). The expression $e \in \mathcal{E}_{\mathcal{V}}$ is called the kinetic law of the reaction. The species $A \in \mathcal{S}$ with $R(A) > 0$ are called the reactants of the reaction and the species with $P(A) > 0$ are called its products. In contrast to SBML, the reactants and products in Core SBML do only permit variables in $\mathcal{S}$. Other variables in $\mathcal{V} \setminus \mathcal{S}$ are ruled out since these are constrained by equations or events. Furthermore, delays are not permitted in arithmetic expressions. An example for a Core SBML reaction with mass-action kinetics is:

$$2A + B \xrightarrow{2.1xA^2B} A + 2.1C$$

where $A, B, C \in \mathcal{S}$ and $x \in \mathcal{V} \setminus \mathcal{S}$. This reaction has the reactants A with coefficient 2 and B with coefficient 1. Its products are A with coefficient 1 and C with coefficient 2.1. Note that the kinetic expression does not only contain the reactants A and B but also the variable $x \in \mathcal{V} \setminus \mathcal{S}$, which can be used to model the volume of the compartment in which A and B reside.

Any species variable $x \in \mathcal{S}$ denotes a positive real numbers, which stands for the species' amount. All other kinds of variables $y \in \mathcal{V} \setminus \mathcal{S}$ denote arbitrary real numbers. Algebraic variables $z \in \mathcal{V}_{\text{alg}}$ can be used to define compartment values and species concentrations.

Definition 2. *Let $\mathcal{V}_{alg} \subseteq \mathcal{V}$. An event E with variables in $\mathcal{V}$ and algebraic variables in $\mathcal{V}_{alg}$ has the following form:*

$$\begin{array}{lll} \textit{events} & E ::= b \Rightarrow u & \textit{where } b \in \mathcal{B}_{\mathcal{V}} \\ \textit{updates} & u, u' ::= x := e \mid u; u' & \textit{where } e \in \mathcal{E}_{\mathcal{V}} \textit{ and } x \in \mathcal{V} \setminus \mathcal{V}_{alg} \end{array}$$

Intuitively, an event E is triggered at the earliest time point when its boolean condition b becomes true. In this case a sequence of updates u are executed from the left to the right. Any update $x := e$ may change the value of some variable $x \in \mathcal{V} \setminus \mathcal{V}_{\text{alg}}$ to the value of the arithmetic expression e at the current time point.

Note that values of algebraic variables $x \in \mathcal{V}_{\text{alg}}$ cannot be updated, since they always must satisfy $x \overset{\circ}{=} expr(x)$. In contrast, the values of differential variables $y \in \mathcal{V}_{\text{diff}}$ may be updated while preserving its equation $\dot{y} = expr(y)$ except for the time point of the update. Also, the values of species can be updated by events.

Definition 3. *A Core SBML network is a tuple $N = (\mathcal{V}, Reacts, expr, init, Evts)$ where $\mathcal{V} = \mathcal{S} \uplus \mathcal{V}_{alg} \uplus \mathcal{V}_{diff}$ is a finite set of variables, Reacts is a finite set of Core SMBL reactions with species in $\mathcal{S}$ and variables in $\mathcal{V}$ and Evts a finite set of events with variables in $\mathcal{V}$ and algebraic variables in $\mathcal{V}_{alg}$. Furthermore:*

$$expr : \mathcal{V} \setminus \mathcal{S} \to \mathcal{E}_{\mathcal{V}}, \qquad init : \mathcal{V} \setminus \mathcal{V}_{alg} \to \mathcal{E}_{\mathcal{V}}.$$

Any reaction network N of Core SBML defines a system of algebraic differential equations that we introduce next. The value of reaction species $x \in \mathcal{S}$ evolves over time according to the following ODE induced by the set of reactions of N:

$$\dot{x} \stackrel{\circ}{=} \sum_{(R,P,e) \in Reacts} e(P(x) - R(x))$$

Furthermore $x \geq 0$ must hold for the trajectories of all species $x \in \mathcal{S}$, so at all time points. Note that the kinetic expression e may depend on any kind of variables in $\mathcal{V}$ not only on species. The values of the algebraic variables $x \in \mathcal{V}_{\text{alg}}$ are specified by the algebraic equation $x \stackrel{\circ}{=} expr(x)$. A differential variable $y \in \mathcal{V}_{\text{diff}}$ comes with a differential equation $\dot{y} \stackrel{\circ}{=} expr(y)$.

The differential-algebraic system of equations of a reaction network N, denoted $dae(N)$, is defined by:

$$\begin{aligned} dae(N) = {} & \textstyle\bigwedge_{x \in \mathcal{S}} \dot{x} \stackrel{\circ}{=} \sum_{(R,P,e) \in Reacts} e(P(x) - R(x)) \\ & \wedge \textstyle\bigwedge_{x \in \mathcal{S}} x \geq 0 \\ & \wedge \textstyle\bigwedge_{x \in \mathcal{V}_{\text{alg}}} x \stackrel{\circ}{=} expr(x) \\ & \wedge \textstyle\bigwedge_{x \in \mathcal{V}_{\text{diff}}} \dot{x} \stackrel{\circ}{=} expr(x) \end{aligned}$$

The function *init* defines the initial value of all variables that are not algebraic. The initial values at time point 0 must satisfy the following algebraic equations now interpreted over the reals.

$$initState(N) \;=\; \textstyle\bigwedge_{x \in \mathcal{V} \setminus \mathcal{V}_{\text{alg}}} x \stackrel{\circ}{=} init(x) \;\wedge\; \bigwedge_{y \in \mathcal{V}_{\text{alg}}} y \stackrel{\circ}{=} expr(y)$$

For instance, the reaction network with $init(x) = 2.1$ and $expr(y) = x + 1$ impose the algebraic equations to define the initial values at time 0:

$$x \stackrel{\circ}{=} 2.1 \wedge y \stackrel{\circ}{=} x + 1$$

We make the additional assumption on all Core SBML networks that there is no cyclic dependency between algebraic equations and initializations. This makes it possible to evaluate $init(x)$ for all nonalgebraic variables x. We note that this acyclicity assumption of of Core SBML is consistent with the SBML specification too.

For instance, the reaction network with $init(y) = x$ and $expr(x) = y$ is ill-formed where $y \in \mathcal{V} \setminus \mathcal{V}_{\text{alg}}$ and $x \in \mathcal{V}_{\text{alg}}$ is an algebraic variable, since the initial values are required to have the following cyclic dependencies:

$$x \stackrel{\circ}{=} y \wedge y \stackrel{\circ}{=} x$$

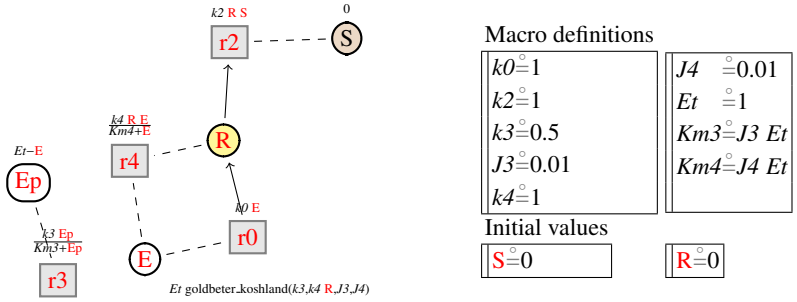

Fig. 3: Core SBML network capturing the essence of the SMBL model B309 in Fig. 2. Note that the variables *E* and *Ep* were removed from the reactants and products of reactions *r3* and *r4* since irrelevant. There are macro definitions for representing the SMBL parameters. Irrelevant variables for concentrations and compartment volumes were eliminated. Initial values are not changed.

We note that Core SBML networks subsume differential algebraic equations. These can be expressed by using algebraic and differential variables. Core SBML also subsumes chemical reaction networks [7]. Delays in differential equations or events are not supported though.

Core SBML networks may be enriched with extra information on top of Definition 3. The first kind concerns the definition of macros that will be discussed in Section 4.2 and the second how to represent compartments, volumes, and concentrations of species, see Section 4.4. The third kind concerns control parameters for events discussed in Section **??**.

4.2 Macros

A subset $\mathcal{V}_{\text{macros}} \subseteq \mathcal{V}_{\text{alg}}$ of macros may be distinguished. Given a macro $x \in \mathcal{V}_{\text{macros}}$ we call the $expr(x)$ the macros' definition. We assume that the definitions of macros are acyclic (in contast to more other algebraic equations). Therefore, all macros can be removed by exhaustively replacing them by their definitions. Macros can be used to represent SBML's parameters possibly with assignment rules. In graphs of Core SMBL networks, we show the macros definitions in a table while non-macro algebraic variables are drawn as nodes in white circles in the graphs of networks.

4.3 Example

We illustrate Core SBML at the networks in Fig. 3; it captures the essence of the SBML model B309 in Fig. 2. The Core SBML network has a single species $\mathcal{S} = \{R\}$, the algebraic variables $\mathcal{V}_{\text{alg}} = \{E, Ep\} \cup \mathcal{V}_{\text{macros}}$, where $\mathcal{V}_{\text{macros}} = \{Et, k_3, k_4, J_3, J_4, \ldots\}$, a single differential variable $\mathcal{V}_{\text{diff}} = \{S\}$. The definitions of the macros are given in Fig. 3. The initial values are given there too. Note that algebraic values don't have initial values.

In contrast to the SBML model in Fig. 2, there are no more variables for species' concentrations. This simplification makes sence given that the volume of compartment of all species was $env = 1$, so that concentrations and amounts did coincide anyway.

We note another differnce to the SBML model B309: the species *Ep* is no more a product of reaction *r3* and the species *E* is no more a reactant of reaction *r4*. In SMBL they existed syntactically, but had no semantic effect on the equations. In Core SBML, such inconsistencies are forbidden a priori by Definition 1.

4.4 Concentrations, Compartments, and Volumes

The second kind of extra information of Core SBML networks serves for representing species's concentations, compartments, and volumes. This is done in such a way that all equations are made explicit, even though this may leads to redundant information. The consistency of this redudant information is verified by the schema of Core SBML.

For each species x, two algebraic variables may be specified optionally. The first variable y stands for the concentration of x, and the second variable c for its compartment. If both are present, then the volume of the compartment of x is $vol = expr(c)$. The schema of Core SBML then imposes that $expr(y) = \frac{x}{c}$. Hence, the equation system $dae(N)$ of the network N contains the equations $y \stackrel{\circ}{=} \frac{x}{c} \wedge c \stackrel{\circ}{=} vol$.

4.5 Control Parameters

The third kind of extra information for Core SBML networks concerns variable that are controlled by events. A subset of control parameters $\mathcal{V}_{ctrl} \subseteq \mathcal{V}_{\text{diff}}$ may be specified. Any control parameter $x \in \mathcal{V}_{ctrl}$ is a differential variable with a fixed rate $expr(x) = 0$, so that it is constant over time (satisfying $\dot{x} \stackrel{\circ}{=} 0$) except for time points when events apply.

The initial value $init(x)$ remains to be specified by Core SBML models. In graphical representations, we will show a table with all control parameters and its initial values. In contrast to other differential variables, control parameters are not drawn as nodes of the network graph.

4.6 XML Syntax

We propose a concrete XML syntax for Core SBML that follows its abstract syntax. As a consequence the concrete syntax of Core SMBL is *not* a fragment of the concrete syntax of SBML. Still the idea is that it captures most of the expressiveness of SBML. This is consistent with the usual role of core languages in the context of programming languages.

Each variable of Core SBML is equipped with its type as "species", "algebraic", "differential", or "control", in contrast to the untyped identifiers such as `<ci> Ep </ci>` of SBML inherited from MathML. For instance, we can define an algebraic variable *Ep* for the SBML species with an assignment rule, a macro [*Ep*] for its concentration, and a macro *env* for its compartment as follows.

```
<variable type="algebraic" id="Ep"
     concentration="conc_Ep" compartment="env">
  <kinetic-expression>
    <minus>
      <expr id="Et"/>
      <expr id="conc_E"/>
    </minus>
  </kinetic-expression>
</variable>
<expression id="conc_Ep" latex-look="[Ep]">
   <divide>
      <var type="algebraic" id="Ep"/>
      <expr id="env"/>
    </divide>
</expression>
<expression id="env"> <constant value="1"/> </expression>
```

Variable references such as `<var type="algebraic" id="Ep"/>` are typed too. Note that the attribute `latex-look` permits to display species more nicely while keeping their identifiers simple. Mathematical operators such as `divide` are named such as in MathML, but can be applied directly (without requiring an additional apply element).

For illustration, we present the Core SBML models for B309, B001, and B111 in our XML format in Sections A.4, B.3, and C.3 of the appendix.

We defined an XML schema for Core SBML documents based on a document type descriptor (DTD) and a Schematron [11]. Compared to XML Schema [22] or RelaxNG [24], our approach has the advantage to nicely localize errors in invalid documents. This enables informative error messages. The DTD defines the hierarchical structure of Core SBML networks. The Schematron ensures the consistency of concentrations, amounts, and compartment volumes. It also verifies that there are no dangling references, i.e. that there is a `variable` definition for any `var` references with the same identifier and the same type. The Schematron also rules out empty identifiers or forbidden symbols in identifiers.

We developed some additional tools for Core SBML networks by using XML technology. We developped a graph drawing tool mapping Core SBML networks to the format of BioComputing's NetworkGraph tool. We also have a script computing the algebraic differential equations of a Core SBML network in XML format. Last not least, we implemented an inverse compiler from Core SBML to SBML.

5 Compiler from a SBML Fragment to Core SBML

Many concepts of SBML and Core SBML correspond in a direct manner. The main semantic difference are the permission of delays in ODEs and events of SMBL. These cannot be translated to Core SBML. Apart of this the syntax of Core SMBL (abstract and XML) is different from that of SMBL in order to guarantee the consistency of the equations. The compiler needs to identify the information in SBML models that is to be ignored in order to resolve inconsistencies based on SMBL's priorities.

Resolving inconsistencies is tedious given the informal character of the SBML specifications. In particular, it requires to precisely understand the relevance of species' attributes such as `@boundaryCondition` and `@constant` which interact in a complex manner. Selecting the right values from attributes `@initialConcentration`, `@initialAmount` and element `<initialAssignment>` is another issue. The details on these engineering aspects of the compiler cannot be presented in detail here.

5.1 Compartments, Volumes, and Concentrations

The proper treatment of compartments, volumes and concentration raises a non-obvious difficulty, given that Core SMBL does not reserve any extra treatment for them besides the consistency validation (see Section 4.4).

In order to explain this difficulty, let us first recall how compartment volumes and concentrations are treated in SBML. Each SBML species A is mapped to some compartment variable c, whose value is equal to that of some arithmetic expression vol standing for the compartments volume. The species A itself is used as a variable for the amount of A. SBML models may refer to a second variable $[A]$ that stands for A's concentration. The ODE for a species A – as infered by Copasi [**?**] or SBML2LaTeX [**?**] as used by the BioModels database [16] – then has the following form, where some other concentration variables such as $[B]$ may occur on the right:

$$\dot{A} \stackrel{\circ}{=} \ldots [A] \ldots [B] \ldots .$$

The variables for the amount and the concentration are coupled via the volume vol of the compartment c of species A, as expressed by the following equation:

$$[A] \stackrel{\circ}{=} A/c \wedge c \stackrel{\circ}{=} vol$$

If $vol = 1$ then $[A] \stackrel{\circ}{=} A$ so we can identify A and $[A]$ so that a single variable is enough. That is why a single variable per species was enough in the Core SBML network for the essence of the SMBL model B309 in Fig. 3.

Our idea for correctly compiling SBML models to Core SBML networks without restrictions on compartment volumes follows the two variable strategy of SBML. The compiler will introduce for any species variable A an algebraic variable $[A]$ for its concentration and another algebraic variable c for its compartment, such that $expr([A]) = A/c$ and $expr(c) = vol$ where vol is the arithmetic expression for the volume of c. We can then use the variable $[A]$ in the kinetic expressions of the reactions of Core SBML as done by SBML. The Core SBML networks produced by the compiler will use the option to store for each species A the relationship to the concentration variable $[A]$, and the compartment variable c, so that $expr([A]) = A/c$ is guaranteed by Core SMBL's schema.

5.2 Examples for the Compiler

The compiler applied to SBML model B309 yields the Core SBML network with the graph in Fig. 4. By replacing the algebraic variables for the concentration by the amount variables (given that the volume of compartment *env* is equal to 1), we obtain the previous

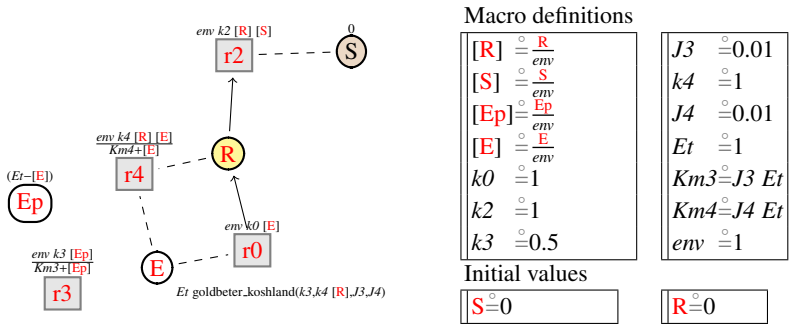

Fig. 4: The Core SBML network compiled from B309 in Fig. 2.

Role	Name	Initial value
Control parameter	flag_MPF	0

Event condition	Event updates
$MPF \leq 0.1 \wedge flag\_MPF = 1$	$M := \frac{M}{2}$
$0.1 < MPF$	$flag\_MPF := 1$

Species	Initial Concentration
cdc13T	0.2
preMPF	0
ste9	1
slp1T	0
slp1	2.2
IEP	0
rum1T	0
SK	0
M	1

Fig. 5: Control variables, events, and initial values of B111.

Core SBML network in Fig. 3. We note that the compartment structure present in the SMBL network is made fully explicit by the Core SBML network produced by our compiler. This structure is consistent with the equations in Fig. 4:

Amount	concentration	compartment
R	[R]	*env*
S	[S]	*env*
Ep	[Ep]	*env*
E	[E]	*env*

We next consider an SBML model with events which is B111 from BioModels. This network has 2 control variables and 2 events for their update, 9 reaction species, and a algebraic species *MPF* defined by an assignement rule. The events for updating the control variables are given in Fig. 5. The compiler yields the Core SBML network in Fig. 6. For each of the 9 reaction species it has a species variables and an algebraic variable for its concentration. Since the volume of compartment *cell* is equal to 1, the concentrationt variables must have the same value as the amount variables, so the network could be simplified.

The SMBL species *MPF* with an assignment rule becomes an algebraic variables in Core SMBL and yields another algebraic variable for the concentration [*MPF*]. We note that some modifiers of reactions *R3*, *R4*, and *R17* are hidden in the definition of the macros *TF*, k_{25} and *kwee*, see Fig. 14.

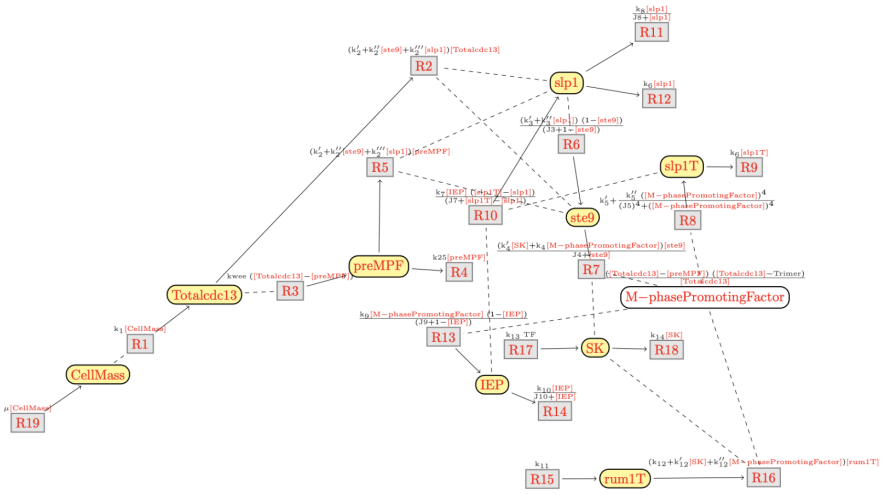

Fig. 6: Graph of the Core SBML network for B111. See Fig. 14 for the macro definitions.

An SMBL model where the compartment size matters is the BioModel BIMOD00000001 that we call B001 for short. The compartment of its species has a very small volume $comp1 = 1e-16$.

5.3 Implementation

We implemented our compiler from a large fragment of SBmML to Core SBML in the XML stylesheet transformation language (XSLT) [25]. We also implemented a straightforward inverse compiler to SBML. Based thereon, we can obtain simulation trajectories for Core SBML networks via Copasi. This permits us to compare the trajectories of the Core SBML networks obtained by our compiler to those of the original SBML model. On few tests on SBML models with dynamic compartment changes, we observed equal trajectories, confirming the correctness of our treatment of compartment volumes.

We applied our compiler to the 548 SBML models in the curated BioModels database. The compilation results are available at http://researchers.lille.inria.fr/niehren/Core-SBML. Out of the 548 curated SBML models, all but 19 could be compiled to Core SBML. Of these 13 contain delays in events and 6 delays in differential equations. Adding these two features to our formal treatment of Core SBML is easy, but practically it would require 2 more days of implementation. We also note that 56 curated BioModels contain piecewise linear functions, that we compiled to conditionals while using the constant *time*.

5.4 Limitations

Another issue is that the SMBL specification supports events with priorities. On the one hand side, the semantics of event priorities is quite complicated, on the other hand

side they are not used in any SBML model of curated part of the BioModels database. Therefore we decided to not add any event priorities to Core SBML.

A limitation of the current compiler is that it ignores most aspects of SBML irrelevant for the dynamic semantics. In particular, this applies to units like mol per liter, but also to the information on the reversibility of reactions (which only talks about the motivation of the kinetic law of the reaction). Furthermore, all XML elements for annotation and notes are ignored too.

6 Compiler of Core SBML to BioCham

BioCham supports pure chemical reaction networks with kinetic expressions and events, while ruling out arithmetic and differential variables. When ignoring its analysis and verification facilities, the abstract syntax of BioCham's modeling language is basically the same as that of Core SBML without algebraic and algebraic variable. Macros for expressions and function definitions are equally available.

Note that the import of SBML models by BioCham 4 follows the single variable approach where $[A] \stackrel{\circ}{=} A$ is valid for all species A. Compartment volumes are ignored all over. This is correct for compartments with fixed volume equal to 1, but incorrect otherwise. The compilation of SBML to Core SBML, in contrast, introduces two different variables for A and $[A]$ and then imposed the expected relationship. In this way, even dynamical changes of compartment volumes can be modeled correctly.

6.1 Compiler

We next show how to map Core SBML networks to BioCham 4. In combination with our compiler from SMBL to Core SBML from Section 5, this can be used to improve on BioCham current SBML import.

First, we notice that our set of arithmetic expression $\mathcal{E}_{\mathcal{V}}$ coincides with that of BioCham 4. In particular, conditionals are supported there but no delays. However, not all build-in functions of SBML coming via MathML are natively supported by BioCham 4. A counter example is *tanh*. But we found BioCham definitions for all those used in the SBML models of the Curated part of the BioModels database.

In particular we can define *tanh* as follows:

```
function(sinh(x) = (((2.71828^x) - (2.71828^(0 - x))) / 2)).
function(cosh(x) = (((2.71828^x) + (2.71828^(0 - x))) / 2)).
function(tanh(x) = (sinh(x) / cosh(x))).
function(coth(x) = (cosh(x) / sinh(x)))
```

Second notice, that BioCham can express any chemical reactions $R \xrightarrow{e} P$ of Core SBML by the statement: `e for R => P.` Third, BioCham 4 can express any algebraic equation $x \stackrel{\circ}{=} e$ by the statement `function(x=e).` Fourth, differential variables of Core SBML can be eliminated at beforehand: It is sufficient to turn any differential variable x into a species, that is produced by an artificial reaction with kinetic expression $expr(x)$ and not consumed by any other reaction. Fifth, the events of Core SBML coincide with those of BioCham 4 up to details of their concrete syntax. For instance the event of B001 can be written as follows in BioCham 4:

```
add_event((t2<Time), kf_0 = 0, kf_3 = 0, kf_7 = 0,
   kf_12 = 0, kf_1 = 0, kf_4 = 0, kf_8 = 0, kf_13 = 0).
% reaction for species variable Time
1 for _ => Time.
```

Sixth, initial values in BioCham must be specified by reals: they cannot be specified by arithmetic expressions, so that BioCham could compute them from other initial values. Therefore, the compiler has to evaluate the arithmetic expressions $init(x)$ to some real number for any $x \in \mathcal{V} \setminus \mathcal{V}_{\text{alg}}$. The evaluation cannot loop, since we assumed that there are no cyclic dependency between algebraic equations and initializations.

Our compiler from Core SBML to BioCham is correct in that maintains the differential algebraic equations of the network and its initial conditions up to equivalence, and also the events of the network. The correctness could be stated and proven formally. We note that BioCham cannot represent compartment structures. Our compiler can safely ignore Core SBML's logical information relating amounts, compartments, and concentrations, since consistency with the equations is garanteed anyway.

6.2 Implementation

We implemented our compiler from Core SBML to BioCham 4 in XSLT. We notice that the concrete syntax varies with the version of BioCham. The concrete syntax of the BioCham 4 networks for the SBML models B309, B001, and B111 can be found in Sections A.3, B.4, and C.4 of the appendix. The full collection of BioCham networks for all curated BioModels is available online at http://researchers.lille.inria.fr/niehren/Core-SBML. In this way, we obtained a second method to simulate Core SBML networks via BioCham. In particular, we could sucessfully simulate some SMBL models with dynamic compartment volumes via BioCham.

Conclusion

We presented the Core SBML, an exchange format for systems biology, which, in contrast to SBML, has a clear formal semantics based on differential algebraic equations. We argued that Core SBML covers a large part of the curated BioModels. In order to cover all of them, it is sufficient to add delays in differential equations and events. Conceptually this is not difficult, but it requires some more implementation efforts. In any case, delays are shown to make the only true difference in expressiveness between SBML and BioCham. We believe that Core SBML can help to reduce the difficulties to bridge the various tools in systems biology. A good next step could be to use Core SBML to revise the SMBL import of the scientific computing language Julia [2]. On this way, we hope that Core SBML will find a large acceptance for tools in systems biology eventually.

Acknowledgements

We thank Sylvain Soliman and François Fages for their unfailable and timely support with BioCham questions. We also enjoyed efficient support from the Copasi people.

Finally, this work was supported by the French National Research Agency (ANR), by funding the project MIGAD (ANR- 21-CE45-0017).

References

1. Allart, E., Niehren, J., Versari, C.: Computing difference abstractions of metabolic networks under kinetic constraints. In: Bortolussi, L., Sanguinetti, G. (eds.) Computational Methods in Systems Biology - 17th International Conference, CMSB 2019, Trieste, Italy, September 18-20, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11773, pp. 266–285. Springer (2019). https://doi.org/10.1007/978-3-030-31304-3_14, https://doi.org/10.1007/978-3-030-31304-3_14
2. Bezanson, J., Edelman, A., Karpinski, S., Shah, V.B.: Julia: A fresh approach to numerical computing. SIAM Review **59**(1), 65–98 (2017). https://doi.org/10.1137/141000671, https://doi.org/10.1137/141000671
3. Calzone, L., Fages, F., Soliman, S.: BIOCHAM: an environment for modeling biological systems and formalizing experimental knowledge. Bioinformatics **22**(14), 1805–1807 (Jul 2006). https://doi.org/10.1093/bioinformatics/btl172, http://dx.doi.org/10.1093/bioinformatics/btl172
4. Euler, L.: Institutionum Calculi Integralis. No. vol. 1 in Institutionum Calculi Integralis, imp. Acad. imp. Saènt. (1768)
5. Fages, F., Soliman, S.: Abstract interpretation and types for systems biology. Theor. Comput. Sci. **403**(1), 52–70 (2008)
6. Fages, F., Gay, S., Soliman, S.: Inferring reaction systems from ordinary differential equations. Theor. Comput. Sci. **599**, 64–78 (2015). https://doi.org/10.1016/j.tcs.2014.07.032, http://dx.doi.org/10.1016/j.tcs.2014.07.032
7. Feinberg, M.: Chemical reaction network structure and the stability of complex isothermal reactors–I. the deficiency zero and deficiency one theorems. Chemical Engineering Science **42**(10), 2229 – 2268 (1987). https://doi.org/http://dx.doi.org/10.1016/0009-2509(87)80099-4, http://www.sciencedirect.com/science/article/pii/0009250987800994
8. Glass, L., Kauffman, S.A.: The logical analysis of continuous, non-linear biochemical control networks. Journal of Theoretical Biology **39**(1), 103–129 (1973). https://doi.org/https://doi.org/10.1016/0022-5193(73)90208-7, https://www.sciencedirect.com/science/article/pii/0022519373902087
9. Harel, D.: Statecharts: a visual formalism for complex systems. Science of Computer Programming **8**(3), 231–274 (Jun 1987). https://doi.org/10.1016/0167-6423(87)90035-9, http://dx.doi.org/10.1016/0167-6423(87)90035-9
10. Hoops, S., Sahle, S., Gauges, R., Lee, C., Pahle, J., Simus, N., Singhal, M., Xu, L., Mendes, P., Kummer, U.: Copasi—a complex pathway simulator. Bioinformatics **22**(24), 3067–3074 (2006)
11. Jelliffe, R.: Schematron (2006), iSO/IEC 19757-3
12. John, M., Nebut, M., Niehren, J.: Knockout Prediction for Reaction Networks with Partial Kinetic Information. In: 14th International Conference on Verification, Model Checking, and Abstract Interpretation. pp. 355–374. Rom, Italy (Jan 2013), http://hal.inria.fr/hal-00692499
13. Kay, M.: The saxon xslt and xquery processor (2004), https://www.saxonica.com
14. Keating, S.M., et al. SBML Level 3 Community members: SBML Level 3: An extensible format for the exchange and reuse of biological models. Molecular Systems Biology **16**(8), e9110 (Aug 2020). https://doi.org/10.15252/msb.20199110

15. Madelaine, G., Lhoussaine, C., Niehren, J.: Attractor Equivalence: An Observational Semantics for Reaction Networks. In: First International Conference on Formal Methods in Macro-Biology. pp. 82–101. Lecture Notes in Bioinformatics, Springer-Verlag, Nouméa, New Caledonia (Sep 2014), https://hal.archives-ouvertes.fr/hal-00990924
16. Malik-Sheriff, R.S., Glont, M., Nguyen, T.V.N., Tiwari, K., Roberts, M.G., Xavier, A., Vu, M.T., Men, J., Maire, M., Kananathan, S., Fairbanks, E.L., Meyer, J.P., Arankalle, C., Varusai, T.M., Knight-Schrijver, V., Li, L., Dueñas-Roca, C., Dass, G., Keating, S.M., Park, Y.M., Buso, N., Rodriguez, N., Hucka, M., Hermjakob, H.: BioModels—15 years of sharing computational models in life science. Nucleic Acids Research **48**(D1), D407–D415 (2020). https://doi.org/10.1093/nar/gkz1055, https://doi.org/10.1093/nar/gkz1055
17. Mizera, A., Pang, J., Qu, H., Yuan, Q.: Taming asynchrony for attractor detection in large boolean networks. IEEE/ACM Transactions on Computational Biology and Bioinformatics **16**(1), 31–42 (2019). https://doi.org/10.1109/TCBB.2018.2850901
18. Niehren, J., Vaginay, A., Versari, C.: Abstract simulation of reaction networks via boolean networks. In: Petre, I., Paun, A. (eds.) Computational Methods in Systems Biology - 20th International Conference, CMSB 2022, Bucharest, Romania, September 14-16, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13447, pp. 21–40. Springer (2022). https://doi.org/10.1007/978-3-031-15034-0_2, https://doi.org/10.1007/978-3-031-15034-0_2
19. Niehren, J., Versari, C., John, M., Coutte, F., Jacques, P.: Predicting Changes of Reaction Networks with Partial Kinetic Information. BioSystems **149**, 113–124 (Jul 2016), https://hal.inria.fr/hal-01239198
20. Paulevé, L., Kolçà, J., Chatain, T., Haar, S.: Reconciling Qualitative, Abstract, and Scalable Modeling of Biological Networks. Nature Communications **11** (2020). https://doi.org/10.1038/s41467-020-18112-5, https://hal.archives-ouvertes.fr/hal-02518582
21. Thomas, R.: Boolean formalization of genetic control circuits. Journal of Theoretical Biology **42**(3), 563–585 (1973). https://doi.org/https://doi.org/10.1016/0022-5193(73)90247-6, https://www.sciencedirect.com/science/article/pii/0022519373902476
22. Thompson, H.S., Beech, D., Maloney, M., Mendelsohn, N.: Xml schema part 1: Structures second edition (2004), http://www.w3.org/TR/xmlschema-1/
23. Vaginay, A., Boukhobza, T., Smaïl-Tabbone, M.: From quantitative SBML models to boolean networks. Appl. Netw. Sci. **7**(1), 73 (2022). https://doi.org/10.1007/s41109-022-00505-8, https://doi.org/10.1007/s41109-022-00505-8
24. Van der Vlist, E.: RELAX NG: A Simpler Schema Language for XML. O'Reilly & Assoc, 1. aufl edn. (2004)
25. W3C: XSL transformations (XSLT) version 3.0 (2017), https://www.w3.org/TR/xslt-30

A Example of BioModel B309

A.1 Graph of Core SBML network for B309

The compiler converts the SBML model B309 with the graph from Fig. 2 to the Core SBML network with the graph in Fig. 4. Given that the volume of compartment env is equal to 1, it can be simplified to the Core SBML network discussed earlier in Fig. 3.

A.2 Core SBML Network for B309 in XML Syntax

The Core SBML network in Fig. 4 has the following concrete XML syntax.

```
<?xml version="1.0" encoding="utf-8"?>
<network id="Tyson2003_NegFB_Homeostasis"
         name="Tyson2003_NegFB_Homeostasis"
         version="2.1"><!--
        +++ start species +++++
      --><!-- species without initial assignment -->
   <variable type="species"
             id="R"
             latex-look="R"
             initial="0"
             concentration="conc_R"
             compartment="env"/>
   <expression id="conc_R" latex-look="[R]">
      <divide>
         <var type="species" id="R"/>
         <expr id="env"/>
      </divide>
   </expression>
   <!-- species with rateRule -->
   <variable type="differential"
             id="S"
             compartment="env"
             latex-look="S"
             initial="0"
             concentration="conc_S">
      <kinetic-expression>
         <constant value="0"/>
      </kinetic-expression>
   </variable>
   <expression id="conc_S" latex-look="[S]">
      <divide>
         <var type="differential" id="S"/>
         <expr id="env"/>
      </divide>
   </expression>
   <!-- species with assignmentRule -->
   <variable type="algebraic"
```

```
            id="Ep"
            latex-look="Ep"
            concentration="conc_Ep"
            compartment="env">
      <kinetic-expression>
         <minus>
            <expr id="Et"/>
            <expr id="conc_E"/>
         </minus>
      </kinetic-expression>
   </variable>
   <expression id="conc_Ep" latex-look="[Ep]">
      <divide>
         <var type="algebraic" id="Ep"/>
         <expr id="env"/>
      </divide>
   </expression>
   <!-- species with assignmentRule -->
   <variable type="algebraic"
            id="E"
            latex-look="E"
            concentration="conc_E"
            compartment="env">
      <kinetic-expression>
         <times>
            <expr id="Et"/>
            <apply fun="goldbeter_koshland">
               <expr id="k3"/>
               <times>
                  <expr id="k4"/>
                  <expr id="conc_R"/>
               </times>
               <expr id="J3"/>
               <expr id="J4"/>
            </apply>
         </times>
      </kinetic-expression>
   </variable>
   <expression id="conc_E" latex-look="[E]">
      <divide>
         <var type="algebraic" id="E"/>
         <expr id="env"/>
      </divide>
   </expression>
   <!-- +++ start reactions ++++++-->
   <reaction id="r0">
      <kinetic-expression>
         <times>
            <expr id="env"/>
            <expr id="k0"/>
```

```
            <expr id="conc_E"/>
         </times>
      </kinetic-expression>
      <product spec="R"/>
      <modifier spec="E"/>
   </reaction>
   <reaction id="r2">
      <kinetic-expression>
         <times>
            <expr id="env"/>
            <expr id="k2"/>
            <expr id="conc_R"/>
            <expr id="conc_S"/>
         </times>
      </kinetic-expression>
      <reactant spec="R"/>
      <modifier spec="S"/>
   </reaction>
   <reaction id="r3">
      <kinetic-expression>
         <divide>
            <times>
               <expr id="env"/>
               <expr id="k3"/>
               <expr id="conc_Ep"/>
            </times>
            <plus><!--  nonconstant parameter without rule nor event-->
               <expr id="Km3"/>
               <expr id="conc_Ep"/>
            </plus>
         </divide>
      </kinetic-expression>
   </reaction>
   <reaction id="r4">
      <kinetic-expression>
         <divide>
            <times>
               <expr id="env"/>
               <expr id="k4"/>
               <expr id="conc_R"/>
               <expr id="conc_E"/>
            </times>
            <plus><!--  nonconstant parameter without rule nor event-->
               <expr id="Km4"/>
               <expr id="conc_E"/>
            </plus>
         </divide>
      </kinetic-expression>
      <modifier spec="R"/>
   </reaction>
```

```
<!-- +++ start functionDefinitions +++++ -->
<function id="goldbeter_koshland">
   <lambda>
      <bvar id="v1"/>
      <bvar id="v2"/>
      <bvar id="J1"/>
      <bvar id="J2"/>
      <divide>
         <times>
            <constant value="2"/>
            <var id="v1"/>
            <var id="J2"/>
         </times>
         <plus>
            <minus>
               <var id="v2"/>
               <var id="v1"/>
            </minus>
            <times>
               <var id="J1"/>
               <var id="v2"/>
            </times>
            <times>
               <var id="J2"/>
               <var id="v1"/>
            </times>
            <power>
               <minus>
                  <power>
                     <plus>
                        <minus>
                           <var id="v2"/>
                           <var id="v1"/>
                        </minus>
                        <times>
                           <var id="J1"/>
                           <var id="v2"/>
                        </times>
                        <times>
                           <var id="J2"/>
                           <var id="v1"/>
                        </times>
                     </plus>
                     <constant value="2"/>
                  </power>
                  <times>
                     <constant value="4"/>
                     <minus>
                        <var id="v2"/>
                        <var id="v1"/>
```

```
                    </minus>
                    <var id="v1"/>
                    <var id="J2"/>
                 </times>
              </minus>
              <divide>
                 <constant value="1"/>
                 <constant value="2"/>
              </divide>
           </power>
        </plus>
     </divide>
  </lambda>
</function>
<!-- +++ start parameters +++++ -->
<expression id="k0" latex-look="k0">
  <constant value="1"/>
</expression>
<expression id="k2" latex-look="k2">
  <constant value="1"/>
</expression>
<expression id="k3" latex-look="k3">
  <constant value="0.5"/>
</expression>
<expression id="J3" latex-look="J3">
  <constant value="0.01"/>
</expression>
<expression id="k4" latex-look="k4">
  <constant value="1"/>
</expression>
<expression id="J4" latex-look="J4">
  <constant value="0.01"/>
</expression>
<expression id="Et" latex-look="Et">
  <constant value="1"/>
</expression>
<!--   with assignment rule -->
<expression id="Km3" latex-look="Km3">
  <times>
     <expr id="J3"/>
     <expr id="Et"/>
  </times>
</expression>
<!--   with assignment rule -->
<expression id="Km4" latex-look="Km4">
  <times>
     <expr id="J4"/>
     <expr id="Et"/>
  </times>
</expression>
```

```
   <!-- +++ start compartments -->
   <!--constant compartment -->
   <expression id="env" latex-look="env">
      <constant value="1"/>
   </expression>
</network>

\end{minited}
<?xml version="1.0" encoding="utf-8"?>
<network id="Tyson2003_NegFB_Homeostasis"
         name="Tyson2003_NegFB_Homeostasis"
         version="2.1"><!--
        +++ start species +++++
      --><!-- species without initial assignment -->
   <variable type="species"
             id="R"
             latex-look="R"
             initial="0"
             concentration="conc_R"
             compartment="env"/>
   <expression id="conc_R" latex-look="[R]">
      <divide>
         <var type="species" id="R"/>
         <!--  compartment without rateRule nor event -->
         <expr id="env"/>
      </divide>
   </expression>
   <!-- rateRule for object S which is neither a compartment nor a parameter-->
   <variable type="differential"
             id="S"
             compartment="env"
             latex-look="S"
             initial="0">
      <kinetic-expression>
         <constant value="0"/>
      </kinetic-expression>
   </variable>
   <expression id="conc_S" latex-look="[S]">
      <divide>
         <var type="differential" id="S"/>
         <!--  compartment without rateRule nor event -->
         <expr id="env"/>
      </divide>
   </expression>
   <!-- species with assignmentRule -->
   <variable type="algebraic"
             id="Ep"
             latex-look="Ep"
             concentration="conc_Ep"
             compartment="env">
```

```
   <kinetic-expression>
      <minus>
         <expr id="Et"/>
         <expr id="conc_E"/>
      </minus>
   </kinetic-expression>
</variable>
<expression id="conc_Ep" latex-look="[Ep]">
   <divide>
      <var type="algebraic" id="Ep"/>
      <!--  compartment without rateRule nor event -->
      <expr id="env"/>
   </divide>
</expression>
<!-- species with assignmentRule -->
<variable type="algebraic"
          id="E"
          latex-look="E"
          concentration="conc_E"
          compartment="env">
   <kinetic-expression>
      <times>
         <expr id="Et"/>
         <apply fun="goldbeter_koshland">
            <expr id="k3"/>
            <times>
               <expr id="k4"/>
               <expr id="conc_R"/>
            </times>
            <expr id="J3"/>
            <expr id="J4"/>
         </apply>
      </times>
   </kinetic-expression>
</variable>
<expression id="conc_E" latex-look="[E]">
   <divide>
      <var type="algebraic" id="E"/>
      <!--  compartment without rateRule nor event -->
      <expr id="env"/>
   </divide>
</expression>
<!-- +++ start reactions ++++++-->
<reaction id="r0">
   <kinetic-expression>
      <times><!-- compartment without assignmentRule -->
         <expr id="env"/>
         <expr id="k0"/>
         <expr id="conc_E"/>
      </times>
```

```
   </kinetic-expression>
   <product spec="R"/>
   <modifier spec="E"/>
</reaction>
<reaction id="r2">
   <kinetic-expression>
      <times><!-- compartment without assignmentRule -->
         <expr id="env"/>
         <expr id="k2"/>
         <expr id="conc_R"/>
         <expr id="conc_S"/>
      </times>
   </kinetic-expression>
   <reactant spec="R"/>
   <modifier spec="S"/>
</reaction>
<reaction id="r3">
   <kinetic-expression>
      <divide>
         <times><!-- compartment without assignmentRule -->
            <expr id="env"/>
            <expr id="k3"/>
            <expr id="conc_Ep"/>
         </times>
         <plus><!--  nonconstant parameter without rule nor event-->
            <expr id="Km3"/>
            <expr id="conc_Ep"/>
         </plus>
      </divide>
   </kinetic-expression>
</reaction>
<reaction id="r4">
   <kinetic-expression>
      <divide>
         <times><!-- compartment without assignmentRule -->
            <expr id="env"/>
            <expr id="k4"/>
            <expr id="conc_R"/>
            <expr id="conc_E"/>
         </times>
         <plus><!--  nonconstant parameter without rule nor event-->
            <expr id="Km4"/>
            <expr id="conc_E"/>
         </plus>
      </divide>
   </kinetic-expression>
   <modifier spec="R"/>
</reaction>
<!-- +++ start functionDefinitions +++++ -->
<function id="goldbeter_koshland">
```

```
<lambda>
   <bvar id="v1"/>
   <bvar id="v2"/>
   <bvar id="J1"/>
   <bvar id="J2"/>
   <divide>
      <times>
         <constant value="2"/>
         <var id="v1"/>
         <var id="J2"/>
      </times>
      <plus>
         <minus>
            <var id="v2"/>
            <var id="v1"/>
         </minus>
         <times>
            <var id="J1"/>
            <var id="v2"/>
         </times>
         <times>
            <var id="J2"/>
            <var id="v1"/>
         </times>
         <power>
            <minus>
               <power>
                  <plus>
                     <minus>
                        <var id="v2"/>
                        <var id="v1"/>
                     </minus>
                     <times>
                        <var id="J1"/>
                        <var id="v2"/>
                     </times>
                     <times>
                        <var id="J2"/>
                        <var id="v1"/>
                     </times>
                  </plus>
                  <constant value="2"/>
               </power>
               <times>
                  <constant value="4"/>
                  <minus>
                     <var id="v2"/>
                     <var id="v1"/>
                  </minus>
                  <var id="v1"/>
```

```
                  <var id="J2"/>
               </times>
            </minus>
            <divide>
               <constant value="1"/>
               <constant value="2"/>
            </divide>
         </power>
      </plus>
   </divide>
</lambda>
</function>
<!-- +++ start parameters +++++ -->
<expression id="k0">
   <constant value="1"/>
</expression>
<expression id="k2">
   <constant value="1"/>
</expression>
<expression id="k3">
   <constant value="0.5"/>
</expression>
<expression id="J3">
   <constant value="0.01"/>
</expression>
<expression id="k4">
   <constant value="1"/>
</expression>
<expression id="J4">
   <constant value="0.01"/>
</expression>
<expression id="Et">
   <constant value="1"/>
</expression>
<!--   with assignment rule -->
<expression id="Km3" latex-look="Km3">
   <times>
      <expr id="J3"/>
      <expr id="Et"/>
   </times>
</expression>
<!--   with assignment rule -->
<expression id="Km4" latex-look="Km4">
   <times>
      <expr id="J4"/>
      <expr id="Et"/>
   </times>
</expression>
<!-- +++ start compartments +++ -->
<!--constant compartment -->
```

```
   <expression id="env" latex-look="env">
      <constant value="1"/>
   </expression>
</network>
```

A.3 Compilation to BioCham

From the Core SBML model of B309 we obtain the following network of BioCham (version 4) by our compiler.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Network: Tyson2003_NegFB_Homeostasis
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%  reaction rules %%%%%%%%%%%%%%%%%%%%%%%%%%%%

% S
0 for _=>S.
% r0
(env * k0 * conc_E) for _=>R.
% r2
(env * k2 * conc_R * conc_S) for R=>_.
% r3
((env * k3 * conc_Ep) / (Km3 + conc_Ep)) for _=>_.
% r4
((env * k4 * conc_R * conc_E) / (Km4 + conc_E)) for _=>_.

%%%% initial values %%%%%%%%%%%%%%%%%%%%%%%%%%

present(R,0).
present(S,0).

% algebraic variables  %%%%%%%%%%%%%%%%%%%%%%%%%

function(conc_R=(R / env)).
function(conc_S=(S / env)).
function(conc_Ep=(Ep / env)).
function(Ep=(Et - conc_E)).
function(conc_E=(E / env)).
function(E=(Et * ((2 * k3 * J4) / (((k4 * conc_R) - k3) +
    (J3 * (k4 * conc_R)) + (J4 * k3) +
    ((((((k4 * conc_R) - k3) + (J3 * (k4 * conc_R))
    + (J4 * k3))^2) -
    (4 * ((k4 * conc_R) - k3) * k3 * J4))^(1 / 2)))))).

% parameters  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function(k0=1).
function(k2=1).
function(k3=0.5).
function(J3=0.01).
function(k4=1).
function(J4=0.01).
function(Et=1).
function(Km3=(J3 * Et)).
function(Km4=(J4 * Et)).

%  compartment volumes
function(env=1).
```

A.4 Original SBML Model B309

The original SBML model of network B309 is given below.

```
<?xml version='1.0' encoding='UTF-8' standalone='no'?>
<sbml xmlns="http://www.sbml.org/sbml/level2/version4" level="2"
      metaid="_1a70283c-bfbc-4a94-afe7-89095f79f871" version="4">
  <model id="Tyson2003_NegFB_Homeostasis" metaid="_466212"
         name="Tyson2003_NegFB_Homeostasis">
    <notes>
      <body xmlns="http://www.w3.org/1999/xhtml">
        <p>This is an SBML implementation the model of homeostastis by
        negative feedback (figure 1g) described in the article:<br/>
        <b>Sniffers, buzzers, toggles and blinkers: dynamics of
        regulatory
        and signaling pathways in the cell.</b>
        <br/>
        Tyson JJ, Chen KC, Novak B. <em>Curr Opin Cell Biol.</em> 2003
        Apr;15(2):221-31. PubmedID:<a
        href="http://www.ncbi.nlm.nih.gov/pubmed/12648679">12648679</a>;
        DOI:<a
        href="http://dx.doi.org/10.1016/S0955-0674(03)00017-6">10.1016/S0955-0674(03)00017-6</a>;<br/>
        </p>
        <p>Abstract:<br/>
        The physiological responses of cells to external and internal stimuli
        are governed by genes and proteins interacting in
        complex networks whose dynamical properties are impossible to
        understand by intuitive reasoning alone.
        Recent advances by theoretical biologists have demonstrated that
        molecular regulatory networks can be
        accurately modeled in mathematical terms. These models shed light on
        the design principles of
        biological control systems and make predictions that have been verified experimentally.
        </p>
```

```
    <p> Originally created by libAntimony v1.4 (using libSBML 3.4.1) </p>
    <p>This model originates from BioModels Database: A Database
    of Annotated Published
    Models (http://www.ebi.ac.uk/biomodels/). It is copyright (c)
    2005-2011
    The BioModels.net Team.<br/>
    For more information see the <a
    href="http://www.ebi.ac.uk/biomodels/legal.html"
    target="_blank">terms of use</a>.<br/>
    To cite BioModels Database, please use: <a
    href="http://www.ncbi.nlm.nih.gov/pubmed/20587024" target="_blank">Li
    C, Donizelli M, Rodriguez N, Dharuri H,
    Endler L, Chelliah V, Li L, He E, Henry A, Stefan MI, Snoep JL, Hucka
    M, Le Novère N, Laibe C (2010)
    BioModels Database: An enhanced, curated and annotated resource for
    published
    quantitative kinetic models. BMC Syst Biol., 4:92.</a>
    </p>
  </body>
</notes>
<annotation>
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
           xmlns:dc="http://purl.org/dc/elements/1.1/"
           xmlns:vCard="http://www.w3.org/2001/vcard-rdf/3.0#"
           xmlns:dcterms="http://purl.org/dc/terms/"
           xmlns:bqmodel="http://biomodels.net/model-qualifiers/"
           xmlns:bqbiol="http://biomodels.net/biology-qualifiers/">
    <rdf:Description rdf:about="#_466212">
      <dc:creator>
        <rdf:Bag>
          <rdf:li rdf:parseType="Resource">
            <vCard:N rdf:parseType="Resource">
              <vCard:Family>Endler</vCard:Family>
              <vCard:Given>Lukas</vCard:Given>
            </vCard:N>
            <vCard:EMAIL>lukas@ebi.ac.uk</vCard:EMAIL>
            <vCard:ORG rdf:parseType="Resource">
              <vCard:Orgname>EMBL-EBI</vCard:Orgname>
            </vCard:ORG>
          </rdf:li>
          <rdf:li rdf:parseType="Resource">
            <vCard:N rdf:parseType="Resource">
              <vCard:Family>Tyson</vCard:Family>
              <vCard:Given>John J</vCard:Given>
            </vCard:N>
            <vCard:EMAIL>tyson@vt.edu</vCard:EMAIL>
            <vCard:ORG rdf:parseType="Resource">
              <vCard:Orgname>Department of Biology, Virginia
              Polytechnic Institute and
              State University, Blacksburg, VA 24061, USA</vCard:Orgname>
```

```
            </vCard:ORG>
          </rdf:li>
        </rdf:Bag>
      </dc:creator>
      <dcterms:created rdf:parseType="Resource">
        <dcterms:W3CDTF>2011-02-10T04:48:32Z</dcterms:W3CDTF>
      </dcterms:created>
      <dcterms:modified rdf:parseType="Resource">
        <dcterms:W3CDTF>2014-04-06T20:04:29Z</dcterms:W3CDTF>
      </dcterms:modified>
      <bqmodel:is>
        <rdf:Bag>
          <rdf:li rdf:resource="http://identifiers.org/biomodels.db/MODEL1102100003"/>
        </rdf:Bag>
      </bqmodel:is>
      <bqmodel:is>
        <rdf:Bag>
          <rdf:li rdf:resource="http://identifiers.org/biomodels.db/BIOMD0000000309"/>
        </rdf:Bag>
      </bqmodel:is>
      <bqmodel:isDescribedBy>
        <rdf:Bag>
          <rdf:li rdf:resource="http://identifiers.org/pubmed/12648679"/>
        </rdf:Bag>
      </bqmodel:isDescribedBy>
      <bqbiol:hasProperty>
        <rdf:Bag>
          <rdf:li rdf:resource="http://identifiers.org/mamo/MAMO_0000046"/>
        </rdf:Bag>
      </bqbiol:hasProperty>
      <bqbiol:hasTaxon>
        <rdf:Bag>
          <rdf:li rdf:resource="http://identifiers.org/taxonomy/131567"/>
        </rdf:Bag>
      </bqbiol:hasTaxon>
      <bqbiol:isVersionOf>
        <rdf:Bag>
          <rdf:li rdf:resource="http://identifiers.org/go/GO:0051098"/>
        </rdf:Bag>
      </bqbiol:isVersionOf>
    </rdf:Description>

  </rdf:RDF>
</annotation>
<listOfFunctionDefinitions>
  <functionDefinition id="goldbeter_koshland" metaid="_466242">
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      <lambda>
        <bvar>
          <ci> v1 </ci>
```

```
</bvar>
<bvar>
  <ci> v2 </ci>
</bvar>
<bvar>
  <ci> J1 </ci>
</bvar>
<bvar>
  <ci> J2 </ci>
</bvar>
<apply>
  <divide/>
  <apply>
    <times/>
    <cn type="integer"> 2 </cn>
    <ci> v1 </ci>
    <ci> J2 </ci>
  </apply>
  <apply>
    <plus/>
    <apply>
      <minus/>
      <ci> v2 </ci>
      <ci> v1 </ci>
    </apply>
    <apply>
      <times/>
      <ci> J1 </ci>
      <ci> v2 </ci>
    </apply>
    <apply>
      <times/>
      <ci> J2 </ci>
      <ci> v1 </ci>
    </apply>
    <apply>
      <power/>
      <apply>
        <minus/>
        <apply>
          <power/>
          <apply>
            <plus/>
            <apply>
              <minus/>
              <ci> v2 </ci>
              <ci> v1 </ci>
            </apply>
            <apply>
              <times/>
```

```
                    <ci> J1 </ci>
                    <ci> v2 </ci>
                  </apply>
                  <apply>
                    <times/>
                    <ci> J2 </ci>
                    <ci> v1 </ci>
                  </apply>
                </apply>
                <cn type="integer"> 2 </cn>
              </apply>
              <apply>
                <times/>
                <cn type="integer"> 4 </cn>
                <apply>
                  <minus/>
                  <ci> v2 </ci>
                  <ci> v1 </ci>
                </apply>
                <ci> v1 </ci>
                <ci> J2 </ci>
              </apply>
            </apply>
            <apply>
              <divide/>
              <cn type="integer"> 1 </cn>
              <cn type="integer"> 2 </cn>
            </apply>
          </apply>
        </apply>
      </apply>
    </lambda>
  </math>
 </functionDefinition>
</listOfFunctionDefinitions>
<listOfUnitDefinitions>
 <unitDefinition id="time" metaid="_1f68b21e-950d-4a11-8db0-a76699ead661" name="s">
  <listOfUnits>
   <unit kind="second" metaid="_067a2aff-e5bd-4ba1-98cb-0b707de517e1"/>
  </listOfUnits>
 </unitDefinition>
 <unitDefinition id="substance"
                 metaid="a11b8c1c-ea6a-4be4-a15b-dd180add236b" name="mole">
  <listOfUnits>
   <unit kind="mole"
         metaid="_56d5798a-0737-4803-bd11-dbba0905306a"/>
  </listOfUnits>
 </unitDefinition>
 <unitDefinition id="per_s"
                 metaid="_3cb5fe4b-0a06-4817-999d-44baface77a1" name="per_s">
```

```
    <listOfUnits>
      <unit exponent="-1" kind="second"
            metaid="_4b1d24ff-10c7-4a9d-b110-520d30defd43"/>
    </listOfUnits>
  </unitDefinition>
  <unitDefinition id="M_per_s"
                  metaid="_5012ac45-6837-4b95-856d-9f7c6cdc4298" name="M_per_s">
    <listOfUnits>
      <unit kind="mole" metaid="ae9afbf7-6d4b-4aad-a672-a488d0efd0e9"/>
      <unit exponent="-1" kind="second" metaid="f757ee55-401c-4ccd-88b3-cc5cd9e58545"/>
      <unit exponent="-1" kind="litre" metaid="b2bc9896-5431-4525-873a-f4cd846cc38c"/>
    </listOfUnits>
  </unitDefinition>
  <unitDefinition id="M" metaid="acfbc63d-c7be-4ddd-859f-67253cf77434" name="M">
    <listOfUnits>
      <unit kind="mole" metaid="_3d00cb92-ab57-453d-a281-fc7b168d545b"/>
      <unit exponent="-1" kind="litre" metaid="_43d0c259-16f1-411f-9066-a58a0611058d"/>
    </listOfUnits>
  </unitDefinition>
  <unitDefinition id="per_M_per_s"
                  metaid="_2de6849f-8d39-4e46-97b3-8265dda6a80f" name="per_M_per_s">
    <listOfUnits>
      <unit exponent="-1" kind="mole" metaid="b5d48f92-49dd-4b0f-9f10-8dc4e9a5b7b3"/>
      <unit kind="litre" metaid="_99822380-2ea1-41a2-8f00-deddc82b66e4"/>
      <unit exponent="-1" kind="second" metaid="_4d5f5779-88d0-4a2f-babb-62c6d4e5a53b"/>
    </listOfUnits>
  </unitDefinition>
</listOfUnitDefinitions>
<listOfCompartments>
  <compartment id="env" metaid="_466213" sboTerm="SBO:0000290" size="1">
    <annotation>
      <rdf:RDF
          xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:bqmodel="http://biomodels.net/model-qualifiers/"
          xmlns:bqbiol="http://biomodels.net/biology-qualifiers/">
        <rdf:Description rdf:about="#_466213">
          <bqbiol:isVersionOf>
            <rdf:Bag>
              <rdf:li rdf:resource="http://identifiers.org/go/GO:0005623"/>
            </rdf:Bag>
          </bqbiol:isVersionOf>
        </rdf:Description>

      </rdf:RDF>
    </annotation>
  </compartment>
</listOfCompartments>
<listOfSpecies>
  <species compartment="env" id="R" initialConcentration="0"
           metaid="_466214" sboTerm="SBO:0000252">
```

```
  <annotation>
    <rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:bqmodel="http://biomodels.net/model-qualifiers/"
        xmlns:bqbiol="http://biomodels.net/biology-qualifiers/">
      <rdf:Description rdf:about="#_466214">
        <bqbiol:isVersionOf>
          <rdf:Bag>
            <rdf:li rdf:resource="http://identifiers.org/chebi/CHEBI:36080"/>
          </rdf:Bag>
        </bqbiol:isVersionOf>

        <bqbiol:hasProperty>
          <rdf:Bag>
            <rdf:li rdf:resource="http://identifiers.org/go/GO:0016301"/>
          </rdf:Bag>
        </bqbiol:hasProperty>
      </rdf:Description>

    </rdf:RDF>
  </annotation>
</species>
<species boundaryCondition="true" compartment="env" id="S"
         initialConcentration="0"
         metaid="_466215" sboTerm="SBO:0000285"/>
<species boundaryCondition="true" compartment="env" id="Ep"
         metaid="_466216"
         sboTerm="SBO:0000252">
  <annotation>
    <rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:bqmodel="http://biomodels.net/model-qualifiers/"
        xmlns:bqbiol="http://biomodels.net/biology-qualifiers/">
      <rdf:Description rdf:about="#_466216">
        <bqbiol:isVersionOf>
          <rdf:Bag>
            <rdf:li rdf:resource="http://identifiers.org/kegg.compound/C00562"/>
          </rdf:Bag>
        </bqbiol:isVersionOf>

        <bqbiol:hasVersion>
          <rdf:Bag>
            <rdf:li rdf:resource="http://identifiers.org/psimod/MOD:00696"/>
          </rdf:Bag>
        </bqbiol:hasVersion>
      </rdf:Description>

    </rdf:RDF>
  </annotation>
</species>
```

```
    <species boundaryCondition="true" compartment="env" id="E"
        metaid="_466219"
       sboTerm="SBO:0000252">
      <annotation>
        <rdf:RDF
            xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:bqmodel="http://biomodels.net/model-qualifiers/"
            xmlns:bqbiol="http://biomodels.net/biology-qualifiers/">
          <rdf:Description rdf:about="#_466219">
            <bqbiol:isVersionOf>
              <rdf:Bag>
                <rdf:li rdf:resource="http://identifiers.org/chebi/CHEBI:36080"/>
              </rdf:Bag>
            </bqbiol:isVersionOf>
          </rdf:Description>

        </rdf:RDF>
      </annotation>
    </species>
  </listOfSpecies>
  <listOfParameters>
    <parameter id="k0" metaid="_466243" sboTerm="SBO:0000035" units="per_s" value="1"/>
    <parameter id="k2" metaid="_466244" sboTerm="SBO:0000036" units="per_M_per_s" value="1"/>
    <parameter id="k3" metaid="_466245" sboTerm="SBO:0000186" units="M_per_s" value="0.5"/>
    <parameter id="J3" metaid="_466248" sboTerm="SBO:0000002" units="dimensionless" value="0.01"/>
    <parameter id="k4" metaid="_466251" sboTerm="SBO:0000025" units="per_s" value="1"/>
    <parameter id="J4" metaid="_466254" sboTerm="SBO:0000002" units="dimensionless" value="0.01"/>
    <parameter id="Et" metaid="_466257" sboTerm="SBO:0000196" units="M" value="1"/>
    <parameter constant="false" id="Km3" metaid="_466260" sboTerm="SBO:0000027" units="M"/>
    <parameter constant="false" id="Km4" metaid="_466263" sboTerm="SBO:0000027" units="M"/>
  </listOfParameters>
  <listOfRules>
    <assignmentRule metaid="_466232" variable="Km3">
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <apply>
          <times/>
          <ci> J3 </ci>
          <ci> Et </ci>
        </apply>
      </math>
    </assignmentRule>
    <assignmentRule metaid="_466235" variable="Km4">
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <apply>
          <times/>
          <ci> J4 </ci>
          <ci> Et </ci>
        </apply>
      </math>
    </assignmentRule>
```

```
  <assignmentRule metaid="_466238" variable="Ep">
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      <apply>
        <minus/>
        <ci> Et </ci>
        <ci> E </ci>
      </apply>
    </math>
  </assignmentRule>
  <assignmentRule metaid="_466241" variable="E">
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      <apply>
        <times/>
        <ci> Et </ci>
        <apply>
          <ci> goldbeter_koshland </ci>
          <ci> k3 </ci>
          <apply>
            <times/>
            <ci> k4 </ci>
            <ci> R </ci>
          </apply>
          <ci> J3 </ci>
          <ci> J4 </ci>
        </apply>
      </apply>
    </math>
  </assignmentRule>
</listOfRules>
<listOfReactions>
  <reaction id="r0" metaid="_466220" reversible="false" sboTerm="SBO:0000176">
    <annotation>
      <rdf:RDF
          xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:bqmodel="http://biomodels.net/model-qualifiers/"
          xmlns:bqbiol="http://biomodels.net/biology-qualifiers/">
        <rdf:Description rdf:about="#_466220">
          <bqbiol:isVersionOf>
            <rdf:Bag>
              <rdf:li rdf:resource="http://identifiers.org/go/GO:0009058"/>
            </rdf:Bag>
          </bqbiol:isVersionOf>
        </rdf:Description>

      </rdf:RDF>
    </annotation>
    <listOfProducts>
      <speciesReference metaid="_024e533e-e719-4fa9-b98a-7a3f250e4482" species="R"/>
    </listOfProducts>
    <listOfModifiers>
```

```
    <modifierSpeciesReference
    metaid="_0f58e551-3ae6-4274-a879-62d8c6612dc6"
    sboTerm="SBO:0000461" species="E"/>
  </listOfModifiers>
  <kineticLaw metaid="bf538c71-85b1-4de9-aeb2-ba76e5a755bf" sboTerm="SBO:0000049">
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      <apply>
        <times/>
        <ci> env </ci>
        <ci> k0 </ci>
        <ci> E </ci>
      </apply>
    </math>
  </kineticLaw>
</reaction>
<reaction id="r2" metaid="_466221" reversible="false" sboTerm="SBO:0000179">
  <annotation>
    <rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:bqmodel="http://biomodels.net/model-qualifiers/"
        xmlns:bqbiol="http://biomodels.net/biology-qualifiers/">
      <rdf:Description rdf:about="#_466221">
        <bqbiol:isVersionOf>
          <rdf:Bag>
            <rdf:li rdf:resource="http://identifiers.org/go/GO:0009056"/>
          </rdf:Bag>
        </bqbiol:isVersionOf>
      </rdf:Description>

    </rdf:RDF>
  </annotation>
  <listOfReactants>
    <speciesReference metaid="_108012dd-11ba-4eb8-b8e4-9597bf2476ca" species="R"/>
  </listOfReactants>
  <listOfModifiers>
    <modifierSpeciesReference
        metaid="_4ca109bf-58ac-42e3-94c7-cbe8dad04767"
        sboTerm="SBO:0000461" species="S"/>
  </listOfModifiers>
  <kineticLaw metaid="c171bdcb-e165-4c1b-940a-f055e99eed93" sboTerm="SBO:0000054">
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      <apply>
        <times/>
        <ci> env </ci>
        <ci> k2 </ci>
        <ci> R </ci>
        <ci> S </ci>
      </apply>
    </math>
  </kineticLaw>
```

```
</reaction>
<reaction id="r3" metaid="_466224" reversible="false" sboTerm="SBO:0000330">
  <annotation>
    <rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:bqmodel="http://biomodels.net/model-qualifiers/"
        xmlns:bqbiol="http://biomodels.net/biology-qualifiers/">
      <rdf:Description rdf:about="#_466224">
        <bqbiol:isVersionOf>
          <rdf:Bag>
            <rdf:li rdf:resource="http://identifiers.org/go/GO:0006470"/>
          </rdf:Bag>
        </bqbiol:isVersionOf>
      </rdf:Description>

    </rdf:RDF>
  </annotation>
  <listOfReactants>
    <speciesReference metaid="_2286d738-1a95-4676-8e71-bbde5b3adb09" species="Ep"/>
  </listOfReactants>
  <listOfProducts>
    <speciesReference metaid="e7d81429-959f-47ee-94eb-47ed2c986168" species="E"/>
  </listOfProducts>
  <kineticLaw metaid="_69717de2-7efe-4f71-a406-48b934fbe87b" sboTerm="SBO:0000029">
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      <apply>
        <divide/>
        <apply>
          <times/>
          <ci> env </ci>
          <ci> k3 </ci>
          <ci> Ep </ci>
        </apply>
        <apply>
          <plus/>
          <ci> Km3 </ci>
          <ci> Ep </ci>
        </apply>
      </apply>
    </math>
  </kineticLaw>
</reaction>
<reaction id="r4" metaid="_466227" reversible="false" sboTerm="SBO:0000216">
  <annotation>
    <rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:bqmodel="http://biomodels.net/model-qualifiers/"
        xmlns:bqbiol="http://biomodels.net/biology-qualifiers/">
      <rdf:Description rdf:about="#_466227">
        <bqbiol:isVersionOf>
```

```
            <rdf:Bag>
              <rdf:li rdf:resource="http://identifiers.org/go/GO:0006468"/>
            </rdf:Bag>
          </bqbiol:isVersionOf>
        </rdf:Description>

      </rdf:RDF>
    </annotation>
    <listOfReactants>
      <speciesReference metaid="_08a384e3-bc10-447d-bba6-77c8979320a5" species="E"/>
    </listOfReactants>
    <listOfProducts>
      <speciesReference metaid="_93eeb224-faac-4b56-9dd6-52e91de8dbce" species="Ep"/>
    </listOfProducts>
    <listOfModifiers>
      <modifierSpeciesReference
          metaid="_1b8a0dfc-2ed2-4bd6-b6f2-eff149e5bed9"
          sboTerm="SBO:0000460" species="R"/>
    </listOfModifiers>
    <kineticLaw metaid="_0ec5b1b0-c47f-42ef-b403-654a6a1e18da" sboTerm="SBO:0000029">
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <apply>
          <divide/>
          <apply>
            <times/>
            <ci> env </ci>
            <ci> k4 </ci>
            <ci> R </ci>
            <ci> E </ci>
          </apply>
          <apply>
            <plus/>
            <ci> Km4 </ci>
            <ci> E </ci>
          </apply>
        </apply>
      </math>
    </kineticLaw>
  </reaction>
 </listOfReactions>
</model>
</sbml>
```

B Example of BioModel B001

B.1 Graph

The graph of the Core SBML network for B001 obtained by our compiler is given in Fig. 7.

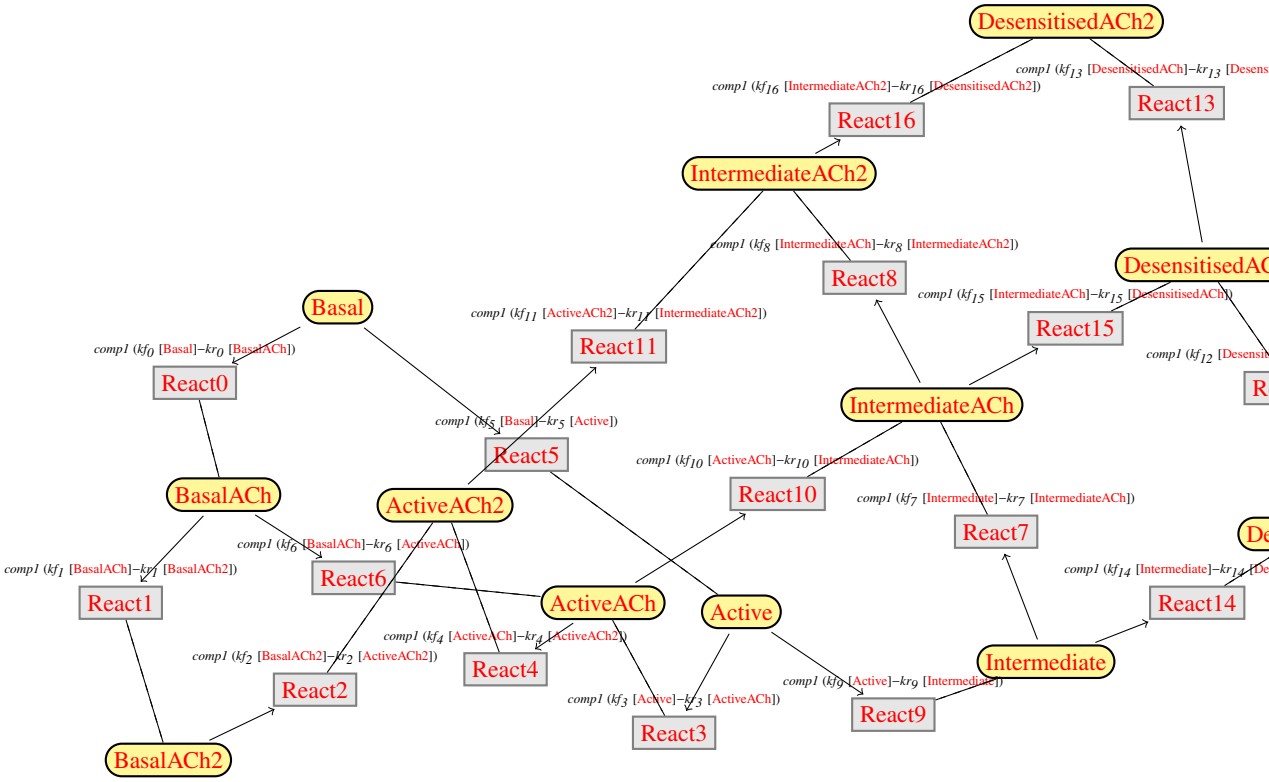

Fig. 7: The graph of the Core SBML model for B001. The volume of compartment is $comp1 = 1e-16$.

B.2 Remaining specification for B001

The macro definitions, initial values, and events of the BioModel BIOMD000000001 are listed in Fig. 12. They are extracted automatically from the Core SBML network in XML format in Section B.3.

B.3 Core SBML Network for B001

The graph of the Core SBML network B001 in Fig. 7 was obtained by our compiler was resented in Fig. 7. Here comes the Core SBML model in XML syntax.

```
<?xml version="1.0" encoding="utf-8"?>
<network id="BIOMD0000000001"
         name="Edelstein1996 - EPSP ACh event"
         version="2.1"><!--
        +++ start species +++++
      --><!-- species without initial assignment -->
```

[BasalACh2]	$\overset{\circ}{=}\frac{\text{BasalACh2}}{comp1}$
[IntermediateACh]	$\overset{\circ}{=}\frac{\text{IntermediateACh}}{comp1}$
[ActiveACh]	$\overset{\circ}{=}\frac{\text{ActiveACh}}{comp1}$
[Active]	$\overset{\circ}{=}\frac{\text{Active}}{comp1}$
[BasalACh]	$\overset{\circ}{=}\frac{\text{BasalACh}}{comp1}$
[Basal]	$\overset{\circ}{=}\frac{\text{Basal}}{comp1}$
[DesensitisedACh2]	$\overset{\circ}{=}\frac{\text{DesensitisedACh2}}{comp1}$
[Desensitised]	$\overset{\circ}{=}\frac{\text{Desensitised}}{comp1}$
[IntermediateACh2]	$\overset{\circ}{=}\frac{\text{IntermediateACh2}}{comp1}$
[DesensitisedACh]	$\overset{\circ}{=}\frac{\text{DesensitisedACh}}{comp1}$
[Intermediate]	$\overset{\circ}{=}\frac{\text{Intermediate}}{comp1}$
[ActiveACh2]	$\overset{\circ}{=}\frac{\text{ActiveACh2}}{comp1}$
kr_0	$\overset{\circ}{=}8000$
kr_1	$\overset{\circ}{=}16000$
kf_2	$\overset{\circ}{=}30000$
kr_2	$\overset{\circ}{=}700$
kr_3	$\overset{\circ}{=}8.64$
kr_4	$\overset{\circ}{=}17.28$
kf_5	$\overset{\circ}{=}0.54$
kr_5	$\overset{\circ}{=}10800$
kf_6	$\overset{\circ}{=}130$
kr_6	$\overset{\circ}{=}2740$
kr_7	$\overset{\circ}{=}4$
kr_8	$\overset{\circ}{=}8$
kf_9	$\overset{\circ}{=}19.7$
kr_9	$\overset{\circ}{=}3.74$
kf_{10}	$\overset{\circ}{=}19.85$
kr_{10}	$\overset{\circ}{=}1.74$
kf_{11}	$\overset{\circ}{=}20$
kr_{11}	$\overset{\circ}{=}0.81$
kr_{12}	$\overset{\circ}{=}4$
kr_{13}	$\overset{\circ}{=}8$
kf_{14}	$\overset{\circ}{=}0.05$
kr_{14}	$\overset{\circ}{=}0.0012$
kf_{15}	$\overset{\circ}{=}0.05$
kr_{15}	$\overset{\circ}{=}0.0012$
kf_{16}	$\overset{\circ}{=}0.05$
kr_{16}	$\overset{\circ}{=}0.0012$
$t2$	$\overset{\circ}{=}20$
$comp1$	$\overset{\circ}{=}1e-16$

Fig. 8: Macro definitions of B001.

kf_0	$\overset{\circ}{=}3000$
kf_1	$\overset{\circ}{=}1500$
kf_3	$\overset{\circ}{=}3000$
kf_4	$\overset{\circ}{=}1500$
kf_7	$\overset{\circ}{=}3000$
kf_8	$\overset{\circ}{=}1500$
kf_{12}	$\overset{\circ}{=}3000$
kf_{13}	$\overset{\circ}{=}1500$

Fig. 9: Initial values of control parameters of B001.

BasalACh2	$\overset{\circ}{=}0$
IntermediateACh	$\overset{\circ}{=}0$
ActiveACh	$\overset{\circ}{=}0$
Active	$\overset{\circ}{=}0$
BasalACh	$\overset{\circ}{=}0$
Basal	$\overset{\circ}{=}1.66057788110262e\text{-}21$
DesensitisedACh2	$\overset{\circ}{=}0$
Desensitised	$\overset{\circ}{=}0$
IntermediateACh2	$\overset{\circ}{=}0$
DesensitisedACh	$\overset{\circ}{=}0$
Intermediate	$\overset{\circ}{=}0$
ActiveACh2	$\overset{\circ}{=}0$

Fig. 10: Initial values of species of B001.

Event	Condition	Updates
RemovalACh	$t2 < time$	$kf_0 := 0$
		$kf_3 := 0$
		$kf_7 := 0$
		$kf_{12} := 0$
		$kf_1 := 0$
		$kf_4 := 0$
		$kf_8 := 0$
		$kf_{13} := 0$

Fig. 11: Events of B001 (using variable *time*).

Fig. 12: Remaining specification of B001.

```
<variable type="species"
          id="BLL"
          latex-look="BasalACh2"
          initial="0"
          concentration="conc_BLL"
          compartment="comp1"/>
<expression id="conc_BLL" latex-look="[BasalACh2]">
   <divide>
      <var type="species" id="BLL"/>
      <expr id="comp1"/>
   </divide>
</expression>
<!-- species without initial assignment -->
<variable type="species"
          id="IL"
          latex-look="IntermediateACh"
          initial="0"
          concentration="conc_IL"
          compartment="comp1"/>
<expression id="conc_IL" latex-look="[IntermediateACh]">
   <divide>
      <var type="species" id="IL"/>
      <expr id="comp1"/>
   </divide>
</expression>
<!-- species without initial assignment -->
<variable type="species"
          id="AL"
          latex-look="ActiveACh"
          initial="0"
          concentration="conc_AL"
          compartment="comp1"/>
<expression id="conc_AL" latex-look="[ActiveACh]">
   <divide>
      <var type="species" id="AL"/>
      <expr id="comp1"/>
   </divide>
</expression>
<!-- species without initial assignment -->
<variable type="species"
          id="A"
          latex-look="Active"
          initial="0"
          concentration="conc_A"
          compartment="comp1"/>
<expression id="conc_A" latex-look="[Active]">
   <divide>
      <var type="species" id="A"/>
      <expr id="comp1"/>
   </divide>
```

```
</expression>
<!-- species without initial assignment -->
<variable type="species"
          id="BL"
          latex-look="BasalACh"
          initial="0"
          concentration="conc_BL"
          compartment="comp1"/>
<expression id="conc_BL" latex-look="[BasalACh]">
   <divide>
      <var type="species" id="BL"/>
      <expr id="comp1"/>
   </divide>
</expression>
<!-- species without initial assignment -->
<variable type="species"
          id="B"
          latex-look="Basal"
          initial="1.66057788110262e-21"
          concentration="conc_B"
          compartment="comp1"/>
<expression id="conc_B" latex-look="[Basal]">
   <divide>
      <var type="species" id="B"/>
      <expr id="comp1"/>
   </divide>
</expression>
<!-- species without initial assignment -->
<variable type="species"
          id="DLL"
          latex-look="DesensitisedACh2"
          initial="0"
          concentration="conc_DLL"
          compartment="comp1"/>
<expression id="conc_DLL" latex-look="[DesensitisedACh2]">
   <divide>
      <var type="species" id="DLL"/>
      <expr id="comp1"/>
   </divide>
</expression>
<!-- species without initial assignment -->
<variable type="species"
          id="D"
          latex-look="Desensitised"
          initial="0"
          concentration="conc_D"
          compartment="comp1"/>
<expression id="conc_D" latex-look="[Desensitised]">
   <divide>
      <var type="species" id="D"/>
```

```
      <expr id="comp1"/>
   </divide>
</expression>
<!-- species without initial assignment -->
<variable type="species"
          id="ILL"
          latex-look="IntermediateACh2"
          initial="0"
          concentration="conc_ILL"
          compartment="comp1"/>
<expression id="conc_ILL" latex-look="[IntermediateACh2]">
   <divide>
      <var type="species" id="ILL"/>
      <expr id="comp1"/>
   </divide>
</expression>
<!-- species without initial assignment -->
<variable type="species"
          id="DL"
          latex-look="DesensitisedACh"
          initial="0"
          concentration="conc_DL"
          compartment="comp1"/>
<expression id="conc_DL" latex-look="[DesensitisedACh]">
   <divide>
      <var type="species" id="DL"/>
      <expr id="comp1"/>
   </divide>
</expression>
<!-- species without initial assignment -->
<variable type="species"
          id="I"
          latex-look="Intermediate"
          initial="0"
          concentration="conc_I"
          compartment="comp1"/>
<expression id="conc_I" latex-look="[Intermediate]">
   <divide>
      <var type="species" id="I"/>
      <expr id="comp1"/>
   </divide>
</expression>
<!-- species without initial assignment -->
<variable type="species"
          id="ALL"
          latex-look="ActiveACh2"
          initial="0"
          concentration="conc_ALL"
          compartment="comp1"/>
<expression id="conc_ALL" latex-look="[ActiveACh2]">
```

```
   <divide>
      <var type="species" id="ALL"/>
      <expr id="comp1"/>
   </divide>
</expression>
<!-- +++ start reactions +++++-->
<reaction id="React0">
   <kinetic-expression>
      <times>
         <expr id="comp1"/>
         <minus>
            <times><!--  control parameter  -->
               <param id="kf_0"/>
               <expr id="conc_B"/>
            </times>
            <times>
               <expr id="kr_0"/>
               <expr id="conc_BL"/>
            </times>
         </minus>
      </times>
   </kinetic-expression>
   <reactant spec="B"/>
   <product spec="BL"/>
   <modifier spec="BL"/>
</reaction>
<reaction id="React1">
   <kinetic-expression>
      <times>
         <expr id="comp1"/>
         <minus>
            <times><!--  control parameter  -->
               <param id="kf_1"/>
               <expr id="conc_BL"/>
            </times>
            <times>
               <expr id="kr_1"/>
               <expr id="conc_BLL"/>
            </times>
         </minus>
      </times>
   </kinetic-expression>
   <reactant spec="BL"/>
   <product spec="BLL"/>
   <modifier spec="BLL"/>
</reaction>
<reaction id="React2">
   <kinetic-expression>
      <times>
         <expr id="comp1"/>
```

```
            <minus>
               <times>
                  <expr id="kf_2"/>
                  <expr id="conc_BLL"/>
               </times>
               <times>
                  <expr id="kr_2"/>
                  <expr id="conc_ALL"/>
               </times>
            </minus>
         </times>
      </kinetic-expression>
      <reactant spec="BLL"/>
      <product spec="ALL"/>
      <modifier spec="ALL"/>
   </reaction>
   <reaction id="React3">
      <kinetic-expression>
         <times>
            <expr id="comp1"/>
            <minus>
               <times><!--  control parameter  -->
                  <param id="kf_3"/>
                  <expr id="conc_A"/>
               </times>
               <times>
                  <expr id="kr_3"/>
                  <expr id="conc_AL"/>
               </times>
            </minus>
         </times>
      </kinetic-expression>
      <reactant spec="A"/>
      <product spec="AL"/>
      <modifier spec="AL"/>
   </reaction>
   <reaction id="React4">
      <kinetic-expression>
         <times>
            <expr id="comp1"/>
            <minus>
               <times><!--  control parameter  -->
                  <param id="kf_4"/>
                  <expr id="conc_AL"/>
               </times>
               <times>
                  <expr id="kr_4"/>
                  <expr id="conc_ALL"/>
               </times>
            </minus>
```

```
      </times>
   </kinetic-expression>
   <reactant spec="AL"/>
   <product spec="ALL"/>
   <modifier spec="ALL"/>
</reaction>
<reaction id="React5">
   <kinetic-expression>
      <times>
         <expr id="comp1"/>
         <minus>
            <times>
               <expr id="kf_5"/>
               <expr id="conc_B"/>
            </times>
            <times>
               <expr id="kr_5"/>
               <expr id="conc_A"/>
            </times>
         </minus>
      </times>
   </kinetic-expression>
   <reactant spec="B"/>
   <product spec="A"/>
   <modifier spec="A"/>
</reaction>
<reaction id="React6">
   <kinetic-expression>
      <times>
         <expr id="comp1"/>
         <minus>
            <times>
               <expr id="kf_6"/>
               <expr id="conc_BL"/>
            </times>
            <times>
               <expr id="kr_6"/>
               <expr id="conc_AL"/>
            </times>
         </minus>
      </times>
   </kinetic-expression>
   <reactant spec="BL"/>
   <product spec="AL"/>
   <modifier spec="AL"/>
</reaction>
<reaction id="React7">
   <kinetic-expression>
      <times>
         <expr id="comp1"/>
```

```
            <minus>
               <times><!--  control parameter  -->
                  <param id="kf_7"/>
                  <expr id="conc_I"/>
               </times>
               <times>
                  <expr id="kr_7"/>
                  <expr id="conc_IL"/>
               </times>
            </minus>
         </times>
      </kinetic-expression>
      <reactant spec="I"/>
      <product spec="IL"/>
      <modifier spec="IL"/>
   </reaction>
   <reaction id="React8">
      <kinetic-expression>
         <times>
            <expr id="comp1"/>
            <minus>
               <times><!--  control parameter  -->
                  <param id="kf_8"/>
                  <expr id="conc_IL"/>
               </times>
               <times>
                  <expr id="kr_8"/>
                  <expr id="conc_ILL"/>
               </times>
            </minus>
         </times>
      </kinetic-expression>
      <reactant spec="IL"/>
      <product spec="ILL"/>
      <modifier spec="ILL"/>
   </reaction>
   <reaction id="React9">
      <kinetic-expression>
         <times>
            <expr id="comp1"/>
            <minus>
               <times>
                  <expr id="kf_9"/>
                  <expr id="conc_A"/>
               </times>
               <times>
                  <expr id="kr_9"/>
                  <expr id="conc_I"/>
               </times>
            </minus>
```

```
      </times>
   </kinetic-expression>
   <reactant spec="A"/>
   <product spec="I"/>
   <modifier spec="I"/>
</reaction>
<reaction id="React10">
   <kinetic-expression>
      <times>
         <expr id="comp1"/>
         <minus>
            <times>
               <expr id="kf_10"/>
               <expr id="conc_AL"/>
            </times>
            <times>
               <expr id="kr_10"/>
               <expr id="conc_IL"/>
            </times>
         </minus>
      </times>
   </kinetic-expression>
   <reactant spec="AL"/>
   <product spec="IL"/>
   <modifier spec="IL"/>
</reaction>
<reaction id="React11">
   <kinetic-expression>
      <times>
         <expr id="comp1"/>
         <minus>
            <times>
               <expr id="kf_11"/>
               <expr id="conc_ALL"/>
            </times>
            <times>
               <expr id="kr_11"/>
               <expr id="conc_ILL"/>
            </times>
         </minus>
      </times>
   </kinetic-expression>
   <reactant spec="ALL"/>
   <product spec="ILL"/>
   <modifier spec="ILL"/>
</reaction>
<reaction id="React12">
   <kinetic-expression>
      <times>
         <expr id="comp1"/>
```

```
            <minus>
               <times><!--  control parameter  -->
                  <param id="kf_12"/>
                  <expr id="conc_D"/>
               </times>
               <times>
                  <expr id="kr_12"/>
                  <expr id="conc_DL"/>
               </times>
            </minus>
         </times>
      </kinetic-expression>
      <reactant spec="D"/>
      <product spec="DL"/>
      <modifier spec="DL"/>
   </reaction>
   <reaction id="React13">
      <kinetic-expression>
         <times>
            <expr id="comp1"/>
            <minus>
               <times><!--  control parameter  -->
                  <param id="kf_13"/>
                  <expr id="conc_DL"/>
               </times>
               <times>
                  <expr id="kr_13"/>
                  <expr id="conc_DLL"/>
               </times>
            </minus>
         </times>
      </kinetic-expression>
      <reactant spec="DL"/>
      <product spec="DLL"/>
      <modifier spec="DLL"/>
   </reaction>
   <reaction id="React14">
      <kinetic-expression>
         <times>
            <expr id="comp1"/>
            <minus>
               <times>
                  <expr id="kf_14"/>
                  <expr id="conc_I"/>
               </times>
               <times>
                  <expr id="kr_14"/>
                  <expr id="conc_D"/>
               </times>
            </minus>
```

```
      </times>
   </kinetic-expression>
   <reactant spec="I"/>
   <product spec="D"/>
   <modifier spec="D"/>
</reaction>
<reaction id="React15">
   <kinetic-expression>
      <times>
         <expr id="comp1"/>
         <minus>
            <times>
               <expr id="kf_15"/>
               <expr id="conc_IL"/>
            </times>
            <times>
               <expr id="kr_15"/>
               <expr id="conc_DL"/>
            </times>
         </minus>
      </times>
   </kinetic-expression>
   <reactant spec="IL"/>
   <product spec="DL"/>
   <modifier spec="DL"/>
</reaction>
<reaction id="React16">
   <kinetic-expression>
      <times>
         <expr id="comp1"/>
         <minus>
            <times>
               <expr id="kf_16"/>
               <expr id="conc_ILL"/>
            </times>
            <times>
               <expr id="kr_16"/>
               <expr id="conc_DLL"/>
            </times>
         </minus>
      </times>
   </kinetic-expression>
   <reactant spec="ILL"/>
   <product spec="DLL"/>
   <modifier spec="DLL"/>
</reaction>
<!-- +++ start parameters +++++ -->
<variable type="control" id="kf_0" latex-look="kf\_0" initial="3000"/>
<expression id="kr_0" latex-look="kr\_0">
   <constant value="8000"/>
```

```
</expression>
<variable type="control" id="kf_1" latex-look="kf\_1" initial="1500"/>
<expression id="kr_1" latex-look="kr\_1">
   <constant value="16000"/>
</expression>
<expression id="kf_2" latex-look="kf\_2">
   <constant value="30000"/>
</expression>
<expression id="kr_2" latex-look="kr\_2">
   <constant value="700"/>
</expression>
<variable type="control" id="kf_3" latex-look="kf\_3" initial="3000"/>
<expression id="kr_3" latex-look="kr\_3">
   <constant value="8.64"/>
</expression>
<variable type="control" id="kf_4" latex-look="kf\_4" initial="1500"/>
<expression id="kr_4" latex-look="kr\_4">
   <constant value="17.28"/>
</expression>
<expression id="kf_5" latex-look="kf\_5">
   <constant value="0.54"/>
</expression>
<expression id="kr_5" latex-look="kr\_5">
   <constant value="10800"/>
</expression>
<expression id="kf_6" latex-look="kf\_6">
   <constant value="130"/>
</expression>
<expression id="kr_6" latex-look="kr\_6">
   <constant value="2740"/>
</expression>
<variable type="control" id="kf_7" latex-look="kf\_7" initial="3000"/>
<expression id="kr_7" latex-look="kr\_7">
   <constant value="4"/>
</expression>
<variable type="control" id="kf_8" latex-look="kf\_8" initial="1500"/>
<expression id="kr_8" latex-look="kr\_8">
   <constant value="8"/>
</expression>
<expression id="kf_9" latex-look="kf\_9">
   <constant value="19.7"/>
</expression>
<expression id="kr_9" latex-look="kr\_9">
   <constant value="3.74"/>
</expression>
<expression id="kf_10" latex-look="kf\_10">
   <constant value="19.85"/>
</expression>
<expression id="kr_10" latex-look="kr\_10">
   <constant value="1.74"/>
```

```
</expression>
<expression id="kf_11" latex-look="kf\_11">
   <constant value="20"/>
</expression>
<expression id="kr_11" latex-look="kr\_11">
   <constant value="0.81"/>
</expression>
<variable type="control" id="kf_12" latex-look="kf\_12" initial="3000"/>
<expression id="kr_12" latex-look="kr\_12">
   <constant value="4"/>
</expression>
<variable type="control" id="kf_13" latex-look="kf\_13" initial="1500"/>
<expression id="kr_13" latex-look="kr\_13">
   <constant value="8"/>
</expression>
<expression id="kf_14" latex-look="kf\_14">
   <constant value="0.05"/>
</expression>
<expression id="kr_14" latex-look="kr\_14">
   <constant value="0.0012"/>
</expression>
<expression id="kf_15" latex-look="kf\_15">
   <constant value="0.05"/>
</expression>
<expression id="kr_15" latex-look="kr\_15">
   <constant value="0.0012"/>
</expression>
<expression id="kf_16" latex-look="kf\_16">
   <constant value="0.05"/>
</expression>
<expression id="kr_16" latex-look="kr\_16">
   <constant value="0.0012"/>
</expression>
<expression id="t2" latex-look="t2">
   <constant value="20"/>
</expression>
<!-- +++ start compartments -->
<!--constant compartment -->
<expression id="comp1" latex-look="comp1">
   <constant value="1e-16"/>
</expression>
<!-- +++ start events -->
<event id="RemovalACh">
   <condition>
      <lt>
         <expr id="t2"/>
         <time/>
      </lt>
   </condition>
   <update><!-- no rule but event -->
```

```
        <var type="control" id="kf_0"/>
        <constant value="0"/>
      </update>
      <update><!-- no rule but event -->
        <var type="control" id="kf_3"/>
        <constant value="0"/>
      </update>
      <update><!-- no rule but event -->
        <var type="control" id="kf_7"/>
        <constant value="0"/>
      </update>
      <update><!-- no rule but event -->
        <var type="control" id="kf_12"/>
        <constant value="0"/>
      </update>
      <update><!-- no rule but event -->
        <var type="control" id="kf_1"/>
        <constant value="0"/>
      </update>
      <update><!-- no rule but event -->
        <var type="control" id="kf_4"/>
        <constant value="0"/>
      </update>
      <update><!-- no rule but event -->
        <var type="control" id="kf_8"/>
        <constant value="0"/>
      </update>
      <update><!-- no rule but event -->
        <var type="control" id="kf_13"/>
        <constant value="0"/>
      </update>
    </event>
</network>
```

B.4 Compilation of B001 to BioCham

From the Core SBML model we obtain the following BioCham 4 network by our compiler.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Network: BIOMD0000000001
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%  reaction rules %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% React0
(comp1 * ((kf_0 * conc_B) - (kr_0 * conc_BL))) for B=>BL.
% React1
(comp1 * ((kf_1 * conc_BL) - (kr_1 * conc_BLL))) for BL=>BLL.
```

```
% React2
(comp1 * ((kf_2 * conc_BLL) - (kr_2 * conc_ALL))) for BLL=>ALL.
% React3
(comp1 * ((kf_3 * conc_A) - (kr_3 * conc_AL))) for A=>AL.
% React4
(comp1 * ((kf_4 * conc_AL) - (kr_4 * conc_ALL))) for AL=>ALL.
% React5
(comp1 * ((kf_5 * conc_B) - (kr_5 * conc_A))) for B=>A.
% React6
(comp1 * ((kf_6 * conc_BL) - (kr_6 * conc_AL))) for BL=>AL.
% React7
(comp1 * ((kf_7 * conc_I) - (kr_7 * conc_IL))) for I=>IL.
% React8
(comp1 * ((kf_8 * conc_IL) - (kr_8 * conc_ILL))) for IL=>ILL.
% React9
(comp1 * ((kf_9 * conc_A) - (kr_9 * conc_I))) for A=>I.
% React10
(comp1 * ((kf_10 * conc_AL) - (kr_10 * conc_IL))) for AL=>IL.
% React11
(comp1 * ((kf_11 * conc_ALL) - (kr_11 * conc_ILL))) for ALL=>ILL.
% React12
(comp1 * ((kf_12 * conc_D) - (kr_12 * conc_DL))) for D=>DL.
% React13
(comp1 * ((kf_13 * conc_DL) - (kr_13 * conc_DLL))) for DL=>DLL.
% React14
(comp1 * ((kf_14 * conc_I) - (kr_14 * conc_D))) for I=>D.
% React15
(comp1 * ((kf_15 * conc_IL) - (kr_15 * conc_DL))) for IL=>DL.
% React16
(comp1 * ((kf_16 * conc_ILL) - (kr_16 * conc_DLL))) for ILL=>DLL.

% bug fix for Time
1 for _ => RealTime.

%%%% initial values %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

present(BLL,0).
present(IL,0).
present(AL,0).
present(A,0).
present(BL,0).
present(B,0.00001660577881102 62).
present(DLL,0).
present(D,0).
present(ILL,0).
```

```
present(DL,0).
present(I,0).
present(ALL,0).

present(RealTime,0.0).

%%%% algebraic variables  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function(conc_BLL=(BLL / comp1)).
function(conc_IL=(IL / comp1)).
function(conc_AL=(AL / comp1)).
function(conc_A=(A / comp1)).
function(conc_BL=(BL / comp1)).
function(conc_B=(B / comp1)).
function(conc_DLL=(DLL / comp1)).
function(conc_D=(D / comp1)).
function(conc_ILL=(ILL / comp1)).
function(conc_DL=(DL / comp1)).
function(conc_I=(I / comp1)).
function(conc_ALL=(ALL / comp1)).

%%%%  parameters %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function(kr_0=8000).
function(kr_1=16000).
function(kf_2=30000).
function(kr_2=700).
function(kr_3=8.64).
function(kr_4=17.28).
function(kf_5=0.54).
function(kr_5=10800).
function(kf_6=130).
function(kr_6=2740).
function(kr_7=4).
function(kr_8=8).
function(kf_9=19.7).
function(kr_9=3.74).
function(kf_10=19.85).
function(kr_10=1.74).
function(kf_11=20).
function(kr_11=0.81).
function(kr_12=4).
function(kr_13=8).
function(kf_14=0.05).
function(kr_14=0.0012).
```

```
function(kf_15=0.05).
function(kr_15=0.0012).
function(kf_16=0.05).
function(kr_16=0.0012).
function(t2=20).

%% compartments %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function(comp1=1e-16).

%%%% events  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
add_event((t2<RealTime),kf_0=0,kf_3=0,kf_7=0,
    kf_12=0,kf_1=0,kf_4=0,kf_8=0,kf_13=0).
```

C Example of BioModel B111

C.1 Graph of Core SBML network for B111

In Fig. 4 we presented a simplified version the Core SBML network. It ignores the unique compartment env, given that its volume is constantly 1 any. The general treatment of compartments yields an equivalent Core SBML network that has the graph in Fig. **??**.

We present the Core SBML network produced by our compiler with the full treatment of compartment volumes from the BioModels SBML network B111.

C.2 Macro Definitions

The of macro definitions of the Core SBML network are given in Fig. 14. For short, we here write *MFP* intead of $MphasePromotingFactor$.

C.3 Core SBML Network for B111

Here comes the Core SBML network for B111 in XML syntax.

```
  <?xml version="1.0" encoding="utf-8"?>
<network id="Novak2001_FissionYeast_CellCycle"
         name="Novak2001_FissionYeast_CellCycle"
         version="2.1"><!--
        +++ start species +++++
      --><!-- species without initial assignment -->
   <variable type="species"
             id="cdc13T"
             latex-look="Total cdc13"
             initial="0.2"
             concentration="conc_cdc13T"
             compartment="cell"/>
   <expression id="conc_cdc13T" latex-look="[Total cdc13]">
      <divide>
         <var type="species" id="cdc13T"/>
```

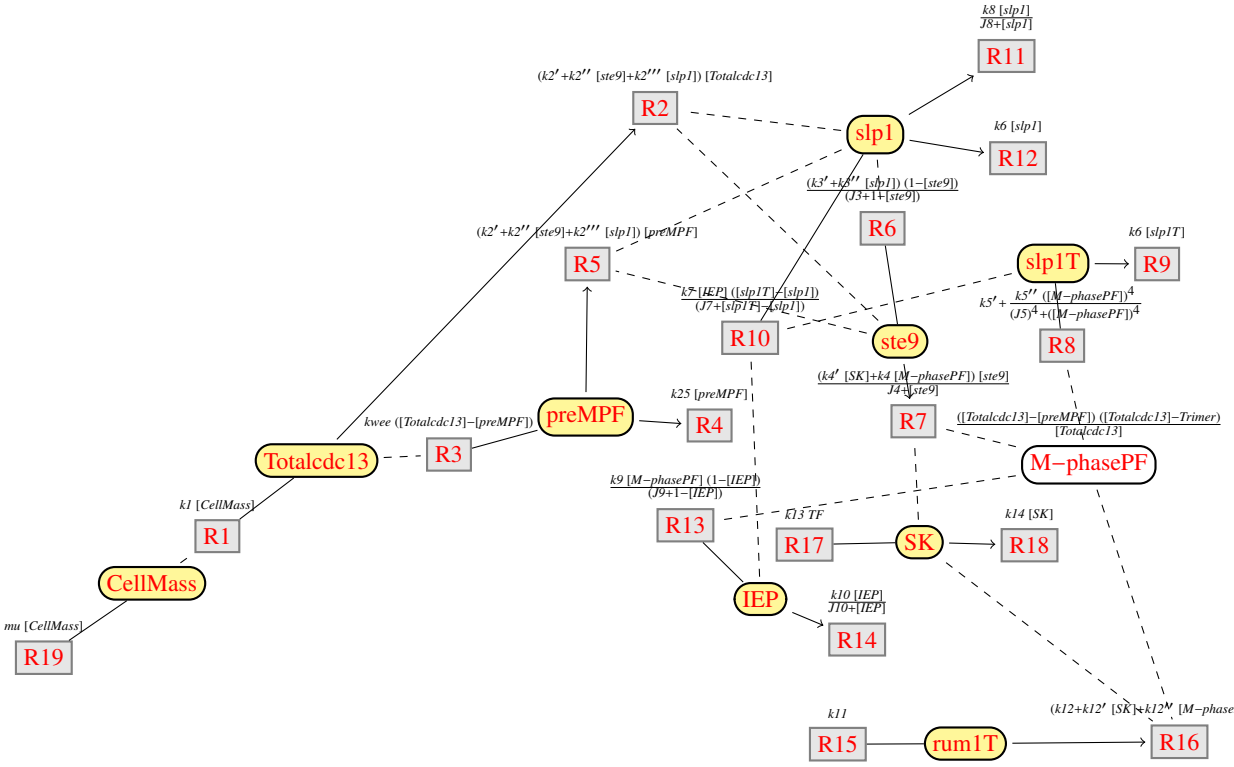

Fig. 13: Graph of Core SBML network for B111.

```
      <expr id="cell"/>
   </divide>
</expression>
<!-- species without initial assignment -->
<variable type="species"
          id="preMPF"
          latex-look="preMPF"
          initial="0"
          concentration="conc_preMPF"
          compartment="cell"/>
<expression id="conc_preMPF" latex-look="[preMPF]">
   <divide>
      <var type="species" id="preMPF"/>
      <expr id="cell"/>
   </divide>
</expression>
<!-- species without initial assignment -->
<variable type="species"
          id="ste9"
```

Macro	Definition
$[Totalcdc13]$	$\frac{\text{Totalcdc13}}{cell}$
$[preMPF]$	$\frac{\text{preMPF}}{cell}$
$[ste9]$	$\frac{\text{ste9}}{cell}$
$[slp1T]$	$\frac{\text{slp1T}}{cell}$
$[slp1]$	$\frac{\text{slp1}}{cell}$
$[IEP]$	$\frac{\text{IEP}}{cell}$
$[rum1T]$	$\frac{\text{rum1T}}{cell}$
$[SK]$	$\frac{\text{SK}}{cell}$
$[CellMass]$	$\frac{\text{CellMass}}{cell}$
$[M{-}phasePF]$	$\frac{\text{M-phasePF}}{cell}$
σ	$[Totalcdc13] + [rum1T] + Kdiss$
$Kdiss$	0.001
$Trimer$	$\frac{2\,[Totalcdc13]\,[rum1T]}{\sigma+\sqrt{((\sigma)^2-4\,[Totalcdc13]\,[rum1T])}}$
TF	$\frac{2\,k15\,[CellMass]\,J16}{(k16'+k16''\,[M{-}phasePF]-k15\,[CellMass])+(k16'+k16''\,[M{-}phasePF])\,J15+k15\,[CellMass]\,J16+\sqrt{(((k16'+k16''\,[M{-}phasePF]-k15\,[CellMass])+(k16'+k16''\,[M{-}ph}$
$k15$	1.5
$J16$	0.01
$k16'$	1
$k16''$	2
$J15$	0.01
$kwee$	$kwee' + (kwee'' - kwee') \frac{2\,Vawee\,Jiwee}{(Viwee\,[M{-}phasePF]-Vawee)+Viwee\,[M{-}phasePF]\,Jawee+Vawee\,Jiwee+\sqrt{(((Viwee\,[M{-}phasePF]-Vawee)+Viwee\,[M{-}phasePF]\,J}$
$kwee'$	0.15
$kwee''$	1.3
$Vawee$	0.25
$Jiwee$	0.01
$Viwee$	1
$Jawee$	0.01
$k25$	$k25' + (k25'' - k25') \frac{2\,Va25\,[M{-}phasePF]\,Ji25}{(Vi25-Va25\,[M{-}phasePF])+Vi25\,Ja25+Va25\,[M{-}phasePF]\,Ji25+\sqrt{(((Vi25-Va25\,[M{-}phasePF])+Vi25\,Ja25+Va25\,[M{-}phasePF]\,Ji25)^2-}$
$k25'$	0.05
$k25''$	5
$Va25$	1
$Ji25$	0.01
$Vi25$	0.25
$Ja25$	0.01
$k1$	0.03
$k2'$	0.03
$k2''$	1
$k2'''$	0.1
$k3'$	1
$k3''$	10
$J3$	0.01
$k4'$	2
$k4$	35
$J4$	0.01
$k5'$	0.005
$k5''$	0.3
$J5$	0.3
$k6$	0.1
$k7$	1
$J7$	0.001
$k8$	0.25
$J8$	0.001
$k9$	0.1
$J9$	0.01
$k10$	0.04
$J10$	0.01
$k11$	0.1
$k12$	0.01
$k12'$	1

```
          latex-look="ste9"
          initial="1"
          concentration="conc_ste9"
          compartment="cell"/>
<expression id="conc_ste9" latex-look="[ste9]">
   <divide>
      <var type="species" id="ste9"/>
      <expr id="cell"/>
   </divide>
</expression>
<!-- species without initial assignment -->
<variable type="species"
          id="slp1T"
          latex-look="slp1T"
          initial="0"
          concentration="conc_slp1T"
          compartment="cell"/>
<expression id="conc_slp1T" latex-look="[slp1T]">
   <divide>
      <var type="species" id="slp1T"/>
      <expr id="cell"/>
   </divide>
</expression>
<!-- species without initial assignment -->
<variable type="species"
          id="slp1"
          latex-look="slp1"
          initial="2.2"
          concentration="conc_slp1"
          compartment="cell"/>
<expression id="conc_slp1" latex-look="[slp1]">
   <divide>
      <var type="species" id="slp1"/>
      <expr id="cell"/>
   </divide>
</expression>
<!-- species without initial assignment -->
<variable type="species"
          id="IEP"
          latex-look="IEP"
          initial="0"
          concentration="conc_IEP"
          compartment="cell"/>
<expression id="conc_IEP" latex-look="[IEP]">
   <divide>
      <var type="species" id="IEP"/>
      <expr id="cell"/>
   </divide>
</expression>
<!-- species without initial assignment -->
```

```
<variable type="species"
          id="rum1T"
          latex-look="rum1T"
          initial="0"
          concentration="conc_rum1T"
          compartment="cell"/>
<expression id="conc_rum1T" latex-look="[rum1T]">
   <divide>
      <var type="species" id="rum1T"/>
      <expr id="cell"/>
   </divide>
</expression>
<!-- species without initial assignment -->
<variable type="species"
          id="SK"
          latex-look="SK"
          initial="0"
          concentration="conc_SK"
          compartment="cell"/>
<expression id="conc_SK" latex-look="[SK]">
   <divide>
      <var type="species" id="SK"/>
      <expr id="cell"/>
   </divide>
</expression>
<!-- species without initial assignment -->
<variable type="species"
          id="M"
          latex-look="Cell Mass"
          initial="1"
          concentration="conc_M"
          compartment="cell"/>
<expression id="conc_M" latex-look="[Cell Mass]">
   <divide>
      <var type="species" id="M"/>
      <expr id="cell"/>
   </divide>
</expression>
<!-- species with assignmentRule -->
<variable type="algebraic"
          id="MPF"
          latex-look="M-phase promoting factor"
          concentration="conc_MPF"
          compartment="cell">
   <kinetic-expression>
      <divide>
         <times>
            <minus>
               <expr id="conc_cdc13T"/>
               <expr id="conc_preMPF"/>
```

```
            </minus>
            <minus>
               <expr id="conc_cdc13T"/>
               <!--  nonconstant parameter without rule nor event-->
               <expr id="Trimer"/>
            </minus>
         </times>
         <expr id="conc_cdc13T"/>
      </divide>
   </kinetic-expression>
</variable>
<expression id="conc_MPF" latex-look="[M-phase promoting factor]">
   <divide>
      <var type="algebraic" id="MPF"/>
      <expr id="cell"/>
   </divide>
</expression>
<!-- +++ start reactions +++++-->
<reaction id="R1">
   <kinetic-expression>
      <times>
         <expr id="k1"/>
         <expr id="conc_M"/>
      </times>
   </kinetic-expression>
   <product spec="cdc13T"/>
   <modifier spec="M"/>
</reaction>
<reaction id="R2">
   <kinetic-expression>
      <times>
         <plus>
            <expr id="k2_prime"/>
            <times>
               <expr id="k2_double_prime"/>
               <expr id="conc_ste9"/>
            </times>
            <times>
               <expr id="k2_triple_prime"/>
               <expr id="conc_slp1"/>
            </times>
         </plus>
         <expr id="conc_cdc13T"/>
      </times>
   </kinetic-expression>
   <reactant spec="cdc13T"/>
   <modifier spec="ste9"/>
   <modifier spec="slp1"/>
</reaction>
<reaction id="R3">
```

```
   <kinetic-expression>
      <times><!--  nonconstant parameter without rule nor event-->
         <expr id="kwee"/>
         <minus>
            <expr id="conc_cdc13T"/>
            <expr id="conc_preMPF"/>
         </minus>
      </times>
   </kinetic-expression>
   <product spec="preMPF"/>
   <modifier spec="cdc13T"/>
</reaction>
<reaction id="R4">
   <kinetic-expression>
      <times><!--  nonconstant parameter without rule nor event-->
         <expr id="k25"/>
         <expr id="conc_preMPF"/>
      </times>
   </kinetic-expression>
   <reactant spec="preMPF"/>
</reaction>
<reaction id="R5">
   <kinetic-expression>
      <times>
         <plus>
            <expr id="k2_prime"/>
            <times>
               <expr id="k2_double_prime"/>
               <expr id="conc_ste9"/>
            </times>
            <times>
               <expr id="k2_triple_prime"/>
               <expr id="conc_slp1"/>
            </times>
         </plus>
         <expr id="conc_preMPF"/>
      </times>
   </kinetic-expression>
   <reactant spec="preMPF"/>
   <modifier spec="ste9"/>
   <modifier spec="slp1"/>
</reaction>
<reaction id="R6">
   <kinetic-expression>
      <divide>
         <times>
            <plus>
               <expr id="k3_prime"/>
               <times>
                  <expr id="k3_double_prime"/>
```

```
                  <expr id="conc_slp1"/>
               </times>
            </plus>
            <minus>
               <constant value="1"/>
               <expr id="conc_ste9"/>
            </minus>
         </times>
         <minus>
            <plus>
               <expr id="J3"/>
               <constant value="1"/>
            </plus>
            <expr id="conc_ste9"/>
         </minus>
      </divide>
   </kinetic-expression>
   <product spec="ste9"/>
   <modifier spec="slp1"/>
</reaction>
<reaction id="R7">
   <kinetic-expression>
      <divide>
         <times>
            <plus>
               <times>
                  <expr id="k4_prime"/>
                  <expr id="conc_SK"/>
               </times>
               <times>
                  <expr id="k4"/>
                  <expr id="conc_MPF"/>
               </times>
            </plus>
            <expr id="conc_ste9"/>
         </times>
         <plus>
            <expr id="J4"/>
            <expr id="conc_ste9"/>
         </plus>
      </divide>
   </kinetic-expression>
   <reactant spec="ste9"/>
   <modifier spec="SK"/>
   <modifier spec="MPF"/>
</reaction>
<reaction id="R8">
   <kinetic-expression>
      <plus>
         <expr id="k5_prime"/>
```

```
            <divide>
               <times>
                  <expr id="k5_double_prime"/>
                  <power>
                     <expr id="conc_MPF"/>
                     <constant value="4"/>
                  </power>
               </times>
               <plus>
                  <power>
                     <expr id="J5"/>
                     <constant value="4"/>
                  </power>
                  <power>
                     <expr id="conc_MPF"/>
                     <constant value="4"/>
                  </power>
               </plus>
            </divide>
         </plus>
      </kinetic-expression>
      <product spec="slp1T"/>
      <modifier spec="MPF"/>
   </reaction>
   <reaction id="R9">
      <kinetic-expression>
         <times>
            <expr id="k6"/>
            <expr id="conc_slp1T"/>
         </times>
      </kinetic-expression>
      <reactant spec="slp1T"/>
   </reaction>
   <reaction id="R10">
      <kinetic-expression>
         <divide>
            <times>
               <expr id="k7"/>
               <expr id="conc_IEP"/>
               <minus>
                  <expr id="conc_slp1T"/>
                  <expr id="conc_slp1"/>
               </minus>
            </times>
            <minus>
               <plus>
                  <expr id="J7"/>
                  <expr id="conc_slp1T"/>
               </plus>
               <expr id="conc_slp1"/>
```

```
            </minus>
         </divide>
      </kinetic-expression>
      <product spec="slp1"/>
      <modifier spec="IEP"/>
      <modifier spec="slp1T"/>
   </reaction>
   <reaction id="R11">
      <kinetic-expression>
         <divide>
            <times>
               <expr id="k8"/>
               <expr id="conc_slp1"/>
            </times>
            <plus>
               <expr id="J8"/>
               <expr id="conc_slp1"/>
            </plus>
         </divide>
      </kinetic-expression>
      <reactant spec="slp1"/>
   </reaction>
   <reaction id="R12">
      <kinetic-expression>
         <times>
            <expr id="k6"/>
            <expr id="conc_slp1"/>
         </times>
      </kinetic-expression>
      <reactant spec="slp1"/>
   </reaction>
   <reaction id="R13">
      <kinetic-expression>
         <divide>
            <times>
               <expr id="k9"/>
               <expr id="conc_MPF"/>
               <minus>
                  <constant value="1"/>
                  <expr id="conc_IEP"/>
               </minus>
            </times>
            <minus>
               <plus>
                  <expr id="J9"/>
                  <constant value="1"/>
               </plus>
               <expr id="conc_IEP"/>
            </minus>
         </divide>
```

```
      </kinetic-expression>
      <product spec="IEP"/>
      <modifier spec="MPF"/>
   </reaction>
   <reaction id="R14">
      <kinetic-expression>
         <divide>
            <times>
               <expr id="k10"/>
               <expr id="conc_IEP"/>
            </times>
            <plus>
               <expr id="J10"/>
               <expr id="conc_IEP"/>
            </plus>
         </divide>
      </kinetic-expression>
      <reactant spec="IEP"/>
   </reaction>
   <reaction id="R15">
      <kinetic-expression>
         <expr id="k11"/>
      </kinetic-expression>
      <product spec="rum1T"/>
   </reaction>
   <reaction id="R16">
      <kinetic-expression>
         <times>
            <plus>
               <expr id="k12"/>
               <times>
                  <expr id="k12_prime"/>
                  <expr id="conc_SK"/>
               </times>
               <times>
                  <expr id="k12_double_prime"/>
                  <expr id="conc_MPF"/>
               </times>
            </plus>
            <expr id="conc_rum1T"/>
         </times>
      </kinetic-expression>
      <reactant spec="rum1T"/>
      <modifier spec="SK"/>
      <modifier spec="MPF"/>
   </reaction>
   <reaction id="R17">
      <kinetic-expression>
         <times>
            <expr id="k13"/>
```

```
         <!--  nonconstant parameter without rule nor event-->
         <expr id="TF"/>
      </times>
   </kinetic-expression>
   <product spec="SK"/>
</reaction>
<reaction id="R18">
   <kinetic-expression>
      <times>
         <expr id="k14"/>
         <expr id="conc_SK"/>
      </times>
   </kinetic-expression>
   <reactant spec="SK"/>
</reaction>
<reaction id="R19">
   <kinetic-expression>
      <times>
         <expr id="mu"/>
         <expr id="conc_M"/>
      </times>
   </kinetic-expression>
   <product spec="M"/>
</reaction>
<!-- +++ start parameters +++++ -->
<variable type="control" id="flag_MPF" latex-look="flag\_MPF" initial="0"/>
<!--   with assignment rule -->
<expression id="sigma" latex-look="\sigma">
   <plus>
      <expr id="conc_cdc13T"/>
      <expr id="conc_rum1T"/>
      <expr id="Kdiss"/>
   </plus>
</expression>
<expression id="Kdiss">
   <constant value="0.001"/>
</expression>
<!--   with assignment rule -->
<expression id="Trimer" latex-look="Trimer">
   <divide>
      <times>
         <constant value="2"/>
         <expr id="conc_cdc13T"/>
         <expr id="conc_rum1T"/>
      </times>
      <plus><!--  nonconstant parameter without rule nor event-->
         <expr id="sigma"/>
         <power>
            <minus>
               <power><!--  nonconstant parameter without rule nor event-->
```

```
                  <expr id="sigma"/>
                  <constant value="2"/>
               </power>
               <times>
                  <constant value="4"/>
                  <expr id="conc_cdc13T"/>
                  <expr id="conc_rum1T"/>
               </times>
            </minus>
            <constant value="0.5"/>
         </power>
      </plus>
   </divide>
</expression>
<!--   with assignment rule -->
<expression id="TF" latex-look="TF">
   <divide>
      <times>
         <constant value="2"/>
         <expr id="k15"/>
         <expr id="conc_M"/>
         <expr id="J16"/>
      </times>
      <plus>
         <minus>
            <plus>
               <expr id="k16_prime"/>
               <times>
                  <expr id="k16_double_prime"/>
                  <expr id="conc_MPF"/>
               </times>
            </plus>
            <times>
               <expr id="k15"/>
               <expr id="conc_M"/>
            </times>
         </minus>
         <times>
            <plus>
               <expr id="k16_prime"/>
               <times>
                  <expr id="k16_double_prime"/>
                  <expr id="conc_MPF"/>
               </times>
            </plus>
            <expr id="J15"/>
         </times>
         <times>
            <expr id="k15"/>
            <expr id="conc_M"/>
```

```
      <expr id="J16"/>
   </times>
   <power>
      <minus>
         <power>
            <plus>
               <minus>
                  <plus>
                     <expr id="k16_prime"/>
                     <times>
                        <expr id="k16_double_prime"/>
                        <expr id="conc_MPF"/>
                     </times>
                  </plus>
                  <times>
                     <expr id="k15"/>
                     <expr id="conc_M"/>
                  </times>
               </minus>
               <times>
                  <plus>
                     <expr id="k16_prime"/>
                     <times>
                        <expr id="k16_double_prime"/>
                        <expr id="conc_MPF"/>
                     </times>
                  </plus>
                  <expr id="J15"/>
               </times>
               <times>
                  <expr id="k15"/>
                  <expr id="conc_M"/>
                  <expr id="J16"/>
               </times>
            </plus>
            <constant value="2"/>
         </power>
         <times>
            <constant value="4"/>
            <minus>
               <plus>
                  <expr id="k16_prime"/>
                  <times>
                     <expr id="k16_double_prime"/>
                     <expr id="conc_MPF"/>
                  </times>
               </plus>
               <times>
                  <expr id="k15"/>
                  <expr id="conc_M"/>
```

```
                  </times>
               </minus>
               <expr id="k15"/>
               <expr id="conc_M"/>
               <expr id="J16"/>
            </times>
         </minus>
         <constant id="div-degree" value="0.5"/>
      </power>
   </plus>
</divide>
</expression>
<expression id="k15">
   <constant value="1.5"/>
</expression>
<expression id="J16">
   <constant value="0.01"/>
</expression>
<expression id="k16_prime">
   <constant value="1"/>
</expression>
<expression id="k16_double_prime">
   <constant value="2"/>
</expression>
<expression id="J15">
   <constant value="0.01"/>
</expression>
<!--   with assignment rule -->
<expression id="kwee" latex-look="kwee">
   <plus>
      <expr id="kwee_prime"/>
      <times>
         <minus>
            <expr id="kwee_double_prime"/>
            <expr id="kwee_prime"/>
         </minus>
         <divide>
            <times>
               <constant value="2"/>
               <expr id="Vawee"/>
               <expr id="Jiwee"/>
            </times>
            <plus>
               <minus>
                  <times>
                     <expr id="Viwee"/>
                     <expr id="conc_MPF"/>
                  </times>
                  <expr id="Vawee"/>
               </minus>
```

```
            <times>
               <expr id="Viwee"/>
               <expr id="conc_MPF"/>
               <expr id="Jawee"/>
            </times>
            <times>
               <expr id="Vawee"/>
               <expr id="Jiwee"/>
            </times>
            <power>
               <minus>
                  <power>
                     <plus>
                        <minus>
                           <times>
                              <expr id="Viwee"/>
                              <expr id="conc_MPF"/>
                           </times>
                           <expr id="Vawee"/>
                        </minus>
                        <times>
                           <expr id="Viwee"/>
                           <expr id="conc_MPF"/>
                           <expr id="Jawee"/>
                        </times>
                        <times>
                           <expr id="Vawee"/>
                           <expr id="Jiwee"/>
                        </times>
                     </plus>
                     <constant value="2"/>
                  </power>
                  <times>
                     <constant value="4"/>
                     <minus>
                        <times>
                           <expr id="Viwee"/>
                           <expr id="conc_MPF"/>
                        </times>
                        <expr id="Vawee"/>
                     </minus>
                     <expr id="Vawee"/>
                     <expr id="Jiwee"/>
                  </times>
               </minus>
               <constant id="div-degree" value="0.5"/>
            </power>
         </plus>
      </divide>
   </times>
```

```
   </plus>
</expression>
<expression id="kwee_prime">
   <constant value="0.15"/>
</expression>
<expression id="kwee_double_prime">
   <constant value="1.3"/>
</expression>
<expression id="Vawee">
   <constant value="0.25"/>
</expression>
<expression id="Jiwee">
   <constant value="0.01"/>
</expression>
<expression id="Viwee">
   <constant value="1"/>
</expression>
<expression id="Jawee">
   <constant value="0.01"/>
</expression>
<!--   with assignment rule -->
<expression id="k25" latex-look="k25">
   <plus>
      <expr id="k25_prime"/>
      <times>
         <minus>
            <expr id="k25_double_prime"/>
            <expr id="k25_prime"/>
         </minus>
         <divide>
            <times>
               <constant value="2"/>
               <expr id="Va25"/>
               <expr id="conc_MPF"/>
               <expr id="Ji25"/>
            </times>
            <plus>
               <minus>
                  <expr id="Vi25"/>
                  <times>
                     <expr id="Va25"/>
                     <expr id="conc_MPF"/>
                  </times>
               </minus>
               <times>
                  <expr id="Vi25"/>
                  <expr id="Ja25"/>
               </times>
               <times>
                  <expr id="Va25"/>
```

```
                  <expr id="conc_MPF"/>
                  <expr id="Ji25"/>
               </times>
               <power>
                  <minus>
                     <power>
                        <plus>
                           <minus>
                              <expr id="Vi25"/>
                              <times>
                                 <expr id="Va25"/>
                                 <expr id="conc_MPF"/>
                              </times>
                           </minus>
                           <times>
                              <expr id="Vi25"/>
                              <expr id="Ja25"/>
                           </times>
                           <times>
                              <expr id="Va25"/>
                              <expr id="conc_MPF"/>
                              <expr id="Ji25"/>
                           </times>
                        </plus>
                        <constant value="2"/>
                     </power>
                     <times>
                        <constant value="4"/>
                        <minus>
                           <expr id="Vi25"/>
                           <times>
                              <expr id="Va25"/>
                              <expr id="conc_MPF"/>
                           </times>
                        </minus>
                        <expr id="Va25"/>
                        <expr id="conc_MPF"/>
                        <expr id="Ji25"/>
                     </times>
                  </minus>
                  <constant id="div-degree" value="0.5"/>
               </power>
            </plus>
         </divide>
      </times>
   </plus>
</expression>
<expression id="k25_prime">
   <constant value="0.05"/>
</expression>
```

```
<expression id="k25_double_prime">
   <constant value="5"/>
</expression>
<expression id="Va25">
   <constant value="1"/>
</expression>
<expression id="Ji25">
   <constant value="0.01"/>
</expression>
<expression id="Vi25">
   <constant value="0.25"/>
</expression>
<expression id="Ja25">
   <constant value="0.01"/>
</expression>
<expression id="k1">
   <constant value="0.03"/>
</expression>
<expression id="k2_prime">
   <constant value="0.03"/>
</expression>
<expression id="k2_double_prime">
   <constant value="1"/>
</expression>
<expression id="k2_triple_prime">
   <constant value="0.1"/>
</expression>
<expression id="k3_prime">
   <constant value="1"/>
</expression>
<expression id="k3_double_prime">
   <constant value="10"/>
</expression>
<expression id="J3">
   <constant value="0.01"/>
</expression>
<expression id="k4_prime">
   <constant value="2"/>
</expression>
<expression id="k4">
   <constant value="35"/>
</expression>
<expression id="J4">
   <constant value="0.01"/>
</expression>
<expression id="k5_prime">
   <constant value="0.005"/>
</expression>
<expression id="k5_double_prime">
   <constant value="0.3"/>
```

```
</expression>
<expression id="J5">
   <constant value="0.3"/>
</expression>
<expression id="k6">
   <constant value="0.1"/>
</expression>
<expression id="k7">
   <constant value="1"/>
</expression>
<expression id="J7">
   <constant value="0.001"/>
</expression>
<expression id="k8">
   <constant value="0.25"/>
</expression>
<expression id="J8">
   <constant value="0.001"/>
</expression>
<expression id="k9">
   <constant value="0.1"/>
</expression>
<expression id="J9">
   <constant value="0.01"/>
</expression>
<expression id="k10">
   <constant value="0.04"/>
</expression>
<expression id="J10">
   <constant value="0.01"/>
</expression>
<expression id="k11">
   <constant value="0.1"/>
</expression>
<expression id="k12">
   <constant value="0.01"/>
</expression>
<expression id="k12_prime">
   <constant value="1"/>
</expression>
<expression id="k12_double_prime">
   <constant value="3"/>
</expression>
<expression id="k13">
   <constant value="0.1"/>
</expression>
<expression id="k14">
   <constant value="0.1"/>
</expression>
<expression id="mu">
```

```
      <constant value="0.005"/>
    </expression>
    <!-- +++ start compartments -->
    <!--constant compartment -->
    <expression id="cell" latex-look="cell">
      <constant value="1"/>
    </expression>
    <!-- +++ start events -->
    <event id="event_0000001">
      <condition>
        <and>
          <leq>
            <expr id="conc_MPF"/>
            <constant value="0.1"/>
          </leq>
          <eq><!--  control parameter  -->
            <var type="control" id="flag_MPF"/>
            <constant value="1"/>
          </eq>
        </and>
      </condition>
      <update><!-- species without boundaryCondition-->
        <var type="species" id="M"/>
        <divide>
          <expr id="conc_M"/>
          <constant value="2"/>
        </divide>
      </update>
      <update><!-- no rule but event -->
        <var type="control" id="flag_MPF"/>
        <constant value="0"/>
      </update>
    </event>
    <event id="event_0000002">
      <condition>
        <lt>
          <constant value="0.1"/>
          <expr id="conc_MPF"/>
        </lt>
      </condition>
      <update><!-- no rule but event -->
        <var type="control" id="flag_MPF"/>
        <constant value="1"/>
      </update>
    </event>
  </network>
```

C.4 Compilation of B111 to BioCham

From the Core SBML model for B111 we obtain the following BioCham 4 network by our compiler.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Network: Novak2001_FissionYeast_CellCycle
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%  reaction rules %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% R1
(k1 * conc_M) for _=>cdc13T.
% R2
((k2_prime + (k2_double_prime * conc_ste9) +
   (k2_triple_prime * conc_slp1)) * conc_cdc13T) for cdc13T=>_.
% R3
(kwee * (conc_cdc13T - conc_preMPF)) for _=>preMPF.
% R4
(k25 * conc_preMPF) for preMPF=>_.
% R5
((k2_prime + (k2_double_prime * conc_ste9) +
    (k2_triple_prime * conc_slp1)) * conc_preMPF) for preMPF=>_.
% R6
(((k3_prime + (k3_double_prime * conc_slp1))
       * (1 - conc_ste9)) / ((J3 + 1) - conc_ste9)) for _=>ste9.
% R7
((((k4_prime * conc_SK) + (k4 * conc_MPF)) *
          conc_ste9) / (J4 + conc_ste9)) for ste9=>_.
% R8
(k5_prime + ((k5_double_prime * (conc_MPF^4)) /
          ((J5^4) + (conc_MPF^4)))) for _=>slp1T.
% R9
(k6 * conc_slp1T) for slp1T=>_.
% R10
((k7 * conc_IEP * (conc_slp1T - conc_slp1)) /
     ((J7 + conc_slp1T) - conc_slp1)) for _=>slp1.
% R11
((k8 * conc_slp1) / (J8 + conc_slp1)) for slp1=>_.
% R12
(k6 * conc_slp1) for slp1=>_.
% R13
((k9 * conc_MPF * (1 - conc_IEP)) /
        ((J9 + 1) - conc_IEP)) for _=>IEP.
% R14
((k10 * conc_IEP) / (J10 + conc_IEP)) for IEP=>_.
```

```
% R15
k11 for _=>rum1T.
% R16
((k12 + (k12_prime * conc_SK) +
   (k12_double_prime * conc_MPF)) *
   conc_rum1T) for rum1T=>_.
% R17
(k13 * TF) for _=>SK.
% R18
(k14 * conc_SK) for SK=>_.
% R19
(mu * conc_M) for _=>M.

%%%% initial values %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

present(cdc13T,0.2).
present(preMPF,0).
present(ste9,1).
present(slp1T,0).
present(slp1,2.2).
present(IEP,0).
present(rum1T,0).
present(SK,0).
present(M,1).

% algebraic variables  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function(conc_cdc13T=(cdc13T / cell)).
function(conc_preMPF=(preMPF / cell)).
function(conc_ste9=(ste9 / cell)).
function(conc_slp1T=(slp1T / cell)).
function(conc_slp1=(slp1 / cell)).
function(conc_IEP=(IEP / cell)).
function(conc_rum1T=(rum1T / cell)).
function(conc_SK=(SK / cell)).
function(conc_M=(M / cell)).
function(conc_MPF=(MPF / cell)).
function(MPF=(((conc_cdc13T - conc_preMPF) *
    (conc_cdc13T - Trimer)) / conc_cdc13T)).

% parameters and expressions  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function(sigma=(conc_cdc13T + conc_rum1T + Kdiss)).
function(Kdiss=0.001).
function(Trimer=((2 * conc_cdc13T * conc_rum1T) /
```

```
   (sigma + sqrt(((sigma^2) - (4 * conc_cdc13T * conc_rum1T))))).
function(TF=((2 * k15 * conc_M * J16) / (((k16_prime +
   (k16_double_prime * conc_MPF)) - (k15 * conc_M)) +
   ((k16_prime + (k16_double_prime * conc_MPF)) * J15) +
   (k15 * conc_M * J16) + sqrt((((((k16_prime +
   (k16_double_prime * conc_MPF)) - (k15 * conc_M)) +
   ((k16_prime + (k16_double_prime * conc_MPF)) * J15) +
   (k15 * conc_M * J16))^2) - (4 * ((k16_prime +
   (k16_double_prime * conc_MPF)) -
   (k15 * conc_M)) * k15 * conc_M * J16))))).
function(k15=1.5).
function(J16=0.01).
function(k16_prime=1).
function(k16_double_prime=2).
function(J15=0.01).
function(kwee=(kwee_prime + ((kwee_double_prime - kwee_prime)
    * ((2 * Vawee * Jiwee) / (((Viwee * conc_MPF) - Vawee) +
    (Viwee * conc_MPF * Jawee) + (Vawee * Jiwee) +
    sqrt((((((Viwee * conc_MPF) - Vawee) +
    (Viwee * conc_MPF * Jawee) + (Vawee * Jiwee))^2) -
    (4 * ((Viwee * conc_MPF) - Vawee) * Vawee * Jiwee)))))))).
function(kwee_prime=0.15).
function(kwee_double_prime=1.3).
function(Vawee=0.25).
function(Jiwee=0.01).
function(Viwee=1).
function(Jawee=0.01).
function(k25=(k25_prime + ((k25_double_prime - k25_prime) *
     ((2 * Va25 * conc_MPF * Ji25) / ((Vi25 - (Va25 * conc_MPF)) +
     (Vi25 * Ja25)+
     (Va25 * conc_MPF * Ji25) + sqrt(((((Vi25 - (Va25 * conc_MPF)) +
     (Vi25 * Ja25) + (Va25 * conc_MPF * Ji25))^2) -
     (4 * (Vi25 - (Va25 * conc_MPF)) * Va25 * conc_MPF * Ji25)))))))).
function(k25_prime=0.05).
function(k25_double_prime=5).
function(Va25=1).
function(Ji25=0.01).
function(Vi25=0.25).
function(Ja25=0.01).
function(k1=0.03).
function(k2_prime=0.03).
function(k2_double_prime=1).
function(k2_triple_prime=0.1).
function(k3_prime=1).
function(k3_double_prime=10).
```

```
function(J3=0.01).
function(k4_prime=2).
function(k4=35).
function(J4=0.01).
function(k5_prime=0.005).
function(k5_double_prime=0.3).
function(J5=0.3).
function(k6=0.1).
function(k7=1).
function(J7=0.001).
function(k8=0.25).
function(J8=0.001).
function(k9=0.1).
function(J9=0.01).
function(k10=0.04).
function(J10=0.01).
function(k11=0.1).
function(k12=0.01).
function(k12_prime=1).
function(k12_double_prime=3).
function(k13=0.1).
function(k14=0.1).
function(mu=0.005).

% for compartment volumes
function(cell=1).

%%%% events  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
add_event(((conc_MPF<=0.1) and (flag_MPF=1)),
   M=(conc_M / 2),
   flag_MPF=0).
add_event((0.1<conc_MPF),
     flag_MPF=1).
```