Systems Biology Markup Language (SBML) Level 3 Proposal: Multistate Features

Nicolas Le Novère, Thomas Simon Shimizu, Andrew Finney
lenov@pasteur.fr,tss26@cus.cam.ac.uk,afinney@cds.caltech.edu

December 11, 2002

Contents

1 Introduction **2**
2 Why this extension? **2**
3 Species [modified] **2**
3.1 Feature [new] . . . 3
3.1.1 State . . . 3
3.1.2 Example of a Feature . . . 3
3.2 InitialState [new] . . . 4
3.2.1 FeatureState . . . 4
3.2.2 Example of an InitialState . . . 4
3.3 Complete example of a species element . . . 5
3.4 Issues . . . 5
4 SpeciesReference [modified] **5**
4.1 Example of a reaction with modified specieReferences . . . 6
5 KineticLaw [modified] **7**
5.1 StateEffect [new] . . . 7
5.1.1 speciesState . . . 8
5.1.1.1 featureCondition . . . 8
5.1.1.2 Example of speciesState . . . 8
5.1.2 States of nascent molecules . . . 8
5.1.2.1 Examples of nascentStates . . . 8
5.2 Example of a kineticLaw influenced by states . . . 9
6 Complete SBML example **10**
References **15**

1 Introduction

This document describes a proposed extension for inclusion in Systems Biology Markup Language (SBML) Level 3. It describes features enabling the inclusion of complexes with several alternative states in models.

This document is not a definition of SBML Level 3 or part of it. This document simply presents various features which could be incorporated into SBML Level 3 as the Systems Biology community wishes. This document is intended for detailed review by that community and to provoke alternative proposals. Throughout this document issues that the authors believe will require further discussion have been highlighted.

For brevity the text of this document is with reference to SBML Level 2 (Finney et al., 2002), i.e. features are described in terms of changes to SBML Level 2. This document uses UML diagrams in the same way except that new features are shown in color.

Note that an alternative introduction to the problem of multistate reactants can be found in Andrew Finney's initial proposal "Complex Species:species with multiple states" (Finney, 2001).

All types proposed in this document will be derived from the `SBase` type. In addition to the `SId` type defined in (Finney et al., 2002), this proposal introduces the `SIdREF` type. One constraint on an attribute of the `SId` type is its uniqueness within the model (or the module, with the forthcoming modular extension). On the contrary, several attributes of the `SIdREF` type can share the same value. However, this value has to be one of the values taken by the attributes of `SId` type.

2 Why this extension?

Many biological macromolecules possess multiple internal states which can affect reaction rates. Typical examples are:

- Different relative atomic coordinates ⇒ Conformational changes or folding
- Covalent modification ⇒ glycosylation, phosphorylation, methylation
- Non-covalent modification ⇒ ion or ligand binding

In modelling reaction systems that involve such molecules, it is possible to treat all the different states as separate species. However the number of possible reactions increases exponentially with the number of reacting species (of course in most cases, not all states will affect every reaction, so the number of reactions which need to be computed separately will be somewhat less than this). Writing out all of these reactions separately is tedious at best, and devastating at worst. It is desirable to have an efficient notation which compresses the redundant information. In addition, some simulators (e.g. StochSim (Morton-Firth and Bray, 1998) or MCell (Stiles et al., 1996)) explicitly consider individual molecules, not populations of molecular species. This calls for a mechanism in SBML which can distinguish between specific instances of the same species.

Andrew's initial proposal (Finney, 2001) allowed a subset of the state-dependent reaction instances that involve different reactant states, but have identical rates, to be grouped together and expressed as a single reaction. This was achieved by defining several new SBML elements such as `complexSpecies` (for defining species with multiple states) and `complexSpeciesInstance` (for distinguishing between specific instances of the same species that take part in a reaction). However, in the case that different states have a different effect on the reaction rate, these had to be defined as separate reactions.

We take a similar, but slightly different approach that introduces some new elements, but attempts to incorporate much of the multistate-specific information by extending the existing SBML 2 elements with optional attributes.

3 Species [modified]

The overall structure of the unfolded `Species` tree is shown in figure 1, and the `Species` type is shown in figure 2.

The "state" of a multi-state molecule is defined collectively by the states of all "features" that it possesses. A "feature" here is a characteristic of the species which can be in one of at least two states that affect certain

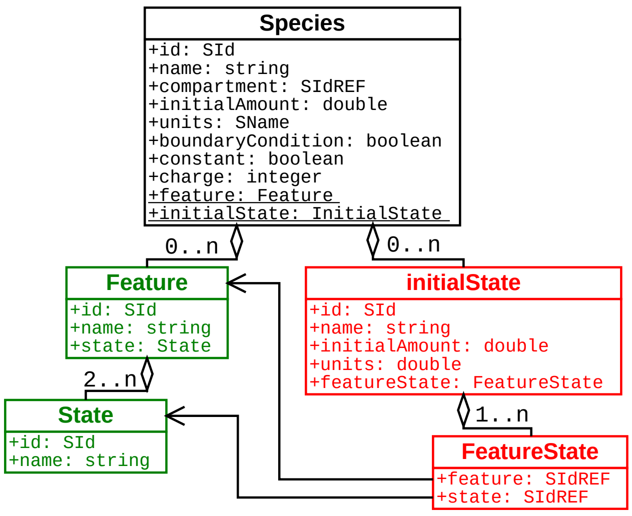

Figure 1: *Complete diagram of the proposed extended `Species` type.*

reaction rates. Therefore the extended `species` element possesses now two new attributes, a `listOfFeatures` and a `listOfInitialStates`.

Species
id: SId
name: string {use="optional"}
compartment: SIdREF
initialAmount: double
units: SName {use="optional"}
boundaryCondition: boolean {use="default" value="false"}
charge: integer {use="optional"}
feature: Feature[0..*] {use="optional"}
initialState: InitialState[0..*] {use="optional"}

Figure 2: *The definition of the proposed extended `Species` type. Addenda are shown in red and green.*

3.1 Feature [new]

The `listOfFeatures` lists all the features of the species which can possess several alternative states.

Feature
id: SId
name: string {use="optional"}
state: State[2..*]

Figure 3: *The definition of a specific Feature attached to a Species type.*

3.1.1 State

Each `feature` element contains a `listOfStates` child, containing at least two `state` element (otherwise one doesn't need this feature, do we?).

3.1.2 Example of a Feature

The following example describes a protein which can exist under various conformations, according to its degree of folding. Therefore we define a feature named "Folding", which here can take three alternative

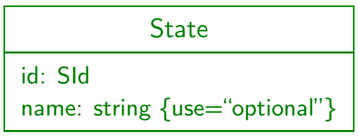

Figure 4: *The definition of one of the states possibly taken by a specific `feature` of a `species` type.*

values: “unfolded”, “folded” and “inactivated” (the latter can correspond to a degradation, a misfolding, or even to an interaction with some kind of other molecule such as a chaperone).

```
<feature name="Folding">
    <listOfStates>
        <state id="F1" name="unfolded">
        <state id="F2" name="folded">
        <state id="F3" name="inactivated">
    </listOfStates>
</feature>
```

To illustrate the differential use of id and name, we’ve used both, although this is obviously not necessary here.

3.2 InitialState [new]

The `listOfInitialStates` expresses the initial amount of each state of the species (that is, specific sets of values taken by each of the features) which is present at the beginning of the simulation. All features with an `initialAmount` different of zero must be listed in the `listOfInitialStates`.

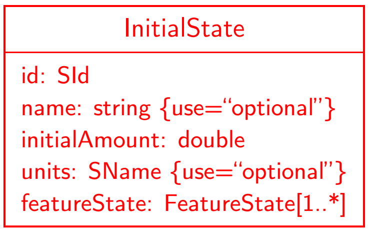
InitialState
id: SId
name: string {use="optional"}
initialAmount: double
units: SName {use="optional"}
featureState: FeatureState[1..*]

Figure 5: *The definition of a specific `InitialState` of a given `Species` type*

The sum of the `initialAmount` of all the `initialStates` (Figure 5) in a `listOfInitialStates` must equal the `initialAmount` of the corresponding species (i.e. all instances of the species are under one state or another).

3.2.1 FeatureState

The `listOfFeatureState` element describes one state of the species, i.e. a unique list of values taken by all the features.

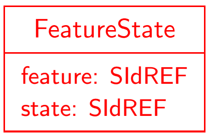

Figure 6: *The definition of a particular state taken by a specific `feature` of a `species` type*

Each `featureState` contains an attribute `feature`, of type Sid, which refers to one of the `feature` elements listed in the `listOfFeature` element. It also contains an attribute `state`, of type Sid, which refers to one of the `state` elements, childs of the element `feature` targeted above.

3.2.2 Example of an InitialState

```
<initialState id="nonactivated" initialAmount="500">
    <listOfFeatureStates>
```

```
        <featureState feature="Folding" state="F1">
        <featureState feature="Phosphorylation" state="P1">
    </listOfFeatureStates>
</initialState>
```

3.3 Complete example of a species element

```
<species name="Species1" initialAmount="1000">
    <listOfFeatures>
        <feature id="Folding">
            <listOfStates>
                <state id="F1" name="unfolded">
                <state id="F2" name="folded">
                <state id="F3" name="inactivated">
            </listOfStates>
        </feature>
        <feature name="Phosphorylation">
            <listOfStates>
                <state id="P1" name="noPhosphate">
                <state id="P2" name="Phosphate">
            </listOfStates>
        </feature>
    </listOfFeatures>
    <listOfInitialStates>
        <initialState id="nonactivated" initialAmount="500">
            <listOfFeatureStates>
                <featureState feature="Folding" state="F1">
                <featureState feature="Phosphorylation" state="P1">
            </listOfFeatureStates>
        </initialState>
        <initialState id="activated" initialAmount="500">
            <listOfFeatureStates>
                <featureState feature="Folding" state="F2">
                <featureState feature="Phosphorylation" state="P2">
            </listOfFeatureStates>
        </initialState>
    </listOfInitialStates>
</species>
```

3.4 Issues

In this proposal, we have attempted to express multistate molecules by extending the `complex` element present in SBML 2. This contrasts from Andrew's initial approach of introducing a novel `complexSpecies` element. We think our approach works quite well, but have we left anything out? Are there any objections to extending the `complex` element using optional attributes?

Confusion could arise from the dual meaning we use for the word "state". A *feature* is one of the many characteristics of a species, which can take a discrete number of **states**. At the same time, a **state** of the *species* itself is defined by a specific set of feature **states**.

The `compartment` element could be conserved for SBML 1 compatibility. However its removal would be coherent with the extension proposed by the ECell group. If a species is defined at the root of the model, it is defined for every compartment (The initialAmount has to be expressed as a concentration then). If a species is defined in a subset of the various compartments, we don't need to specify the compartments since all the definitions are located within the compartments themselves.

4 SpeciesReference [modified]

We present a mechanism which enable the distinction between specific instances of species without any change of the `reaction` element, but affects the `speciesReference` instead (figure 7)

Another possibility, not explored further here, would not affect the `speciesReference`, but would instead adds a `listOfSpeciesInstance` to the `reaction` element.

The addition of the `id` attribute to the `speciesReference` element allows different instances of the same species to be distinguished within a reaction. In the `kineticLaw`, reference to these id would be used instead

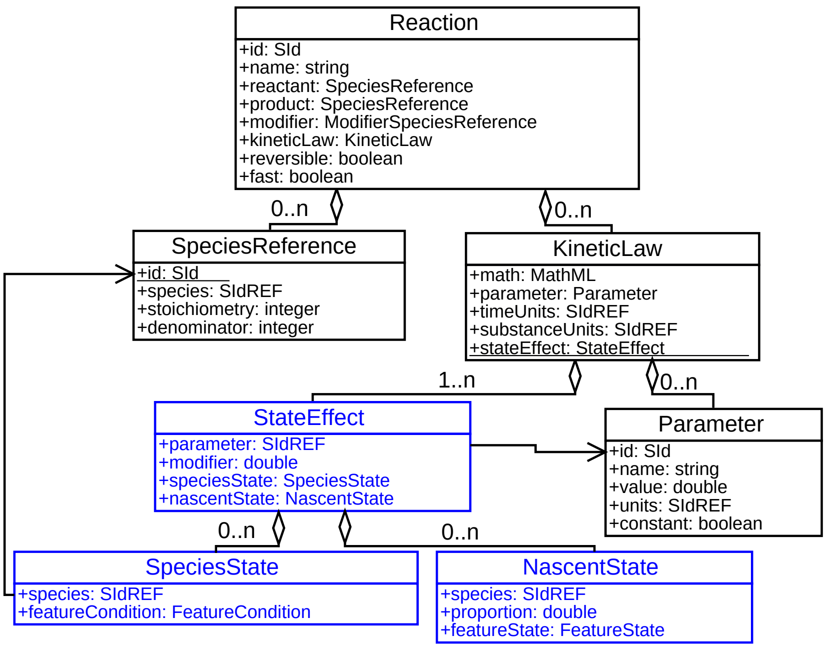

Figure 7: *Complete diagram of the proposed extended* `Reaction` *type.*

of direct references to the species ids. It takes the same value as the `species` by default, so it need not be explicitly defined for reactants that don't need specific instances identified (i.e. if the reactant and product species do not have multiple states, as in SBML 2).

4.1 Example of a reaction with modified specieReferences

An example of a reaction which requires specific instances to be identified is the reversible transfer of a phosphate group between two molecules of species A:

$$A1P + A2 \rightleftharpoons A1 + A2P$$

For this reaction, `listOfReactants` and `listOfProducts` would look like this (see figure 10 for the definition of `stateEffect` and figure 12 for the definition of `featureCondition`):

```
<listOfReactants>
    <speciesReference id="A1" species="A" />
    <speciesReference id="A2" species="A" />
</listOfReactants>
<listOfProducts>
    <speciesReference id="A1" species="A" />
    <speciesReference id="A2" species="A"/>
</listOfProducts>
```

Note that both `speciesReference` elements A1 and A2 point to the same species A.

SpeciesReference
id: SId
species: SIdREF
stoichiometry: integer {use="default" value="1"}
denominator: integer {use="default" value="1"}

Figure 8: *The definition of the extended speciesReference element. Addendum is underlined.*

5 KineticLaw [modified]

We use the concept of a "reaction rate modifier", which defines the effect of the reactant state on a reaction. This has not to be coufused with the modifier species. This allows all state-dependent instances of a reaction to be expressed as a single reaction.

Because the value of a reaction rate modifier can depend on the state of more than one species, we need to tie it to kinetic parameters rather than individual species concentrations.

In this proposal a `listOfStateEffects` is an optional element, child of the `kineticLaw` element (Figure 9).

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

Figure 9: *The definition of the proposed extended `kineticLaw` element. Addenda is shown underlined.*

5.1 StateEffect [new]

The `stateEffect` element specifies which parameter it will modify (referring to its id), the actual value of the modifier (as a real-valued number) a `listOfSpeciesStates` and a `listOfNascentStates` (figure 10).

StateEffect
parameter: SIdREF
modifier: double {use="default" value="0"}
speciesState: SpeciesState[0..*]
nacentState: NascentState[0..*]

Figure 10: *The definition of the stateEffect element.*

The "modifier" attribute is the coefficient which modulates the reaction velocity. It is multiplied with the term of the kinetic law which involves the species to which it is associated. Within the framework of the mass action law, this is effectively equivalent to modifying the quantity of the reacting species.

`speciesState` (there might be a better name for this) is a new element used to specify the set of conditions under which this `stateEffect` applies. The elements of the `listOfSpeciesStates` are interpreted using the AND operator, so the conditions in all elements of the list must be satisfied for the `stateEffect` to apply. If the newly created molecule is also a multistate complex, it is necessary to specify all the state of all of its features. This is easily implemented by creating a new element called `nascentState`.

It is not necessary to have at least one `speciesState` defined. For instance, two simple molecule could give a multistate one. In such a case, we do not need any `speciesState` but we could need a `nascentState` element.

5.1.1 speciesState

Each `speciesState` element refers to either a `species` or a `speciesReference` id (see section 4) and possesses a `listOfFeatureConditions`.

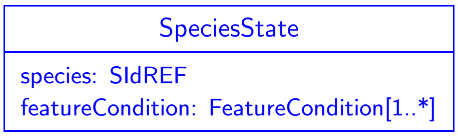

Figure 11: *The definition of the speciesState element.*

5.1.1.1 featureCondition

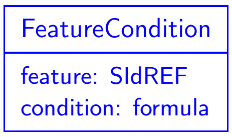

Figure 12: *The definition of the `featureCondition` element.*

Each `featureCondition` element uses logic expressions (consisting of parentheses and the operators AND, OR and NOT) to define the states of each feature that the `stateEffect` applies to. For instance, if a feature has five states A, B, C, D and E, and a `stateEffect` applies to states A or B, one could either write the condition as “A OR B”, or “NOT (C OR D OR E)”.

(Comment: The data type "LogicExpression" could be called LogExpression, LExpression or even LExp, or again BooleanExpression or BoolExpression.)

States which do not match the `listOfFeatureConditions` of any of the `stateEffects` of a reaction are assumed to have a modifier of 0 (i.e. they cannot take part in the reaction).

5.1.1.2 Example of speciesState

```
<speciesState species="species1">
    <listOfFeatureConditions>
        <featureCondition feature="Folding" condition="unfolded OR inactivated"/>
        <featureCondition feature="Phosphorylation" condition="noPhosphate"/>
    </listOfFeatureConditions>
</speciesState>
```

5.1.2 States of nascent molecules

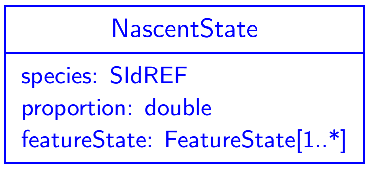

Figure 13: *The definition of a specific `nascentState` of a given `species` type*

If the created molecules come to existence under a specific set of nascent states, which well defined probabilities, we fill the `nascentState` elements. Each element contains an attribute which specify the probability of that state (the default is 1. It is up to the parser to rescale everything if the sum of the `proportion` elements over all the `nascentState` elements is more than 1).

5.1.2.1 Examples of nascentStates

Here is the result of a ligand binding affecting the activity of a species. The species has two features: the presence of ligand, and the activity.

```
<listOfNascentState>
```

```
        <nascentState species="A" proportion="0.9">
            <listOfFeatureState>
                <featureState feature="ligand" state="bound"/>
                <featureState feature="activity" state="active"/>
            </listOfFeatureState>
        </nascentState>
        <nascentState species="A" proportion="0.1">
            <listOfFeatureState>
                <featureState feature="ligand" state="bound"/>
                <featureState feature="activity" state="inactive"/>
            </listOfFeatureState>
        </nascentState>
    </listOfNascentState>
```

In case the nascent state depend on the state(s) of the reactants, we have to enumerate the various reactions, with each `listOfNascentState` limited to one element.

In case no nascent states are defined but the species is nevertheless a multistate one, or in case a feature is missing in the `listOfFeatureState`, the feature values are allocated randomly with a stochastic algorithm, or evenly with a deterministic algorithm.

5.2 Example of a kineticLaw influenced by states

Reactants A and B bind to produce reactant C. Both A and B can be in one of two states, phosphorylated, or not (denoted by a subscript of 1 or 0, respectively). The rates are:

- $A_0 + B_0 \rightarrow C$ — modifier = 0
- $A_0 + B_1 \rightarrow C$ — modifier = 0.6
- $A_1 + B_0 \rightarrow C$ — modifier = 0.2
- $A_1 + B_1 \rightarrow C$ — modifier = 1

In addition, there is a 0.9 probability that the resulting product C is under the active conformation.

```
    <kineticLaw>
        <math xmlns=http://www.w3.org/1998/Math/MathML">
            <apply>
                <times />
                <ci>kf</ci>
                <ci>A</ci>
                <ci>B</ci>
            </apply>
        </math>
        <listOfParameters>
            <parameter name ="kf" value="1000">
        </listOfParameters>
        <listOfStateEffect>
            <stateEffect parameter="kf" modifier="0.6">
                <listOfSpeciesStates>
                    <speciesState species="A">
                        <listOfFeatureConditions>
                            <featureCondition feature="Phosphorylation" condition="nophosphate">
                        </listOfFeatureConditions>
                    </speciesState>
                    <speciesState species="B">
                        <listOfFeatureConditions>
                            <featureCondition feature="Phosphorylation" condition="phosphate">
                        </listOfFeatureConditions>
                    </speciesState>
                </listOfSpeciesStates>
                <listOfNascentState>
                    <nascentState id="C" proportion="0.9">
                        <listOfFeatureState>
                            <featureState feature="Activity" state="active"/>
                        </listOfFeatureState>
                    </nascentState>
                    <nascentState id="C" proportion="0.1">
                        <listOfFeatureState>
                        <featureState feature="Activity" state="inactive"/>
                    </listOfFeatureState>
                </nascentState>
            </listOfNascentState>
        </stateEffect>
        <stateEffect parameter="kf" modifier="0.2">
            <listOfSpeciesStates>
                <speciesState species="A">
                    <listOfFeatureConditions>
                        <featureCondition feature="Phosphorylation" condition="phosphate">
                    </listOfFeatureConditions>
                </speciesState>
                <speciesState species="B">
                    <listOfFeatureConditions>
                        <featureCondition feature="Phosphorylation" condition="nophosphate">
                    </listOfFeatureConditions>
                </speciesState>
            </listOfSpeciesStates>
            <listOfNascentState>
                <nascentState id="C" proportion="0.9">
                    <listOfFeatureState>
                        <featureState feature="Activity" state="active"/>
                    </listOfFeatureState>
                </nascentState>
                <nascentState id="C" proportion="0.1">
                    <listOfFeatureState>
                        <featureState feature="Activity" state="inactive"/>
                    </listOfFeatureState>
                </nascentState>
            </listOfNascentState>
        </stateEffect>
        <stateEffect parameter="kf" modifier="1.0">
            <listOfSpeciesStates>
                <speciesState species="A">
                    <listOfFeatureConditions>
                        <featureCondition feature="Phosphorylation" condition="phosphate">
                    </listOfFeatureConditions>
                </speciesState>
                <speciesState species="B">
                    <listOfFeatureConditions>
                        <featureCondition feature="Phosphorylation" condition="phosphate">
                    </listOfFeatureConditions>
                </speciesState>
            </listOfSpeciesStates>
            <listOfNascentState>
                <nascentState id="C" proportion="0.9">
                    <listOfFeatureState>
                        <featureState feature="Activity" state="active"/>
                    </listOfFeatureState>
                </nascentState>
                <nascentState id="C" proportion="0.1">
                    <listOfFeatureState>
                        <featureState feature="Activity" state="inactive"/>
                    </listOfFeatureState>
                </nascentState>
            </listOfNascentState>
        </stateEffect>
    </listOfStateEffects>
</kineticLaw>
```

6 Complete SBML example

Note that this example has intentionally been made independent of StochSim. In StochSim, the features of multistate complexes are represented by binary flags, so each can only have two states. In this example the feature "Folding" possess three states (this could be encoded in StochSim using two binary flags).

(note: This exemple should be improved to demonstrate nascentStates with proportions not equal to 1).

```
<?xml version="1.0"?>
<sbml xmlns=http://www.sbml.org/sbml/level2" version="1" level="2">
    <model id="ExampleMultipleState" />
```

```
<notes>
    <body xmlns="http://www.w3.org/1999/xhtml">
        <p>This model exemplifies the use of the extension proposed by the StochSim team.</p>
        <p>The main reaction is Species1 + Species2 -> Species3</p>
        <p>Species1 possesses 2 features affecting the reaction rate.
           The "Folding" exists under three states "unfolded", "folded" and "inactivated",
           the "Phosphorylation" under two "noPhosphate" and "Phosphate".</p>
        <p>The features are regulated by two interconversions and one cross-phosphorylation.</p>
        <p>Species1-unfolded -> Species1-folded</p>
        <p>Species1-folded -> Species1-inactivated</p>
        <p>Species1-folded + Species1-inactivated-P <=> Species1-folded-P + Species1-inactivated</p>
    </body>
</notes>

<listOfParameters>
    <parameter id="N_A" value="6.022e23" />
</listOfParameters>

<listOfCompartments>
    <compartment id="compOne" />
</listOfCompartments>

<listOfSpecies>
    <species id="Species1" initialAmount="1000">
        <listOfFeatures>
            <feature id="Folding">
                <listOfStates>
                    <state id="unfolded"/>
                    <state id="folded"/>
                    <state id="inactivated"/>
                </listOfStates>
            </feature>
            <feature id="Phosphorylation">
                <listOfStates>
                    <state id="noPhosphate"/>
                    <state id="Phosphate"/>
                </listOfStates>
            </feature>
        </listOfFeatures>
        <listOfInitialStates>
            <initialState id="nonactivated" initialAmount="500">
                <listOfFeatureStates>
                    <featureState feature="Folding" state="unfolded"/>
                    <featureState feature="Phosphorylation" state="noPhosphate"/>
                </listOfFeatureStates>
            </initialState>
            <initialState name="activated" initialAmount="500">
                <listOfFeatureStates>
                    <featureState feature="Folding" state="folded"/>
                    <featureState feature="Phosphorylation" state="Phosphate"/>
                </listOfFeatureStates>
            </initialState>
        </listOfInitialStates>
    </species>
    <species name="Species2" initialAmount="1000" />
    <species name="Species3" initialAmount="0" />
</listOfSpecies>

<listOfReactions>
    <reaction  id="main">
        <listOfReactants>
            <speciesReference species="Species1" stoichiometry="1"/>
            <speciesReference species="Species2" stoichiometry="1"/>
        </listOfReactants>
        <listOfProducts>
            <speciesReference name="Species3" stoichiometry="1"/>
        </listOfProducts>
        <kineticLaw>
            <math xmlns=http://www.w3.org/1998/Math/MathML">
                <apply>
                    <times>
                    <apply>
                    <times />
                    <ci>kf</ci>
                    <ci>Specie1</ci>
                    <ci>Specie2</ci>
                    <apply>
                </apply>
                <ci>compOne</ci>
                <ci>N_A</ci>
            </apply>
        </math>
        <listOfParameters>
            <parameter id="kf" value="2500" />
        </listOfParameters>
        <listOfStateEffects>
            <stateEffect parameter="kf" modifier="0.5">
                <listOfSpeciesState>
                    <speciesState species="species1">
                        <listOfFeatureConditions>
                            <featureCondition feature="Folding" condition="folded"/>
                            <featureCondition feature="Phosphorylation" condition="noPhosphate"/>
                        </listOfFeatureConditions>
                    </speciesState>
                </listOfSpeciesState>
            </stateEffect>
            <stateEffect parameter="kf" modifier="1">
                <listOfSpeciesState>
                    <speciesState species="species1">
                        <listOfFeatureConditions>
                            <featureCondition feature="Folding" condition="folded"/>
                            <featureCondition feature="Phosphorylation" condition="Phosphate"/>
                        </listOfFeatureConditions>
                    </speciesState>
                </listOfSpeciesState>
            </stateEffect>

           <!-- This one is optional (since the default modifier is 0),
           and will be omitted in later listOfStateEffects.
           It is here to exemplify the logical feature conditions -->

            <stateEffect parameter="kf" modifier="0">
                <listOfSpeciesState>
                    <speciesState species="species1">
                        <listOfFeatureConditions>
                            <featureCondition feature="Folding" condition="unfolded or inactivated"/>
                        </listOfFeatureConditions>
                    </speciesState>
                </listOfSpeciesState>
            </stateEffect>
        </listOfStateEffects>
    </kineticLaw>
</reaction>
<reaction id="folding">
    <listOfReactants>
        <speciesReference species="Species1" stoichiometry="1" />
    </listOfReactants>
    <listOfProducts>
        <speciesReference species="Species1" stoichiometry="1">
    </listOfProducts>
    <kineticLaw>
        <math xmlns=http://www.w3.org/1998/Math/MathML">
            <apply>
                <times>
                <apply>
                    <times />
                    <ci>kf</ci>
                    <ci>Specie1</ci>
                </apply>
                <ci>compOne</ci>
                <ci>N_A</ci>
            </apply>
        </math>
        <listOfParameters>
```

```
                <parameter name="kf" value="1000" />
            </listOfParameters>
            <listOfStateEffects>
                <stateEffect parameter="kf" modifier="1">
                    <listOfSpeciesState>
                        <speciesState species="species1">
                            <listOfFeatureConditions>
                                <featureCondition feature="Folding" condition="unfolded"/>
                            </listOfFeatureConditions>
                        </speciesState>
                    </listOfSpeciesState>
                    <listOfNascentState>
                        <nascentState id="species1" proportion="1">
                            <listOfFeatureState>
                                <featureState feature="Folding" state="folded"/>
                            </listOfFeatureState>
                        </nascentState>
                    </listOfNascentState>
                </stateEffect>
            </listOfStateEffects>
        </kineticLaw>
    </reaction>
    <reaction name="inactivation">
        <listOfReactants>
            <speciesReference species="Species1" stoichiometry="1" />
        </listOfReactants>
        <listOfProducts>
            <speciesReference species="Species1" stoichiometry="1">
        </listOfProducts>
        <kineticLaw>
            <math xmlns=http://www.w3.org/1998/Math/MathML">
                <apply>
                    <times>
                    <apply>
                        <times />
                        <ci>kf</ci>
                        <ci>Specie1</ci>
                    </apply>
                    <ci>compOne</ci>
                    <ci>N_A</ci>
                </apply>
            </math>
            <listOfParameters>
                <parameter id="kf" value="100" />
            </listOfParameters>
            <listOfStateEffects>
                <stateEffect parameter="kf" modifier="1">
                    <listOfSpeciesState>
                        <speciesState species="species1">
                            <listOfFeatureConditions>
                                <featureCondition feature="Folding" condition="folded"/>
                            </listOfFeatureConditions>
                        </speciesState>
                    </listOfSpeciesState>
                    <listOfNascentState>
                        <nascentState id="species1" proportion="1">
                            <listOfFeatureState>
                                <featureState feature="Folding" state="inactivated"/>
                            </listOfFeatureState>
                        </nascentState>
                    </listOfNascentState>
                </stateEffect>
            </listOfStateEffects>
        </kineticLaw>
    </reaction>

    <reaction name="trans-phosphorylation">
        <listOfReactants>
            <speciesReference id="Species1A" species="Species1" stoichiometry="1" />
            <speciesReference id="Species1B" species="Species1" stoichiometry="1" />
        </listOfReactants>
        <listOfProducts>
    <speciesReference id="Species1A" species="Species1" stoichiometry="1" />
    <speciesReference id="Species1B" species="Species1" stoichiometry="1" />
</listOfProducts>
<kineticLaw>
     <math xmlns=http://www.w3.org/1998/Math/MathML">
         <apply>
             <times>
             <apply>
                 <minus>
                 <apply>
                     <times />
                     <ci>kf</ci>
                     <ci>Specie1A</ci>
                     <ci>Specie1B</ci>
                 </apply>
                 <apply>
                     <times />
                     <ci>kr</ci>
                     <ci>Specie1A</ci>
                     <ci>Specie1B</ci>
                 </apply>
             </apply>
             <ci>compOne</ci>
             <ci>N_A</ci>
         </apply>
     </math>
     <listOfParameters>
         <parameter id="kf" value="100" />
         <parameter id="kr" value="0" />
     </listOfParameters>
     <listOfStateEffects>
         <stateEffect parameter="kf" modifier="1">
             <listOfSpeciesState>
                 <speciesState species="species1A">
                     <listOfFeatureConditions>
                         <featureCondition feature="Folding" condition="folded"/>
                         <featureCondition feature="Phosphorylation" condition="noPhosphate"/>
                     </listOfFeatureConditions>
                 </speciesState>
                 <speciesState species="species1B">
                     <listOfFeatureConditions>
                         <featureCondition feature="Folding" condition="inactivated"/>
                         <featureCondition feature="Phosphorylation" condition="Phosphate"/>
                     </listOfFeatureConditions>
                 </speciesState>
             </listOfSpeciesState>
             <listOfNascentState>
                 <nascentState species="species1A" proportion="1">
                     <listOfFeatureState>
                         <featureState feature="Phosphorylation" state="Phosphate"/>
                         <featureState feature="Folding" state="folded"/>
                     </listOfFeatureState>
                 </nascentState>
                 <nascentState species="species1B" proportion="1">
                     <listOfFeatureState>
                         <featureState feature="Phosphorylation" state="noPhosphate"/>
                         <featureState feature="Folding" state="inactivated"/>
                     </listOfFeatureState>
                 </nascentState>
             </listOfNascentState>
         </stateEffect>
         <stateEffect parameter="kr" modifier="1">
             <listOfSpeciesState>
                 <speciesState species="species1A">
                     <listOfFeatureConditions>
                         <featureCondition feature="Folding"  condition="folded"/>
                         <featureCondition feature="Phosphorylation" condition="Phosphate"/>
                     </listOfFeatureConditions>
                 </speciesState>
                 <speciesState species="species1B">
                     <listOfFeatureConditions>
                         <featureCondition feature="Folding" condition="inactivated"/>
```

```
                                         <featureCondition feature="Phosphorylation" condition="noPhosphate"/>
                                     </listOfFeatureConditions>
                                  </speciesState>
                             </listOfSpeciesState>
                             <listOfNascentState>
                                 <nascentState species="species1A" proportion="1">
                                     <listOfFeatureState>
                                         <featureState feature="Phosphorylation" state="noPhosphate"/>
                                         <featureState feature="Folding" state="folded"/>
                                     </listOfFeatureState>
                                 </nascentState>
                                 <nascentState species="species1B" proportion="1">
                                     <listOfFeatureState>
                                         <featureState feature="Phosphorylation" state="Phosphate"/>
                                         <featureState feature="Folding" state="inactivated"/>
                                     </listOfFeatureState>
                                 </nascentState>
                             </listOfNascentState>
                         </stateEffect>
                     </listOfStateEffects>
                 </kineticLaw>
             </reaction>
         </listOfReactions>
    </model>
</sbml>
```

References

Finney, A. (2001). Possible extension to the systems biology markup language. complex species: species with multiple states. Technical report, ERATO Kitano Systems Biology Workbench Development Group. Internal Discussion Document.

Finney, A., Hucka, M., and Boulouri, H. (Augist 23, 2002). Systems biology markup language (sbml) level 2: Structures and facilities for basic model definitions. working draft revision 2. Technical report, Systems Biology Workbench Development Group. Internal Discussion Document.

Morton-Firth, C. and Bray, D. (1998). Predicting temporal fluctuations in an intracellular signalling pathway. *J. Theor. Biol.*, 192:117–128. available via the World Wide Web `http://www.zoo.cam.ac.uk/comp-cell/StochSim.html`.

Stiles, J., Van Helden, D., Bartol, TM, J., Salpeter, E., and Salpeter, M. (1996). iniature endplate current rise times <100 ms from improved dual recordings can be modeled with passive acetylcholine diffusion from a synaptic vesicle. *Proc. Natl. Acad. Sci. USA*, 93. available via the World Wide Web `http://www.mcell.cnl.salk.edu/`.