,230.8 0.8 10.8 0.8 1,,230.8 0.8 10.8 0.8 1,,230.8 0.8 10.8 0.8 1,,230.8 0.8 10.8 0.8 1,

Systems Biology Markup Language (SBML) Level 3 Proposal: multistate components

Nicolas Le Novère, Anika Oellrich
lenov@ebi.ac.uk,anika@ebi.ac.uk

June 23, 2007

Contents

1 Introduction

This document describes a mechanism to describe the state of a component based on a vector of state variables. It is meant to be included in the extension of the Systems Biology Markup Language (SBML) Level 3 that describes features enabling the multi-component complexes (Finney, 2004).

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.

For brevity, the text of this document is with reference to SBML Level 2 Version 3 (Hucka et al., 2007), i.e. features are described in terms of changes to SBML Level 2 Version 3. This document uses UML diagrams in the same way, except that new features are shown in color.

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
- Multimeric state.

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 increase exponentially with the number of reacting species Tolle and Le Novère (2006). Writing out all of these reactions separately is tedious at best, and maybe impossible (modelling Calcium/Calmodulin kinase II with one activity state, binding of ATP and calmodulin, and only one phosphorylation per monomer requires ... 1 billion of billion reactions). Moreover, in most cases not all states will affect every reaction, so the number of reactions which need to be computed separately is considerably lower than all the possible reactions. It is therefore desirable to have an efficient notation which compresses the redundant information. In addition, some simulators (e.g. StochSim (Morton-Firth and Bray, 1998; Le Novère and Shimizu, 2001) 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.

3 Description of the state variables of a SpeciesType

The list of state variables, and the list of possible values taken by the state variable are specific of a type of species, and therefore described with the `SpeciesType`. A `speciesType` can carry any number of state variables. A state variable can exist under any number of alternative values.

Comment: Should-we allow state-variable taking continuous values? Should-we make possible to describe state variable values with a mathematical function?

The following code describe a speciesType CaMKII that possesses two state variables, "activity" and "phosphorylation".

```
<speciesType id="CaMKII" name="Calcium/Calmodulin Kinase II">
  <listOfStateVariables>
    <stateVariable id="activ" name="activity">
      <listOfPossibleStateVariableValues>
        <possibleStateVariableValue id="inact" name="inactive" />
        <possibleStateVariableValue id="act"   name="active"   />
        <possibleStateVariableValue id="inhib" name="inhibited"/>
      </listOfPossibleStateVariableValues>
    </stateVariable>
    <stateVariable id="T286" name="threonine 286">
      <listOfPossibleStateVariableValues>
        <possibleStateVariableValue id="unphos" name="non-phosphorylated"/>
        <possibleStateVariableValue id="phos"   name="phosphorylated"    />
      </listOfPossibleStateVariableValues>
    </stateVariable>
  </listOfStateVariables>
</speciesType>
```

A species belonging to this speciesType could therefore exist under six different states, characterised by the following state variable vectors:

```
(inact, unphos)
(inact, phos)
(act, unphos)
(act, phos)
(inhib, unphos)
(inhib, phos)
```

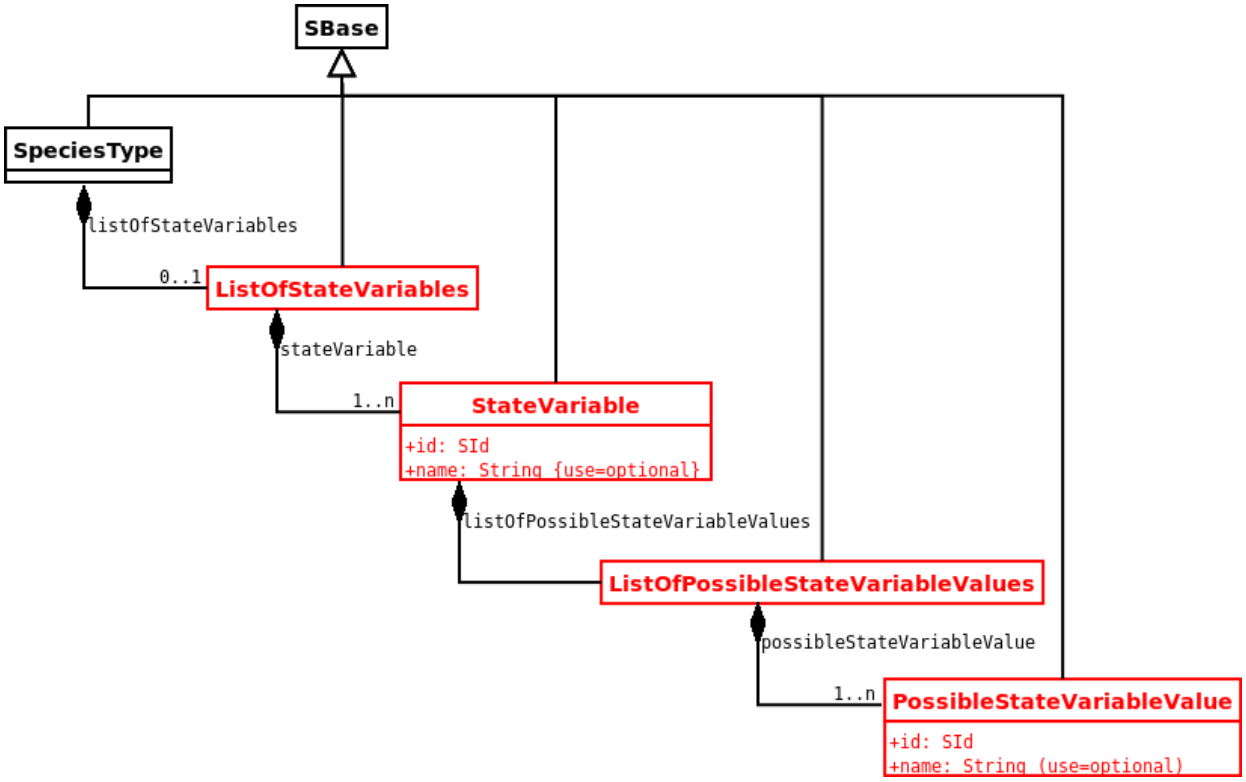


Figure 1: *Elements and attributes necessary to the description of the state variables of a* `SpeciesType`*, and all their possible values.*

4 Declaration of the initial states of a Species

One can describe the state of some instances of a species type (i.e. a subpopulation of the pool represented by a species) at the start of a simulation by setting some of their state variables. The state of a species instance is set-up using the element `SpeciesInstance`. The state variable vector of a species instance is described by a list of state variable instances.

Not all the state variables of a species instance need to be specified. The state variables that are unspecified are considered as wildcard, and their value is choosen by the simulation tool.

The quantity of species instances with a given vector of state variable values to be present at the start of the simulation can be expressed in amount, that is the number (in *unit of substance*) of species instances, in concentration, that is the number (in *unit of substance*) of species instances divided by the size of the compartment containing the species, or in proportion, that is the number (resp. concentration) of species instances relative to the total amount (resp. concentration) of this species. In the latter case, quantities and concentrations of species instances can be obtained by multiplying the proportion by the initialAmount or initialConcentration of the species. The units of substance and of size are the one defined in the `species` and the relevant `compartment`.

The following example describes a situation where the simulation starts with 10% of the CaMKII molecules are active (that is 50) and 90% are inactive (that is 450). None of the molecules is phosphorylated.

```
<species id="K_PSD" name="CaMKII in postsynaptic density"
         speciesType="CaMKII" compartment="PSD" initialAmount="500">
  <listOfInitialSpeciesInstances>
    <initialSpeciesInstance initialProportion="0.9" >
      <listOfStateVariableInstances>
        <stateVariableInstance stateVariable="activ" >
          <listOfStateVariableValues>
            <stateVariableValue possibleStateVariableValue="inact" />
          </listOfStateVariableValues>
        </stateVariableInstance>
        <stateVariableInstance stateVariable="T286" >
```

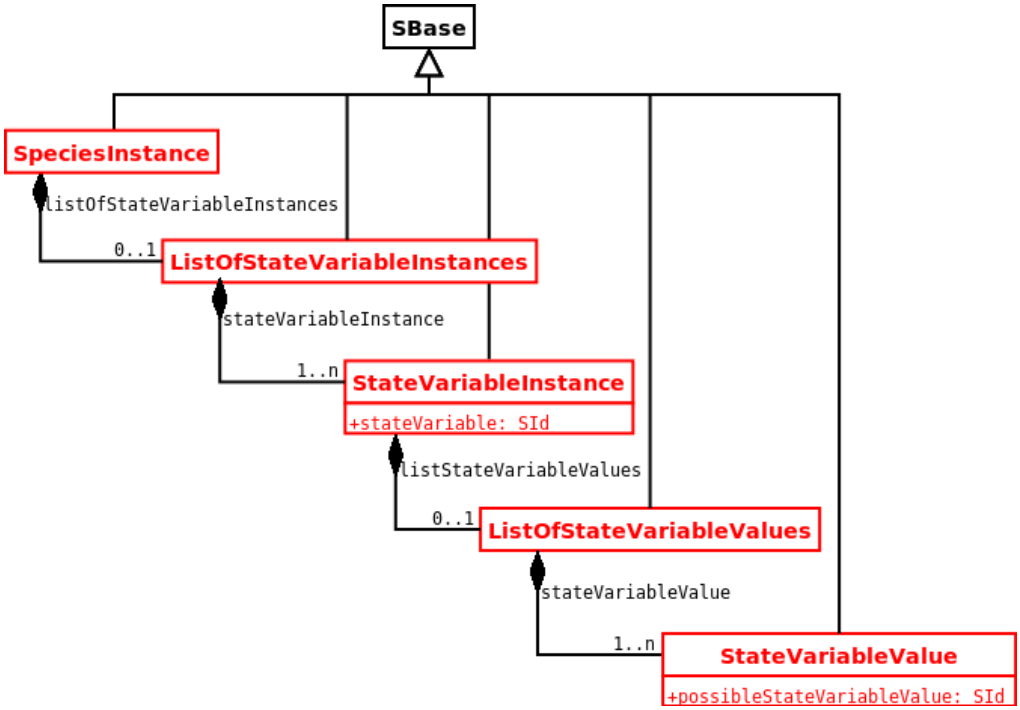


Figure 2: *Diagram of the proposed* `SpeciesInstance`*, that describe the state of a species instance.*

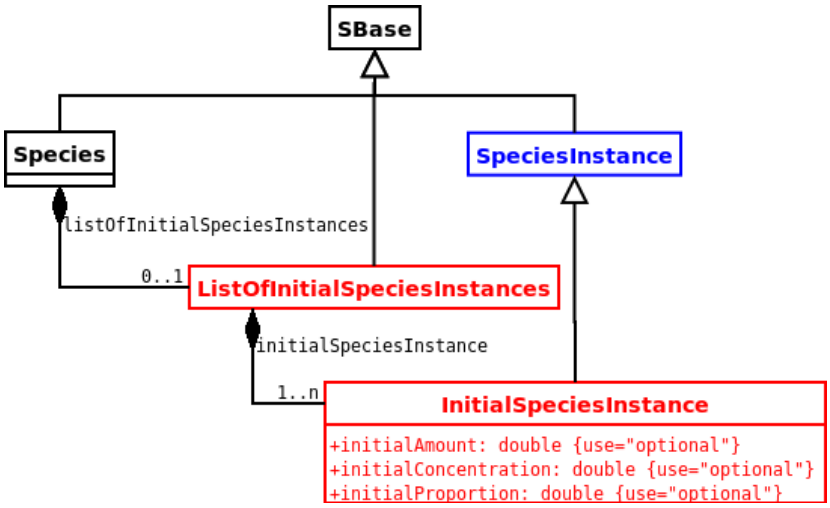


Figure 3: *Diagram of the proposed extended* `Species`*, with the description of the initial states of the corresponding species instances.*

```
      <listOfStateVariableValues>
        <stateVariableValue possibleStateVariableValue="unphos" />
      </listOfStateVariableValues>
    </stateVariableInstance>
  </listOfStateVariableInstances>
</initialSpeciesInstance>
<initialSpeciesInstance initialProportion="0.1" >
  <listOfStateVariableInstances>
    <stateVariableInstance stateVariable="activ" >
      <listOfStateVariableValues>
        <stateVariableValue possibleStateVariableValue="act" />
      </listOfStateVariableValues>
    </stateVariableInstance>
    <stateVariableInstance stateVariable="T286" >
      <listOfStateVariableValues>
```

```
            <stateVariableValue possibleStateVariableValue="unphos" />
          </listOfStateVariableValues>
        </stateVariableInstance>
      </listOfStateVariableInstances>
    </initialSpeciesInstance>
  </listOfInitialSpeciesInstances>
</species>
```

Note that the example above contains speciesInstances with states fully specified. But some state variable values could be let unspecified. It is then up to the software to decide of the distribution over the possible state variable values. For instance, the following example states that 100 of the 500 molecules have to be phosphorylated. However, the state of activity is let to the tool (The activity could be set-up by a rapid equilibrium before each iteration for instance. In that case, to set a value here is useless).

```
<species id="K_PSD" name="CaMKII in postsynaptic density"
         speciesType="CaMKII" compartment="PSD" initialAmount="500">
  <listOfInitialSpeciesInstances>
    <initialSpeciesInstance initialAmount="100" >
      <listOfStateVariableInstances>
        <stateVariableInstance stateVariable="T286" >
          <listOfStateVariableValues>
            <stateVariableValue possibleStateVariableValue="phos" />
          </listOfStateVariableValues>
        </stateVariableInstance>
      </listOfStateVariableInstances>
    </initialSpeciesInstance>
    <initialSpeciesInstance initialAmount="400" >
      <listOfStateVariableInstances>
        <stateVariableInstance stateVariable="T286" >
          <listOfStateVariableValues>
            <stateVariableValue possibleStateVariableValue="unphos" />
          </listOfStateVariableValues>
        </stateVariableInstance>
      </listOfStateVariableInstances>
    </initialSpeciesInstance>
  </listOfInitialSpeciesInstances>
</species>
```

State variable are not obligatory binary entities, and the initial status of a state variable could be incompletely specified. The following example shows that the molecules can be either active or inactive, but not inhibited at the beginning of the simulation.

```
<species id="K_PSD" name="CaMKII in postsynaptic density"
         speciesType="CaMKII" compartment="PSD" initialAmount="500">
  <listOfInitialSpeciesInstances>
    <initialSpeciesInstance initialAmount="500" >
      <listOfStateVariableInstances>
        <stateVariableInstance stateVariable="activ" >
          <listOfStateVariableValues>
            <stateVariableValue possibleStateVariableValue="act" />
            <stateVariableValue possibleStateVariableValue="inact" />
          </listOfStateVariableValues>
        </stateVariableInstance>
      </listOfStateVariableInstances>
    </initialSpeciesInstance>
  </listOfInitialSpeciesInstances>
</species>
```

State variable values can be set separately, and the actual initial species instances reconstructed by the software at run time. This allows to encode very complex cases, where the state variable distributions are obtained independently. The following two examples should lead to the same distribution of state variable vectors:

```
(inact, unphos) = 0.1
(inact, phos)   = 0.2
(act, unphos)   = 0.3
(act, phos)     = 0.4
```

The first example fully specify the state variable vector:

```
<species id="K_PSD" name="CaMKII in postsynaptic density"
         speciesType="CaMKII" compartment="PSD" initialAmount="500">
  <listOfInitialSpeciesInstances>
    <initialSpeciesInstance initialProportion="0.1" >
      <listOfStateVariableInstances>
        <stateVariableInstance stateVariable="activ" >
          <listOfStateVariableValues>
            <stateVariableValue possibleStateVariableValue="inact" />
          </listOfStateVariableValues>
        </stateVariableInstance>
        <stateVariableInstance stateVariable="T286" >
          <listOfStateVariableValues>
            <stateVariableValue possibleStateVariableValue="unphos" />
          </listOfStateVariableValues>
        </stateVariableInstance>
      </listOfStateVariableInstances>
    </initialSpeciesInstance>
    <initialSpeciesInstance initialProportion="0.2" >
      <listOfStateVariableInstances>
        <stateVariableInstance stateVariable="activ" >
          <listOfStateVariableValues>
            <stateVariableValue possibleStateVariableValue="inact" />
          </listOfStateVariableValues>
        </stateVariableInstance>
        <stateVariableInstance stateVariable="T286" >
          <listOfStateVariableValues>
            <stateVariableValue possibleStateVariableValue="phos" />
          </listOfStateVariableValues>
        </stateVariableInstance>
      </listOfStateVariableInstances>
    </initialSpeciesInstance>
    <initialSpeciesInstance initialProportion="0.3" >
      <listOfStateVariableInstances>
        <stateVariableInstance stateVariable="activ" >
          <listOfStateVariableValues>
            <stateVariableValue possibleStateVariableValue="act" />
          </listOfStateVariableValues>
        </stateVariableInstance>
        <stateVariableInstance stateVariable="T286" >
          <listOfStateVariableValues>
            <stateVariableValue possibleStateVariableValue="unphos" />
          </listOfStateVariableValues>
        </stateVariableInstance>
      </listOfStateVariableInstances>
    </initialSpeciesInstance>
    <initialSpeciesInstance initialProportion="0.4" >
      <listOfStateVariableInstances>
        <stateVariableInstance stateVariable="activ" >
          <listOfStateVariableValues>
            <stateVariableValue possibleStateVariableValue="act" />
          </listOfStateVariableValues>
        </stateVariableInstance>
        <stateVariableInstance stateVariable="T286" >
          <listOfStateVariableValues>
            <stateVariableValue possibleStateVariableValue="phos" />
          </listOfStateVariableValues>
        </stateVariableInstance>
      </listOfStateVariableInstances>
    </initialSpeciesInstance>
  </listOfInitialSpeciesInstances>
</species>
```

On the contrary, in the following we specify independently the distributions of the different state variables:

```
<species id="K_PSD" name="CaMKII in postsynaptic density"
         speciesType="CaMKII" compartment="PSD" initialAmount="500">
  <listOfInitialSpeciesInstances>
    <initialSpeciesInstance initialProportion="0.3" >
      <listOfStateVariableInstances>
        <stateVariableInstance stateVariable="activ" >
          <listOfStateVariableValues>
            <stateVariableValue possibleStateVariableValue="inact" />
          </listOfStateVariableValues>
```

```
        </stateVariableInstance>
      </listOfStateVariableInstances>
    </initialSpeciesInstance>
    <initialSpeciesInstance initialProportion="0.7" >
      <listOfStateVariableInstances>
        <stateVariableInstance stateVariable="activ" >
          <listOfStateVariableValues>
            <stateVariableValue possibleStateVariableValue="act" />
          </listOfStateVariableValues>
        </stateVariableInstance>
      </listOfStateVariableInstances>
    </initialSpeciesInstance>
    <initialSpeciesInstance initialProportion="0.4" >
      <listOfStateVariableInstances>
        <stateVariableInstance stateVariable="T286" >
          <listOfStateVariableValues>
            <stateVariableValue possibleStateVariableValue="unphos" />
          </listOfStateVariableValues>
        </stateVariableInstance>
      </listOfStateVariableInstances>
    </initialSpeciesInstance>
    <initialSpeciesInstance initialProportion="0.6" >
      <listOfStateVariableInstances>
        <stateVariableInstance stateVariable="T286" >
          <listOfStateVariableValues>
            <stateVariableValue possibleStateVariableValue="phos" />
          </listOfStateVariableValues>
        </stateVariableInstance>
      </listOfStateVariableInstances>
    </initialSpeciesInstance>
  </listOfInitialSpeciesInstances>
</species>
```

5 Usage of species instances in a reaction affecting state variable values

The state of a species can be involved in two different ways in a reaction. Firstly, a species can react differently according to its state. This information is stored in a set of reacting species instances, that describes the effect of the different states of a `SimpleSpeciesReference` (that is a reactant, a product or a modifier) on a kineticLaw. Secondly, a reaction can generate species instances in different states. This information is stored in a set of nascent species instances on reactant and products.

Because reactions can be reversible, any speciesReference can carry both a `listOfReactingSpeciesInstances` and a `listOfNascentSpeciesInstances`. In the case of an irreversible reaction, the reactants cannot carry a `listOfNascentSpeciesInstances` and the products cannot carry a `listOfReactingSpeciesInstances`.

A modifier can only carry a `listOfReactingSpeciesInstances`, that precises the states of the species acting as a modifier that actually affect the reaction.

The MathML element of the `speciesInstanceInfluence` describes the mathematical transformation to apply to the kineticLaw of the reaction when this speciesInstance is affecting the reaction. A mandatory parameter, not listed in the `listOfParameters` is the id of the reaction, representing the value produced by the `kineticLaw`. Because the order of the mathml operations can be significant, so is the order of the speciesReferences: (f(x) * A) - B is different from (f(x) - B) * A

The following example shows that: a) CaMKII cannot be phosphorylated when it is inactive *and* phosphorylated, and b) the result of a phosphorylation is a phosphorylated molecule that active in 90% of the cases. Note that below we use the attribute `species` on the `speciesReference`. A use of `speciesType` instead, a proposed extension of the core SBML, would result in a generalised reaction, that could take place in any compartment where a species of this speciesType is present.

```
<listOfReactants>
  <speciesReference species="K_PSD">
    <listOfReactingSpeciesInstances>
      <reactingSpeciesInstance>
        <listOfStateVariableInstances>
          <stateVariableInstance stateVariable="activ" >
```

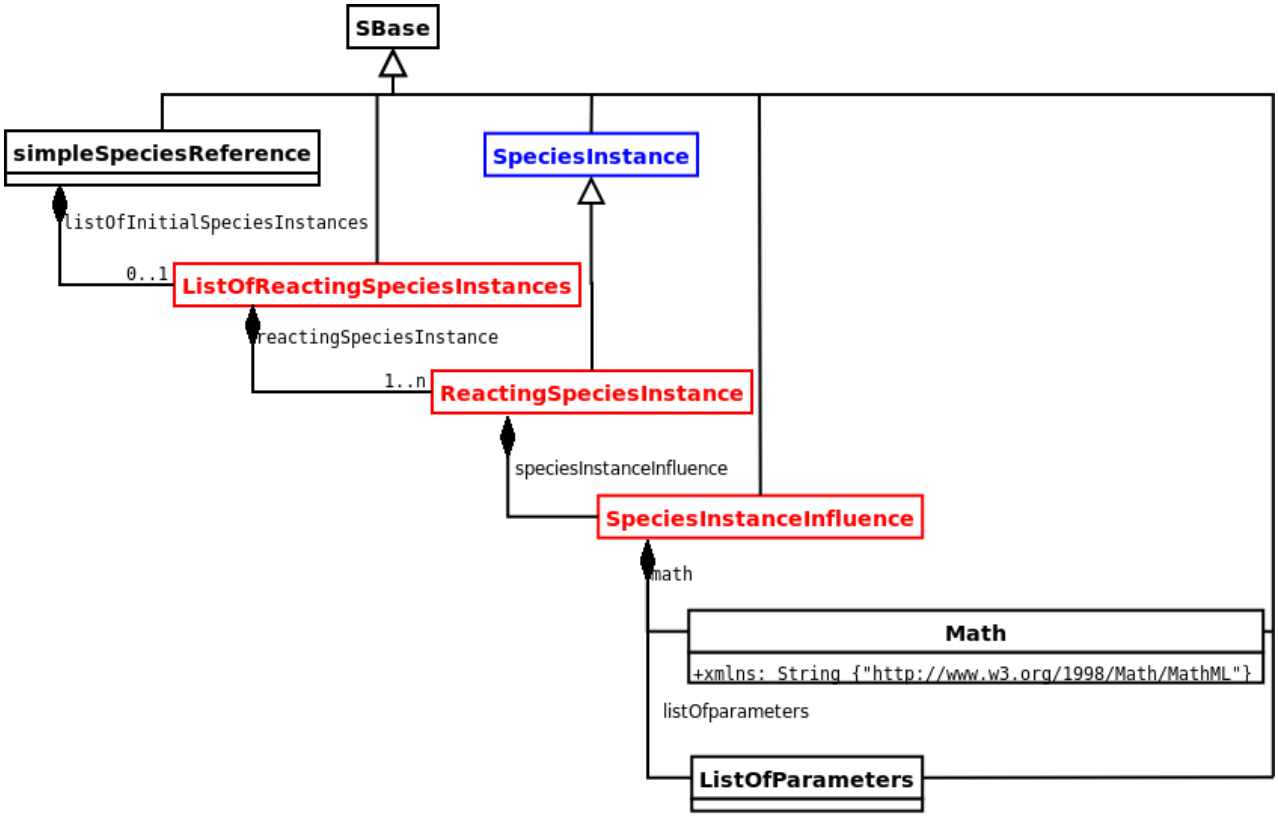


Figure 4: *Complete diagram of the proposed extended* `simpleSpeciesReference` *type, that affects reactants, products and modifiers.*

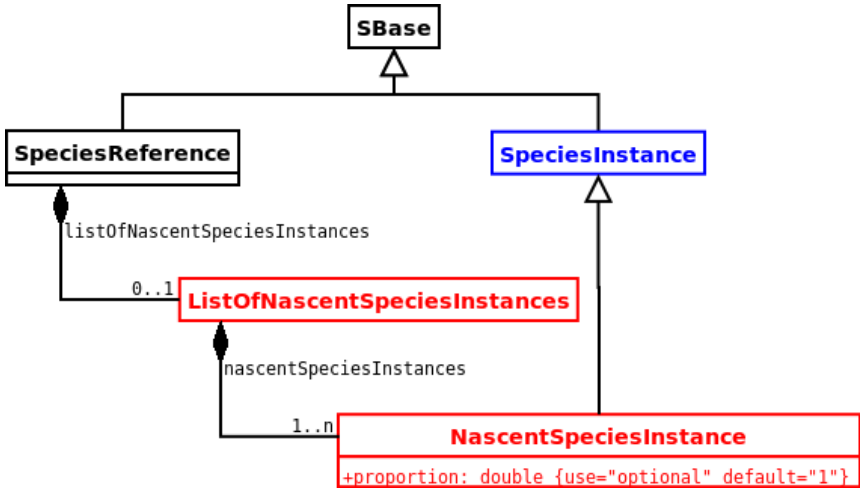


Figure 5: *Complete diagram of the proposed extended* `speciesReference` *type, that only affects reactants and products.*

```
    <listOfStateVariableValues>
      <stateVariableValue possibleStateVariableValue="inact" />
    </listOfStateVariableValues>
  </stateVariableInstance>
  <stateVariableInstance stateVariable="T286" >
    <listOfStateVariableValues>
      <stateVariableValue possibleStateVariableValue="phosp" />
    </listOfStateVariableValues>
  </stateVariableInstance>
</listOfStateVariableInstances>
<speciesInstanceInfluence>
```

```
          <math xmlns="http://www.w3.org/1998/Math/MathML">
            <apply>
              <times/>
              <ci>rPhosp</ci>
              <ci>Prel</ci>
            </apply>
          </math>
          <listOfParameters>
            <parameter id="Prel" value="0">
          </listOfParameters>
        </speciesInstanceInfluence>
      </reactingSpeciesInstance>
    </listOfReactingSpeciesInstances>
  </speciesReference>
</listOfReactants>
<listOfProducts>
  <speciesReference species="K_PSD">
    <listOfNascentSpeciesInstances>
      <nascentSpeciesInstance proportion="0.9">
        <listOfStateVariableInstances>
          <stateVariableInstance stateVariable="activ" >
            <listOfStateVariableValues>
              <stateVariableValue possibleStateVariableValue="inact" />
            </listOfStateVariableValues>
          </stateVariableInstance>
          <stateVariableInstance stateVariable="T286" >
            <listOfStateVariableValues>
              <stateVariableValue possibleStateVariableValue="phosp" />
            </listOfStateVariableValues>
          </stateVariableInstance>
        </listOfStateVariableInstances>
      </nascentSpeciesInstance>
      <nascentSpeciesInstance proportion="0.1">
        <listOfStateVariableInstances>
          <stateVariableInstance stateVariable="activ" >
            <listOfStateVariableValues>
              <stateVariableValue possibleStateVariableValue="inact" />
            </listOfStateVariableValues>
          </stateVariableInstance>
          <stateVariableInstance stateVariable="T286" >
            <listOfStateVariableValues>
              <stateVariableValue possibleStateVariableValue="phosp" />
            </listOfStateVariableValues>
          </stateVariableInstance>
        </listOfStateVariableInstances>
      </nascentSpeciesInstance>
    </listOfNascentSpeciesInstances>
  </speciesReference>
</listOfProducts>
```

We should actually write that CaMKII cannot be phosphorylated when it inactive *or* phosphorylated. That would require the creation of two reactingSpeciesInstances, with one specified stateVariableInstance each, rather than one reactingSpeciesInstance with two specified stateVariableInstances.

A state variable not mentioned in the nascentSpeciesInstance stays unchanged accross the reaction.

6 Example of use in a multi-component model

The following describes the phosphorylation of a monomer of CaMKII by the neighbouring monomer. To describe the dimer we use the structures described in Andrew Finney's multi-component proposal Finney (2004).

First we need to create a dimer of CaMKII subunits. We do not need it for the phosphorlation itself, and the complex could emerge from a dimerisation reaction. However, the complex will carry its own state variable that will affect the reaction. This needs to be declared.

```
<listOfspeciesTypes>
  <speciesType id="CaMKII" name="Calcium/Calmodulin Kinase II">
    <listOfStateVariables>
      <stateVariable id="activ" name="activity">
```

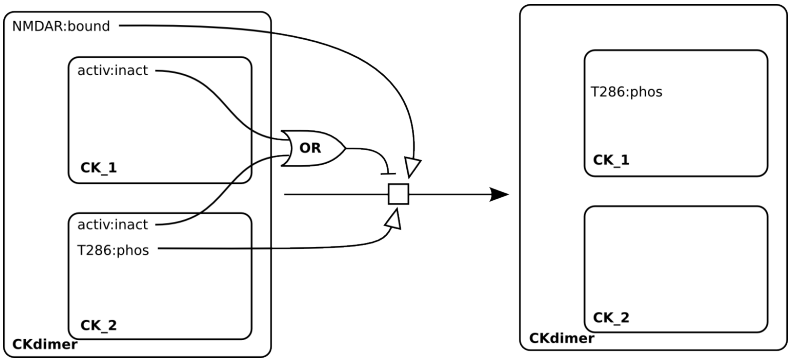
```
        <listOfPossibleStateVariableValues>
          <possibleStateVariableValue id="inact" name="inactive" />
          <possibleStateVariableValue id="act"   name="active"   />
          <possibleStateVariableValue id="inhib" name="inhibited"/>
        </listOfPossibleStateVariableValues>
      </stateVariable>
      <stateVariable id="T286" name="threonine 286">
        <listOfPossibleStateVariableValues>
          <possibleStateVariableValue id="unphos" name="non-phosphorylated"/>
          <possibleStateVariableValue id="phos"   name="phosphorylated"    />
        </listOfPossibleStateVariableValues>
      </stateVariable>
    </listOfStateVariables>
  </speciesType>
  <speciesType id="CKdimer" name="Calcium/Calmodulin Kinase II dimer">
    <listOfStateVariables>
      <stateVariable id="NMDAR" name="binding to NMDA receptor">
        <listOfPossibleStateVariableValues>
          <possibleStateVariableValue id="bound"   />
          <possibleStateVariableValue id="unbound" />
        </listOfPossibleStateVariableValues>
      </stateVariable>
    </listOfStateVariables>
    <listOfSpeciesTypeInstances>
      <speciesTypeInstance id="iCK_1" speciesType="CaMKII" />
      <speciesTypeInstance id="iCK_2" speciesType="CaMKII" />
    </listOfSpeciesTypeInstances>
  </speciesType>
</listOfSpeciesType>
```

The declaration of the `species` in the example page 4 remains the same, except the following addition, that says 20% of the CaMKII dimers are bound to the NMDA receptor:

```
<species id="K2_PSD" name="CaMKII dimer in postsynaptic density"
         speciesType="CKdimer" initialAmount="250" compartment="PSD">
  <listOfInitialSpeciesInstances>
    <initialSpeciesInstance initialProportion="0.2" >
      <listOfStateVariableInstances>
        <stateVariableInstance stateVariable="NMDAR" >
          <listOfStateVariableValues>
            <stateVariableValue possibleStateVariableValue="bound" />
          </listOfStateVariableValues>
        </stateVariableInstance>
      </listOfStateVariableInstances>
    </initialSpeciesInstance>
    <initialSpeciesInstance initialProportion="0.8" >
      <listOfStateVariableInstances>
        <stateVariableInstance stateVariable="NMDAR" >
          <listOfStateVariableValues>
            <stateVariableValue possibleStateVariableValue="unbound" />
          </listOfStateVariableValues>
        </stateVariableInstance>
      </listOfStateVariableInstances>
    </initialSpeciesInstance>
  </listOfInitialSpeciesInstances>
</species>
```

Because there are 500 CaMKII molecules, and 250 dimers declared, we see that all the CaMKII exist as dimers. Two approaches can be used to describe the effect of state variables on the reaction. The first one is to describe the tree of component's state variables in the reactingSpeciesInstances and nascentSpeciesInstances. Another approach, favoured in this example is to "flattened" the graph, and list the state variables, optionally identifying them using the id of the relevant speciesTypeInstance (note that the additional attribute `speciesTypeInstance` is not described in the UML above).

The following example shows the effect of various state variables on the velocity of the phosphorylation of one CaMKII monomer by the adjacent one. Binding of the dimer to NMDA receptor increases the velocity. Inactivity of any of the monomers suppress the reaction. Since the attribute `speciesTypeInstance` is not set, the rule concern *all* the state variables "activ", of any of the components. Finally, the phosphorylation of one monomer favours the phosphorylation of the neighbouring one:




```
<listOfReactants>
    <listOfReactingSpeciesInstances>
      <reactingSpeciesInstance>
        <listOfStateVariableInstances>
          <stateVariableInstance stateVariable="NMDAR" >
            <listOfStateVariableValues>
              <stateVariableValue possibleStateVariableValue="bound" />
            </listOfStateVariableValues>
          </stateVariableInstance>
        </listOfStateVariableInstances>
        <speciesInstanceInfluence>
          <math xmlns="http://www.w3.org/1998/Math/MathML">
            <apply>
              <times/>
              <ci>rPhosp</ci>
              <ci>Pnr</ci>
            </apply>
          </math>
          <listOfParameters>
            <parameter id="Pnr" value="10">
          </listOfParameters>
        </speciesInstanceInfluence>
      </reactingSpeciesInstance>
      <reactingSpeciesInstance>
        <listOfStateVariableInstances>
          <stateVariableInstance stateVariable="activ" >
            <listOfStateVariableValues>
              <stateVariableValue possibleStateVariableValue="inact" />
            </listOfStateVariableValues>
          </stateVariableInstance>
        </listOfStateVariableInstances>
        <speciesInstanceInfluence>
          <math xmlns="http://www.w3.org/1998/Math/MathML">
            <apply>
              <times/>
              <ci>rPhosp</ci>
              <ci>Prel</ci>
            </apply>
          </math>
          <listOfParameters>
            <parameter id="Prel" value="0">
          </listOfParameters>
        </speciesInstanceInfluence>
      </reactingSpeciesInstance>
      <reactingSpeciesInstance>
        <listOfStateVariableInstances>
          <stateVariableInstance speciesTypeInstance="iCK_2" stateVariable="T286" >
            <listOfStateVariableValues>
              <stateVariableValue possibleStateVariableValue="phospho" />
            </listOfStateVariableValues>
          </stateVariableInstance>
        </listOfStateVariableInstances>
        <speciesInstanceInfluence>
          <math xmlns="http://www.w3.org/1998/Math/MathML">
            <apply>
              <times/>
```

```
            <ci>rPhosp</ci>
            <ci>P286</ci>
          </apply>
        </math>
        <listOfParameters>
          <parameter id="P286" value="100">
        </listOfParameters>
      </speciesInstanceInfluence>
    </reactingSpeciesInstance>
  </listOfReactingSpeciesInstances>
</speciesReference>
</listOfReactants>
<listOfProducts>
  <speciesReference species="K2_PSD">
    <listOfNascentSpeciesInstances>
      <nascentSpeciesInstance proportion="1">
        <listOfStateVariableInstances>
          <stateVariableInstance stateVariable="activ" >
            <listOfStateVariableValues>
              <stateVariableValue possibleStateVariableValue="inact" />
            </listOfStateVariableValues>
          </stateVariableInstance>
          <stateVariableInstance speciesTypeInstance="iCK_1" stateVariable="T286" >
            <listOfStateVariableValues>
              <stateVariableValue possibleStateVariableValue="phosp" />
            </listOfStateVariableValues>
          </stateVariableInstance>
        </listOfStateVariableInstances>
      </nascentSpeciesInstance>
    </listOfNascentSpeciesInstances>
  </speciesReference>
</listOfProducts>
```

Note this reaction describes the phosphorylation of any of the two monomers. Indeed, while the dimer is declared as a speciesReference, the monomers are declared as speciesTypeInstance. CK_1 and CK_2 are interchangeable. Therefore, this single reaction declaration replaces all the following:

```
( unbound, (inact, unphos), (inact, unphos) ) -> ( unbound, (inact, unphos), (inact, phos) )
( unbound, (inact, unphos), (act, unphos) )   -> ( unbound, (inact, unphos), (act, phos) )
( unbound, (act, unphos), (inact, unphos) )   -> ( unbound, (act, unphos), (inact, phos) )
( unbound, (act, unphos), (act, unphos) )     -> ( unbound, (act, unphos), (act, phos) )
( unbound, (inact, phos), (inact, unphos) )   -> ( unbound, (inact, phos), (inact, phos) )
( unbound, (inact, phos), (act, unphos) )     -> ( unbound, (inact, phos), (act, phos) )
( unbound, (act, phos), (inact, unphos) )     -> ( unbound, (act, phos), (inact, phos) )
( unbound, (act, phos), (act, unphos) )       -> ( unbound, (act, phos), (act, phos) )
( unbound, (inact, unphos), (inact, unphos) ) -> ( unbound, (inact, phos), (inact, unphos) )
( unbound, (inact, unphos), (act, unphos) )   -> ( unbound, (inact, phos), (act, unphos) )
( unbound, (act, unphos), (inact, unphos) )   -> ( unbound, (act, phos), (inact, unphos) )
( unbound, (act, unphos), (act, unphos) )     -> ( unbound, (act, phos), (act, unphos) )
( unbound, (inact, unphos), (inact, phos) )   -> ( unbound, (inact, phos), (inact, phos) )
( unbound, (inact, unphos), (act, phos) )     -> ( unbound, (inact, phos), (act, phos) )
( unbound, (act, unphos), (inact, phos) )     -> ( unbound, (act, phos), (inact, phos) )
( unbound, (act, unphos), (act, phos) )       -> ( unbound, (act, phos), (act, phos) )
( bound, (inact, unphos), (inact, unphos) )   -> ( bound, (inact, unphos), (inact, phos) )
( bound, (inact, unphos), (act, unphos) )     -> ( bound, (inact, unphos), (act, phos) )
( bound, (act, unphos), (inact, unphos) )     -> ( bound, (act, unphos), (inact, phos) )
( bound, (act, unphos), (act, unphos) )       -> ( bound, (act, unphos), (act, phos) )
( bound, (inact, phos), (inact, unphos) )     -> ( bound, (inact, phos), (inact, phos) )
( bound, (inact, phos), (act, unphos) )       -> ( bound, (inact, phos), (act, phos) )
( bound, (act, phos), (inact, unphos) )       -> ( bound, (act, phos), (inact, phos) )
( bound, (act, phos), (act, unphos) )         -> ( bound, (act, phos), (act, phos) )
( bound, (inact, unphos), (inact, unphos) )   -> ( bound, (inact, phos), (inact, unphos) )
( bound, (inact, unphos), (act, unphos) )     -> ( bound, (inact, phos), (act, unphos) )
( bound, (act, unphos), (inact, unphos) )     -> ( bound, (act, phos), (inact, unphos) )
( bound, (act, unphos), (act, unphos) )       -> ( bound, (act, phos), (act, unphos) )
( bound, (inact, unphos), (inact, phos) )     -> ( bound, (inact, phos), (inact, phos) )
( bound, (inact, unphos), (act, phos) )       -> ( bound, (inact, phos), (act, phos) )
( bound, (act, unphos), (inact, phos) )       -> ( bound, (act, phos), (inact, phos) )
( bound, (act, unphos), (act, phos) )         -> ( bound, (act, phos), (act, phos) )
```

References

Finney, A. (2004). Systems biology markup language (sbml) level 3 proposal: Multi-component species features. Technical report. Internal Discussion Document.

Hucka, M., Finney, A., Hoop, S., Keating, S., and Novère, L. (2007). Systems biology markup language (sbml) level 2: Structures and facilities for basic model definitions. Technical report. available via the World Wide Web `http://sbml.org/`.

Le Novère, N. and Shimizu, T. S. (2001). StochSim: Modelling of stochastic biomolecular processes. *17*, 17:575–576.

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/`.

Tolle, D. and Le Novère, N. (2006). Particle-based stochastic simulation in systems biology. *Current Bioinformatics*, 1:315–320.