# Systems Biology Markup Language (SBML) Level 3 Proposal: Multi-component Species Features

Andrew Finney
afinney@cds.caltech.edu

March 31, 2004

## Contents

# 1 Terms of Reference

This document describes proposed features for inclusion in Systems Biology Markup Language (SBML) Level 3 that will enable the description of large chemical entities that are composed from other chemical entities. (Entities of this type were previously classed as 'complex species'.)

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.

This document is not the first proposal to support multi-component species (Le Novère et al., 2003) and supersedes a previous proposal by the author (Finney, 2001).

Throughout this document issues that the author believes 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. In addition for brevity the UML diagrams in this proposal show only new attributes and types for SBML Level 3.

All types proposed in this document will be derived from the `SBase` type.

# 2 Acknowledgements

This proposal has benefitted from discussions the author had with Martin Ginkel, Jörg Stelling, Nicolas Le Novere, Fabian Campagne, Jeremy Zucker, Robert Phair, Larry Lok, Michael Blinov and Roger Brent. In particular many of the ideas presented here are similar to those developed by the Molecular Sciences Institute and the T-10 Cell Signalling Group (Goldstein et al., 2001) at Los Alamos National Laboratories.

Support for the development of this proposal came from the following the National Human Genome Research Institute (USA) and the Japan Science and Technology Corporation under the ERATO Kitano Symbiotic Systems Project. Additional support was provided by the California Institute of Technology (USA), the University of Hertfordshire (UK) and the Molecular Sciences Institute (USA).

I would also like to thank Sarah Keating for proofreading and commenting on this document.

# 3 Aims

This proposal aims to support the representation of the following concepts that are not easily represented in SBML Level 2:

- the common description of biochemical entities that can then be located in different compartments
- the common description of biochemical reactions that can then be located in different compartments
- the hierarchical description of biochemical entities through the composition of other biochemical entities
- the description of biochemical entities through simple associative composition
- the description of biochemical entities through graphs of other biochemical entities where arcs represent kinds of bonding
- the description of generalized biochemical reactions that avoids the enumeration of many species states and reactions

In particular this proposal aims to enable the description of, for example, proteins which can contain many phosphorylation states, complexes of these proteins and models of signalling pathways which contain these proteins.

# 4 Overview of Proposal

Figures 1 and 2 show a UML diagram for the prosed new classes. Section 5 demonstrates, how instances of these classes can be assembled to achieve the aims of the proposal. The examples in Section 5 effectively define a staged road map of how the features described in this proposal could be added to SBML. The proposal is described more formally in section 6. Section 6; which can be considered as a reference section.

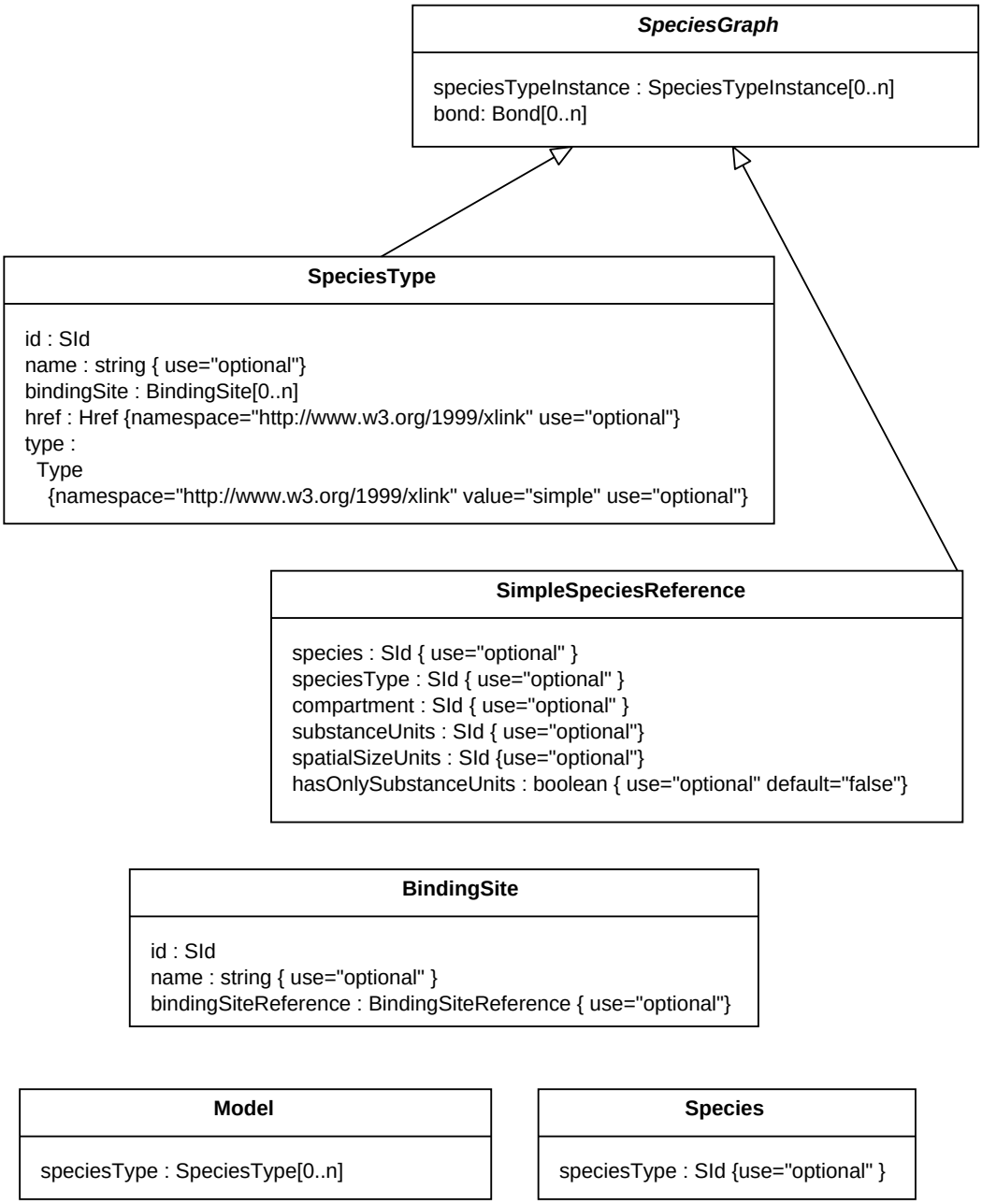


**Figure 1:** *The types and attributes introduced into SBML by this proposal. This diagram only shows new classes and fields: all SBML Level 2 structures are assumed to be present. This diagram is continued in Figure 2*

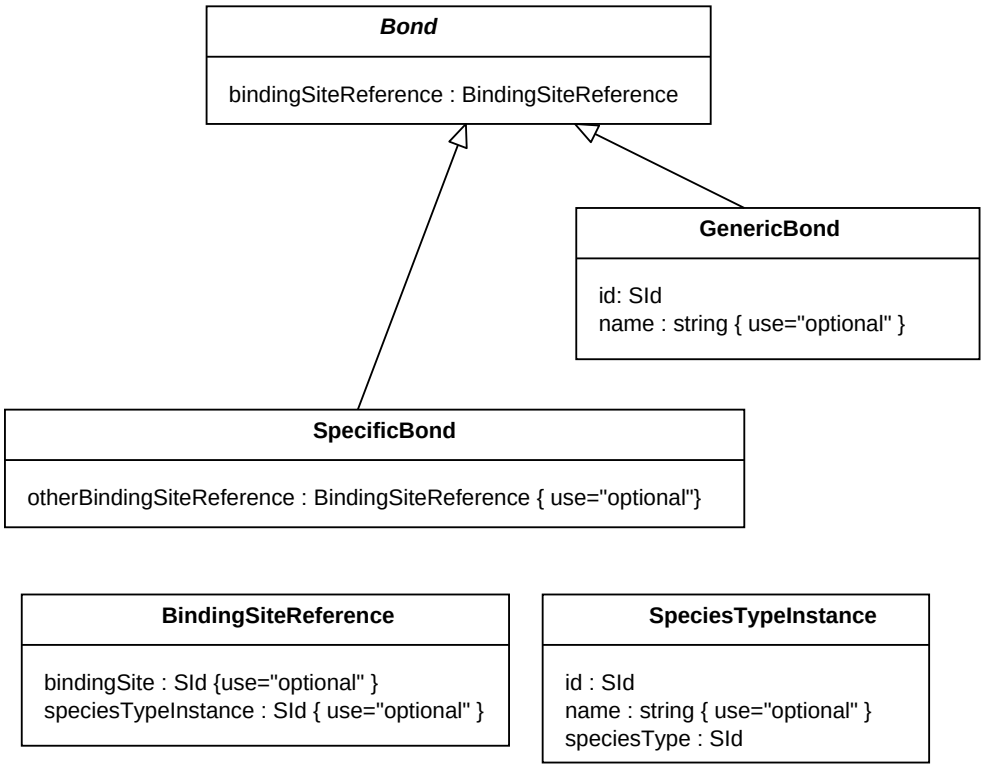


**Figure 2:** *The types and attributes introduced into SBML by this proposal. This diagram only shows new classes and fields: all SBML Level 2 structures are assumed to be present. This diagram is a continuation of the diagram in Figure 1*

# 5 Tutorial on the Proposed Features

## 5.1 Terminology

The following terminology is used in this document:

- *chemical entity* any individual chemical object e.g. a calcium ion, a phosphate, a protein, or a lipid.
- *compartment* a well stirred container of chemical entities
- *species type* a type of chemical entity, specifically a set of chemical entities with exactly the same chemical form,
- *species* a pool of chemical entities of the same species type located in a specific compartment

## 5.2 The Definition of Chemical Entities across Compartments

Consider a model where we have a species type which exists in more than one compartment. For example we might wish to model Aspartate in a Cytosol compartment and in the Mitochondrial Matrix. In SBML Level 2 we have to explicitly define each pool of Aspartate located in a separate compartment, using a `Species` structure as shown in Figure 3

```
<model id="malate_aspartate_shuttle1">
    <listOfCompartments>
        <compartment id="Cytosol"/>
        <compartment id="Mitochondrial_Matrix"/>
    </listOfCompartments>
    <listOfSpecies>
        <species id="Aspartate_in_Cytosol" compartment="Cytosol"/>
        <species id="Aspartate_in_Mitochondrial_Matrix" compartment="Mitochondrial_Matrix"/>
    </listOfSpecies>
</model>
```

**Figure 3:** *`malate_aspartate_shuttle1` Model with the same type of chemical entity located in different compartments.*

In SBML Level 2 there is no formal way to relate these species together. Under this proposal we can do this by representing a chemical entity type such as, Aspartate, with a `SpeciesType` structure and then refer to the `SpeciesType` from the `Species` structures. We can thus transform the `malate_aspartate_shuttle1` Model in Figure 3 to that shown in Figure 4.

This model does not introduce any new variables that are not present in `malate_aspartate_shuttle1` it simply identifies `Aspartate_in_Cytosol` and `Aspartate_in_Mitochondrial_Matrix` as being separate pools of the same chemical entity.

The `malate_aspartate_shuttle1` example is still a valid model under this proposal. For backwards compatibility the `speciesType` attribute on `Species` is not mandatory.

It is not possible to locate a `SpeciesType` in a `Compartment` more than once, for example, it is not possible for two `Species` structures to have the same `speciesType` and `compartment` values.

You cannot refer to `SpeciesType` structures from MathML structures under this proposal.

## 5.3 Generalized Reactions: The Definition of Reactions across Compartments

Just as we might wish to give a common identifier to chemical entities distributed across several compartments we might wish to have some common object describing reactions between those chemical entities that is independent of the compartments in which the reactions occur.

For example consider the representation of the transamination reaction, a reversible reaction that converts Aspartate to Oxaloacetate in both the Cytosol and Mitochondrial Matrix.

```
<model id="malate_aspartate_shuttle2">
    <listOfCompartments>
        <compartment id="Cytosol"/>
        <compartment id="Mitochondrial_Matrix"/>
    </listOfCompartments>
    <listOfSpeciesTypes>
        <speciesType id="Aspartate"/>
    </listOfSpeciesTypes>
    <listOfSpecies>
        <species
            id="Aspartate_in_Cytosol"
            speciesType="Aspartate"
            compartment="Cytosol"/>
        <species
            id="Aspartate_in_Mitochondrial_Matrix"
            speciesType="Aspartate"
            compartment="Mitochondrial_Matrix"/>
    </listOfSpecies>
</model>
```

**Figure 4:** *malate_aspartate_shuttle2 Model which uses a SpeciesType to link species of the same type of chemical entity that are located in different compartments.*

We could extend `malate_aspartate_shuttle2` using `SpeciesType` structures combined with other SBML Level 2 structures as shown in Figure 5 on page 7. Under this proposal we can replace the 2 reactions in Figure 5 with a single reaction structure as shown in Figure 6 on page 8.

All the `SimpleSpeciesReference` structures (that is modifiers, reactants and products) refer to species in the same compartment. This means that, under this proposal, it is not possible to define a transport reaction, that is a reaction which moves chemical entities between compartments, using this form.

### 5.3.1 Defining the explicit location of a `SimpleSpeciesReference`

The location of a species pool can be made explicit in a `SimpleSpeciesReference` structure without referring to a `Species` structure. This can be achieved by using the proposed optional `compartment` field which refers to a `Compartment` structure to indicate the location of the given `SpeciesType`. For example consider the transport reaction shown in Figure 7 on page 8 which can be added to the model in Figure 6 on page 8.

All the `SimpleSpeciesReference` structures of a reaction should simultaneously either (a) be located (i.e. have values for the `species` or `compartment` attributes); or (b) apply to any compartment (i.e. not have values for the `species` and `compartment` attributes). *This restriction is not essential but simplifies the interpretation of the proposed format.*

*This feature could be introduced later in the SBML development road map. It is however an essential component of features introduced later.*

### 5.3.2 Defining Kinetic Laws for Generalized Reactions

As defined in the examples above it is not possible to compose the kinetic law of these generalized reactions since there is no symbol that refers to either the modifiers, reactants or products or the reaction species pools. However under this proposal the `id` field of a `SimpleSpeciesReference` becomes a symbol that can be used in the `KineticLaw` of the enclosing `Reaction`.

*Here I am assuming that the `id` field on `SimpleSpeciesReference` is introduced by a new version of SBML Level 2. This `id` field is in the global symbol namespace despite, for the purposes of this proposal, only having scope in the enclosing `Reaction`. If this is problematic then perhaps we could consider an additional attribute to declare the symbol.*

Figure 8 on page 9 shows the `Transamination` reaction, from model `malate_aspartate_shuttle4`, modified to include a rate law.

```
<model id="malate_aspartate_shuttle3">
    <listOfCompartments>
        <compartment id="Cytosol"/>
        <compartment id="Mitochondrial_Matrix"/>
    </listOfCompartments>
    <listOfSpeciesTypes>
        <speciesType id="Aspartate"/>
        <speciesType id="Oxaloacetate"/>
    </listOfSpeciesTypes>
    <listOfSpecies>
        <species
            id="Aspartate_in_Cytosol"
            speciesType="Aspartate"
            compartment="Cytosol"/>
        <species
            id="Aspartate_in_Mitochondrial_Matrix"
            speciesType="Aspartate"
            compartment="Mitochondrial_Matrix"/>
        <species
            id="Oxaloacetate_in_Cytosol"
            speciesType="Oxaloacetate"
            compartment="Cytosol"/>
        <species
            id="Oxaloacetate_in_Mitochondrial_Matrix"
            speciesType="Oxaloacetate"
            compartment="Mitochondrial_Matrix"/>
    </listOfSpecies>
    <listOfReactions>
        <reaction id="Transamination_in_Cytosol" reversible="true">
            <listOfReactants>
                <speciesReference species="Aspartate_in_Cytosol"/>
            </listOfReactants>
            <listOfProducts>
                <speciesReference species="Oxaloacetate_in_Cytosol"/>
            </listOfProducts>
        </reaction>
        <reaction id="Transamination_in_Mitochondrial_Matrix" reversible="true">
            <listOfReactants>
                <speciesReference species="Aspartate_in_Mitochondrial_Matrix"/>
            </listOfReactants>
            <listOfProducts>
                <speciesReference species="Oxaloacetate_in_Mitochondrial_Matrix"/>
            </listOfProducts>
        </reaction>
    </listOfReactions>
</model>
```

**Figure 5:** *The malate_aspartate_shuttle3 model which has duplicate reactions for each compartment.*

### 5.3.3 The Unit Attributes of `SimpleSpeciesReference`

To make the units of species explicit in kinetic laws under this proposal `SimpleSpeciesReference` structures have the attributes `substanceUnits`, `spatialSizeUnits` and `hasOnlySubstanceUnits`. These have the same semantics as the corresponding attributes on `Species`.

## 5.4 Species Implied from `SpeciesTypes`

Under this proposal `Species` structures are used to indicate the initial conditions and/or attributes of species and don't represent the complete set of species. In fact this proposal does not assume that an interpreter (e.g. simulator) of models in the proposed format would use species as it's fundamental representational form, for example an interpreter may represent individual chemical entities as distinct objects. In SBML Level 2 the model's `species` list is a complete enumeration of the pools of chemical entities. In this proposal this set of species is a subset of the complete set of species defined by the model. The remaining species are implied by `SpeciesType` structures, as described in this section, or are implied by reactions generalized to

```
<model id="malate_aspartate_shuttle4">
    <listOfCompartments>
        <compartment id="Cytosol"/>
        <compartment id="Mitochondrial_Matrix"/>
    </listOfCompartments>
    <listOfSpeciesTypes>
        <speciesType id="Aspartate"/>
        <speciesType id="Oxaloacetate"/>
    </listOfSpeciesTypes>
    <listOfSpecies>
        <species
            id="Aspartate_in_Cytosol"
            speciesType="Aspartate"
            compartment="Cytosol"/>
        <species
            id="Aspartate_in_Mitochondrial_Matrix"
            speciesType="Aspartate"
            compartment="Mitochondrial_Matrix"/>
        <species
            id="Oxaloacetate_in_Cytosol"
            speciesType="Oxaloacetate"
            compartment="Cytosol"/>
        <species
            id="Oxaloacetate_in_Mitochondrial_Matrix"
            speciesType="Oxaloacetate"
            compartment="Mitochondrial_Matrix"/>
    </listOfSpecies>
    <listOfReactions>
        <reaction id="Transamination" reversible="true">
            <listOfReactants>
                <speciesReference speciesType="Aspartate"/>
            </listOfReactants>
            <listOfProducts>
                <speciesReference speciesType="Oxaloacetate"/>
            </listOfProducts>
        </reaction>
    </listOfReactions>
</model>
```

**Figure 6:** *The `malate_aspartate_shuttle4` model which has a single reaction which is potentially located in all compartments.*

```
<reaction id="Malate_Transport" reversible="false">
    <listOfReactants>
        <speciesReference speciesType="Malate" compartment="Cytosol"/>
    </listOfReactants>
    <listOfProducts>
        <speciesReference speciesType="Malate" compartment="Mitochondrial_Matrix"/>
    </listOfProducts>
</reaction>
```

**Figure 7:** *The `Malate_Transport` model a transport reaction which refers to `SpeciesType` structures in specific compartments.*

cover classes of `SpeciesTypes`, see Section 5.8.

For each `SpeciesType` there exists a species of the given type in each compartment in the model unless there already exists an equivalent `Species` structure located in that compartment. These implied species always have an initial concentration or substance amount of zero and are never constant nor boundary conditions. This means that constant or boundary condition species or species with any initial concentration must be made explicit using a `Species` structure.

So if we consider the model `malate_aspartate_shuttle4` on page 8 we can omit any species structures which we wish to model as having an initial concentration of zero. This is shown in Figure 9 where we assume that only `Aspartate` species have an initial concentration.

```
<reaction id="Transamination" reversible="true">
    <listOfReactants>
        <speciesReference id="S1" speciesType="Aspartate"/>
    </listOfReactants>
    <listOfProducts>
        <speciesReference speciesType="Oxaloacetate"/>
    </listOfProducts>
    <kineticLaw>
        <math xmlns="http://www.w3.org/1998/MathMathML">
            <apply>
                <times/>
                <cn>1.1</cn>
                <ci>S1</ci>
            </apply>
        </math>
    </kineticLaw>
</reaction>
```

**Figure 8:** *The `Transamination` reaction from Figure 6 modified to include a kinetic law.*

```
<model id="malate_aspartate_shuttle5">
    <listOfCompartments>
        <compartment id="Cytosol"/>
        <compartment id="Mitochondrial_Matrix"/>
    </listOfCompartments>
    <listOfSpeciesTypes>
        <speciesType id="Aspartate"/>
        <speciesType id="Oxaloacetate"/>
    </listOfSpeciesTypes>
    <listOfSpecies>
        <species
            id="Aspartate_in_Cytosol"
            speciesType="Aspartate"
            compartment="Cytosol"
            initialConcentration="1"/>
        <species
            id="Aspartate_in_Mitochondrial_Matrix"
            speciesType="Aspartate"
            compartment="Mitochondrial_Matrix"
            initialConcentration="1"/>
    </listOfSpecies>
    <listOfReactions>
        <reaction id="Transamination" reversible="true">
            <listOfReactants>
                <speciesReference speciesType="Aspartate"/>
            </listOfReactants>
            <listOfProducts>
                <speciesReference speciesType="Oxaloacetate"/>
            </listOfProducts>
        </reaction>
    </listOfReactions>
</model>
```

**Figure 9:** *The `malate_aspartate_shuttle5` model with a reduced set of `Species` structures.*

*The concept of implied species could be introduced later in the SBML development road map. It is however an essential component of features introduced later.*

## 5.5 Simple Multi-Component Chemical Entities

In this proposal `SpeciesType` structures can be composed from instances of other `SpeciesType` structures. These instances are encoded as `SpeciesTypeInstance` structures. For example see the `Pheromone_Response` model, shown with XML and diagram form in Figure 10.

This indicates that `SteComplex` is a complex made up of one instance each of the proteins `Ste5`, `Ste11`,

```
<model "Pheromone_response">
    <listOfSpeciesTypes>
        <speciesType id="Ste5"/>
        <speciesType id="Ste11"/>
        <speciesType id="Ste7"/>
        <speciesType id="Fus3"/>
        <speciesType id="SteComplex">
            <listOfSpeciesTypeInstances>
                <speciesTypeInstance id="iSte5" speciesType="Ste5"/>
                <speciesTypeInstance id="iSte11" speciesType="Ste11"/>
                <speciesTypeInstance id="iSte7" speciesType="Ste7"/>
                <speciesTypeInstance id="iFus3" speciesType="Fus3"/>
            </listOfSpeciesTypeInstances>
        </speciesType>
    </listOfSpeciesTypes>
</model>
```

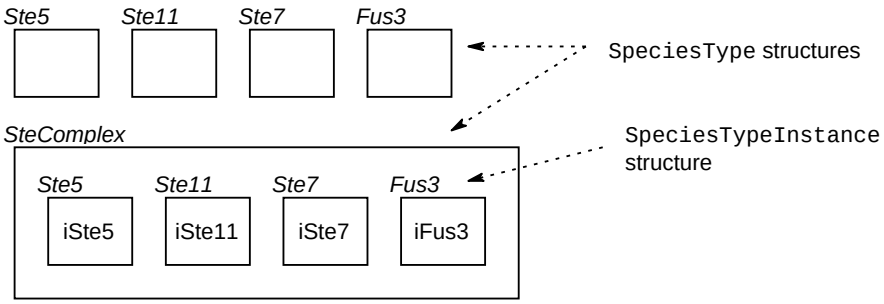


**Figure 10:** *The `Pheromone_response` model*

`Ste7` and `Fus3`. The individual instances are always identified to enable the components of homodimers to be separately identified. In the diagram a rectangle not enclosed within another rectangle represents a `SpeciesType` structure. A rectangle with a normal typeface label represents a `SpeciesTypeInstance` structure and is enclosed within another rectangle representing a `SpeciesType`.

We can also describe reactions using this form on `SpeciesReference` structures. For example we can describe the binding of `Ste11` to `Ste5` with the `Reaction` structure shown in Figure 11. In this diagram the rectangle without italic along the top edge represent a `SimpleSpeciesReference` structure which contains one or more `SpeciesTypeInstance` structures.

Although the identity of `SpeciesTypeInstance` structures is declared in each `SimpleSpeciesReference` structure these identities have scope throughout a reaction. The `SpeciesTypeInstance id` fields with the same value in the same reaction refer to the same chemical entity. By giving `SpeciesTypeInstance id` fields the same values in the reactants and products of a reaction we indicate that the entity is only modified by the reaction rather than being created or destroyed by the reaction. For example the reaction `binding_Ste5_Ste11` could be encoded as shown in Figure 12.

This encoding indicates that, for the purposes of the model, `Ste5` and `Ste11` are not modified when they bond. The distinction between `binding_Ste5_Ste11` and `binding_Ste5_Ste11_v2` is only descriptive however the latter form is used as the basis for more complex semantics later. A model like `pheromone_response` with or without characterized reactions including kinetic laws doesn't encapsulate a model that can be simulated because it does not specify any species of non-zero concentration.

```
<reaction id="binding_Ste5_Ste11">
    <listOfReactants>
        <speciesReference speciesType="Ste11"/>
        <speciesReference speciesType="Ste5"/>
    </listOfReactants>
    <listOfProducts>
        <speciesReference>
            <listOfSpeciesTypeInstances>
                <speciesTypeInstance id="iSte5" speciesType="Ste5"/>
                <speciesTypeInstance id="iSte11" speciesType="Ste11"/>
            </listOfSpeciesTypeInstances>
        </speciesReference>
    </listOfProducts>
</reaction>
```

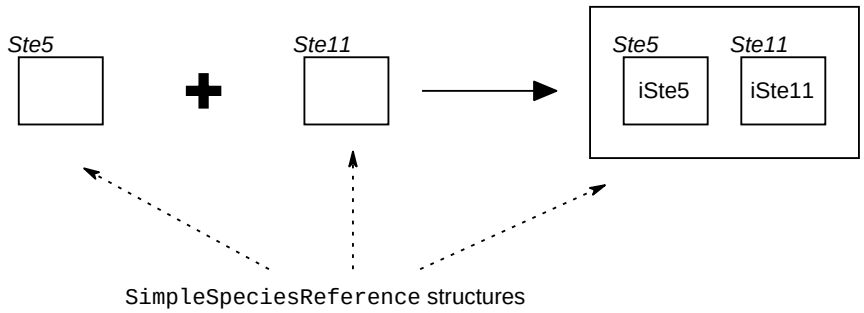


**Figure 11:** *The binding_Ste5_Ste11 reaction, operating in the context of the species types defined in Figure 10. This indicates that the reaction binding_Ste5_Ste11 creates a complex consisting of Ste5 and Ste11 entities from unbound Ste5 and Ste11 entities.*

```
<reaction id="binding_Ste5_Ste11_v2">
    <listOfReactants>
        <speciesReference>
            <listOfSpeciesTypeInstances>
                <speciesTypeInstance id="iSte5" speciesType="Ste5"/>
            </listOfSpeciesTypeInstances>
        </speciesReference>
        <speciesReference>
            <listOfSpeciesTypeInstances>
                <speciesTypeInstance id="iSte11" speciesType="Ste11"/>
            </listOfSpeciesTypeInstances>
        </speciesReference>
    </listOfReactants>
    <listOfProducts>
        <speciesReference>
            <listOfSpeciesTypeInstances>
                <speciesTypeInstance id="iSte5" speciesType="Ste5"/>
                <speciesTypeInstance id="iSte11" speciesType="Ste11"/>
            </listOfSpeciesTypeInstances>
        </speciesReference>
    </listOfProducts>
</reaction>
```

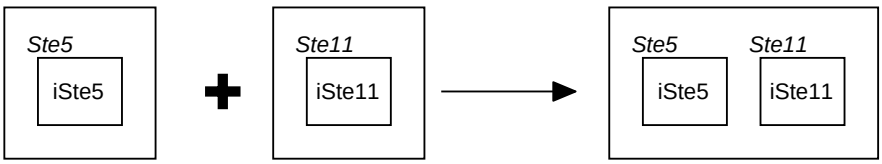


**Figure 12:** *The binding_Ste5_Ste11_v2 reaction*

## 5.6 Multi-component Chemical Entities with explicit bonds

The forms described in section 5.5 capture some but not all the relevant knowledge of chemical entities that we might wish to model. In this section I describe how chemical bond information is captured. The bond information on a `SimpleSpeciesReference` or a `SpeciesType` is a graph linking the `SpeciesTypeInstance` structures together. The description of a structure using chemical bonds requires the identification of binding sites on `SpeciesType` structures, using `BindingSite` structures, and then the enumeration of the bonds on those binding sites using `SpecificBond` structures. `SpecificBond` structures consist of pairs of `BindingSiteReference` structures. We can redefine the model `pheromone_response` along these lines as shown in Figure 13 on page 13 with a corresponding diagram in Figure 14 on page 14. In the diagram the lines perpendicular to the outer edge of `SpeciesType` structures represent `BindingSite` structures. `SpecifcBond` structures are represented as lines joining `SpeciesTypeInstance` structures.

`SpecificBond` structures can be used to indicate unbound binding sites, simply by containing a single `BindingSiteReference` structure, as shown the reaction in Figure 15 on page 15. In this diagram unbound binding sites are represented as disconnected lines perpendicular to the edge of a `SpeciesTypeInstance`. A `SpeciesType` structure must not leave the state of a binding site undefined or ambiguous. Section 5.8 describes how a `SimpleSpeciesReference` can refer to several different `SpeciesType` structures where the class represents a range of states for one or more binding sites.

The level of decomposition of a biochemical system into chemical entities and their binding sites and bonds is not defined by this proposal. This proposal is designed to support arbitrary decomposition schemes which capture knowledge at different resolutions in the same model. The underlying chemistry represented by a given binding site state is also not defined by this proposal.

An underlying principle of this proposal is that the binding representation described in this section can be used to represent the reversible covalent modification of proteins including, for example, phosphorylation and dephosphorylation. The example model shown in Figure 16 on page 16 represents the phosphorylation of `Ste11` by `Ste20`. A diagram of this model is shown in Figure 17 on page 17. This digram is divided into two parts by a horizontal lines. The objects above the line represent the `SpeciesTypes` in the model and the rest of the diagram represents the reaction in the model. The object parallel to the reaction arrow represents the reaction's modifier.

This model deliberately does not model the involvement of ATP or ADP molecules demonstrating how the level of detail of the biological knowledge captured by the proposed standard is arbitrary. As a result not all the instances of species types in the list of reactants are present in the list of products. This is valid in this proposal: the structural details of chemical entity transformation do not have to be fully elucidated. In fact the reaction shown in Figure 18 on page 17 is valid even if it is implausible from a biochemical perspective.

`SpeciesGraph` structures, that is `SpeciesType` and `SimpleSpeciesReference` structures, can contain a number of disconnected components (as described previously in Section 5.5). This means that a list of `Bond` structures in a `SpeciesGraph` does not have to comprise a connected graph. In this case the `SpeciesGraph` still represents a single entity where the complete set of bonds is not specified. As an example the consider the model shown in Figure 19 on page 18. A diagram of this model is shown in Figure 20 on page 19. Note that this example could have been encoded so that the product of the reaction is simple a reference to `SpeciesType D`. The form used, although redundant, demonstrates that the `SimpleSpeciesReference` structures of a reaction can contain disconnected components.

## 5.7 Importing `SpeciesType` structures into a model

This proposal allows for the incorporation of `SpeciesType` structures into a model which are encoded outside the model. This is achieved by using `SpeciesType` structures that import a species type definition rather than creating a definition. This is achieved by using XLink attributes (DeRose et al., 2001). An example is shown in Figure 21 on Page 20.

```
<model "pheromone_response_v2">
    <listOfSpeciesTypes>
        <speciesType id="Ste5">
            <listOfBindingSites>
                <bindingSite id="r241">
                <bindingSite id="r463">
                <bindingSite id="r744">
            </listOfBindingSites>
        </speciesType>
        <speciesType id="Ste11"/>
            <listOfBindingSites>
                <bindingSite id="site">
            </listOfBindingSites>
        </speciesType>
        <speciesType id="Ste7"/>
            <listOfBindingSites>
                <bindingSite id="site">
            </listOfBindingSites>
        </speciesType>
        <speciesType id="Fus3"/>
            <listOfBindingSites>
                <bindingSite id="site">
            </listOfBindingSites>
        </speciesType>
        <speciesType id="SteComplex">
            <listOfSpeciesTypeInstances>
                <speciesTypeInstance id="iSte5" speciesType="Ste5"/>
                <speciesTypeInstance id="iSte11" speciesType="Ste11"/>
                <speciesTypeInstance id="iSte7" speciesType="Ste7"/>
                <speciesTypeInstance id="iFus3" speciesType="Fus3"/>
            </listOfSpeciesTypeInstances>
            <listOfBonds>
                <specificBond>
                    <bindingSiteReference speciesTypeInstance="iSte5" bindingSite="r241"/>
                    <otherBindingSiteReference
                        speciesTypeInstance="iFus3" bindingSite="site"/>
                </specificBond>
                <specificBond>
                    <bindingSiteReference speciesTypeInstance="iSte5" bindingSite="r463"/>
                    <otherBindingSiteReference
                        speciesTypeInstance="iSte11" bindingSite="site"/>
                </specificBond>
                <specificBond>
                    <bindingSiteReference speciesTypeInstance="iSte5" bindingSite="r744"/>
                    <otherBindingSiteReference
                        speciesTypeInstance="iSte7" bindingSite="site"/>
                </specificBond>
            </listOfBonds>
        </speciesType>
    </listOfSpeciesTypes>
</model>
```

**Figure 13:** *The pheromone_response_v2 model which demonstrates the use of BindingSite and SpecificBond structures.*

## 5.8 Reactions generalized to cover classes of Multi-component Chemical Entities

In this section a further method of representing generalized reactions is described. Through this representation scheme the set of species and species types is implied rather than having to be fully enumerated. Under this proposal the bonding concept is extended in reactions so that it is possible for a reactant, product or modifier to refer to a set of closely related species that have a similar but not identical chemical structure. This is achieved the use of `GenericBond` structures within `SimpleSpeciesReference` structures.

`GenericBond` is a alternative `Bond` type. Reactions containing a `GenericBond` can potentially apply to a large set of complex species types including those not explicitly defined in the model thus reducing the number of reactions, complex species types and species that need to be enumerated in a given system. So

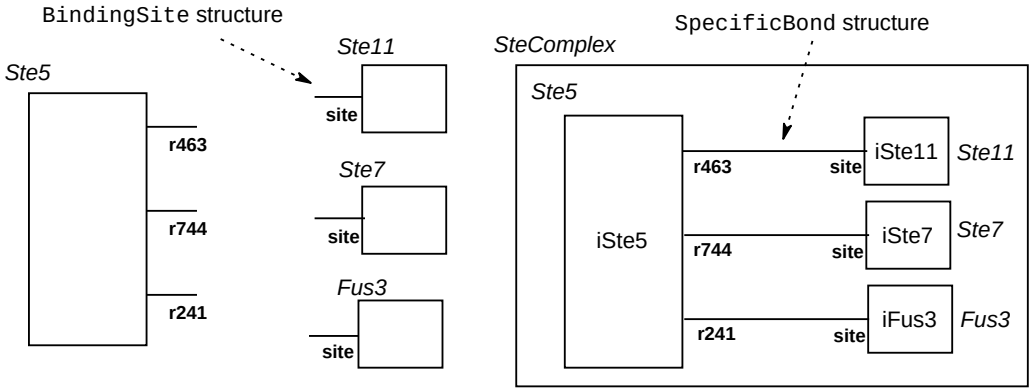


**Figure 14:** *The Diagram of the* `pheromone_response_v2 model`

just as the complete set of species that the reaction set operates on does not need to be enumerated nor does the complete set of species types need to be enumerated.

When a reaction is applied to the given state of one or more reactants, a `GenericBond` structure in the reaction is assigned the state of a binding site. The assignment can be to an empty entity if the match is to an empty binding site or to a unspecified binding site on an unspecified chemical entity. Whatever is assigned to the `GenericBond` in the set of reactants is transferred to the set of products.

The simple abstract example model shown in Figure 22 on page 21 uses this generalization mechanism redundantly. A diagram of this model is shown in Figure 23 on page 22. In this diagram the `GenericBond` structures are represented with large bold italic text labels adjacent to their associated `BindingSite`. The reaction `generic` defines how entities `A` and `B` bind together without changing the state of one of the binding sites on `A`.

A more concrete example model is shown in fragments in Figures 24 to 26 on pages 22 to 24. The reactions in Figures 25 and 26 (which operate on the species types encoded in Figure 24) taken together represent the fact that the binding of Ste11 to Ste50 is not mutually exclusive to Ste11 binding to Ste5. Figure 25 shows reaction `bind_Ste11_Ste50` which binds Ste11 to Ste50 and is generalized to cover all states of the Ste11 to Ste5 binding site. Figure 26 shows reaction `bind_Ste11_Ste5` binding Ste11 to Ste5 and is generalized to cover all states of the Ste11 to Ste50 binding site.

Note that this proposal restricts the use of `GenericBond` structures to ensure that is feasible to interpret systems of reactions containing `GenericBond` structures. In particular a reaction containing `GenericBonds` can't be reversible.

### 5.8.1 Missing Binding Sites Infers Unchanged State

Labelling the connection point of generic bonds enables the modeler to specify a reaction which moves an component of unspecified type from one binding site to another. However in the majority of cases the binding site of such a component is not changed by the reaction i.e. the binding site state is completely irrelevant to the reaction. The encoding of these cases is simplified: if a binding site is both unchanged by a reaction *and* the reaction generalized to cover all states of that binding site then that binding site is simply omitted from the `Reaction` structure entirely. This is the case for the binding sites referenced by the `GenericBond` structures in the `bind_Ste11_Ste5` reaction shown in Figure 26 on page 24 and thus we can simplify this reaction by omitting the `GenericBond` structures as shown in Figure 27 on page 25. This simple generalization syntax can't be applied to `SpeciesType` structures. The state of all binding sites must be resolved in a `SpeciesType`.

```
<reaction id="binding_Ste5_Ste11_v3">
    <listOfReactants>
        <speciesReference>
            <listOfSpeciesTypeInstances>
                <speciesTypeInstance id="iSte5" speciesType="Ste5"/>
            </listOfSpeciesTypeInstances>
            <listOfBonds>
                <specificBond>
                    <bindingSiteReference speciesTypeInstance="iSte5" bindingSite="r241"/>
                </specificBond>
                <specificBond>
                    <bindingSiteReference speciesTypeInstance="iSte5" bindingSite="r463"/>
                </specificBond>
                <specificBond>
                    <bindingSiteReference speciesTypeInstance="iSte5" bindingSite="r744"/>
                </specificBond>
            </listOfBonds>
        </speciesReference>
        <speciesReference>
            <listOfSpeciesTypeInstances>
                <speciesTypeInstance id="iSte11" speciesType="Ste11"/>
            </listOfSpeciesTypeInstances>
            <listOfBonds>
                <specificBond>
                    <bindingSiteReference speciesTypeInstance="iSte11" bindingSite="site"/>
                </specificBond>
            </listOfBonds>
         </speciesReference>
    </listOfReactants>
    <listOfProducts>
        <speciesReference>
            <listOfSpeciesTypeInstances>
                <speciesTypeInstance id="iSte5" speciesType="Ste5"/>
                <speciesTypeInstance id="iSte11" speciesType="Ste11"/>
            </listOfSpeciesTypeInstances>
            <listOfBonds>
                <specificBond>
                    <bindingSiteReference speciesTypeInstance="iSte5" bindingSite="r241"/>
                </specificBond>
                <specificBond>
                    <bindingSiteReference speciesTypeInstance="iSte5" bindingSite="r463"/>
                    <otherBindingSiteReference
                        speciesTypeInstance="iSte11" bindingSite="site"/>
                </specificBond>
                <specificBond>
                    <bindingSiteReference speciesTypeInstance="iSte5" bindingSite="r744"/>
                </specificBond>
            </listOfBonds>
        </speciesReference>
    </listOfProducts>
</reaction>
```

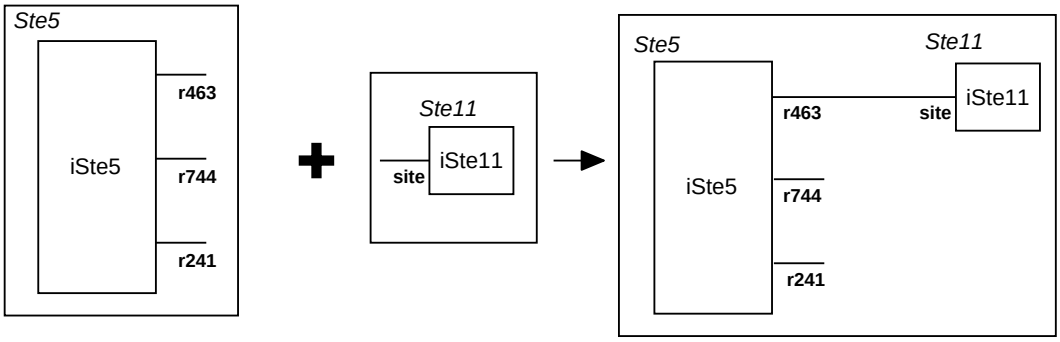


**Figure 15:** *The* `binding_Ste5_Ste11_v3` *reaction which demonstrates the representation of unbound states. For example* `BindingSite` `r744` *on* `SpeciesTypeInstance` `iSte5` *is unbound before and after the reaction*

```
<model id="Phosphorylation_model">
    <listOfSpeciesTypes>
        <speciesType id="Phosphate">
            <listOfBindingSites>
                <bindingSite id="site"/>
            </listOfBindingSites>
        </speciesType>
        <speciesType id="Ste11">
            <listOfBindingSites>
                <bindingSite id="S302"/>
            </listOfBindingSites>
        </speciesType>
    </listofSpeciesTypes>
    <listOfReactions>
        <reaction id="Phosphorylation">
            <listOfReactants>
                <speciesReference>
                    <listOfSpeciesTypeInstances>
                        <speciesTypeInstance id="iSte11" speciesType="Ste11"/>
                    </listOfSpeciesTypeInstances>
                    <listOfBonds>
                        <specificBond>
                            <bindingSiteReference
                                speciesTypeInstance="iSte11" bindingSite="S302"/>
                        </specificBond>
                        <specificBond>
                            <bindingSiteReference
                                speciesTypeInstance="iSte11" bindingSite="T307"/>
                        </specificBond>
                    </listOfBonds>
                </speciesReference>
            </listOfReactants>
            <listOfProducts>
                <speciesReference>
                    <listOfSpeciesTypeInstances>
                        <speciesTypeInstance id="iSte11" speciesType="Ste11"/>
                        <speciesTypeInstance id="iPhosphate_1" speciesType="Phosphate"/>
                        <speciesTypeInstance id="iPhosphate_2" speciesType="Phosphate"/>
                    </listOfSpeciesTypeInstances>
                    <listOfBonds>
                        <specificBond>
                            <bindingSiteReference speciesTypeInstance="iSte11"
                                bindingSite="S302"/>
                            <otherBindingSiteReference
                                speciesTypeInstance="iPhosphate_1" bindingSite="site"/>
                        </specificBond>
                        <specificBond>
                            <bindingSiteReference
                                speciesTypeInstance="iSte11" bindingSite="T307"/>
                            <otherBindingSiteReference
                                speciesTypeInstance="iPhosphate_2" bindingSite="site"/>
                        </specificBond>
                    </listOfBonds>
                </speciesReference>
            </listOfProducts>
        </reaction>
    </listOfReactions>
<model>
```

**Figure 16:** *The* `Phosphorylation_model` *model, a diagram of this model is shown in Figure 17. This model demonstrates how the proposed structures can be used to represent phosphorylation reactions.*

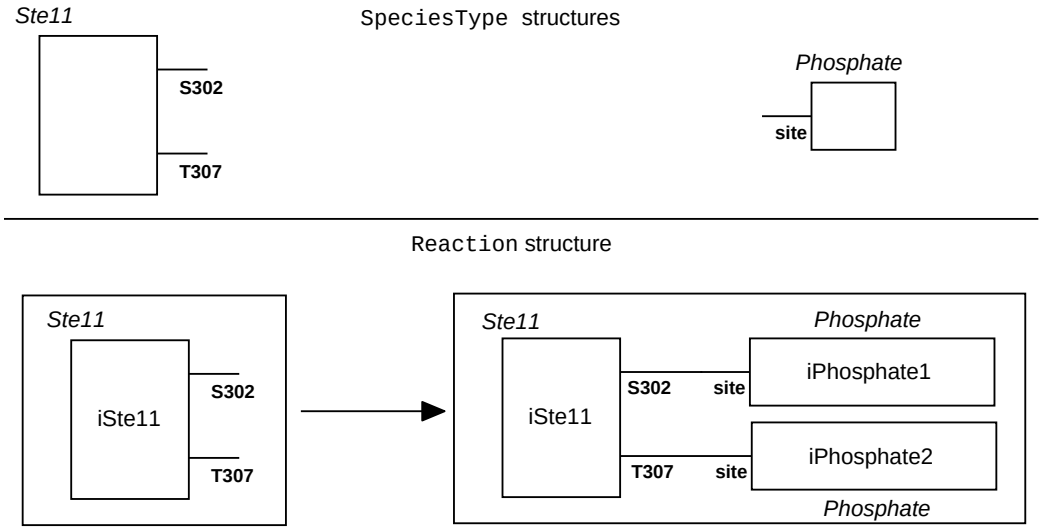


**Figure 17:** *Diagram of the* `Phosphorylation_model` *model*

```
<model id="demo">
    <listOfSpeciesTypes>
        <speciesType id="Ste20"/>
        <speciesType id="Ste11"/>
    </listofSpeciesTypes>
    <listOfReactions>
        <reaction id="Implausible">
            <listOfReactants>
                <speciesReference>
                    <listOfSpeciesTypeInstances>
                        <speciesTypeInstance id="iSte11" speciesType="Ste11"/>
                    </listOfSpeciesTypeInstances>
                </speciesReference>
            </listOfReactants>
            <listOfProducts>
                <speciesReference>
                    <listOfSpeciesTypeInstances>
                        <speciesTypeInstance id="iSte20" speciesType="Ste20"/>
                    </listOfSpeciesTypeInstances>
                </speciesReference>
            </listOfModifiers>
        </reaction>
    </listOfReactions>
</model>
```

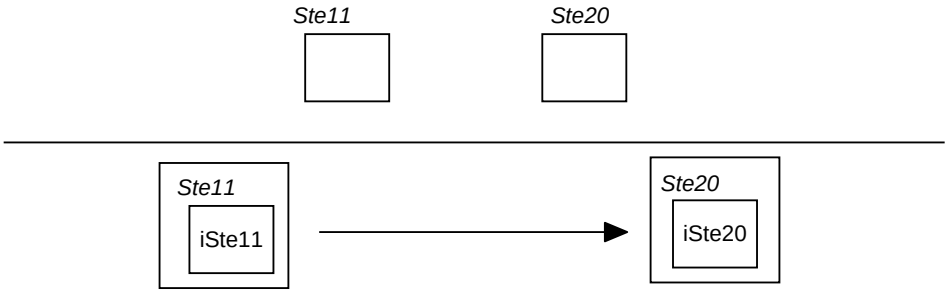


**Figure 18:** *The* `demo` *model which shows how a reaction operating on components can transform those components.*

```
<model id="disconnected_parts">
    <listOfSpeciesTypes>
        <speciesType id="A"/>
        <speciesType id="B">
            <listOfBindingSites>
                <bindingSite id="b">
            </listOfBindingSites>
        </speciesType>
        <speciesType id="C">
            <listOfBindingSites>
                <bindingSite id="c">
            </listOfBindingSites>
        </speciesType>
        <speciesType id="D">
            <listOfSpeciesTypeInstances>
                <speciesTypeInstance id="iA" speciesType="A"/>
                <speciesTypeInstance id="iB" speciesType="B"/>
                <speciesTypeInstance id="iC" speciesType="C"/>
            </listOfSpeciesTypeInstances>
            <listOfBonds>
                <specificBond>
                    <bindingSiteReference speciesTypeInstance="iB" bindingSite="b"/>
                    <otherBindingSiteReference speciesTypeInstance="iC" bindingSite="c"/>
                </specificBond>
            </listOfBonds>
        </speciesType>
    </listOfSpeciesTypes>
    <listOfReactions>
        <listOfReactants>
            <speciesReference>
                <listOfSpeciesTypeInstances>
                    <speciesTypeInstance id="iA" speciesType="A"/>
                 </listOfSpeciesTypeInstances>
            </speciesReference>
            <speciesReference>
                <listOfSpeciesTypeInstances>
                    <speciesTypeInstance id="iB" speciesType="B"/>
                    <speciesTypeInstance id="iC" speciesType="C"/>
                </listOfSpeciesTypeInstances>
                <listOfBonds>
                    <specificBond>
                        <bindingSiteReference speciesTypeInstance="iB" bindingSite="b"/>
                        <otherBindingSiteReference speciesTypeInstance="iC" bindingSite="c"/>
                    </specificBond>
                </listOfBonds>
            </speciesReference>
        </listOfReactants>
        <listOfProducts>
            <speciesReference>
                <listOfSpeciesTypeInstances>
                    <speciesTypeInstance id="iA" speciesType="A"/>
                    <speciesTypeInstance id="iB" speciesType="B"/>
                    <speciesTypeInstance id="iC" speciesType="C"/>
                </listOfSpeciesTypeInstances>
                <listOfBonds>
                    <specificBond>
                        <bindingSiteReference speciesTypeInstance="iB" bindingSite="b"/>
                        <otherBindingSiteReference speciesTypeInstance="iC" bindingSite="c"/>
                    </specificBond>
                </listOfBonds>
            </speciesReference>
        </listOfProducts>
    </listOfReactions>
</model>
```

**Figure 19:** *The `disconnected_parts` model which shows how reactions can operate on disconnected components.*

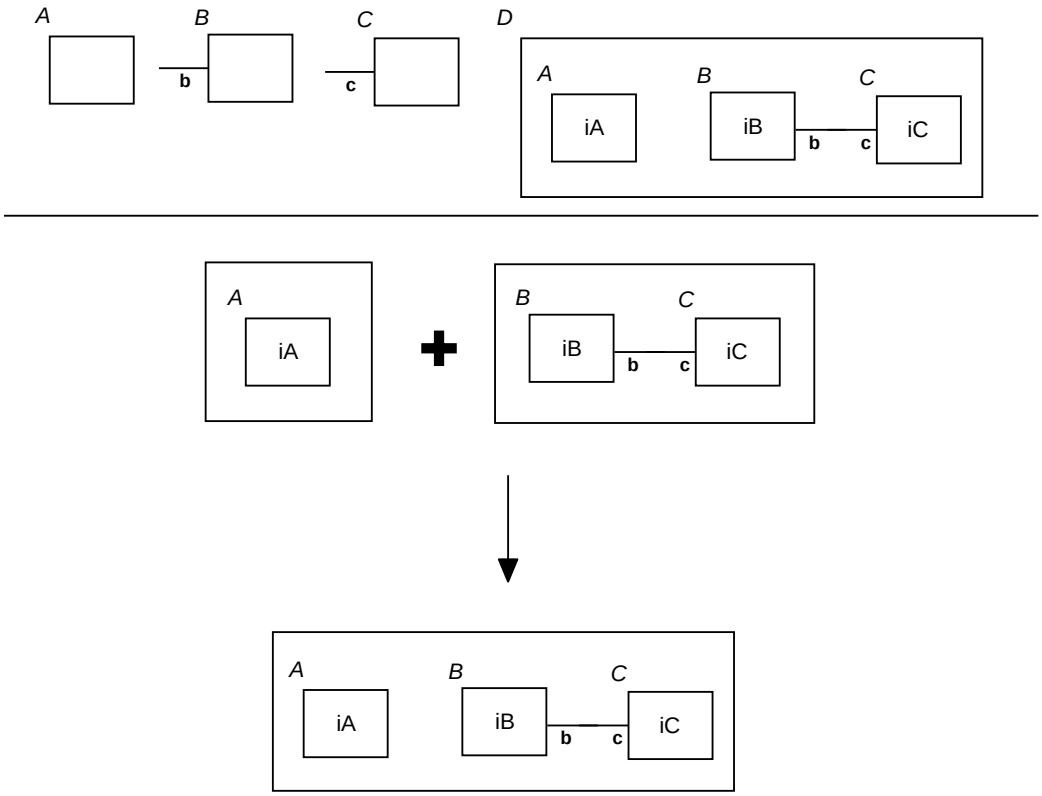


**Figure 20:** *A diagram of the* `disconnected_parts` *model.*

In source.xml:

```
<?xml version="1.0"?>
<sbml
    xmlns="http://www.sbml.org/sbml/level3"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    level="3
    version="1">

    <model id="imported">
        <listOfSpeciesTypes>
            <speciesType id="simple">
                <listOfBindingSites>
                    <bindingSite id="a"/>
                    <bindingSite id="b"/>
                </listOfBindingSites>
            </speciesType>
        </listOfSpeciesTypes>
    </model>
</sbml>
```

In importing.xml:

```
<?xml version="1.0"?>
<sbml
    xmlns="http://www.sbml.org/sbml/level3"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    level="3
    version="1">

    <model id="importing">
        <listOfSpeciesTypes>
           <speciesType
               xlink:type="imported_simple"
               xlink:href=
"source.xml#xpointer(/sbml/model/listOfSpeciesTypes/speciesType[@id=%22inner%22])"/>
           </speciesType>
        </listOfSpeciesTypes>
    </model>
</sbml>
```

**Figure 21:** *A simple example of* `SpeciesType` *import. The file* `importing.xml` *imports the* `SpeciesType` `simple` *from file* `source.xml`*. Within* `import.xml` *this type is referred to by the identifier* `imported_simple`*.*

```
<model id="generalized">
    <listOfSpeciesTypes>
        <speciesType id="A">
            <listOfBindingSites>
                <bindingSite id="a"/>
            </listOfBindingSite>
        </speciesType>
        <speciesType id="B">
            <listOfBindingSites>
                <bindingSite id="b1"/>
                <bindingSite id="b2"/>
            </listOfBindingSites>
        </speciesType>
    </listOfSpeciesTypes>
    <listOfReactions>
        <reaction id="generic">
            <listOfReactants>
                <speciesReference>
                    <listOfSpeciesTypeInstances>
                        <speciesTypeInstance id="iA" speciesType="A"/>
                    </listOfSpeciesTypeInstances>
                    <listOfBonds>
                        <specificBond>
                            <bindingSiteReference speciesTypeInstance="iA" bindingSite="a"/>
                        </specificBond>
                    </listOfBonds>
                </speciesReference>
                <speciesReference>
                    <listOfSpeciesTypeInstances>
                        <speciesTypeInstance id="iB" speciesType="B"/>
                    </listOfSpeciesTypeInstances>
                    <listOfBonds>
                        <specificBond>
                            <bindingSiteReference speciesTypeInstance="iB" bindingSite="b1"/>
                        </specificBond>
                        <genericBond id="X">
                            <bindingSiteReference speciesTypeInstance="iB" bindingSite="b2"/>
                        </genericBond>
                    </listOfBonds>
                </speciesReference>
            </listOfReactants>
            <listOfProducts>
                <speciesReference>
                    <listOfSpeciesTypeInstances>
                        <speciesTypeInstance id="iA" speciesType="A"/>
                        <speciesTypeInstance id="iB" speciesType="B"/>
                    </listOfSpeciesTypeInstances>
                    <listOfBonds>
                        <specificBond>
                            <bindingSiteReference speciesTypeInstance="iA" bindingSite="a"/>
                            <otherBindingSiteReference
                                speciesTypeInstance="iB" bindingSite="b1"/>
                        </specificBond>
                        <genericBond id="X">
                            <bindingSiteReference speciesTypeInstance="iB" bindingSite="b2"/>
                        </genericBond>
                    </listOfBonds>
                </speciesReference>
            </listOfProducts>
        </reaction>
    </listOfReactions>
</model>
```

**Figure 22:** *The `generalized` model which shows how a reaction can be applied to a set of chemical entities. A diagram of this model is shown in Figure 23. The state of `BindingSite` `b2` is not relevant to the reaction: the reaction applies to a `B` chemical entity in any state.*

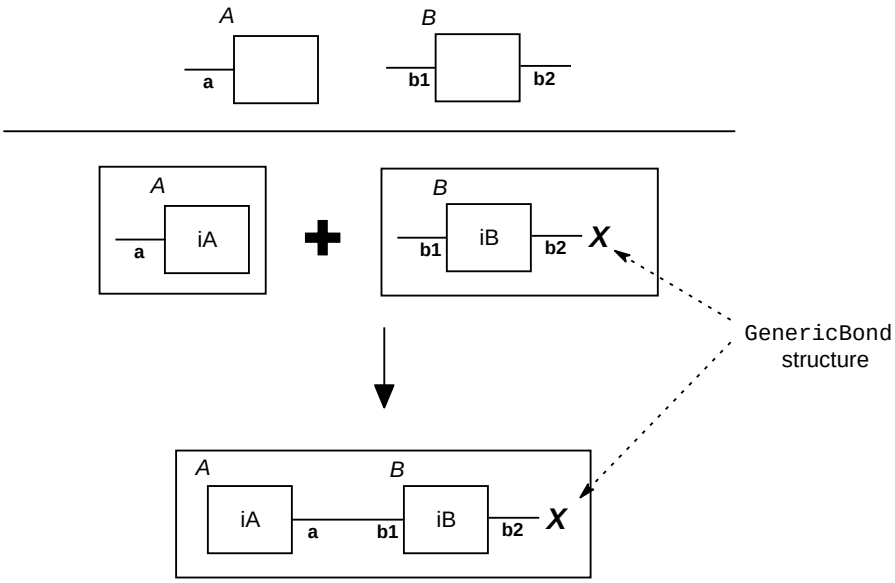


**Figure 23:** *A diagram of the* `generalized` *model shown in Figure 22*

```
<listOfSpeciesTypes>
    <speciesType id="Ste5">
        <listOfBindingSites>
            <bindingSite id="R463_514"/>
        </listOfBindingSites>
    </speciesType>
    <speciesType id="Ste50">
        <listOfBindingSites>
            <bindingSite id="SAM"/>
        </listofBindignSites>
    </speciesType>
    <speciesType id="Ste11">
        <listOfBindingSites>
            <bindingSite id="N_term"/>
            <bindingSite id="SAM"/>
        </listOfBindingSites>
    </speciesType>
</listOfSpeciesTypes>
```

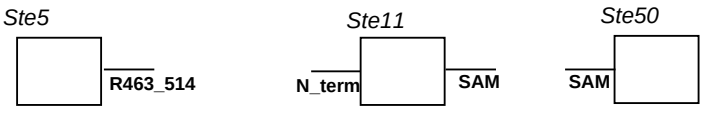


**Figure 24:** *The species types used in Figures 25, 26 and 27.*

```
<reaction id="bind_Ste11_Ste50">
    <listOfReactants>
        <speciesReference>
            <listOfSpeciesTypeInstances>
                <speciesTypeInstance id="iSte11" speciesType="Ste11"/>
            </listOfSpeciesTypeInstances>
            <listOfBonds>
                <genericBond id="X">
                    <bindingSiteReference bindingSite="N_term" speciesTypeInstance="iSte11"/>
                </genericBond>
                <specificBond>
                    <bindingSiteReference bindingSite="SAM" speciesTypeInstance="iSte11"/>
                </specificBond>
            </listOfBonds>
        </speciesReference>
        <speciesReference>
            <listOfSpeciesTypeInstances>
                <speciesTypeInstance id="iSte50" speciesType="Ste50"/>
            </listOfSpeciesTypeInstances>
            <listOfBonds>
                <specificBond>
                    <bindingSiteReference bindingSite="SAM" speciesTypeInstance="iSte50"/>
                </specificBond>
            </listOfBonds>
        </speciesReference>
    </listOfReactants>
    <listOfProducts>
        <speciesReference>
            <listOfSpeciesTypeInstances>
                <speciesTypeInstance id="iSte11" speciesType="Ste11"/>
                <speciesTypeInstance id="iSte50" speciesType="Ste50"/>
            </listOfSpeciesTypeInstances>
            <listOfBonds>
                <genericBond id="X">
                    <bindingSiteReference bindingSite="N_term" speciesTypeInstance="iSte11"/>
                </genericBond>
                <specificBond>
                    <bindingSiteReference bindingSite="SAM" speciesTypeInstance="iSte11"/>
                    <OtherBindingSiteReference
                        bindingSite="SAM" speciesTypeInstance="iSte50"/>
                </specificBond>
            </listOfBonds>
        </speciesReference>
    </listOfProducts>
</reaction>
```

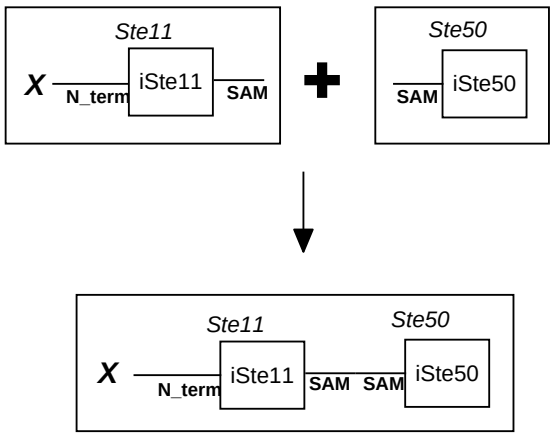


**Figure 25:** *The* `bind_Ste11_Ste50` *generalized binding reaction that operate on the types defined in Figure 24*

```
<reaction id="bind_Ste11_Ste5">
    <listOfReactants>
        <speciesReference>
            <listOfSpeciesTypeInstances>
                <speciesTypeInstance id="iSte11" speciesType="Ste11"/>
            </listOfSpeciesTypeInstances>
            <listOfBonds>
                <genericBond id="X">
                    <bindingSiteReference bindingSite="SAM" speciesTypeInstance="iSte11"/>
                </genericBond>
                <specificBond>
                    <bindingSiteReference bindingSite="N_term" speciesTypeInstance="iSte11"/>
                </specificBond>
            </listOfBonds>
        </speciesReference>
        <speciesReference>
            <listOfSpeciesTypeInstances>
                <speciesTypeInstance id="iSte5" speciesType="Ste5"/>
            </listOfSpeciesTypeInstances>
            <listOfBonds>
                <specificBond>
                    <bindingSiteReference bindingSite="R463_514" speciesTypeInstance="iSte5"/>
                </specificBond>
            </listOfBonds>
        </speciesReference>
    </listOfReactants>
    <listOfProductss>
        <speciesReference>
            <listOfSpeciesTypeInstances>
                <speciesTypeInstance id="iSte11" speciesType="Ste11"/>
                <speciesTypeInstance id="iSte5" speciesType="Ste5"/>
            </listOfSpeciesTypeInstances>
            <listOfBonds>
                <genericBond id="X">
                    <bindingSiteReference bindingSite="SAM" speciesTypeInstance="iSte11"/>
                </genericBond>
                <specificBond>
                    <bindingSiteReference bindingSite="N_term" speciesTypeInstance="iSte11"/>
                    <otherBindingSiteReference
                        bindingSite="R463_514" speciesTypeInstance="iSte5"/>
                </specificBond>
            </listOfBonds>
        </speciesReference>
    </listOfProductss>
</reaction>
```

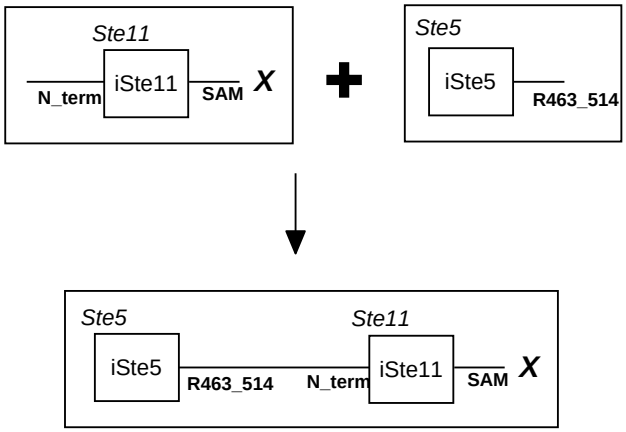


**Figure 26:** *The `bind_Ste11_Ste5` generalized reaction that operates on the types defined in Figure 24*

```
<reaction id="bind_Ste11_Ste5_v2">
    <listOfReactants>
        <speciesReference>
            <listOfSpeciesTypeInstances>
                <speciesTypeInstance id="iSte11" speciesType="Ste11"/>
            </listOfSpeciesTypeInstances>
            <listOfBonds>
                <specificBond>
                    <bindingSiteReference bindingSite="N_term" speciesTypeInstance="iSte11"/>
                </specificBond>
            </listOfBonds>
        </speciesReference>
        <speciesReference>
            <listOfSpeciesTypeInstances>
                <speciesTypeInstance id="iSte5" speciesType="Ste5"/>
            </listOfSpeciesTypeInstances>
            <listOfBonds>
                <specificBond>
                    <bindingSiteReference bindingSite="R463_514" speciesTypeInstance="iSte5"/>
                </specificBond>
            </listOfBonds>
        </speciesReference>
    </listOfReactants>
    <listOfProducts>
        <speciesReference>
            <listOfSpeciesTypeInstances>
                <speciesTypeInstance id="iSte11" speciesType="Ste11"/>
                <speciesTypeInstance id="iSte5" speciesType="Ste5"/>
            </listOfSpeciesTypeInstances>
            <listOfBonds>
                <specificBond>
                    <bindingSiteReference bindingSite="N_term" speciesTypeInstance="iSte11"/>
                    <otherBindingSiteReference
                        bindingSite="R463_514" speciesTypeInstance="iSte5"/>
                </specificBond>
            </listOfBonds>
        </speciesReference>
    </listOfProducts>
</reaction>
```

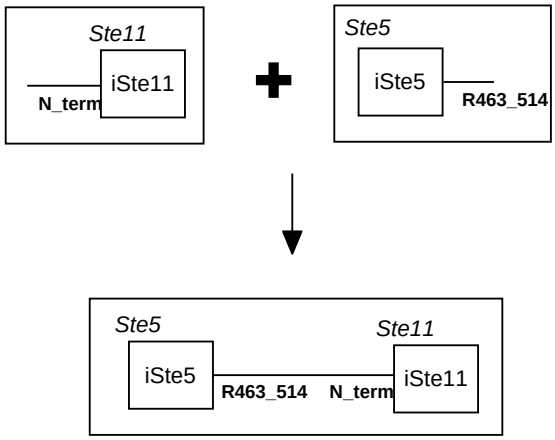


**Figure 27:** *The `bind_Ste11_Ste5_v2` reaction that operates on the types defined in Figure 24 on page 22 and is a simplification of the `bind_Ste11_Ste5` reaction shown in Figure 26 on page 24. The `SAM` binding site on `Ste11` is not changed by this reaction and the reaction is generalized to cover all states of that binding site. As a result the `SAM` binding site on `Ste11` can and has been omitted from the reaction.*

## 5.9 Hierarchal Species Types and Type Equivalence

This proposal supports the hierarchial assembly of `SpeciesType` structures to an arbitrary depth. The examples referenced so far have deliberately used only structures of limited hierarchal depth. This section describes the support for hierarchial assembly in the proposal.

`SpeciesType` structures can encapsulate a graph of instances of species types whilst exposing a subset of the available binding sites. The implementation of this consists of a reference, on a `BindingSite` structure, to a binding site on a chemical entity internal to the `SpeciesType`. The example model in Figure 28 on Page 27 demonstrates this feature. In particular note `BindingSite p` on `SpeciesType C`. The `p` refers to `BindingSite y` on `SpeciesTypeInstance iB`. `p` is synonymous with `y` on `iB`. In the diagram `p` is shown as a line which starts on the edge of `iB` and ends outside `C`.

This proposal contains a simple scheme for species type equivalence (described in more detail in Section 6.3). This equivalence is used for resolving whether for example the products of different reactions refer to the same species. In this scheme a distinction is made between species types enclosing one or more other species type instances and those that do not. I'll call those types which do not contain `SpeciesTypeInstance` structures *simple* all other types I'll call *complex*. Two simple species types are never equivalent however two complex species types can be. To evaluate the equivalence of *complex* types we first normalize them into an equivalent form where all species type instances are *simple* species types. This normalization process simply removes the intermediate levels in the hierarchy. Species type equivalence then considers a normalized type as a graph, formed by the species type instances, which are graph nodes, and bonds, which are graph arcs. Two species types are equivalent if their graphs are equivalent.

The `Complex2 SpeciesType` in Figure 29 on Page 28 is equivalent to the `Complex SpeciesType` in Figure 28 on Page 27.

```
<model id="Hierarchical">
    <listOfSpeciesTypes>
        <specesType id="A">
            <listOfBindingSites>
                <bindingSite id="site"/>
            </listOfBindingSites>
        </speciesType>
        <specesType id="B">
            <listOfBindingSites>
                <bindingSite id="x"/>
                <bindingSite id="y"/>
            </listOfBindingSites>
        </speciesType>
        <specesType id="C">
            <listOfSpeciesTypeInstances>
                <speciesTypeInstance id="iA" speciesType="A"/>
                <speciesTypeInstance id="iB" speciesType="B"/>
            </listOfSpeciesTypeInstances>
            <listOfBonds>
                <specificBond>
                    <bindingSiteReference speciesTypeInstance="iA" bindingSite="site"/>
                    <otherBindingSiteReference speciesTypeInstance="iB" bindingSite="x"/>
                </specificBond>
            </listOfBonds>
            <listOfBindingSites>
                <bindingSite id="p">
                    <bindingSiteReference speciesTypeInstance="iB" bindingSite="y"/>
                </bindingSite>
            </listOfBindingSites>
        </speciesType>
        <speciesType id="Complex">
            <listOfSpeciesTypeInstances>
                <speciesTypeInstance id="iA" speciesType="A"/>
                <speciesTypeInstance id="iC" speciesType="C"/>
            </listOfSpeciesTypeInstances>
            <listOfBonds>
                <specificBond>
                    <bindingSiteReference speciesTypeInstance="iA" bindingSite="site"/>
                    <otherBindingSiteReference speciesTypeInstance="iC" bindingSite="p"/>
                </specificBond>
            </listOfBonds>
        </speciesType>
    </listOfSpeciesTypes>
</model>
```

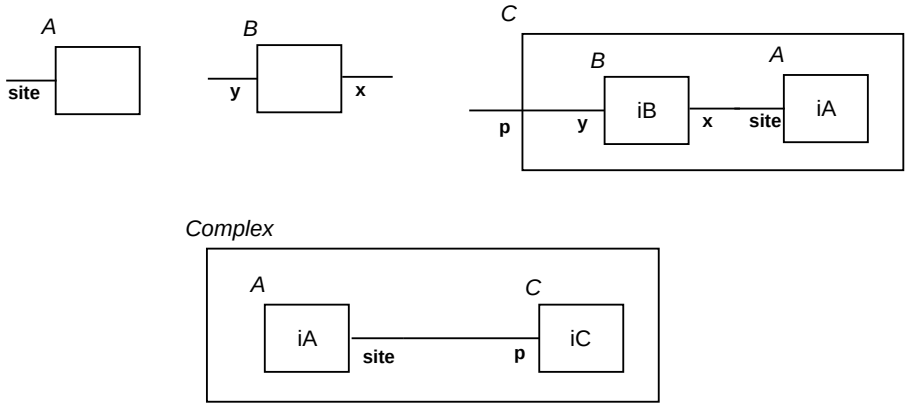


**Figure 28:** *The `Hierarchical` model which demonstrates the graph encapsulation facilities of the proposal. The `BindingSite` `p` on `SpeciesType` `C` exposes the uncommitted `BindingSite` `y` on `SpeciesTypeInstance` `iB`.*

```
<speciesType id="Complex2">
    <listOfSpeciesTypeInstances>
        <speciesTypeInstance id="iA1" speciesType="A"/>
        <speciesTypeInstance id="iA2" speciesType="A"/>
        <speciesTypeInstance id="iB" speciesType="C"/>
    </listOfSpeciesTypeInstances>
    <listOfBonds>
        <specificBond>
            <bindingSiteReference speciesTypeInstance="iA1" bindingSite="site"/>
            <otherBindingSiteReference speciesTypeInstance="iB" bindingSite="y"/>
        </specificBond>
        <specificBond>
            <bindingSiteReference speciesTypeInstance="iA2" bindingSite="site"/>
            <otherBindingSiteReference speciesTypeInstance="iB" bindingSite="x"/>
        </specificBond>
    </listOfBonds>
</speciesType>
```

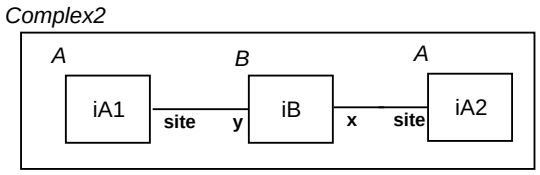


**Figure 29:** *The `Complex2` `SpeciesType` that is equivalent to the `Complex` type in Figure 28 on Page 27*

# 6 Formal Definition of Proposal

## 6.1 Introduction

This proposal builds on the structures and semantics of SBML Level 2. Section 6.2 describes the various structures introduced by this proposal and also describes how SBML Level 2 structures are augmented by this proposal.

This proposal describe operations on chemical entities without describing specific instances of those entities. The *species* term used in this document refers to a pool of chemical entities of identical structure located in a specific compartment. The term *species type* refers to a set of *species* all with identical structure across the entire modelled system. The issue of equivalence among species and among species types is described in more detail in Section 6.4. The `Species` and `SpeciesType` structures are the subsets of *species* and *species types* that are made explicit a SBML document. The remaining species types are defined by reactions. The remaining species are implied by the existence of species types.

Just as a model doesn't identify individual chemical entities nor does is it necessarily need to identify specific species and species types which are operated on by the reactions in the model. Instead reactions imply the existence of classes of species types. This is described in detail in Section 6.5.

## 6.2 Proposed Classes in Detail

This section describes in detail each class in the proposal as shown in figure 1. As in the diagram only new or extended classes are described in this section. All level 2 structures and basic semantics are assumed to be part of the proposal. For each new or extended class the definition of the class and its fields are described.

`id` fields on structures, in this section, enclosed within a `SpeciesType` structure are unique to that structure only. `id` fields on structures, in this section, enclosed within a `Reaction` structure are unique to that reaction only (apart from the `id` fields of `SimpleSpeciesReferences` which are unique to a whole model).

### 6.2.1 Bond

The abstract base class `Bond` represents one or more chemical bonds or forces holding two chemical entities together enabling them to form a larger chemical entity. A `bond` can represent covalent and non-covalent bonds. The existence of a `bond` can imply some modification of the chemical entities. For example a phosphorylated protein can be represented in a model as a `bond` between a protein and a phosphate group. In such a model the fate of the hydrogen atom bound to the phosphorylation site may or may not be represented explicitly and does not affect the identity of the protein. A `Bond` structure consists of one `BindingSiteReference` field, `bindingSite`, which indicates where the bond has effect. The linkage of chemical entities together to form a larger chemical entity can be represented without using `bonds`. See section 5.5.

### 6.2.2 BindingSite

A `BindingSite` structure represents a logical site where a bond may form on the chemical entity represented by the enclosing `SpeciesType` structure. A `BindingSite` may represent a set of physical binding sites which are treated as a single entity for the purposes of the model. A `BindingSite` structure consists of

- `id`, a mandatory `SId` field to identify the site
- `name`, an optional string field (see SBML Level 2)
- `bindingSiteReference`, an optional field containing a `BindingSiteReference` structure. When this attribute is present the binding site is exposing an internal binding site on a `SpeciesTypeInstance`. This `bindingSiteReference` refers to a binding site internal to the enclosing `SpeciesType` which is not referenced by another `bindSiteReference`.

A `bindingSiteReference` value must be present if the enclosing `SpeciesType` is *complex* (contains one or more structures of type `SpeciesTypeInstance`). This means that a `SpeciesType` cannot 'introduce' a new

binding site that doesn't exist on the chemical entities that make up the SpeciesType. This restriction is designed to ensure that evaluating type equivalence is straightforward.

### 6.2.3 BindingSiteReference

A reference to an instance of a BindingSite in a given SpeciesGraph. A BindingSiteReference structure consists of

- speciesTypeInstance, a SId field, which refers to a SpeciesTypeInstance within the enclosing SpeciesGraph; and
- bindingSite, a SId field, which refers to a binding site on that instance. The bindingSite must be declared on the SpeciesType referenced by the SpeciesTypeInstance.

The combination of attribute values of a given BindingSiteReference structure cannot occur in any other BindingSiteReference structure within the same SpeciesGraph structure.

### 6.2.4 GenericBond

A GenericBond structure represents a chemical bond between a specific binding site and an unspecified chemical entity which is unchanged by a reaction. GenericBond is a subtype of Bond which is in turn a subtype of SBase.

On GenericBond the id field, inherited from SBase, becomes mandatory and identifies a connection to the unspecified chemical entity. The bindingSiteReference field represents the binding site that the entity is connected to. GenericBond structures can only occur within Reactions. (Specifically they can only occur in the bond array/list field in SimpleSpeciesReference structures.) The values of GenericBond id fields are specific to, and unique within, the containing reaction. All GenericBond id fields with the same value in the same reaction refer to the same chemical entity. GenericBond id fields in different reactions refer to different chemical entities.

### 6.2.5 Model

See SBML Level 2 for the existing definition of model. This proposal adds to a Model structure a new field: *speciesType* which consists of a list of SpeciesType structures.

### 6.2.6 Reaction

A reaction is either located inside a specific compartment, across more than 2 or more compartments or potentially in all compartments. This final case is indicated by the absence of compartment and species fields on all enclosed SimpleSpeciesReference structures. In this case the Reaction is replaced with a set of Reaction structures, one for each Compartment in the model. In these Reaction structures the SimpleSpeciesReference structures the match with Species in the given Compartment. See section 5.3 for examples of reactions generalized across compartments. Sections 6.5 and 6.6 describe how reactions should be interpreted.

The following rules apply to reaction structures:

1. A SpeciesTypeInstance id value must be unique across the set of reactant SpeciesReference structures on a given reaction. A SpeciesTypeInstance id value must be unique across the set of product SpeciesReference structures on a given reaction.
2. A reference to the same specific BindingSite on a specific SpeciesTypeInstance must occur in the set of reactants and the set of products. This means that a binding site state determined in the reactants cannot be 'lost' from the set of products nor can the binding site state be generated from an undetermined state by the reaction.
3. A binding site reference missing entirely from a reaction implies a reaction generalized to cover all states of that binding site as follows. Consider a BindingSite of a SpeciesType referred to by

a SpeciesTypeInstance in a Reaction. If this binding site is not referenced in any place within the reaction, i.e. its is 'missing', then the reaction is equivalent to one in which the BindingSite *is* referenced by a GenericBond structure with a unique identity occurring in both the set of reactants and products. Such a reaction will not modify the state of the given BindingSite and is thus generalized to cover all states of that BindingSite.

4. If the reaction contains any GenericBond structures then the reaction must have a reversible attribute of false. *This restriction applies to simplify the enumeration of implied species. Species can be matched and transformed in single defined direction.*

### 6.2.7 SimpleSpeciesReference

See SBML Level 2 for the existing definition of SimpleSpeciesReference. SimpleSpeciesReference is the base class for: (a) SpeciesReference the type used to represent the reactants and products of a reaction and (b) ModifierSpeciesReference the type used to represent the modifiers of a reaction. SimpleSpeciesReference structures can only occur within a reaction. In this proposal a SimpleSpecies-Reference can refer to a set of Species both those that are explicitly defined and those that are created through generalized reactions (see section 5.8). (A SimpleSpeciesReference on its own does not imply the existence of a species.) In this proposal a SimpleSpeciesReference becomes a subtype of SpeciesGraph.

SimpleSpeciesReference has the following fields:

- id, an optional SId field. The value of this field can be used as a symbol, enclosed in MathML ci elements, within the KineticLaw structure of the enclosing Reaction structure.
- substanceUnits, spatialSizeUnits and hasOnlySubstanceUnits, these optional fields have the same semantics as the corresponding attributes on Species. These attributes default to the values of matching Species structures before following the Species semantics, for example, the spatial dimensions of the compartment in which the species is located must match those of the spatialSizeUnits. *This means that a reaction may refer to a species in its kinetic law using different units to that used in the species' initial definition. It is expected that a SBML interpreter will perform a units conversion.*
- species, this SId field is present in Level 2 however we now make this field optional. This field refers to a Species that is involved in the reaction. If this field is present then the fields inherited from SpeciesGraph as well as the compartment, bond and speciesType fields are not available.
- speciesType, this SId field refers to a SpeciesType that is involved in the reaction. If this field is present then the fields inherited from SpeciesGraph as well as the Species and bond fields are not available. If the compartment field is present then the SimpleSpeciesReference refers to the Species of the given SpeciesType located in the given compartment; otherwise the SimpleSpeciesReference refers to a the set of Species of the given SpeciesType.
- compartment, this SId field refers to a Compartment where the matching species are located.

If a SimpleSpeciesReference contains SpeciesTypeInstance structures then stoichiometry field must have a value of one (the default value). *This restriction applies because it is not possible (at least within this scheme) to identify the n separate entities that would match with a SpeciesReference structure with a stoichiometry of n where $n \neq 1$. Without this clear identification the interpretation of the product structures of the reaction is impossible.*

### 6.2.8 Species

As in SBML Level 2 a Species structure represents a pool of a given chemical entity located in a specific compartment. This proposal introduces one optional SId field, speciesType which refers to the SpeciesType (chemical entity) to be located in the Compartment referenced by the compartment field. There can only one Species structure in a model with a given pair of values for the speciesType and compartment attributes i.e. a given SpeciesType cannot be located in the same Compartment more than once.

When the speciesType field is not present then the Species structure is equivalent to a Species structure which does contain a speciesType field. This field would refer to a SpeciesType that is not referenced anywhere else in the model. In short a Species structure without a speciesType field has a 'hidden' SpeciesType associated with it. This hidden species type contains no binding sites or species type instance structures.

### 6.2.9 SpeciesGraph

SpeciesGraph is an abstract base class. A SpeciesGraph structure represents a type of chemical entity or a set of types of chemical entities of a specific common form. The form of these entities is defined as a graph where the nodes are SpeciesTypeInstance structures and the arcs are Bond structures. The graph can be disconnected indicating that the details of how parts of the chemical entities are associated are not relevant to the model (see Section 5.5 for examples).

A SpeciesGraph structure is composed of the following fields:

- speciesTypeInstance, this is an optional list of SpeciesTypeInstance structures that form the SpeciesGraph. If this list is not present then the SpeciesGraph simply represents an chemical entity for which the detail of its composition is not relevant to the model.
- bond, an optional list of Bond structures that can include both SpecificBond and GenericBond structures. This list links the chemical entities enumerated in the speciesTypeInstance field.

### 6.2.10 SpeciesType

The class SpeciesType represents a type of chemical entity. The existence of a SpeciesType implies the existence of a species of that type in every compartment. These species have zero concentration and a boundary and constant attribute values of false. These implied species are overloaded by Species structures of the given SpeciesType. A set of equivalent SpeciesTypes only imply a single species per Compartment. The equivalence of SpeciesTypes is defined in Section 6.3.

SpeciesType is derived from SpeciesGraph, and has the following fields:

- id a mandatary SId field that identifies the SpeciesType
- name, an optional string field (see SBML Level 2)
- bindingSite, an optional BindingSite list, which contains the set of binding sites that are located on the SpeciesType.
- href an optional XLink field which contains an XPointer (DeRose et al., 2002) string that points to an SpeciesType structure. The content of this field is restricted to consist only of the form, in BNF:

```
Xpointer ::=
  (URI)"#xpointer(/sbml/model"
  modelreference*
  "/listOfSpeciesTypes/speciesType[@id=%22" SId "%22])"

modelreference ::= "/listOfSubmodels/model[@id=%22" SId "%22]"
```

  (This definition supports the use of submodels as proposed in Finney (2003))

  *Having this restricted form means that parsers won't be required to parse and execute the whole XPath syntax which may require SBML streams to be stored in DOM form for access by generic XPointer evaluators. However tools that can interpret the full XPath syntax would still be able to interpret the XLink attributes. This restricted form is unambiguous unlike some potential XPath strings.*

- type, which must have the value simple, simply indicates the XLink type of SpeciesTypeInstance.

speciesTypeInstance and bond fields are inherited from SpeciesGraph. The bonds list inherited from SpeciesGraph must only contain SpecificBond structures. A simple example of the use SpeciesType structures is given in section 5.2. SpeciesType structures have the following restrictions on their form:

1. A SpeciesType structure can either define:
   - a SpeciesType, using the speciesTypeInstance, bond and bindingSite fields; or
   - import a definition from another SpeciesType structure either inside or outside the current document, using the href and type fields. The type field must be present if the href field is used.
2. Consider all the BindingSite structures of the SpeciesType structures referenced by the SpeciesTypeInstance structures in a given SpeciesType structure. Each of these BindingSite structures should be referenced exactly once by a BindingSiteReference structure enclosed in the SpeciesType structure. This means that the status of a BindingSite can't be left undefined or ambiguous by a SpeciesType.
3. A SpeciesType structure can only import a single SpeciesType. The SpeciesType structures referenced from within an imported SpeciesType are not imported with it. The identifiers of the BindingSite structures in an imported SpeciesType can be used unchanged in the importing model. A SpeciesType structure containing a href field value is equivalent to an SpeciesType structure with the same form as the referenced SpeciesType. Any reference to an SpeciesType refers through any chain of SpeciesType structures to the original SpeciesType which does not contain a href field. All SpeciesType structures referring, in this way, to the same SpeciesType are equivalent.

### 6.2.11 SpeciesTypeInstance

A SpeciesTypeInstance structure represents the occurrence of a chemical entity of a given SpeciesType within a SpeciesGraph. A SpeciesTypeInstance structure has the following fields:

- id a mandatory SId field that identifies the SpeciesTypeInstance. This field is unique to the enclosing SpeciesGraph structure and the enclosing Reaction structure if it exists. Two SpeciesTypeInstance structures with the same id in a reaction refer to the same chemical entity.
- name, an optional string field (see SBML Level 2)
- speciesType, a mandatory SId field which refers to the SpeciesType that the SpeciesTypeInstance is an instance of.

### 6.2.12 SpecificBond

SpecificBond is a subtype of Bond. SpecificBond represents either (a) one or more chemical bonds between two explicitly identified binding sites or (b) an unoccupied binding site. A bond structure consists of two BindingSiteReference structures: bindingSiteReference and otherBindingSiteReference. bindingSiteReference is inherited from Bond. otherBindingSiteReference is optional.

The SpecificBond structure represents the state in which bindingSite is unbound if otherBindingSiteReference is not present. If otherBindingSiteReference is present the bindingSite and otherBindingSiteReference structures represent the 2 binding sites that are linked by a bond. In which case neither binding site has privileged semantics. See section 5.6 for examples.

## 6.3 Equivalence of Species Types

This section defines the equivalence of species types and applies equally to the species types explicitly defined by SpeciesType structure and those implied by the Reaction structures. An interpreter of models in the proposed format may evaluate the equivalence of species types for a number of reasons including:

- to determine if 2 Species structures are equivalent and thus determine that the model is invalid;
- to divide the set of species types into equivalent subsets so as to create a species in each compartment for each subset; and
- to match the products of reactions with the set of species in a modelled system.

Before we define species type equivalence some definitions are required:

- a `SpeciesGraph` is *simple* if it contains no `SpeciesTypeInstance` structures;
- a `SpeciesGraph` is *complex* if it contains one or more `SpeciesTypeInstance` structures.
- a *standard* `SpeciesGraph` is a complex `SpeciesGraph` that does not contain any `BindingSite` structures.
- a `SpeciesGraph` structure is *normalized* if the `SpeciesGraph` is standard and the set of `SpeciesType` structures, referred to by the set of `SpeciesTypeInstance` structures, are simple i.e. it consists of one hierarchical level.

The above definitions apply to both types of `SpeciesGraph`: `SpeciesType` or `SimpleSpeciesReference`.

Species type equivalence is defined as a process with 3 stages described by the following sections in order. First the species types are transformed into a standard `SpeciesGraph` as described in Section 6.3.1. Second these `SpeciesGraph` are normalized (flattened), as described in Section 6.3.2. Finally if these normalized structures can then be matched, as described in Section 6.3.3, then the species types are equivalent.

### 6.3.1 Transforming a species type into a standard `SpeciesGraph`

The transformation of a `SpeciesType` into a standard `SpeciesGraph` depends on whether it is complex or not:

- A simple `SpeciesType` structure is transformed by initially creating a `SpeciesGraph` that contains one `SpeciesTypeInstance` structure which refers to the original simple `SpeciesType`. All binding sites of the simple `SpeciesType` are referenced as unbound in a set of `SpeciesBond` structures within the new normalized `SpeciesGraph`. The result is a new standard complex `SpeciesGraph` which refers to the original simple species type. The complex `SpeciesGraph` is used for equivalence evaluation i.e its passed to the following stages.
- A complex `SpeciesType` with one or more `BindingSite` structures is transformed to remove the those `BindingSite` structures. Each `BindingSite` structure is replaced by a `SpecificBond` structure containing just the `BindingSiteReference` structure that was contained in the `BindingSite` structure. This means that the original binding sites are made unbound by this process (instead of being available for binding in a different context).

### 6.3.2 Normalization of Species Graphs

This section describes how the form of species graphs is normalized for the matching process. This normalization process simply reduces the given standard `SpeciesGraph` to a single hierarchical level.

The normalization process consists of iteratively replacing any `SpeciesTypeInstance` structures that refer to complex `SpeciesType` structures with a set of new `SpeciesTypeInstance` structures and `SpecificBond` structures that are copies of that occurring within the complex `SpeciesType` structure. These new structures are linked into the same 'outer' `Bond` structures as the `SpeciesTypeInstance` structures they replace.

### 6.3.3 Normalized Species Graph Equivalence

When evaluating the equivalence of normalized `SpeciesGraph` structures the following aspects are not directly relevant:

- `id` on `SpeciesTypeInstance`
- the order of structures within lists `SpecificBond` structures representing unbound binding sites

The `id` on `SpeciesTypeInstance` is only used to provide linkage within a graph.

In this section we consider a normalized `SpeciesGraph` to be a formal graph. Two `SpeciesGraph` structures are equivalent if their formal graphs are equivalent. A formal representation of a graph is

**Definition** (A Simple Graph) A *simple graph* $G$ is a tuple $G = (G_V, G_E, L, I, J, s, t, \ell, i, j)$ consisting of

- a finite set of nodes (or "vertices") $G_V$ and a finite set of arcs (or "edges") $G_E$ where $G_v \cap G_E = \emptyset$,
- two total mappings $s, t : G_E \to G_V$ ("source and target"),
- a set of node labels $L$,
- a set of arc source labels $I$,
- a set of arc target labels $J$,
- a total mapping $\ell : G_V \to L$ ("node labelling")
- a total mapping $i : G_E \to I$ ("arc source labelling")
- a total mapping $j : G_E \to J$ ("arc target labelling")

The nodes and arcs of a graph are also collectively called the "objects" of the graph (or "graph objects"). Note that in this definition that we are labelling the source and target 'ends' of each arc.

*This definition is variant of the form taken from (Rudolf, 1998)*

In this formulation of the formal graphs the graph nodes, $G_V$, are the `SpeciesTypeInstance` structures and the arcs, $G_E$, are the `SpecificBond` structures. The source and target of an arc is determined by the `speciesTypeInstance` field of the `BindingSiteReference` structures enclosed in the given `SpecificBond`. The arc direction (the distinction between the source and target) is determined by a consistent ordering over `speciesType` attributes of the `speciesTypeInstance` structures.

The label, $\ell(v)$ of a node, $v$ is a XLink reference to the `SpeciesType` referenced indirectly by the `speciesType` field of the node. The referenced `SpeciesType` must be simple and must not contain a `href` value (the chain of `href` values should be completely evaluated). Two XLink labels are equal when they refer to the same `SpeciesType` structure. The source label, $i(e)$ of an arc, $e$, is formed from the `bindingSite` attribute of a `BindingSiteReference` structure on the given `SpecificBond` structure. The arc target label, $j(e)$ of an arc, $e$, is formed from the `bindingSite` attribute of the other `BindingSiteReference` structure.

Formally a graph is equivalent to another if there exists a graph morphism, which is a mapping of one graph's object sets into the other's, with some restrictions to preserve the graph's structure and it's typing information:

**Definition** (Graph Morphism) A *graph morphism* $m : L \to G$ between two simple graphs

- $L = (L_V, L_E, L_L, I_L, J_L, s_L, t_L, \ell_L, i_L, j_L)$ and
- $G = (G_V, G_E, L_G, I_G, J_G, s_G, t_G, \ell_G, i_G, j_G)$

is a pair of total one to one mappings $m = (m_V : L_V \to G_V, m_E : L_E \to G_E)$, where the following restrictions apply:

1. $\mid L_V \mid = \mid G_V \mid$
2. $\mid L_E \mid = \mid G_E \mid$
3. $\forall e \in L_E$ :
   - $m_V(s_L(e)) = s_G(m_E(e))$
   - $m_V(t_L(e)) = t_G(m_E(e))$

4. $\forall v \in L_V : \ell_L(v) = \ell_G(m_V(v))$
5. $\forall e \in L_E : i_L(e) = i_G(m_E(e))$
6. $\forall e \in L_E : j_L(e) = j_G(m_E(e))$

*This definition is variant of the form taken from (Rudolf, 1998)*

### 6.3.4 Implications of `SpeciesType` Equivalence

Two or more equivalent `SpeciesType` structures can co-exist in the same model (of course the `id` attributes must have different values). Section 6.4 describes how `Species` equivalence is derived from `SpeciesType` equivalence.

## 6.4 `Species` Equivalence

Two `Species` are equivalent if they are located in the same compartment and their associated `SpeciesType` structures are equivalent. A model must not contain equivalent `Species`. The species implied by the reactions in a model, described in Section 6.5, are inherently not equivalent.

## 6.5 Simplification of Reactions

This section describes in outline how the proposed `Reaction` structures can be transformed into SBML Level2 reaction structures. Any reaction that follows the Level 2 form obviously ignores this process. This process definition is used as a way to define the semantics of reactions relative to SBML Level 2. An interpreter of the proposal format may or may not actually implement this transformation. The process has the following stages:

1. This transformation process starts by replacing a reaction which does not refer to a `Compartment` with a set of reactions one for each defined compartment in the model. (The `compartment` field on the `SimpleSpeciesReference` structures are set to refer to the given `Compartment`.)
2. At this point those `SimpleSpeciesReference` structures that have a `speciesType` attribute can be replaced by a `species` that refers to the species that represents the given `SpeciesType` in the given `Compartment`. This species may be already be explicitly defined by a `Species` structure or already defined as an implicit species as described in Section 6.2.10.
3. For those `SimpleSpeciesReference` structures that are not transformed by the previous stage the next step is to normalize the `SimpleSpeciesReference` structures according to the process described in 6.3.2. In this normalization process any `GenericBond` structures should be treated as if they were one half of a `SpecificBond` structure.
4. If the `Reaction` does not contain `GenericBond` structures then the normalized graph is matched to the set of `SpeciesType` in the model as described in Section 6.3.3. For those that do not match with an existing `SpeciesType` a new `SpeciesType` is created corresponding to the normalized graph. The graph is then discarded and the `SimpleSpeciesReference` is replaced by a simple reference to the `Species` corresponding to the `SpeciesType` as described in stage 2 above.
5. If `Reaction` does contain `GenericBond` structures then the `Reaction` is interpreted with all other `Reaction` structures of that type as described in Section 6.6.

## 6.6 Semantics of Reactions containing `GenericBond` structures

This section describes the semantics of reactions containing `GenericBond` structures. These reactions are transformed as described in Section 6.5 and then interpreted together as described in this section.

### 6.6.1 Simple Framework for Operational Semantics

For the definition of the semantics of reactions we will consider a model with a simulator to be form of AI *production system*.

> The major elements of an AI production system are a *global database*, a set of *production rules*, and a *control system*. [...] Depending on the application, this database may be as simple as a small matrix of numbers or as complex as a large, relational, index file structure. (The reader should not confuse the phrase, "global database," as it is used [here], with the databases of database systems.)
>
> The production rules operate on the global database. Each rule has a *precondition* that is either satisfied or not by the global database. If the precondition is satisfied, the rule can be *applied*. Application of the rule changes the database. The control system chooses which applicable rule should be applied and ceases computation when a termination condition on the global database is satisfied. (Nilsson, 1982)

In the context of this proposal reactions are considered to be rules. The global database is formed by the modelled species which are pools of chemical entities. The initial state of the global database consists of the explicitly defined `Species` structures and the species implied from `SpeciesType` structures (see Section 6.2.10) `Reaction` structures (see Section 6.5).

It is expected that many analyzes of models will require the computation of the set of species however this proposal does not depend on any particular representation scheme for species within a given software analysis system. The set of species and species types are properties of the global database. A given representation scheme may only deal with individual chemical entities. Similarly another representation may not track individual entities but instead compute the concentration of species.

A reaction matches the set of reactants, its precondition, to the set of species, in the global database. The effect of the reaction is to remove reactant entities and create product entities within the set of species. In this scheme modifiers are treated as if they occur once in both the set reactants and the set of products.

For the purposes of this definition the control system is idealized. Real software systems will in practice create approximations of this control system. The ideal control system simply applies all matching reactions concurrently at the rate defined by the reaction' kinetic laws. Unlike a AI production system this proposal does not define any termination conditions.

### 6.6.2 Matching of Reactants Containing `GenericBond` Structures to Species

The matching of reactant `SpeciesReference` structures, to species is similar to equivalence between species. (`SpeciesReference` is a subclass of `SimpleSpeciesReference`.) To enable the definition of a match between a `SpeciesReference` and a species we first require a definition of a *generic graph* and definition of `SpeciesReference` in terms of a *generic graph*. The use *generic graph* here is to capture the semantics of `GenericBond` structures which are contained within the `SpeciesReference`.

**Definition** (A Generic Graph) A *generic graph* $G$ is a tuple $G = (G_V, G_E, G_X, L, I, J, s, t, u, \ell, i, j)$ consisting of

- a finite set of nodes (or "vertices") $G_V$, a finite set of arcs (or "edges") $G_E$ and a finite set of generic nodes $G_X$ where
  1. $G_V \cap G_E = \emptyset$
  2. $G_V \cap G_X = \emptyset$
  3. $G_E \cap G_X = \emptyset$
- two total mappings $s, t : G_E \to G_V$ ("source and target"),
- a total mapping $u : G_X \to G_V$ ("generic connection"),

- a set of arc source labels $I$,
- a set of arc target labels $J$,
- a total mapping $\ell : G_V \cup G_X \rightarrow L$ ("node labelling")
- a total mapping $i : G_E \rightarrow I$ ("arc source labelling")
- a total mapping $j : G_E \rightarrow J$ ("arc target labelling")

The nodes, generic nodes and arcs of a graph are also collectively called the "objects" of the generic graph (or "generic graph objects").

The formulation of a `SimpleSpeciesReference` as a generic graph is similar to that of a `SpeciesType` to a simple graph. The graph nodes, $G_V$, are the `SpeciesTypeInstance` structures, arcs, $G_E$, are the `SpecificBond` structures and generic nodes, $G_X$, are the `GenericBond` structures. The source, $s$ and target, $t$, of an arc are determined by the `speciesTypeInstance` field of the `BindingSiteReference` structures enclosed in the given `SpecificBond`. The arcs are directed where arc direction should be determined using an consistent ordering over the set of simple species types. The generic connections of a generic node, $u$ are determined by the `speciesTypeInstance` field of the `BindingSiteReference` structure enclosed in the given `GenericBond`. The label, $\ell(v)$ of a node, $v$ is a XLink reference to the `SpeciesType` referenced indirectly by the `speciesType` field of the node. The referenced `SpeciesType` must be simple and must not contain a `href` value (the chain of `href` values should be completely evaluated). Two XLink labels are equal when they refer to the same `SpeciesType` structure. The label or type of an arc is formed from the pair of `bindingSite` attributes of the given `SpecificBond` structures. The arc direction determines the ordering of each arc's `bindingSite` attributes in the arc's label. The label of a generic node is the `bindingSite` attribute of the given `GenericBond` structure.

When matching a `SimpleSpeciesReference` (a generic graph) to a species we are in fact matching the `SimpleSpeciesReference` to the normalized `SpeciesGraph` (a simple graph) of the species type associated with the given species. A match of generic graph to a simple graph is given by a graph morphism, which is a mapping of the generic graph's object sets into the simple graph's, with some restrictions to preserve the generic graph's structure and it's typing information:

**Definition** (Generic Graph Morphism) A *generic graph morphism* $n : L \rightarrow G$ between

- a generic graph $L = (L_V, L_E, L_X, L_L, I_L, J_L, s_L, t_L, u_L, \ell_L, i_L, j_L)$ and
- a simple graph $G = (G_V, G_E, L_G, I_G, J_G, s_G, t_G, \ell_G, i_G, j_G)$

is a triple

$n = (n_V : L_V \rightarrow G_V, n_E : L_E \cup L_X \rightarrow G_E, n_X : L_X \rightarrow G_E)$

where $n_V$ and $n_E$ are total mappings; $n_X$ is a partial mapping and the following restrictions apply:

1. $\forall x \in L_X : n_V(u(x)) = s_G(n_X(x)) \vee n_V(u(x)) = t_G(n_X(x)) \vee n_X(x) = \emptyset$
2. $\forall e \in L_E$ :
   - $n_V(s_L(e)) = s_G(n_E(e))$
   - $n_V(t_L(e)) = t_G(n_E(e))$
3. $\forall v \in L_V : \ell_L(v) = \ell_G(n_V(v))$
4. $\forall e \in L_E : i_L(e) = i_G(n_E(e))$
5. $\forall e \in L_E : j_L(e) = j_G(n_E(e))$

6. $\forall x \in L_X : \ell_L(x) = \begin{cases} \emptyset & \text{if } n_X(x) = \emptyset \\ i_G(n_X(x)) & \text{if } n_V(u(x)) = s_G(n_X(x)) \\ j_G(n_X(x)) & \text{if } n_V(u(x)) = t_G(n_X(x)) \end{cases}$

The mapping $n_X$ from the set of generic bonds, $L_X$, to specific bonds, $G_E$, represents a set of assignments which are used to determine the effect of a reaction. This is described further in Section 6.6.3.

### 6.6.3 The Effect of Reactions Containing `GenericBond` Structures

Reactions containing `GenericBond` structures have the same general effect as reactions without these structures: reactants are consumed, products are created and modifiers are left unchanged. In this section we describe the effect of a reaction as graph transformation process. The resulting product species graphs can then be matched to species. In this transformation process we take each set of matching reactant species graphs. All the `SpeciesTypeInstance`, `BindingSiteReference` and `SpecificBond` structures in these graphs that are mapped from equivalent structures in the reactant `SpeciesReference` structures are discarded. The result is that subgraphs of the matching reactant species graphs remain. These subgraphs are composed of only those components not made explicit by the corresponding reactant `SpeciesReference` structures.

The product graphs are created starting from the species graphs in the product `SpeciesReference` structures. The `GenericBond` structures are replaced by `SpecificBond` structures via the mapping $n_X$ created by the matching process. These structures are then combined with the subgraphs that are left from the reactant species to form the set of product species graphs. If these species graphs are not equivalent to existing species types in the global database then the global database will now include those species types and associated species.

### 6.6.4 Matching Reactions Containing `GenericBond` Structures

A `Reaction` which does not contain any `GenericBond` structures represents a single *reaction instance*. A `Reaction` that does contain `GenericBond` structures represents a set of reaction instances where each instance in this set is a concrete reaction with generic graph mappings for all the reactant and modifier `SimpleSpeciesReference` structures. There is a separate reaction instance for all possible combinations of these graph mappings. Each reaction instance exists concurrently with the other instances and operates at the rate determined by the reaction's kinetic law.

*This means that some care must be taken when formulating the kinetic laws of this type of reaction. Typically to obtain the correct results kinetic laws should be defined so that the rate of the reaction is directly proportional to the amount of the species matched using `GenericBond` structures.*

## References


DeRose, S., Maler, E., and Orchard, D. (2001). XML Linking Language (XLink) Version 1.0 W3C Recommendation. Available via the World Wide Web at `http://www.w3.org/TR/2000/REC-xlink-20010627/`.

DeRose, S., Ron Daniel, J., Grosso, P., Maler, E., Marsh, J., and Walsh, N. (2002). XML Pointer Language (XPointer) Version 1.0 W3C Working Draft 16 august 2002. Available via the World Wide Web at `http://www.w3.org/TR/2002/WD-xptr-20020816/`.

Finney, A. (2001). Internal discussion document possible extension to the Systems Biology Markup Language Complex Species and Species Graphs. Available via the World Wide Web at `http://www.cds.caltech.edu/~afinney/CplxSpecies.pdf`.

Finney, A. (2003). Systems Biology Markup Language (SBML) Level 3 Proposal: Model composition features. Available via the World Wide Web at `http://www.cds.caltech.edu/~afinney/model-composition.pdf`.

Finney, A., Hucka, M., and Bolouri, H. (2002). Systems Biology Markup Language (SBML) Level 2: Structures and facilities for model definitions. Available via the World Wide Web at `http://www.sbw-sbml.org/`.

Goldstein, B., Faeder, J. R., Hlavacek, W. S., Blinov, M. L., Redondoc, A., and Wofsy, C. (2001). Modeling the early signaling events mediated by Fc$\varepsilon$RI. *Molecular Immunology*, 38:1213–1219.

Le Novère, N., Shimizu, T. S., and Finney, A. (2003). Systems Biology Markup Language (SBML) Level 3 Proposal: Multistate Features. Available via the World Wide Web at `http://sbml.org/multistates.pdf`.

Nilsson, N. J. (1982). *Principles of Artifical Intelligence*. Springer-Verlag.

Rudolf, M. (1998). Utilizing constraint satification techniques for efficient graph pattern matching. In *6th International Workshop on Theory and Application of Graph Transformations*.