Technical Report LA-UR-04-5445

Rule-based modeling of multi-component species. Proposal for SBML level 3.

Michael L. Blinov[1]*, James R. Faeder[1], Byron Goldstein[1], Andrew Finney[2], and William S. Hlavacek[1]

[1]Theoretical Biology and Biophysics Group, Los Alamos National Laboratory, USA
[2]University of Hertfordshire, UK

October 6, 2004

Contents

*To whom correspondence should be addressed at mblinov@lanl.gov

ABSTRACT

Recently we have developed BioNetGen software (Blinov et al., 2004; see also `http://cellsignaling.lanl.gov/bionetgen`) that allows a user to create computational models that characterize the dynamics of a signal transduction system composed of multi-component species, their enzymatic activities, potential post-translational modifications, and interactions of the domains of proteins. In the current proposal we introduce XML description that utilizes a graph-based approach for specifying multi-component species, and rule-based approach for specifying their activities and interactions. This XML description is proposed to serve as a proposal for SBML level 3.

1 Introduction

1.1 Problem of combinatorial complexity

A problem that one confronts when attempting to model a signaling system is combinatorial complexity, i.e., the need to consider a large number of potential chemical species because molecules involved in signaling each can be modified in a number of ways and can combine to form a variety of multi-component complexes. The molecules involved in cellular signaling typically consist of multiple domains and/or smaller molecular sites, such as sites that are substrates for protein tyrosine kinases. Each domain and site interacts with one or more binding partners, and these binding partners can contain multiple domains and sites (T. Pawson and P. Nash, 2003). The activities of domains and sites can be enzymatically modified, for example through phosphorylation or dephosphorylation reactions. For any signaling system, one must consider a spectrum of ephemeral complexes, composed of multiple components, each of which can occupy numerous modification states. For example, a receptor which contains $N = 10$ sites at which phosphate can be added or removed, can occupy $2^{10} = 1024$ different states. Thus, we need to account for $2^N = 1024$ **multi-state species**. If dimerization is required for signaling, one has to consider on the order of $2^{2N} = 1,048,576$ receptor dimers, i.e. **multi-component multi-state species**. Considering that each phosphorylation site may bind proteins that can be in different states themselves, the number of different states goes up dramatically. It was mentioned by Wofsy et al. (1997) that a large number of possible molecular species should be considered for signal transduction systems of the immune system. The problem of keeping track of a large number of species has been recognized as a serious challenge by several modelers, e.g. Arkin (2001), Endy and Brent (2001), Bray (2003), Hlavacek et al.(2003), Goldstein et al. (2004). Currently, the problem of generating and analysis of reaction networks with a large number of multi-state multi-component species is addressed by several teams, including
– Bioglyphics and BALSA (`http://www.csi.washington.edu/teams/modeling/projects/BALSA/`),
– BioNetGen (`http://cellsignaling.lanl.gov/bionetgen/`),
– BioSPICE (`https://community.biospice.org/`),
– Cellerator (`http://www.cellerator.info/`),
– little b (`http://vcp.med.harvard.edu/papers.html`),
– MONOD and Moleculizer (`http://monod.molsci.org/`) ,
– SigTran (`http://csi.washington.edu/teams/modeling/projects/sigtran/`),
and StochSim (`http://info.anat.cam.ac.uk/groups/comp-cell/StochSim.html`).

1.2 Rule-based domain-oriented modeling and SBML level 2 description

The problem of combinatorial complexity can be demonstrated for the case of FcεRI (Figure 1), a tetrameric complex that consists of an α chain, which contains the extracellular IgE binding site, a β chain, which interacts with the Src-family protein tyrosine kinase (PTK) Lyn, and two identical disulfide-linked γ chains, which interact with the PTK Syk. The β and γ chains each contain a single cytoplasmic immunoreceptor tyrosine-based activation motif (ITAM) which, when phosphorylated, serve as binding sites for Lyn and Syk. The model for interactions of kinases Lyn and Syk with FcεRI receptor with two *gamma* chains lumped together accounts for 354 distinct chemical species (Goldstein et al., 2002). The following procedure was

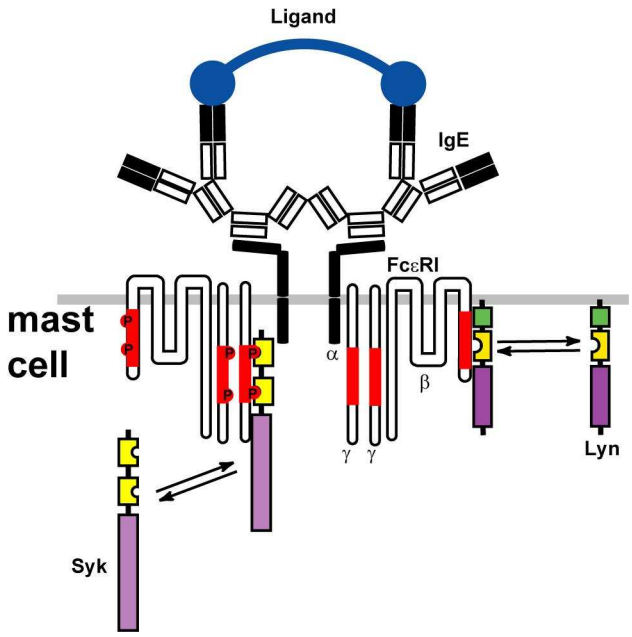


Figure 1: *Example of a multicomponent species*

followed to create a reaction network for this system:

1. Identify components and their interactions to generate a set of possible chemical species that participate in the signaling process being studied,
2. develop reaction rules for these species, to construct a chemical reaction network,
3. determine values of reaction rate constants and initial concentrations based on direct measurements and other experimental observations,
4. solve the equations numerically to obtain predictions for a given set of initial conditions,
5. to compare simulation results with experiments, add up the concentrations of species with a particular characteristic, e.g. phosphorylated gamma ITAM.

This approach has been reviewed by Hlavacek et al. (2003) and Goldstein et al. (2004), and implemented in general-purpose software, BioNetGen (Blinov et al., 2004; `http://cellsignaling.lanl.gov/bionetgen/`). To start modeling of a system, a modeler has to derive from his or her knowledge of the system the following information:

- molecules to be modeled.
- interaction and modification domains of the molecules.
- activities of these domains.
- rules of activities and interactions among domains and molecules.
- characteristics that correspond to measured quantities.

The molecules, their domains, their activities and interactions imply a set of possible chemical species and transitions among them.

All this information is contained in a single plain text file, which is processed by BioNetGen (algorithmic description will be soon available at `http://cellsignaling.lanl.gov/bionetgen/`) to create a reaction network, namely

- a set of all chemical species corresponding to specified components.
- a set of all transitions among these species, with one reaction rate assigned to all reactions among species satisfying specified conditions.
- multiple sets of species, that correspond to measured quantities.

The first two sets form a reaction network that can be written in SBML level 2 format (which BioNetGen generates). But the information about domains and their activities and interactions, as well as sets of species with particular characteristics, is not contained in a reaction network written in SBML level 2.

Thus, BioNetGen input and output files store more information than is apparent in SBML level 2 format.

Moreover, models generated with BioNetGen that include only few proteins, account for several thousand chemical species. An input describing FcϵRI receptor signaling system containing 2 signaling proteins, bivalent ligand and a receptor with two interaction sites consists of 150 lines. Model for FcϵRI, generated by BioNetGen is a set of 3,680 chemical reactions among 354 chemical species. It's representation in SBML level 2 format consists of 43,509 lines and takes 1.4 MB. Meanwhile BioNetGen was used to generate and analyze systems accounting for up to 45,000 species. Hence, SBML format that declares each species and reaction separately is not suitable for describing of large biological systems.

1.3 Aims of this proposal

The main goal of this proposal is to develop SBML level 3 standard that corresponds to data structures that can be easily translated into an Object-Oriented Programming language. The ultimate goal is to facilitate software development for modeling of multicomponent species. The development of this standard goes in parallel with the development of software capable of using this standard.

We set aims of this proposal corresponding to ones stated in SBML Level 3 charter (`http://www.sbw-sbml.org/sbml-discuss/archive/msg00174.html`).

The desired XML description should be able to

- store the information about multicomponent species and their components (domains), their enzymatic activities, potential post-translational modifications, conformational changes, and interactions of the domains.
- Incorporate hierarchical levels of biological information:
  - species interactions, as in the SBML level 2,
  - protein domain interactions, as in BioNetGen input file,
  - individual amino acids, polypeptide chains, sites and domains, as in protein databases;
- Store the information about experimentally observable features, like all species containing specific molecule, domain in a specific state etc.
- Allow SBML level 2 to be used as a part of the proposed XML description without any modifications.
- Incorporate species that can be located in different compartments, including species that are located in several compartments simultaneously. Define reactions across compartments, including trafficking.
- Be compatible with graphical diagrammatic representation of the proposed XML format. It is illustrated for each example within the paper. Additional XML tags are required to specify this diagrammatic representation. This representation could be an extension of the Layout proposal (Gauges et al, 2004), currently under discussion by SBML community, thus we don't discuss it now.

- Be scalable, i.e. a user can specify the level of detail required for a model within the same data structure;
- Be flexible, i.e. a user can change a model by adding or removing certain components without essential changes to the model, e.g. by adding new species or new protein domains and their interactions;
- Be expandable, for example, polypeptide chains interactions can be added to the model of protein domain interactions;
- Allow automatic data-mining from different databases (e.g. Swiss-Prot database at `http://us.expasy.org/sprot/`).
- Allow future extension to incorporate information about the 3D structure of molecules and chemical species.

This proposal benefited much from the SBML level 3 proposals by Finney (2004) and Le Novere et al. (2002), from discussions with Paul Loriaux and feedback from Andrew Finney. Similarities and differences between the proposed XML description and the earlier proposals are discussed in detail in Section 4.5.

2 Multi-component species and species graphs

2.1 Multi-component species and interactions

Typically, a multi-component species consists of several chemical molecules, such as proteins, that are associated into a complex. Each protein itself is a multi-component species generally comprised of multiple domains and sites (Pawson and Nash, 2003).

Some domains serve as binding sites for bimolecular interactions: protein interaction domains recognize specific types of sites in proteins and other biomolecules and direct the association of polypeptides with one another and with phospholipids, small molecules, or nucleic acids . For example, the Src homology 2 (SH2) domain recognizes phosphorylated tyrosines.

Non-covalent protein-protein interactions can be forestalled by modifications of these domains, such that covelent binding of a phosphate group to a tyrosine residue of a protein substrate (phosphorylation). These modifications can be reversed (e.g., a tyrosine can be dephosphorylated by a protein tyrosine phosphatase).

Protein-protein interactions are usually site-specific, but are affected by other protein domains, directly non-participating in binding, such as catalytic domain. A PTK domain, for example, catalyzes phosphorylation, which leads to recognition of this residue by SH2 domain of binding protein. The activities (both binding and catalytic) of protein domains may vary and are regulated by post-translational modifications, which may be catalyzed by signaling proteins. For example, the activity of a PTK domain can be upregulated by autophosphorylation of its activation loop, and the affinity of an SH2 domain for phosphotyrosine can be upregulated by PTK-mediated tyrosine phosphorylation.

Binding events and conformational changes can also affect the activities of signaling proteins.

Thus, to account for all possible player even for a simple protein-protein interaction, we need to identify and model multiple components within each of interacting proteins.

2.2 Graph representation of multi-component species

In our view, the fundamental objects of a biochemical reaction network are "components," collections of which form "molecules," collections of which form "complexes." In the graph representation, nodes of a graph are associated with components, and edges of a graph are associated with bonds between components. A molecule is defined as a set of components that can be treated as a unit, such as the components of a polypeptide chain or multimeric protein. A molecule is represented graphically by a box surrounding a set of nodes that represents each component of the molecule. Thus, bonds effectively connect molecules, with each

molecule having possible large number of bonds, as in Figure 2a, which illustrates an example of receptor tyrosine kinase (RTK) dimer.

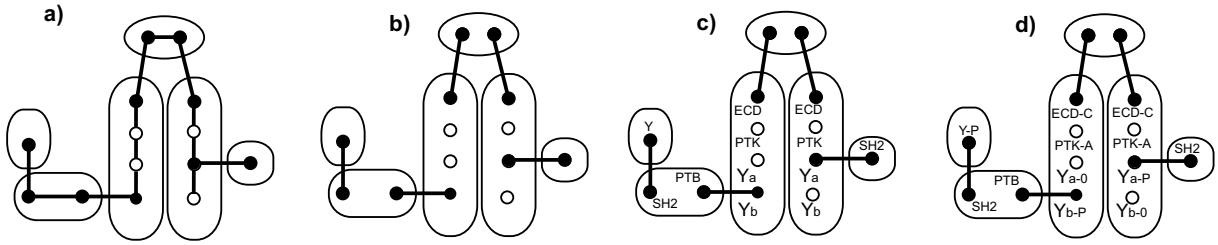


Figure 2: *Species graph of a receptor tyrosine kinase signaling complex. (a) A species graph with all bonds declared. (b) A species graph with only bonds between molecules declared. (c) A species graph with labels defining names of components. (d) A species graph with labels defining names and internal states of components.*

Edges within molecules may be unaffected by signaling, and thus don't need to be specified, as in Figure 2b. Thus, only edges that are subject to addition or removal during signaling are declared.

Although components are denoted by identical nodes (which may or may not be filled by the reasons explained in section 3.2), they all represent different functional domains, thus, they are assigned labels – names, such that extra-celular domain (ECD), SH2 domain, PTK, phosphotyrosine binding domain (PTB), tyrosine residue (Y) etc, as in Figure 2c.

Next, components may be assigned labels declaring the internal states of the component, as in Figure 2d. We introduce states of components for the following reasons. In some cases, it is simply convenient to introduce states, as when complexes can be indicated by the bound states of the components of a scaffold-like molecule. In other cases, in the absence of 3D structural information, we need states to distinguish the conformations of a molecule. After ECD is bound to a ligand, it can undergo conformational change and be in several modification states, e.g ECD-C. The enzymatic activity of a protein tyrosine kinase (PTK) domain is typically upregulated by (auto)phosphorylation of tyrosines in the activation loop of the kinase, which causes a conformational change. To distinguish the inactive and active forms of such a kinase, we need to track its conformational state, or equivalently the phosphorylation state of its activation loop. Internal state is an optional attribute, and may be omitted if modifications of a domain are unknown, as for SH2 and PTB domain.

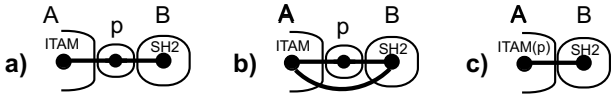


Figure 3: *Different schemes for declaration of phosphorylated ITAM – SH2 interaction*

In other cases, component states are not strictly required but they are biophysically justified and they simplify and/or clarify the representation. For example, consider a phosphorylation-dependent interaction between two proteins, say between protein "A" with a protein motif like the immunoreceptor tyrosine-based activation motif (ITAM) and a second protein "B" with a Src homology 2 (SH2) domain. Let's say that the ITAM of "A" and the SH2 domain of "B" interact when a tyrosine in "A"s ITAM is phosphorylated. It is not biophysical (and it is ambiguous) to represent this interaction as a linear graph with three nodes (ITAM, p, and SH2) and two unlabeled edges as follows: ITAM—p—SH2, as in Figure 3a. The ITAM and SH2 actually interact, with the interaction being affected by the phosphorylation state of ITAM. Thus, a more realistic graph looks like that of Figure 3b. It is more biophysical to write ITAM(p)—SH2, where (p) indicates that the ITAM is in the phosphorylated state, as in Figure 3c. One could write ITAM(u) to indicate the unphosphorylated state. Regardless, the introduction of the (p) and (u) states simplifies the representation, because now it is unnecessary to label or interpret the meaning of the edges. With ITAM—p—SH2, one must understand that the bond between the ITAM and the phosphate group is a covalent bond and that the bond between the phosphorylated ITAM and the SH2 domain is a non-covalent bond. One needs to

know this because it is impossible for the phosphate group covalently bound to the ITAM to leave with the SH2 domain in a dissociation reaction. Thus, ITAM(p)—SH2 is clearer, because one could never make the mistake of allowing the phosphate group of the ITAM to become associated with the SH2 domain; in this representation, it is clear that the phosphate group is covalently bound to the ITAM not the SH2 domain.

3 XML description of multi-component species and interactions among them

3.1 General XML template

SBML level 2 describes `species` and `reactions` among them. Our approach is to

- construct `molecules` from `components` (proteins, their domains, amino acids or any other molecules and atoms)
- construct `species` and `speciesGroups` from `molecules` connected by `bonds`
- specify `reactions` for interactions among `species` and `reactionRules` for interactions among `speciesGroups`.

The general scheme for XML encoding is as follows:

```
<listOfMolecules>
.........
.........molecules used to construct species. They themself are constructed from components.
.........
</listOfMolecules>

<listOfSpecies>
.........
.........uniquely defined species, used to define concentrations.
.........They are constructed as sets of Molecules connected with bonds.
.........
</listOfSpecies>

<listOfSpeciesGroups>
.........
.........groups of species, used to define reaction rules and observable features.
.........SpeciesGroup may contain a single species or a set of species having the same structure
.........
</listOfSpeciesGroups>

<listOfReactions>
.........
.........reactions among species, identical to SBML level 2 if species
.........are specified in listOfSpecies
.........
</listOfReactions>

<listOfReactionRules>
.........
.........reaction rules applied to speciesGroups in order to generate all
.........possible species and interactions among them.
.........
</listOfReactionRules>

<listOfObservables>
.........
.........Features of the system that need to be reported
.........
</listOfObservables>
```

3.2 Components and molecules

The nodes in a species graph are `components` (e.g., sites and domains of proteins), which have two attributes (see Figure 4a) :

- multiple `componentStates` (optional) ;

- componentConnectivity (required). We need to specify the connectivity of a node, for example, to write a group of species in which a particular component of a species must be unbound. Groups of species are discussed in section 3.4
  - componentConnectivity="bound" – a component (node) that is connected by a bond is denoted by a filled circle.
  - componentConnectivity="unbound" – a component (node) that is unconnected by a bond is denoted by an open (unfilled) circle.
  - componentConnectivity="unspecified" – a component (node) that may be either connected or unconnected by a bond is denoted by a semi-filled circle. It's a default state and can be omitted in both XML description and graphical representation.

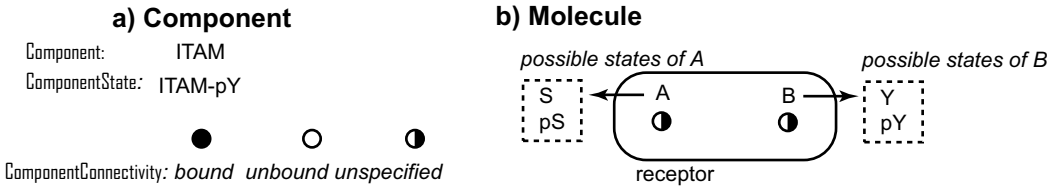


Figure 4: *Definition of (a) a component and (b) a molecule*

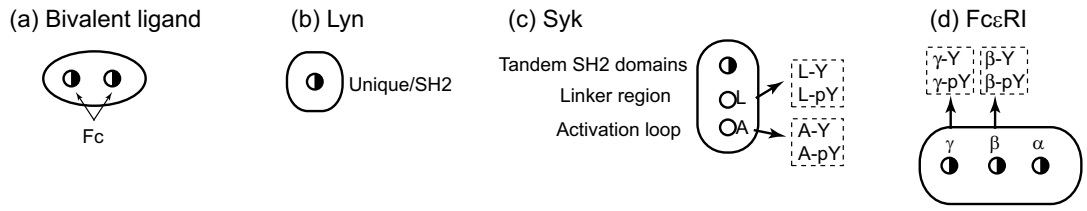
.

A molecule is defined as a set of components that can be treated as a unit, such as the components of a polypeptide chain or multimeric protein. A molecule is represented graphically by a box surrounding a set of nodes that represents each component of the molecule. Like components, molecules are assigned names. Figures 4b and 5 illustrates graphical and XML description of molecules.

The internal states of a molecule and its connectivity to external components is determined by the componentStates and componentConnectivities of molecule's components. If a component can be in several componentStates, then molecule can be in a large number of possible configurations. In the examples in Figure 5 Syk and FceRI molecules give rise to 4 different chemical species each.

Optional attribute for each component is multiplicity - if there are N identical domains within a molecule, this domain has multiplicity N. Contrary to the componentState, multiplicity of the component within molecule does not increase the number of possible chemical species consisting of one molecule, thus there is only one species of molecule Lig. By default multiplicity="1".

Additionally, attribute compartment can be specified for both molecule and component, for the purpose of specifying that a molecule or a component can be located within the specified compartment. Thus, a single molecule can be located across multiple compartments. We discuss the use of compartment attribute in more details in section 4.1.




```
<molecule id="Lig" compartment="extracellular">
      <listOfComponents>
      <component id="Fc" multiplicity="2"/>
      </listOfComponents>
</molecule>
<molecule id="Lyn" compartment="membrane">
      <listOfComponents>
      <component id="SH2"/>
      </listOfComponents>
</molecule>
<molecule id="Syk" compartment="intracellular" >
      <listOfComponents>
      <component id="SH2"/>
      <component id="L" componentConnectivity="unbound">
            <listOfComponentStates>
            <componentState id="Y"/>
            <componentState id="pY"/>
            </listOfComponentStates>
      </component>
      <component id="A" componentConnectivity="unbound">
            <listOfComponentStates>
            <componentState id="Y"  />
            <componentState id="pY"/>
            </listOfComponentStates>
      </component>
      </listOfComponents>
</molecule>
<molecule id="FcerI">
      <listOfComponents>
      <component id="alpha" compartment="extracellular"/>
      <component id="beta" compartment="membrane">
            <listOfComponentStates>
            <componentState id="Y"  />
            <componentState id="pY"/>
            </listOfComponentStates>
      </component>
      <component id="gamma" compartment="intracellular">
            <listOfComponentStates>
            <componentState id="Y"  />
            <componentState id="pY"/>
            </listOfComponentStates>
      </component>
      </listOfComponents>
</molecule>
```

Figure 5: *Examples of molecules in FcεRI receptor system. (a) Bivalent ligand with two identical components, defined by multiplicity="2". (b) Lyn molecule with a single binding site that has no* `componentStates`*. (c) Syk molecule that has one binding site for FcεRI gamma component and two phosphorylation sites that are unbound. (d) FcεRI multi-subunit complex, with one extracellular binding site and two intracellular phosphorylation/binding sites.*

3.3 Species

A chemical species is declared

- either as in SBML level 2: `<species id="S1" initialAmount="0">`
- or as a single `molecule` having all of its components fully defined, as in Figure 6. A component is fully defined if its internal state is specified and its connection with other components is specified.

```
<species id="SykPS" initialAmount="10">
        <listOfMoleculesIncluded>
        <moleculeIncluded molecule="Syk" id="Syk1">
                <listOfComponentsIncluded>
                <componentIncluded component="SH2" componentConnectivity="unbound"/>
                <componentIncluded component="L" componentState="Y" componentConnectivity="unbound"/>
                <componentIncluded component="A" componentState="pY" componentConnectivity="unbound"/>
                </listOfComponentsIncluded>
        </moleculeIncluded>
        </listOfMoleculesIncluded>
</species>
```

- or as a set of connected `molecules`, with each molecule in the set having all of its components and `bonds` among connected components fully defined, as in Figure 6. Graphically, if a component is bound to another component, then the nodes representing the two components are joined by an edge. Connected nodes are filled, and unconnected nodes are unfilled.

```
<species id="R-Lyn" initialAmount="10">
      <listOfMoleculesIncluded>
      <moleculeIncluded molecule="Lyn" id="Lyn1">
            <listOfComponentsIncluded>
            <componentIncluded component="SH2" id="LSH2" componentConnectivity="bound"/>
            </listOfComponentsIncluded>
      </moleculeIncluded>
      <moleculeIncluded molecule="FceRI" id="R1">
            <listOfComponentsIncluded>
            <componentIncluded component="alpha" componentConnectivity="unbound"/>
            <componentIncluded component="beta"  id="beta1" componentState="pY"
                                                    componentConnectivity="bound"/>
            <componentIncluded component="gamma" componentState="Y"
                                                 componentConnectivity="unbound"/>
            </listOfComponentsIncluded>
      </moleculeIncluded>
      </listOfMoleculesIncluded>
      <listOfBonds>
      <bond id="R1-Lyn">
            <listOfComponentsConnected>
            <componentConnected="LSH2"/>
            <componentConnected="beta1"/>
            </listOfComponentsConnected>
      </bond>
      </listOfBonds>
</species>
```

`species` can be viewed as graphs with all `components` within participating `molecules` being fully defined. Undirected and unlabeled edges in a graph represent `bonds` between `components`. As `components` are grouped into `molecules`, `bonds` effectively connect molecules, with each molecule having possible large number of bonds.

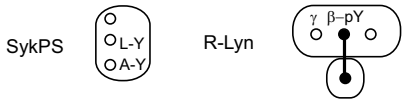


Figure 6: *Examples of uniquely defined species: SykPS species consists of a single molecule, and R-Lyn is a complex of two molecules. Use of id is optional only if required for bonds declaration.*

Each multicomponent `species` is declared by

- `listOfMoleculesIncluded` that specifies `molecules` and `components` included into the species.

- If `componentIncluded` is used in `bonds`, it must be assigned a unique id within species declaration.
- Each `componentState` and `componentConnectivity` must be fully specified.

- `listOfBonds` specifies bonds between these `componentIncluded` (if there are more than one `componentIncluded`).
  - `bond` connects only two `components`. These components must be in specific `componentState`, and have an attribute `componentConnectivity="bound"`.
  - `component` can have only one bond at a time (it corresponds to a biological definition that no more than one molecule can dock to a docking site at a time).

Figure 7 demonstrates a complex species both graphically and as XML description.

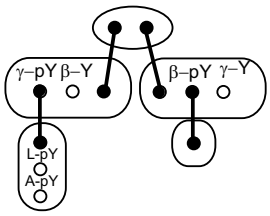


Figure 7: *Example of species, consisting of a receptor-dimer with one receptor associated with Lyn protein, and the second bound to Syk protein phosphorylated in both linker region and activation loop.*

```
<species id="D-S-L" initialAmount="10">
      <listOfMoleculesIncluded>
      <moleculeIncluded Molecule="Syk" id="Syk1">
           <listOfComponentsIncluded>
           <componentIncluded component="SH2" id="SH2" componentConnectivity="bound"/>
           <componentIncluded component="L" componentState="pY" componentConnectivity="unbound"/>
           <componentIncluded component="A" componentState="pY" componentConnectivity="unbound"/>
           </listOfComponentsIncluded>
      </moleculeIncluded>
      <moleculeIncluded molecule="Lyn" id="Lyn1">
            <listOfComponentsIncluded>
            <componentIncluded component="SH2" id="LSH2" componentConnectivity="bound"/>
            </listOfComponentsIncluded>
      </moleculeIncluded>
      <moleculeIncluded molecule="Lig" id="Ligand">
            <listOfComponentsIncluded>
            <componentIncluded component="Fc" id="Fc1" componentConnectivity="bound"/>
            <componentIncluded component="Fc" id="Fc2" componentConnectivity="bound"/>
            </listOfComponentsIncluded>
      </moleculeIncluded>
      <moleculeIncluded molecule="FceRI" id="R1">
            <listOfComponentsIncluded>
            <componentIncluded component="alpha" id="alpha1" componentConnectivity="bound"/>
            <componentIncluded component="beta" componentState="Y" componentConnectivity="unbound"/>
            <componentIncluded component="gamma" id="gamma1" componentState="pY"
                                                    componentConnectivity="bound"/>
            </listOfComponentsIncluded>
      </moleculeIncluded>
      <moleculeIncluded molecule="FceRI" id="R2">
            <listOfComponentsIncluded>
            <componentIncluded component="alpha" id="alpha2" componentConnectivity="bound"/>
            <componentIncluded component="beta"  id="beta2" componentState="pY"
                                                    componentConnectivity="bound"/>
            <componentIncluded component="gamma" componentState="Y" componentConnectivity="unbound"/>
            </listOfComponentsIncluded>
      </moleculeIncluded>
      </listOfMoleculesIncluded>

      <listOfBonds>
      <bond id="R1-syk">
            <listOfComponentsConnected>
            <componentConnected="gamma1"/>
            <componentConnected="LSH2"/>
            </listOfComponentsConnected>
      </bond>
```

```
      <bond id="R2-lyn">
            <listOfComponentsConnected>
            <componentConnected="beta2"/>
            <componentConnected="LSH2"/>
            </listOfComponentsConnected>
      </bond>
      <bond id="R1-lig">
            <listOfComponentsConnected>
            <componentConnected="Fc1"/>
            <componentConnected="alpha1"/>
            </listOfComponentsConnected>
      </bond>
      <bond id="R2-lig">
            <listOfComponentsConnected>
            <componentConnected="Fc2"/>
            <componentConnected="alpha2"/>
            </listOfComponentsConnected>
       </bond>
       </listOfBonds>
</species>
```

3.4 Groups of species

Groups of chemical species with specified shared features can be seen as graphs that do not completely specify `componentStates` and `componentConnectivity`. The `bonds`, `componentConnectivity` and `componentStates` that are specified define the distinguishing features of a group.

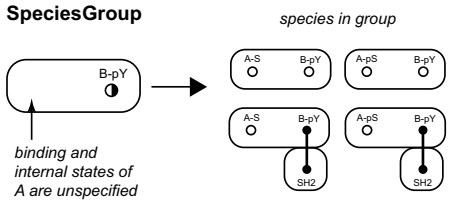


Figure 8: *An example of `speciesGroup` and a set of chemical species that belong to this group.*

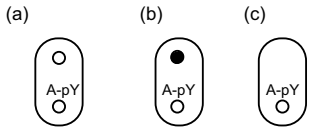
For a molecule "A" specified in Figure 4, Figure 8 demonstrates an example of `speciesGroup` along with some of chemical species containing in this group. In general, given a set of chemical species, `speciesGroup` consists of all chemical species among the set in which component B of the indicated molecule is in state pY. Because the internal state of component "A" and the connectivity of "B" are unspecified in the rule, chemical species selected by the rule can have different states of "A" and different bound states of "B" as shown. Additional species, depending of the set of species being tested, could belong to the group, as would be the case if component "A" was attached to a binding partner in one of the possible species.

Each `speciesGroup` is declared similarly to `species`, but there is no requirement for each component to be fully specified.

- `listOfMoleculesIncluded` specifies `molecules` and `components` included into the species.
  - If `componentIncluded` is used in `bonds`, it must be assigned a unique id within species declaration.
  - If `componentState` is not specified, then all the states listed in the `listOfComponents` can be realized, leading to multiple chemical species.
  - If `componentConnectivity` is not specified, `speciesGroup` includes species that contain molecules connected to this component, as in Figure 8.
- `listOfBonds` specifies bonds between `moleculeIncluded` (if there are more than one `moleculeIncluded`).
  - `bond` connects only two `components`. These components can be in specific `componentState` or in a non-specified state.

– `component` can have only one bond at a time.

Figures 9 and 10 demonstrate `speciesGroups` that include Syk molecule with phosphorylated "A" component in different molecular complexes.

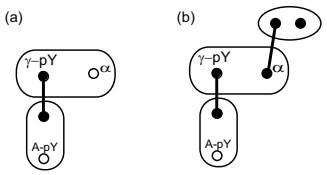



```
<speciesGroup id="Free-SykPS">
      <listOfMoleculesIncluded>
      <moleculeIncluded molecule="Syk">
            <listOfComponentsIncluded>
            <componentIncluded component="SH2" componentConnectivity="unbound"/>
            <componentIncluded component="A" componentState="pY" componentConnectivity="unbound"/>
            </listOfComponentsIncluded>
      </moleculeIncluded>
      </listOfMoleculesIncluded>
</speciesGroup>

<speciesGroup id="Aggr-SykPS">
      <listOfMoleculesIncluded>
      <moleculeIncluded molecule="Syk">
            <listOfComponentsIncluded>
            <componentIncluded component="SH2" componentConnectivity="bound"/>
            <componentIncluded component="A" componentState="pY" componentConnectivity="unbound"/>
            </listOfComponentsIncluded>
      </moleculeIncluded>
      </listOfMoleculesIncluded>
</speciesGroup>

<speciesGroup id="All-SykPS">
      <listOfMoleculesIncluded>
      <moleculeIncluded molecule="Syk">
            <listOfComponentsIncluded>
            <componentIncluded component="A" componentState="pY" componentConnectivity="unbound"/>
            </listOfComponentsIncluded>
      </moleculeIncluded>
      </listOfMoleculesIncluded>
</speciesGroup>
```

Figure 9: *(a) Free-SykPS group consists of all cytosolic Syk molecules with "A" component phosphorylated. This group consists of two species: with L component being either unphosphorylated or phosphorylated. (b) Aggr-SykPS group consists of all species that include a Syk molecule bound to FcεRI with "A" component phosphorylated. This group consists of 208 species: 16 FcεRI receptor-monomers with aggregated Syk molecule, and 192 complexes of FcεRI receptor-dimers. (c) All-SykPS group consists of all species that include a Syk molecule with "A" component phosphorylated. This group consists of 210 species. SH2 component is not specified, as its connectivity is unspecified.*




```
<speciesGroup id="SykPS-Monomer">
      <listOfMoleculesIncluded>
      <moleculeIncluded Molecule="Syk" id="Syk1">
           <listOfComponentsIncluded>
           <componentIncluded component="SH2" id="SSH2" componentConnectivity="bound"/>
           <componentIncluded component="A" componentState="pY"/>
           </listOfComponentsIncluded>
      </moleculeIncluded>
      <moleculeIncluded molecule="FceRI" id="R1">
            <listOfComponentsIncluded>
            <componentIncluded component="alpha" componentConnectivity="unbound"/>
            <componentIncluded component="gamma" id="gamma1" componentState="pY"
                                                        componentConnectivity="bound"/>
            </listOfComponentsIncluded>
      </moleculeIncluded>
      </listOfMoleculesIncluded>
      <listOfBonds>
      <bond id="R1-syk">
            <listOfComponentsConnected>
            <componentsConnected component="gamma1"/>
            <componentsConnected component="SSH2"/>
            </listOfComponentsConnected>
      </bond>
      </listOfBonds>
</speciesGroup>

<speciesGroup id="SykPS-Dimer">
      <listOfMoleculesIncluded>
      <moleculeIncluded Molecule="Syk">
           <listOfComponentsIncluded>
           <componentIncluded component="SH2" id="SSH2" componentConnectivity="bound"/>
           <componentIncluded component="A" componentState="pY"/>
           </listOfComponentsIncluded>
      </moleculeIncluded>
      <moleculeIncluded molecule="Lig">
            <listOfComponentsIncluded>
            <componentIncluded component="Fc" id="Fc1" componentConnectivity="bound"/>
            <componentIncluded component="Fc" componentConnectivity="bound"/>
            </listOfComponentsIncluded>
      </moleculeIncluded>
      <moleculeIncluded molecule="FceRI">
            <listOfComponentsIncluded>
            <componentIncluded component="alpha" id="alpha1" componentConnectivity="bound"/>
            <componentIncluded component="gamma" id="gamma1" componentState="pY"
                                                        componentConnectivity="bound"/>
            </listOfComponentsIncluded>
      </moleculeIncluded>
      <listOfBonds>
      <bond id="R1-syk">
            <listOfComponentsConnected>
            <componentsConnected component="gamma1"/>
            <componentsConnected component="SSH2"/>
            </listOfComponentsConnected>
      </bond>
      <bond id="R1-Lig">
            <listOfComponentsConnected>
            <componentsConnected component="alpha1"/>
            <componentsConnected component="Fc1"/>
            </listOfComponentsConnected>
      </bond>
      </listOfBonds>
</speciesGroup>
```

Figure 10: *(a) `speciesGroup` that includes Syk molecule with "A" component phosphorylated bound to phosphorylated gamma component of a receptor-monomer. This `speciesGroup` consists of 16 species: combinations of Syk molecule in 4 different states, FcεRI molecule in 2 different states, and Lyn that can be bound in 2 different ways (to free and phosphorylated beta component) (b) `speciesGroup` that includes Syk molecule bound to phosphorylated gamma component of a receptor-dimer. This group consists of 192 species.*

3.5 Observables

Sometimes, multiple `species` and `speciesGroups` correspond to a specific experimental observable. A typical declaration of an `observable` looks as follows:

```
<observable id="Ob1">
      <listOfSpecies>
            <species id="S1"/>
            <species id="S2"/>
      </listOfSpecies>
      <listOfSpeciesGroups>
            <speciesGroup id="SG1"/>
            <speciesGroup id="SG2"/>
      </listOfSpeciesGroups>
</observable>
```

Figure 11 provides an example of an observable that includes all FcεRI molecules in a system phosphorylated on at least one site:

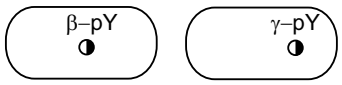


Figure 11: *Observable that includes all FcεRI molecules in a system phosphorylated on at least one site*

```
<observable id="Receptor-phosph">
   <listOfSpeciesGroups>
      <speciesGroup id="Receptor-beta-phosph">
         <listOfMoleculesIncluded>
            <moleculeIncluded molecule="FceRI">
               <listOfComponentsIncluded>
                  <componentIncluded component="beta" componentState="pY"/>
               </listOfComponentsIncluded>
            </moleculeIncluded>
         </listOfMoleculesIncluded>
      </speciesGroup>
      <speciesGroup id="Receptor-gamma-phosph">
         <listOfMoleculesIncluded>
            <moleculeIncluded molecule="FceRI">
               <listOfComponentsIncluded>
                  <componentIncluded component="gamma" componentState="pY"/>
               </listOfComponentsIncluded>
            </moleculeIncluded>
         </listOfMoleculesIncluded>
      </speciesGroup>
   </listOfSpeciesGroups>
</observable>
```

If `speciesGroup id="Receptor-beta-phosph"` was specified elsewhere, the same group can be declared as

```
<observable id="Receptor-phosph">
   <listOfSpeciesGroups>
      <speciesGroupReference speciesGroup="Receptor-beta-phosph"/>
   </listOfSpeciesGroups>
</observable>
```

3.6 Reactions and reaction rules

In the current SBML level 2 `reaction` is specified by `listOfReactants`, `listOfProducts`, and `kineticLaw`. Reactants and products are referenced to `species`.

```
<reaction id="Reac1" reversible="false">
      <listOfReactants>
            <speciesReference species="S1"/>
      </listOfReactants>
      <listOfProducts>
            <speciesReference species="S2"/>
      </listOfProducts>
      <kineticLaw formula="10*S1"/>
</reaction>
```

In the proposed XML scheme, in addition to reactions we introduce reaction rules, that reference to `speciesGroups`:

```
<reactionRule id="ReacRule1" reversible="false">
      <listOfReactants>
            <speciesGroupReference speciesGroup="SG1"/>
      </listOfReactants>
      <listOfProducts>
            <speciesGroupReference speciesGroup="SG2"/>
      </listOfProducts>
      <kineticLaw formula="10*SG1"/>
</reactionRule>
```

Reaction rules are used to generate chemical reactions from a list of chemical species by identifying sets of reactants and products. Each `reactionRule` is comprised a `speciesGroup` representing reactants, a `speciesGroup` representing products, and a kinetic law.

Applying a `reactionRule` to a set of chemical species consists of the following steps:

- identify the group of species corresponding to each reactant `speciesGroup`, as described in section 3.4;
- for each combination of reactant species drawn from these groups, the rule is applied by replacing `componentStates`, `componentConnectivities` and `bonds` in reactant species with the corresponding `componentStates`, `componentConnectivities` and `bonds` in `speciesGroup` of products to define the products. In carrying out this replacement, `componentStates`, `componentConnectivities` and `bonds` that are not specified in the product `speciesGroup` are inherited from reactants.
- the product species are then checked against the current list of chemical species and added to the list if they are not already present. The generated reaction can also be checked against the list of previously generated reactions to prevent duplication of reactions or to identify overlap between rules.

Note that if a reactions' `speciesGroup` consists of a single species, then the containing `reactionRule` will be equivalent to a `reaction`.

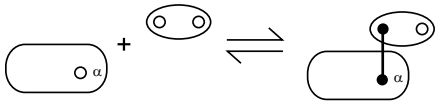
If `speciesGroups` "free-monomer", "bound-ligand" and "dimer" are previously defined, ligand-binding and receptor aggregation reaction may be written in the following form (for clarity of XML examples, we describe only `listOfReactants` and `listOfProducts`; `kineticLaw` is identical to SBML level 2 standard and is omitted):

```
<reactionRule id="Ligand_bind" reversible="true">
      <listOfReactants>
            <speciesGroupReference speciesGroup="free-monomer"/>
            <speciesReference species="ligand"/>
      </listOfReactants>
      <listOfProducts>
            <speciesGroupReference speciesGroup="bound-monomer"/>
      </listOfProducts>
</reactionRule>

<reactionRule id="Dimerization" reversible="true">
      <listOfReactants>
            <speciesGroupReference speciesGroup="free-monomer"/>
            <speciesGroupReference speciesGroup="bound-ligand"/>
      </listOfReactants>

      <listOfProducts>
            <speciesGroupReference speciesGroup="dimer"/>
      </listOfProducts>
</reactionRule>
```

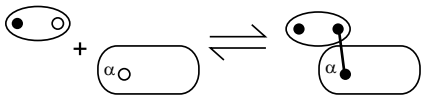
The scheme above may not express in enough detail the effect of a reaction, namely, a correspondence between components of reactants and products. From the content it may be unclear how binding may occur, because binding is specific to a component. Since unique id's for components are not given in the above representation, components corresondence is not defined. The above description works only if each component within `listOfReactants` and `listOfProducts` is unique. Thus, it is necessary to redeclare each `speciesGroup` implicitely with id's of reactants components inherited by products. Figures 12 and 13 illustrates the same reactions of ligand binding and receptor aggregation.



```
<reactionRule id="Ligand_bind" reversible="true">
<!-- 1. Ligand binding-->
   <listOfReactants>
      <speciesGroupReference speciesGroup="free-monomer">
          <listOfMoleculeIncluded>
             <moleculeIncluded molecule="FceRI">
                <listOfComponentsIncluded>
                   <componentIncluded component="alpha" id="alpha1" componentConnectivity="unbound"/>
                </listOfComponentsIncluded>
             </moleculeIncluded>
          </listOfMoleculesIncluded>
      </speciesGroupReference>
      <speciesReference species="ligand">
          <listOfMoleculesIncluded>
             <moleculeIncluded molecule="Lig">
                <listOfComponentsIncluded>
                   <componentIncluded component="Fc" id="Fc1" componentConnectivity="unbound"/>
                   <componentIncluded component="Fc" id="Fc2" componentConnectivity="unbound"/>
                </listOfComponentsIncluded>
             </moleculeIncluded>
          </listOfMoleculesIncluded>
      </speciesReference>
   </listOfReactants>
   <listOfProducts>
      <speciesGroupReference speciesGroup="bound-monomer">
         <listOfMoleculesIncluded>
           <moleculeIncluded molecule="FceRI">
             <listOfComponentsIncluded>
                <componentIncluded component="alpha" id="alpha1" componentConnectivity="bound"/>
             </listOfComponentsIncluded>
           </moleculeIncluded>
           <moleculeIncluded molecule="Lig">
             <listOfComponentsIncluded>
                <componentIncluded component="Fc" id="Fc1" componentConnectivity="bound"/>
                <componentIncluded component="Fc" id="Fc2" componentConnectivity="unbound"/>
             </listOfComponentsIncluded>
           </moleculeIncluded>
         </listOfMoleculesIncluded>

         <listOfBonds>
           <bond id="R1-Lig">
             <listOfComponentsConnected>
                <componentsConnected component="alpha1"/>
                <componentsConnected component="Fc1"/>
             </listOfComponentsConnected>
           </bond>
         </listOfBonds>
      </speciesGroupReference>
   </listOfProducts>
</reactionRule>
```

Figure 12: *Ligand binding reaction: free ligand binds to FceRI. Species "ligand" consists of a single molecue, but SpeciesGroup "free-monomer" consists of 24 chemical species. Thus, this reaction rule corresponds to 24 reversible chemical reactions.*



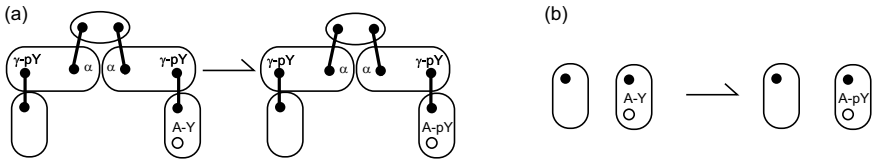

```
<reactionRule id="Dimerization" reversible="true">
<!-- 2. Ligand-induced aggregation-->
   <listOfReactants>
      <speciesGroupReference speciesGroup="free-monomer">
          <listOfMoleculesIncluded>
             <moleculeIncluded molecule="FceRI">
                <listOfComponentsIncluded>
                   <componentIncluded component="alpha" id="alpha1" componentConnectivity="unbound"/>
                </listOfComponentsIncluded>
             </moleculeIncluded>
          </listOfMoleculesIncluded>
      </speciesGroupReference>
      <speciesGroupReference speciesGroup="bound-ligand">
         <listOfMoleculesIncluded>
             <moleculeIncluded molecule="Lig">
                <listOfComponentsIncluded>
                   <componentIncluded component="Fc" id="Fc1" componentConnectivity="unbound"/>
                   <componentIncluded component="Fc" componentConnectivity="bound"/>
                </listOfComponentsIncluded>
             </moleculeIncluded>
          </listOfMoleculesIncluded>
      </speciesGroupReference>
   </listOfReactants>

   <listOfProducts>
      <speciesGroupReference speciesGroup="dimer">
         <listOfMoleculesIncluded>
            <moleculeIncluded molecule="FceRI" id="R1">
                <listOfComponentsIncluded>
                   <componentIncluded component="alpha" id="alpha1" componentConnectivity="bound"/>
                </listOfComponentsIncluded>
             </moleculeIncluded>
             <moleculeIncluded molecule="Lig">
                <listOfComponents>
                   <componentIncluded component="Fc" id="Fc1" componentConnectivity="bound"/>
                   <componentIncluded component="Fc" componentConnectivity="bound"/>
                </listOfComponents>
             </moleculeIncluded>
          </listOfMoleculesIncluded>

          <listOfBonds>
             <bond id="R1-Lig">
                <listOfComponentsConnected>
                   <componentsConnected component="alpha1"/>
                   <componentsConnected component="Fc1"/>
                </listOfComponentsConnected>
             </bond>
          </listOfBonds>
      </speciesGroupReference>
   </listOfProducts>
</reactionRule>
```

Figure 13: *Ligand-induced aggregation of receptors (576 reversible reactions).*

Figure 14 illustrates `reactionRule` for Syk transphoshorylation by Syk. Figure 14(a) illustrates the "real" reaction graph. The reactants are choosen from the list of chemical species generated by reaction rules, and the only possibility of two Syk molecules coexisting within one species is being bound to two different receptors aggregated through a ligand. Therefore Figure 14(b) uniquely defines this `reactionRule`, and the XML description corresponds to it. This simplification may be dangerous, as if other `reactionRules` will be changed, the set of possible species may be different, and two Syk molecules may be in a combination that does not allow for transphosphorylation.

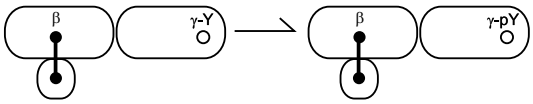



```
<reactionRule id="Syk-Syk-transphosph" reversible="false">
<!-- 8. Transphosphorylation of Syk by Syk-->
    <listOfReactants>
       <speciesGroupReference speciesGroup="Syk-Syk-unphosp">
           <listOfMoleculesIncluded>
              <moleculeIncluded Molecule="Syk" id="Syk1">
                 <listOfComponentsIncluded>
                    <componentIncluded component="SH2" componentConnectivity="bound"/>
                 </listOfComponentsIncluded>
              </moleculeIncluded>
              <moleculeIncluded Molecule="Syk" id="Syk2">
                 <listOfComponentsIncluded>
                    <componentIncluded component="SH2" componentConnectivity="bound"/>
                    <componentIncluded component="A" componentState="Y"/>
                 </listOfComponentsIncluded>
              </moleculeIncluded>
           </listOfMoleculesIncluded>
       </speciesGroupReference>
    </listOfReactants>

    <listOfProducts>
       <speciesGroupReference speciesGroup="Syk-Syk-phosp">
           <listOfMoleculesIncluded>
              <moleculeIncluded Molecule="Syk" id="Syk1">
                 <listOfComponentsIncluded>
                    <componentIncluded component="SH2" componentConnectivity="bound"/>
                 </listOfComponentsIncluded>
              </moleculeIncluded>
              <moleculeIncluded Molecule="Syk" id="Syk2">
                 <listOfComponentsIncluded>
                    <componentIncluded component="SH2"  componentConnectivity="bound"/>
                    <componentIncluded component="A" componentState="pY"/>
                 </listOfComponentsIncluded>
              </moleculeIncluded>
           </listOfMoleculesIncluded>
       </speciesGroupReference>
    </listOfProducts>
 </reactionRule>
```

Figure 14: *Transphosphorylation of Syk by Syk (128 irreversible reactions). (a) The real species graph. (b) The reduced species graph. XML code corresponds to the reduced graph.*

Figure 15 illustrates `reactionRule` for gamma chain transphoshorylation by Lyn. Lyn can be bound to both receptors in a dimer, but can transphosphorylate gamma only on adjacent receptor. Hence, we can not make the same simplification as in the previous Figure. Ligand is not shown because the transphosphorylation can occur any time two receptors are within the same dimer.




```
<reactionRule id="Lyn-gamma-transphosph" reversible="false">
 <!-- 6. Transphosphorylation of gamma chain by Lyn-->
    <listOfReactants>
       <speciesGroupReference speciesGroup="unphosph-gamma-Lyn">
           <listOfMoleculesIncluded>
              <moleculeIncluded Molecule="Lyn">
                 <listOfComponentsIncluded>
                    <componentIncluded component="SH2" id="LSH2" componentConnectivity="bound"/>
                 </listOfComponentsIncluded>
              </moleculeIncluded>
              <moleculeIncluded molecule="FceRI" id="R1">
                 <listOfComponentsIncluded>
                    <componentIncluded component="beta" id="beta1" componentConnectivity="bound"/>
                 </listOfComponentsIncluded>
              </moleculeIncluded>
              <moleculeIncluded molecule="FceRI" id="R2">
                 <listOfComponentsIncluded>
                    <componentIncluded component="gamma" id="gamma2" componentState="Y"
                                                         componentConnectivity="unbound"/>
                 </listOfComponentsIncluded>
              </moleculeIncluded>
           </listOfMoleculesIncluded>
           <listOfBonds>
              <bond id="R1-Lyn">
                 <listOfComponentsConnected>
                    <componentsConnected component="beta1"/>
                    <componentsConnected component="LSH2"/>
                 </listOfComponentsConnected>
              </bond>
           </listOfBonds>
       </speciesGroupReference>
    </listOfReactants>

    <listOfProducts>
       <speciesGroupReference speciesGroup="phosph-gamma-Lyn">
           <listOfMoleculesIncluded>
              <moleculeIncluded Molecule="Lyn">
                 <listOfComponentsIncluded>
                    <componentIncluded component="SH2" id="LSH2" componentConnectivity="bound"/>
                 </listOfComponentsIncluded>
              </moleculeIncluded>
              <moleculeIncluded molecule="FceRI" id="R1">
                 <listOfComponentsIncluded>
                    <componentIncluded component="beta" id="beta1" componentConnectivity="bound"/>
                 </listOfComponentsIncluded>
              </moleculeIncluded>
              <moleculeIncluded molecule="FceRI" id="R2">
                 <listOfComponentsIncluded>
                    <componentIncluded component="gamma" id="gamma2" componentState="gamma-pY"
                                                           componentConnectivity="unbound"/>
                 </listOfComponentsIncluded>
              </moleculeIncluded>
           </listOfMoleculesIncluded>
           <listOfBonds>
              <bond id="R1-Lyn">
                 <listOfComponentsConnected>
                    <componentsConnected component="beta1"/>
                    <componentsConnected component="LSH2"/>
                 </listOfComponentsConnected>
              </bond>
           </listOfBonds>
       </speciesGroupReference>
    </listOfProducts>
 </reactionRule>
```

Figure 15: *Transphosphorylation of gamma chain by Lyn (72 irreversible reactions).*

4 Discussion

4.1 Compartment attribute

`Compartment` attribute is very similar to `componentState`. The difference is that `componentStates` are specific to a molecule, and `compartments` are specified in the `listOfCompartment` for the whole system. Both attributes are optional. If compartments are declared in the `listOfCompartment`, each component must have an attribute `compartment` either specifically declared, or inherited from molecule/species declaration. If `compartment` is specified in molecule/component definition, it can not be changed.

Otherwise, a reaction across compartments may be specified as

```
<reactionRule id="relocation" reversible="true">
      <listOfReactants>
            <speciesGroupReference speciesGroup="A-extracellular">
                <listOfMoleculesIncluded>
                   <moleculeIncluded molecule="A" compartment="extracellular"/>
                </listOfMoleculesIncluded>
            </speciesGroupReference>
      </listOfReactants>
      <listOfProducts>
            <speciesGroupReference speciesGroup="A-intracellular">
                <listOfMoleculesIncluded>
                   <moleculeIncluded molecule="A" compartment="intracellular"/>
                </listOfMoleculesIncluded>
            </speciesGroupReference>
      </listOfProducts>
</reactionRule>
```

In this case each molecule and component in the `listOfSpecies` used to specify the set of seed species must have `compartment` attribute specified.

The same function can be taken by `componentState` attribute. It is illustrated in the following simple example of species "A" trafficking from compartment 1 to compartment 2: a molecule is endowed with a "compartment" component, the state of which indicates the location of the molecule.

```
<molecule id="A">
      <listOfComponents>
      <component id="compartment"/>
            <listOfComponentStates>
            <componentState id="compartment-1"/>
            <componentState id="compartment-2"/>
            </listOfComponentStates>
      </component>
      </listOfComponents>
<molecule>

<reactionRule id="relocation" reversible="true">
      <listOfReactants>
            <speciesGroupReference speciesGroup="A-compartment1">
                  <listOfMoleculesIncluded>
                  <moleculeIncluded molecule="A">
                        <listOfComponentsIncluded>
                        <componentIncluded component="compartment" componentState="1">
                        </listOfComponentsIncluded>
                  </moleculeIncluded>
                  </listOfMoleculesIncluded>
            </speciesGroupReference>
      </listOfReactants>
      <listOfProducts>
            <speciesGroupReference speciesGroup="A-compartment2">
                  <listOfMoleculesIncluded>
                  <moleculeIncluded molecule="A">
                        <listOfComponentsIncluded>
                        <componentIncluded component="compartment" componentState="2">
                        </listOfComponentsIncluded>
                  </moleculeIncluded>
                  </listOfMoleculesIncluded>
            </speciesGroupReference>
      </listOfProducts>
</reactionRule>
```

The attribute `compartment` can be used to track species belonging to the same compartment and to declare `observables`. Here is an example of observable that reports all the species that belong to compartment "extracellular":

```
<observable id="extracellular">
      <listOfSpeciesGroups>
      <speciesGroups>
            <molecule id="Lig" compartment="extracellular"/>
            <molecule id="FcerI">
```

```
                <listOfComponents>
                <component id="alpha" compartment="extracellular"/>
                </listOfComponentStates>
          </molecule>
     </speciesGroups>
     </listOfSpeciesGroups>
</observable>
```

Furthermore, reaction rules can include information about compartment states, such that only species in the same compartment react. A single molecule or a species can bridge multiple compartments, as FcεRI molecule in an example considered. Kinetic laws treat compartment the same way as in SBML level 2, with user to declare explicitely what compartment to use in `reactionRule`.

4.2 Elimination of bonds feature

We may omit an object `bonds` from XML description without loss of functionality, by introducing of an attribute `componentConnected` instead of `componentConnectivity`. `componentConnected` references to the unique name of `component` that is connected to it.

```
<reactionRule id="Ligand_bind1">
     <listOfReactants>
          <speciesGroupReference id="free-monomer">
               <listOfMoleculesIncluded>
               <moleculeIncluded molecule="FceRI" id="R1">
                    <listOfComponentsIncluded>
                    <componentIncluded component="alpha" id="alpha1" componentConnected="unbound"/>
                    </listOfComponentsIncluded>
               </moleculeIncluded>
               </listOfMoleculesIncluded>
          </speciesGroupReference>
          <speciesGroupReference id="ligand">
                <listOfMoleculesIncluded>
                <moleculeIncluded molecule="Lig">
                     <listOfComponentsIncluded>
                     <componentIncluded component="Fc" id="Fc1" componentConnected="unbound"/>
                     <componentIncluded component="Fc" id="Fc2" componentConnected="unbound"/>
                     </listOfComponentsIncluded>
                </moleculeIncluded>
                </listOfMoleculesIncluded>
          </speciesGroupReference>
     </listOfReactants>

     <listOfProducts>
          <speciesReference id="bound-monomer">
               <listOfMoleculesIncluded>
               <moleculeIncluded molecule="Lig">
                    <listOfComponentsIncluded>
                    <componentIncluded component="Fc" id="Fc1" componentConnected="A1"/>
                    <componentIncluded component="Fc" id="Fc2" componentConnected="unbound"/>
                    </listOfComponentsIncluded>
               </moleculeIncluded>
               <moleculeIncluded molecule="FceRI" id="R1">
                    <listOfComponentsIncluded>
                    <componentIncluded component="alpha" id="alpha1" componentConnected="Fc1"/>
                    </listOfComponentsIncluded>
               </moleculeIncluded>
               </listOfMoleculesIncluded>

          </speciesReference>
     </listOfProducts>
</reactionRule>
```

This scheme has an advantage of a shorter form, but we declined to use it. We decided to introduce `bonds` as a separate object and not as an attribute to `component` because of the following reasons:

- `bonds` closely resemble `edges` introduced for graph XML description like GraphML (`http://graphml.graphdrawing.org/`). In these descriptions `edge` connects two nodes in the way similar to `bond` connects two components.
- `bonds` allow for adding additional attributes (e.g. `type="covalent"`).

4.3 Possible extension: logic and range

The current proposal, as well as BioNetGen input, does not require boolean symbols (AND/OR/NOT) or declaration of a range for some states. Meanwhile, it may be useful to introduce these attributes:

```
<speciesGroup id="A" initialAmount="10">
       <listOfMoleculesIncluded>
       <moleculeIncluded molecule="A1" id="a1">
              <listOfComponentsIncluded>
              <componentIncluded component="S1" componentState=NOT "phosph"/>
```

```
                <componentIncluded component="S2" componentState="phosph1" OR "phosph" />
                <componentIncluded component="S3" componentState=["state1",...,"state10"]/>
                </listOfComponentsIncluded>
        </moleculeIncluded>
        </listOfMoleculesIncluded>
</speciesGroup>
```

4.4 Scalability: variable names

We defined the smallest (**components**) and the largest elements (**species**) of XML syntaxis. Controlled vocabularies (CVs) can be used to define all the names of intermediate elements, which are embedded one into another and can be represented in many different ways:

```
species <-- molecules <-- components
species <-- proteins  <-- domains <-- sites <-- components
species <-- proteins  <-- domains <-- sites <-- polypeptide_chains <-- components
species <-- proteins  <-- domains <-- sites <-- polypeptide_chains <-- amino_acids <-- components
```

In addition, the following guidelines for XML format can be implemented:

- Any number of building blocks (molecules, domains and proteins, etc) that form species can be specified.
- **Species** are constructed from building blocks using the same rules as building blocks can be constructed from other building blocks.
- The structure of all building blocks is defined using the same rules, i.e. all building blocks have the same **attributes**.

4.5 Comparisson with the proposal by Finney, 2004 and Le Novere, 2002

This paper is based on the ideas developed in the course of our attempt to model and understand FcεRI signaling. The problem of modeling combinatorial complexity was first mentioned by our group in 1997 (Wofsy et al., 1997). Our modeling approach was reported by Goldstein et al. (2002), reviewed by Hlavacek et al. (2003) and Goldstein et al. (2004), and implemented in BioNetGen software (Blinov et al., 2004).

The starting point for this writing was SBML level 3 proposals by (Finney, 2004) and (Le Novere, 2002). Here, we discuss the principal differences between the proposals.

When comparing our proposal with the proposal by (Le Novere, 2002), one can see that multistate definitions are similar. The main differences are summarized below:

- The current proposal works with multi-component species constructed from several molecules, while proposal by (Le Novere, 2002) defines multi-state species, but can not be easily used to specify multi-molecular species.
- Proposal by (Le Novere, 2002) assumes that each species should be manually specified. Our proposal assumes that SBML format specifies key features of the system, then software will generate the list of actual species.

When comparing our proposal with the proposal by Finney (2004), one can see that bond definitions are similar. The main differences are summarized below:

- The current proposal is based on a graph representation explicitely declared.
- The current proposal works with molecular components as basic units, and proposal by (Finney, 2004) introduces **bindingComponent** as a secondary tool to help in designing multi-component species.
- **bindingSite** in (Finney, 2004) is restricted to binding only and may not be in several states. Thus, it can not represent, for instance, a kinase domain. Contrary to that, **component** in this proposal is a generic component of **molecule**, that can undergo modifications, be in several states, and can be located in different compartments.
- **speciesGroup** structure in the current proposal can represent either a single uniquely-defined species, or a set of species consisting of given **molecules** connected by **bonds**.
- **specieType** and **species** in (Finney, 2004) can not be be stretched across multiple compartments, as in our proposal with components located in different compartments
- A set of species (**observables**) is a completely new feature that was introduced when BioNetGen was developed. It represents observable features of the model.

5 Acknowledgement

This paper has benefited from discussions with Paul Loriaux, Andrew Finney, Mike Hucka, Matthew Fricke, Hamid Bolouri, Eric Mjolsness, and Andre Levchenko.

Support for the development of this proposal came from the National Institutes of Health and the Department of Energy.

We thank Mike Hucka for TeX templates that were used to create this document, and Andrew Finney for commenting the first version of this document.

6 References

A. Arkin. (2001). Synthetic cell biology. Curr Opinion Biotechnol 12:638–644.

M. L. Blinov, J. R. Faeder, W. S. Hlavacek. (2003). BioNetGen: Software for generating mathematical/computational models that account comprehensively and precisely for the full spectrum of molecular species implied by user-specified activities, potential modifications and interactions of the domains of signaling molecules. Available via the World Wide Web at `http://cellsignaling.lanl.gov/bionetgen`

M. L. Blinov, J. R. Faeder, B. Goldstein, W. S. Hlavacek. (2004). BioNetGen: software for rule-based modeling of signal transduction based on the interactions of molecular domains. Bioinformatics (in press

D. Bray. (2003). Molecular prodigality. Science 299:1189–1190

D. Endy, R. Brent. (2001). Modelling cellular behaviour. Nature 409:391–395

J. R. Faeder, M. L. Blinov, and W. S. Hlavacek. (2004). Graphical Rule-Based Representation of Signal-Transduction Networks. Technical Report LA-UR 04-6182.

J. R. Faeder, W. S. Hlavacek, I. Reischl, M. L. Blinov, H. Metzger, A. Redondo, C. Wofsy, and B. Goldstein. (2003). Investigation of early events in FceRI-mediated signaling using a detailed mathematical model. J. Immunol. 170: 3769-81

A. Finney. (2004). Systems Biology Markup Language (SBML) Level 3 Proposal: Multi-component Species Features. Available via the World Wide Web at `http://sbml.org`

A. Finney, M. Hucka, and H. Bolouri. (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/`.

R. Gauges, U. Rost, S. Sahle and K. Wegner. (2004). Including Layout Information in SBML Files Version 2.1 Available via the World Wide Web at
`http://projects.eml.org/bcb/sbml/level2/20040630/SBMLLayoutExtension-20040630.pdf`

B. Goldstein, J. R. Faeder, W. S. Hlavacek, M. L. Blinov, A. Redondo, and C. Wofsy. (2002). Modeling the early signaling events mediated by aggregation of FcϵRI. Mol Immunol. 38: 1213-1219.

B. Goldstein, J. R. Faeder and W. S. Hlavacek. (2004). Mathematical and computational Models of Immune-Receeptor Signaling. Nat. Rev. Immunol. 4, 445-456.

W. S. Hlavacek, J. R. Faeder, M. L. Blinov, A. S. Perelson, B. Goldstein. (2003). The complexity of complexes in signal transduction. Biotechnol. Bioeng. 84: 783-794.

N. Le Novere, T. S. Shimuzu, A. Finney. (2002). Systems Biology Markup Language (SBML) Level 3 Proposal: Multi-state Features. Available via the World Wide Web at `http://sbml.org`

T. Pawson and P. Nash (2003). Assembly of cell regulatory systems through protein interaction domains. Science 300:445 452.

C. Wofsy, C. Torigoe, U. Kent, H. Metzger, Byron Goldstein. (1997). Exploiting the difference between intrinsic and extrinsic kinases: Implications for regulation of signaling by immunoreceptors. Journal of Immunology 159: 5984-5992

7 Appendix

A SBML Level 3 encoding of FcεRI model

We demonstrate two equivalent encodings for FcεRI model described by Faeder et al. (2003). The one in Figure 16(a) illustrates full descriptions of reactants and products in reactionRule, independent on other reactionRule. Changing a single `reactionRule` wil not affect other `reactionRules`. The `listOfReactionRules` in Figure 16(b) illustrates simplified reaction rules, that work together to represent the same system. Both encodings are available at `http://cellsignaling.lanl.gov/sbml/`. For simplicity, kinetic laws are omitted. Models are validated against XML.

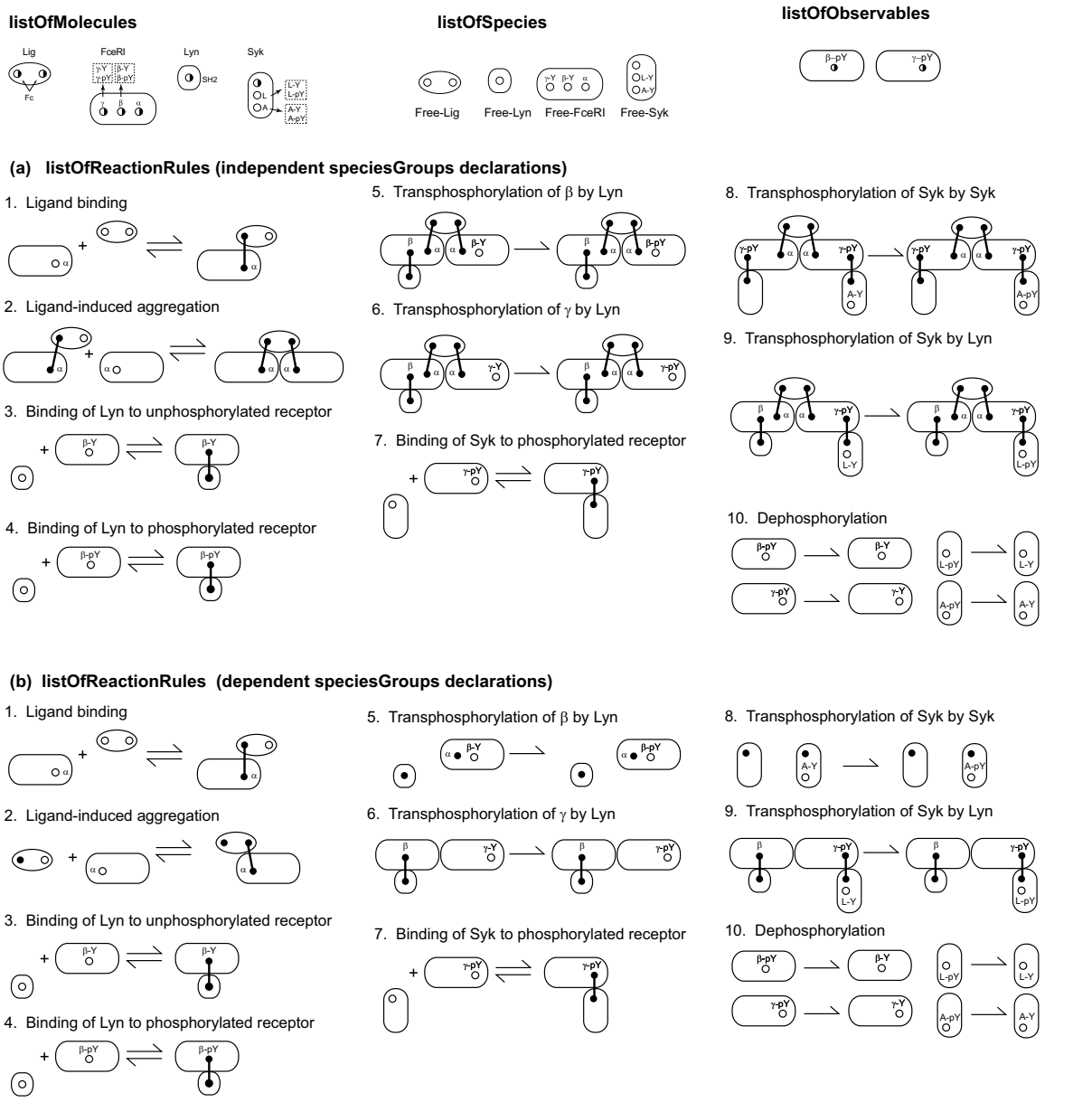


Figure 16: *Graphical representation of FcεRI model. (a) Full speciesGroups descriptions in reactionRules (b) SpeciesGroups declaration that take into account a pool of species generated by other reactionRules.*

B Related software development

We put this proposal together as part of our effort to extend the capabilities of our BioNetGen software, specifically to define XML data structures that could be used in future versions. These proposed data structures are now based on a graphical scheme for representing chemical species and groups of chemical species, which is presented in the LA-UR-04-6182 technical report. A second report, now in preparation, will describe iterative procedures by which the graphical representation of a system can be interpreted to obtain a mathematical model starting from a set of reaction rules (or generalized reactions as you call them) and a seed set of chemical species to which the rules are initially applied. In the future, the steps of model generation will involve graph rewriting and solving problems of graph and subgraph isomorphism. The use of graphs to represent chemical species and groups of chemical species and the formulation of reaction rules as graph rewriting rules will allow one to model systems that cannot be modeled now with BioNetGen and to specify models, using a GUI in development, in a way that is both more comprehensible for biologists and more powerful for modelers, e.g., by making model specification more automatic. The main advance is that the connectivity of molecules in a complex is now represented systematically and explicitly. The string-based method of representation used in the current version of BioNetGen can be viewed as a special case of the new graphical method of representation.

In publically released BioNetGen software (Binov et al., 2004) all the species in a model should be enumerated explicitly prior to reactions declaration. In the proposed format, as well as in the latest BioNetGen version, species can be generated using reaction rules. In particular, formation of any number of complexes of specified topology, including infinite receptor chains, is possible, and controlled only by simulation parameters.

BioNetGen input file is intended to be written manually. The XML description described here is intended to be clear enough to allow for manual editing and analysis, but is not intended to be written by hands. It is supposed to serve as the internal data structure for models created with a Graphical User Interface (GUI). The corresponding graphical representation illustrated in this paper will serve as a GUI for BioNetGen software currently under development. Information about graphical representation (positions of components etc) will be stored either within this XML as additional tags, or in a satellite XML file, and is not demonstrated here.

BioNetGen input file is a single source of all the information about the system and about simulation parameters. In the proposed format we assume that a list of simulation parameters for the software does not need to be incorporated into this data structure, but can be implemented in software. For example, to deal with possible infinite chains a software should be able to truncate chains up to a certain length etc.

C Processing SBML files by current BioNetGen software

Currently, BioNetGen inputs text file (see `http://cellsignaling.lanl.gov/bionetgen/`) and may output files in SBML level 2 format. FcεRI receptor complex in this format is described as:

```
$MultiStateMolecules=<<"END_MULTISTATEMOLECULES";

R1     2,4,6    # number of components and their states

#-------------------- Component 1 (alpha) can be in 2 states----------
#  0 - FreeExt
#  1 - BoundExt
#-------------------- Component 2 (beta) can be in 4 states----------
#  0 - Free
#  1 - Lyn bound constitutively
#  2 - Phosphorylated, no Lyn
#  3 - Phosphorylated, recruited Lyn
#-------------------- Component 3 (gamma) can be in 6 states----------
#  0 - Free
#  1 - Phosphorylated, no Syk
#  2 - Phosphorylated, Syk bound
#  3 - Phosphorylated, SykPL bound
#  4 - Phosphorylated, SykPS bound
#  5 - Phosphorylated, SykPLPS bound

END_MULTISTATEMOLECULES
#------------------Complex formation--------------

$complexes=<<"END_COMPLEXES";
```

```
R1(1,*,*).R1(1,*,*)           # dimers R1

END_COMPLEXES
```

This format does not handle bonds between components, but only between molecules. Currently, all BioNetGen input files can be written in the restricted version of the proposed SBML format. Namely, instead of separate declaration of molecules and their components as in Figure 5, we declare FcεRI molecule with `componentStates` that include association of molecules.

```
<molecule id="FcerI">
      <listOfComponents>
      <component id="alpha" compartment="extracellular">
            listOfComponentStates>
            <componentState id="free" />
            <componentState id="bound"/>
            </listOfComponentStates>
      <component id="beta" compartment="membrane">
            <listOfComponentStates>
            <componentState id="beta-Y" />
            <componentState id="beta-Y-Lyn"/>
            <componentState id="beta-pY"/>
            <componentState id="beta-pY-Lyn"/>
            </listOfComponentStates>
      </component>
      <component id="gamma" compartment="extracellular">
            <listOfComponentStates>
            <componentState id="gamma-Y"  />
            <componentState id="gamma-pY"/>
            <componentState id="gamma-pY-Syk"/>
            <componentState id="gamma-pY-Syk-PS"/>
            <componentState id="gamma-pY-Syk-PL"/>
            <componentState id="gamma-pY-Syk-PS-PL"/>
            </listOfComponentStates>
      </component>
      </listOfComponents>
</molecule>
```

We are actively working on the next version of BioNetGen, which will be able to interpret all the features included in the proposed Level 3 standards. With a little time (once we agree on the standard), we will add a parser from SBML level 3 to BioNetGen, which already outputs SBML Level 2.