A STUDY ON SYSTEMS MODELING FRAMEWORKS AND THEIR INTEROPERABILITY

by

José-Juan Tapia Valenzuela

M.S. in Computer Science, Monterrey Institute of Technology, 2009

Submitted to the Graduate Faculty of

the Department of Computational and Systems Biology in partial fulfillment

of the requirements for the degree of

Ph.D. in Computational Biology

University of Pittsburgh

2016

UNIVERSITY OF PITTSBURGH

DEPARTMENT OF COMPUTATIONAL AND SYSTEMS BIOLOGY. SCHOOL OF MEDICINE, UNIVERSITY OF PITTSBURGH

This dissertation was presented

by

José-Juan Tapia Valenzuela

It was defended on

June 14, 2016

and approved by

James R. Faeder, Ph.D., Associate Professor

Daniel M. Zuckerman, Ph.D., Associate Professor

Nathan L. Clark, Ph.D., Assistant Professor

Russell S. Schwartz, Ph.D., Associate Professor

Michael L. Blinov, Ph.D., Assistant Professor

Dissertation Advisor: James R. Faeder, Ph.D., Associate Professor

A STUDY ON SYSTEMS MODELING FRAMEWORKS AND THEIR INTEROPERABILITY

José-Juan Tapia Valenzuela, PhD

University of Pittsburgh, 2016

What is in a model? is a question that any systems modeler has asked at least once when working with a biological system. To answer this question we must understand how every species and reaction in a model are biologically related to each other, which in the case of cellular systems requires an understanding of local protein and gene interactions at the domain level. Typical modeling approaches like reaction-networks do not explicitly encode this information, which can obfuscate the assumptions made by a model and limit their further analysis, comparison and re-use. Additionally, the multi-state nature of the elements in a model can lead to a combinatorial explosion of the interactions in the model, pushing reaction-networks to their limits. Rule-based modeling is a modeling paradigm that addresses these issues through the use of a graph-based representation of each species's interaction domains and their local interactions. In this paradigm, reaction *rules* describe a class of reactions and are represented as graph rewriting operations that encode biological processes like bond formation and post-translational modification. This allows for a concise, explicit encoding of how species interact, modify and are related to each other while at the same time managing the enormous number of potential molecular interactions in a multi-state system.

In this work I present two different approaches that aim to bring the benefits of rule-based modeling's graph representation to other modeling paradigms: *Atomizer* and *MCell-R*. Atomizer is an algorithm for the extraction of structural and process information from reaction-network models, which I use to encode the model in a rule-based format. Once we have an explicit understanding of what is in a model and how entities in a model interact, we can then perform meta-modeling operations like model analysis, alignment, comparison and visualization, which I demonstrate through

the application of Atomizer to a large dataset of reaction network models. MCell-R is a framework for the efficient modeling and simulation of multi-state, multi-component spatial systems. The framework consists of an integration of the NFsim rule-based simulation engine together with the MCell spatial modeling system, which highlights the utility of bringing rule-based paradigms into reaction based platforms.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1.0 MODELING OF BIOCHEMICAL SYSTEMS

A model is an abstraction of reality in which we extract a set of elements of interest to generate a simplified understanding about a system. Models like those used in mathematics formalize this process by explicitly defining the assumptions that the model makes and the set of conclusions that can be extracted from them. This approach is helpful for understanding a system, communicating this understanding, validating our understanding against experimental data and making predictions about reality [1]. Modeling in cell biology in particular is often used to enhance our understanding of system behavior by starting from known causalities and deriving the biological and mathematical circuitry that explains this information. Deeper understanding of concepts such as homeostasis, feedback, noise and their relationship to cell biology have come from biological systems models [2].

This chapter introduces several modeling frameworks and formalisms for the representation of biochemical systems. In Section 1.1 I introduce the terminology, techniques and formats used to model biochemical reaction-networks. In Section 1.2 I introduce Rule-based modeling as a way to deal with some of the limitations present in the encoding of reaction-networks. Next, I present several simulation approaches to systems models in Section 1.3. Finally, in Section 1.4 I talk about the consideration of spatial aspects in biochemical models.

1.1 REACTION-NETWORK MODELS

A *reaction-network* is a well-studied formalism for the kinetic modeling of biochemical systems. In its most basic form a Reaction-Network Model (RNM) is composed of a set of *biochemical species*, a set of *reactions* that describe possible transformation pathways between those species

and a set of *reaction rates*. The total set of reactions is known as a *reaction network*. An individual reaction can be exemplified as the mapping 1.1 where a set of species in the left hand side defined as the *reactants* is transformed into the set of species in the right hand side called the *products*. Finally, k_1 is the reaction rate at which the reactants transform into products.

$$< reactants > \xrightarrow{k_1} < products > \tag{1.1}$$

Reaction-network models present a disadvantage in the fact that the existence of all possible species and reactions must be known at modeling time. However, the simplicity of its representation makes it a suitable option for smaller to medium models (fewer than a few hundred reactions [3]). For example, a preliminary survey reveals that neurotransmission, pharmacokinetic and cell metabolism models in general have less than 50 reactions and species.

There exist several large repositories that aggregate reaction-network models published in the literature [4, 5]. At the same time there is also a large collection of modeling tools that allow users to define a system using RNM's. Some example of these tools are COPASI [6], CellDesigner [7] and Tellerium [8].

1.1.1 Encoding of RNM's: The Systems Biology Markup Language (SBML)

As the body of modeling literature has expanded, so has the need to have a unified set of modeling semantics and a standard exchange format that facilitates knowledge communication between systems modelers. In 2000, a joint initiative group was formed between several modeling groups to address such a need, and thus developed an XML-based machine readable modeling standard known as the *Systems Biology Markup Language* (SBML [9]). The SBML standard was designed to be a framework for the communication of all kinds of modeling information while being agnostic to the underlying software and conceptual frameworks, including but not limited to ODE-based reaction network models (e.g. signaling networks, BMD19 [10]), conductance-based models (e.g. Hodgin-Huxley, BMD20 [11]), neural models (e.g. spiking networks, BMD127 [12]), pharmacokinetic models (e.g. BMD234 [13]) and infectious disease modeling (e.g. MODEL1008060001 [14]).

There is a large collection of SBML models in the literature. In particular, the BioModels database [15] is an aggregation service containing thousands of models that can be easily queried using simple keywords, like author or publication information or keywords included by the database curators.

An SBML model is composed of several sections related to the definition of an RNM model, like *parameters*, the total set of *reactions*, *species* and the *initial population* pools in the simulations. Additionally, when a model, reaction or molecule is defined in a SBML document it can optionally include *annotation* information that the modeler decides to include as meta-data, including mapping information to biological databases like UNIPROT [16], REACTOME [17] or KEGG [18].

Annotation information is extremely important when determining the chemical or biological structure of a given species [19]. A reaction in an RNM is strictly a transformation of reactants into products that contains no biological information about the underlying process it is modeling. Annotation information can partially fill this gap by providing descriptors that provide information about the composition of a complex. This is important for higher level operations like model analysis or comparison that require us not only to understand how the elements in a model relate to each other but also how elements can be mapped between models or to real world objects. A drawback of this approach is that this is dependent on the modeler or database-curators to include this information. The BioModels database has a curated subset of partially and fully annotated models, and a much larger subset unannotated models [15]. Unfortunately even the annotated subset often lacks key portions that prevents its usage for the higher level operations I mentioned [20], and studies have shown that current annotation approaches do not provide a full description of a system as required by those tools [21].

1.1.2 Visualization of RNMs

A picture is worth a thousand words is a maxim often said in the context of communication, and it is equally important for the transmission of information between people. As the complexity of a model increases in terms of the number of actors and entities in the system, so does the need to generate high-level, user understandable representations of a model that accurately convey the

information encoded in a mathematical framework. [22].

There is a number of different ways to visually represent the information encoded by a biological model, however at the highest level they can be classified into two groups [23]:

- *Contact maps* provide a molecule-centric visualization of a model where emphasis is on the structural components of the system (represented as nodes in a graph) and binding interactions (represented as edges connecting those nodes).
- *Flow maps* on the other hand are useful for getting a process-centric visualization of a model that emphasize temporal order and signal flow. *Petri-nets* are an example of flow-maps in which models are represented as a directed bipartite graph. In this representation the nodes are 'processes' (the reactions) and 'conditions' (the set of reactants and products). Finally, the directed edges indicate which nodes are pre and postconditions for a given process node.

One of the earliest visualization approaches were the *Molecular Interaction Diagrams* [24]. The diagrams are contact maps that provide a simple visual notation illustrating domains, motifs, covalent modifications, binding and catalysis interactions (Fig. 1).

Another visualization standard, the *Systems Biology Graphical Notation* (SBGN) was developed by the same community that developed SBML as a successor to molecular interaction maps with the goal of making a standard that was more amenable to the needs of the biochemical modeling community. This was done by proposing a set of standard notations and XML-based encoding. Furthermore, recognizing the differing needs of the modeling community it proposed three different substandards, which I illustrate in Fig. 2.

An important thing to note is that, with the exception of process diagrams [7], all of these visualizations diagrams need to be manually generated from the model description. This is less of an issue for smaller models, but the task becomes more and more daunting as the number of agents and events in the system reaches the hundreds or thousands of entities. There has been extensive research on how to automatically generate high level visualizations from a model definition that has been enriched with meta-data that describes the mapping of the model entities to real world objects [25], however an approach that allows the generation of a full view that illustrates the structure and dynamics of the reaction-network remains an elusive target [23].

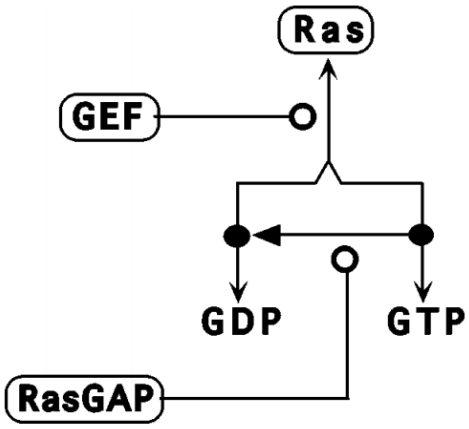


Figure 1: Example of conversions between the GTP and GDP bound states of Ras as illustrated by a Molecular Interaction Map (MIM). The figure shows GDP and GTP competing for a binding site of Ras. GEF facilitates the binding and unbinding processes while $RasGAP$ facilitates the $GTPase$ activity of Ras.

1.1.3 Higher level operations: Analysis, alignment and comparison of Reaction-Network models

The number of existing models in the RNM literature has increased drastically in recent years [4], and so has the size and complexity of each individual model [26]. As such, there is an increased need to generate high-level, user understandable representations of a model that accurately convey the information encoded in a mathematical framework. Furthermore, beyond understanding the contents of a single model, the modeling community is now looking into leveraging the existing collections of models for the study and construction of new modeling information. ([19]). This has encouraged the development of different techniques for the analysis, comparison and aggregation of modeling data.

In this work I introduce the term *model informatics* as a way to refer to the programmatic study of a set of models through a variety of operations like model comparison, querying, alignment and merging [26, 27]. Even though each one of these operations has its own difficulties and requirements, there are some aspects that they have in common: In order to understand *what's in a model*,

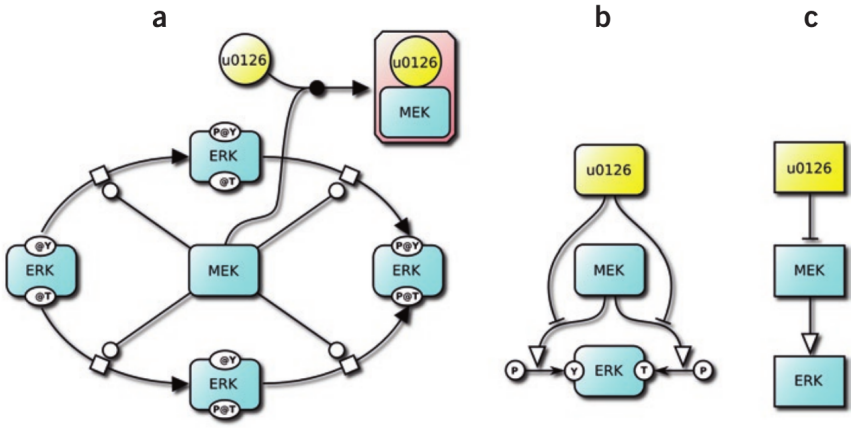


Figure 2: Example of protein phosphorylation catalyzed by an enzyme (image taken from [22]) visualized using the three approaches provided by SBGN. **A:** *Process diagrams* are petri-nets that explicitly display the species state space of the system and the transitions between those states. **B:** The *Entity Relationship* diagram (which fall under the category of contact maps) allow the user to view all the relationships in which an entity participates regardless of the temporal aspects. **C:** An *Activity Flow* diagram depicts the flow of information in a reaction network independent of the state of the system.

we have to determine how elements inside a model are related to each other, how elements between models are related to each other and how elements in a model relate to real world objects [26].

Model analysis, visualization and querying requires careful mapping about the relationship of models to each other. For example, if we wanted to query a model database for all models where Ras has a competitive binding site for GDP and GTP like we saw in Fig. 1, it requires precise user or annotation information that specifies the relationship between these three species. However, the syntax used in reaction-network model specifications does not explicitly allow the user to encode any of this information into the model definition, so modelers use meta-data and annotation information to make up for this limitation.

There is a number of model alignment and comparison tools that have been released by the RNM modeling community like semanticSBML [20] and SemGen [19, 21]. I go into more detail about the specifics of those tools in Chapter 3.

1.1.4 Limitations to Reaction-Network modeling

As we have seen in this section, there are two main limitations intrinsic to reaction-network models. The first has to deal with the fact that the total set of molecular agents and the reaction network that describes their interactions must be known prior to the execution of a model. This is not a problem for smaller models, however for models whose species entities have a multi-state, multi component nature this can lead to combinatorial explosion in the reaction state-space modeled the system. For example, if we considered a single molecule with ten phosphorylation sites, this results in $2^{10} = 1024$ possible states in which the molecule can be, with a proportional number of reactions denoting the transition between those states.

The second limitation refers to the limited semantics and encoding power associated to each individual reaction: it is not possible to attach any kind of meaning to reaction events or species agents without relying on (often insufficient) annotation meta-data, which restricts further static analysis and meta-modeling operations, like model comparison.

1.1.5 Simulation of reaction-network models

Once a model has been specified the user can perform a number of simulation operations that allow us to extract kinetic data from a model representation [28]. One such method is the Gillespie algorithm (SSA) for sampling kinetic trajectories from a discrete stochastic system [29, 30]. Most simulation frameworks that wish to solve a biochemical model while considering stochastic effects in the limit of a well-mixed system implement some form of the Gillespie algorithm [6, 7, 31, 32].

The Gillespie algorithm can be summarized as follows: Given a model definition simulated for an X number of steps, at each simulation step a reaction event is triggered and the reactant populations updated according to the stoichiometry vector specified by the reaction. The time till the next reaction is sampled from an exponential distribution parametrized by the reaction's propensities ($\Delta t = -log(u)/a_{tot}$ where u is a standard uniform random number and a_{tot} is the sum of all propensities). After a reaction is fired the reaction propensities are updated to reflect the new population vector.

Alternatively, in the limit of large concentrations where stochastic effects have less impact a reaction network can be solved as a system of differential equations. For example, given a stoichiometry matrix $S = (s_{i,j})$ and a propensity vector $\overrightarrow{f} = (f_j)_{j=1}^{N_R}$, the set of differential equations that describe the system dynamics can be written as:

$$\frac{d\overrightarrow{x}}{dt} = S\overrightarrow{f}(\overrightarrow{x})$$

As a simple example, consider a model composed of species $\{A, B, C, A\_B, A\_B\_C\}$ and reactions:

$$R1 : A + B \xrightarrow{k_1} A\_B$$
$$R2 : A\_B + C \xrightarrow{k_2} A\_B\_C$$

The stoichiometry matrix for this system is as follows:

	R1	R2
A	-1	0
B	-1	0
C	0	-1
A_B	1	-1
A_B_C	0	1

While the propensity vector would be $[k_1 * [A] * [B], k_2 * [A\_B] * [C]]$. From here, the system of differential equations is:

$$\begin{aligned}
\frac{dA}{dt} &= -k1 * [A] * [B] \\
\frac{dB}{dt} &= -k1 * [A] * [B] \\
\frac{dC}{dt} &= -k2 * [A_B] * [C] \\
\frac{dA_B}{dt} &= k1 * [A] * [B] - k2 * [A_B] * [C] \\
\frac{dA_B_C}{dt} &= k2 * [A_B] * [C]
\end{aligned}$$

This system ODE's can be numerically solved to generate a time trajectory [33] or used together with dynamic analysis tools for numerical analysis of ODE's, for example to perform dynamic stability analysis.

1.2 RULE-BASED MODELS

Rule-based modeling (RBM) is an approach to modeling biochemical kinetics in which biological components like genes and proteins are modeled as multi-state, multi-component objects. Their interactions are governed by rules that specify the context under which these reactions occur [34,

35]. This has the advantage of specifying a model that explicitly keeps track of the biochemical history of how a series of products came to be.

Consider the reaction we used in the previous section: $A+B \xrightarrow{k_1} A\_B$. When we encode species {`A,B,A_B`} as structured objects, this reaction can be represented instead as shown in Fig. 3A, where species `A,B` are structures that individually contain one binding subunit, or *component*. Using this approach, complex `A_B` is explicitly encoded as a complex unit composed of sub-units `A,B`. The basis of rule-based modeling is encoding species as graph objects and reaction rules as graph operations acting on those objects. For example, non-covalent bonding is encoded as an operation involving the creation of a bond between nodes.

Similarly, the multi-state aspect of rule-based modeling allow us to encode a phosphorylation reaction $A\_B \xrightarrow{k_{cat}} A\_P + B$ using a similar approach as shown in Fig. 3B. In this example I show the second step of a Michaelis-Menten process encoded as two graph operations: the deletion of a bond-edge between molecules {`A,B`} and the phosphorylation of `A` encoded as a variable change. This approach has an advantage over traditional RNM representations by being able to clearly encode the conformational changes that a reaction event applies to the set of reactants, and maintaining a clear stoichiometric trail of how products came to be.

Additionally rule-based modeling simplifies the encoding of scalable models by addressing the problem of combinatorial complexity. Proteins interact in a combinatorial number of ways to generate a large number of potential complexes. Rules make use of patterns such that the possible contexts under which a chemical process can occur are expressed in a succinct, simplified way. Consider for example the following example system. There exist the basic molecule types, {`S,A,B,C`} such that substrate molecule `S` can bind to molecules {`A,B,C`} without any cooperativity or competitivity restrictions. The full reaction-network describing this system would be as follows:

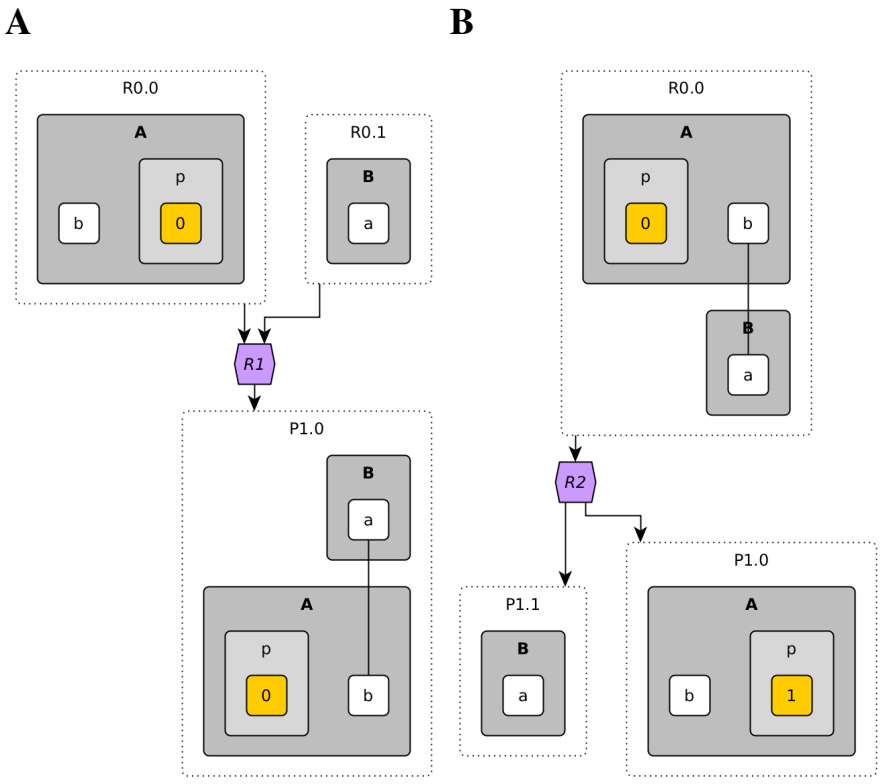


Figure 3: Reaction rules for biological events are encoded as graph operations. A: Covalent bond formation encoded as the addition of a bond between two molecule nodes. Molecules `A`, `B` each have a binding subunit that allows them to form a complex when the reaction fires. B: Phosphorylation of molecule `A` by `B` and non-covalent bond destruction encoded as a state modification and edge deletion operations.

$$
\begin{aligned}
S + A &\rightarrow S\_A \\
S + B &\rightarrow S\_B \\
S + C &\rightarrow S\_C
\end{aligned}
\qquad
\begin{aligned}
S\_A + B &\rightarrow S\_A\_B \\
S\_A + C &\rightarrow S\_A\_C \\
S\_B + A &\rightarrow S\_A\_B \\
S\_B + C &\rightarrow S\_B\_C \\
S\_C + A &\rightarrow S\_A\_C \\
S\_C + B &\rightarrow S\_B\_C
\end{aligned}
\qquad
\begin{aligned}
S\_A\_B + C &\rightarrow S\_A\_B\_C \\
S\_A\_C + B &\rightarrow S\_A\_B\_C \\
S\_B\_C + A &\rightarrow S\_A\_B\_C
\end{aligned}
$$

However this can be done with three simple rules using a rule-based modeling's pattern based approach as shown in Fig 4. This is done by separating the transformation process into two parts: a *reaction center*, which describes the molecular subunits undergoing a conformational change and the *reaction context*, which describes the conditions under which said change takes place but otherwise is not modified. In the case of the rules shown in the figure the reaction center is composed of those components that are involved in the bond formation process, in this example that would be `S` binding to `A,B,C`. The unlisted components in each rule are the implicit reaction context. In the context of RBM's being unlisted means that we do not care about the binding status of those components when considering whether to fire the rule.

A reaction context can also introduce explicit restrictions on the firing conditions of a reaction rule. Consider instead the example in Fig. 3A. In this figure the reaction center is composed of components `A(b)` and `B(a)`, which explicitly require state `A(p)` to be in an unphosphorylated state for the rule to fire.

An RBM specification can be simulated and solved in a number of different ways, like expanding the rule set into the full reaction-network, or using a network-free approach, however these details are abstracted away from the user during model definition such that he only has to specify the system logic [36]. Thus RBM is an effort to make the creation of computational models simpler and accessible to a wider audience. There has been different implementations of RBM in the community, like BioNetGen [37], Kappa [38] and SIMMUNE [39].

In the next section I provide an overview of the terminology used by our reference RBM implementation, BioNetGen, and that we will be using throughout our thesis. For a more formal definition of the terms used in rule-based modeling I refer the reader to [35].

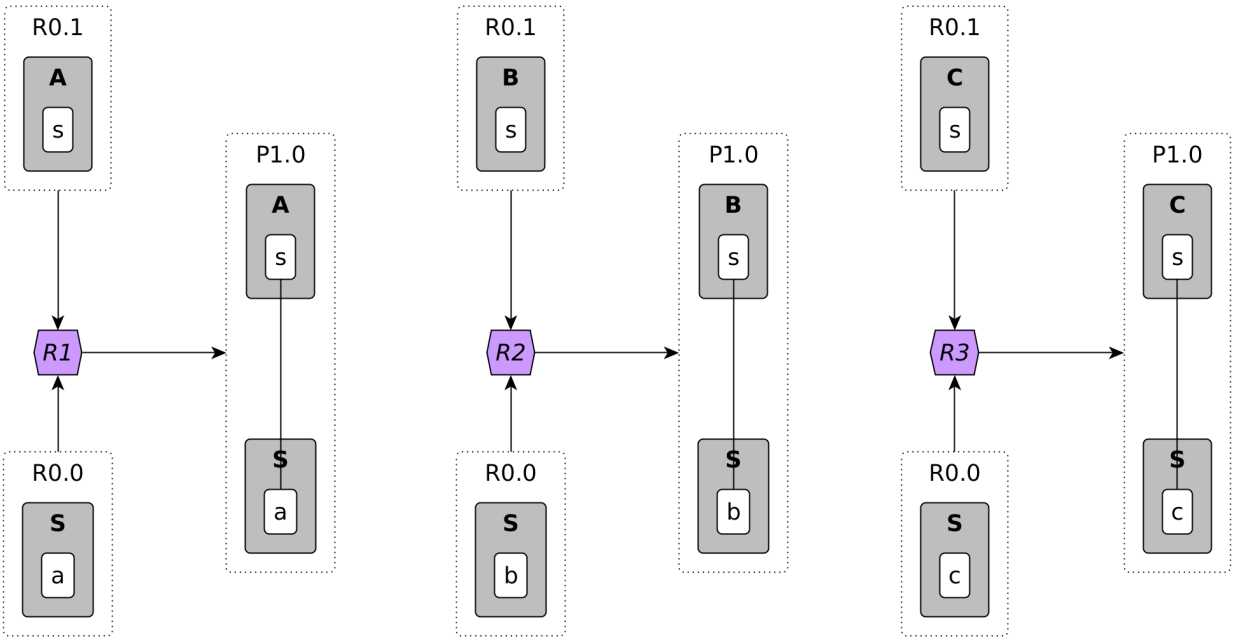


Figure 4: Binding of a substrate molecule to molecules {`A,B,C`}. Rule-based model simplifies model specification by focusing only on the events about which the modeler cares, in this case the binding of substrate to each respective molecule (the reaction center). We couple this with RBM's *don't tell don't care* policy, where the status of molecular components that are not listed within a reaction rule specification are considered as wildcards (the reaction context). In the case of the rules listed in this figure, the bound or unbound status of those components that are not part of the reaction center is irrelevant to the firing of that particular rule event.

1.2.1 BioNetGen: A rule-based modeling framework

In BNGL a "*molecule*" is defined as a single chemical, encoded as a container with multiple internal subunits. For example, A(b ~ P, c, d) is a molecule with three components b, c, d where component "b" is in state "P". In this context, a "*component*" is defined to be a binding subunit, while a "*state*" is a modification flag associated to a molecular subunit. Components and states encode the functional structure of their containing molecule.

The term "*modification*" is used as an umbrella term for several processes like phosphorylation, methylation and others. In BNGL, the ~ symbol is used to express that a component is modified: A(phospo ~ P) means that component phospo is in state P.

A "*species*" is defined as a collection of one or more molecules bound together through its components. In this thesis I use the term "complex" to define those species with at least two molecules. A(b!1).B(a!1) is an example of an species formed by molecules [A, B] bound together. The "!" symbol is used in BNGL to denote a bond.

A "*reaction*" is defined as the transformation(s) between a tuple of species defined to be the reactants into a tuple of species defined to be the products following a rate law. The transformation is defined as the stoichiometric difference between the product and the reactant.

A "*rule*" is a compressed representation of a set of one or more reactions. Wildcards and patterns are used to generalize the different contexts where a rule can take place. A pattern specification includes the definition of one or more molecules with optional specification of their connectivity and their states [37]. *Reaction center* refers to the components that undergo direct modification between reactants and products. All other information that surrounds the reaction center is denoted as the rule's *context*.

For example, consider the following rule where two molecules A, B merge together to form a complex: A(b, c) + B(a) → A(b!1, c).B(a!1) k1. Molecules (A, B) are the reactants while the product is the A(b!1, c).B(a!1) complex. Finally k1 is the rate constant. In this specific example the A(**b**) and B(**a**) components are the reaction center, while A(**c**) is the rule's context (since it does not undergo any change).

Rule-based modeling is a graph-based language where chemical species are encoded as nodes and the relationships between them as edges. Chemical processes are encoded as graph operations

over the set of nodes and edges defining the species in the model. BioNetGen recognizes six kinds of processes a set of reactants can undergo during the firing of a reaction rule. In the following list I describe those processes along with the graph operation that encodes them.

- Creation of a bond: This operation typically encodes non-covalent binding interactions between proteins and nucleic acids. RBM encodes this as the creation of an edge between two complexes.
- Deletion of a bond: A counterpart for the previous operation, the destruction of a non-covalent bond is encoded by the deletion of an edge.
- Changing an internal variable: Post-translational modification refers to the generally enzymatic modification of proteins by attaching a functional group (phosphate, acetate, methyl, etc.) to it. RBM encodes this functionality by using internal variables associated to individual molecules.
- Adding and deleting a molecule: The addition and removal of chemical agents from the model's reactant pool can be simply encoded by the addition or removal of a node.
- Compartment change: As part of its compartmental extension [40], BioNetGen encodes subcellular compartment location information as variables associated with the molecules in a complex, such that when a complex crosses a compartmental barrier this is simply encoded as a variable change. See Section 4.2.1.1 for more details on compartmental BioNetGen.

1.2.2 Visualization of rule-based models

In Fig. 4 I showed a rule visualization graph approach that illustrates reaction events as graph operations. Although this provides an accurate representation of individual events it does not offer by itself a global perspective of what is the role of a given reaction rule or the reactants involved in it into the larger scheme of things.

The structured graph encoding available in a rule-based modeling specification allows us to generate a number of model-wide visualization approaches using both the contact map and flow map approaches. I show in Fig. 5 examples of contact map visualizations for the FcεRI model. This approach, which is closest to SBGN entity relationship diagram focuses on visualizing the role of different molecule agents across the system in the form of its modification states or binding interactions. Additionally, contact maps show a clear view of what are the basic molecular building

blocks in the system and how they are related to each other. The main disadvantage of this approach is that it does not encode temporal information about the activation order of different events. In Fig 6 I show two examples of flow map visualizations for RBM's. A flow map is a process-centric visualization that illustrates the temporal relationship between the reaction events in a model. Flow maps generated from a RBM model definition hold an advantage over those generated from RNM's (like SBGN process diagrams) since instead of showing the entire reaction network it focuses instead on the compressed set defined by the reaction rules in the RBM description. Additional to this it is also possible to determine how does the state of the system influences the firing of a rule, either as a reaction center or as a reaction context.

Two examples of flow charts are the Kappa story and the regulatory graph. Figure 6A presents a visualization provided by the Kappa rule-based modeling platform called a story. A story details the order in which simulation events happen to generate an observable of interest. This visualization is a targeted flow chart for a specific end-point state. On the other hand regulatory graphs provide a global visualization of the entire model. In [23, 41] I showed a framework that allows for the semi automated generation of this kind of diagrams from the model description and minimal user input.

1.2.3 Comparison of rule-based models

The strong semantics imposed by graph encoding of biological events in Rule-based modeling allows us to precisely define the operations performed by the reaction processes in a model, a feature that naturally extends to other operations like visualization and comparison. In [44] we introduced a framework called MOSBIE that allows for the comparison of the species between two BioNetGen models through the contact map specification. The framework (Fig. 7) allows for a side by side comparison of the contact maps associated to a set of models that automatically highlights the structural similarities and differences between them.

However, although RBM facilitates model comparison by resolving the intra-model relationship between its species, it is still necessary to provide the mapping of species between models in order to perform a model comparison operation. In MOSBIE, this information is manually provided by the user by matching the name space of those molecules.

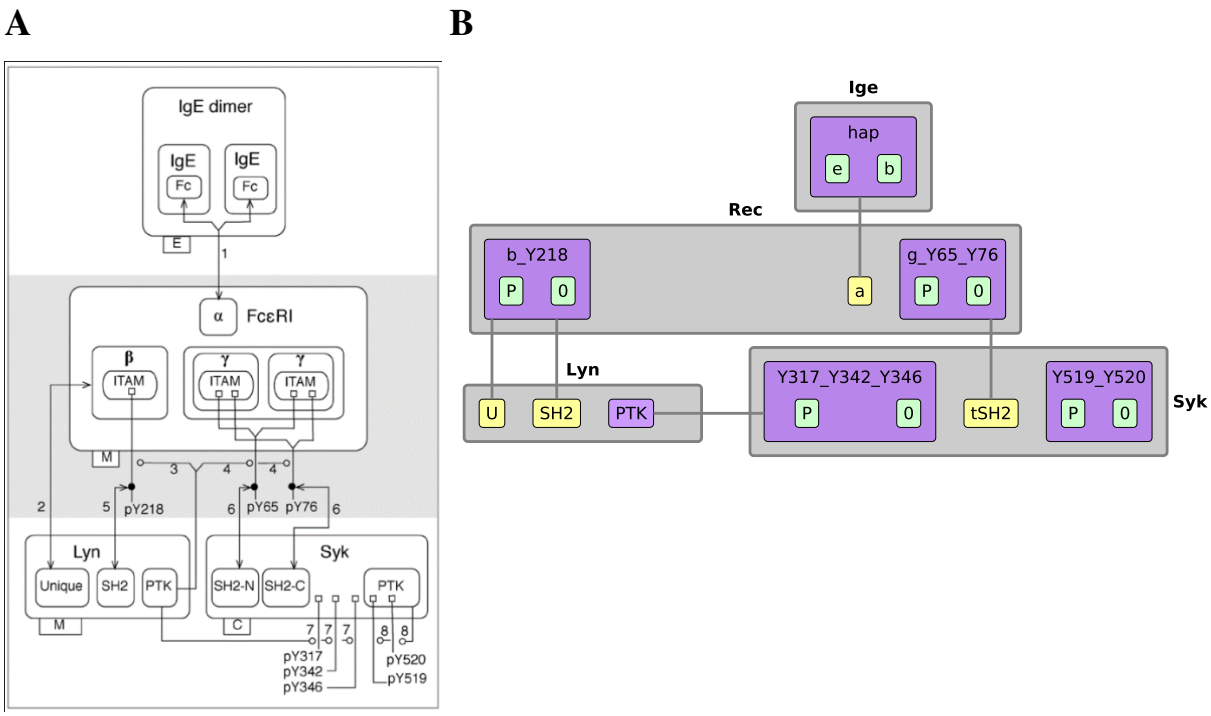


Figure 5: Contact map visualization for the FcεRI signaling pathway. Both approaches illustrate molecules as nested graphs with binding and modification sub-units, and edges encoding non-covalent bond relationships. A shows an extended contact map taken from [42]. B shows a contact map as implemented by the BioNetGen framework.

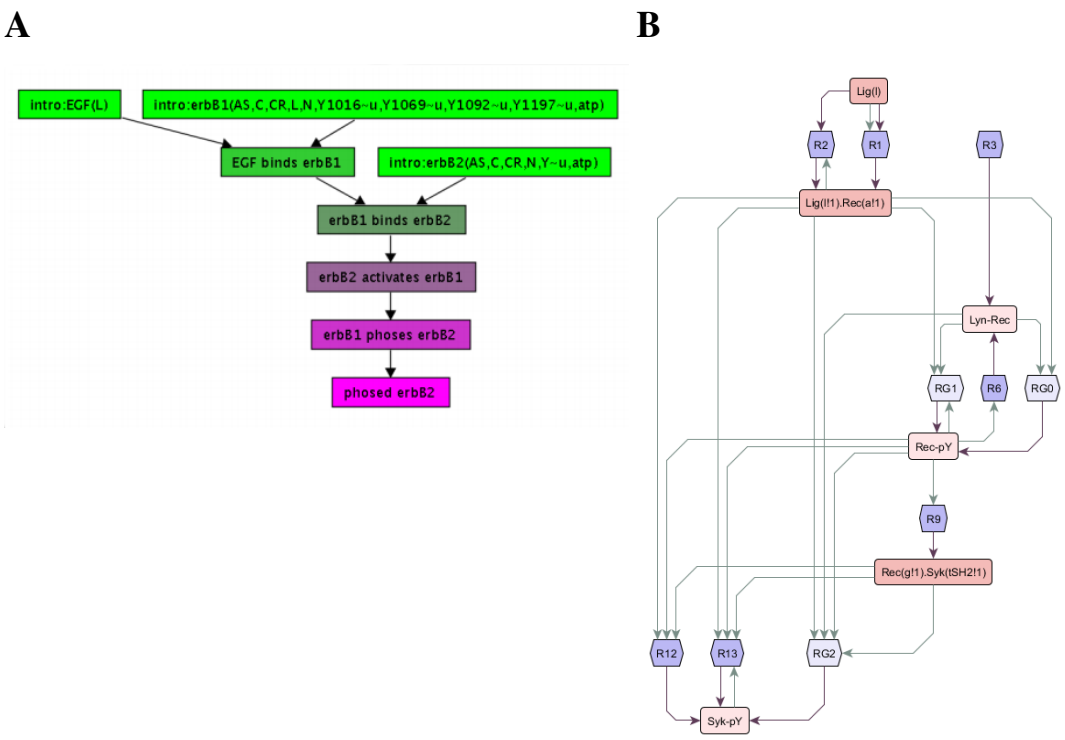


Figure 6: Flow map visualizations for the EGF-EGFR and FcεRI signaling pathway. This approach offers a process centric view which focuses on the temporal order in which events take place. A shows a visualization generated by Kappa called a 'story' (image taken from [43]). Nodes in this graph are reaction processes while edges encode pre-requirements for those processes. B shows a regulatory graph illustrating the FcεRI signaling pathway (same model as Fig. 5). Blue nodes are processes while red nodes is a graph pattern involved in the process, either as a reaction center (red arrows) or as reaction context (gray arrows).

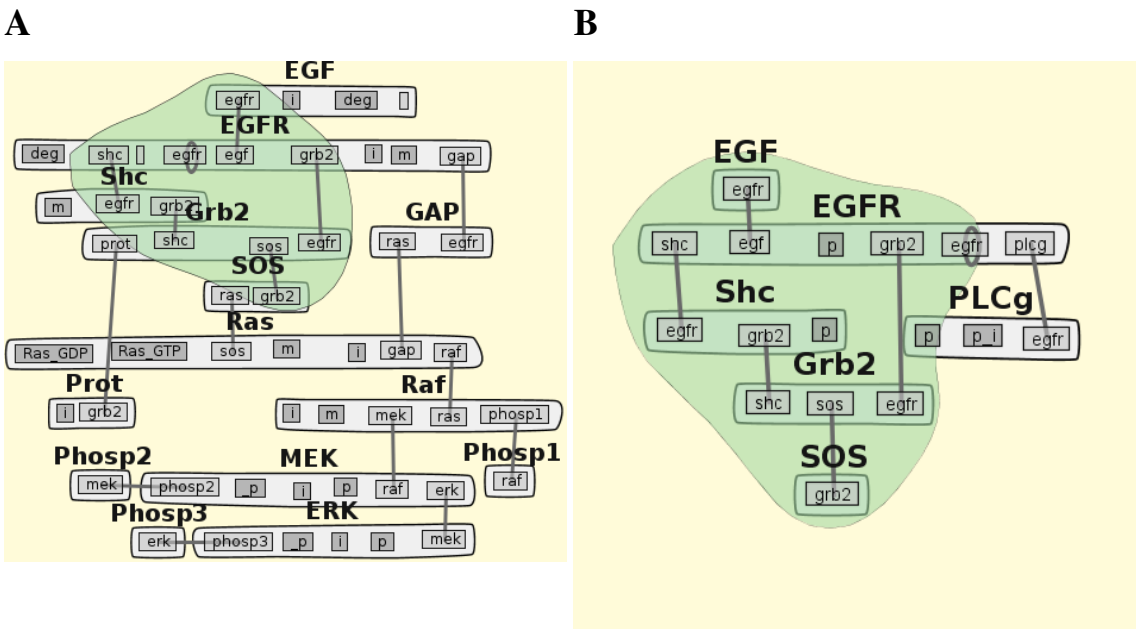


Figure 7: MOSBIE comparison of two models of the EGFR signaling cascade (A: Schoberl [10] and B: Kholodenko [45]). The green overlay highlights the set of overlapping molecules and components.

1.2.4 Limitations to rule-based modeling

The ability to encode molecules and complexes as structured objects comes with a significant cost in the complexity of the model's syntax. Whereas in RNM we only had to worry about simple lists of reactants and products, in RBM we have to consider molecules subdomains and graph transformations, which significantly increases the modeling burden on the user. However, a number of tools have been developed that address this problem by providing a graphical interface to rule-based modeling, like SIMMUNE [46], BioUML [47] and VCell [48].

Because of this accessibility barrier, combined with the fact that RBM is a relatively new modeling approach when compared to reaction-network modeling, there is a much larger set of modeling literature to draw from using reaction-networks when compared to that of rule-based modeling. Additionally, the lack of a common model specification standard for RBM's has resulted

in a fragmented community where each modeling tool encodes their system in different, often non-compatible formats. As a result of this there is also a smaller number of model informatics oriented tools that would allow modelers to share information and build on each other's efforts. The development of the SBML extension *SBML-multi* is a promising effort towards a common format that allows the exchangeable specification of rule-based models.

Finally, the lack of proper annotation standards in rule-based models has hampered the development of RBM comparison tools. Past work has explored alternatives like appending biological information in the form of accompanying documentation [42, 49], while a preliminary specification for a formal annotation scheme for annotating rule-based models [50] has been recently proposed. It is our belief that common modeling and annotation specification standards will encourage more community efforts that make up for RBM's current deficiencies.

1.3 SIMULATION OF RULE-BASED MODELS

1.3.1 Network-based simulation

A simple way to simulate a rule-based model is to bring it to a form amenable to RNM simulators, a process known as network generation. In order to generate the set of reactions that a set of rules encodes, the following process is followed:

- The reaction rules are applied to the initial set of species to get a set of valid seed species-reactants mappings
- For every valid species-reaction pattern combination we apply the corresponding graph operation to obtain a product pattern.
- Unique products are added back to the species set. Unique reactions are added to the generated reaction set
- The process is repeated until no new products can be added.

Once a network has been generated the model can be simulated using any of the approaches I outlined in Section 1.1.5, like the Gillespie algorithm and ODE solvers.

1.3.2 Network-free simulation

The simulation algorithms for reaction-networks, such as those based on differential equations or the classic implementation of the Gillespie algorithm, require the enumeration of all possible species and reactions in the system prior to the execution of the model. This approach is not convenient when combinatorial complexity is an issue. As the number of species and reactions in a model increases so does the execution time of the model, since the combinatorial complexity of these algorithms is dependent on the number of reactions [51, 52]. Rule-based modeling addresses this problem during model specification by allowing the use of rule-patterns such that the modeler can focus on the biological events encoded by the model while abstracting away the biological context in which they happen. However, this minimalistic model definition does not extend to model simulation if we need to generate the network prior to the time series generation.

Network-free (NF) simulation avoids the need to generate a network by instantiating every individual molecule pattern in the system as a separate data structure [51, 53]. This is in contrast to network-based approaches were the molecules in the system are lumped together into single population pools. This particle data structure can be individually matched to any of the reaction rules in the system using subgraph pattern matching. Based on this we construct a lookup table that keeps track of all particles and the reaction rules they are associated with. The algorithm then performs a Gillespie simulation that acts on the set of reaction rules in the system using this mapping table as an auxiliary data structure for performing operations like propensity calculation and selecting which reactants to modify when a reaction fires. The pseudo-code for the algorithm is shown in Algorithm 1.

The computational cost of this approach is independent of the network size, and is instead dependent on the number of rules r in the simulation. There is an initialization cost though that is $O(rp)$ where p is the number of molecular agents in the system, which involves building up the reaction-molecular sites auxiliary mapping table. As such, NFsim is best suited to system where the reaction-network is significantly larger than the set of rules in the system, which is the case in any system that suffers from combinatorial expansion.

There has been a number of simulators implementing the Network-Free approach released in the last few years like NFsim [51], RuleMonkey [53], KaSim [54] and DYNSTOCH [55]. A more

Algorithm 1 Network free simulation pseudocode

Input: $\{R_i, P_i, T_i, k_i\}_{i=1}^{N_R}$, M_0 {rules as a (reactants, products, transformation, rate) tuple and molecule objects at time 0}

Output: M_n {final set of molecules at time n}

for $i = 0$ to N_R **do**
  for $j = 0$ to $|R_i|$ **do**
    $sites[i][j] \leftarrow$ {reactant j sites in M_0}
  end for
  a[r] $\leftarrow k_i \prod_{j=1}^{N_R} sites[i][j]$
end for
$a_{tot} = \sum_{N_R} a$
while $t < t_{end}$ **do**
  $\Delta t, r \leftarrow$ {Select Δt and next rule following Gillespie' direct method}
  $sp \leftarrow$ {Extract a randomly selected set of patterns from $sites[r]$}
  $m_{modified} \leftarrow$ {select molecules connected to sites $\in sp$}
  Apply transformation T_r to sites in sp
  Updates $sites$ and propensities a with information in $m_{modified}$ and other newly created molecules
  $a_{tot} \leftarrow \sum_{N_R} a$
  $t \leftarrow \Delta t$
end while

in depth review of the state of the art of network-free simulators is provided in Section 4.1.

1.4 SPATIAL MODELING

Beyond the modeling of well-mixed systems, the spatio-temporal non-homogeneous distribution of species is important for the accurate representation of multiple biochemical systems [56, 57]. There is a wide range of frameworks that allow a user to model spatial phenomena, from solvers that allow the study of space as a continuous medium treated through partial differential equations [32], all the way down to representations that allow the stochastic modeling of individual species particles [58, 59]. In Section 4.1 I provide a review of the state of the art of spatial modeling frameworks. For the rest of this introductory section I will focus on particle spatial modeling in general and MCell in particular since its technical details are directly relevant to Chapter 4.

MCell is a Monte-Carlo, particle-based spatial simulation engine for the simulation of biochemical systems at the subcellular level and at microscopic timescales. This approach is appropriate to simulate the spatial aspects of a variety of processes like neurotransmission at chemical synapses in the brain and peripheral nervous systems [60, 61], modeling of chemotaxis [62] and other cellular processes [63].

The MCell framework consists of two sections: The first is the MDL language, which defines the simulation setup for our system consists of several sections, some of them similar to the ones we would find in the BNGL and SBML languages:

- Initialization: time step, number of iterations
- Species definitions: the base MCell implementation is a *reaction-network* based framework and as such the full list of species types must be provided in the model specification. Other spatial parameters like diffusion constants are also defined here.
- Reaction definitions, where the full reaction network is specified. MDL supports mass action kinetics and a specific set of units for the rate constants ($1/(Mols)$ for bimolecular volume reactions and $1/(um^2Ns)$ for bimolecular surface reactions)
- Reaction data output, which defines functions of the population levels from a subset of the model's species of interest to the user.

Additionally, there are several model definition aspects that are unique to a spatial model:

- Geometry definition: The geometry is defined as a set of triangulated surface meshes representing the membrane of a subcellular domain. When those meshes are watertight they can be used to compartmentalize the reactions in the system
- Molecule placement: The initial particle pools can be placed in the spatial volumes and membranes defined in the model

Once a model has been defined, it is passed as input to the simulation engine. The lifetime of a molecule particle in the MCell simulation engine [59] is illustrated in Fig. 8. When a particle is created, MCell schedules a destruction time for the particle that will trigger if the species does not react with any other particle throughout its lifetime. Unimolecular reactions in MCell are used to calculate the lifetime of a particle: Instead of probing for the probability of a given particle reaction at every timestep, its total lifetime is precomputed based on the unimolecular reactions it participates in, where the lifetime of a particle is chosen from the exponential probability distribution of lifetimes $\rho(t) = \frac{1}{k}e^{-kt}$.

In the meantime the particle will diffuse throughout the volume. The velocity vector for a species particle at a given time step is a function of the particle diffusion constant and the user specified timestep. MCell tracks the trajectory of a particle using ray tracing to probe for collisions with surfaces and other molecules.

Given a collision event MCell will probe the simulation engine for the probability of these two particles reacting. The calculation of reaction probabilities in MCell is a two step process [64]. For a reactant set $X = \{x_1..\}$ and product superset Y, we first group all reactions with the same set of reactants:

$$
\begin{aligned}
X &\rightarrow Y_0 && k_1 \\
X &\rightarrow Y_1 && k_2 \\
&.... \\
X &\rightarrow Y_n && k_n
\end{aligned}
$$

To calculate the total reaction rate as $k = \sum_{i=1}^{n} k_i$. From here we can make a stochastic determination of whether a reaction occurs. If a reaction is selected to occur, we can then choose to fire one of the reactions paths above with probability k_i/k. The challenge for MCell then is to make sure that the reaction probabilities calculated on a per-particle collision basis leads to a reaction rate that matches the dynamics predicted by the mass-action.

Bimolecular reactions are driven by particle collision events. As mentioned earlier the reaction probability per collision is a function of the user defined rate and different factors depending on the dimensionality on where the reaction occurs. Below I provide a brief overview of these reaction events and their associated probabilities. For the full derivation please refer to [60].

- Surface-Surface reactions: Surfaces in MCell are triangular tiles that cover the mesh elements of the surface. The probability of a reaction occurring given two colliding particles located in neighboring surface mesh tiles over time Δt is $p = \frac{k}{A}$, where A is the tile area.
- Volume-surface reactions: This probability is a function of the area of the tile the surface reactant is placed in and the diffusion speed of the volume particle. $p = \frac{\sqrt{\pi}k}{Av}$ where the characteristic velocity v is $v = \lambda/\Delta t$ and $\lambda = \sqrt{4Dt}$ is the diffusion length constant, with D being the diffusion constant.
- Volume-volume reactions: This probability is similar to volume-surface reactions but it now takes into account two velocity vectors v_i and v_j which account for the diffusion velocities of the two particles undergoing collision. Since in MCell particles are dimensionless, for the purposes of collision the particles are assumed to be centered on a disk of area $A_D = \pi r^2$ such that it sweeps and checks for collisions in a cylinder that extends in the direction of its velocity vector. Using this assumption we calculate the volume-volume reaction probability as $p = \frac{\sqrt{\pi}}{2(v_i+v_j)A_D}k$.

1.5 THESIS OUTLINE

This thesis presents the integration of rule-based modeling principles to the analysis and simulation of models from multiple sources and paradigms. In Chapters 2 and Chapter 3 I will present

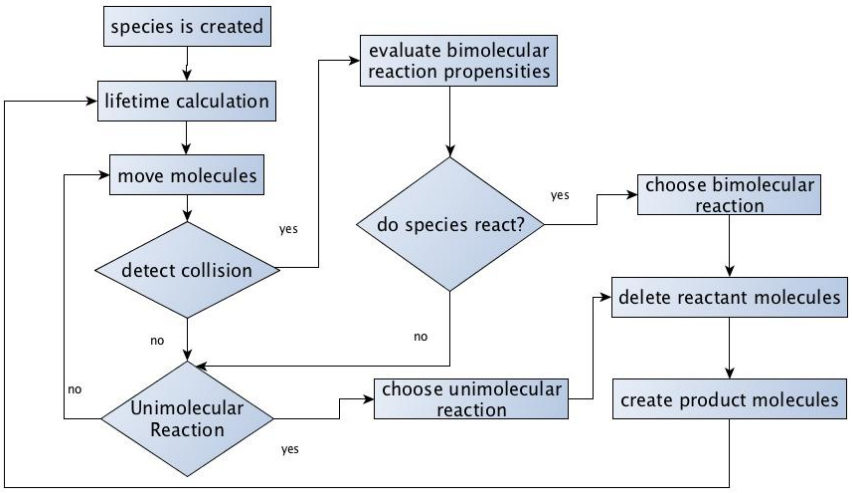


Figure 8: Flow chart illustrating the lifetime of a particle in the MCell simulation engine

Atomizer, an algorithm for the extraction of implicit assumptions from reaction-network models to obtain a rule-based modeling representation. Chapter 2 focuses on the analysis of a model's structural assumptions, or what are the molecular interactions and modification mechanisms present in a model. Chapter 3 focuses on the analysis of process assumptions that describe the temporal ordering between those structural interactions discovered in the previous chapter. The information recovered through the Atomizer algorithm allows us to perform meta-modeling operations like model analysis, alignment, comparison and visualization, which I demonstrate through the application of Atomizer to a large dataset of reaction network models and the development of a model analysis pipeline. In Chapter 4 I present MCell-R, a framework that brings together the spatial modeling power of the MCell simulation platform together with NFsim, a network-free simulation algorithm for rule-based models. This allows us to model and simulate multi-state, multi-component spatial systems in an efficient way, which I demonstrate through the application of MCell-R to two systems that present these features: FcεRI and BLBR.

2.0 THE ATOMIZER: EXTRACTING STRUCTURAL INFORMATION FROM REACTION NETWORK MODELS

The first step in the process of building a dynamical model of a cell regulatory process is to identify the sets of components and interactions to be included. The chemical behavior of biological pathways can then be cast in terms of systems of differential equations. This is a convenient approach, given the large and well studied body of literature about how to model such dynamics and networks [15]. A reaction-network model (RNM) is composed of a series of species, reactions and rate laws from which the set of differential equations that describe the system dynamics can be derived. Species in RNM's are encoded as structureless objects [A, B, C], and reactions involving these structureless objects carry no explicit information about the way in which products are related to its reactants (for example, A + B → C). As a result, RNM's present a disadvantage in the fact that the existence of all possible species must be known in advance, instead of the structure of a complexes being the explicit aggregation or reconfiguration of its constituent reactants. This representation where no structure information is encoded has the side effect of obscuring some of the original assumptions made by a modeler [19], which can later become an issue when performing model analysis, comparison and reuse [21].

In contrast, rule-based models (RBM) are a graph based approach to encoding biological dynamics where building blocks like genes and proteins are modeled as structured objects. Non-covalent binding domains and post-translation modification sites are explicitly encoded as component subunits associated to a molecule. Reactions are then stated as graph rewriting rules, such that the creation of a bond between two reactants is explicitly encoded as an edge creation operation between two molecular binding domains. Similarly, molecules have internal state variables associated to them that allow for the explicit representation of modification operations [34, 35]. Using these features a modeler can specify a system that explicitly keeps track of the compositional bio-

chemistry of all species.

There are several advantages to bringing the model to a form where binding and modification structural information is explicit. Model informatics operations like analysis and comparison rely on a system specification that allows for the understanding of how the components in a model are related to each other, for example in global contact map visualizations. RBM explicit encoding of molecular interactions and modification states facilitates the generation of this kind of map [23, 44, 65]. Additionally, this high-level representation of a model's structural elements can be used to enhance the understanding of the model by the general community.

Conversely, even though a large number of models have been encoded using RNM (as evidenced by databases like BioModels [15] and the VCell collection of SBML models [32]), the limited amount of information they include constrains the kinds of analysis and the complexity of the model informatics operations that can be performed [20]. For example, model comparison and aggregation algorithms need information about how elements in a model relate to each other, but this information cannot be extracted from the definition alone without additional considerations or meta-data information. In contrast, the information contained in an RBM provides a way to perform this analysis in a straightforward way, however the number of models encoded as rule-based is significantly lower than that of reaction-network modeling, which limits the scale of the analysis we can perform.

In this chapter I provide a method for extracting structural assumptions from RNM's called *Atomizer*. By transforming a reaction-network to a form where this information is explicitly encoded in the model description I leverage the large collection of existing reaction-network models to perform model informatics operations like analysis, comparison and aggregation.

2.0.1 Chapter organization

The chapter is organized as follows: In Section 2.1 I evaluate the state of the art regarding the extraction of structural information from RNM. In Section 2.2 I present the details for our solution, Atomizer, for extracting structural information from reaction networks, while in Section 2.4 I present our validation tests for the atomization framework. Finally, Sections 2.5 and 2.6 show application of Atomizer to the Biomodels database, reporting on both specific examples and sum-

mary statistics about the ammount and extent of atomization recovery.

2.1 DETERMINING INTRA-MODEL RELATIONSHIPS: A REVIEW

The comparison of two models starts from the premise that within the entities that comprise each model there is an overlapping subset that will allow us to study the ways in which the two models are similar. By extracting this subset of species and studying the similarities and differences of their role in each model we can perform higher order tasks like model alignment and aggregation. Following this, model comparison has been defined as a two-part problem consisting of *intra-model* species mapping where we analyze and obtain a representation of how species are in a model related to each other, and an *inter-model* species mapping step when we use this representation to obtain a meaningful comparison between two models [26]. However, it is often the case that extracting these mappings and characterizing the role of each participant species throughout the model is a non-trivial task. For example, in RNM's like those encoded in the SBML format the relationship between two species can only be indirectly extracted by studying the reactions in which these two species are participants. That is, the lack of strong reaction semantics that define the biological relationship of reactants to products prevents the modeler from encoding this information explicitly [19].

A *model ontology* provides a formal definition and clarifies the intended semantics for the data encoded by a model [21]. In rule-based modeling this ontology is a part of the language design where biological processes are encoded as graph operations that explicitly denote how the reactants are transformed into products. In reaction-network modeling this information has to be extracted from external documentation and annotations attached to the model.

A number of frameworks have been developed that programmatically generate a model ontology from a reaction-network model that describes the model and the relationship of the elements in it [19, 66]. By providing a description that is directly comparable between models we can then characterize the entity intersection between them [20] and provide an inter-model mapping. Among the different existing frameworks that construct a model ontology that explicitly describes the intra-model entity mapping, the two factors that differentiate each framework are the kind of

input they use to construct the mapping ontology and the kind of information encoded in the ontology itself.

The input information used to construct the ontology can fall in one of the following categories:

- *Reaction Network analysis based frameworks*: These frameworks make use of the reactions information to quantify the relationship degree between products and reactants. For example, a reaction of the kind `A + B -> C` can be used to define `C` in terms of `A` and `B`. This is done progressively over the entire reaction network to construct the model's ontology. Maggiolo's extraction framework uses this approach [67].
- *Species analysis based frameworks*: This kind of analysis relies on information associated with the biochemical species. Analysis of this information can be further subdivided into the following:
  - Lexical analysis based techniques, which can be used to extract information from the model namespace. For example in a reaction like `A -> A_P` determining that `A_P` corresponds to a phosphorylated version of molecule `A` allows us to characterize the relationship between `A` and `A_P` in a more precise way than if the species had names not representative of their functionality.
  - Annotation based frameworks: A more structured version of the previous approach, analysis of annotation information relies on the modeler including meta-data that maps a given model entity to information in an external database. *SemanticSBML* [66] and *SemGen* [19] use this approach.
- *Hybrid frameworks* use a combination of the approaches enumerated above. Our own framework *Atomizer* is an example of such framework [3].

Using this information it is then possible to construct the intra-model mapping ontology. Depending on the information encoded in this ontology it can take two forms:

- Species information: A species vector representation states every species in a model as a linear combination of simpler species. However, this representation does not state how those species combine with each other to produce a more complex representation. For example, given reaction `A + B <-> C` this representation would encode complex C = $\{A, B\}$ without specifying how `A` and `B` are related. Both Maggiolo's framework and semanticSBML use this encoding.

- Species + Process information: This representation characterizes not only the species, but also the processes in the system and how they denote a relationship between those species. For example, given the same reaction `A + B <-> C` this representation would encode complex C as a species formed of A forming a non-covalent bond with B. Atomizer uses this approach to encode its ontology information.

2.1.1 Petri-net analysis

In [67, 68], Maggiolo-Schettini et al. introduce a petri-net analysis framework for defining an intra-model mapping ontology. The analysis algorithm found in this framework considers reactions of the kind shown in 2.1, where the right hand side is defined as a linear combination of the reactants and 2.2, (where the right hand side is defined as a modification of the single reactant).

$$A + B \leftrightarrow C \tag{2.1}$$

$$A \rightarrow D \tag{2.2}$$

In contrast, an approach that is solely reliant on stoichiometry will have difficulty defining the composition of the products when they cannot be clearly mapped to the reactants. For example, consider the following set of reactions in addition to the reactions defined above:

$$A + D \leftrightarrow E + G \tag{2.3}$$

$$C \rightarrow H + I \tag{2.4}$$

Products $\{E, G\}$ in reaction 2.3 have an ambiguous mapping to reactants $\{A, D\}$. Similarly in reaction 2.4 since reactant C was stated to be composed of $\{A, B\}$, its mapping to products $\{H, I\}$ is unclear. In order to solve this ambiguity Maggiolo-Schettini's algorithm makes use of user provided information in the form of pre-processed reactions and dynamically inserted dummy species in order to resolve reaction information that cannot be resolved through stoichiometry alone (like reactions where there is a catalysis-induced process, when there is ambiguity regarding the

exact reactant to product mapping or in general when conservation of mass is broken in a particular reaction).

The resulting ontology is a map listing all the species in the system versus a discrete vector describing the building components of each species. In an RNM formed of reactions 2.1 and 2.2, this ontology would be $\{ A = [], B = [], C = [A, B] \}$.

2.1.2 Annotation based frameworks example

In [66] Schulz et al. describe their framework *semanticSBML*, which provides a way of performing model alignment for SBML models through the model's associated meta-data. semanticSBML considers a pair of partially annotated SBML files, where the annotations are indicative of the biochemical structure of the SBML species in the system.

This approach uses an n-dimensional vector (where n is the number of different annotations included in the system) that quantifies how strongly a given species is described by a certain annotation set. In the example in Fig. 9 these feature vectors would have the form $\{ x_r, x_g, x_b \}$ where each component is a number in the [0,1] range denoting how much a given species is defined by a particular feature (red, green and blue in this case). S1 would then be assigned feature vector $\{ 1, 0, 0 \}$, S2 is assigned feature vector $\{ 0, 1, 0 \}$ and S3:$\{ 0, 0, 1 \}$.

Given these partially annotated models as input, for each model *semanticSBML* propagates this information through the system such that all the species in the system can be defined in terms of the existing annotation information. In order to propagate this information to the non-annotated species the algorithm considers a so-called *propagation matrix* that measures the connectivity between elements in the model as measured by the information contained in the reaction network, and then performs a process akin to information diffusion. Fig. 9B shows an example of how the model ontology information would look like after the application of the propagation algorithm. Finally, the model uses the ontology generated for each system to map elements between models using a similarity scores based on the feature vectors. An extended example illustrating a model ontology generated using semanticSBML is included in section 3.3.2.

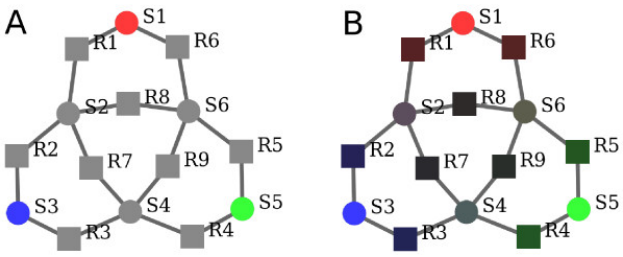


Figure 9: Feature propagation in semanticSBML. Panel A shows a petri net containing six species where only the first three contain annotation information. Panel B

2.1.3 Drawbacks and limitations of existing ontology extraction approaches

The most important drawback found in the frameworks described in sections 2.1.1 and 2.1.2 is the limited amount of information encoded in the output ontology. In both frameworks a given species' ontology is represented as a vector encoding a linear combination of simpler species. In the case of semanticSBML, this vector consists of a set of numbers evaluating the degree of relationship between a known base set of species (annotated subset) to the rest of the model (non-annotated subset). This encoding excludes all process information related to the stoichiometric composition of a complex (for example, it cannot encode oligomers) or the compositional relationship between its components.

In the case of Maggiolo et al.'s framework the system conserves stoichiometric information by encoding a species ontology as the set of its constituent species. This approach has the benefit of differentiating between chemical complexes that contain the same proportion of species but differ in the stoichiometry of its constituent components like dimers (or k-mers in general). However a set representation still suffers from the same problem as semanticSBML where the encoding contains no information about the biological relationship or the kind of processes that allows these biological building blocks.

In terms of its input, both *semanticSBML* and *SemGen* rely on information extracted from species and process annotation information in order to generate the intra-model species mapping. This approach shows strong limitations since the semantics associated with these annotations are not strongly defined [19, 27]. For example, the most common annotation found in the SBML

repository BioModels is the qualifier operator *isVersionOf*. *isVersionOf* is an operator that takes two arguments $\{A, B\}$ and indicates that B is an ontologically narrower concept than A, however this interpretation can and has been abused to be an umbrella term that roughly defines that two elements are related to each other without specifying how [21].

Moreover, the expressive power of process annotations is often not sufficient for the purposes of model alignment and comparison. Process annotations are typically encoded using the Systems Biology Ontology (SBO) which characterize the different entities in a model to a given biological classification. For processes it can classify a reaction as complexation, modification transport, etc. However this annotation system is not capable of encoding operator information indicating how the reactants and products in a process participate and influence this biological operation, which hampers the amount of information we can extract from the model.

Input wise, Maggiolo's et al. algorithm strongly relies on the stoichiometry information contained in the reaction-network. However, this approach is only able to recover complexation processes. For all other processes for example `A-> A_P` it can only assign weak relationship semantics similar to the *isVersionOf* operator. In order to address this problem, the framework gives the user the ability to input user information that fills in these gaps.

There are multiple advantages to having a model ontology that is capable of directly encoding intra-model mapping information using strongly defined graph semantics [35]. In [41] I showed the advantage of encoding a machine-readable version of this process information ontology for the purpose of static analysis, in this particular case when applied to model process visualization. Similarly, in [44] we presented a model alignment visualization tool that makes use of information encoded using such an ontology to get an automated high level view of the functional differences between the model entities.

2.2 THE ATOMIZER: FROM REACTION-NETWORKS TO GRAPHS

I introduce an algorithmic approach called *Atomizer* for the automated extraction of molecular structure information from RNM's. This is done by analyzing implicit information found in reaction-networks, like reaction-stoichiometry, naming conventions and model meta-data. The

algorithm adds the structure extracted this way back into the model by encoding it into the rule-based modeling (RBM) paradigm [37]. RBM employs a hierarchical set of data structures that allows for the explicit encoding of molecular interaction and modification information.

More specifically, the atomization algorithm works such that when given a set of molecular species together with a reaction network denoting transformations between this set of species, it will extract the basic set of building blocks from which all complexes are built, it will encode the set of species in the original model as graph objects and rewrite the reaction network as a set of graph rewriting rules that specify the conformational changes that allow the transformation from reactants to products. This means atomization specifies whether a reaction includes the creation or deletion of a non-covalent bond, a post-translational modification in a given species or the creation or deletion of a molecular entity. This ultimately leads to an extraction of the molecular structure and a clearer understanding of the changes each reaction induces together with the conditions necessary for each process to fire.

The atomization algorithm is a three-step process illustrated in Fig. 10 and further described in Algorithm 2. In the first step the algorithm iterates over every reaction to extract information about *how products can be stated as the combination of its reactants* (Section 2.2.1), using the stoichiometric and naming convention information I mentioned earlier. Additionally Atomizer will also process each species individually to extract annotation information and other meta-data information belonging to each species independent of their role in a reaction. This information is placed in a data structure called the species composition table (SCT). The second step *propagates the information* found in each reaction across all species in the SCT. This leads to the identification of the model's elemental substrates, that is those molecules with no compositional dependencies. Additionally it allows us to state all other molecular complexes as the combination or modification of these basic building blocks (Section 2.2.2). The last step uses this compositional information to *resolve the structure for every species using a graph representation* (binding and modification components) which is encoded using an RBM syntax (Section 2.2.3)

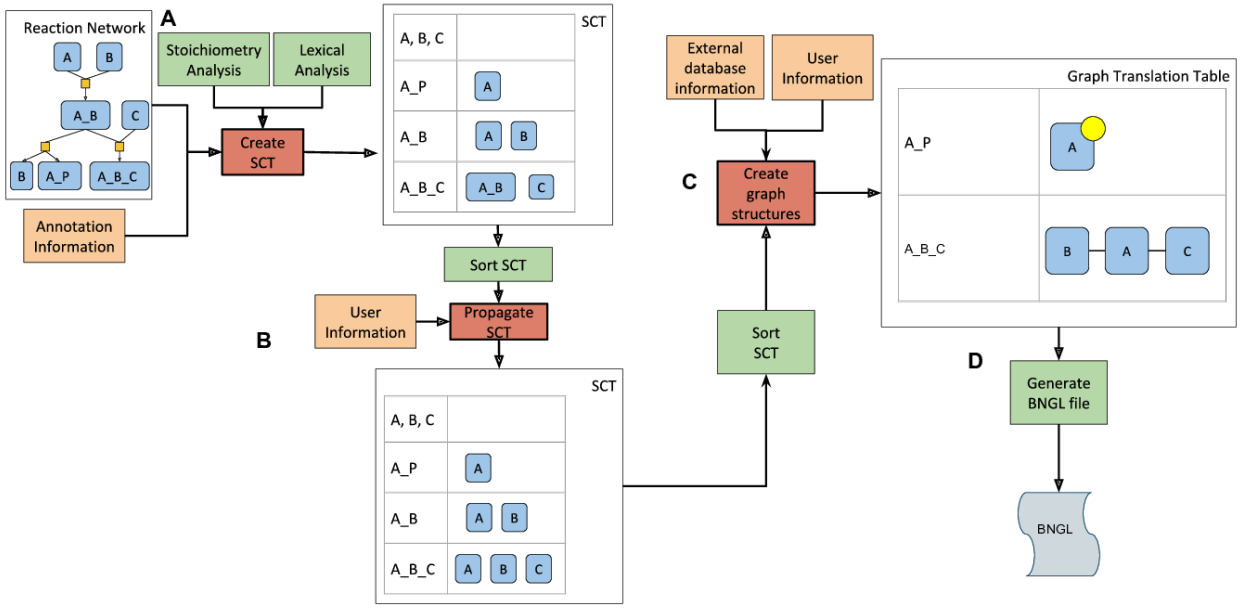


Figure 10: General overview of the atomization algorithm. The algorithm's input is the reaction network specification encoded in SBML, annotation information associated to the model and model-specific user information. Highlighted in red are the main steps of the Atomization algorithm, in green we find auxiliary methods and orange represents external information that aids in the atomization process. **A**: The algorithm starts by iterating over the set of reactions in the model to obtain the stoichiometric composition of every species in the model. Atomizer places this in a structure called the species composition table (SCT). **B**: After sorting the SCT according to the number of dependencies each species has, Atomizer iterates over the table to identify those elements that form the basic building blocks of the model, that is those molecules that cannot be stated as a modification or complexation of simpler elements. All complex elements in the model are then defined as a combination of these simpler elements. **C**: Atomizer creates a graph definition for every species in the model going from the simplest to the most complex sorted by the number of binding or modification operations necessary to create such a complex. The algorithm makes use of external protein-protein interaction or user information to determine the location of an active domain in a complex when they cannot be determined from reaction information alone. **D** Finally, Atomizer use the translation table to create a rule-based specification of the molecules, complexes and reactions in the system and encoded in the BioNetGen language.

Algorithm 2 Atomization general procedure

Input: $Reactions, Species$

Output: $graphStructures$ {A dictionary containing RBM definitions for every species in $Species$}

1: **for** $r \leftarrow 1..|Reactions|$ **do**
2: **for all** $prod \in Reactions[r]$ **do**
3: $SCT[prod] \leftarrow analyzeComposition(Reactions[r], prod)$ {Return a reactant $\rightarrow$ product mapping as defined by r}
4: **end for**
5: **end for**
6: **for all** $s \in Species$ **do**
7: $SCT[s] \leftarrow extractMetadata(s)$ {Annotation information associated to s}
8: **end for**
9: $sort(SCT)$ {re-sort according to the complexity of the resolved SCT entry}
10: **for** $s \leftarrow 1..|SCT|$ **do**
11: SCT[s] = resolveEntry($SCT[0 : s]$) {Resolve the information in the SCT using previously agreed upon entries}
12: **end for**
13: $sort(SCT)$ {Sort according to the complexity of the SCT entry}
14: **for** $s \leftarrow 1..|SCT|$ **do**
15: $graphStructures[s] \leftarrow$ createRBMStructures($SCT[s]$, $graphStructures[0 : s]$) {Resolve the graph structure using simpler species as a basis}
16: **end for**
17: **return** $graphStructures$

2.2.1 Generating the species composition table

The species composition table is a dictionary of the species in the system, with each entry in the dictionary containing a set of stoichiometry vectors describing the compositional structure of a complex.

Let us illustrate how the SCT is built using the example defined by the set of species X_{basic} = A, B, C, D, E, A_P. The set R_{basic} that describes the system's reactions can be seen in equation list 11A.

The first step in building the species composition table consists in processing every reaction to determine how their products can be stated in terms of their respective reactants, which is done by identifying the kind of transformations and processes present in a reaction. In the example from Fig. 11 we can manually identify several kinds of transformations: synthesis of A, degradation of B, complexation and decomplexation of [A, B, D] and state modification of A.

In order to identify these I consider stochiometric (2.2.1.1), lexical (2.2.1.2) and annotation information (2.2.1.3).

2.2.1.1 Stoichiometric information Knowing the number of reactants and products present in a reaction is often enough to discern the kind of transformation process it is encoding [3, 69]. For example having two reactants transforming into one product in a reversible way is sufficient for a reaction to be considered complexation. In the example above this is true for reactants A_B and A_B_C. Having 1 reactant transforming into 1 product is indicative of a transformation reaction, as is the case for product A_P. I show in Table 1 a summary of how stochiometry information is used to fill the SCT in different scenarios.

Formally, a reaction can be represented as a transformation $\{r_1..r_n\} \rightarrow \{p_1..p_m\}$ with n reactants and m products. The structural transformation a reaction applies can be determined from stoichiometry alone if each element in the reactant set can be unambiguously mapped to a single element in the product set. This is straightforward for all synthesis, degradation and modification reactions. For complexation reactions an ambiguity appears where a reaction A + B → C can be equally interpreted as the binding of A to B, or the modification of A to C together with the destruction of B. Adding the requirement for a bidirectional transformation removes this ambiguity for

A

$$\emptyset \rightarrow A$$

$$B \rightarrow \emptyset$$

$$A + B \rightarrow A\_B$$

$$A\_B \rightarrow A + B$$

$$C + A\_B \rightarrow A\_B\_C$$

$$A\_B\_C \rightarrow C + A\_B$$

$$D + A\_B \rightarrow A\_D + B$$

$$A \rightarrow A\_P$$

B

Species	SCT
A,B,C,D	[]
A_B	[A,B]
A_B_C	[A,B,C]
A_D	[A,D]
A_P	[A]

C

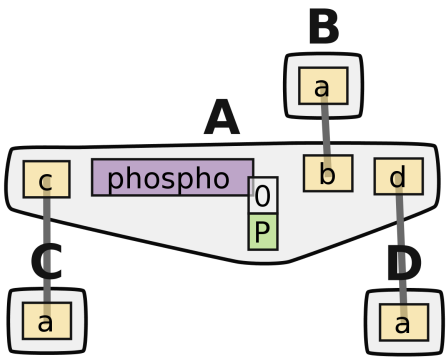


Figure 11: Example model: A Reaction List. B Species Composition Table (SCT): Each entry enumerates the base elements of a given species. C Contact Map. In order from outermost to innermost, the boxes represent molecules, components and modification states. Lines between components represent potential binding interactions.

Table 1: Interpretation of reaction activity based on stoichiometry

Reactants	Products	Interpretation	SCT Entry
`∅`	`A`	Synthesis	No entry
`B`	`∅`	Degradation	No entry
`A, B`	`A_B`	Need more information	No entry
`A_B`	`A,B`	Complexation	`A_B = [A,B]`
`A`	`A_P`	Modification	`A_P = [A]`

most scenarios.

An important thing to highlight is that, although we can easily characterize certain reactions as synthesis or degradation, *no structural information can be extracted from them*.

2.2.1.2 Lexical analysis Other kinds of reactions do not allow for a structure determination that is reliant on stoichiometry alone when elements in the reactant set cannot be reliably mapped to the product set. For example, consider the reaction from the example system $\mathtt{D} + \mathtt{A\_B} \rightarrow \mathtt{A\_D} + \mathtt{B}$. Even when we consider other reactions that resolve the structure of some of the species involved in this reaction like $\mathtt{A} + \mathtt{B} \rightarrow \mathtt{A\_B}$, it is still not possible to discern the structure of `A_D`. However, from a human analysis point of view, our intuition would tell us that the underscore symbol is being consistently used as a binding symbol, and that `A_D` uses this syntax to express the binding of `A` to `D`.

In order to formalize this intuition and include it into our *reactant* → *product* mapping process I developed a lexical analysis engine that scans and detects molecular units contained in the string representation of a complex. By resolving the string representation in a way that allow us to perform an accurate, sub-unit level match between reactants and products we can now understand what kind of transformation a reaction is applying. For example, in $\mathtt{D} + \mathtt{A\_B} \rightarrow \mathtt{A\_D} + \mathtt{B}$ it is possible to discern basic units $\mathtt{A}, \mathtt{D}, \mathtt{B}$. Thus, novel complex `A_D` is defined as composed of basic

molecules A, D.

The lexical analysis procedure is explained in Algorithm 3 below. Briefly, the lexical analysis engine starts by obtaining the list of the most common modifications in a model and the species pairs associated to them. (For example, consider modification $\_P$ and species pair $\{A, A\_P\}$ in the example system). After obtaining this knowledge, I iterate over all modification reactions in the system such that their reactant and product information can be mapped to each other in terms of the total set of species in the system and the set of modifications I have found.

In the example system this would result in novel information for two reactions. For reaction D + A_B → A_D + B the algorithm would be able to ascertain that A_D = [A, D]. For reaction A → A_P I would get more information than that determined by stoichiometry analysis: A_P would be the result of applying a particular type of modification to A. Although structurally this still results in a state change operation as assessed by stoichiometry analysis, specific modification information can be included as meta-data in the final model.

Algorithm 3 Analyze naming conventions: Get list of common modification strings in a model

Input: S, R, SCT {List of species and reactions in a model, partially filled SCT with stoichiometry}

Output: STC {Species composition table filled with information extracted from lexical analysis}

1: $D \leftarrow analyzeNamingConventions(S)$ {Algorithm 6}
2: **for all** $r \in R$ **do**
3: $unitpairMatches \leftarrow matchReactantProducts(r, S, D)$ {Algorithm 7}
4: **for all** $product \in r_{products}$ **do**
5: $STC[product] \leftarrow constructFromMatches(unitPairMatches[product])$
6: **end for**
7: **end for**
8: **return** SCT

2.2.1.3 Annotation information : The annotation associated to a species object are often mapped through a so called qualifier. In SBML, the BioModels qualifiers like '*hasPart*' and '*isVersionOf*', are specially useful to determine the biological structure of a given SBML entity when they are present. In the current version of our software they are used to resolve ambiguities

when the previous two methods are not sufficient to determine the structure of a compound.

Atomizer separates annotations into two groups based on their arity:

- One to many annotations: `BQB_HAS_PART, BQB_HAS_VERSION` describe a $n : 1$ relationship between two sets of entities such that it is possible to establish a partial composition relationship between them. Since annotation information is not ensured to completely define the composition of a complex it is not possible to use it as the sole source of information for an SCT entry, however Atomizer uses it as a way to resolve complexation ambiguities, that is when there is more than one, non compatible way to define a complex. In those cases I select the complexation candidate that matches the annotation information available to us.
- Unary annotations: `BQB_IS_VERSION_OF,BQB_IS,BQB_ENCODES` describe $1 : 1$ relationships that allow two molecules to be written in terms of a base state and a modified state. Similar to complexation, Atomizer uses this to resolve modification ambiguities.

2.2.2 SCT information propagation

The second step, *SCT consolidation* has three main purposes: It will extract the set of molecule types in the model: the subset of species that cannot be defined as the combination or modification of another species set. It will define all other reactants in the table as a combination of one or more of these basic building blocks species. For example, in our example model this means that entry `A_B_C = [A_B, C]` becomes `A_B_C = [A, B, C]`.

Finally, the algorithm will also serve as a sanity check for the consistency of the model: if there is more than one pathway that defined the structure of a species those pathways should result into the same conformational definition. The algorithm can even detail whether the issue is a violation of conservation of mass (stoichiometric definitions conflict) or in the naming conventions used for the model (naming conventions definitions conflict).

As a preliminary step, I will sort the SCT according to the weights calculated by Algorithm 8. After that, for every entry in the SCT Atomizer resolves its composition according to the procedure defined in Algorithm 4.

Algorithm 4 Return a single species definition in terms of basic molecule typesets

Input: s, SCT {An species whose composition I will resolve and the SCT}

Output: $resolvedDefinition$

1: **for all** $speciesEntry \in SCT$ **do**
2: **for all** $candidates \in SCT[speciesEntry]$ **do**
3: **for all** $subunit \in candidates$ **do**
4: **if** isMoleculeType($subunit$) **then**
5: $resolvedCandidate \leftarrow subunit$ {Does not have any dependencies}
6: **else**
7: $resolvedCandidate \leftarrow$ resolveSCT($subunit, SCT$) {get dependencies}
8: **end if**
9: **end for**
10: $resolvedCandidatesList \leftarrow resolvedCandidate$
11: **end for**
12: **end for**
13: **if** consistencyCheck($resolvedCandidatesList$) **then**
14: **return** $resolvedCandidatesList$[0] {If all candidates resolve to same molecule types and stoichiometries}
15: **else**
16: **return** $incosistentDefinitionError$
17: **end if**

2.2.3 Generating the translation table

The final step creates an RBM graph representation for every species in the system based on the SCT. For every complex in the table (e.g. `C = [A, B]`), Atomizer will resolve the connectivity of every individual node to create a fully connected graph. At the same time it will also resolve internal variables (e.g. `A_P = [A]` and give them an RBM encoding in the form of component states.

The three operations that I consider are the following,

1. Basic molecule type: The species makes no references to another reactant, in which case an empty species is created.
2. Complexation: When a species's SCT entry indicates that it is composed of two or more molecules. The main challenge in this scenario is to ensure that all molecules within the set form a fully connected graph that is biologically consistent. In order to find the correct binding subunits within a complex types Atomizer will depend on previously found information in the reaction-network, external protein-protein interaction databases and user provided information. (Section 2.2.3.2).
3. Modification: When a species makes reference to a single molecule and there's a mapping reference in the naming convention dictionary connecting the two. Similar to complexation, Atomizer will query for external information sources when the correct modification subunit cannot be determined from the reaction-network alone. (Section 2.2.3.1).

Finally, after the entire relationship graph is processed the information in its basic reactants (components and states that were discovered later during the component creation process) will be propagated throughout the entire set of species.

2.2.3.1 Modification When processing an SCT entry that represents a modification process, I decide on the component and states to be added to this species depending on the type of modification involved. For example, in a phosphorylation process $\mathtt{A}, \mathtt{A\_P}$ I would create an entry $D[A\_P] = \mathtt{A(P \sim P)}$, while the original molecule A gets updated with $\mathtt{A(P \sim \emptyset)}$ (where state 0 is used as a universal symbol throughout our translation for any relaxed, unmodified state).

In cases where the molecular subunit containing the active site is ambiguous (for example, during the modification of a complex with more than one subunit) atomizer can query Pathway Commons [70] for active site information. If information cannot be obtained this way the user must provide information about the location of the active site. Otherwise, Atomizer will log a failure during the translation process regarding a modification operation ambiguity.

2.2.3.2 Complexation A species complex structure is formed from the aggregation of the structure of its constituent elements. The objective of this routine is such that given a set of molecule nodes I will attempt to create a fully connected graph by defining edges that denote biologically relevant molecular interactions between those molecule nodes.

For a given iteration of the algorithm, consider `C` as a complex formed from elementary molecules M = $\{ m_1..m_n \}$. If there are only two molecules in the complex, I define the two molecules in that pair to be connected, I add a binding component that denotes this fact for use in future iterations of the complexation algorithm and I proceed to the next molecule.

Otherwise if $|M| > 2$, the algorithm creates a superset $P = \{ \{ m_x \} \forall m_x \in M \}$ formed of singleton sets containing every individual molecule. The algorithm then iterates over every set pair in P: for a given pair $\{ p_x, p_y \}$ the algorithm evaluates whether $\{ p_x \cap p_y \neq \emptyset \}$, where the intersection operator is defined over whether two molecule nodes belonging to each of the two sets have an edge that connects them. This connection information is extracted either from previous iterations of the algorithm, or from querying the external databases BioGrid [71] and Pathway Commons [70] for relevant molecule-molecule interaction information between two of the molecules in these sets. If an interaction between molecules $m_1 \in p_x$ and $m_2 \in p_y$ exists, the binding sites connecting them are marked as occupied and removed from consideration for the next iterations (a binding site can only contain one edge) and the two sets are merged into one. Finally, the process repeats until the intersection of all sets in P is empty or $|P| = 1$ (all molecules are connected).

Let us illustrate this with the previous example `A_B`. Given paired molecules `A` and `B` I create a binding pair (`A`, `B`) since they are the only two molecules in the complex. This information is encoded in BioNetGen through the syntax `A(b!1,P ~ 0).B(a!1)`. (The `P` component comes from the phosphorylation reaction described in the previous section).

Let us also define A_B_C from the example case, which is defined from molecule types

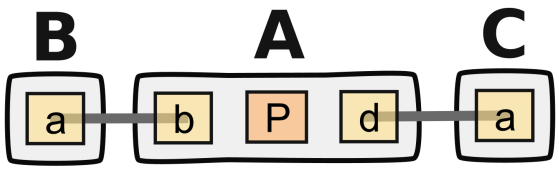


Figure 12: Resulting graph from binding $\{A, B, C\}$ together to form molecule A_B_C

molecules A, B, C. The resulting species would be $\texttt{A(b, P} \sim \texttt{0)}.\texttt{B(a)}.\texttt{C()}$. When trying to create the binding pairs for this complex I will end up with $\{\{A, B\}, C\}$ (since I know from the previous reaction that A and B bind together). However when binding C to the rest of the graph atomizer will need to be provided with external information denoting the graph structure of these nodes. If I have reason to believe that A binds to C (either through BioGrid, PathwayCommons or user information), I add binding sites $\texttt{A(c)}, \texttt{C(a)}$ and merge the two sets.

When creating the final bonds for our species A_B_C, the end result will be the graph illustrated in Fig. 12, $\texttt{A(b!1, P} \sim \texttt{0, c!2)}.\texttt{B(a!1)}.\texttt{C(a!2)}$.

If Atomizer cannot resolve the structure of a complex ($|P| > 1$) it will log a complexation ambiguity error for the user to review.

2.2.4 Algorithm complexity

The algorithm executes in polynomial time with a worst case execution time of $O(nm)$ during the SCT generation phase, $O(n\ log(m)\ log(n))$ during the SCT information propagation phase, and $O(n\ log(n))$ during the graph generation phase, where n is the number of species, m is the number of reactions, and $n < m$ for most models. However, the main execution bottleneck of the algorithm is contingent on the network access time when Atomizer needs to consult external databases like PathwayCommons and BioGrid. These databases and consulted when Atomizer needs to verify whether two species in the system interact. In the worst case scenario the system will need to perform $O(n\ log(n))$ requests where each request can take up to half a second for slow connections. In practical terms though, the average case requires only a fraction of that number

depending on the number of molecular interactions in the system that cannot be discovered through the reaction-network alone.

2.3 REFINING THE TRANSLATION

Finally, Atomizer provides a log describing the decisions it took and ambiguities it found during the translation process; information that is particularly useful for the user to assess the accuracy of the translation and understand details about how the model was encoded. I refer to species `A_B_C` = [`A`, `B`, `C`] as a potential example of such an ambiguity: `A_B_C` is a three-part complex where only the (`A`, `B`) pair has a known binary interaction within the model. However when it comes to binding `C` to (`A`, `B`) it is not clear from reaction or lexical analysis alone whether the binding domain is located in the `A` or `B` subunit. If Atomizer cannot find any information regarding this binding interaction in external databases, the atomization log will inform the user of this problem along with the the potential binding candidates that it considered (I include a full list of the information Atomizer returns in Table 5)

In order to clear these issues Atomizer provides the option to include a model specific configuration file where the user incorporates his own knowledge about the model into the translation process. After analyzing the log, the user can choose to either accept the model definition as is or to further refine the model through the options described above. Information can be passed to Atomizer in one of two ways:

- By providing the stoichiometry of a complex (and thus leaving it to Atomizer to deduce the graph structure). In the example system, an example of user information would be to define species $A\_D = \{A, D\}$. In a normal scenario this information would be picked up by the lexical analysis engine, but if the model has non intuitive molecule naming practices this would be an alternative way of providing Atomizer with stoichiometry information.
- By providing a partial graph structure for a given complex. This is useful in cases were atomizer cannot resolve a given complexation or modification operation. In the example system this entry could provide atomizer information that $\{A, C\}$ form a non-covalent bond represented by graph `A(c!1).C(a!1)`.

2.4 ATOMIZATION ANALYSIS AND VALIDATION USING THE BIOMODELS DATABASE

I tested our system on the curated and non curated datasets from the 29th release of the BioModels database. This dataset presents a biologically heterogeneous set of models as described in Section 1.1.1. In order to quantify the validity and completeness of the atomization process I defined three different metrics. The first metric, illustrated in Fig. 13A is based on Atomizer's self reported completeness metric as defined by the number of user information tokens necessary for complete atomization. As I described in Section 2.3, these information requests come from complexes whose graph structure cannot be determined given an ambiguity in the precise nature of the complexation or modification subunits within the species. The figure reports that for more than 90% of the models in the test set, less than 5 information tokens are necessary (user information passed to Atomizer). This is a strong indicator of the degree of automation of the atomization process.

The second metric I employed was based on a common naming convention within the models in the BioModels curated dataset: Most complexes use an underscore or a dash to represent a binding relationship between two species names (for example, `A + B -> A_B`). When I atomize this reaction it becomes `A(b) + B(a) -> A(b!1).B(a!1)`, that is, the number of underscores in the species namespace decreases as we increase the atomization completion of a model. I applied this metric heuristic to a set of 341 models that were found to use the underscore naming convention according to preliminary analysis and report the results in Fig. 13B. These results corroborate the information reported in the previous paragraph: 90% of the models require no extra information according to the underscore heuristic.

Finally, in order the measure the validity of the atomization process it is necessary to compare it against a golden standard that provides a baseline for encoding structural information in an SBML model. A good proxy for this information are the BioModels qualifiers [72], a kind of annotation information that can be found in an SBML model as species and reaction meta-data. These qualifiers are used to encode the kind of relationship we can find between a model element and a real world biological element encoded in a database like Uniprot [16], KEGG [18], etc. In particular, qualifiers in the set [*hasPart, is, isVersionOf, isHomologTo*] are indicative of the structural composition of a species. Although the structural information encoded by annotation

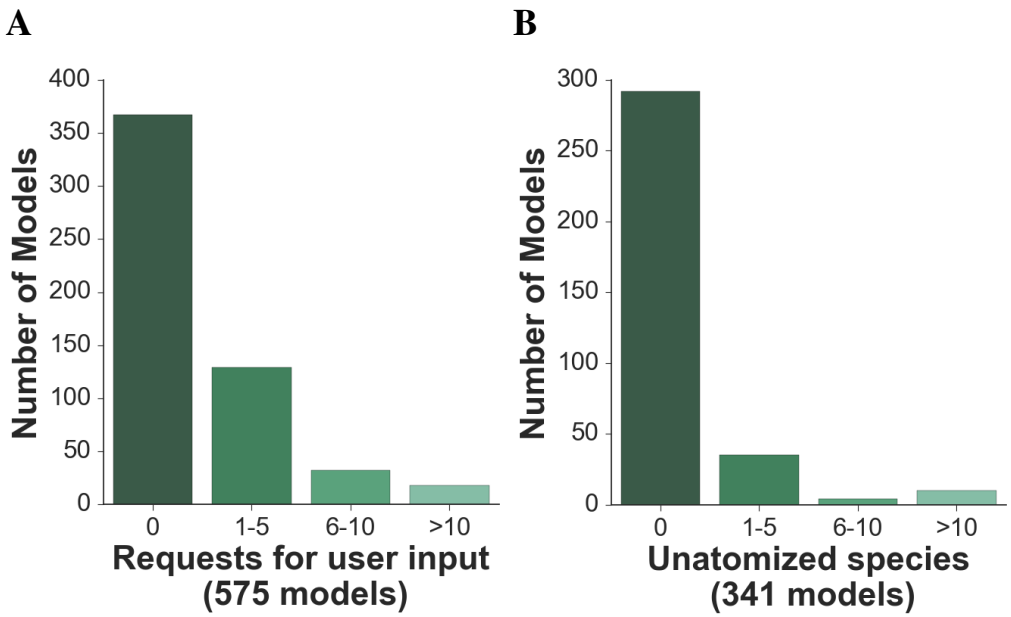


Figure 13: Two metrics measuring the completeness of the atomization process by reporting the number of false negatives. **A** is Atomizer's self reported metric indicative of the number of pieces of user information necessary to complete the atomization process. These can either be complexes whose graph configuration is not clear, or the modification of a complex where the active site cannot be determined. Chart **B** illustrates the "underscore metric", which makes use of a common naming convention to encode a complexation relationship: the underscore ($\texttt{A} + \texttt{B} -> \texttt{A\_B}$). The metric measures the number of species in a model with two or more underscores where Atomizer could find no structure information. Together, these metrics show that out of the measured models more than 90% require little to no user information in order to complete the atomization process.

information has limitations as I further explored in Section 2.1, the information that *is* present can be used as a comparison baseline.

Let us illustrate our validation test procedure with an hypothetical example shown in the left hand side of Fig. 14. This figure shows a simple reaction network where the species nodes are annotated with colored diamond markers. I use this information to establish a relationship net-

work, shown in the right hand side of Fig. 14B, where the nodes represent the set of annotated species in a model and the edges denote that the annotation of one species is a subset of the other. This is a weaker relationship in terms of establishing the structural composition of a complex than that established by Atomizer's Species Composition Table (SCT), which defines a full composition relationship between building blocks and complexes. Hence, unless the annotations contain information not present in the reaction network the annotation SCT will strictly be a subset of the Atomizer's SCT. The validation test is as follows, for every node pair $\{n1, n2\}$ whose nodes are connected by an edge in the annotation SCT, it should be possible to trace a path between the same nodes in the Atomizer's SCT. That is, there should be an intersection between the Atomizer's composition definition for each node pair connected by an edge in the annotation SCT.

The test was run by atomizing the 575 models in the curated dataset (after turning off the annotation engine from the atomization process to avoid priming the results with the information against which I am attempting to validate). Our validation test shows that out of the 575 models in the curated dataset 512 clear this condition without problems. Of the remaining 63 models, 35 contained models where annotation information included relationships not specified by the reaction network. After accounting for these relationships and removing them from the validation process the models pass the SCT comparison test.

The remaining 28 models were composed of models where Atomizer reported problems of the kind reported in Fig. 13A. That is, Atomizer had partial compositional information about certain species in the model, but not enough to resolve the structure. However, the log file also keeps information regarding those compositional candidates that Atomizer had calculated up to the point where reported the error. After recovering and accounting for that information into the validation process, those models also pass the test, reflecting the fact that the partial information was not contradicting what is known about the model.

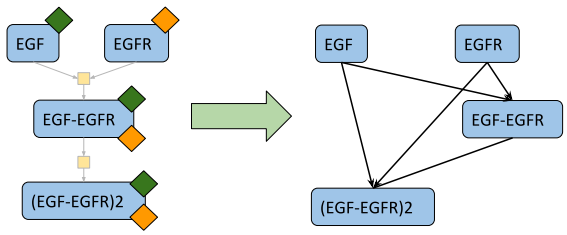
2.4.1 Simulation validation

In order to evaluate the degree of compatibility of the Atomizer with the SBML standard I tested it against the Online SBML Test Suite, a conformance testing system that allows simulation framework developers to validate the support for the SBML standard.

A

Species	Qualifier	Annotation
EGF	IS	◆
EGFR	IS	◆
EGF-EGFR	HAS_PART	◆◆
(EGF-EGFR)2	HAS_PART	◆◆

B



C

Species	Annotation SCT	Atomization SCT
EGF	[]	[]
EGFR	[]	[]
EGF-EGFR	[EGF,EGFR,(EGF-EGFR)2]	[EGF,EGFR]
(EGF-EGFR)2	[EGF,EGFR,EGF-EGFR]	[EGF,EGF,EGFR,EGFR]

Figure 14: **A**: The annotations associated to a species (represented as colored diamonds) are used to map elements in a model to real world objects. This information can be used to obtain a relationship network orthogonal to the atomization SCT as shown in **B**. **C**: Annotation vs Atomization species composition table. The annotation species composition table is created from those species that share at least one annotation. The validation test consists in verifying that the contents established by the annotation SCT is strictly a subset of that of the atomization SCT in terms of molecule type composition.

An individual test in the SBML Test suite contains the following information:

- The model encoded in an SBML format
- Settings for an ODE numerical solver, including start time, duration, number of output steps, absolute and relative error tolerances and expected observables.
- Reference time series data encoded as a CSV file (comma-separated values)

The time series generated by BioNetGen is then passed to a special purpose tool provided by the SBML team called the *SBMLTestRunner* [73]. The test runner automates the validation process by allowing the user to specify which aspects of the SBML specification will be tested and performing appropriate time series comparison tests to ensure parity between different modeling platforms. I show an screenshot of a test successfully validated by the validation program in Fig 15.

In our case, I designed a test suite that tested all features relevant to the definition of an ODE model, excluding features not supported by the BioNetGen simulator (factorial functions, variable dependent stoichiometries, differential algebraic equations and delay equations). The features that were tested (and passed) were:

- Assignment rules
- Function definition
- Initial assignment of species using constant expressions
- SBML supported representations for parameter, species, reactions
- Two and three dimensional compartments

2.5 ATOMIZATION METRICS FOR THE BIOMODELS DATASET

Once we have asserted that the atomization process is returning structural information that is consistent with what we know about the models, the next step is to characterize what kind of information do we get about the models once their structure is made explicit.

Our first approach was to evaluate the distribution of models according to how amenable they were to the atomization process. This is measured as the degree to which the reaction processes in a model can be encoded as graph operations through the following metrics:

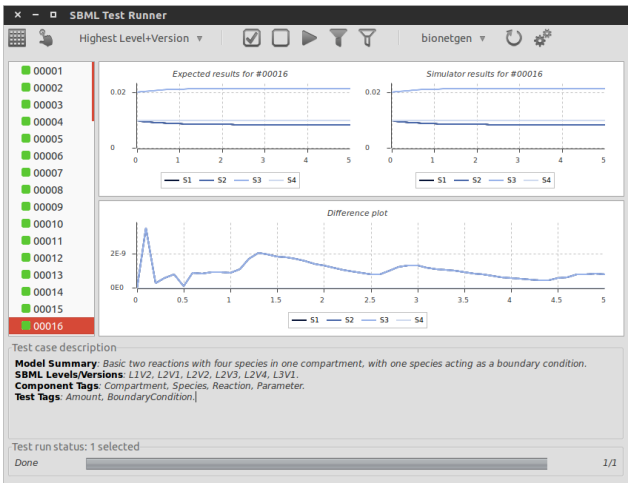
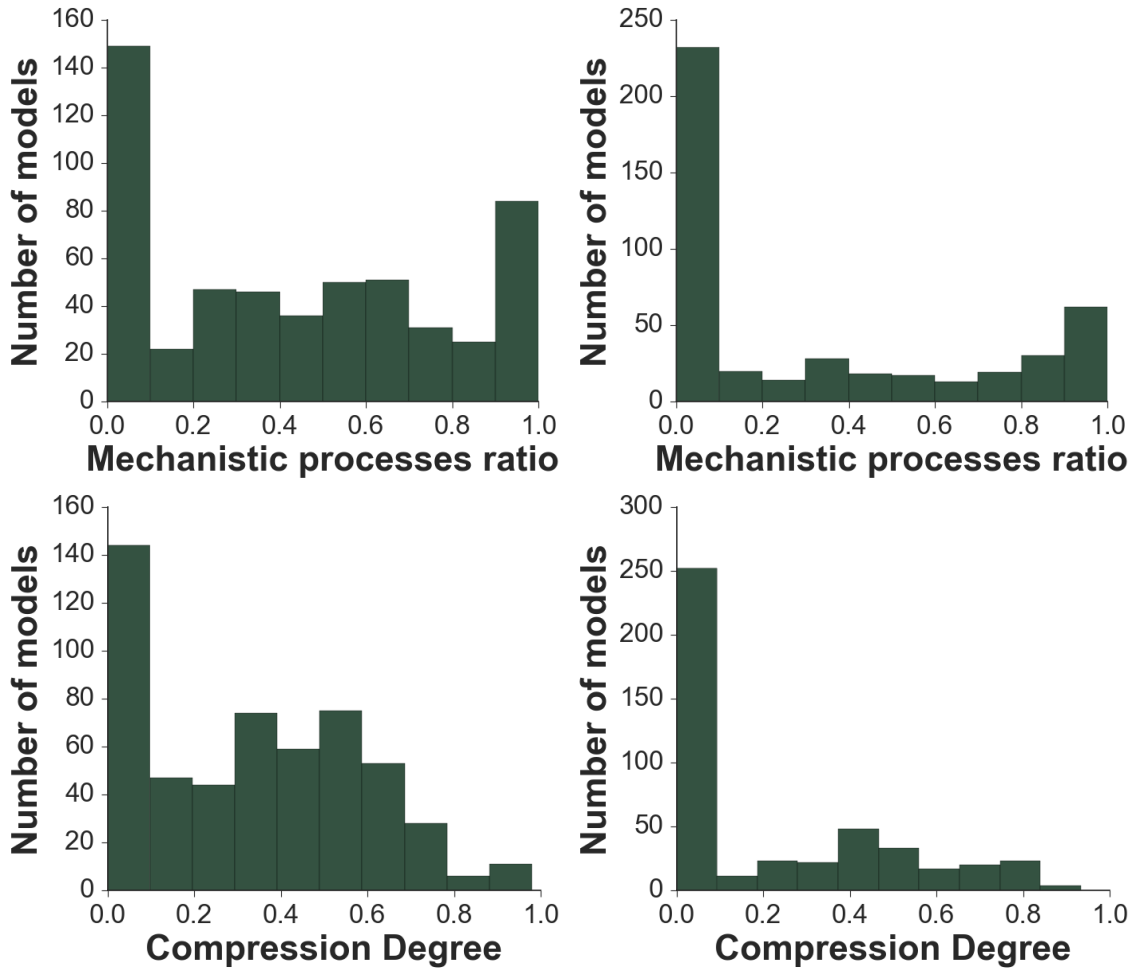


Figure 15: Screenshot of the SBML Test Runner interface

- Mechanistic processes ratio: This is defined as the ratio of reactions that contain at least one mechanistic process (creation of a bond, deletion of a bond, post-translational modification) to the total number of reactions. This is a measure of how many processes could be reconstructed as a graph operation.
- Compression is defined as one minus the ratio of molecule types in the atomized BNGL model to the total number of species in the original SBML model. Compression is a measure of how much information in the model is encoded as the combination of a few basic building blocks. A higher compression score is indicative a higher percentage of mechanistic processes.

The results (presented in Fig. 16) were obtained after applying only one round of atomization (no user information was provided to the models). The distribution of models is roughly bimodal between those models that make for good atomization candidates (models containing mechanistic processes that can be explained as the combination of a small number of molecule types) and those that are not (models containing phenomenological processes, synthesis and degradation reactions governed by complex rate functions).

Atomizing a dataset of models allows us to understand differences in the way those models encode transformations. In particular, in order to further understand what are the fundamental differences between atomized models and those models that were originally encoded as rule based



A Curated dataset **B** Non-curated dataset

Figure 16: Amenability of model in the biomodels database to atomization measured by ratio of mechanistic processes and compression. Ratio of mechanistic processes is defined as the ratio of reactions which contain at least one complexation or modification process. Compression is defined as one minus the ratio of total species in a SBML model to the number of molecule types detected by Atomizer.

models I performed a comparative analysis of the number and nature of the functional domains from each molecule type in the curated and non curated datasets when compared to that a reference set of rule-based models. I present a comparison graph visualizing our results in Fig. 17, which shows the distribution of binding and modification component across all datasets. In this comparison I am excluding those models with no mechanistic processes. Of particular interest is the number of components per molecule type found during our analysis of the BioModels database, which is indicative of the number of processes a molecule is involved in. Furthermore, analysis reveals that most BioModels rely first on phenomenological processes and then on modification processes rather than binding interactions as evidenced by the average number of binding and modification components in each database.

2.6 USE CASE: BMD19 (SCHOEBERL 2002)

In order to illustrate the translation process I present our atomization for the 19th model from the curated set from the BioModels database (BMD19) which describes EGFR (Epidermal Growth Factor receptor) signaling [10]. Figure 18 presents the final contact map for our translated model.

The atomized version is able to correctly recover critical interactions (EGF-EGFR, EGFR dimerization, EGFR phosphorylation) and molecular structure. In this model, component `m` represents phosphorylation and component `i` represents internalization. The original SBML model contains 100 species in total; however as shown in Table 2, atomization reveals that only 20 of them are basic building blocks (or molecule types in BioNetGen lingo) with the other 80 being derivatives of these.

I present in Table 2 a summary of our findings after analyzing the atomized model. Compression, as I have defined earlier, is a measure of the amount of information about the system that can be expressed as the combination of a limited set of molecules. (Which biologically would correspond to the basic set of genes and proteins whose activation and combination produce complex behavior). The number of processes refers to the number of individual actions present in a model, where an action can be one of adding or deleting a bond, adding or deleting a molecule, changing the state of a component or changing a species' spatial compartment. A single reaction can contain

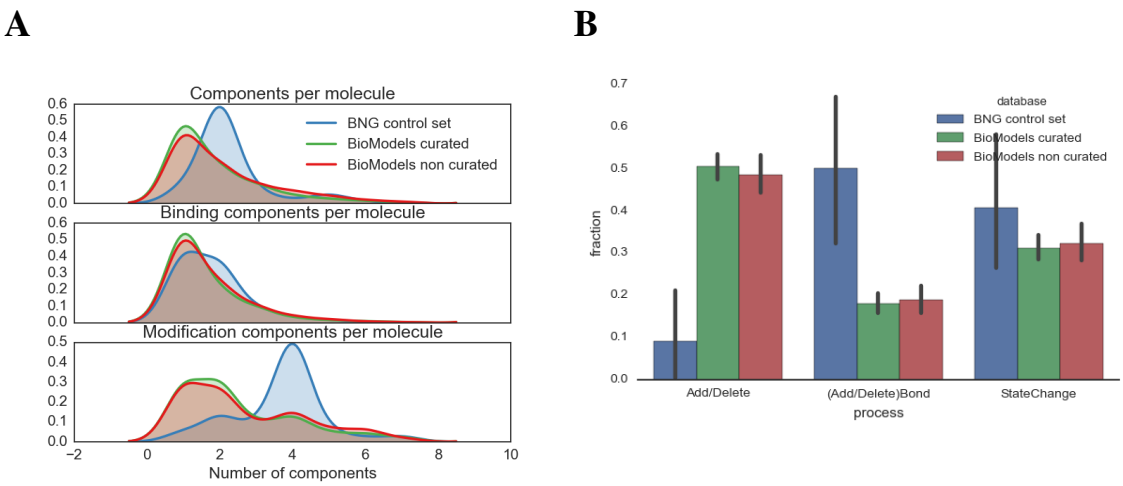


Figure 17: Analysis of the functionality of the molecule types and reaction processes found in the BioModels database for models with a non-zero number of mechanistic processes (creation and deletion of a bond or state change). In A I show the average number of components according to their functionality in three datasets: BioModels curated, non curated and a reference set of rule-based models from the literature. In B I compare the same datasets in terms of the distribution of its reaction processes separated by species creation/destruction, bond creation/destruction and state modification. Our findings show that in general RNM models contain fewer binding sites, a comparable number of modification sites and processes and a much greater reliance on phenomenological reactions when compared to models that were coded using rule-based modeling from the beginning.

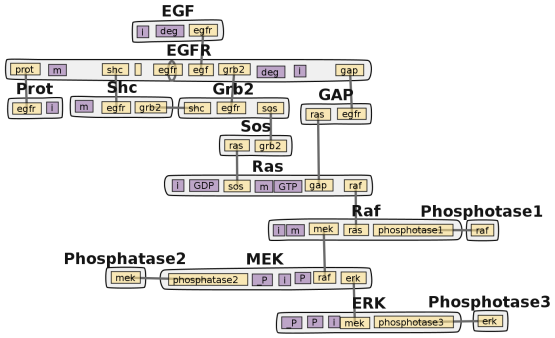


Figure 18: Atomized Shoeberl's model contact map.

multiple individual processes.

Table 2: BioModels statistics

Property	BMD19
Total number of reactions	242
Original number of species	100
Number of molecule types	20
Number of processes	567
Number of atomic processes	534
Compression	0.8

2.6.1 Analyzing the atomization log

Atomizer provides a user-readable log that quantifies the confidence it has in the different structure recovery decisions taken during the translation process, along with the different solutions and steps it took (or requests for user information) in those cases where there was most ambiguity. Please refer to Section 2.3 for more details.

For BioModels 19, there was one atomization related error that the system returns:

```
ERROR:ATO202:['EGF_EGFRm2_GAP_Shcm_Grb2_Sos_Ras_GDP_Prot', ...,
'EGF_EGFRm2_GAP_Grb2_Prot']:  We need information to resolve the bond
```

`structure of these complexes. Please choose among the possible binding candidates that had the most observed frequency in the reaction network or provide a new one: (('EGF', 'Prot'), ('EGFR', 'Prot'), ('GAP', 'Prot'), ('Grb2', 'Prot'))`

Atomizer essentially cannot resolve the structure of complexes including molecule type `Prot`, and unlike other molecules in the pathway no relevant information was found in BioGrid or Pathway Commons. In this case the user can provide information that `Prot` binds to `EGFR` (which we learn from the original publication in [10]) to complete the atomization process. An example of a user configuration file used to resolve this issue is provided in Appendix C.

2.7 BIOMODELS STRUCTURE COMPARISON

As I have described before, model alignment and comparison can be described as a two-part problem consisting of intra-model species mapping and inter-model species mapping [26]. Atomization contributes towards this process by resolving the structural hierarchy of the species within a model such that all complexes are stated in terms of their basic building blocks, thus addressing intra-model mapping. Furthermore, the reduced molecular name space provided by atomization facilitates the mapping of species between models since I only have to map together the molecule types information. Once this is normalized the inter-model mapping of the remaining complex and derived species will be automatically addressed through the intra-model mapping information.

The inter-model mapping process consists on detecting what is the overlapping set of species entities between two models and giving this information to the model alignment algorithm. For the examples presented in this thesis I developed a model alignment tool that works by normalizing the namespace used by each model included in the comparison set. The algorithmic details of this tool are presented in Section 3.4.2.

For the remainder of this section I will focus on a set of molecular structure comparison examples for models in the EGFR-MAPK signaling cascade. The comparison was done by using the model comparison tool MOSBIE [44], which is used to visually highlight the structural similarities and differences between a set of models.

2.7.1 Use case: BMD48 vs BMD19

BMD48 and BMD19 are two systems that model the EGFR signaling cascade. Additional to this BMD19 contains information for the 3-stage MAPK cascade. The models were atomized using the techniques described in the methods section together with user provided model-specific information to resolve ambiguities (user information files provided in the supplement). Finally the species namespace of BMD48 was normalized to be the same of BMD19.

In Fig. 19 I present an example of the tool applied to compare the structural information in Schoeberl's BMD19 [10] and Kholodenko's BMD48 [45]. The two models show a strong overlap in the way they design the structural interaction between the `EGF`, `EGFR`, `Shc` and `Grb2` molecules. For visual comparison I present the network visualization as presented in the original paper for each model in Fig. 20. The main issue that prevents a full reaction-network from being a readable representation and comparison tool for reaction models is its high graph size and edge-density [74], specially when compared to more compact representations like the contact map.

2.7.2 Structural comparative study of a set of models in the EGFR-MAPK cascade

Once I have brought a set of models to a rule-based form, aligning those models and comparing them on a structure level only requires normalizing the namespace associated to the molecule types used by each model in the set. Once the set of models has the same namespace I can apply the model comparison tool MOSBIE to get a graphical representation of the shared elements and interactions between the models.

I made a comparative study of a set of the 7 of the biggest EGFR models in the BioModels database. In this set I included the previously mentioned BMD19 and BMD48. Additionally I considered BMD49 [75], BMD151 [76], BMD205 [77], BMD543 [78] from the curated dataset and model MODEL0975191032 [79] from the non-curated dataset.

MOSBIE provides a similarity matrix that can be used to quantify the pairwise similarity scores between a set of models (Table 3). I show a full model pair comparison in the Appendix, Table 46, which illustrates the kind of tasks that are enabled by the atomization process. Finally, I created an aggregated model to get a graph visualization of the mechanisms from the EGFR pathway that have been studied in the literature (as represented by the seven models in our set). The aggregated

model, along with individual views illustrating the contribution from different submodels to the aggregated model, is shown in Fig. 21.

2.8 DISCUSSION AND LIMITATIONS

In this chapter I have presented the Atomizer framework for the recovery of structural and context model assumptions from reaction networks. Atomizer makes use of reaction stoichiometry and lexical analysis aided by external biological interaction databases to determine the basic building blocks in a model and to characterize the way those building blocks interact. By obtaining an explicit understanding of this information I am able to encode a model using a rule-based representation. I presented an analysis of several models in the EGFR-MAPK pathway in order to illustrate the different aspects of the translation process.

To evaluate the efficiency of our system I applied it to the curated and non-curated sections of the BioModels database: a collection of hundreds of models of various kinds like cell metabolism or signaling pathways. Our results show strong scores on most models in the database. There's a set of 130 models that are largely driven by synthesis and degradation reactions with complex reaction rates that made for poor atomization candidates, however all other models showed high evaluation on the metrics I defined with over 80% of the models analyzed recovering some structure for all species and reactions in the system.

I continue the analysis by comparing the composition of molecule types between models in the BioModels database to a collection of rule-based models in the literature. The results of our analysis suggest that while the network approach does not limit the use of explicit mechanisms, it makes it harder to recapitulate or infer them after construction. On the other hand, the rule-based framework was designed to be a more appropriate abstraction for hierarchically structured biochemical entities such as proteins and signaling complexes.

Once we have a model that was successfully atomized, the main question becomes what are the advantages the translated model have over the original one. What kind of information or analysis potential do we gain by making all the structure information that was already there, although implicit and scattered, into an explicit representation. One of the most direct qualitative advantages

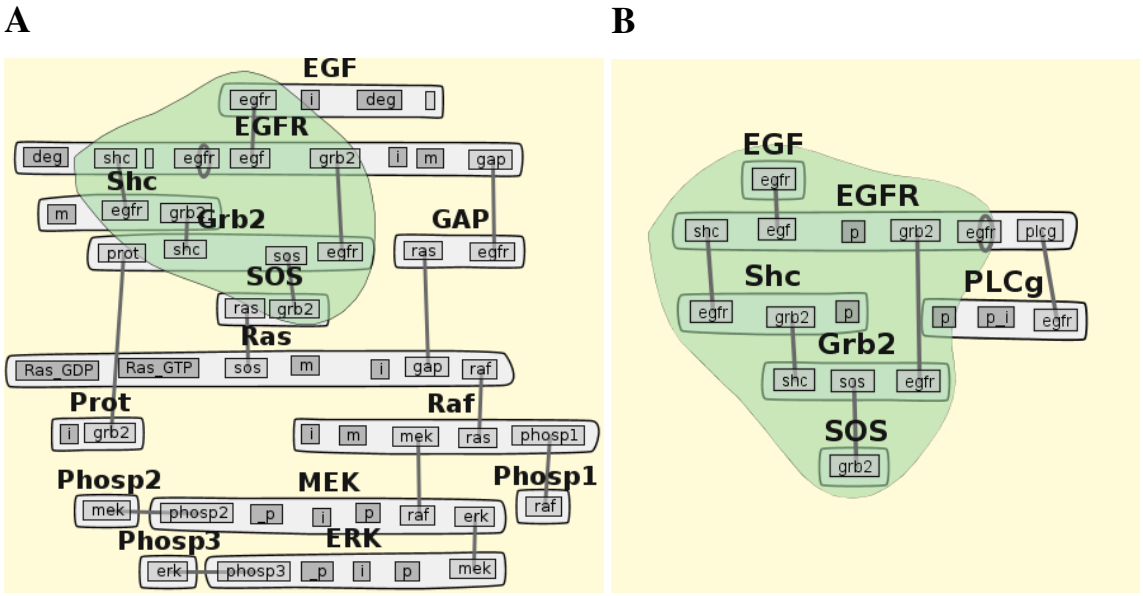


Figure 19: MOSBIE comparison of BMD19 (A) and BMD48 (B). The comparison illustrates structural similarities shared between two. A molecule-centric visualization that contains structural information empowers the user to detect the functional similarities between the entities of two models.

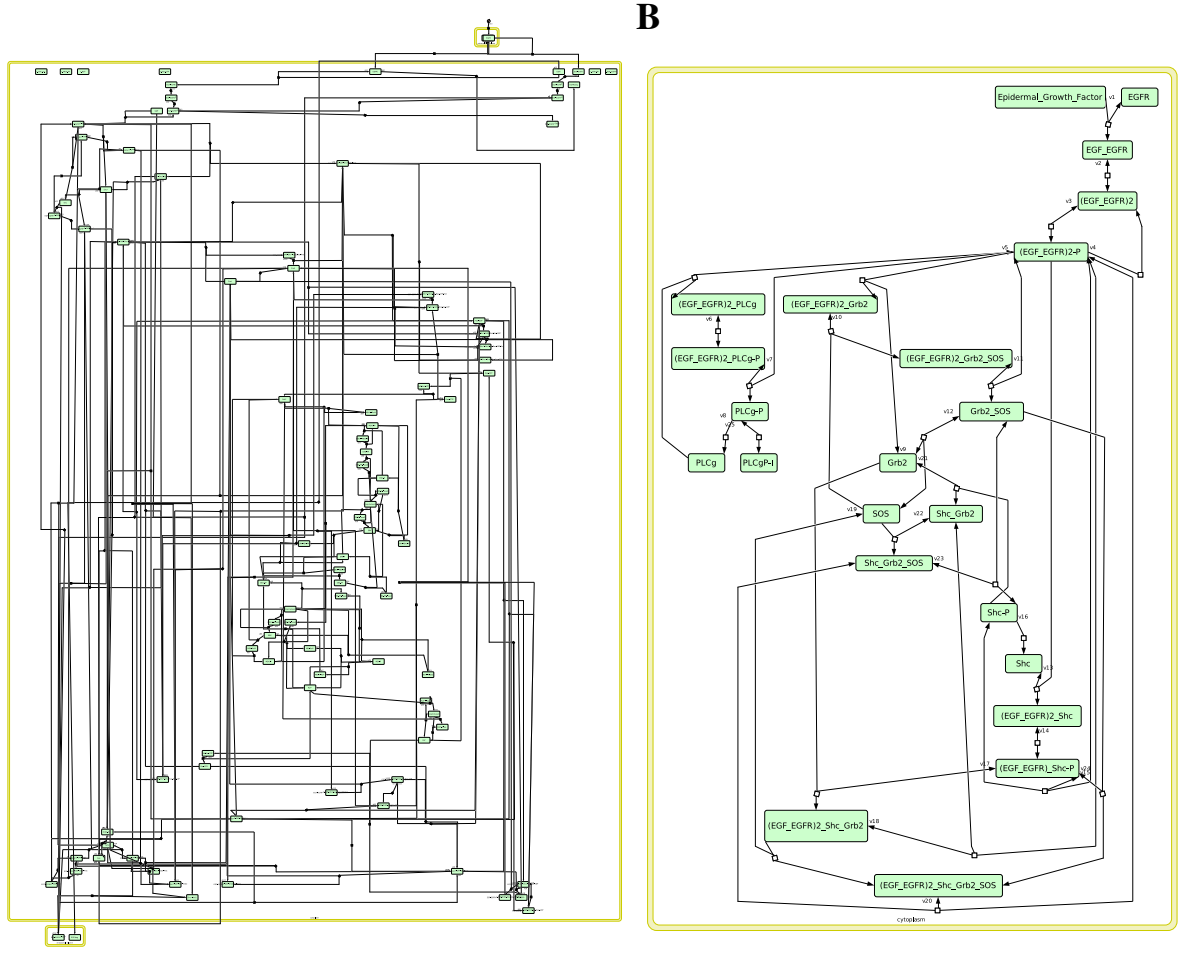


Figure 20: Reaction-network diagrams generated from the original SBML models for BMD19 (A) and BMD 48 (B). The petri-nets shown here were generated using CellDesigner. A process-centric visualization without any structural information embedded highlights the difficulty of understanding and comparing the functional similarities between two models.

Table 3: Comparison matrix for the scoring as calculated by MOSBIE for the dataset studied in this section. Score ranges [0-1], a higher score represents a better match.

	bmd49	bmd48	bmd205	MODEL0975	bmd151	bmd543	bmd19
bmd19	0.438	0.375	0.563	0.250	0.500	0.500	—
bmd48	0.714	—	0.111	0.031	0.053	0.032	0.375
bmd49	—	0.714	0.111	0.063	0.158	0.097	0.438
bmd151	0.158	0.053	0.211	0.158	—	0.613	0.500
bmd205	0.111	0.111	—	0.156	0.211	0.129	0.563
bmd543	0.097	0.032	0.129	0.097	0.613	—	0.500
MODEL0975	0.063	0.031	0.156	—	0.158	0.097	0.250

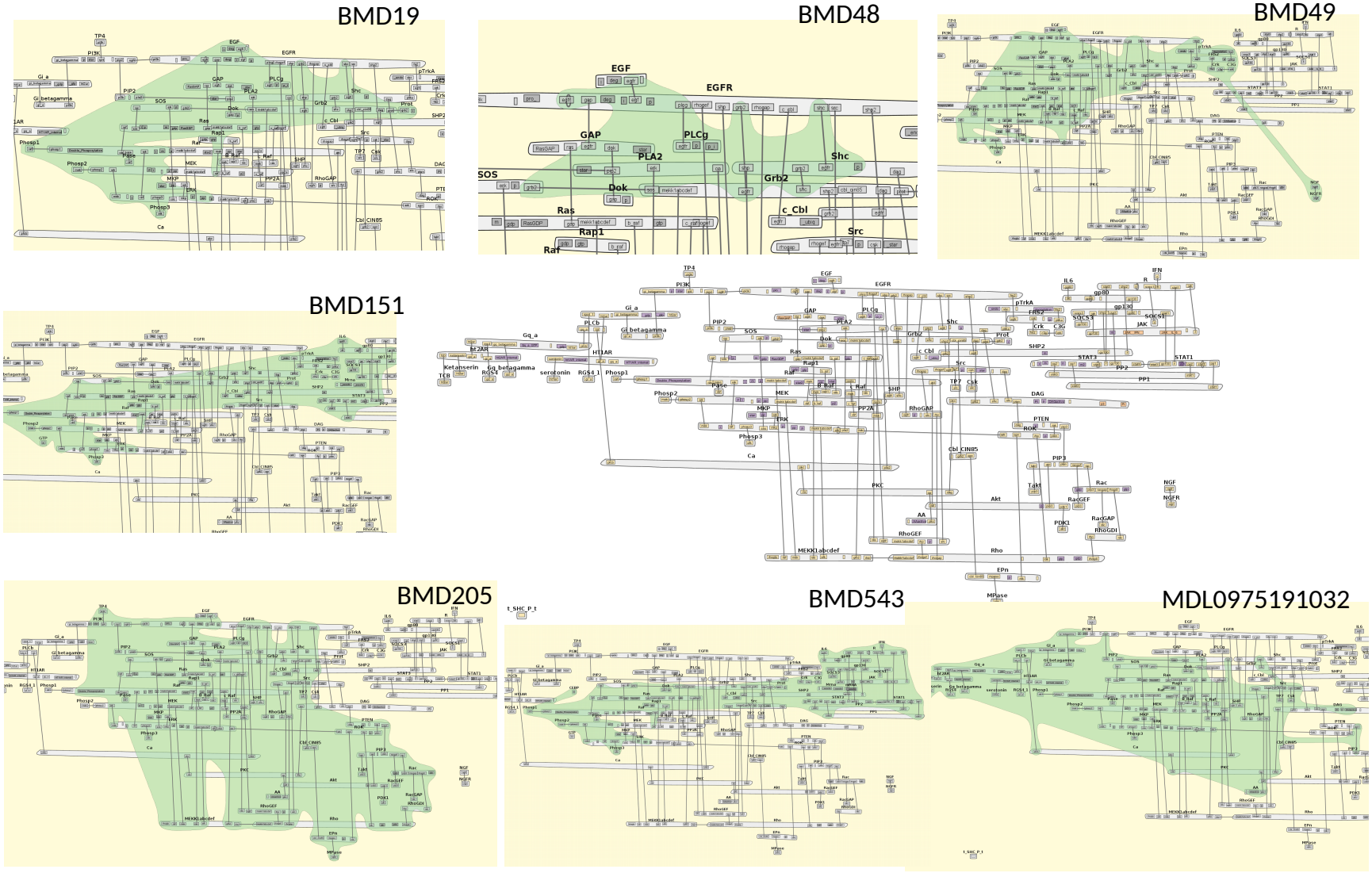


Figure 21: Comparison chart between several models in EGFR-MAPK signaling pathway using MOSBIE. The central diagram shows a contact map for a model constructed from merging models 19, 48, 49, 151, 205, 543 from the curated dataset and MODEL0975191032 from the non-curated dataset.

is the automatic generation of a contact map. Being able to visualize the different structural components in a model and their relationship to each other is a powerful aid towards model reusability.

The modularity provided by an explicit structural representation of a system's molecules can be further exploited for model comparison [44]. Information obtained from an atomized model can be used to match and compare molecules across different systems in great detail. Model comparison along with more complex operations like model alignment and fusion, can be thought as the application of two kind of operations: mapping the relationships of molecules within a model and mapping molecule similarities between models. When using RNM models these operations have scalability issues on the total number of species within a system and as such are only appropriate for small to medium models. However applying the same operations to rule-based models scale on the number of molecule types, that is, the basic building blocks, making the application of these algorithms practical for larger models.

There has been previous work related to performing model informatics through the comparison, alignment and merging of RNM models based on their annotation information. Applications like semanticSBML [20] and SemGen [19] make use of semantic annotations in order to compare models, calculate the distance between them, align them, and eventually merge them. However this is limited to those models that have been curated to contain this information. Schulz et al. report in their original 2011 paper that 69% of the elements contained in the 17th release of the BioModels database (249 models) contained SBO annotation information (one of the kind of ontologies that is used to encode information relative the structural and contextual information). The 28th release of BioModels (which was used for this work) has become substantially bigger, containing more than 500 models in the curated subset. However, as I mentioned earlier, the number of species that contain relevant annotation information is 50% in the curated set, and the number is much smaller for the non-curated subset.

Information obtained from Atomizer can also be used to give feedback to the modeling community with the structural and context information we have recovered. I describe in Section 3.4.3 an application designed to enrich the species and reaction annotation space of a model. The Atomizer can also perform other kinds of sanity checks regarding a model's consistency, including verifying for conservation of mass in a model or detecting molecule creation cycles (for example, A→B→C→A) that can inform a modeler about some of the subtler nuances of the assumptions he

or she is making about a system.

That said, structural information is not enough to recapitulate the dynamics in a system. Although a rule-based modeling encoding as a graph explicitly states how two entities interact, I have not studied the way this graph encoding can determine the *conditions* and *context* in which two entities interact. In the next chapter I will focus on how a rule-based modeling encoding can help us extract implicit reaction context assumptions from reaction-network models.

3.0 EXTRACTING PROCESS INFORMATION FRON RNM'S FOR THE ANALYSIS AND COMPARISON OF MODELS

The development of novel ways to encode information into a model specification is often driven by the need to make data more shareable and interoperable. Annotation information like that found in the SBML and CellML languages for example is designed to specify the biological composition and the identify of the compounds in a model, or the biological relevance of a reaction process, among other uses. This information in turn is used to answer questions like how are elements in a model related to each other (intra model relationships) and how are elements between models related to each other (inter model relationships). As mentioned in the previous chapter, several frameworks like semanticSBML [20] and SemGen [19] have arisen that attempt to make use of annotation information in order to aid different tasks like model composition, comparison and alignment.

A common point between these two annotation based model aggregation approaches is their focus on the use of *species* annotations. These annotations are typically references to external protein identification databases like UNIPROT and KEGG, which aid when mapping the species of different models. However when dealing with *process* annotations the options are much more limited. There are two main kinds of process annotations as enumerated below:

- Annotations pointing to individual databases that describe the biological process being modeled. For example, *Reactome* ([17]) is a database of protein-protein interactions can provide the specific biological reference for a given process.
- Systems Biology Ontology: SBO ([80]) is an annotation framework that provides semantic information about the model's components. Among its annotation categories it can label the processes in a model as biochemical reactions (binding, degradation, dissociation), transport

reactions, among others. That is, it provides a high level description of the kind of process being modeled.

A major drawback of process annotations as they are currently implemented is their lack of expressive power when defining how a set of products is specifically affected by their corresponding reactants [21]. Consider for example, the following reaction network composed of molecules $= \{A, B, C\}$ and reactions 3.1, 3.2. Even though existing annotation semantics can easily describe these reactions as processes that form a non-covalent bond, they cannot indicate whether in reaction 3.2 reactant C binds to a binding subunit belonging to molecule A or B.

$$A + B \rightarrow A\_B \tag{3.1}$$

$$A\_B + C \rightarrow A\_B\_C \tag{3.2}$$

More generally, there is not an annotation based way to map the operators defined through the Systems Biology Ontology to the reactant operands or the product results. The inability to do this in an automated way increases the difficulty of mapping together the different processes found in two reaction models.

In Chapter 2 I described how the explicit encoding of structural operations is an asset that aids in understanding the composition of the different biological complexes inside a model. Morever, an RBM representation provides a finely grained encoding of a model's structure. This structure is used in turn to precisely define the semantics that allow the transformation from reactants into products, either as complexation, post-translational modification, transport, synthesis or degradation. Finally, as we will see in this chapter, when we are able to pinpoint what are the structural changes that a given reaction operation performs we can also identify the preconditions that allow this change to happen.

The remainder of this chapter is organized as follows. In Section 3.1 I will provide an overview of how process information is encoded in RBM's and RNM's and the implications of this encoding. In Section 3.2 I will describe the methods that the community has developed to visualize process information in both RNM's and RBM's. In Section 3.3 I will present a novel visualization

technique that is optimal for the information encoded in atomized reaction-network models along with several application use cases. Finally, in Section 3.4 I will present how we have built a model analysis and comparison pipeline for RNM's based on the information we have extracted from Atomizer.

3.1 ENCODING OF PROCESS INFORMATION IN RBM'S AND RNM'S

The molecular components in a reaction can be separated into two groups: The *reaction center*, which contain those molecular components that undergo a chemical change through the firing of a process, and the *context*, which refers to those components that state the conditions for said change to take place.

I will use as a reference the EGFR signaling cascade rule-based model published in Blinov et al. in [81]. Take for example the EGFR dimerization event shown in Fig. 22A. This graph includes two EGF-EGFR complexes binding together through the `EGFR(egfr)` component (highlighted in green). The rule specification also includes another component which determines the context of the reaction: `EGFR(egf!1).EGF(egfr!1)`. The reaction would then be interpreted as follows: An EGFR-EGFR dimerization event can occur when two EGFR-EGF complexes meet with each other.

RBM uses a "*Don't care don't write*" approach to modeling context information [35, 38]; this means that when specifying a rule only the information that is conditioning the triggering of a process needs to be included (which explicitly defines the context). This is conducive to a modeling approach that encourages only introducing such conditions when they have been experimentally proven. Indeed, the EGFR molecule from the model in Fig. 22A contains multiple other components, but none of them are marked as relevant for the dimerization event. This approach allows us to determine whether the way a reaction changes the model's state influences the firing conditions of a separate reaction further down a signaling cascade.

In contrast, context information is not encoded explicitly in an RNM structureless reaction representation. Much like structural information where annotation information can supplement the lack of structure, it is possible to use annotations systems like the Systems Biology Ontology

(SBO) to more precisely identify the kind of operation being performed. Despite this, there exists no annotation standard that gives a modeler the resolution necessary to include context information as part of a reaction process [19].

However, even if context information in RNM's is not being encoded implicitly there are still implicit assumptions that are imposed by the model to the chemical processes being simulated. Consider once again reactions 3.1 and 3.2. Reaction 3.1 is interpreted as follows: When molecule A and molecule B *are both fully unbound*, a reaction which binds them may occur. Similarly, reaction 3.2 reads as: whenever *A and B are already bound together* C and come and bind to A. If for example, one wanted to remove the condition "A and B need to already be bound" for C to bind, an independent reaction stating this event would need to be included. This is the essence of the bottom-up approach to reaction-network models.

Atomization allows us to explicitly encode process information that was only implicit in a RNM, and thus, it is possible to study the properties of context information in RNM's through static analysis and RBM-based techniques. Consider for example the dimerization reaction from the atomizer BMD48 shown in Fig 22B. This graph includes two EGF-EGFR complexes binding together through the `EGFR(egfr)` component (highlighted in green). The rule specification also includes other components both in bound (like the bond `EGFR(egf!1).EGF(egfr!1)`) and unbound states (every other component in the `EGFR` molecule). This second set of components determines the context of the reaction. The reaction would then be interpreted as follows: An EGFR-EGFR dimerization event can occur when two EGFR-EGF complexes whose other binding sites are otherwise fully free meet with each other.

3.2 VISUALIZATION OF MODEL PROCESS INFORMATION

Process information can be visualized through the use of flow charts. SBGN process diagrams are a common visualization standard for RNM that is used to visualize individual reactions in detail. They can be represented as petri nets with entity nodes (representing SBML species, that is, a pool of a particular chemical complexes) and process nodes. Directed edges are used to describe the inputs and outputs of a particular reactant process. Figure 23 represents an example of such a map

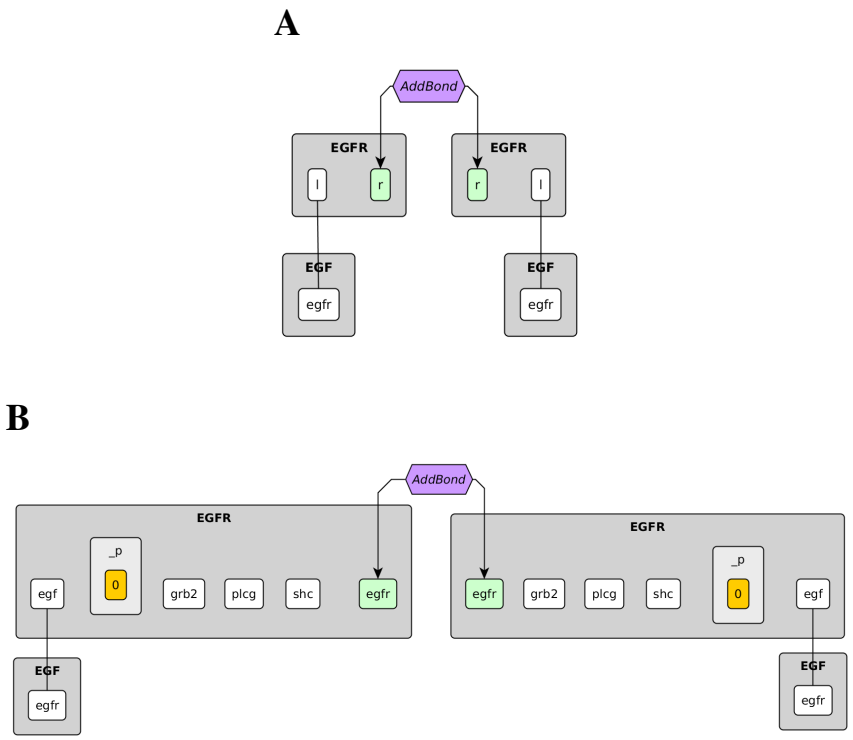


Figure 22: Reaction rule graph visualization of EGFR dimerization mechanism in two implementations of this mechanism. **A**: Shows the dimerization event from [81], a model that was encoded at an RBM from the beginning. RBM encourages a model encoding style that encourages minimizing the amount of context included as a pre-requirement for firing a reaction. In this case the only requirement for the dimerization of EGFR is for EGFR to already be bound to EGF. **B**: shows a visualization of the same system originally encoded as an RNM and later atomized. The difference between both system is representative of the differences in process encoding between RNM's and RBM's: RNM's approach starts from only permitting a very narrow definition of a given event from a single reaction, in this case, the EGF-EGFR complex can dimerize when its other components are strictly fully unbound, while RBM approaches the problem by explicitly restricting the state space through the use of reaction context components.

created for BMD48. Although process diagrams are a full visualization of the information flow in the system, their understandability and visual comprehension scales poorly with the number of

reactions [74]. Morever, much like the RNM they visualize, they do not contain any structure or context information.

In [41] we presented a process visualization approach for rule-based models called the regulatory graph. An RBM regulatory graph shows the relationship between a reaction rule and atomic patterns. It is built by decomposing the reactants and products in a reactant process into their constituent atomic patterns, namely the reaction center and its context. I present an example of such a regulatory graph in Fig. 24 for Blinov's version of the EGFR signaling cascade. Atomic patterns in a regulatory graph contain all the structural information present in the original RBM. Edges in the regulatory graph are representative of the reaction center or context relationships between a set of atomic partners and the process they are connected with. Regulatory graphs are thus a way of representing process information that scales well with the number of reactions in the model (since it scales instead on the number of rules). Indeed, when considering the atomic pattern and process nodes alone, the regulatory graph representation gives a more compressed story of the information flow in the system.

On the other hand, the number of edges in a regulatory graph scales linearly with the complexity of the context included in each reaction process. Typically, RBM's are designed with the intent of minimizing context relationships [35], however this is not the case for RNM's or RBM's that were atomized from RNM's as mentioned in the previous section: An RNM specification imposes strong implicit context constraints. When made explicit by atomization (and thus reflected in the regulatory graph) their visual complexity causes regulatory graphs to become a poor process visualization tool. Consider for example the regulatory graph for the atomized BMD48 shown in 25. Even though BMD48 models the same underlying mechanism as Blinov's version of the system, the reaction-network encodes much stronger constraints in the representation of the model that are made evident in the regulatory graph representation. This visualization makes a clear example of the issue with implicit context in RNM, it also makes regulatory graphs a less appealing option for visualizing processes for atomized models.

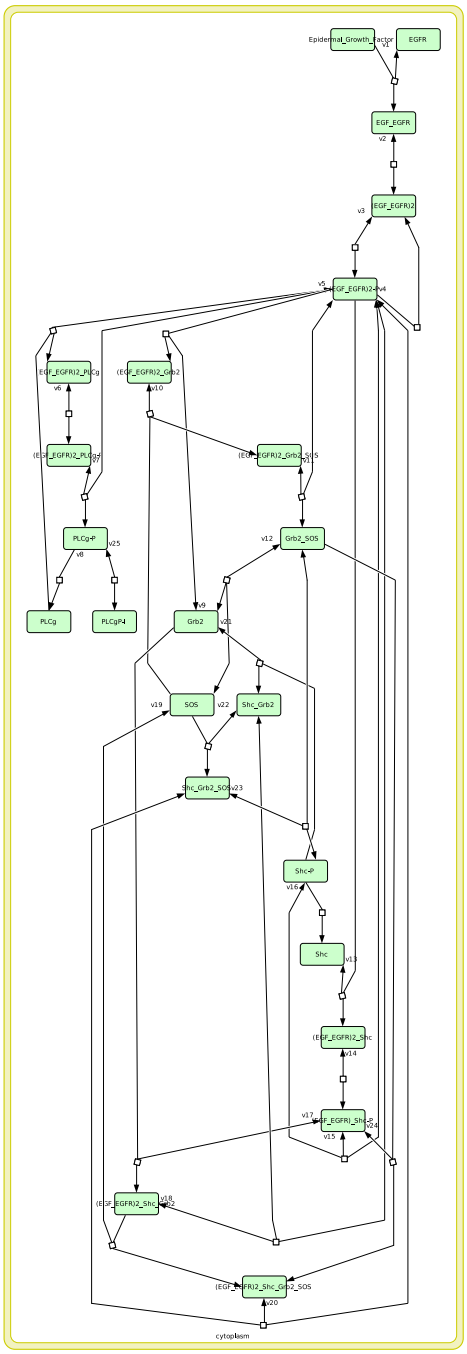


Figure 23: Pretri net showing process information for BMD48 encoded in a reaction network diagram. The graph was generated using CellDesigner

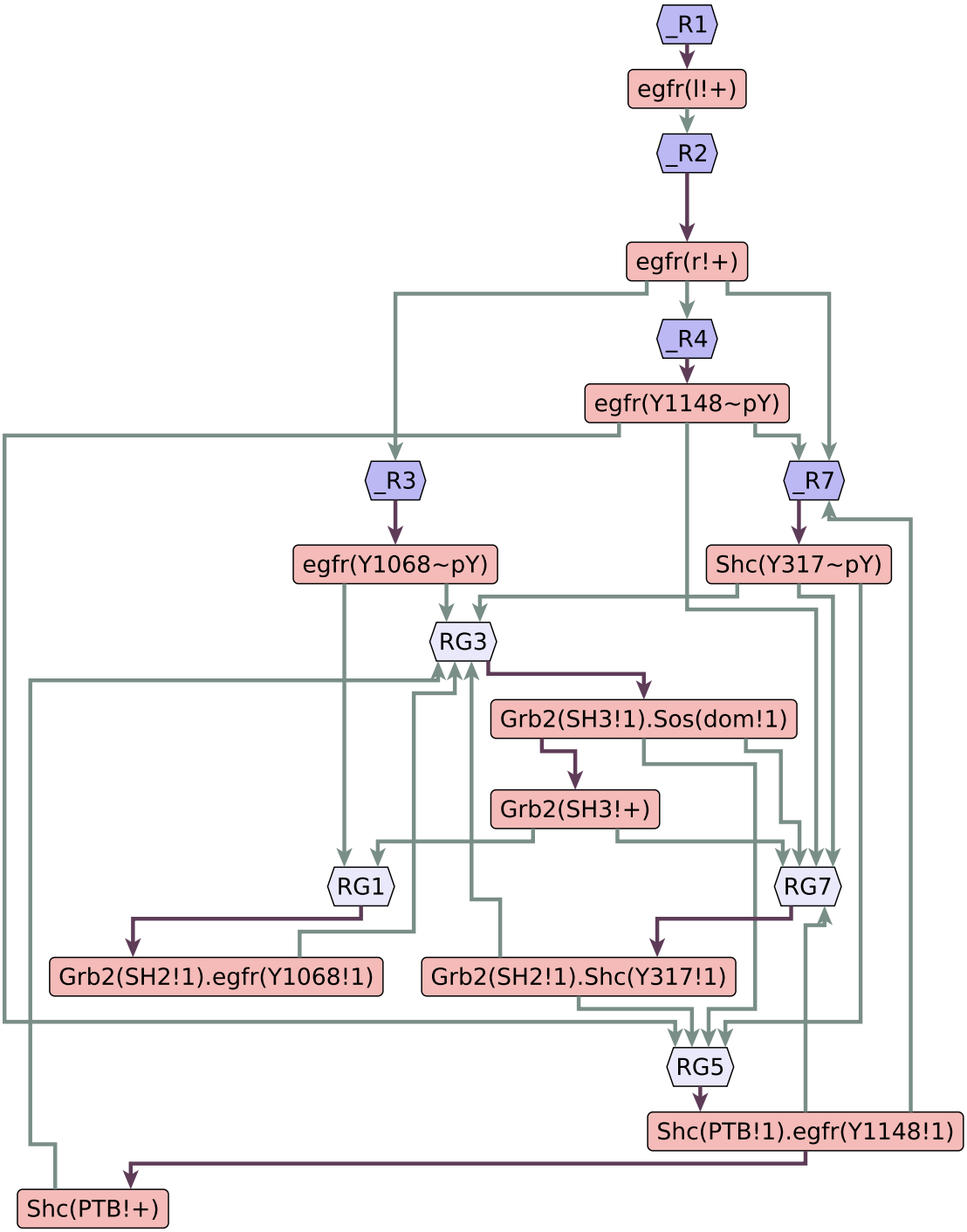


Figure 24: Regulatory graph for the EGFR signaling cascade in the Blinov 2006 model [81]. The regulatory graph is a bipartite graph where blue nodes represent processes and red nodes represent a graph motif. Purple edges represent a reaction center relationship, while gray nodes represent a reaction context pre-requirement. Regulatory graphs represent a compact representation of the process mechanisms encoded by a model.

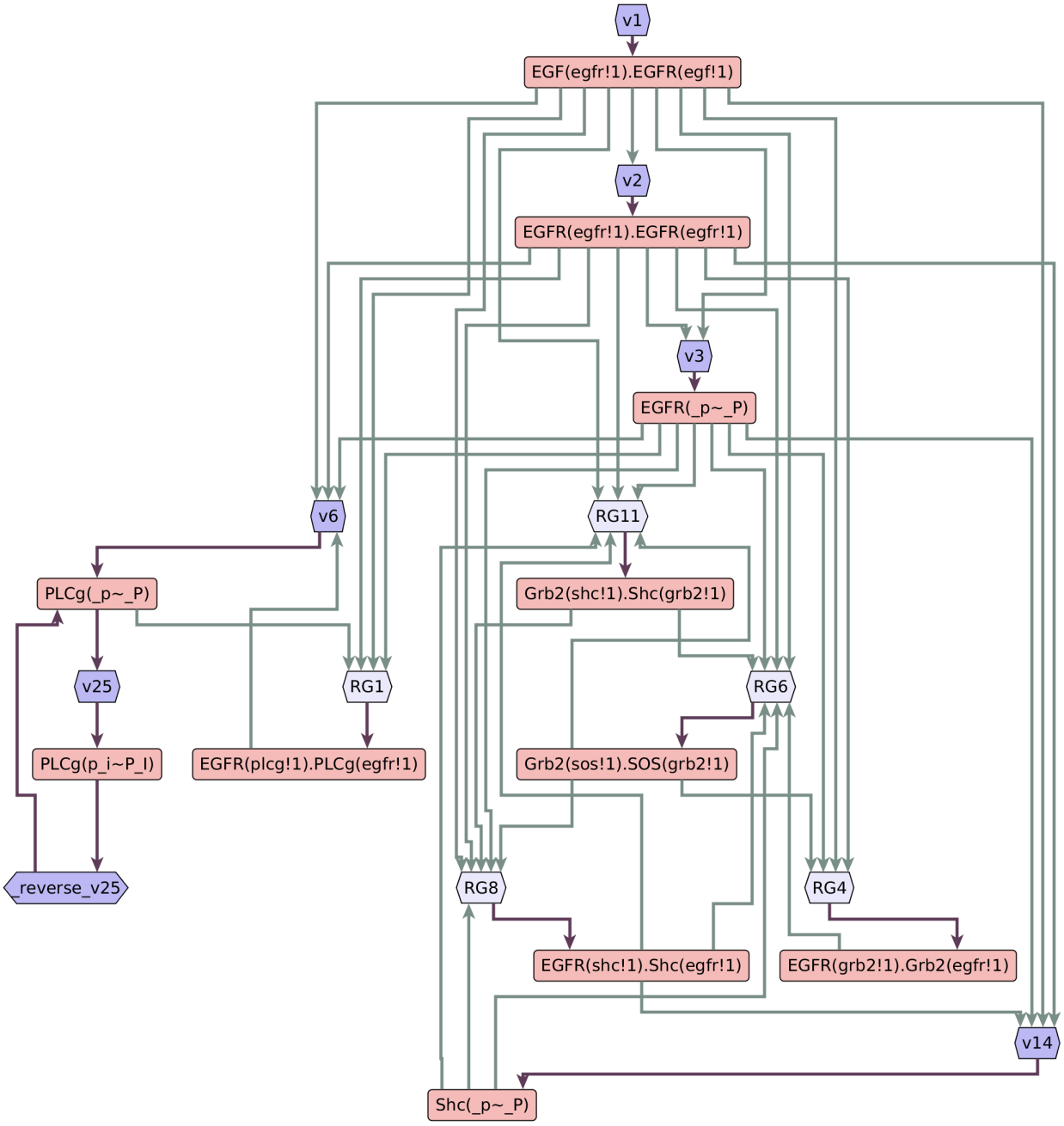


Figure 25: Chart showing process information for BMD48 encoded as a regulatory graph (RuleViz)

3.3 STATE TRANSITION DIAGRAMS

Complex dynamical systems with modular components are frequently modeled by state machines and related abstractions. In the same way, the component state space of a molecule in an RBM and the reaction rule transitions connecting those states serves as a programmable representation of a molecule's interaction dynamics [82]. As such, a state transition diagram (STD) encodes how the chemical state of a molecule is traversed.

Here I introduce the use of STD's for the representation of structured biochemical processes. In Fig. 26 I present an STD representation of the example model originally presented in Fig. 11. The STD is a bipartite graph defined as follows: every state node represents a molecular component ON/OFF configuration set represented as a bit vector. For example, the basic start node in the graph has all its components $\mathtt{A}(\mathtt{b},\mathtt{c},\mathtt{d},\mathtt{p})$ are turned off. Every process node is representative of a reaction rule in the original model that results in a molecular state transition. For example, the edge that connects the first node to the node where only the $\mathtt{A}(\mathtt{b})$ component is activated illustrates reaction $\mathtt{A}+\mathtt{B}\rightarrow\mathtt{A\_B}$ in the original model. The label in a process node indicates the *reaction center* of the reaction it came from. Through this representation I get a visual understanding of the causal relationship between the different processes that affect a molecule in a model.

I have also designed the STD to be visually and conceptually close to that of SBGN process description diagrams [83]. Process description (PD) diagrams are a standard for the visualization of reaction-network models provided by the COMBINE initiative [84] which provide a representation of the change encoded by a model by illustrating how different entities in the system proceed from one form to the other. When combined with the modularity and resolution achieved by an RBM encoding, they aid in understanding the role of an atomized set of molecular processes inside a system.

3.3.1 Use case: BMD48

I have presented multiple visualization diagrams for Kholodenko's version of the EGFR signaling pathway in previous charts. The contact map focuses on the structural components and relationships found in a model. The regulatory graph focuses on the relationship of different processes

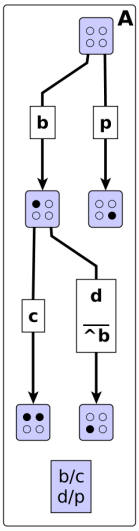


Figure 26: STD for molecule `A` from the system introduced in Fig. 11. The STD is a bipartite graph where nodes can be either molecular states represented by a bit vector or process nodes. The label in the process nodes highlights the *reaction center*, which is the molecular component that is undergoing a transformation. An annotation node is provided at the bottom of the diagram describing the bit positional distribution in state nodes.

throughout the system in terms of reaction center and context relationships. The state transition diagram is still a process-centric flow map, but it focuses instead on the role a given molecule plays throughout the model. See for example Fig. 27A. This graph quickly brings to the front the role the EGFR molecule plays in the system (EGF binding, dimerization, phosphorylation and competitive binding to $\{Grb2, Shc, PLC-\gamma\}$. The same is true for the role of Grb2 (Figure 27B shows competitive binding to EGFR/Shc, binding to SOS and return to the base state). This visualization is representative not only of the comparison prowess of rule-based model, but also of the information underneath that allowed this visualization and that would allow other kinds of static analysis not presented here.

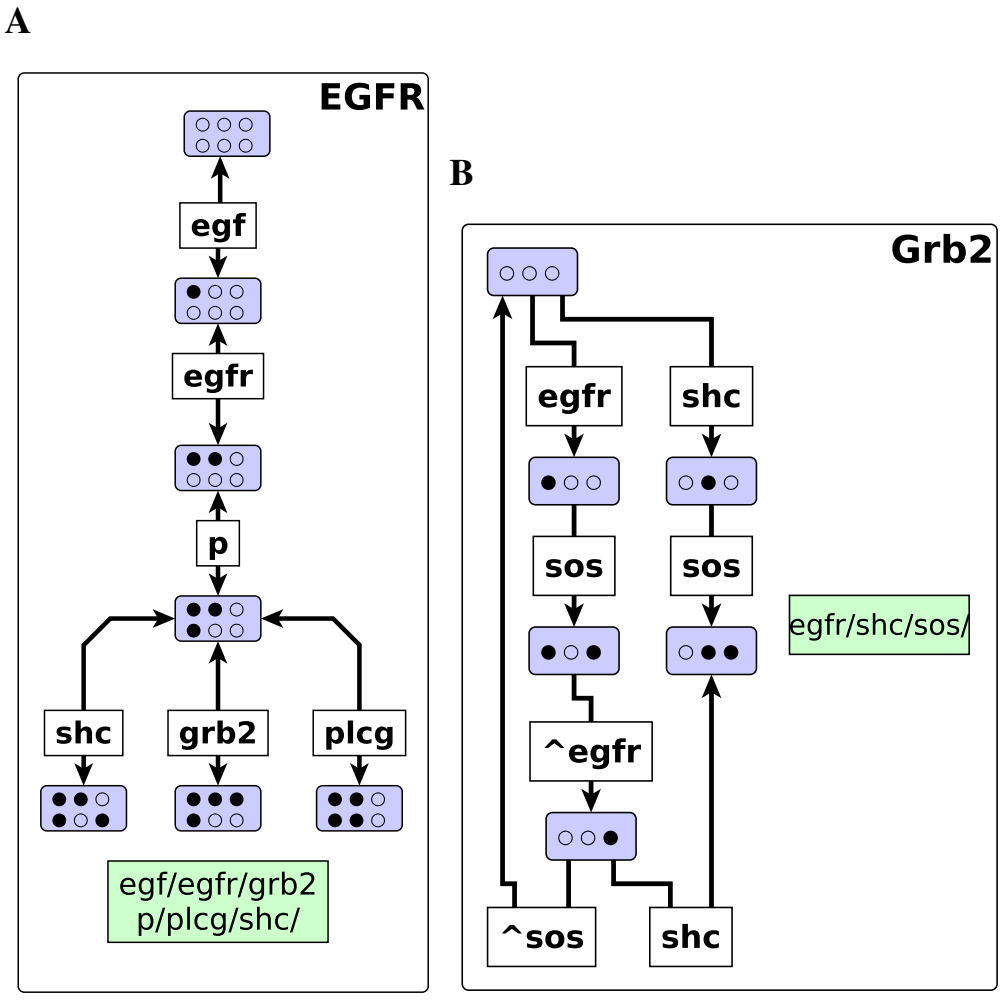


Figure 27: State transition diagram for the `EGFR` and `Grb2` molecules in model BMD48. The state transition diagram focuses on role a given molecule performs throughout the execution of a model while keeping context information and temportal dependencies.

3.3.2 Comparison Use case: BMD9 and BMD11

In order to study how the model alignment capabilities of Atomizer compare to that of other model alignment frameworks like semanticSBML. I selected the alignment example found in [66]: a comparison between BMD9 [85] and BMD11 [86], two closely related mechanistic systems found in the curated dataset from the BioModels database that model the 3-stage MAPK signaling cascade.

Although the namespace used by each model is significantly different, the biological identifier annotations associated to the elements in the MAPK cascade are the same, a requirement for the semanticSBML alignment process.

As described in Section 2.1.2, semanticSBML works by defining an intra-model species mapping based on annotation information. This mapping is defined as a feature vector which can also be directly used as a means to compare species between models; conditional on the species intersection set between the models involved in the alignment process being annotated with the same biological identifiers. The alignment in semanticSBML works by individually matching every species in the reference model to that with the closest feature vector in the comparison model. The authors of the original framework recognize that this process is highly dependent on the model having a sufficient amount of annotation to generate the intra-model mapping ontology. Furthermore this mapping only returns the set of intersecting species without analyzing the role those species perform on their respective models.

In contrast Atomizer's approach seeks to fully resolve the multi-state multi-component structure of a model's species. This approach allows us to obtain comparisons that detail the degree of overlap between two models. As an example I present an alignment obtained by comparing the atomized models BMD9 and BMD11 in Fig. 28. The atomization process itself takes the place of the inter-molecule mapping process, while the namespace normalization tool described in Section 3.4.2 does the inter model mapping. The MOSBIE comparison identifies the degree of structural similarity between the two models: in fact they model the same set of molecular interactions and modification relationships. A contact map comparison of the models illustrating structural similarities is shown in Fig. 28A.

In order to study if this similarity in the structure extended to the model processes, I used a model alignment and comparison tool (presented in section 3.4.2) to compare both models, however the analysis performed by the tool indicates that there is no difference in the structure or process information encoded by either model. The side by side visual comparison in Fig. 28B confirms this.

Given that there is no difference between the species and the reactions found in the model, the only remaining possible difference would be associated to the parametrization of the model. I show in Fig. 28C a side-by-side comparison of the STDs obtained by analyzing the MEK molecule for

each model, including the rate constants associated to each process affecting MEK. This visualization makes clear that the reaction rates associated to each model differ.

This comparison highlights the utility of a rule-based modeling representation in performing static analysis between two models and generating visualizations that highlight structural and process differences between them. Furthermore this alignment, although helped by annotation is not fully dependent on its existence.

3.3.3 Use case: BMD151 and BMD543

In Section 2.7.2 I performed a structural comparison on a set of models from the EGFR-MAPK cascade based on the molecule structure similarity. I will now look take a closer look at the two models that showed the highest structure similarity index: BMD151 and BMD543. BMD151 [76] is a model related to the regulatory mechanisms of IL-6 signal transduction and its interaction with the STAT3 and MAPK pathways. BMD543 [78] elaborates on this by studying the crosstalk mechanism of the same model together with IFN-γ signaling. As seen from Fig. 46, BMD151 and BMD543 have a very strong correspondence in their contact maps, to the point that, on a structure level, BMD151 is completely a subset of BMD543.

In order to see whether this correspondence extended to the process information in each model I studied the STD diagram from the gp130 molecule for each model, which I present in Fig. 29B. The initial series of processes gp130 is involved in is the same for both BMD151 and BMD543: gp130 reacts with JAK, with enables binding to gp80 and the binding of this complex to either SHP2, STAT3 or SOCS3. It is here when it differs and BMD151 considers and additional series of steps where gp130 can subsequently bind to STAT3 and SHP2 after binding to SOCS3. Finally, it also considers a full unbinding reaction where gp130 retyrns to the free state.

This comparison highlights the importance of being able to compare process differences between models. Even if two systems study the same biological pathways their implementation of the underlying interaction mechanism may differ in significant ways like the case presented in this section. RBM's help towards the understanding of inter-model process differences by increasing the resolution in which we can study the molecular space state traversal of a model, and thus giving the ability to perform a procedural analysis and comparison.

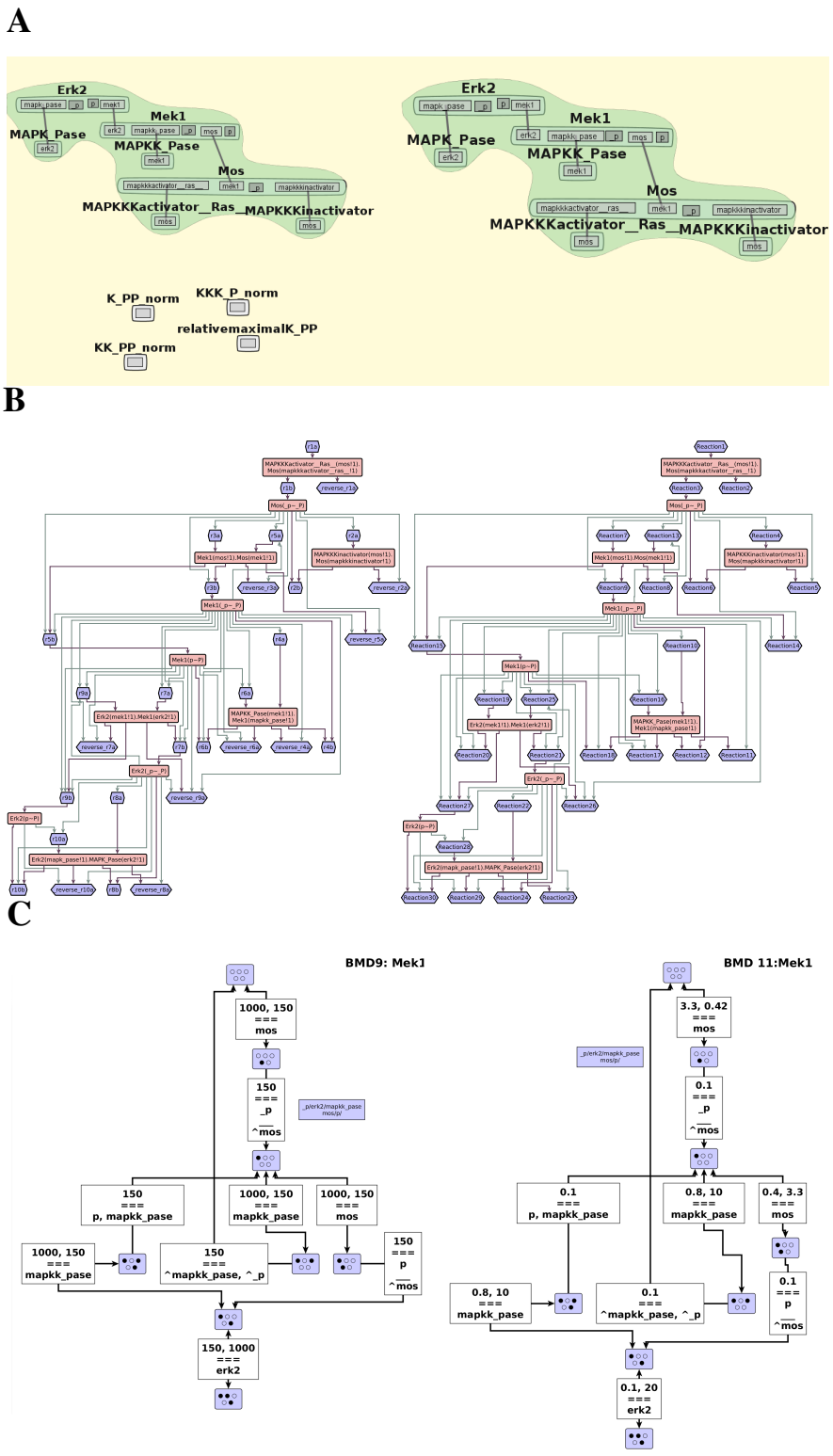
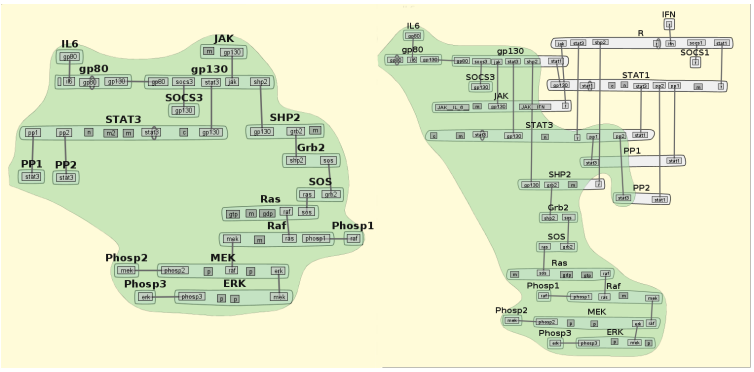


Figure 28: Comparison between the atomized BMD9 (left) and BMD11 (right). A Contact map structure alignment for BMD9 and BMD11. B: Regulatory graph comparison for BMD9 and BMD11. The regulatory graph reveals that both models contain the same mechanisms to model the MAPK signaling cascade. C: STD comparison between the MEK molecule in BMD9 and BMD11. The models contain the same interactions and process configuration, albeit parametrized differently.

A



B

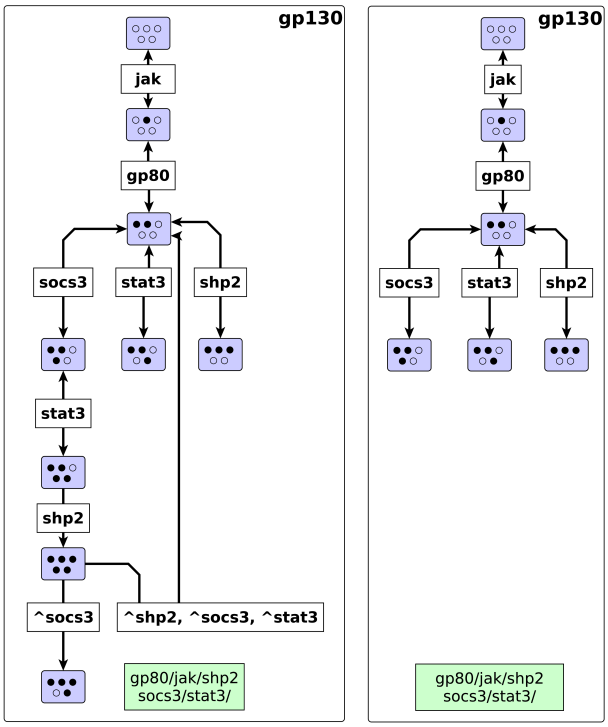


Figure 29: Comparison between BMD151 (left) and BMD543 (right). (A): Contact maps generated by the MOSBIE comparison tool illustrating the similarities between BMD151 and BMD543's contact maps [44]. The green overlay represents those elements that are the same between the two models. Structure-wise BMD151 is completely contained within BMD543. (B): State transition diagram comparison between BMD151 and BMD543. The figure illustrates the ways two models contain significant differences in how they model a particular process despite the fact they describe the same system, in particular BMD151 studies additional pathways and state configurations not found otherwise in BMD543.

3.4 A FRAMEWORK FOR THE COMPARISON AND ANALYSIS OF MODELS

The implicit structural and process assumptions I have extracted from RNM can become a powerful tool for the analysis, visualization and comparison of models. In the last section I used state transition diagrams as a way of undertanding the role of the `EGFR` and `Grb2` molecules in BMD48. Additionally, in Chapter 2 I explored the use of a contact map for the structural comparison of systems modeling the same pathway. Finally, I have covered a set of concepts (process assumptions, model state space) and visualizations (regulatory graph, state transition diagrams).

I have put together these tools as a framework for the comparative study of models, illustrated in Fig. 30. The framework is implemented as a set of web tools located in a central application hub hosted at `www.ratomizer.appspot`. *Ratomizer* allows the user to Atomize a reaction-network encoded as an SBML file. The user can then choose and subsequently visualize it using contact, regulatory and state transition diagrams (Section 3.4.4). Alternatively *Ratomizer* offers facilities to normalize the species namespace use by two models and subsequently compare the state space covered by each reaction network (Section 3.4.2). Next, the information that has been recovered by Atomizer can be partially encoded back into the original SBML model as part of an annotation backpropagation routine (Section 3.4.3). Finally, I have begun the development of a web portal for the hosting and compilation of rule-based models called RuleHub (Section 3.4.5).

3.4.1 Atomization interface

I have designed an atomization web-based tool that facilitates the atomization refinement process during which the user provides additional information to Atomizer that improves the translation quality (Fig. 31A)

In order to illustrate how Atomizer integrates user information into the translation process, I selected a model from the non curated dataset, the Chang et al. model of ERK activation [79]. The model contains 116 species and 133 reactions with no meta-data information, which makes it a good test case for how much user information is necessary to atomize a complex signaling system.

I present in Fig. 31B a table detailing the amount of information that was provided to Atomizer during four iterations of the atomization process. Below are a description of the critical errors

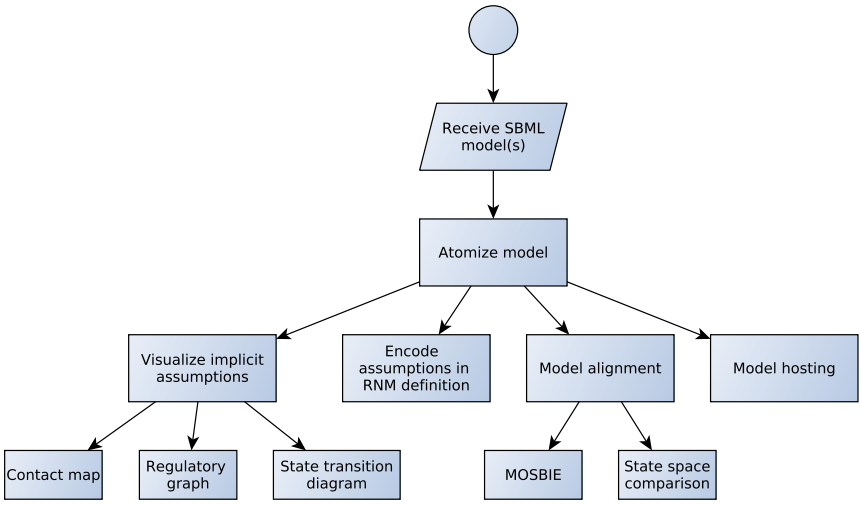


Figure 30: Model analysis and comparison framework built using the different tools I have developed around Atomizer (namespace normalization, state space comparison, state transition diagrams) and existing tools for RBM study (contact maps, regulatory graphs).

Atomizer reports during the first iteration of the algorithm with their corresponding error codes as defined in Table 5:

- There are several independent ways to stimulate the transition from GDP bound G-protein α to G-α-GTP, which causes issues in correctly resolving the stoichiometry for complexes involving G-α-GTP (Error code ATO202) . These issues can be simultaneously solved by explicitly defining G-α-GTP as a possible state for G-α.
- The model uses the '*internal*', '*active*' and '*star*' tags to model the translocation and activation of chemical complexes (Error code SCT212). Atomizer needs some guidance in understanding how this modification tag applies to each individual subunit in a complex. For example, consider complex $PKC\_Ca\_DAGactive$, where PKC possesses the active domain in question. We provide this information to Atomizer by defining the stoichiometry of this complex as $\{PKCactive, Ca, DAG\}$.

After accounting for these issues, most of the stoichiometry of the model is now resolved in the second iteration. Atomizer reports the following final critical issues:

- There are some complexes whose stoichiometry is known but their binding configuration cannot be determined. In particular Atomizer requests assistance to determine the binding of 5-HT receptors to the G-proteins one on side and the binding of Sos to Ras on the other (Error code ATO202). The reason is that there exists more than one possible graph configuration for these graphs and no relevant information was found in the reference protein-protein interaction databases.
- There are complexes where more than one possible binding configuration was discovered in our reference protein-protein interaction databases (Error code ATO102). Atomizer presents the user with the options it considered and asks him to choose one among them.

During the third iteration I use the user specification file to normalize some of the species names to use more common biological identifiers (RasGDP and RasGTP as modifications of Ras), and to perform a sanity check over some specific warnings related to the decisions Atomizer took. Briefly those are:

- Inferences Atomizer did during some of the more phenomenological parts of the model (Warning code LAE002). For example, `PA`, phosphatidic acid gets transformed into `PI` phosphoinositol, a key lipid for signaling through $\texttt{PA} \rightarrow \texttt{PI}$. Although Atomizer determined in this case that PI is a modified form of PA, we may wish to keep their definition as separate molecule types.
- Species where the lexical analysis engine offered different results than those offered by the stoichiometry analysis engine (Warning codes SCT111-SCT113). This is not necessarely a problem as many reactions complexation reactions $A+B \rightarrow Am\_B$ may include a modification element as part of the binding process, however Atomizer flags these cases for the user to review.

I present in Fig 31C a contact map for the final version of the model. The contact map illustrates critical interactions included in the model like the binding off G-protein to the 5-HT receptors and ERK activation. Finally, Fig 31D shows a state transition diagram for the ERK molecule in the system. The visualization concisely illustrates the state space navigated by ERK: Phosphorylation of ERK that leads to its binding to Raf and PLA2, along with a dephosphorylation chain. The

contact maps for each intermediate version along with the user configuration files are provided in the Appendix C, along with several additional examples.

3.4.2 Model alignment

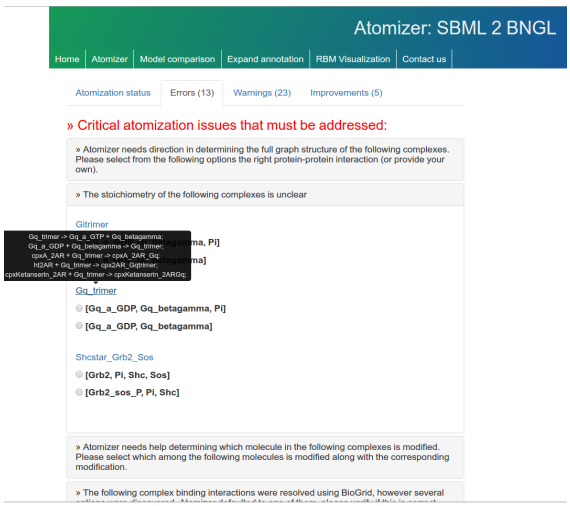
I provide a model alignment tool for BNGL models in the Ratomizer framework. The alignment and comparison tool provides an inter-model mapping interface (shown in a screenshot in Fig. 32A) where molecules are automatically mapped according to their names. The user can then further refine this process by mapping any unmatched molecules or changing the names of the molecules in the intersection set. Once this process is complete the user can proceed to the model comparison section (Fig. 32B) that enumerates the following features:

- The molecular intersection set: What are those molecules that occur in both systems according to the information provided in the inter-molecular map.
- The process intersection set: When considering a molecule as a state transition diagram, what is the intersection in the state space coverage achieved by both models. This is a useful statistic to understand whether in two models, despite studying the same set of interactions differ in the mechanisms underpinning those interactions.
- A normalized BNGL file: The model alignment tool provides download link to BNGL files with a normalized namespace between them as specified by the mapping tool. This allows for further analysis using external tools like MOSBIE.

3.4.3 Annotation propagation

Determining the atomization of a particular model allows us to understand the relationship of its constituent elements and the nature of its processes. In the previous chapter I have described how this information can be encoded using the graph structures found in Rule-based modeling, however it is also possible to partially encode this information as meta-data attached to the original SBML model. In particular there is a direct correspondence between the species stoichiometry information contained in the Species Composition Table (SCT), and the information encoded by the *Biological*

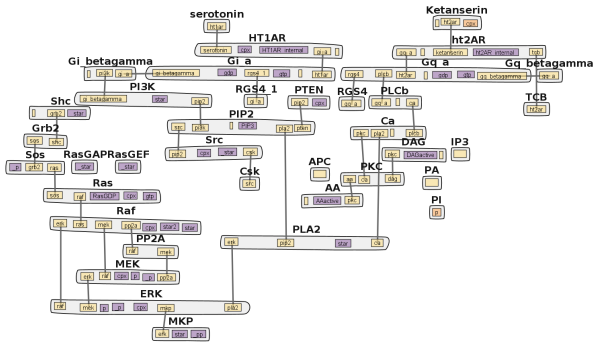
A



B

Iter.	Mols.	# errors	# info.
1st	50	11	0
2nd	43	3	14
3rd	39	1	17
4th	40	0	30

C



D

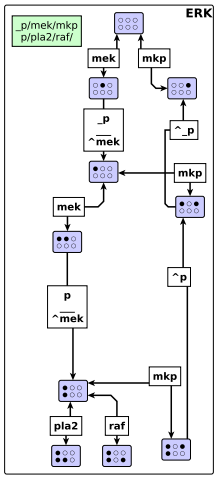


Figure 31: Atomization and visualization for the Chang 2009 ERK activation model. (**A**) An screenshot from the reaction Atomizer web interface, a tool that facilitates communication with the user to improve the atomization quality. (**B**) The atomization process may require the user to provide the stoichiometry or graph structure of a complex when this information cannot be determined from the model alone. The atomization log directs the user as to how much information is needed (# errors) which the user can provide as entries in a model configuration file (# info). (**C**) The contact map highlights several key structural features in the model like binding of the HT1-Receptors to G-protein and ERK activation (**D**) The state transition diagram provides a view into how a molecule type navigates its state space throughout the model.

Qualifiers found in the BioModels database [72]. These *Biological Qualifiers* encode a relationship between a model element and its annotation. The relationship can be, among others:

- is: This operator is used to detail the identity of a given model element by linking it to a referenced resource (biological entity B)
- hasPart: This operator is used to link a complex to the description of its components.
- hasVersion: A version operator: This relation can be used to represent a modified version or an isoform of a biological entity.

These operators can then be used to relate contents in the model with contents from an external biological database. Which database depends on the kind of entity we are aiming to encode: for molecules and species we typically link to references in GO [87], Uniprot [16] or KEGG [18]. For reactions a common database is Reactome [17] to attach a biological meaning. Additional to this we also use the Systems Biology Ontology (SBO [88]) to clarify the semantic meaning of the reactions in our model. The process is as follows: Atomizer can categorize the reactions in our system as one or more of several actionable processes add-bond, delete-bond, change-state, change-compartment, create-species and delete-species.

3.4.4 Model visualization

Ratomizer presents a web version of visualization tools I have developed in the past like contact maps [65] illustrated in Fig. 34, regulatory graph [41], state transition diagrams and support for SBGN process diagrams. The models are available both in GML as provided by BioNetGen or for visualization online using the Cytoscape JS API. The tool also provides a novel export to entity relationship diagrams as defined by the SBGN standard.

3.4.5 Model hosting: Rulehub

Rulehub provides access to a growing collection of both user-contributed and automatically translated quantitative models of biochemical systems encoded in the rule-based modeling paradigm (RBM [35, 37]). Although there is an existing need for models to exchange information through

tools like a centralized knowledge base and a common annotation system, previous attempts to develop a rule-based modeling database (GetBonnie,Rulebase) are now defunct. RuleHub attempts to fill this void by offering similar capabilities (hosting, editing, analysis, and executing) along with planned integration with the functionality currently offered by Ratomizer. RuleHub is implemented using the Google App Engine and available at http://www.rulehub.org/.

I implemented an annotation system based on the MIRIAM [89] format that describes the structure of the model and its constituent components. RuleHub contains an interface that allows users to upload pre-annotated models, to add this information during the submission process, or to automatically retrieve it server-side if the modeler used identifiable naming conventions. An uploaded model is also associated with a simulation time series and a model contact map, which RuleHub generates automatically and is a concise graphical representation of an RBM displaying the molecules, molecular components, component states, and component binding interactions. Model annotation information allows users of RuleHub to retrieve models based on any biological keyword.

The database will be populated using existing models in the BioNetGen library, the RuleBase and the GetBonNie [90] databases. Additionally I will import reaction network models that have been atomized from the BioModels database.

3.4.6 Ratomizer technical details

The Ratomizer website frontend is a web server implemented using the Google App Engine framework [91], while the backend in charge of running the atomizer, BioNetGen and visualization engines is a server hosted on the Amazon Elastic Computing service. The typical workflow for the Ratomizer platform starts with the user sending an input model (either in SBML or BNGL formats depending on the application). The frontend then communicates with the computing backend for the file to be processed. The result is then sent back to the web interface for the user to interact with it. The source code for the application is available at https://github.com/RuleWorld/ratomizer.

3.5 DISCUSSION AND LIMITATIONS

The sharing, comparison and reuse of models can only happen if the community has a common understanding of the problem being tackled. As such, it is important for a model specification to include as much information on the assumptions a system is making about reality in order to make sure that any work that builds upon these models shares the same assumptions, the same view of the world. In particular, understanding *what is in a model* require us to have a clear vision of how elements in a model are related to each other and how elements in a model relate to components in other models. In Chapter 2 I presented Atomizer, a system that allow us to extract structural information from reaction-network models, which is then used to undertand and answer what is in a model. This information is then encoded in a graph-based format as a rule-based model

Now in Chapter 3 I extend on this work to answer an additional question: *How do events happen in a model?* When comparing two models that model the same biological pathway, once we identify the intersection betweeen the set of basic species in each model, the next questions we have to ask ourselves is if there is a difference in how the dynamics of the model are represented. For example, the set of preconditions necessary for a reaction to fire. If there is a difference, then it becomes a question of how to best present this information to the user.

Classic RNM encoding formats do not allow for the encoding of process information. RNM annotations partially solve this problem [21], but they still do not have enough expressive power to fully encode a reaction as a full $\{ reactant, preconditions \} \rightarrow \{ product \}$ process because of their inability to encode molecular domain level information. As such, process visualizations that make use of this information suffer from the same limitation. RBM representations do have enough expressive power to represent this information as a rule's reaction center and reaction products.

In [41] we presented several visualization techniques that allow us to analyze this information like regulatory graphs. However this approach is ill-suited for reaction-network models that have been atomized because of the greater density of context information present in a reaction-network model specification. I introduced a state-transition diagram representation that has been enriched with reaction context information for the explicit visualization of the state space of the molecules in a model. Visualizing molecules at this resolution allow us to evaluate not only whether two models include the same system and the same interactions, but whether the temporal ordering

between those interactions change or whether a model includes mechanisms that the other does not. Furthermore it also allow us to pinpoint and visualize parametrization differences at the molecular domain resolution level. I have presented several use cases that showcase these scenarios Finally, I have developed an array of model analysis tools and an online web portal for accessing them at www.ratomizer.appspot.

The set of algorithms and tools I have presented in Chapters 2 and 3 are aimed towards empowering model informatics of reaction-network models by bringing them to a form amenable to model visualization, analysis, and alignment. In the future I would like to build on this effort by providing a framework that automates the model merging process. As of this moment, the atomization process allows a user to extract the structural and process assumptions for reaction-network models. At the same time I provide tools to visualize and quantify any differences between two models that go through the atomization pipeline, and a namespace normalization tool that allow us to start the merging of molecule types between two models. However, beyond this static and dynamic analysis of the model is necessary in order to obtain a meaningful merging of the process information in those models.

The ability to explicitly encode, study and compare model assumptions is essential when combining models from different sources and paradigms. For example projects like Big Mechanism [92] or whole-cell modeling [93] aim to put together a large meta-model from smaller systems that were developed themselves using different data sources (RNA-seq, single-cell variation, protein expression), model heterogeneous networks (metabolic, signaling, transcriptional), with vastly different time scale resolutions (metabolism, transcription, replication, growth) and where data is sparse. In such projects it is most important to encode models in a form that maximizes modularity and allows for the normalization of each model's underlying assumptions [94]. I believe that Atomizer together with the set of model informatics tools I have developed will greatly empower modelers to leverage existing modeling data to develop the models needed in these scenarios.

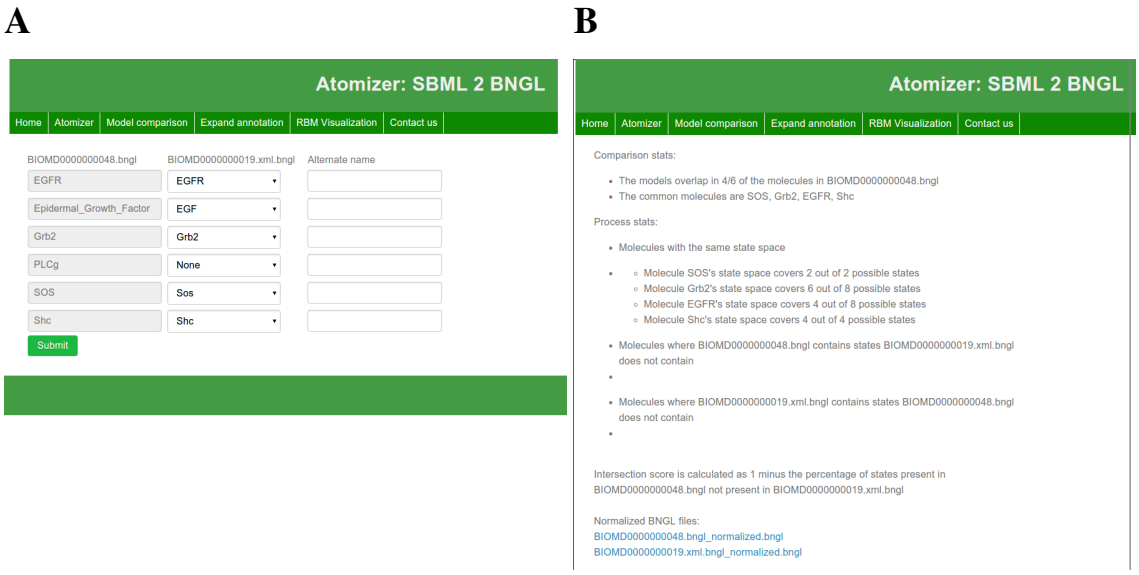


Figure 32: Screenshots from the inter-model mapping tool provided by Ratomizer. A: The tool automatically matches up elements with similar naming conventions, leaving the possibility for the user to further refine this process by changing the mapping. B: Screenshot showing summary statistics presented by the alignment tool. The summary highlights the molecular and process intersection set between both models. It also offers two files with a normalized namespace for further comparison using external tools.

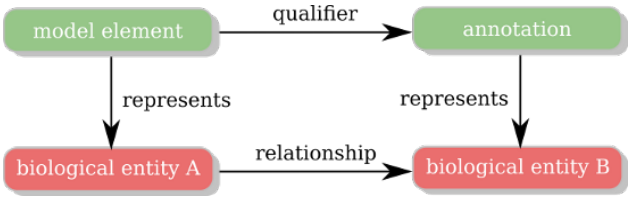


Figure 33: Annotation structure of Biological Qualifiers in BioModels.net. These kind of qualifiers define the relationship between a biological object represented by a model element and its annotation.

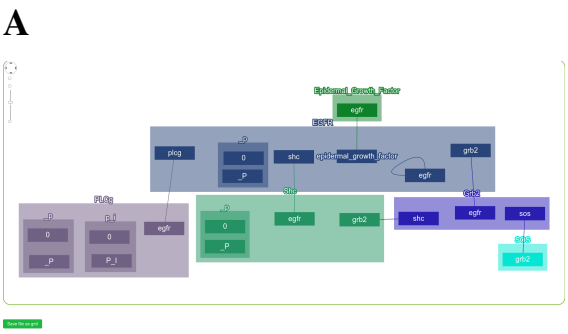


Figure 34: Screenshot from the model visualization tool available in Ratomizer. A Contact map as depicted in Cytoscape JS for BMD48.

4.0 DEVELOPMENT OF A PARTICLE-RESOLUTION NETWORK-FREE SPATIAL MODELING FRAMEWORK

The spatio-temporal heterogeneous distributions of biomolecules has an important impact on the function of biochemical systems [56]. For this reason, a wide range of spatial modeling platforms that rely on the specification and simulation of reaction-networks have been developed in the past [6, 7, 31]. However this approach becomes untenable when dealing with multi-state multi-component systems where combinatorial complexity becomes an issue, both in terms of defining a model with a significantly large reaction network and when simulating this system. In this chapter I describe MCell-R, a platform that extends the spatial simulation capabilities present in the MCell spatial simulation platform with the BioNetGen syntax to specify rule-based models together with the network-free simulator NFsim which can deal with multi-state events without the need of generating a reaction-network.

4.1 INTRODUCTION

Computational modeling has become a most important tool in characterizing the dynamics of complex reaction-networks [34]. In traditional modeling approaches a modeler is expected to define the species types of interest together with a reaction-network that specifies the kinetics of the system. Once the model has been specified it can be simulated using a number of methods depending on the resolution at which we want to resolve the dynamics: If the number of individual reactants in the system is high it is possible to extract the bulk properties of the model by numerically solving ordinary differential equations. Conversely if the number of individual reactants is small then stochastic effects have an important influence in the dynamics of the system and as such ap-

proaches that consider those effects like the Gillespie algorithm are preferable [30]. A number of platforms like CellDesigner [7] and COPASI [6] allow for the specification and simulation of reaction-networks. This approach is suitable for systems with a small to medium number of species and reactions since every molecular actor and reaction event needs to be manually specified in the model.

Spatial considerations can be added to a model by taking into account the effects of particle diffusion together with compartments and compartment boundaries that restrict the movement of molecules in a system. A spatial system can be specified by defining the species types in the system, the reaction-network connecting them, the geometry in which the simulation takes place and the spatial parameters that govern the simulation.

Finally, similar to how non-spatial simulation methods can be divided according to the resolution at which the system is solved, spatial simulators can be placed in a scale ranging from continuous and deterministic to discrete and stochastic. On the deterministic end we have modeling platforms that state a model as a set of partial differential equations (PDE), which provide a continuous description of the kinetic and spatial mechanisms governing the system's dynamics (VCell [32] is an example of a simulation platform that implements this approach). On the opposing end we have particle-based methods that instantiate every molecule in the system, an approach suitable for systems with a smaller number of agents where stochastic effects need to be considered. Smoldyn [58] and MCell [59] are examples of simulation platforms that allow for models to be defined this way.

A common thread connecting all of these approaches is that they are reliant on the definition of a full reaction-network during model specification, that is, the full pool of species and chemical reactions must be known before the simulation starts. This approach tends to be less desirable as the combinatorial complexity of the system increases. For example, a receptor with 10 phosphorylation sites can exist in $2^{10} = 1024$ states, with a proportional number of reactions. Rule-based modeling is a paradigm that was developed for dealing with such systems where species are posed as *multi-state/multi-component* entities [36]. Its graph pattern-based approach to model specification allows the full reaction-network to be generated from a much smaller number of reaction-rules. BioNetGen is an example of a framework that implements this approach.

On the spatial side a number of simulators have integrated rule-based modeling principles

into the model specification. SIMMUNE [46] uses a subvolume-based PDE approach for reaction dynamics such that the necessary equations are generated within each subvolume based on local concentrations from the global set of rules. However, this approach neglects stochastic effects and will break for models where the behavior of agents with low population pools have a large effect on the behavior of the system. BioNetGen has a compartmental extension that considers the division of the system into well mixed subvolumes [40]; however, this approach ignores all diffusion effects and considers all subvolumes to be well-mixed. The Stochastic Simulation Compiler (SSC [95]) uses a subvolume-based approach [96] combined with a spatial version of the Gillespie algorithm such that given a rule-based definition, the algorithm pre-compiles the expanded reaction-network into assembly-language for efficient simulation. However, the software is only available on a limited number of platforms, not including Microsoft Windows. SpatialKappa [97, 98] is an extension to the KaSim simulation suite that also implements next-subvolume diffusion. Unfortunately the simulator only supports a limited set of geometric shapes. In [99] a hybrid algorithm that combines SpatialKappa with the NEURON simulation platform [100] was presented that allows for more complex geometries, however it remains a prototype since it does not implement particle diffusion. Smoldyn, a particle-based spatial simulator has been integrated with Moleculizer [58, 101], a framework for generating a reaction-network from a rule specification. SpringSalad [102] is another particle-based spatial algorithm that implements a multi-state multi-component specification, while also incorporating features not present in other implementations like volume exclusion. However it sidesteps the issue of network complexity by severely limiting the context in which biological reactions can occur and the kind of biological events that can be modeled. Finally the current implementation is limited to a simple rectangular geometry with reflecting boundary conditions.

Moreover, a common problem present in all of these spatial simulators is the need for the reaction-network to be generated from the rule-specification. Even if an RBM approach facilitates the specification of a model, this advantage does not naturally extend to the simulation of the system if the reaction network needs to be fully pre-computed. For example, in [52] it was shown that generating the full reaction network for a CaMKII system on a standard 2.54GHz Intel Xeon processor would take 290 years. For this reason a number of network-free simulation approaches have been developed that avoid the need to simulate such a network [51, 53, 56].

The basic premise of network-free simulators is to individually store in memory every molecular species in the system as an independent object, such that their progress is tracked throughout the course of the simulation. The algorithm then proceeds to directly map the set of reaction rules in the system (instead of the full reaction-network) to these particle agents whenever a biological event is scheduled to occur. If an event is triggered and there are matching particles then these are instantiated as reactants to create the set of products specified by the reaction rule. This approach avoids the need to pre-compute the full reaction network at the cost of keeping the complete set of molecular agents in memory, and thus its memory cost will scale linearly on the number of rules and particles instead of the full reaction and species set. Since the number rules is typically much lower than the number of reactions, this can be a substantial improvement [35]. Some examples of non-spatial simulation platforms that implement a network free approach are StochSim [56], RuleMonkey [53] and NFsim [51].

On the spatial side there is a small number of modeling platforms that implement a network-free approach. SRSIM [57, 103] is a particle-based simulation system, that given a BioNetGen model specification gives the user the option to either generate the full network beforehand or to use a network-free approach. The underlying spatial engine behind SRSIM is the popular MD code LAMMPS [104], which ties SRSIM to the nanoscale (ns) regime of simulation, which is outside the scope of cell-signaling systems. MCell has a submodule called *Macromolecules*, that allows for the definition of multi-state entities. In this approach a molecule has slots, proxies for the binding and modification subunits typically associated to multi-component complexes [105]. A big drawback of this module though is that entities using the macromolecules module cannot diffuse.

4.2 MCELL-R: A NETWORK FREE MULTI STATE SPATIAL SIMULATION FRAMEWORK

Our main objective in this chapter is to construct a rule-based, network-free spatial modeling framework that provides accurate simulation results at the particle resolution scale, that is appropriate for systems where combinatorial complexity is a concern, and that is able to handle an

arbitrary geometry complexity. For this purpose I leverage the power of the MCell spatial simulation engine for handling the particle-resolution simulation of spatial events at the microscopic timescale. In order to enhance this functionality to one where multi-state multi-component actors and events are considered I extended the MCell biological event scheduler, which originally was contingent on the existence of a user defined reaction-network, to instead delegate event scheduling to the network-free simulation engine "NFsim". NFsim will handle the mapping of events involving multi-state objects to reaction rules, and return this information in a format MCell can understand for reaction scheduling.

Finally, on the model specification side I developed a language extension on top of the MCell language called *MDLr* that allows for the specification of multi-component actors and events by using a subset of the BioNetGen language. In order to implement this framework the following extensions to the MCell and NFsim frameworks were implemented:

NFsim:

- Encapsulation of the functionality present in NFsim as a stand-alone API such that it can be incorporated in other simulation frameworks (Section 4.2.1)
- Implementation of the compartmental BioNetGen specification in NFsim such that spatial considerations can be taken into account during the selection of graph-based events (Section 4.2.1.1).
- Implementation of a hierarchical namespace framework such that attributes and properties can be attached to complexes, compartments, molecule types, reaction rules (Section 4.2.1.2)
- Implementation of properties that made use of the hierarchical namespace framework like diffusion calculation for multi component complexes.

MCell:

- Extensions to the MDL specification language and the MCell internal model representation to define, initialize and track multi-state, multi-component objects (Section 4.2.2)
- Extensions to the MCell event scheduler such that it can handle network-free events, structured objects and their properties by communicating with NFsim (Section 4.2.4)

I present a UML class diagram showcasing the MCell-R architecture that was implemented in Fig. 35. Finally, I present some use cases and application examples for the MCell-R framework in

Section 4.3.

In the following sections I present the technical details of each of these features.

4.2.1 Extensions to NFsim

In order to leverage the simulation capabilities present in NFsim so that they can be integrated into other simulation frameworks I designed and implemented an API around the NFsim engine called *libNFsim*. libNFsim exposes the model specification setup and simulation functionality present in the NFsim suite as a set of library calls that can be integrated into 3rd party simulation platforms as a shared library. I showcase in Fig. 36 the methods available in out first release of the library. The methods can be summarized as follows:

- **Model setup**: The initialization of a libNFsim model is done through a BNG-XML model definition. This specification is then used to create the data structures that NFsim will be using during the simulation: Parameters, molecule types, reaction rules and observables. Once these data structures are created the model is check-pointed such that the user can always reset to this point. After this the user can explore different regions of the state space of the model by providing initialization setups defining the species pools or reaction rate parameters. (see for example Fig. 37A).
- **Model simulation**: libNFsim allows the user to simulate the model once it has been initialized. The configuration parameters can either be a simulation time (number of seconds or steps) or a simulation specification: The user can control which reaction to fire, like in Fig. 37B, or it can leave it to NFsim's stochastic engine.
- **Model state querying**: The user can query either static properties in the model, like its compartment structure, molecule types, etc. More importantly, it is also possible to query dynamic properties like reaction instances that are currently active (non-zero propensity, Fig. 37C) or the current particle populations (Fig. 37D) in the system.

4.2.1.1 NFsim spatial considerations

In [40] Harris et al. introduced the ability for BioNetGen modelers to specify compartmental information into a rule-based specification. Compartments are idealized, well-mixed spatial subvolumes that introduce restrictions into how molecules particle

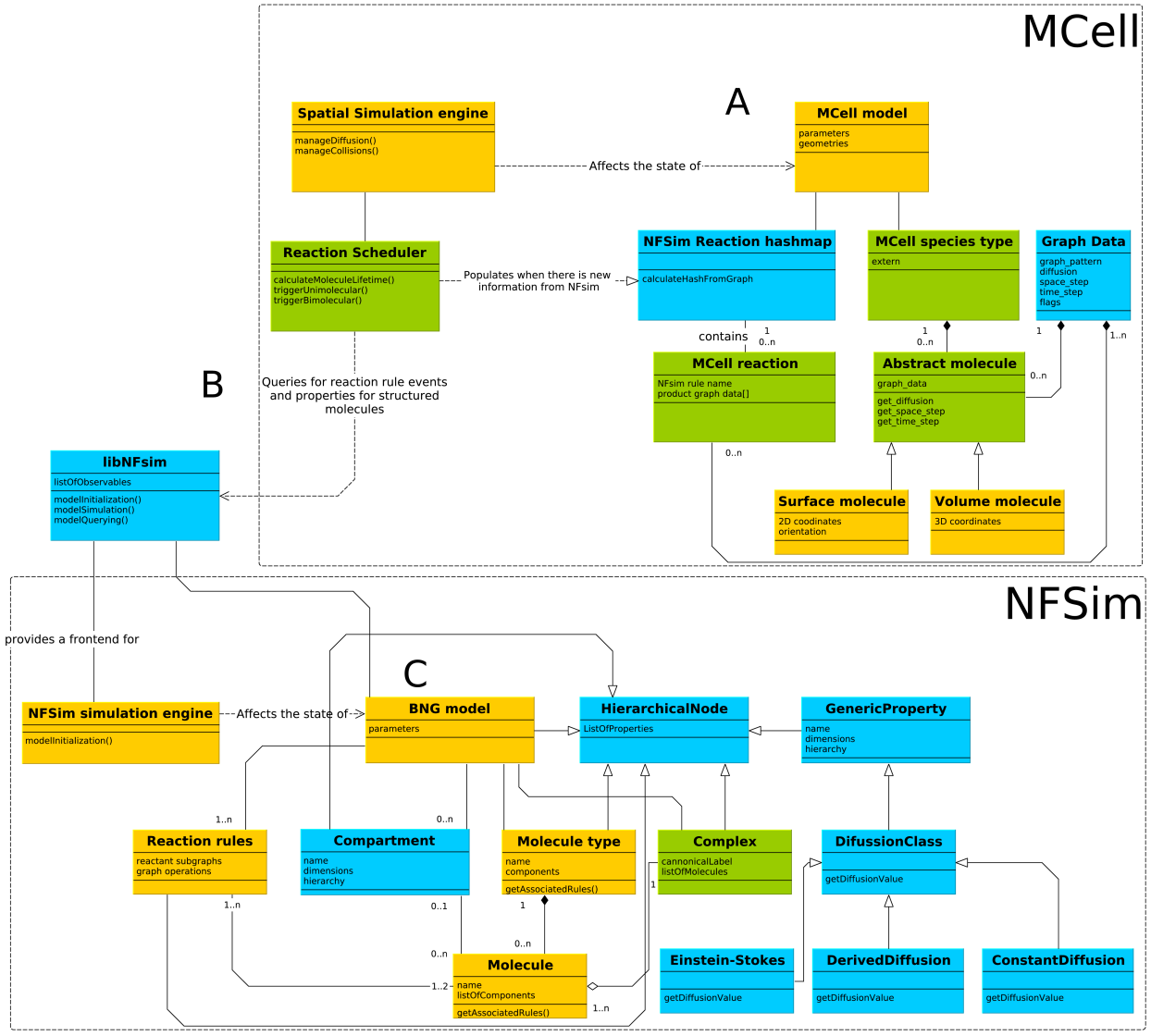


Figure 35: UML class diagram describing the internal model representation in the MCell-R architecture together with the simulator integration framework that bridges the spatial and structured models. Modules highlighted in blue were created for the MCell-NFsim integration while green modules highlights those modules that were modified. A: An MCell internal model representation contains parameter, geometry, species types and particle instances information relevant to the current state of the system. B: In MCell-R the reaction scheduler is in charge of querying libNFsim for possible reaction rule events given different modelling events (particle collision, particle death, etc.) and the graph patterns associated to the particles in those events. C: The BNG model keeps information about reaction rules, parameters, compartments and molecule types and seed species. Spatial aspects like compartments and diffusion considerations where also added to the NFsim data model.

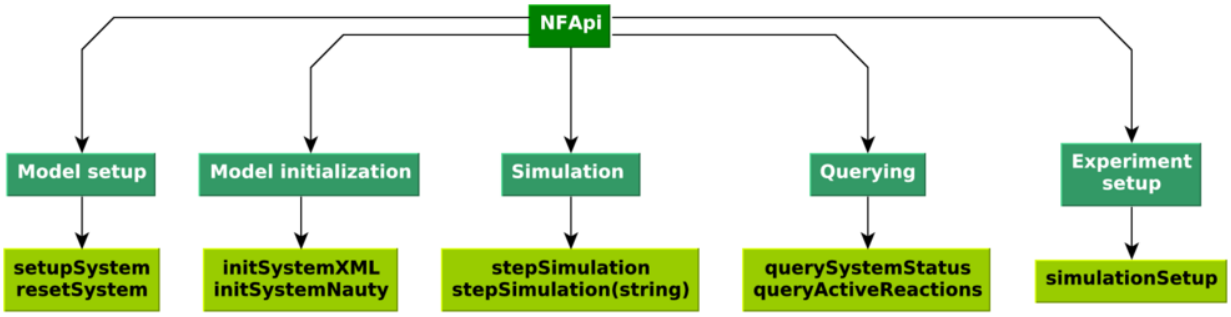


Figure 36: Diagram illustrating different methods available in the libNFsim API classified by their functionality. *Model setup* methods provide libNFsim with the basic model definition and checkpointing functionality. This is then combined with the *model initialization* methods where the user can set the initial species pools to analyze different states of the model. The experiment setup methods allows the user to set a full simulation setup with species populations, usage of the simulation methods to control which reactions are fired and the querying methods to control the kind of output that we wish to study.

can interact based on their location. A compartment is defined by its name, size, dimensionality and compartment hierarchy connecting the compartments. (for example, a 3D Extra-cellular matrix that contains a 2D plasma membrane that contains a 3D cytoplasm). A species graph pattern can then be assigned to span across one or multiple examples: see for example the compartmental patterns in Fig. 38 which illustrates how compartments provide an spatial abstraction to a rule-based graph definition. The compartmental specification had not been implemented into the NFsim simulation platform before MCell-R.

Even though most of the spatial aspects of the simulation like collision and diffusion are handled in MCell-R, it is still necessary for graph patterns to maintain a level of spatial awareness in the following scenarios:

- Unbinding of volume-surface complexes: For example, in Fig. 38, the spatial location of the volume elements from patterns ii and iv can only be determined from the compartment information present in the reactant complex.

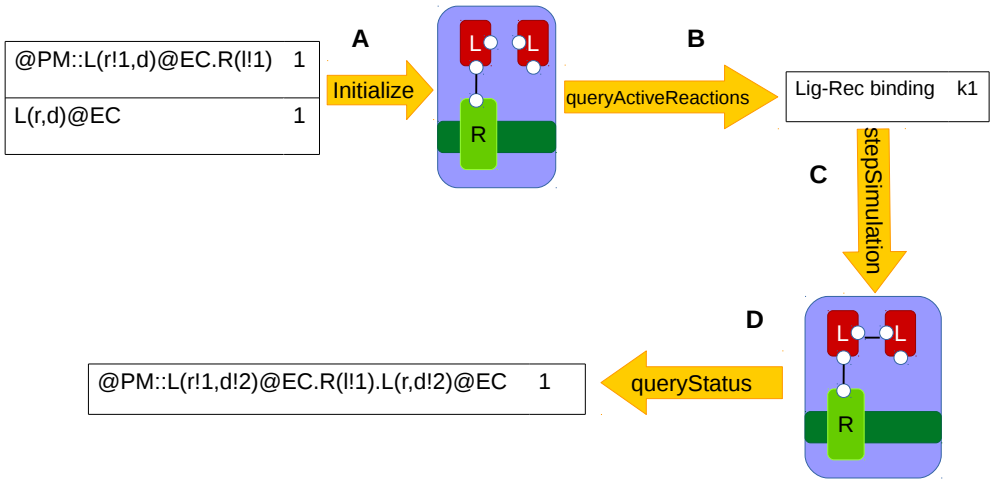


Figure 37: Chart showcasing a simulation workflow using libNFsim, including different API functions (represented as arrows) and the expected inputs and outputs (tables). **A**: Simulation initialization is done by providing a simple list of reactant populations. **B**: Once a system is initialized the user can query for the list of reactions rules with non-zero propensities and the reaction-rates associated with them. **C**: The user can instruct the libNFsim simulator to fire a specific reaction-rule; this is useful when the reaction that actually fires has to be chosen externally like is the case in MCell-R. **D**: At any time the user can query the state of the population by querying the full pool of complexes.

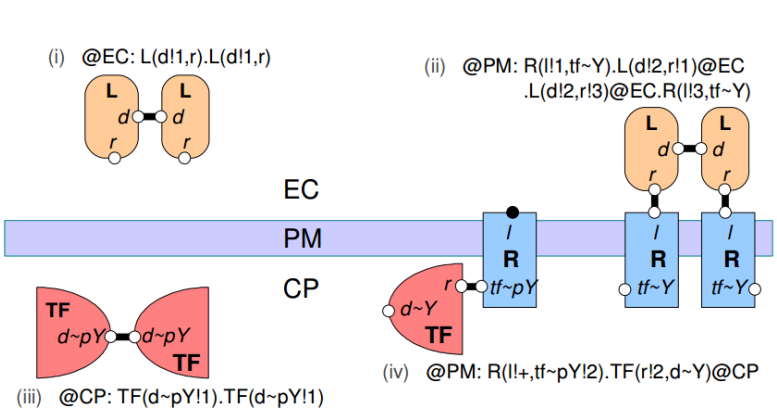


Figure 38: Spatial patterns interactions in compartmental BioNetGen (figure taken from [40]). i) volume dimer located in the extra cellular matrix. ii) Receptor-dimer complex. The species is localized to the plasma membrane but the ligand subunits are in the EC. iii) TF dimers localyzed to the cytosol (CP) iv) plasma membrane complex where the TF subunit is in the cytosol while the receptor substrate is in the PM.

- Compartment aware reaction rules: The modeler may wish to restrict the spatial scenarios in which one or more reactants interact to produce products

As such, the compartmental specification was implemented in the version of NFsim used for interaction with MCell-R. NFsim keeps compartment information in the data structure associated to molecular graph patterns and communicates it to MCell through the reaction graph strings used to identify a complex. This specification was implemented into the NFsim internal model definition for communication with MCell and handling of different events like surface-volume reactions and membrane transport.

Additionally, I also include considerations for the calculation of diffusion constants in NFsim. Before MCell-R, MCell assigned a common diffusion constant to all particles belonging to the same species type, which requires for the modeler to define *a priori* every species combination that can exist in the system. Since in a rule-based approach this is not the case I need to instead implement a scheme where the diffusion constant of a newly discovered complex can be calculated from its constituent parts.

MCell-R allows for two ways on how to calculate a complex's diffusion constant: It can either be based on the particle radius of each individual subunit or on the individual diffusion of the complex's constituent parts. In either case, both approaches follow the Einstein-Stokes equation [106] as an approximation where particles are assumed to be a sphere diffusing in an uniform medium, such that the diffusion constant can be calculated as:

$$D = \frac{k_B T}{6\pi\eta\, r} \tag{4.1}$$

where k_B is the Boltzmann Constant, T is the system's temperature in Kelvin, η is the medium's viscosity and r is the sphere's Stokes radius. For membrane diffusion I take Saffman and Delbruck's assumption of a cylindrical particle diffusing in a uniform membrane [107]. The 2-D diffusion is thus calculated as:

$$D = \frac{k_B\, T}{4\pi\mu h} * (\log\frac{\mu h}{\eta r} - \gamma) \tag{4.2}$$

Where γ is the Euler constant, h is the thickness of the membrane, μ is the viscosity of the membrane and r is the radius of the cylindrical molecule.

When particles combine to form larger complexes I assume that the complexes diffuse as though they are spherical and their volumes add upon binding [106]. Since the radius of a molecule scales as the cube root of its volume or square root of its area, for a complex composed of n molecules its radius scales as Eq. 4.3 for a volume complex and 4.4 for a surface complex.

$$r_{complex} = \sqrt[3]{\sum_n r_n^3} \tag{4.3}$$

$$r_{surface} = \sqrt[2]{\sum_n r_n^2} \tag{4.4}$$

Alternatively, the diffusion of a complex can be calculated using the diffusion of its constituent parts. For the Einstein-Stokes equation listed above the diffusion coefficient scales as the inverse of the molecules' radius. The diffusion coefficient can thus be calculated as follows:

$$D_{volume} = \left(\sum_n D_n^{-3} \right)^{-1/3} \tag{4.5}$$

$$D_{surface} = \left(\sum_n D_n^{-2} \right)^{-1/2} \tag{4.6}$$

4.2.1.2 Extensions to the NFsim data model In 35C I highlight in blue additional classes and interfaces that have been implemented for the NFsim data model. In particular our most important change was adding to NFsim the concept of hierarchical namespaces: A namespace in this context is a set of properties associated to a single element in the BioNetGen graph hierarchy. Consider for example the Einstein-Stokes formula for the diffusion of 3-D complexes. The formula is a function of the Boltzmann Constant (a system-wide property), the temperature (which can either be system-wide or compartment specific), the compartment's viscosity and the complex's Stoke-Radius, which is a function of the radius of its subunits. If we consider the diffusion of a particular complex it must have access to all the aforementioned variables. This can be solved if we consider a complex as a part of a hierarchy where it is contained by a compartment and the system-wide variables while at the same time being a container for its constituent sub-units. Moreover, a complex has properties associated to itself like its diffusion function.

Our implementation of hierarchical namespaces thus allows a given entity to access the variables associated to its containers and its subunits as requested. Container relationships are dynamic and dependent on the state of the system. A property can be assigned to a given entity in the BNG-XML model specification as a `ListOfProperties` child entry associated to a Model, Compartment, Molecule Type, Reaction Rule, Species or another Property.

4.2.2 MDLr: A format for spatially defined rule-based models

MDLr is an extension of the MDL language defined as a set of preprocessor directives that allow the user to introduce multi-state multi-component elements into an MCell model definition. MDL sections that have been extended with MDLr features, which are marked with a hash symbol, are the molecule types definitions, reaction rules, population seeding and model observables sections. I present below individual examples for each section in the following section and provide a full grammar definition for the MDLr language in Appendix D).

4.2.2.1 System constants

```
#SYSTEM_CONSTANTS
{
   temperature = T
   gamma = gamma
   kb = KB
   Nav = Nav
   Rs = Rs
   Rc = Rc
   width = 0.01
}
```

4.2.2.2 Molecule types A molecule type are the basic building blocks that are used for forming larger structured complexes. They can have a diffusion function associated to them describing how the diffusion of a particle instancing this molecule type will be calculated.

```
/* Molecule type section */
#DEFINE_MOLECULES
{
  Lig(l,l)
  {
      DIFFUSION_CONSTANT_3D = "Einstein_Stokes(rs=Rs, temperature=T)"

  }
  Rec(a){
      DIFFUSION_CONSTANT_2D = "Saffman_Delbruck"
}
```

4.2.2.3 Reaction rules The reactions follow the same syntax defined by the BioNetGen specification, however the reaction rate units must follow the MCell convention: per Mol per second for bimolecular volume reactions and per micron squared per second for bimolecular surface reactions.

```
    /* Reaction rule definition */
    #DEFINE_REACTIONS
    {
        /* Ligand-receptor binding      */
       Rec(a) + Lig(l,l) <-> Rec(a!1).Lig(l!1,l)  [kp1, km1]
        /* Receptor-aggregation */
       Rec(a) + Lig(l,l!+) <-> Rec(a!2).Lig(l!2,l!+)  [kp2, km2]
    }
```

4.2.2.4 Seed species The seed species/instantiate second is used both for model particle initialization and compartment-geometry mapping. A compartment can have the following variables associated to it: a parent (volume compartment container), a membrane (specifying the surface compartment name and the surface face associated to that compartment) and the viscosity associated to each.

Particle initialization follows the same syntax as traditional MCell particle initialization with an additional field: `MOLECULE`, indicating the BioNetGen graph pattern for this pool of particles.

```
  /* Seed species definition */
  #INSTANTIATE Scene OBJECT
  {
    EC OBJECT EC {
  VISCOSITY = mu_EC
}
    CP OBJECT CP {
      PARENT = EC
      VISCOSITY = mu_CP
      MEMBRANE = PM OBJECT CP[ALL]
      MEMBRANE_VISCOSITY = mu_PM
    }

     ligand_rel RELEASE_SITE
     {
      SHAPE = Scene.EC[ALL] - Scene.CP[ALL]
      MOLECULE = Lig(l,l)@EC
      NUMBER_TO_RELEASE = Lig_tot
      RELEASE_PROBABILITY = 1
     }
     ...
  }
```

4.2.2.5 Observables The MDL reaction output section was similarly changed to match the syntax followed by BNGL observables.

```
  /* Observables bloc */
  #REACTION_DATA_OUTPUT
  {
     STEP = 1e-6

     Molecules Rec Rec(a)
     Molecules Monomers Lig(l!+, l)
     Species Dimers Lig(l!+, l!+)

  }
```

Once a model is defined and ready for simulation, an MDLr preprocessor is in charge of extracting the rule-based graph information from an input file and generating two separate files: A rule-based model definition that is used for NFsim initialization encoded in the BioNetGen language (BNGL), and an MCell model spatial definition encoded in MDL. These model definitions can then be used as inputs for the MCell-R suite. I show an overview of these process in Fig. 39

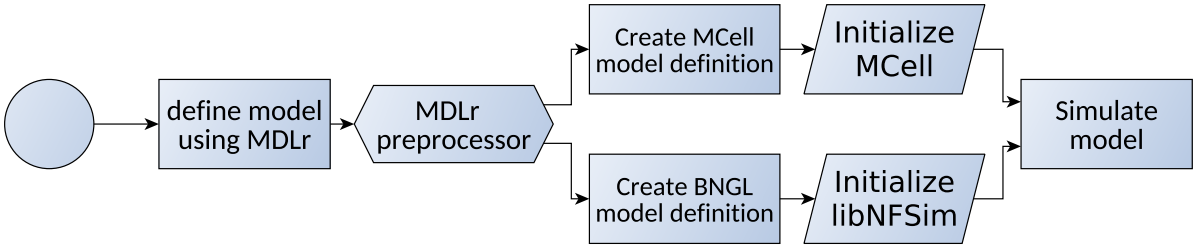


Figure 39: Flow chart illustrating how an MDL-spatial definition is processed and fed to the MCell and NFsim frameworks

4.2.3 Extensions to the MCell data model

The changes implemented in MCell-R center around calls made between the simulation engine and the reaction event scheduler. Other events like particle diffusion and collision are handled by the spatial engine, however as outlined in our pipeline specification and class diagrams, MCell will outsource all considerations about the multi-state specification of molecules, their matching to set of rules in the system and creation of reaction events as understood by the spatial simulator to libNFsim. The architecture of the system thus follows

That said, it is still necessary for MCell to keep track of the individual graph identity of each particle in the system for communication purposes with network free library whenever an event involving these species needs to be scheduled. Additionally it is necessary for MCell to know *a priori* which molecule types are considered multi-state in the first place such that their handling can be properly redirected to the network free simulator. For this purpose on the model specification level the following changes were introduced to the MDL language:

Proxy molecule types: A given molecule type can be marked with the `EXTERN` qualifier in the model definition, which instructs MCell to delegate all calculations about a molecule's lifetime and

the reactions it is associated with to an external simulation engine (NFsim in our implementation). All multi-component particle are then instantiated associated from these proxy types. An example of this definition is shown below.

```
// MDL molecule definition block
DEFINE_MOLECULES
{
    volume_proxy //proxy molecule type.
    {
        DIFFUSION_CONSTANT_3D = KB*T/(6*PI*mu_EC*Rs)
        EXTERN //new element
    }
    surface_proxy //proxy surface type.
    {
        DIFFUSION_CONSTANT_2D =
          ↪  KB*T*LOG((mu_PM*h/(Rc*(mu_EC+mu_CP)/2))-gamma)/(4*PI*mu_PM*h)
        EXTERN //new element
    }
}
```

Graph patterns: MCell associates a graph pattern string specification with every particle instance in the system that is used during communication with libNFsim. Typically this information is provided by NFsim whenever a molecule instance is created as a result of a biolgical event during the course of a simulation, but at initialization time this information has to be provided independently as part of the seed species definition. Our graph string specification is the format used by the Nauty algorithm to encode graph canonical labels.

```
INSTANTIATE Scene OBJECT
{
...
    Release_Site_s1 RELEASE_SITE //bng:@EC::Lig(l,l,s~Y)
    {
        SHAPE = Scene.EC[ALL] - Scene.CP[ALL]
        MOLECULE = volume_proxy
        NUMBER_TO_RELEASE = 50
        RELEASE_PROBABILITY = 1
        GRAPH_PATTERN = "c:l~NO_STATE!3,c:l~NO_STATE!3,c:s~Y!3,m:Lig@EC!0!2!1," //new
          ↪  element
    }
        Release_Site_s2 RELEASE_SITE //bng:@PM::Rec(a)
    {
        SHAPE = Scene.CP[ALL]
        MOLECULE = surface_proxy'
        NUMBER_TO_RELEASE = 100
        RELEASE_PROBABILITY = 1
        GRAPH_PATTERN = "c:a~NO_STATE!1,m:Rec@EC!0,"
```

```
    }
...
}
```

Graph data: As I mentioned earlier internally MCell needs to associate multi-state multi-component information to every individual particle. In order to accomplish this I created a data object associating the graph pattern information and spatial parameters that are a function of a complex's multi-component structure to a given particle.

4.2.4 Extensions to the MCell simulation platform

Model simulation wise several extensions to the MCell engine were introduced. In the introductory chapter I presented in Fig. 8 a flow chart illustrating the life time of a molecule in a classical particle-based simulation. In particular there are three stages of a particle's lifetime that I wish to bring attention to since they will need to be modified for the integration of MCell to NFsim: particle creation, particle collision and reaction triggering. I present in Fig. 40 a revised version of this flow chart showcasing the changes I have made into this process.

4.2.4.1 One note before we begin: optimizations

In classic MCell, the simulation framework makes use of a *reaction lookup table* that allows the simulator to quickly map a set of reactants to the reactions associated to it. In MCell-R, the spatial simulation engine forgoes use of this map in favor of an NFsim graph lookup table for querying the reaction events and graph structures associated to a given set of spatial structured particles. These objects could be created on demand whenever there is a given spatial event and graph data is obtained from libNFsim, however there is merit to keeping these structures in memory once created to minimize object creation and garbage collection upkeep time and to minimize repetitive calls to libNFsim.

On the flip side one of the main motivations in using libNFsim in the first place is to avoid the issue of the combinatorial explosion in the size of the reaction network and species state space: it would defeat the original purpose to store every single reaction event encountered by the simulation if the reaction-network reaches the thousands or hundreds of thousands of reactions. In order to compromise I make use of a least recently used (LRU) hashmap. The least recently used cache structure is a linked list that only keeps the n most recently queried elements in memory while

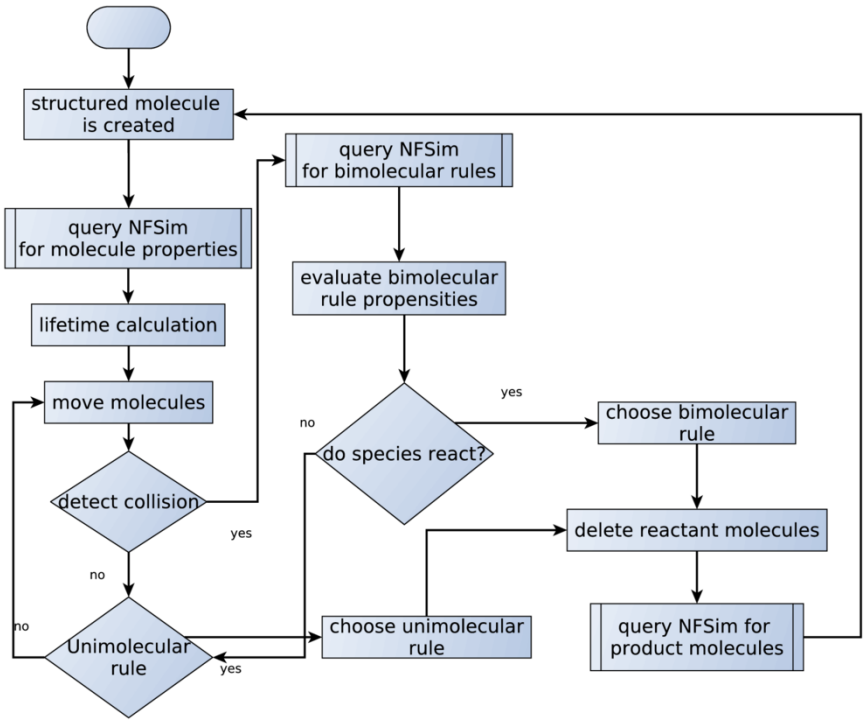


Figure 40: Flow chart illustrating the lifetime of a structured particle in the MCell-R simulation engine

purging the rest of the elements once the list size goes above this threshold (the linked list is ordered from most to least recently used). This structure will be referred to as the $NFsimHashmap$.

4.2.4.2 Particle creation In traditional MCell when a new particle is brought into the system (either as a result of a reaction firing and creating new products or from a user-defined species release source) MCell determines the set of unimolecular reactions that matches the newly created particle by comparing against a hash lookup table. Based on the reaction rates k from the matching set of reactions it calculates the particle's lifetime from the exponential probability distribution of lifetimes $\rho(t) = \frac{1}{k}e^{-kt}$. Finally, once the lifetime of a particle is over, the reaction scheduler is used to choose one among the list of unimolecular reactions associated to this particle. The reaction is then fired, the reactant particle is destroyed and new products are created.

In order to integrate NFsim into the lifetime calculation process it is sufficient for the interfacing algorithm to provide the set of propensities that belong to the unimolecular reactions matching the graph pattern associated to a particle. The process is then as shown in Algorithm 5.

Algorithm 5 MCell-R unimolecular lifetime calculation

Input: $inputGraph \leftarrow$ Graph pattern from particle being created

Output: $k_{set} \leftarrow$ Rate constants from unimolecular reactions associated to $inputGraph$

1: **if** $inputGraph \in NFsimHashmap$ **then**
2: **return** getReaction($NFsimHashmap, inputGraph$)
3: **end if**
4: Reset libNFsim
5: Initialize libNFsim model with population of one $inputGraph$
6: $activeReactions \leftarrow$ Query for all non-zero unimolecular reactions
7: $k_{set} \leftarrow \{(name_x, rate_x)\ \forall x \in activeReactions\}$
8: $mcellReaction \rightarrow$ Create reaction object from k_{set}
9: storeReaction($NFsimHashmap, inputGraph, mcellReaction$)
10: **return** $mcellReaction$

Finally, after a set number of simulation steps when the particle lifetime's is over the MCell scheduler will select one reaction rule among the set of unimolecular events associated to it. libNFsim will then be queried one last time to obtain the graph properties (graph structure string,

diffusion constant) associated to the products formed from the firing of this reaction.

4.2.4.3 Particle collision In MCell [60] bimolecular reactions are invoked any time a collision event occurs between two species to calculate the probability of any two colliding particles reacting. Computing this probability is separated into two steps: given a pair of reactants that can produce n different sets of products at rates $\{k_1..k_n\}$ I use the total reaction rate $k = \sum_{i=1}^{n} k_i$ to determine whether a reaction happens. If a reaction occurs then one of the product sets i is picked at random with probability k_i/k (like I would do in a standard implementation of the Gillespie Algorithm).

MCell ensures that the probability of two colliding particles reacting is consistent with the kinetics predicted by mass action. This probability is defined to be equal to equation 4.7 for volume-surface reactions, 4.8 for volume-volume reactions and 4.9 for surface-surface reactions, where $v = \lambda/\Delta\, t$ is the characteristic velocity of the volume particles involved in the collision, λ is the diffusion length constant and A is the collision area.

$$p_{vs} = \frac{\sqrt{\pi}k}{Av} \tag{4.7}$$

$$p_{vv} = \frac{\sqrt{\pi}k}{2A(v_i + v_j)} \tag{4.8}$$

$$p_{ss} = \frac{k}{A}\Delta\, t \tag{4.9}$$

I draw attention to the fact that all of these reaction probabilities are functions of the bulk reaction rate together with other spatial data. As such, having NFsim provide the same reaction rate constants for those reactions involved in a collision does not alter the dynamics of the system or the exactness of the algorithm. Finally, once a reaction has been selected NFsim is queried once again to obtain the structure graph information belonging to the products.

4.2.4.4 Volume-surface reactions Special care needs to be taken when simulating complexes that span across multiple compartments, in particular during complex creation and destruction events. For example, consider the dissociation of the receptor-transcription factor complex from Fig. 38. The way MCell and cBNGL ensure that the TF volume subunit is placed in the right

compartment (inside the membrane) makes different considerations: In MCell a surface molecule has an orientation attached to it that specifies whether the front of the molecule is pointing towards the inside or the outside of a given membrane. In MCell and non-structural MDL, an unbinding event can be stated thus in two different ways. Consider a species called 'surface complex' located in the plasma membrane (PM) and whose front is pointing towards the extra cellular matrix (EC) and pack points towards the cytoplasm (CP). I then apply a reaction:

```
surface_complex' ->  volume' + surface'
```

Where the surface complex is unbound into two products: a surface receptor whose front still points in the same direction as the original reactant, and a volume object that is placed in the outside of the cell. Suppose now that given the same initial reactants I apply the following reaction instead.

```
surface_complex' -> volume, + surface'
```

The resulting volume object would now be released into the cytoplasm. In MCell-R instead of alignment I make use of compartments. These two same events would be encoded by the following rules

```
volume(l!1)@EC.surface(r!1)@PM -> volume(l)@EC + surface(r)@PM
volume(l!1)@CP.surface(r!1)@PM -> volume(l)@CP + surface(r)@PM
```

Where the '@Compartment' syntax is used to explicitly encode the reaction volume or surface a given chemical will be placed in. (In the case of complexes with both volume and surface subunits, the entire complex is considered a surface reactant). The BioNetGen compartment specification uses the concept of hierarchy to indicate which is the inside and which is the outside of a given 2D membrane. However, although compartmental information is encoded into MDLr and NFsim's internal model representation, in the current implementation of MCell-R compartmental information is *not* explicitly encoded into MCell object. Instead, I perform a compartment-orientation mapping step where, given a surface-volume reaction, NFsim provides information about compartment hierarchy and the individual compartment location of each subunit in a com-

plex at reaction execution time for the purpose of determining where the products being unbound will be placed.

4.2.4.5 Spatial parameters A species type in MCell has a diffusion parameter associated to it that is used to calculate the velocity vector indicating the direction of a particle that instantiates this species type. This vector is further used to calculate other parameters like the reaction propensity during particle collision or the behavior of a particle that collides with a wall.

Since in MCell-R structured particles are in principle instances of the same proxy species type it is necessary to delegate the calculation of values that are a function of the diffusion constant to other methods. These methods will instead query NFsim for spatial parameters associated to a graph object, these values are provided to MCell during the reaction product calculation stage. In order to refactor the MCell architecture without affecting the functionality of other modules, function pointers that query for spatial parameters were associated with each *particle instance* in the simulation. The function pointers then make sure to return the appropriate values, whether it is the constants associated to the particle's species type in the case of a traditional MCell type, the values returned by NFsim in the case of a structured particle, or some other yet undefined external algorithm.

4.3 VALIDATING THE SIMULATOR

In order to validate MCell-R's model specification and simulation capabilities I made use of the `writeMDL` feature present in BioNetGen. This MDL export functionality (which I presented in [108]) is a pipeline that receives a compartmental BioNetGen model specification as input, generates a reaction-network and then uses this to construct an MDL specification based on the compartmental information which can then be fed into traditional MCell. Although the multi-state multi-component explicit specification is lost from the model it can be implicitly recovered from the model observables defined by the user.

Moreover, since MDLr uses the same syntax for its molecule types, species, observables and reaction rules sets it is possible to initialize both a cBNGL and MDLr models with minimal change.

Table 4: Kinetic and population parameters for the BLBR model

Category	Parameter	Value
Initial populations	Ligands	5973
	Receptors	300
Reaction Rates	Free binding rate	1.084e6 1/(M s)
	Cross linking rate	3.372e8 1/(M s)
	Unbinding rate	0.01 1/(M s)
Spatial parameters	EC volume	27 μm^3
	CP volume	9 μm^3

Finally, the time series generated from the observables and simulations obtained from both model specifications are directly comparable.

I have assembled a series of tests validating both basic functionality and more complex models in [109]. In the following section I present a subset of those models to compare the distributions obtained from an ensemble of stochastic simulations obtained from both approaches in order to validate the accuracy of the MCell-R implementation.

4.3.1 Use case: BLBR

The bivalent ligand, bivalent receptor model [110] is a simple polymerization model consisting of two basic molecule types: A ligand `L(r,r)` with two binding sites and a receptor `R(l,l)` with two binding sites. The objective of this test is to stress test MCell-R ability to deal with a model where the network size is unbounded. Indeed, the BLBR system can create polymer chains as long as the number of receptors in the system. The parameters are defined in Table 4. Finally the reactions contained in this model are as follows:

```
L(r,r) + R(l) -> L(r!1,r).R(l!1) kp1 #singling bound
L(r,r!+) + R(l) -> L(r!1,r!+).R(l!2) kp2 #crosslinking
L(r!1).R(l) -> L(r,r) R(l) koff #unbinding
```

Since the reaction-network generated by this model can easily reach a number of reactions in

the order of hundreds of thousands depending on the initial population pools, I instead compared the MCell-R execution to an NFsim well-mixed simulation setup for the same model. Two thousand trajectories were generated for each of the simulators and the resulting distributions were tested using the 2-sample Kolmogorov-Smirnoff test for each observable at 1.6 second intervals over the 80s of simulation time. The statistical results are $mean(p) > 0.6$ and $min(p) > 0.1$ over the set of p-values. The comparison matrix is shown in Fig. 41 for visual analysis. The visual comparison and statistical tests confirm that the well mixed and spatial network free versions draw from the same underlying probability distribution and hence produce the same time series results.

4.3.2 Use case: FcεRI

The Fcε receptor complex belongs to a pathway that plays a central role in inducing the inflammatory response of the immune system to allergens. The dynamics of this pathway and the structure of the elements involved in it are illustrated in the contact map and regulatory graph in Fig. 42. This model has a large state space: the receptor has a beta and gamma subunit that can be in different states of being unbound, bound to different docking sites, activated and inactivated. Furthermore the multivalent ligand can crosslink the receptor to form large aggregates. Network generation shows that the model contains 354 unique species and 3680 different reactions, which is still manageable to generate but presents a good first test bed as an accuracy test for the simulator.

I include the full model definition in Appendix E, however, briefly this model includes four basic components: A `Ligand` in the extra cellular matrix, `Syk` (spleen tyrosine kinases) in the cytosol and a `Receptor` together with the tyrosine-protein kinase `Lyn` in the plasma membrane. This model is a good test of the spatial accuracy in the simulator given that it contains volume-surface and surface-surface reactions of varying time scales and in sufficient numbers. I simulated 2700 trajectories for each of the standard MCell and MCell-R versions of the model. I extracted the species distributions for each observable at different points during the simulation and apply a Kolmogorov-Smirnoff to test the statistical similarity of both distributions. A matrix comparing the distribution of the observables (X-axis) at different time points (Y-axis) is shown in Fig. 43.

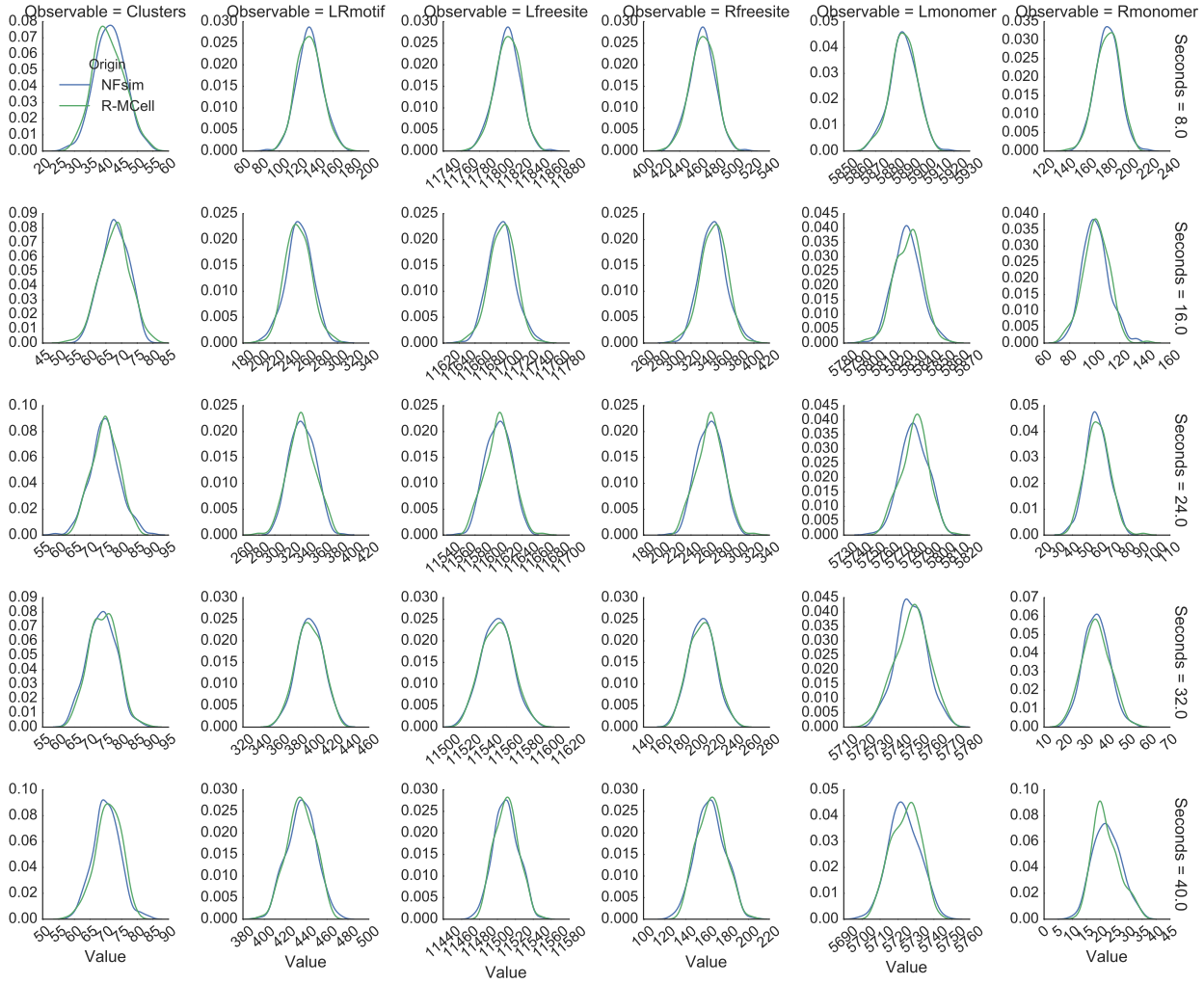


Figure 41: Matrix showing a visual comparison the distribution of different observables (Y-axis) evaluated at different timepoints (X-axis) for a spatial model of the BLBR system simulated using NFsim (in blue) and our own MCell-R simulation system (in green). Applying a two-sample Kolmogorov-Smirnoff test over the set of distributions confirms the accuracy of the results

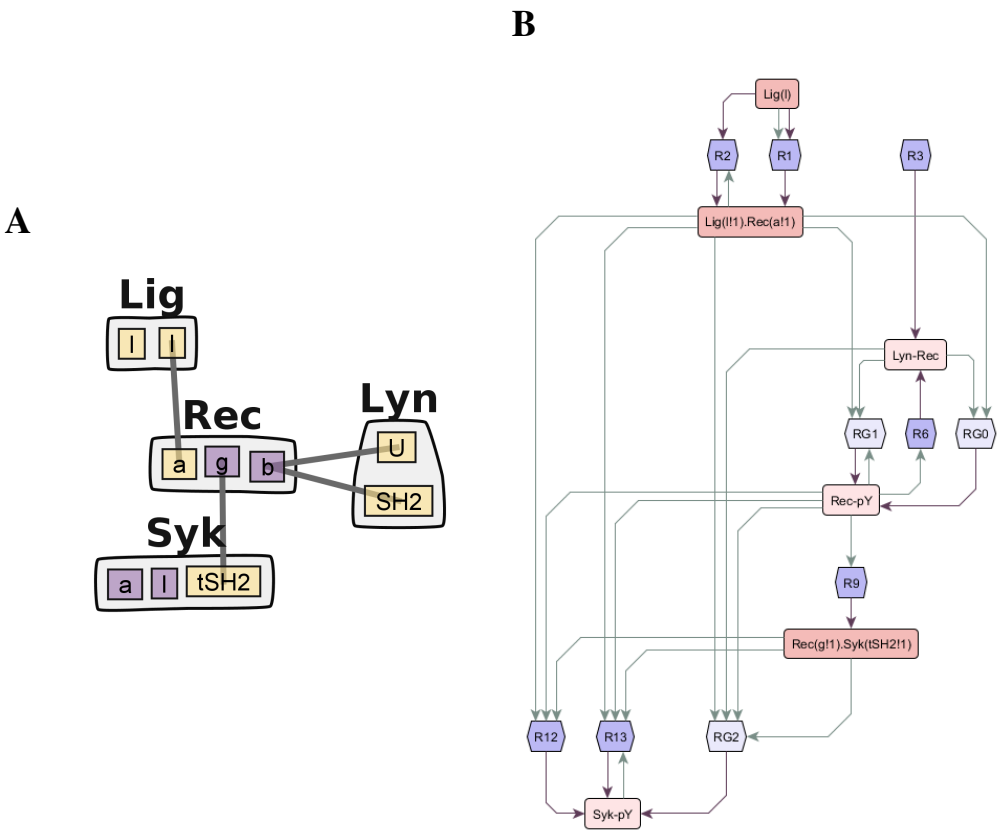


Figure 42: The FcεRI pathway. A: Contact map of the pathway. The model involves an extra cellular ligand, a receptor and Lyn in the plasma membrane and Syk in the cytoplasm. B: Regulatory graph of the model showing the information flow. The system starts one one side with Ligand-receptor pairing and Lyn-Receptor aggregation. The aggregation of these complexes leads to receptor phosphorylation on the β and γ sites, which leads to Syk binding to the receptor through its γ subunit and Syk phosphorylation.

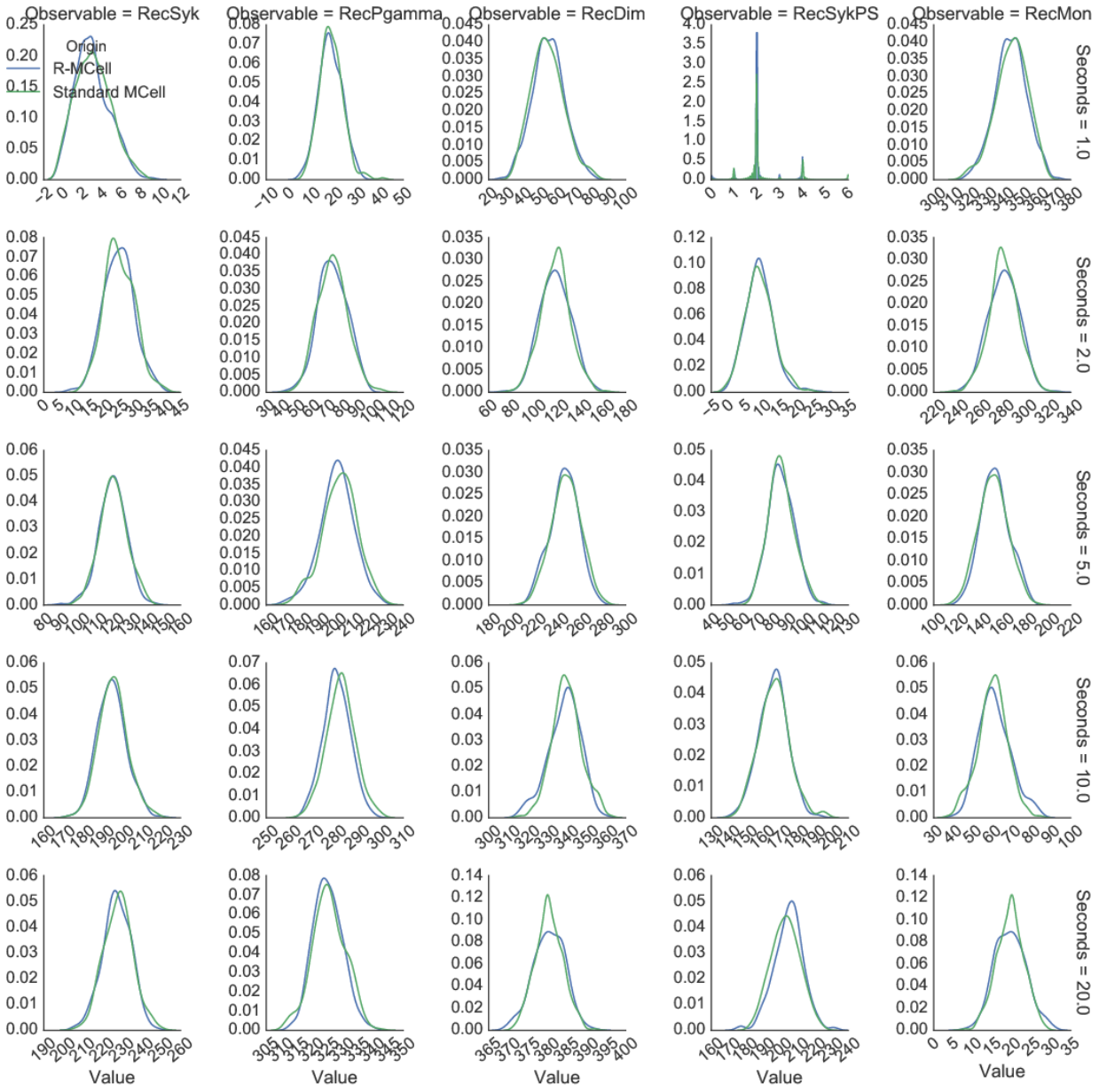


Figure 43: Matrix showing a visual comparison the distribution of different observables (Y-axis) evaluated at different timepoints (X-axis) for a spatial model of the FcεRI pathway simulated using MCell (in blue) and our own MCell-R simulation system (in green). Applying a two-sample Kolmogorov-Smirnoff test over the set of distributions confirms the accuracy of the results

4.3.3 Use case: TLBR

The trivalent ligand, bivalent receptor model [35, 110] is a immunoreceptor aggregation model based on FcεRI. This model is also the simplest model that presents biochemical reactant graphs of unbounded complexity, and as such makes for a good stress test of MCell-R's performance. I show in Fig. 44A the contact map showing the multi-component nature of the structures in the model, and in Fig. 44B the rule events contained in the system. I include the full model specification in Appendix F.

The comparison matrix shows agreement between the well-mixed simulation and MCell-R up to the 8 second mark, however, after that point the complex observable distribution starts showing strong differences, which eventually influences other distributions. An analysis of why this happens reveals a limitation of our current approach and the way time steps are configured in MCell:

MCell simulation's time step is a user-configurable parameter that is supplied to the simulator at the beginning of a given run. The length of the simulation time step is directly proportional to the reaction probability for each event in the reaction network. This is typically only an issue if the probability of a given reaction is more than one (which means that some reaction events will be missed, since MCell is predicting that more than one event is happening in that time interval). In classic MCell the user can easily avoid this by manually ensuring the time step is short enough that all the probabilities for the reaction network are one or less.

However, this is not the case for MCell-R where the full reaction-network is not known ahead of time. Consider now some of the cross-linking reactions that take place in the TLBR system, where two large clusters can potentially bind together in tens or hundreds of different ways depending on the number of subunits in each cluster, which directly affects the reaction probability by the same proportion. In order to get accurate dynamics for the simulation it would be necessary to manually set a vanishingly small simulation time step or to implement an adaptive time step functionality for MCell.

Alternatively, as a more realistic solution it would be necessary to stop considering complexes in the model as point particles, which leads to the problematic assumption that all binding sites are accessible to the binding candidates at collision time. Instead, it would be better to consider the complexes as real geometries where only a portion of the binding interface is accessible depending

on the way two structures collide.

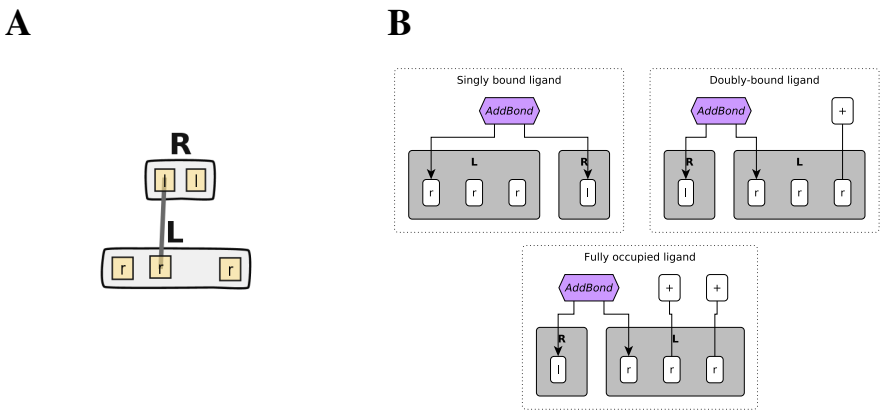


Figure 44: The TLBR model is a structurally simple system containing a ligand with three binding sub-units and a receptor with two binding subunits. The system contains three binding rules describing the binding process to form a singly, doubly and fully bound ligand. Despite this apparently simplicity the state space of the system is virtually infinite given that I can form infinite Ligand-Receptor chains, making the network generation of the system an impossible task if I wish to generate all reactions.

4.4 CONCLUSIONS AND FUTURE WORK

As the need to model systems on a larger scale becomes more ubiquitous [92, 93] so does the need for modeling frameworks that can keep up with the need to encode systems that present combinatorial complexity. In this chapter I presented a spatial framework that fills in that need, MCell-R: an extension to the MCell spatial modeling engine for the modeling of multi-state multi-component system through a rule-based modeling model definition framework and a network-free simulation approach powered by NFsim.

The MCell-R framework development consisted of three main sections: The development of *libNFsim*, a programmable interface for the integration of the NFsim simulator and data structures

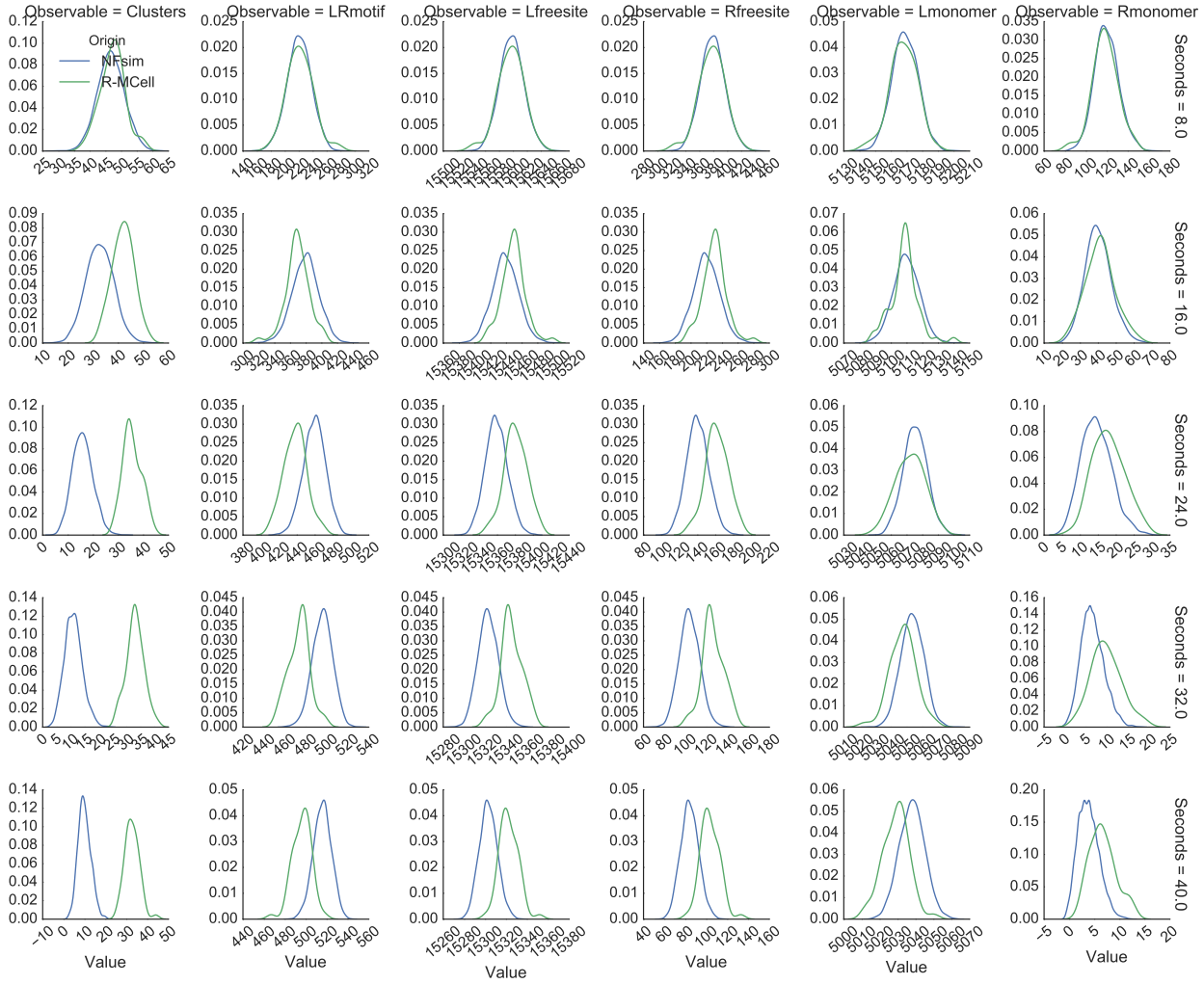


Figure 45: Matrix showing a visual comparison the distribution of different observables (Y-axis) evaluated at different timepoints (X-axis) for a spatial model of the trivalent ligand bivalent receptor system simulated using NFsim (in blue) and our own MCell-R simulation system (in green). Visual analysis reveals important discrepancies in the cluster production.

into other modeling platforms. The extension of the NFsim data model to handle arbitrary properties, particularly spatial properties, and the extension of the MCell data model and simulation algorithm to communicate with external routines for the scheduling of reaction events.

I tested and validated our framework with three systems that present combinatorial complexity: the FcεRI signaling pathway, the bivalent ligand bivalent receptor (BLBR) system and the trivalent ligand bivalent receptor system (TLBR). There are other biological systems that present combinatorial complexity that would benefit from our implementation like any model that presents multivalency and oligomerization like LAT aggregation [111], nephrin-Nck-N-Wasp signaling [52, 112], or CaMKII [52] with its large dodecameric structure.

The development of *libNFsim* also opens the door for the integration of the network-free framework with other platforms. For example, WESTPA [113] implements the weighted ensemble algorithm for the accurate and efficient sampling of rate events in models of varying complexity. In [114] I presented an exploratory integration of BioNetGen and WESTPA, however one of the main bottlenecks encountered was the lack of a clear programming interface between the two frameworks, which made it rely on an I/O based integration. With the development of libNFsim I will be addressing one of the main issues that implementation encountered, leading towards the full integration of rule-based modeling into the WESTPA plugin ecosystem.

There are some limitations to our current framework as evidenced by systems with unbounded graph complexity like TLBR. The MCell platform currently uses a minimum time step that is manually set at the beginning of the simulation. This approach is useful when MCell only had to deal with a predefined reaction-network since the user could use the time-step as a tunable optimization setting. However this falls apart when the full reaction-network and its associated rate parameters are generated on the fly and will result in the simulation giving inaccurate results. The solution then is to implement a system where MCell can set the time step of the simulation as needed by the reactions in the simulation that need to be fired.

Additionally, MCell (and by extension MCell-R) is currently a framework that considers a set of point particles in a diffusible media at microscopic timescales, and adds multi-state multi-component considerations to the particles and the associated reaction events. In future work I would like to add the concept of excluded-volume to the particles, that is that instead of every particle being a point in space it has a volume which is a function of its constituent structure. This

volume should have an effect on the spatial simulation by excluding the presence of other particles or walls from the volume it occupies.

Finally, there are currently no visualization capabilities available for MCell-R. This can be implemented by modifying the existing format used by MCell for visualization purposes by appending graph information. Visualization parameters like the size and color of the glyph used to represent a given pattern can then be calculated as a function of the graph properties, like the number of sub-units in the complex.

5.0 CONCLUSION

As the amount of biological data available to us has increased in recent years, so has the need to generate increasingly complex models that provide an adequate representation for this data. One way to address this problem is to leverage the existing modeling literature to construct descriptions that aggregate the dynamics of each individual modeling component to simulate a larger system. In order to perform this operation given a set of models, three questions must be answered: How do the elements in a model relate to each other? How do the elements between two models relate to each other? How do the elements in a model relate to real world objects? [26] Model informatics refers to the set of processes that allows us to answer these issues through operations like the analysis, comparison and visualization of modeling data.

However, this has remained an elusive goal given the limited amount of information encoded in the most popular mathematical modeling approaches. For example, reaction-networks models often obfuscate information about the biological processes that allow a set of reactants to transform into products [3, 21], like non-covalent bond formation or post-translation modifications. Developments in modeling paradigms in the last decade like rule-based modeling [36, 38] offer a way to address this limitation by providing a model representation that explicitly encodes information about how the elements in model relate to each other, which also facilitates the mapping of modeling data between different systems [44].

In Chapter 2, I presented *Atomizer*, an algorithm for extracting mechanistic information from reaction-network models. Atomizer employs a variety of analysis engines in order to recover an explicit representation of this information, which is then used to encode the model as a rule-based model. With this transformation comes the possibility of performing model informatics operations like model visualization through contact maps or visual comparison of models. I presented several use cases for Atomizer, both in its applications to individual models in the EGFR signaling cascade

and in its application to a large collection of reaction-network models in the BioModels database. Our results show that the information recovered by Atomizer and the visualizations generated from this recovered information are an asset for the understanding, analysis and visualization of models.

In Chapter 3, I expanded on other model informatics applications to the Atomized set of models I recovered from the previous chapter. First, I introduced a novel visualization technique for rule-based models: state transition diagrams. These diagrams are a molecule-centric view of the system that elucidates the set of requirements and temporal ordering associated with a given molecule's processes. In the second half I presented the Reaction Atomizer (Ratomizer), a model informatics framework that allows for the study, analysis, visualization, alignment and comparison of reaction-network models. This framework was implemented as a set of web-tools to facilitate their use and application to the translated library of models.

In recent years there has been an increasing number of efforts that seek to generate whole-cell models from the aggregation of simpler models like the whole-cell modeling project [93], and Big Mechanism [92]. These projects have recognized the limitations of current reaction-network approaches both when attempting to simulate systems where combinatorial complexity is a concern and when trying to combine reaction-network modeling data into larger model [94]. Rule-based modeling addresses both of this concerns and enables a number of model informatics operations can be applied. In this work I have presented a number of tools that facilitate the transition from current reaction-network modeling approaches and that provide a number of tools for the analysis of existing modeling data. I believe that the continued development of aggregated models together with the tools that allows us to study and understand these systems will provide a strong founndation for the development of the next-generation of models.

At the same time, the multi-state multi-component nature of a biochemical system can present a challenge not only in the modeling, but also in the simulation complexity of the system dynamics. In these systems, the presence of molecules with multiple domains which can be potentially bound or activated at the same time leads to a combinatorial explosion in the state space of a multi-unit complex and the number of potential reactions in a reaction-network. To address this problem, several 'network-free' simulators that use rule-based modeling principles have been developed that allow for the efficient simulation of systems where combinatorial complexity is a concern.

In Chapter 4 I introduced a novel modeling framework for the particle-based *spatial* encoding

and simulation of biochemical systems called MCell-R. The motivation for such a framework is the spatial simulation of systems like CaMKII and FcεRI where both the complexity of the biochemical circuitry and the spatial effects on the simulation must be addressed to obtain an accurate representation of a biochemical system. In order to address this issue I combined the MCell particle-based spatial algorithm together with the NFsim rule-based simulation framework. MCell-R was validated and tested through its application to several representative systems like FcεRI and the bivalent ligand bivalent receptor system.

Finally, I have presented the possibility of extending existing simulation platforms with rule-based principles. The result was the creation of *libNFsim*, a library implementation of the NFsim simulation framework. This API can be used as an engine that keeps track of the multi-state, multi-component nature of a given biological system. In Chapter 4 I combined this library with the MCell simulation platform, however in principle this library could be combined with other spatial or compartmental frameworks. I believe this to be the second half of the equation that will allow for the modeling and simulation of whole-cell systems using rule-based principles as the underlying design.

BIBLIOGRAPHY

[1] Epstein, J. M., 2008. Why model? *Journal of Artificial Societies and Social Simulation* 11:12.

[2] Gunawardena, J., 2014. Models in biology:accurate descriptions of our pathetic thinking. *BMC biology* 12:1.

[3] Tapia, J.-J., and J. R. Faeder, 2013. The Atomizer: Extracting Implicit Molecular Structure from Reaction Network Models. *Proceedings of the ACM Conference on Bioinformatics, Computational Biology and Biomedicine 2013. In press* .

[4] Chelliah, V., C. Laibe, and N. L. Novère, 2013. BioModels database: a repository of mathematical models of biological processes. *Encyclopedia of Systems Biology* 134–138.

[5] Lloyd, C. M., J. R. Lawson, P. J. Hunter, and P. F. Nielsen, 2008. The CellML model repository. *Bioinformatics* 24:2122–2123.

[6] Hoops, S., S. Sahle, R. Gauges, C. Lee, J. Pahle, N. Simus, M. Singhal, L. Xu, P. Mendes, and U. Kummer, 2006. COPASIa complex pathway simulator. *Bioinformatics* 22:3067–3074.

[7] Funahashi, A., Y. Matsuoka, A. Jouraku, M. Morohashi, N. Kikuchi, and H. Kitano, 2008. CellDesigner 3.5: a versatile modeling tool for biochemical networks. *Proceedings of the IEEE* 96:1254–1265.

[8] Medley, J. K., K. Choi, and H. M. Sauro, 2016. A portable library to support the SBML Layout Extension. *bioRxiv* 035725.

[9] Hucka, M., A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, A. P. Arkin, B. J. Bornstein, D. Bray, A. Cornish-Bowden, et al., 2003. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics* 19:524–531.

[10] Schoeberl, B., C. Eichler-Jonsson, E. D. Gilles, and G. Müller, 2002. Computational modeling of the dynamics of the MAP kinase cascade activated by surface and internalized EGF receptors. *Nature Biotechnol.* 20:370–375.

[11] Hodgkin, A. L., and A. F. Huxley, 1952. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology* 117:500.

[12] Izhikevich, E. M., et al., 2003. Simple model of spiking neurons. *IEEE Transactions on neural networks* 14:1569–1572.

[13] Tham, L.-S., L. Wang, R. A. Soo, S.-C. Lee, H.-S. Lee, W.-P. Yong, B.-C. Goh, and N. H. Holford, 2008. A pharmacodynamic model for the time course of tumor shrinkage by gemcitabine+ carboplatin in non–small cell lung cancer patients. *Clinical Cancer Research* 14:4213–4218.

[14] Munz, P., I. Hudea, J. Imad, and R. J. Smith, 2009. When zombies attack!: mathematical modelling of an outbreak of zombie infection. *Infectious Disease Modelling Research Progress* 4:133–150.

[15] Li, C., M. Donizelli, N. Rodriguez, H. Dharuri, L. Endler, V. Chelliah, L. Li, E. He, A. Henry, M. I. Stefan, J. L. Snoep, M. Hucka, N. Le Novère, and C. Laibe, 2010. BioModels Database: An enhanced, curated and annotated resource for published quantitative kinetic models. *BMC Systems Biology* 4:92.

[16] Consortium, U., et al., 2011. Reorganizing the protein space at the Universal Protein Resource (UniProt). *Nucleic acids research* gkr981.

[17] Joshi-Tope, G., M. Gillespie, I. Vastrik, P. D'Eustachio, E. Schmidt, B. de Bono, B. Jassal, G. Gopinath, G. Wu, L. Matthews, et al., 2005. Reactome: a knowledgebase of biological pathways. *Nucleic acids research* 33:D428–D432.

[18] Kanehisa, M., and S. Goto, 2000. KEGG: kyoto encyclopedia of genes and genomes. *Nucleic acids research* 28:27–30.

[19] Gennari, J. H., M. L. Neal, M. Galdzicki, and D. L. Cook, 2011. Multiple ontologies in action: Composite annotations for biosimulation models. *Journal of biomedical informatics* 44:146–154.

[20] Schulz, M., F. Krause, N. Le Novere, E. Klipp, and W. Liebermeister, 2011. Retrieval, alignment, and clustering of computational models based on semantic annotations. *Molecular systems biology* 7:512.

[21] Gennari, J. H., D. D. Neal, Maxwelland Cook, and Carlson, 2015. Semantic Annotation with SBML and CellML Models. *In* COMBINE.

[22] Le Novere, N., M. Hucka, H. Mi, S. Moodie, F. Schreiber, A. Sorokin, E. Demir, K. Wegner, M. I. Aladjem, S. M. Wimalaratne, et al., 2009. The systems biology graphical notation. *Nature biotechnology* 27:735–741.

[23] Sekar, J., 2015. Rule-based Modeling of Cell Signaling: Advances in Model Construction, Visualization and Simulation. http://d-scholarship.pitt.edu/26674/.

[24] Kohn, K. W., 2001. Molecular interaction maps as information organizers and simulation guides. *Chaos: An Interdisciplinary Journal of Nonlinear Science* 11:84–97.

[25] Czauderna, T., C. Klukas, and F. Schreiber, 2010. Editing, validating and translating of SBGN maps. *Bioinformatics* 26:2340–2341.

[26] Randhawa, R., C. A. Shaffer, and J. J. Tyson, 2010. Model composition for macromolecular regulatory networks. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on* 7:278–287.

[27] Neal, M. L., M. T. Cooling, L. P. Smith, C. T. Thompson, H. M. Sauro, B. E. Carlson, D. L. Cook, and J. H. Gennari, 2014. A reappraisal of how to build modular, reusable models of biological systems. *PLoS Comput Biol* 10:e1003849.

[28] Waltemath, D., R. Adams, F. T. Bergmann, M. Hucka, F. Kolpakov, A. K. Miller, I. I. Moraru, D. Nickerson, S. Sahle, J. L. Snoep, et al., 2011. Reproducible computational biology experiments with SED-ML-the simulation experiment description markup language. *BMC systems biology* 5:1.

[29] Gillespie, D. T., 1976. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of computational physics* 22:403–434.

[30] Gillespie, D. T., 1977. Exact stochastic simulation of coupled chemical reactions. *The journal of physical chemistry* 81:2340–2361.

[31] Smith, L. P., F. T. Bergmann, D. Chandran, and H. M. Sauro, 2009. Antimony: a modular model definition language. *Bioinformatics* 25:2452–2454.

[32] Moraru, I. I., J. C. Schaff, B. M. Slepchenko, M. Blinov, F. Morgan, A. Lakshminarayana, F. Gao, Y. Li, and L. M. Loew, 2008. Virtual Cell modelling and simulation software environment. *IET systems biology* 2:352–362.

[33] Hindmarsh, A. C., P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, and C. S. Woodward, 2005. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software (TOMS)* 31:363–396.

[34] Chylek, L. A., L. A. Harris, C.-S. Tung, J. R. Faeder, C. F. Lopez, and W. S. Hlavacek, 2014. Rule-based modeling: a computational approach for studying biomolecular site dynamics in cell signaling systems. *Wiley Interdisciplinary Reviews: Systems Biology and Medicine* 6:13–36.

[35] Hogg, J. S., L. A. Harris, L. J. Stover, N. S. Nair, and J. R. Faeder, 2014. Exact hybrid particle/population simulation of rule-based models of biochemical systems. *PLoS Comput Biol* 10:e1003544.

[36] Blinov, M. L., J. R. Faeder, B. Goldstein, and W. S. Hlavacek, 2004. BioNetGen: software for rule-based modeling of signal transduction based on the interactions of molecular domains. *Bioinformatics* 20:3289–3291.

[37] Faeder, J. R., M. L. Blinov, and W. S. Hlavacek, 2009. Rule-based modeling of biochemical systems with BioNetGen. *In* Systems Biology, Springer, 113–167.

[38] Danos, V., J. Feret, W. Fontana, R. Harmer, and J. Krivine, 2007. Rule-based modelling of cellular signalling. *In* CONCUR 2007–Concurrency Theory, Springer, 17–41.

[39] Meier-Schellersheim, M., and G. Mack, 1999. SIMMUNE, a tool for simulating and analyzing immune system behavior. *arXiv preprint cs/9903017* .

[40] Harris, L. A., J. S. Hogg, and J. R. Faeder, 2009. Compartmental rule-based modeling of biochemical systems. *In* Winter Simulation Conference. Winter Simulation Conference, 908–919.

[41] Sekar, J. A., J.-J. Tapia, and J. R. Faeder, 2015. Visualizing Regulation in Rule-based Models. *arXiv preprint arXiv:1509.00896* .

[42] Chylek, L. A., B. Hu, M. L. Blinov, T. Emonet, J. R. Faeder, B. Goldstein, R. N. Gutenkunst, J. M. Haugh, T. Lipniacki, R. G. Posner, et al., 2011. Guidelines for visualizing and annotating rule-based models. *Molecular bioSystems* 7:2779–2795.

[43] Danos, V., J. Feret, W. Fontana, R. Harmer, and J. Krivine, 2009. Rule-based modelling and model perturbation. *In* Transactions on Computational Systems Biology XI, Springer, 116–137.

[44] Wenskovitch, J. E., L. A. Harris, J. J. Tapia, J. R. Faeder, and G. E. Marai, 2014. MOSBIE: a tool for comparison and analysis of rule-based biochemical models. *BMC bioinformatics* 15:316.

[45] Kholodenko, B. N., O. V. Demin, G. Moehren, and J. B. Hoek, 1999. Quantification of short term signaling by the epidermal growth factor receptor. *Journal of Biological Chemistry* 274:30169–30181.

[46] Zhang, F., B. R. Angermann, and M. Meier-Schellersheim, 2013. The Simmune Modeler visual interface for creating signaling networks based on bi-molecular interactions. *Bioinformatics* btt134.

[47] Kolpakov, F. A., 2002. BioUML-Framework for visual modeling and simulation of biological systems. *In* Proceedings of the International Conference on Bioinformatics of Genome Regulation and Structure. 130–133.

[48] Blinov, M. L., D. Vasilescu, I. I. Moraru, L. M. Loew, and J. C. Schaff, 2016. Rule-Based Modeling and Simulation for Beginners: Intuitive Graphical Interface within Virtual Cell. *Biophysical Journal* 110:494a.

[49] Creamer, M. S., E. C. Stites, M. Aziz, J. A. Cahill, C. W. Tan, M. E. Berens, H. Han, K. J. Bussey, D. D. Von Hoff, W. S. Hlavacek, et al., 2012. Specification, annotation, visualization and simulation of a large rule-based model for ERBB receptor signaling. *BMC systems biology* 6:1.

[50] Misirli, G., M. Cavaliere, W. Waites, M. Pocock, C. Madsen, O. Gilfellon, R. Honorato-Zimmer, P. Zuliani, V. Danos, and A. Wipat, 2015. Annotation of rule-based models with formal semantics to enable creation, analysis, reuse and visualization. *Bioinformatics* btv660.

[51] Sneddon, M. W., J. R. Faeder, and T. Emonet, 2011. Efficient modeling, simulation and coarse-graining of biological complexity with NFsim. *Nature methods* 8:177–183.

[52] Michalski, P., and L. Loew, 2012. CaMKII activation and dynamics are independent of the holoenzyme structure: an infinite subunit holoenzyme approximation. *Physical biology* 9:036010.

[53] Colvin, J., M. I. Monine, R. N. Gutenkunst, W. S. Hlavacek, D. D. Von Hoff, and R. G. Posner, 2010. RuleMonkey: software for stochastic simulation of rule-based models. *BMC bioinformatics* 11:1.

[54] Danos, V., J. Feret, W. Fontana, and J. Krivine, 2007. Scalable simulation of cellular signaling networks. *In* Programming Languages and Systems, Springer, 139–157.

[55] Colvin, J., M. I. Monine, J. R. Faeder, W. S. Hlavacek, D. D. Von Hoff, and R. G. Posner, 2009. Simulation of large-scale rule-based models. *Bioinformatics* 25:910–917.

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

[57] Grünert, G., and P. Dittrich, 2010. Using the SRSim software for spatial and rule-based modeling of combinatorially complex biochemical reaction systems. *In* Membrane Computing, Springer, 240–256.

[58] Andrews, S. S., N. J. Addy, R. Brent, and A. P. Arkin, 2010. Detailed simulations of cell biology with Smoldyn 2.1. *PLoS Comput Biol* 6:e1000705.

[59] Stiles, J. R., T. M. Bartol, et al., 2001. Monte Carlo methods for simulating realistic synaptic microphysiology using MCell. *Computational neuroscience: realistic modeling for experimentalists* 87–127.

[60] Kerr, R. A., T. M. Bartol, B. Kaminsky, M. Dittrich, J.-C. J. Chang, S. B. Baden, T. J. Sejnowski, and J. R. Stiles, 2008. Fast Monte Carlo simulation methods for biological reaction-diffusion systems in solution and on surfaces. *SIAM Journal on Scientific Computing* 30:3126–3149.

[61] Meriney, S. D., and M. Dittrich, 2013. Organization and function of transmitter release sites at the neuromuscular junction. *The Journal of physiology* 591:3159–3165.

[62] Rappel, W.-J., and H. Levine, 2008. Receptor noise limitations on chemotactic sensing. *Proceedings of the National Academy of Sciences* 105:19270–19275.

[63] Kerr, R. A., H. Levine, T. J. Sejnowski, and W.-J. Rappel, 2006. Division accuracy in a stochastic model of Min oscillations in Escherichia coli. *Proceedings of the National Academy of Sciences of the United States of America* 103:347–352.

[64] Bartol, T. M., M. Dittrich, and J. R. Faeder, 2015. MCell. *Encyclopedia of Computational Neuroscience* 1673–1676.

[65] Smith, A. M., W. Xu, Y. Sun, J. R. Faeder, and G. E. Marai, 2012. RuleBender: integrated modeling, simulation and visualization for rule-based intracellular biochemistry. *BMC bioinformatics* 13:S3.

[66] Schulz, M., E. Klipp, and W. Liebermeister, 2012. Propagating semantic information in biochemical network models. *BMC bioinformatics* 13:18.

[67] Pardini, G., P. Milazzo, and A. Maggiolo-Schettini, 2014. Identification of components in biochemical pathways: extensive application to SBML models. *Natural Computing* 13:351–365.

[68] Pardini, G., P. Milazzo, and A. Maggiolo-Schettini, 2015. Component identification in biochemical pathways. *Theoretical Computer Science* .

[69] Maggiolo-Schettini, A., P. Milazzo, and G. Pardini, 2013. Application of a Semi-automatic Algorithm for Identification of Molecular Components in SBML Models. *arXiv preprint arXiv:1309.7689* .

[70] Cerami, E. G., B. E. Gross, E. Demir, I. Rodchenkov, Ö. Babur, N. Anwar, N. Schultz, G. D. Bader, and C. Sander, 2011. Pathway Commons, a web resource for biological pathway data. *Nucleic acids research* 39:D685–D690.

[71] Stark, C., B.-J. Breitkreutz, T. Reguly, L. Boucher, A. Breitkreutz, and M. Tyers, 2006. BioGRID: a general repository for interaction datasets. *Nucleic acids research* 34:D535–D539.

[72] Consortium, C., 2010. BioModels.net qualifiers. http://co.mbine.org/standards/qualifiers.

[73] Hucka, M., 2014. SBML Test Suite. http://sbml.org/Software/SBML_Test_Suite.

[74] Ghoniem, M., J.-D. Fekete, and P. Castagliola, 2005. On the readability of graphs using node-link and matrix-based representations: a controlled experiment and statistical analysis. *Information Visualization* 4:114–135.

[75] Sasagawa, S., Y.-i. Ozaki, K. Fujita, and S. Kuroda, 2005. Prediction and validation of the distinct dynamics of transient and sustained ERK activation. *Nature cell biology* 7:365–373.

[76] Singh, A., A. Jayaraman, and J. Hahn, 2006. Modeling regulatory mechanisms in IL-6 signal transduction in hepatocytes. *Biotechnology and bioengineering* 95:850–862.

[77] Ung, C. Y., H. Li, X. H. Ma, J. Jia, B. W. Li, B. C. Low, and Y. Z. Chen, 2008. Simulation of the regulation of EGFR endocytosis and EGFR-ERK signaling by endophilin-mediated RhoA-EGFR crosstalk. *FEBS letters* 582:2283–2290.

[78] Qi, Y.-f., Y.-x. Huang, H.-y. Wang, Y. Zhang, Y.-l. Bao, L.-g. Sun, Y. Wu, C.-l. Yu, Z.-b. Song, L.-h. Zheng, et al., 2013. Elucidating the crosstalk mechanism between IFN-gamma and IL-6 via mathematical modelling. *BMC bioinformatics* 14:41.

[79] Chang, C.-w., E. Poteet, J. A. Schetz, Z. H. Gümüş, and H. Weinstein, 2009. Towards a quantitative representation of the cell signaling mechanisms of hallucinogens: measurement and mathematical modeling of 5-HT1A and 5-HT2A receptor-mediated ERK1/2 activation. *Neuropharmacology* 56:213–225.

[80] Courtot, M., N. Juty, C. Knüpfer, D. Waltemath, A. Zhukova, A. Dräger, M. Dumontier, A. Finney, M. Golebiewski, J. Hastings, et al., 2011. Controlled vocabularies and semantics in systems biology. *Molecular systems biology* 7.

[81] Blinov, M. L., J. R. Faeder, B. Goldstein, and W. S. Hlavacek, 2006. A network model of early events in epidermal growth factor receptor signaling that accounts for combinatorial complexity. *Biosystems* 83:136–151.

[82] Yang, J., X. Meng, and W. S. Hlavacek, 2010. Rule-based modelling and simulation of biochemical systems with molecular finite automata. *IET systems biology* 4:453–466.

[83] Moodie, S., N. Le Novere, E. Demir, H. Mi, and F. Schreiber, 2011. Systems biology graphical notation: process description language level 1. *Nature Precedings* .

[84] Nickerson, D., 2015. The computational modeling in biology network. http://co.mbine.org/about.

[85] Huang, C.-Y., and J. E. Ferrell, 1996. Ultrasensitivity in the mitogen-activated protein kinase cascade. *Proceedings of the National Academy of Sciences* 93:10078–10083.

[86] Levchenko, A., J. Bruck, and P. W. Sternberg, 2000. Scaffold proteins may biphasically affect the levels of mitogen-activated protein kinase signaling and reduce its threshold properties. *Proceedings of the National Academy of Sciences* 97:5818–5823.

[87] Consortium, G. O., et al., 2004. The Gene Ontology (GO) database and informatics resource. *Nucleic acids research* 32:D258–D261.

[88] Juty, N., and N. le Novère, 2013. Systems Biology Ontology. *Encyclopedia of Systems Biology* 2063–2063.

[89] Novere, N. L., A. Finney, M. Hucka, U. S. Bhalla, F. Campagne, J. Collado-Vides, E. J. Crampin, M. Halstead, E. Klipp, P. Mendes, et al., 2005. Minimum information requested in the annotation of biochemical models (MIRIAM). *Nature Biotechnology* 23:1509–1515.

[90] Hu, B., G. M. Fricke, J. R. Faeder, R. G. Posner, and W. S. Hlavacek, 2009. GetBonNie for building, analyzing and sharing rule-based models. *Bioinformatics* 25:1457–1460.

[91] Google, 2015. Google App Engine. https://cloud.google.com/appengine/docs.

[92] Cohen, P. R., 2015. DARPA's Big Mechanism program. *Physical biology* 12:045008.

[93] Karr, J. R., J. C. Sanghvi, D. N. Macklin, M. V. Gutschow, J. M. Jacobs, B. Bolival, N. Assad-Garcia, J. I. Glass, and M. W. Covert, 2012. A whole-cell computational model predicts phenotype from genotype. *Cell* 150:389–401.

[94] Waltemath, D., J. Karr, F. Bergmann, V. Chelliah, M. Hucka, M. Krantz, W. Liebermeister, P. Mendes, C. Myers, P. Pir, et al., 2016. Toward community standards and software for whole-cell modeling .

[95] Lis, M., M. N. Artyomov, S. Devadas, and A. K. Chakraborty, 2009. Efficient stochastic simulation of reaction–diffusion processes via direct compilation. *Bioinformatics* 25:2289–2291.

[96] Elf, J., A. Doncic, and M. Ehrenberg, 2003. Mesoscopic reaction-diffusion in intracellular signaling. *In* SPIE's First International Symposium on Fluctuations and Noise. International Society for Optics and Photonics, 114–124.

[97] Sorokina, O., A. Sorokin, J. D. Armstrong, and V. Danos, 2013. A simulator for spatially extended kappa models. *Bioinformatics* 29:3105.

[98] Wilson-Kanamori, J. R., and J. Kanamori, 2015. Defining complex rule-based models in space and over time .

[99] Sterratt, D. C., O. Sorokina, and J. D. Armstrong, 2014. Integration of rule-based models and compartmental models of neurons. *In* Hybrid Systems Biology, Springer, 143–158.

[100] Carnevale, N. T., and M. L. Hines, 2006. The NEURON book. Cambridge University Press.

[101] Lok, L., and R. Brent, 2005. Automatic generation of cellular reaction networks with Moleculizer 1.0. *Nature biotechnology* 23:131–136.

[102] Michalski, P. J., and L. M. Loew, 2016. SpringSaLaD: A Spatial, Particle-Based Biochemical Simulation Platform with Excluded Volume. *Biophysical journal* 110:523–529.

[103] Gruenert, G., B. Ibrahim, T. Lenser, M. Lohel, T. Hinze, and P. Dittrich, 2010. Rule-based spatial modeling with diffusing, geometrically constrained molecules. *BMC bioinformatics* 11:307.

[104] Plimpton, S., 1995. Fast parallel algorithms for short-range molecular dynamics. *Journal of computational physics* 117:1–19.

[105] Stefan, M. I., T. M. Bartol, T. J. Sejnowski, and M. B. Kennedy, 2014. Multi-state modeling of biomolecules. *PLOS Comput Biol* 10:e1003844.

[106] Tolle, D. P., and N. Le Novère, 2010. Meredys, a multi-compartment reaction-diffusion simulator using multistate realistic molecular complexes. *BMC systems biology* 4:24.

[107] Saffman, P., and M. Delbrück, 1975. Brownian motion in biological membranes. *Proceedings of the National Academy of Sciences* 72:3111–3113.

[108] Sullivan, D. P., R. Arepally, R. F. Murphy, J.-J. Tapia, J. R. Faeder, M. Dittrich, and J. Czech, 2015. Design Automation for Biological Models: A Pipeline that Incorporates Spatial and Molecular Complexity. *In* Proceedings of the 25th edition on Great Lakes Symposium on VLSI. ACM, 321–323.

[109] Tapia, J. J., 2016. MCell-R validation test set. https://github.com/jjtapia/mcellRules.

[110] Goldstein, B., and A. S. Perelson, 1984. Equilibrium theory for the clustering of bivalent cell surface receptors by trivalent ligands. Application to histamine release from basophils. *Biophysical journal* 45:1109.

[111] Nag, A., M. I. Monine, J. R. Faeder, and B. Goldstein, 2009. Aggregation of membrane proteins by cytosolic cross-linkers: theory and simulation of the LAT-Grb2-SOS1 system. *Biophysical journal* 96:2604–2623.

[112] Falkenberg, C. V., M. L. Blinov, and L. M. Loew, 2013. Pleomorphic ensembles: Formation of large clusters composed of weakly interacting multivalent molecules. *Biophysical journal* 105:2451–2460.

[113] Zwier, M. C., J. L. Adelman, J. W. Kaus, A. J. Pratt, K. F. Wong, N. B. Rego, E. Suarez, S. Lettieri, D. W. Wang, M. Grabe, et al., 2015. WESTPA: An interoperable, highly scalable software package for weighted ensemble simulation and analysis. *Journal of Chemical Theory and Computation* 11:800–809.

[114] Donovan, R. M., J.-J. Tapia, D. P. Sullivan, J. R. Faeder, R. F. Murphy, M. Dittrich, and D. M. Zuckerman, 2016. Unbiased Rare Event Sampling in Spatial Stochastic Systems Biology Models Using a Weighted Ensemble of Trajectories. *PLoS Comput Biol* 12:e1004611.

APPENDIX A

ATOMIZATION DETAILS

In this appendix chapter I provide an pseudo-code description for the subroutines used during atomization.

These algorithms (Algorithms 6 and 7) are invoked as subroutines inside Algorithm 3 as part of the lexical analysis engine.

Algorithm 6 Analyze naming conventions: Get list of common modification strings in a model

Input: S, t {species in a model, edit distance threshold}

Output: D {Dictionary containing a model's common modifications and species associated with them}

1: $S_{pairs} \leftarrow SXS$ {all species pairs}
2: **for all** $pair \in S_{pairs}$ **do**
3: $distance, edit = levenshtein(pair)$ {get the numeric distance and the edit instructions}
4: **if** $isValid(edit)$ AND $distance < threshold$ **then**
5: $D[edit] \leftarrow pair$ {an edit is considered valid if it only adds characters}
6: **end if**
7: **end for**
8: **return** D

The weight calculation algorithm (Algorithm 8) is used before and after the SCT information propagation step (Algorithm 4)

Algorithm 7 Match reactant subunits to product subunits

Input: r, S, D {single reaction, list of species and modification dictionary}

Output: *matches* {dictionary of subunit pairs matching reactants to products, or an error if conservation of mass is violated}

1: **for all** $reactant \in r_{reactant}$ **do**
2: $subunits_{reactants} \leftarrow splitIntoSubunits(reactant, S)$ {returns a list of the smallest possible subunits found in S}
3: **end for**
4: **for all** $sub_r \in subunits_{reactants}$ **do**
5: **for all** $product \in r_{products}$ **do**
6: $subunits_{product} \leftarrow splitIntoSubunits(product, S)$
7: **for all** $sub_p \in subunits_{product}$ **do**
8: **if** $sub_r == sub_p$ **then**
9: $matches[product] \leftarrow \{ sub_r, sub_p \}$
10: remove $\{ sub_r, sub_p \}$ from *subunits* {match if I can pair them using naming conventions in D}
11: **break**
12: **else if** $fuzzyMatch(sub_r, sub_p, D)$ **then**
13: $matches[product] \leftarrow \{ sub_r, sub_p \}$
14: remove $\{ sub_r, sub_p \}$ from *subunits*
15: **break**
16: **end if**
17: **end for**
18: **end for**
19: **end for**
20: **if** $len(subunits_{reactants}) == 0$ AND $len(subunits_{products}) == 0$ **then**
21: **return** *matches*
22: **else**
23: **return** *conservationOfMassError*
24: **end if**

Algorithm 8 Obtain the weights for the entries in an SCT table

Input: SCT, $sctEntry \leftarrow$ an individual entry in the species composition table

```
1: if SCT[entry] == [] then
2:    return 1 {base case: this is a molecule type}
3: end if
4: for all sctEntry ∈ SCT do
5:    for all individualCandidate ∈ sctEntry do
6:       for all species ∈ individualCandidate do
7:          W += measureWeight(species,SCT)
8:       end for
9:    end for
10: end for
11: return W
```

Table 5: Atomization error

level	code	type	message
error	SCT201	sct	dependency cycle detected on 0
info	SCT001	sct	candidates from stoichiometry information are non self-consistent. Using lexical analysis
warning	SCT111	sct	Species 0 has conflicting definitions between one stoichiometric candidate (1) and naming conventions 2. Choosing the latter
warning	SCT112	sct	'Stoichiometry analysis result in non self-consistent definitions and conflicts with lexical analysis for species 0: stoichiometry(1)!= naming(2). Selecting lexical analysis
warning	SCT113	sct	For species 0: candidates 1 agree on the basic components but naming conventions cannot determine specific modifications. Selecting 2 based on longest match
warning	SCT114	sct	Stoichiometry conflict between reaction information 0 and lexical analysis 1 for molecule 2. Choosing reaction information
error	SCT211	sct	Cannot converge to solution for species 0: conflicting stoichiometry definitions 1 and no lexical information can be used
error	SCT212	sct	I don't know how this single candidate is modified.
error	SCT213	sct	I dont know how these are modified and I have no way to make an educated guess. Politely refusing to translate 0=1=2'
error	ATO202	atomization	We don't know how 0 and 1 bind together. Not atomizing

level	code	type	message
info	ATO001	atomization	Binding information found in BioGrid/Pathwaycommons for 0
warning	ATO102	atomization	According to BioGrid/Pathwaycommons there's more than one way to bind 0 and 1 together: 2. Defaulting to 3-4
error	ATO201	atomization	We don't know how 0 and 1 bind together and there's no relevant BioGrid/Pathway commons information. Not atomizing
info	CTX001	context	Redundant bonds detected between molecules 0 in species 1'
info	CTX002	context	Used reaction context information to determine that the bond 0 in species 1 is not likely
info	CTX003	context	Species with suspect information were found. Information dumped to 0_context.log
info	LAE001	lexical analysis	Lexical relationship inferred between 0: user information confirming it is required
info	LAE002	lexical analysis	adding forced transformation: 0:1:2'
info	LAE003	lexical analysis	common root common root 0=1:2
info	LAE004	lexical analysis	Discovered naming convention through reaction 0
info	LAE005	lexical analysis	added relationship through existing convention in reaction 0
info	LAE006	lexical analysis	Mapped an orphaned species to existing reactants using greedy matching
info	ANN001	annotation	0 was determined to be the same as 1 according to annotation information.'
info	ANN002	annotation	0 was determined to be partially match 1 according to annotation information.'
info	ANN003	annotation	0 is thought to be partially composed of reactants 1 according to annotation information. Please verify stoichiometry'
info	ANN004	annotation	Added equivalence from annotation information

level	code	type	message
error	ANN201	annotation	0 and 1 have a direct correspondence according to reaction information however their annotations are completely different. Not atomizing
error	ANN202	annotation	0 can be mapped to 1 through naming conventions but the annotation information does not match. Not atomizing
info	SIM001	simulation	In reaction 0: the left hand side has been determined to never activate and has been to rate 0
info	SIM002	simulation	n reaction 0: the right hand side has been determined
warning	SIM101	simulation	WARNING:Simulation:0 reported as function: but usage is ambiguous
warning	SIM102	simulation	In reaction 0: rates cannot be divided into left hand side and right hand side but reaction is marked as reversible
warning	SIM103	simulation	WARNING:Simulation:1-D compartments are not supported. Changing for 2-D compartments for 0. Please verify this does not affect simulation
warning	SIM104	simulation	Model contains no natural reactions: all reactions are produced by SBML rules
warning	SIM105	simulation	rate rules (0) are not properly supported in BioNetGen simulator
warning	SIM106	simulation	Parameter 0 corresponds both as a non zero parameter and a rate rule: verify behavior
error	SIM201	simulation	Variables that are both changed by an assignment rule and reactions are not supported in BioNetGen simulator. The variable will be split into two
error	SIM202	simulatiion	BNG cannot handle delay functions in function
error	SIM203	simulation	ERROR:Simulation':'The file contains no reactions
error	SIM204	simulation	Found usage of "inf" inside function 0"
info	SUM001	summary	File contains 0 molecules out of 1 original SBML species

level	code	type	message
debug	MSC001	misc	Unregistered modification
error	MSC201	misc	Elements 0 and 1 produce the same translation. Emptying 1
error	MSC202	misc	A connection could not be established to biogrid
error	MSC03	msic	A connection could not be established to uniprot

APPENDIX B

EGFR MODEL COMPARISON SET

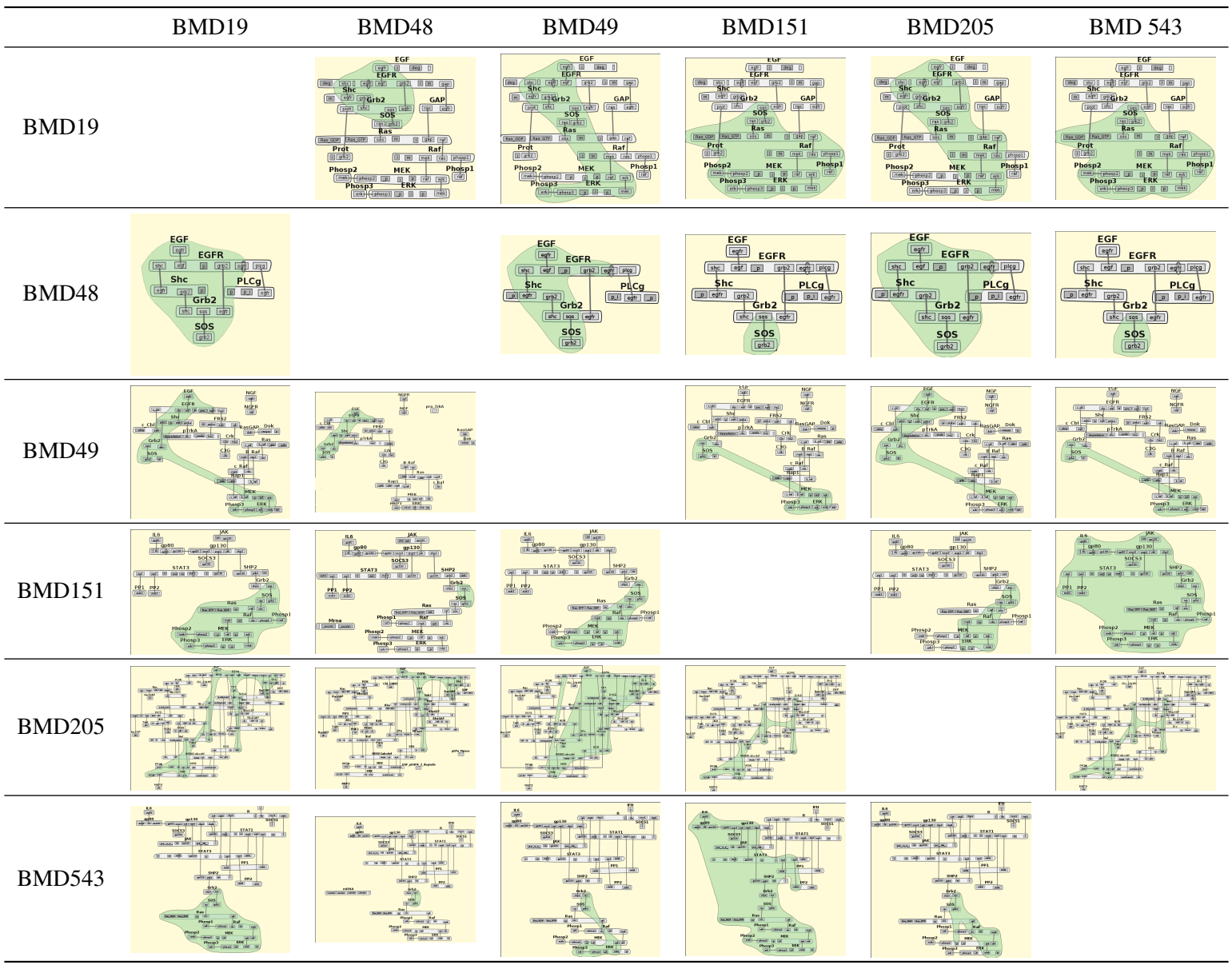


Figure 46: Structure comparison between EGFR-MAPK models. Each row shows the contact map for a particular model. The green highlighting illustrates the similarities between the base model and the comparison model referenced in the column marker.

APPENDIX C

ATOMIZATION USER CONFIGURATION FILES

BMD48

I used our Ratomizer tool (http://ratomizer.appspot.com/translate) to translate the files provided in this section. In Fig. 47 I present a screenshot of the tool while translating BMD48 ([45]). The configuration for BMD48 (shown below) includes the definition of a binding interface between EGFR and PLC $-\gamma$, along with the resolution of several issues related to inconsistent naming convention use in the model. For example, the binding of Sch_P together with Grb2 is encoded in the original model as Shc_Grb2. Atomizer requests the user to make sure that the translation is correct in this case.

```
"reactionDefinition" : [
],
"complexDefinition" : [
["EGFR", [["EGFR", "plcg", []]]],
["PLCg", [["PLCg", "egfr", []]]]
]
,
"modificationcationDefinition": {
   "Shc_Grb2":["Grb2",  "Shc_P"],
   "__EGF_EGFR__2_PLCg_P":["EGFR_p",  "EGFR",  "EGF",  "EGF",  "PLCg_P"],
   "__EGF_EGFR__2_Shc_Grb2":["EGFR_p",  "EGFR",  "EGF",  "EGF",  "Grb2",  "Shc_P"],
   "__EGF_EGFR__2_Shc_Grb2_SOS":["EGFR_p",  "EGFR",  "EGF",  "EGF",  "Grb2",  "SOS",
    ↪  "Shc_P"]
}
```

Finally, the full model is provided below. Atomizer provides annotation information along with unit conversion to molecule counts.

```
###
#@BQM_IS:['http://identifiers.org/biomodels.db/BIOMD0000000048',
 ↪  'http://identifiers.org/biomodels.db/MODEL6624193277']
#@BQB_OCCURS_IN:['http://identifiers.org/go/GO:0007173']
#@creatorName:'Snoep Jacky L'
#@creatorEmail:'jls@sun.ac.za'
```

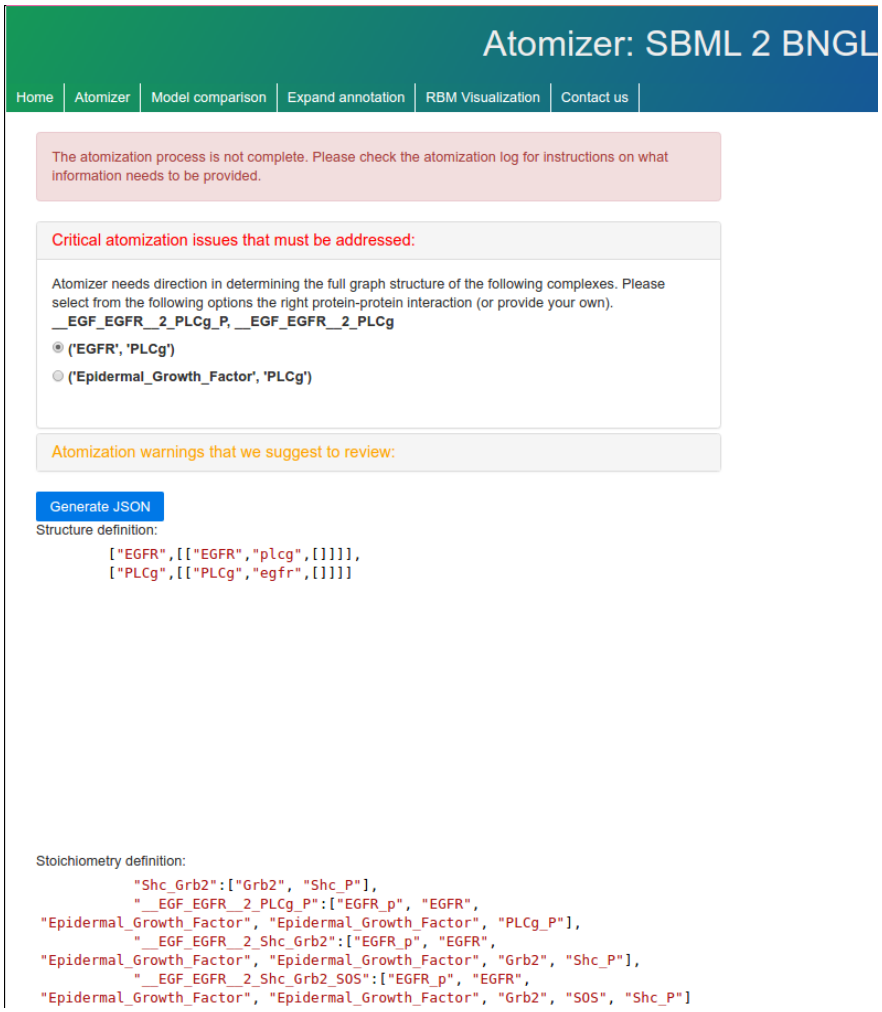


Figure 47: Screenshot of the Atomization web tool while refining the translation of BMD48. The model requests for help for deciding the way the PLC-γ molecule type binds to other complexes, along with other minor issues

```
#@notes:'This is an atomized translation of an SBML model created on 06/07/2016. The
 ↪  original model has 23 molecules and 25 reactions. The translated model has 6
 ↪  molecules and 25 rules'
#@BQM_IS_DESCRIBED_BY:['http://identifiers.org/pubmed/10514507']
#@BQB_HAS_TAXON:['http://identifiers.org/taxonomy/10116']
###
begin model
begin parameters
    r1_k1f 0.003000
    r1_k1b 0.060000
    r2_k2f 0.010000
    r2_k2b 0.100000
    r3_k3f 1.000000
    r3_k3b 0.010000
    r4_V4 450.000000
    r4_K4 50.000000
    r5_k5f 0.060000
    r5_k5b 0.200000
    r6_k6f 1.000000
    r6_k6b 0.050000
    r7_k7f 0.300000
    r7_k7b 0.006000
    r8_V8 1.000000
    r8_K8 100.000000
    r9_k9f 0.003000
    r9_k9b 0.050000
    r10_k10f 0.010000
    r10_k10b 0.060000
    r11_k11f 0.030000
    r11_k11b 0.004500
    r12_k12f 0.001500
    r12_k12b 0.000100
    r13_k13f 0.090000
    r13_k13b 0.600000
    r14_k14f 6.000000
    r14_k14b 0.060000
    r15_k15f 0.300000
    r15_k15b 0.000900
    r16_V16 1.700000
    r16_K16 340.000000
    r17_k17f 0.003000
    r17_k17b 0.100000
    r18_k18f 0.300000
    r18_k18b 0.000900
    r19_k19f 0.010000
    r19_k19b 0.021400
    r20_k20f 0.120000
    r20_k20b 0.000240
    r21_k21f 0.003000
    r21_k21b 0.100000
    r22_k22f 0.030000
    r22_k22b 0.064000
    r23_k23f 0.100000
    r23_k23b 0.021000
```

```
    r24_k24f 0.009000
    r24_k24b 0.042900
    r25_k25f 1.000000
    r25_k25b 0.030000
end parameters
begin compartments
    #volume units: L
    cell  3  3e-12 #cytoplasm
end compartments
begin molecule types
    #^ bqbiol:is uniprot:Q9Z1I1
    SOS(grb2)
    #^ bqbiol:is uniprot:P62994
    Grb2(egfr, shc, sos)
    #^ bqbiol:isVersionOf uniprot:Q9QX70
    EGFR(_p~_P~0, egfr, EGF, grb2, plcg, shc)
    #^ bqbiol:isVersionOf uniprot:Q5M824
    Shc(_p~_P~0, egfr, grb2)
    #^ bqbiol:is uniprot:P07522
    EGF(egfr)
    #^ bqbiol:isVersionOf interpro:IPR001192
    #^ bqbiol:hasVersion uniprot:P10686,  uniprot:P24135
    PLCg(_p~_P~0, egfr, p_i~P_I~0)
end molecule types
begin seed species
    @cell:EGF(egfr)@cell 1228488.0 # (680.0 * 1e-09)mol/L * 6.022e23/mol *3e-12L #EGF
     ↪  #EGF
    @cell:EGFR(_p~0, egfr, EGF, grb2, plcg, shc)@cell 180660.0 #original
     ↪  100.0nanomole/L  #EGFR #R
    @cell:PLCg(_p~0, egfr, p_i~0)@cell 189693.0 #original 105.0nanomole/L  #PLCg #PLCg
    @cell:Grb2(egfr, shc, sos)@cell 153561.0 #original 85.0nanomole/L  #Grb2 #Grb
    @cell:SOS(grb2)@cell 61424.4 #original 34.0nanomole/L  #SOS #SOS
    @cell:Shc(_p~0, egfr, grb2)@cell 270990.0 #original 150.0nanomole/L  #Shc #Shc
end seed species
begin observables
    Species EGF_compartment @cell:EGF(egfr)@cell #EGF
    Species EGFR_compartment @cell:EGFR(_p~0, egfr, EGF, grb2, plcg, shc)@cell #EGFR
    Species EGF_EGFR_compartment @cell:EGFR(_p~0, egfr, EGF!41, grb2, plcg,
     ↪  shc)@cell.EGF(egfr!41)@cell #EGF_EGFR
    Species __EGF_EGFR__2_compartment @cell:EGFR(_p~0, egfr!10, EGF!42, grb2, plcg,
     ↪  shc)@cell.EGFR(_p~0, egfr!10, EGF!41, grb2, plcg,
     ↪  shc)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell #(EGF_EGFR)2
    Species __EGF_EGFR__2_P_compartment @cell:EGFR(_p~0, egfr!10, EGF!42, grb2, plcg,
     ↪  shc)@cell.EGFR(_p~_P, egfr!10, EGF!41, grb2, plcg,
     ↪  shc)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell #(EGF_EGFR)2-P
    Species PLCg_compartment @cell:PLCg(_p~0, egfr, p_i~0)@cell #PLCg
    Species __EGF_EGFR__2_PLCg_compartment @cell:EGFR(_p~0, egfr!10, EGF!42, grb2,
     ↪  plcg, shc)@cell.EGFR(_p~_P, egfr!10, EGF!41, grb2, plcg!43,
     ↪  shc)@cell.PLCg(_p~0, egfr!43, p_i~0)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell
     ↪  #(EGF_EGFR)2_PLCg
```

```
    Species __EGF_EGFR__2_PLCg_P_compartment @cell:EGFR(_p~0, egfr!10, EGF!42, grb2,
      ↪ plcg, shc)@cell.EGFR(_p~_P, egfr!10, EGF!41, grb2, plcg!43,
      ↪ shc)@cell.PLCg(_p~_P, egfr!43, p_i~0)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell
      ↪ #(EGF_EGFR)2_PLCg-P
    Species PLCg_P_compartment @cell:PLCg(_p~_P, egfr, p_i~0)@cell #PLCg-P
    Species Grb2_compartment @cell:Grb2(egfr, shc, sos)@cell #Grb2
    Species __EGF_EGFR__2_Grb2_compartment @cell:EGFR(_p~0, egfr!10, EGF!42, grb2,
      ↪ plcg, shc)@cell.EGFR(_p~_P, egfr!10, EGF!41, grb2!13, plcg,
      ↪ shc)@cell.Grb2(egfr!13, shc, sos)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell
      ↪ #(EGF_EGFR)2_Grb2
    Species SOS_compartment @cell:SOS(grb2)@cell #SOS
    Species __EGF_EGFR__2_Grb2_SOS_compartment @cell:EGFR(_p~0, egfr!10, EGF!42, grb2,
      ↪ plcg, shc)@cell.EGFR(_p~_P, egfr!10, EGF!41, grb2!13, plcg,
      ↪ shc)@cell.Grb2(egfr!13, shc,
      ↪ sos!40)@cell.SOS(grb2!40)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell
      ↪ #(EGF_EGFR)2_Grb2_SOS
    Species Grb2_SOS_compartment @cell:Grb2(egfr, shc, sos!40)@cell.SOS(grb2!40)@cell
      ↪ #Grb2_SOS
    Species Shc_compartment @cell:Shc(_p~0, egfr, grb2)@cell #Shc
    Species __EGF_EGFR__2_Shc_compartment @cell:EGFR(_p~0, egfr!10, EGF!42, grb2, plcg,
      ↪ shc)@cell.EGFR(_p~_P, egfr!10, EGF!41, grb2, plcg, shc!12)@cell.Shc(_p~0,
      ↪ egfr!12, grb2)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell #(EGF_EGFR)2_Shc
    Species __EGF_EGFR___Shc_P_compartment @cell:EGFR(_p~0, egfr!10, EGF!42, grb2,
      ↪ plcg, shc)@cell.EGFR(_p~_P, egfr!10, EGF!41, grb2, plcg,
      ↪ shc!12)@cell.Shc(_p~_P, egfr!12,
      ↪ grb2)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell #(EGF_EGFR)_Shc-P
    Species Shc_P_compartment @cell:Shc(_p~_P, egfr, grb2)@cell #Shc-P
    Species __EGF_EGFR__2_Shc_Grb2_compartment @cell:EGFR(_p~0, egfr!10, EGF!42, grb2,
      ↪ plcg, shc)@cell.EGFR(_p~_P, egfr!10, EGF!41, grb2!13, plcg,
      ↪ shc!12)@cell.Shc(_p~_P, egfr!12, grb2!4)@cell.Grb2(egfr!13, shc!4,
      ↪ sos)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell #(EGF_EGFR)2_Shc_Grb2
    Species Shc_Grb2_compartment @cell:Shc(_p~_P, egfr, grb2!4)@cell.Grb2(egfr, shc!4,
      ↪ sos)@cell #Shc_Grb2
    Species __EGF_EGFR__2_Shc_Grb2_SOS_compartment @cell:EGFR(_p~0, egfr!10, EGF!42,
      ↪ grb2, plcg, shc)@cell.EGFR(_p~_P, egfr!10, EGF!41, grb2!13, plcg,
      ↪ shc!12)@cell.Shc(_p~_P, egfr!12, grb2!4)@cell.Grb2(egfr!13, shc!4,
      ↪ sos!40)@cell.SOS(grb2!40)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell
      ↪ #(EGF_EGFR)2_Shc_Grb2_SOS
    Species Shc_Grb2_SOS_compartment @cell:Shc(_p~_P, egfr, grb2!4)@cell.Grb2(egfr,
      ↪ shc!4, sos!40)@cell.SOS(grb2!40)@cell #Shc_Grb2_SOS
    Species PLCgP_I_compartment @cell:PLCg(_p~0, egfr, p_i~P_I)@cell #PLCgP-I
end observables
begin functions
    functionRate0() = (0.003) / 6.022e23
    functionRate1() = 0.01 * 2
    functionRate3() = 450.0 / (50.0 + __EGF_EGFR__2_P_compartment)
    functionRate4() = (0.06) / 6.022e23
    functionRate6() = 0.3
    functionRate7() = 1.0 / (100.0 + PLCg_P_compartment)
    functionRate8() = (0.003) / 6.022e23
```

```
    functionRate9() = (0.01) / 6.022e23
    functionRate10() = 0.03
    functionRate11() = 0.0015
    functionRate12() = (0.09) / 6.022e23
    functionRate14() = 0.3
    functionRate15() = 1.7 / (340.0 + Shc_P_compartment)
    functionRate16() = (0.003) / 6.022e23
    functionRate17() = 0.3
    functionRate18() = (0.01) / 6.022e23
    functionRate19() = 0.12
    functionRate20() = (0.003) / 6.022e23
    functionRate21() = (0.03) / 6.022e23
    functionRate22() = 0.1
    functionRate23() = (0.009) / 6.022e23
end functions
begin reaction rules
    v1: EGFR(_p~0, egfr, EGF, grb2, plcg, shc)@cell + EGF(egfr)@cell <-> EGFR(_p~0,
     ↪ egfr, EGF!41, grb2, plcg, shc)@cell.EGF(egfr!41)@cell functionRate0(), r1_k1b
    v2: EGFR(_p~0, egfr, EGF!41, grb2, plcg, shc)@cell.EGF(egfr!41)@cell + EGFR(_p~0,
     ↪ egfr, EGF!41, grb2, plcg, shc)@cell.EGF(egfr!41)@cell <-> EGFR(_p~0, egfr!10,
     ↪ EGF!42, grb2, plcg, shc)@cell.EGFR(_p~0, egfr!10, EGF!41, grb2, plcg,
     ↪ shc)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell functionRate1(), r2_k2b
    v3: EGFR(_p~0, egfr!10, EGF!42, grb2, plcg, shc)@cell.EGFR(_p~0, egfr!10, EGF!41,
     ↪ grb2, plcg, shc)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell <-> EGFR(_p~0,
     ↪ egfr!10, EGF!42, grb2, plcg, shc)@cell.EGFR(_p~_P, egfr!10, EGF!41, grb2,
     ↪ plcg, shc)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell (r3_k3f)/2, r3_k3b
    v4: EGFR(_p~0, egfr!10, EGF!42, grb2, plcg, shc)@cell.EGFR(_p~_P, egfr!10, EGF!41,
     ↪ grb2, plcg, shc)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell -> EGFR(_p~0,
     ↪ egfr!10, EGF!42, grb2, plcg, shc)@cell.EGFR(_p~0, egfr!10, EGF!41, grb2, plcg,
     ↪ shc)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell functionRate3()
    v5: EGFR(_p~0, egfr!10, EGF!42, grb2, plcg, shc)@cell.EGFR(_p~_P, egfr!10, EGF!41,
     ↪ grb2, plcg, shc)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell + PLCg(_p~0, egfr,
     ↪ p_i~0)@cell <-> EGFR(_p~0, egfr!10, EGF!42, grb2, plcg, shc)@cell.EGFR(_p~_P,
     ↪ egfr!10, EGF!41, grb2, plcg!43, shc)@cell.PLCg(_p~0, egfr!43,
     ↪ p_i~0)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell functionRate4(), r5_k5b
    v6: EGFR(_p~0, egfr!10, EGF!42, grb2, plcg, shc)@cell.EGFR(_p~_P, egfr!10, EGF!41,
     ↪ grb2, plcg!43, shc)@cell.PLCg(_p~0, egfr!43,
     ↪ p_i~0)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell <-> EGFR(_p~0, egfr!10,
     ↪ EGF!42, grb2, plcg, shc)@cell.EGFR(_p~_P, egfr!10, EGF!41, grb2, plcg!43,
     ↪ shc)@cell.PLCg(_p~_P, egfr!43, p_i~0)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell
     ↪ r6_k6f, r6_k6b
    v7: EGFR(_p~0, egfr!10, EGF!42, grb2, plcg, shc)@cell.EGFR(_p~_P, egfr!10, EGF!41,
     ↪ grb2, plcg!43, shc)@cell.PLCg(_p~_P, egfr!43,
     ↪ p_i~0)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell <-> PLCg(_p~_P, egfr,
     ↪ p_i~0)@cell + EGFR(_p~0, egfr!10, EGF!42, grb2, plcg, shc)@cell.EGFR(_p~_P,
     ↪ egfr!10, EGF!41, grb2, plcg, shc)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell
     ↪ functionRate6(), (r7_k7b) / 6.022e23
    v8: PLCg(_p~_P, egfr, p_i~0)@cell -> PLCg(_p~0, egfr, p_i~0)@cell functionRate7()
```

```
v9: Grb2(egfr, shc, sos)@cell + EGFR(_p~0, egfr!10, EGF!42, grb2, plcg,
    shc)@cell.EGFR(_p~_P, egfr!10, EGF!41, grb2, plcg,
  ↪ shc)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell <-> EGFR(_p~0, egfr!10, EGF!42,
  ↪ grb2, plcg, shc)@cell.EGFR(_p~_P, egfr!10, EGF!41, grb2!13, plcg,
  ↪ shc)@cell.Grb2(egfr!13, shc, sos)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell
  ↪ functionRate8(), r9_k9b
v10: EGFR(_p~0, egfr!10, EGF!42, grb2, plcg, shc)@cell.EGFR(_p~_P, egfr!10,
    EGF!41, grb2!13, plcg, shc)@cell.Grb2(egfr!13, shc,
  ↪ sos)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell + SOS(grb2)@cell <-> EGFR(_p~0,
  ↪ egfr!10, EGF!42, grb2, plcg, shc)@cell.EGFR(_p~_P, egfr!10, EGF!41, grb2!13,
  ↪ plcg, shc)@cell.Grb2(egfr!13, shc,
  ↪ sos!40)@cell.SOS(grb2!40)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell
  ↪ functionRate9(), r10_k10b
v11: EGFR(_p~0, egfr!10, EGF!42, grb2, plcg, shc)@cell.EGFR(_p~_P, egfr!10,
    EGF!41, grb2!13, plcg, shc)@cell.Grb2(egfr!13, shc,
  ↪ sos!40)@cell.SOS(grb2!40)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell <->
  ↪ Grb2(egfr, shc, sos!40)@cell.SOS(grb2!40)@cell + EGFR(_p~0, egfr!10, EGF!42,
  ↪ grb2, plcg, shc)@cell.EGFR(_p~_P, egfr!10, EGF!41, grb2, plcg,
  ↪ shc)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell functionRate10(), (r11_k11b) /
  ↪ 6.022e23
v12: Grb2(egfr, shc, sos!40)@cell.SOS(grb2!40)@cell <-> Grb2(egfr, shc, sos)@cell +
  ↪ SOS(grb2)@cell functionRate11(), (r12_k12b) / 6.022e23
v13: Shc(_p~0, egfr, grb2)@cell + EGFR(_p~0, egfr!10, EGF!42, grb2, plcg,
    shc)@cell.EGFR(_p~_P, egfr!10, EGF!41, grb2, plcg,
  ↪ shc)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell <-> EGFR(_p~0, egfr!10, EGF!42,
  ↪ grb2, plcg, shc)@cell.EGFR(_p~_P, egfr!10, EGF!41, grb2, plcg,
  ↪ shc!12)@cell.Shc(_p~0, egfr!12, grb2)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell
  ↪ functionRate12(), r13_k13b
v14: EGFR(_p~0, egfr!10, EGF!42, grb2, plcg, shc)@cell.EGFR(_p~_P, egfr!10,
    EGF!41, grb2, plcg, shc!12)@cell.Shc(_p~0, egfr!12,
  ↪ grb2)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell <-> EGFR(_p~0, egfr!10, EGF!42,
  ↪ grb2, plcg, shc)@cell.EGFR(_p~_P, egfr!10, EGF!41, grb2, plcg,
  ↪ shc!12)@cell.Shc(_p~_P, egfr!12,
  ↪ grb2)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell r14_k14f, r14_k14b
v15: EGFR(_p~0, egfr!10, EGF!42, grb2, plcg, shc)@cell.EGFR(_p~_P, egfr!10,
    EGF!41, grb2, plcg, shc!12)@cell.Shc(_p~_P, egfr!12,
  ↪ grb2)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell <-> EGFR(_p~0, egfr!10, EGF!42,
  ↪ grb2, plcg, shc)@cell.EGFR(_p~_P, egfr!10, EGF!41, grb2, plcg,
  ↪ shc)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell + Shc(_p~_P, egfr, grb2)@cell
  ↪ functionRate14(), (r15_k15b) / 6.022e23
v16: Shc(_p~_P, egfr, grb2)@cell -> Shc(_p~0, egfr, grb2)@cell functionRate15()
v17: EGFR(_p~0, egfr!10, EGF!42, grb2, plcg, shc)@cell.EGFR(_p~_P, egfr!10,
    EGF!41, grb2, plcg, shc!12)@cell.Shc(_p~_P, egfr!12,
  ↪ grb2)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell + Grb2(egfr, shc, sos)@cell <->
  ↪ EGFR(_p~0, egfr!10, EGF!42, grb2, plcg, shc)@cell.EGFR(_p~_P, egfr!10, EGF!41,
  ↪ grb2!13, plcg, shc!12)@cell.Shc(_p~_P, egfr!12, grb2!4)@cell.Grb2(egfr!13,
  ↪ shc!4, sos)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell functionRate16(),
  ↪ r17_k17b
```

```
    v18: EGFR(_p~0, egfr!10, EGF!42, grb2, plcg, shc)@cell.EGFR(_p~_P, egfr!10,
         EGF!41, grb2!13, plcg, shc!12)@cell.Shc(_p~_P, egfr!12,
      ↪  grb2!4)@cell.Grb2(egfr!13, shc!4,
      ↪  sos)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell <-> Shc(_p~_P, egfr,
      ↪  grb2!4)@cell.Grb2(egfr, shc!4, sos)@cell + EGFR(_p~0, egfr!10, EGF!42, grb2,
      ↪  plcg, shc)@cell.EGFR(_p~_P, egfr!10, EGF!41, grb2, plcg,
      ↪  shc)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell functionRate17(), (r18_k18b) /
      ↪  6.022e23
    v19: SOS(grb2)@cell + EGFR(_p~0, egfr!10, EGF!42, grb2, plcg, shc)@cell.EGFR(_p~_P,
         egfr!10, EGF!41, grb2!13, plcg, shc!12)@cell.Shc(_p~_P, egfr!12,
      ↪  grb2!4)@cell.Grb2(egfr!13, shc!4,
      ↪  sos)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell <-> EGFR(_p~0, egfr!10, EGF!42,
      ↪  grb2, plcg, shc)@cell.EGFR(_p~_P, egfr!10, EGF!41, grb2!13, plcg,
      ↪  shc!12)@cell.Shc(_p~_P, egfr!12, grb2!4)@cell.Grb2(egfr!13, shc!4,
      ↪  sos!40)@cell.SOS(grb2!40)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell
      ↪  functionRate18(), r19_k19b
    v20: EGFR(_p~0, egfr!10, EGF!42, grb2, plcg, shc)@cell.EGFR(_p~_P, egfr!10,
         EGF!41, grb2!13, plcg, shc!12)@cell.Shc(_p~_P, egfr!12,
      ↪  grb2!4)@cell.Grb2(egfr!13, shc!4,
      ↪  sos!40)@cell.SOS(grb2!40)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell <->
      ↪  Shc(_p~_P, egfr, grb2!4)@cell.Grb2(egfr, shc!4, sos!40)@cell.SOS(grb2!40)@cell
      ↪  + EGFR(_p~0, egfr!10, EGF!42, grb2, plcg, shc)@cell.EGFR(_p~_P, egfr!10,
      ↪  EGF!41, grb2, plcg, shc)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell
      ↪  functionRate19(), (r20_k20b) / 6.022e23
    v21: Grb2(egfr, shc, sos)@cell + Shc(_p~_P, egfr, grb2)@cell <-> Shc(_p~_P, egfr,
      ↪  grb2!4)@cell.Grb2(egfr, shc!4, sos)@cell functionRate20(), r21_k21b
    v22: Shc(_p~_P, egfr, grb2!4)@cell.Grb2(egfr, shc!4, sos)@cell + SOS(grb2)@cell <->
      ↪  Shc(_p~_P, egfr, grb2!4)@cell.Grb2(egfr, shc!4, sos!40)@cell.SOS(grb2!40)@cell
      ↪  functionRate21(), r22_k22b
    v23: Shc(_p~_P, egfr, grb2!4)@cell.Grb2(egfr, shc!4,
      ↪  sos!40)@cell.SOS(grb2!40)@cell <-> Grb2(egfr, shc,
      ↪  sos!40)@cell.SOS(grb2!40)@cell + Shc(_p~_P, egfr, grb2)@cell functionRate22(),
      ↪  (r23_k23b) / 6.022e23
    v24: EGFR(_p~0, egfr!10, EGF!42, grb2, plcg, shc)@cell.EGFR(_p~_P, egfr!10,
         EGF!41, grb2, plcg, shc!12)@cell.Shc(_p~_P, egfr!12,
      ↪  grb2)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell + Grb2(egfr, shc,
      ↪  sos!40)@cell.SOS(grb2!40)@cell <-> EGFR(_p~0, egfr!10, EGF!42, grb2, plcg,
      ↪  shc)@cell.EGFR(_p~_P, egfr!10, EGF!41, grb2!13, plcg, shc!12)@cell.Shc(_p~_P,
      ↪  egfr!12, grb2!4)@cell.Grb2(egfr!13, shc!4,
      ↪  sos!40)@cell.SOS(grb2!40)@cell.EGF(egfr!41)@cell.EGF(egfr!42)@cell
      ↪  functionRate23(), r24_k24b
    v25: PLCg(_p~_P, egfr, p_i~0)@cell <-> PLCg(_p~0, egfr, p_i~P_I)@cell r25_k25f,
      ↪  r25_k25b
  end reaction rules
  end model
```

BMD19 Configuration for BMD19 includes the definition of a binding interface between `EGFR` and `Prot`. For namespace normalization purposes `Ras_GDP` and `Ras_GTP` were resolved as a

molecule type Ras that have a subunits be in states GDP, GTP. This information is not included in the model but can be easily provided externally.

```
"reactionDefinition" : [

],
"complexDefinition" : [
["Prot",[["Prot", "egfr",[]]]],
["EGFR",[["EGFR","prot",[]]]]
],
"modificationDefinition": {
    "Ras":[],
    "Ras_GTP":["Ras"],
    "Ras_GDP":["Ras"]
}
```

MODEL0975191032

In Fig. 48 I present the intermediate contact maps produced for the translation of Chang's model of ERK activation described in Section 3.4.1. The user configuration files used to provide each configuration (generated using the Reaction Atomizer web tool) are also provided below.

First atomization refinement. The configuration focuses on resolving the structure of G-protein related species and some other small complexes

---

```
"reactionDefinition" : [

],
"complexDefinition" : [


],
"modificationDefinition": {
    "Gq_a_GDP":["Gq_a"],
    "Gq_a_GTP":["Gq_a"],
    "Gi_a_GDP":["Gi_a"],
    "Gi_a_GTP":["Gi_a"],
    "AAactive":["AA"],
    "DAGactive":["DAG"],
    "cpxA_HT1AR_internal":["serotonin","HT1AR_internal"],
    "Sos_P":["Sos"],
    "Grb2_sos_P":["Grb2","Sos_P"],
    "Gitrimer":["Gi_a_GDP", "Gi_betagamma"],
    "Gq_trimer":["Gq_a_GDP", "Gq_betagamma"],
    "Shc_star":["Shc"],
    "PKC_Ca_DAGactive":["PKC","Ca","DAGactive"],
    "PKC_DAG_AAactive":["PKC","DAG","AAactive"],
    "HT1AR_internal":["HT1AR"],
    "cpxERKP_MKP_PPstar":["cpxERKP_MKP_PP"]
}
```

Second atomization refinement. This iteration adds structure regarding the binding of HT-Receptors to G protein and the binding of SOS to Ras.

```
"reactionDefinition" : [

],
"complexDefinition" : [
    ["Sos",[["Sos","rasgdp",[]]]],
    ["RasGDP",[["RasGDP","sos",[]]]],

    ["Gq_a",[["Gq_a","2ar",[]]]],
    ["2AR",[["2AR","gq_a",[]]]],
    ["Gi_a",[["Gi_a","ht1ar",[]]]],
    ["HT1AR",[["HT1AR","gi_a",[]]]]

],
"modificationDefinition": {
    "Gq_a_GDP":["Gq_a"],
    "Gq_a_GTP":["Gq_a"],
    "Gi_a_GDP":["Gi_a"],
    "Gi_a_GTP":["Gi_a"],

    "AAactive":["AA"],
    "DAGactive":["DAG"],
    "cpxA_HT1AR_internal":["serotonin","HT1AR_internal"],
    "Sos_P":["Sos"],
    "Grb2_sos_P":["Grb2","Sos_P"],
    "Gitrimer":["Gi_a_GDP", "Gi_betagamma"],
    "Gq_trimer":["Gq_a_GDP", "Gq_betagamma"],
    "Shc_star":["Shc"],
    "PKC_Ca_DAGactive":["PKC","Ca","DAGactive"],
    "PKC_DAG_AAactive":["PKC","DAG","AAactive"],
    "HT1AR_internal":["HT1AR"],
    "cpxERKP_MKP_PPstar":["cpxERKP_MKP_PP"]

}
```

Final iteration. This version refines the species names used in the model.

```
"reactionDefinition" : [

],
"complexDefinition" : [
    ["Sos",[["Sos","ras",[]]]],
    ["Ras",[["Ras","sos",[]]]],

    ["Gq_a",[["Gq_a","ht2ar",[]]]],
    ["Gi_a",[["Gi_a","ht1ar",[]]]],
```

```
    ["HT1AR",[["HT1AR","gi_a",[]]]],
    ["ht2AR",[["ht2AR","gq_a",[]]]]

],
"modificationDefinition": {
    "Gq_a_GDP":["Gq_a"],
    "Gq_a_GTP":["Gq_a"],
    "Gi_a_GDP":["Gi_a"],
    "Gi_a_GTP":["Gi_a"],
    "cpxERKP_MEKPP":["ERK_P", "MEK_PP"],
    "cpxMEKP_PP2A":["MEK_P", "PP2A"],
    "cpxMEKPP_PP2A":["MEK_PP", "PP2A"],
    "cpxMEKP_Rafstar":["MEK_P", "Rafstar"],

    "ht2AR":[],
    "PIP2":[],
    "PIP3":["PIP2"],
    "DAG":[],
    "PA":[],
    "PI":[],
    "Pi":[],
    "APC":[],
    "RasGDP":["Ras"],
    "RasGTP":["Ras"],
    "PKCactive":["PKC"],
    "cpxA_HT1AR_internal":["serotonin","HT1AR_internal"],
    "Sos_P":["Sos"],
    "Grb2_sos_P":["Grb2","Sos_P"],
    "Gitrimer":["Gi_a_GDP", "Gi_betagamma"],
    "Gq_trimer":["Gq_a_GDP", "Gq_betagamma"],
    "Shc_star":["Shc"],
    "PKC_Ca_DAGactive":["PKCactive","Ca","DAG"],
    "PKC_DAG_AAactive":["PKCactive","DAG","AA"],
    "HT1AR_internal":["HT1AR"],
    "cpxERKP_MKP_PPstar":["cpxERKP_MKP_PP"],
    "cpxKetanserin_2AR":["ht2AR", "Ketanserin"],
    "cpxKetanserin_2ARGq":["Gq_a", "Gq_betagamma", "Ketanserin", "ht2AR"]
}
```

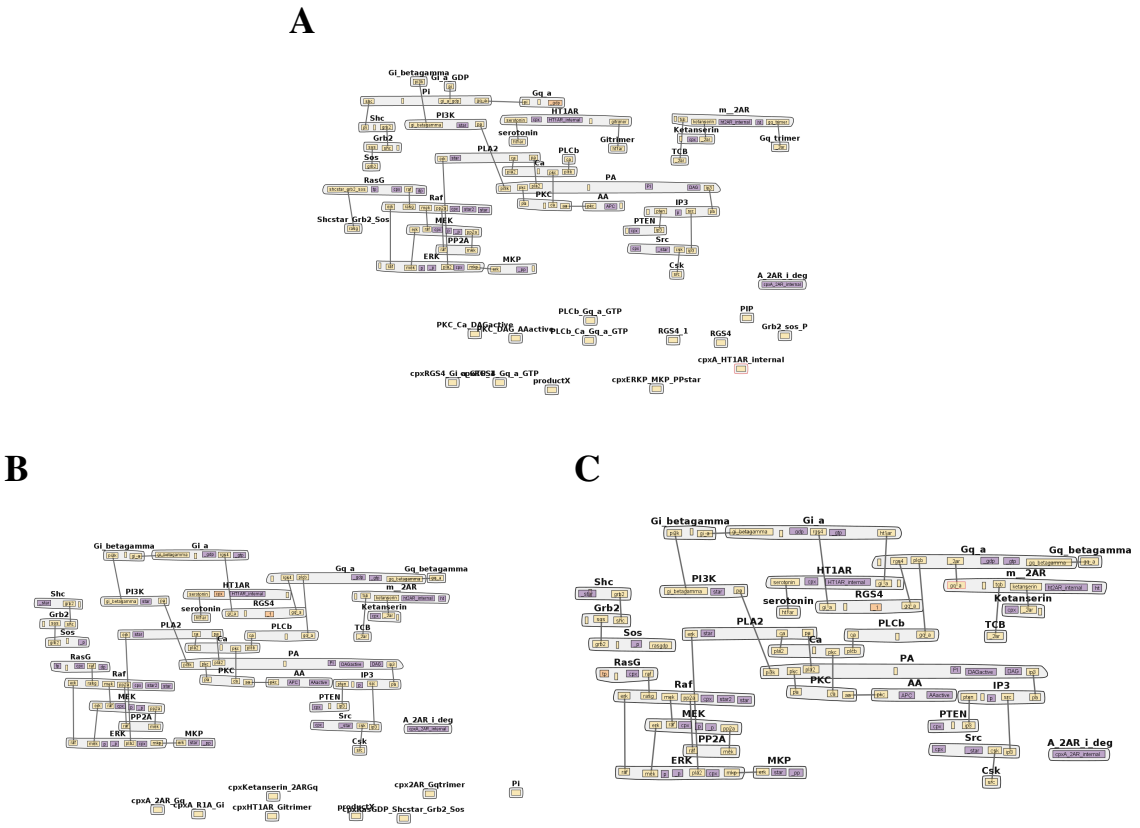


Figure 48: Atomization of the Chang model at different stages. A: No user configuration. B: Basic conflict information regarding the stoichiometry of different complexes C: Adds information regarding the structure of some missing complexes

APPENDIX D

MDLR GRAMMAR

```
#misc
number = Word(nums)
integer = Combine(Optional(plusorminus) + number)
real = Combine(integer +
               Optional(point + Optional(number)) +
               Optional(e + integer))
numarg = (real | integer)
identifier = Word(alphas, alphanums + "_")
dotidentifier = Word(alphas, alphanums + "_" + ".")
bracketidentifier = identifier + lbracket + Word(alphas) + rbracket
statement = Group(identifier + equal + (quotedString | restOfLine))

#section enclosure grammar
bracket_enclosure = nestedExpr( '{', '}')
section_enclosure_ = Forward()
nestedBrackets = nestedExpr('[', ']', content=section_enclosure_)
nestedCurlies = nestedExpr('{', '}', content=section_enclosure_)
section_enclosure_ << (statement | Group(identifier + ZeroOrMore(identifier) +
 ↪ nestedCurlies) |  Group(identifier + '@' + restOfLine) | Word(alphas, alphanums +
 ↪ "_[]") | identifier | Suppress(',') | '@' | real)

# bng pattern definition
name = Word(alphanums + '_')
species = Suppress('()') + Optional(Suppress('@' + Word(alphanums + '_-'))) +
 ↪ ZeroOrMore(Suppress('+') + Word(alphanums + "_" + ":#-")
                                   + Suppress("()") + Optional(Suppress('@' +
                                    ↪ Word(alphanums + '_-'))))

component_definition = Group(identifier.setResultsName('componentName') +
 ↪ Optional(Group(tilde +
 ↪ delimitedList(identifier|integer,delim='~')).setResultsName('state')))
component_statement = Group(identifier.setResultsName('componentName') +
 ↪ Optional(Group(Suppress(bang) + (numarg | '+' | '?')).setResultsName('bond')) + \
                     Optional(Group(Suppress(tilde)+ (identifier |
                      ↪ integer)).setResultsName('state')))

```

```
29  molecule_definition = Group(identifier.setResultsName('moleculeName') +
30                              Optional((lparen + delimitedList(component_definition,
                                 ↪  delim=',').setResultsName('components') + rparen)) +
31                              Optional(Group('@' + Word(alphanums +
                                 ↪  '_-')).setResultsName('moleculeCompartment')))
32  molecule_instance = Group(identifier.setResultsName('moleculeName') +
33                            Optional((lparen + delimitedList(component_statement,
                               ↪  delim=',').setResultsName('components') + rparen)) +
34                            Optional(Group('@' + Word(alphanums +
                               ↪  '_-')).setResultsName('moleculeCompartment')))
35  species_definition = Group(Optional(Group('@' + Word(alphanums +
     ↪  '_')).setResultsName('speciesCompartment') + Suppress('::')) +
36                             delimitedList(molecule_instance,
                                ↪  delim='.').setResultsName('speciesPattern'))
37  reaction_definition = Group(Group(delimitedList(species_definition,
     ↪  delim='+')).setResultsName('reactants') + (uni_arrow | bi_arrow) +
38                              Group(delimitedList(species_definition,
                                 ↪  delim='+')).setResultsName('products') +
39                              Group(lbracket + (numarg | identifier) + Optional(comma +
                                 ↪  (numarg| identifier)) +
                                 ↪  rbracket).setResultsName('rate'))
40
41  # generic hash section grammar
42  hashed_section = (hashsymbol + Group(OneOrMore(name) + bracket_enclosure))
43
44  # hash molecule_entry
45  molecule_entry = Group(molecule_definition + Optional(Group(lbrace +
     ↪  ZeroOrMore(statement) + rbrace)))
46  hashed_molecule_section = Group(hashsymbol + Suppress(define_molecules_) + lbrace +
     ↪  OneOrMore(molecule_entry) + rbrace)
47
48  # hash reaction entry
49  hashed_reaction_section = Group(hashsymbol + Suppress(define_reactions_) + lbrace +
     ↪  OneOrMore(reaction_definition) + rbrace)
50
51  # hash observable entry
52  count_definition = Group(count_ + lbracket +
     ↪  species_definition.setResultsName('speciesPattern') + Suppress(',') + identifier +
     ↪  rbracket)
53  observable_entry = Group(lbrace + Group(delimitedList(count_definition,
     ↪  delim='+')).setResultsName('patterns') + rbrace + Suppress('=>') +
     ↪  quotedString.setResultsName('outputfile'))
54  hashed_observable_section = Group(hashsymbol + Suppress(reaction_data_output_) + lbrace
     ↪  + OneOrMore(observable_entry | statement) + rbrace)
55
56  # hash initialization entry
57  key = identifier + Suppress('=')
58  value = restOfLine
59  release_site_definition = Group(identifier.setResultsName('name') + release_site_ +
     ↪  lbrace + dictOf(key,value).setResultsName('entries') + rbrace)
60  object_definition = Group(identifier.setResultsName('compartmentName') +
     ↪  Suppress(object_) + (bracketidentifier | identifier) + (nestedExpr('{',
     ↪  '}',content=statement)).setResultsName('compartmentOptions'))
```

```
hashed_initialization_section = Group(hashsymbol + Suppress(instantiate_) + identifier.setResultsName('name') +
                                identifier.setResultsName('type') + lbrace + Group(ZeroOrMore(release_site_definition | object_definition)).setResultsName('entries') + rbrace )

other_sections = section_enclosure_
#statement = Group(identifier + equal + (quotedString | OneOrMore(mathElements)))  + Suppress(LineEnd() | StringEnd())
grammar = ZeroOrMore(Suppress(other_sections) | Suppress(statement) | hashed_molecule_section.setResultsName('molecules') | hashed_reaction_section.setResultsName('reactions') |
                     hashed_observable_section.setResultsName('observables') | hashed_initialization_section.setResultsName('initialization') | Suppress(hashed_section))

nonhashedgrammar = ZeroOrMore(Suppress(statement) | Suppress(hashed_section) | Dict(other_sections))


statementGrammar = ZeroOrMore(statement | Suppress(other_sections) | Suppress(hashed_section))
```

APPENDIX E

FCεRI MODEL DEFINITION

Category	Parameter	Value
Initial populations (molecules)	Ligands	6000
	Receptors	400
	Syk	400
	Lyn	280
Reaction Rate	Ligand receptor binding	$1e5/(Ms)$
	Receptor aggregation	$1.660e-04/(um^2Ns)$
	Constitutive Lyn-Rec binding	$1.66/(um^2Ns)$
	Constitutive Lyn-Rec undinding	20/s
	Lyn-Rec binding through SH2	$1.66/(um^2Ns)$
	Lyn-Rec SH2 unbinding	$0.12/s$
	Syk-Rec binding through tSH2	$1e8/(Ms)$
	Syk-Rec tSH2 unbinding	$0.13/s$
	Beta transphosphorylation by constitutive Lyn	$30/s$
	Beta transphosphorylation by SH2 Lyn	$100/s$
	Gamma transphosphorylation by constitutive Lyn	$1/s$
	Gamma transphosphorylation by SH2 bound Lyn	$3/s$
	Syk transphosphorylation by constitutive Lyn	$30/s$
	Syk transphosphorylation by SH2 bound Lyn	$100/s$
	Syk Transphosphorylation by Syk not phosphorylated on a loop	$100/s$
	Syk transphosphorylation by Syk phosphorylated on a loop	$200/s$
	Membrane Syk dephosphorylation	$0.1/s$
	Cytosol Syk dephosphorylation	$0.1/s$
Spatial parameters	EC volume	27 μm^3
	CP volume	9 μm^3
	CP surface area	21 μm^2

```
ITERATIONS = 3000
TIME_STEP = 5e-06
VACANCY_SEARCH_DISTANCE = 100

INCLUDE_FILE = "icogeometry.mdl"

DEFINE_SURFACE_CLASSES
{
   reflect {
   REFLECTIVE = ALL_MOLECULES
   }
```

```
}

MODIFY_SURFACE_REGIONS
{
   EC[wall]
   {
      SURFACE_CLASS = reflect
   }
   EC[obj_wall]
   {
      SURFACE_CLASS = reflect
   }
   CP[PM]
   {
      SURFACE_CLASS = reflect
   }
   CP[obj_wall]
   {
      SURFACE_CLASS = reflect
   }
}


/* Model Parameters */
   Nav = 6.022e8               /* Avogadro number based on a volume size of 1 cubic um
     ↪  */
   rxn_layer_t = 0.01
   vol_wall = 0.88/rxn_layer_t  /*Surface area*/
   vol_EC = 39
   vol_PM = 0.01/rxn_layer_t  /*Surface area*/
   vol_CP = 1
   Lig_tot = 6.0e3
   Rec_tot = 4.0e2
   Lyn_tot = 2.8e2
   Syk_tot = 4e2
   kp1 = 0.000166057788110262*Nav
   km1 = 0.00
   kp2 = 1.66057788110262e-06/rxn_layer_t
   km2 = 0.00
   kpL = 0.0166057788110262/rxn_layer_t
   kmL = 20
   kpLs = 0.0166057788110262/rxn_layer_t
   kmLs = 0.12
   kpS = 0.0166057788110262*Nav
   kmS = 0.13
   pLb = 30
   pLbs = 100
   pLg = 1
   pLgs = 3
   pLS = 30
   pLSs = 100
   pSS = 100
   pSSs = 200
   dm = 0.1
```

```
   dc = 0.1

/* Diffusion bloc */
   T = 298.15      /* Temperature, K */
   h = rxn_layer_t      /* Thickness of 2D compartment, um */
   Rs = 0.002564      /* Radius of a (spherical) molecule in 3D compartment, um */
   Rc = 0.0015      /* Radius of a (cylindrical) molecule in 2D compartment, um */
   gamma = 0.5722      /* Euler's constant */
   KB = 1.3806488e-19      /* Boltzmann constant, cm^2.kg/K.s^2 */
   mu_EC = 1e-9      /* Viscosity in compartment EC, kg/um.s */
   mu_PM = 1e-9      /* Viscosity in compartment PM, kg/um.s */
   mu_CP = 1e-9      /* Viscosity in compartment CP, kg/um.s */



#DEFINE_MOLECULES
{
  Lig(l,l)
  {
      DIFFUSION_CONSTANT_3D = KB*T/(6*PI*mu_EC*Rs)
  }
  Lyn(U,SH2)
  {
      DIFFUSION_CONSTANT_2D =
        ↪  KB*T*LOG((mu_PM*h/(Rc*(mu_EC+mu_CP)/2))-gamma)/(4*PI*mu_PM*h)
  }
  Syk(tSH2,l~Y~pY,a~Y~pY)
  {
      DIFFUSION_CONSTANT_3D = KB*T/(6*PI*mu_CP*Rs)
  }
  Rec(a,b~Y~pY,g~Y~pY)
  {
      DIFFUSION_CONSTANT_2D =
        ↪  KB*T*LOG((mu_PM*h/(Rc*(mu_EC+mu_CP)/2))-gamma)/(4*PI*mu_PM*h)
  }

}



#DEFINE_REACTIONS
{
    /* Ligand-receptor binding      */
     Rec(a) + Lig(l,l) <-> Rec(a!1).Lig(l!1,l)  [kp1, km1]

    /* Receptor-aggregation*/
     Rec(a) + Lig(l,l!+) <-> Rec(a!2).Lig(l!2,l!+)  [kp2, km2]

    /* Constitutive Lyn-receptor binding  */
     Rec(b~Y) + Lyn(U,SH2) <-> Rec(b~Y!1).Lyn(U!1,SH2)  [kpL, kmL]

    /* Transphosphorylation of beta by constitutive Lyn  */
```

```
 Lig(l!1,l!2).Lyn(U!3,SH2).Rec(a!2,b~Y!3).Rec(a!1,b~Y) ->
   ↪  Lig(l!1,l!2).Lyn(U!3,SH2).Rec(a!2,b~Y!3).Rec(a!1,b~pY)  [pLb]

/* Transphosphorylation of gamma by constitutive Lyn  */
 Lig(l!1,l!2).Lyn(U!3,SH2).Rec(a!2,b~Y!3).Rec(a!1,g~Y) ->
   ↪  Lig(l!1,l!2).Lyn(U!3,SH2).Rec(a!2,b~Y!3).Rec(a!1,g~pY)  [pLg]

/* Lyn-receptor binding through SH2 domain  */
 Rec(b~pY) + Lyn(U,SH2) <-> Rec(b~pY!1).Lyn(U,SH2!1)  [kpLs, kmLs]

/* Transphosphorylation of beta by SH2-bound Lyn  */
 Lig(l!1,l!2).Lyn(U,SH2!3).Rec(a!2,b~pY!3).Rec(a!1,b~Y) ->
   ↪  Lig(l!1,l!2).Lyn(U,SH2!3).Rec(a!2,b~pY!3).Rec(a!1,b~pY)  [pLbs]

/* Transphosphorylation of gamma by SH2-bound Lyn  */
 Lig(l!1,l!2).Lyn(U,SH2!3).Rec(a!2,b~pY!3).Rec(a!1,g~Y) ->
   ↪  Lig(l!1,l!2).Lyn(U,SH2!3).Rec(a!2,b~pY!3).Rec(a!1,g~pY)  [pLgs]

/* Syk-receptor binding through tSH2 domain  */
 Rec(g~pY) + Syk(tSH2) <-> Rec(g~pY!1).Syk(tSH2!1)  [kpS, kmS]

/* Transphosphorylation of Syk by constitutive Lyn  */
 Lig(l!1,l!2).Lyn(U!3,SH2).Rec(a!2,b~Y!3).Rec(a!1,g~pY!4).Syk(tSH2!4,l~Y) ->
   ↪  Lig(l!1,l!2).Lyn(U!3,SH2).Rec(a!2,b~Y!3).Rec(a!1,g~pY!4).Syk(tSH2!4,l~pY)
   ↪  [pLS]

/* Transphosphorylation of Syk by SH2-bound Lyn  */
 Lig(l!1,l!2).Lyn(U,SH2!3).Rec(a!2,b~pY!3).Rec(a!1,g~pY!4).Syk(tSH2!4,l~Y) ->
   ↪  Lig(l!1,l!2).Lyn(U,SH2!3).Rec(a!2,b~pY!3).Rec(a!1,g~pY!4).Syk(tSH2!4,l~pY)
   ↪  [pLSs]

/* Transphosphorylation of Syk by Syk not phosphorylated on aloop  */
 Lig(l!1,l!2).Syk(tSH2!3,a~Y).Rec(a!2,g~pY!3).Rec(a!1,g~pY!4).Syk(tSH2!4,a~Y) ->
   ↪  Lig(l!1,l!2).Syk(tSH2!3,a~Y).Rec(a!2,g~pY!3).Rec(a!1,g~pY!4).Syk(tSH2!4,a~pY)
   ↪  [pSS]

/* Transphosphorylation of Syk by Syk phosphorylated on aloop  */
 Lig(l!1,l!2).Syk(tSH2!3,a~pY).Rec(a!2,g~pY!3).Rec(a!1,g~pY!4).Syk(tSH2!4,a~Y) ->
   ↪  Lig(l!1,l!2).Syk(tSH2!3,a~pY).Rec(a!2,g~pY!3).Rec(a!1,g~pY!4).Syk(tSH2!4,a~pY)
   ↪  [pSSs]

/* Dephosphorylation of Rec beta  */
 Rec(b~pY) -> Rec(b~Y)  [dm]

/* Dephosphorylation of Rec gamma  */
 Rec(g~pY) -> Rec(g~Y)  [dm]

/* Dephosphorylation of Syk at membrane  */
 Syk(tSH2!+,l~pY) -> Syk(tSH2!+,l~Y)  [dm]
 Syk(tSH2!+,a~pY) -> Syk(tSH2!+,a~Y)  [dm]

/* Dephosphorylation of Syk in cytosol  */
 Syk(tSH2,l~pY) -> Syk(tSH2,l~Y)  [dc]
 Syk(tSH2,a~pY) -> Syk(tSH2,a~Y)  [dc]
```

```
}


#INSTANTIATE Scene OBJECT
{
  EC OBJECT EC {}
  CP OBJECT CP {
    PARENT = EC
    MEMBRANE = PM OBJECT CP[ALL]
  }

   ligand_rel RELEASE_SITE
   {
    SHAPE = Scene.EC[ALL] - Scene.CP[ALL]
    MOLECULE = @EC::Lig(l,l)
    NUMBER_TO_RELEASE = Lig_tot
    RELEASE_PROBABILITY = 1
   }
   lyn_rel RELEASE_SITE
   {
    SHAPE = Scene.CP[ALL]
    MOLECULE = @PM::Lyn(U,SH2)
    NUMBER_TO_RELEASE = Lyn_tot
    RELEASE_PROBABILITY = 1
   }
   syk_rel RELEASE_SITE
   {
    SHAPE = Scene.CP[ALL]
    MOLECULE = @CP::Syk(tSH2,l~Y,a~Y)
    NUMBER_TO_RELEASE = Syk_tot
    RELEASE_PROBABILITY = 1
   }
   receptor_rel RELEASE_SITE
   {
    SHAPE = Scene.CP[ALL]
    MOLECULE = @PM::Rec(a,b~Y,g~Y)
    NUMBER_TO_RELEASE = Rec_tot
    RELEASE_PROBABILITY = 1
   }

}

/* Observables bloc */
#REACTION_DATA_OUTPUT
{
   STEP = 5e-4

    /*LynFree*/
    {COUNT[Lyn(U,SH2), WORLD] } => "./react_data/LynFree.dat"

    {COUNT[Rec(b~pY!?), WORLD] } => "./react_data/RecPbeta.dat"
    {COUNT[Rec(a!1).Lig(l!1,l), WORLD] + COUNT[Rec(a), WORLD]} =>
     ↪  "./react_data/RecMon.dat"
    {COUNT[Rec(a!1).Lig(l!1,l!2).Rec(a!2), WORLD]} => "./react_data/RecDim.dat"
```

```
{COUNT[Lig(l!1,l!2).Lyn(U!3,SH2).Rec(a!2,b!3).Rec(a!1,b), WORLD]} =>
 ↪  "./react_data/RecRecLigLyn.dat"

{COUNT[Rec(g~pY),WORLD] + COUNT[Rec(g~pY!+), WORLD] } =>
 ↪  "./react_data/RecPgamma.dat"
{COUNT[Rec(g~pY!1).Syk(tSH2!1), WORLD] } => "./react_data/RecSyk.dat"
{COUNT[Rec(g~pY!1).Syk(tSH2!1,a~pY), WORLD] } => "./react_data/RecSykPS.dat"
```

APPENDIX F

TLBR MODEL DEFINITION

Category	Parameter	Value
Initial populations	Ligands	4200
	Receptors	300
	Free binding equilibrium constant	$1.0084e8/(M)$
	Cross linking equilibrium constant	$3.372e10/(M)$
	unbinding rate	$0.01/s$
	Free binding rate constant	$1.0084e6/(Ms)$
	Cross linking rate constant	$3.372e8/(Ms)$
Spatial parameters	EC volume	$27\mu m^3$
	CP volume	$9\mu m^3$
	CP surface area	$21\ \mu m^2$

```
ITERATIONS = 100000
TIME_STEP = 5e-05
VACANCY_SEARCH_DISTANCE = 100

INCLUDE_FILE = "icogeometry.mdl"

DEFINE_SURFACE_CLASSES
{
   reflect {
    REFLECTIVE = ALL_MOLECULES
   }
}

MODIFY_SURFACE_REGIONS
{

   CP[ALL]
   {
      SURFACE_CLASS = reflect
   }
}
```

```
/* Model Parameters */
   Nav = 6.022e8               /* Avogadro number based on a volume size of 1 cubic um
    ↪  */
   rxn_layer_t = 0.01
   vol_wall = 56.5695045056029  /*Surface area*/
   vol_EC = 39
    Lig_tot = 4200
    Rec_tot = 300
    cTot  =0.84
    beta  =50
    koff  =0.01

    kp1 = 1.0084e6
    kp2 = 3.372e8



#DEFINE_MOLECULES
{
  L(r,r,r)
  {
      DIFFUSION_CONSTANT_3D = 1e-8

  }
  R(l,l){
      DIFFUSION_CONSTANT_3D = 1e-9

  }
}

#DEFINE_REACTIONS
{
    R(l) + L(r,r,r)     <-> R(l!1).L(r!1,r,r)      [kp1, koff]
    R(l) + L(r,r,r!+)   <-> R(l!1).L(r!1,r,r!+)    [kp2, koff]
    R(l) + L(r,r!+,r!+) <-> R(l!1).L(r!1,r!+,r!+)  [kp2, koff]
}


#INSTANTIATE Scene OBJECT
{
  EC OBJECT EC {}
  CP OBJECT CP {
    PARENT = EC
    MEMBRANE = PM OBJECT CP[ALL]
  }

   ligand_rel RELEASE_SITE
   {
    SHAPE = Scene.CP[ALL]
    MOLECULE = L(r,r,r)@CP
    NUMBER_TO_RELEASE = Lig_tot
```

```
    RELEASE_PROBABILITY = 1
   }
      receptor_rel RELEASE_SITE
   {
    SHAPE = Scene.CP[ALL]
  MOLECULE = R(l,l)@CP
    NUMBER_TO_RELEASE = Rec_tot
    RELEASE_PROBABILITY = 1

   }
}

/* Observables bloc */
#REACTION_DATA_OUTPUT
{
   STEP = 1e-6
    Species    Clusters  R(l!0).L(r!0,r!1).R(l!1)  // Any species with crosslinked
     ↪ receptors
    Molecules  LRmotif   L(r!0).R(l!0)
    Molecules  Lfreesite L(r)
    Molecules  Rfreesite R(l)
    Species    Lmonomer  L(r,r,r)
    Species    Rmonomer  R(l,l)
    Molecules  Ltot      L()
    Molecules  Rtot      R()
}
```