RULE-BASED MODELING OF CELL SIGNALING: ADVANCES IN MODEL CONSTRUCTION, VISUALIZATION AND SIMULATION

by

John Arul Prakash Sekar

B.Tech. Industrial Biotechnology, Anna University, 2007

Submitted to the Graduate Faculty of

School of Medicine in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2015

UNIVERSITY OF PITTSBURGH

SCHOOL OF MEDICINE

This dissertation was presented

by

John Arul Prakash Sekar

It was defended on

December 4th, 2015

and approved by

James R. Faeder, Computational & Systems Biology

Daniel M. Zuckerman, Computational & Systems Biology

Chakra S. Chennubhotla, Computational & Systems Biology

Alexander D. Sorkin, Cell Biology

Russell S. Schwartz, Biological Sciences, Carnegie-Mellon University

Thesis Director: Daniel M. Zuckerman, Computational & Systems Biology

Dissertation Advisor: James R. Faeder, Computational & Systems Biology

Rule-based Modeling of Cell Signaling: Advances in Model Construction, Visualization and Simulation

John Arul Prakash Sekar, PhD

University of Pittsburgh, 2015

Rule-based modeling is a graph-based approach to specifying the kinetics of cell signaling systems. A reaction rule is a compact and explicit graph-based representation of a kinetic process, and it matches a class of reactions that involve identical sites and identical kinetics. Compact rule-based models have been used to generate large and combinatorially complex reaction networks, and rules have also been used to compile databases of kinetic interactions targeting specific cells and pathways. In this work, I address three technological challenges associated with rule-based modeling. First, I address the ability to generate an automated global visualization of a rule-based model as a network of signal flows. I showed how to analyze a reaction rule and extract a set of bipartite regulatory relationships, which can be aggregated across rules into a global network. I also provide a set of coarse-graining approaches to compress an automatically generated network into a compact pathway diagram, even for models with 100s of rules. Second, I resolved an incompatibility between two recent advances in rule-based modeling: network-free simulation (which enables simulation without generating a reaction network), and energy-based rule-based modeling (which enables specifying a model using cooperativity parameters and automated accounting of free energy). The incompatibility arose because calculating the reaction rate requires computing the reaction free energy in an energy-based model, and this requires knowledge of both reactants and products of the reaction, but the products are not available in a network-free simulation until after the reaction event has fired. This was resolved by expanding each energy-based rule into a number of normal reaction rules for which reaction free energies can be calculated unambiguously. Third, I demonstrated a particular type of modularization that is based on treating a set of rules as a module. This enables building models from combinations of modular hypotheses and supplements the other modularization strategies such as macros, types and energy-based compression.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. James R. Faeder for his continued support and encouragement throughout the development of this thesis. I have learnt a lot from him that I will carry forward as a part of my own journey in science, and he has been a source of inspiration for me, both as an academic and as a person. I am very glad to be able to build on his pioneering research, and I wish to carry the same optimism he has when asking the hard questions.

I would like to thank the other members of my committee, Dr. Daniel M. Zuckerman, Dr. Chakra S. Chennubhotla, Dr. Russell S. Schwartz and Dr. Alexander D. Sorkin. They have pushed me to do better and have encouraged me to take a wider perspective beyond the narrow confines of a problem to be solved, and this thesis has benefited from that.

I would like to thank Leonard A. Harris, Justin S. Hogg, and Jose-Juan Tapia, equal parts friends, mentors and colleagues. I could not have asked for a more compatible set of people to interact with and learn from. I shall have many pleasant memories of interacting with them and with the other members of the Faeder lab.

Over the years, I was blessed with a number of friends who were smart, motivated and intellectual. Their eagerness to engage me in lively conversation has been a great source of joy, and their continued support has been indispensable for my personal development.

Most importantly, I would like to thank my parents Jacintha and Antony Sekar, my sister Sheeba and my brother-in-law Jeremiah Jasher. They are the rock on which I stand, and their love and unquestioning support makes everything possible.

1.0 RULE-BASED MODELING – A REVIEW

1.1 MECHANISTIC MODELS OF CELL SIGNALING

One of the hallmarks of a living cell is that its chemical composition responds dynamically to internal and external chemical stimuli. Characterizing the composition and dynamic behavior of a chemical system is a frequently encountered problem in the study and perturbation of cell responses, whether it is the movement of a bacterium towards nutrients or the response of a cancer cell to a secreted growth factor. A mechanistic model seeks to build the system from its molecular parts, encode the chemical properties at the level of individual molecules, and then predict *in silico* the behavior of the system under different conditions. However, biochemical systems are very complex and current experimental approaches can only explore a fraction of the chemical composition at a time. As a result, model building, prediction, experimentation and verification have to be performed in an iterative loop to constantly update our knowledge about the system [1]. A large number of software tools and environments have been developed to address each step of the "systems biology" loop [2]. In this thesis, I focus on a few key components: building a model of a chemical system, visualizing such models to improve comprehension and understanding, and simulating models for predictive analysis.

The reaction network specification is the classical framework used for building chemical systems. However, biochemical molecules and complexes are highly structured and modular objects [3], which are not handled natively by the reaction network specification. The rule-based modeling specification is a recent development designed to handle arbitrarily structured molecules and complexes [4]. The technology and methods developed in this thesis are applicable to rule-based models.

1.2 REACTION NETWORKS

1.2.1 Specification

The predominant mathematical framework for modeling cell signaling is the reaction network. A typical **reaction** looks like this:

$$EGF + EGFR \xrightarrow{k} EGF\_EGFR$$

EGF, EGFR and EGF_EGFR are called **chemical species** and the reaction consumes the species EGF and EGFR and produces the species EGF_EGFR and proceeds at a rate proportional to the rate constant k and the reactant concentrations. The semantics of the species labels are a matter of convention. For example, to those familiar with growth factor signaling in human and cancer cells, EGF is recognizable as the epidermal growth factor molecule, EGFR as the cognate receptor molecule that binds EGF, and by convention, EGF_EGFR is the complex of EGF with EGFR.

A **reaction network** is a set of chemical species and reactions. The reaction network has a usual representation as a set of ordinary differential equations (ODEs) and the time-evolution of species concentrations can be simulated using ODE integration (species concentrations take continuous numeric values) or stochastic simulation (species concentrations take integer values).

There are many publically available ODE solvers, e.g. CVODE from SUNDIALS[5]. There also many stochastic simulators, mostly variants of Gillespie's stochastic simulation algorithm[6], [7].

1.2.2 Community Efforts

There are efforts in the biochemical modeling community to standardize representations of reaction networks and promote collaboration and reuse of published models. The Systems Biology Markup Language is a widely used interchange format that is supported by many software frameworks (sbml.org,[8]). Standard vocabularies and semantics have also been proposed for model annotation (MIRIAM [9]), simulation (SED-ML [10]), and visualization (SBGN [11]). The COMBINE initiative (co.mbine.org) meets regularly with the goal of improving these standards with community guidance. A review of current standards can be found here [12].

BioModels is a central database that collects and curates standard versions of models published in the literature (biomodels.org, [13]). However, as of 2015, less than half of the submitted models have been curated (data provided by biomodels.org), and an even smaller percentage of them have been annotated to the full extent possible (data not shown). This is because reaction networks are being constructed that are increasingly large in size (e.g. Chen et al. provide a model of ErbB signaling with more than 800 reactions [14]), and significant investment of time and human resources is necessary to encode semantic annotations of species and reactions in a manner that is consistent across models [12], [15]–[17].

1.2.3 Combinatorial Complexity

Many of the problems with using reaction networks can be traced to the complexity of biochemical systems. Biochemical molecules and complexes are largely composed of modular elements (domains, motifs and binding sites), and a single molecule or complex can access many combinations of these elements, each unique combination being called a **microstate** [18], [19]. For example, consider a protein with 10 phosphorylation sites. The particular configuration where none of the sites are phosphorylated is an example of a microstate. Because each site could be unphosphorylated or phosphorylated, the protein has a total of 2^10 (1024) unique microstates. The phenomenon where a small number of modular elements generates a combinatorially large state space is called **combinatorial complexity** [18], [19].

In spite of the combinatorial complexity, many microstates could behave similarly where the function of the protein is concerned. A group of microstates that are indistinguishable when measuring a particular property or behavior is called a **macrostate**. For example, the function of the above protein may simply be dependent on whether it has at least one site phosphorylated or none. If so, the function may be characterized in terms of two macrostates: unphosphorylated (containing one microstate) and phosphorylated (containing 1023 macrostates). Now, suppose the protein had two domains named X and Y, and X has 4 of those phosphorylation sites, while Y has the remaining 6. Also, suppose a second function of the protein is modulated by the phosphorylation status of X and Y individually. Characterizing this function requires that the protein be represented using four macrostates: XY (1 microstate), XpY (15 microstates), XYp (63 microstates), and XpYp (945 microstates). This highlights an important aspect of biochemical complexity: the relevant organization of microstates into macrostates is strongly dependent on which inputs and outputs are being considered.

Combinatorial complexity could be an evolutionarily favorable feature. Evolution models have shown that network complexity that arises from modularity tends to increase under selection pressure [20], since it allows for selection of both robust and sensitive responses, adaptation of existing structures for different functions, and fitness against deleterious mutations [20], [21]. Combinatorial complexity cannot be ignored if models are to be constructed that are useful outside the narrow context in which they were first built.

1.2.4 Issues

The number of valid microstates in a model depends on (i) the number of modular elements hypothesized by the modeler and (ii) the number of valid combinations of the hypothesized modular elements. The first part relates to the assumptions made by the modeler about the system, and the second part can be formally derived from those assumptions. For example, in Figure 1-1, we see three types of molecules A, B, C. A has two sites that bind B and C respectively when phosphorylated. Given these model assumptions, one can compute that the molecule A exists in 4 states: the unphosphorylated state A, the two states where only one site is phosphorylated Ap0 and A0p, and the doubly phosphorylated state App. The molecules B and C exist in 1 state each. There are 2 possible A-B complexes since there are two states of molecule A in which the B-binding site is phosphorylated (Ap0-B and App-B). With similar reasoning, there are 2 possible A-C complexes (A0p-C and App-C). There is only 1 A-B-C complex (App-B-C). In all, the system has 11 microstates.

However, when building a reaction network, the modeler is free to choose an arbitrary macrostate organization, say A'=A, Ap'={Ap0, A0p, App}, AB'={Ap0-B,App-B}, AC'={A0p-C,App-C}, ABC'=App-C. The choices made are not apparent in the specification itself, for example, given only A', Ap', AB', AC' and ABC', we cannot infer how many sites are on A, B and C and how phosphorylation affects binding at those sites. The modeler is also free to make arbitrary inclusions and exclusions to the state space, which override the formally derived set of possibilities and introduce hidden errors and bias. The arbitrariness in defining micro-to-macro mappings hinders the reuse and aggregation of models. For example, a species EGFR_p in one model may refer to phosphorylation at a different site compared to EGFR_p in another model. Because of these issues, reaction networks become opaque to those not involved in its construction and require significant investment of time to understand, reuse or modify beyond their original use [22].

The community response to these specification issues is to improve species annotation [17], standardize network representation [8] and increase resources for manual curation [13]. However, they do not address the fundamental nature of these issues: (i) there is a size disparity between the model assumptions and the reaction network, (ii) actions that require human effort (editing, aggregating, annotating, curating) scale poorly for large networks, and (iii) manual curation does not guarantee correctness when enumerating combinations.

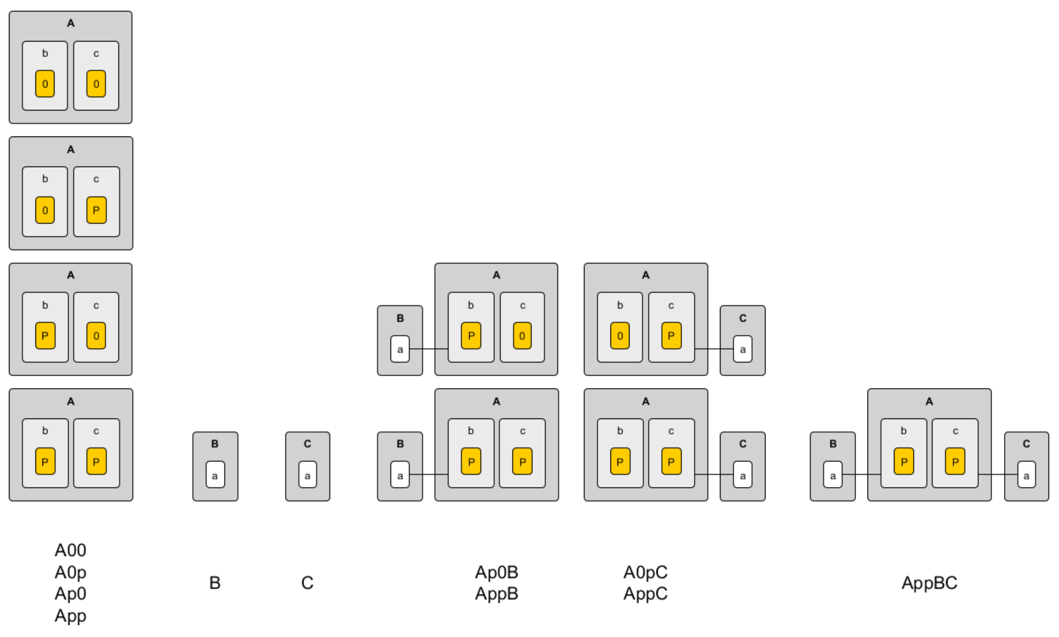


Figure 1-1 An example of combinatorial complexity. Consider a model with molecule types A, B, C. A has sites b,c that can be in 0 (unphosphorylated) and P (phosphorylated) states respectively and which bind molecules B and C respectively on phosphorylation. These assumptions translate to 11 different microstates organized under 6 macrostates: 4 A molecules, 1 B molecule, 1 C molecule, 2 A-B complexes, 2 A-C complexes and 1 A-B-C complex. A reaction network may be built by choosing 6 macrostates as shown above, or any other arbitrary choice. Although the state space is determined formally by the assumptions, the reaction network modeler is still free to override them by including or excluding chemical species at will, which introduces error and bias.

1.3 RULE-BASED MODELS

Rule-based models improve upon reaction network models by using a graph-based specification for molecules, complexes and reaction classes. A number of rule-based frameworks exist, with BioNetGen [4], [23]–[26], Kappa [27]–[29], and Simmune [30]–[32] being the most popular ones and with very similar graph abstractions.

1.3.1 Graph syntax – Molecules and Complexes

In order to represent a chemical species such as a molecule or complex in a reaction network, the modeler would have to first specify an informative label, define the semantics of the label convention used, and if they are so inclined, encode the structures present in the species in the form of annotations. In contrast, rule-based models encode molecules and complexes using graphs with consistent semantics. The structure of the graph explicitly encodes the sites and binding interactions present in the species and the graph itself serves as an identity for the species, since its uniqueness can be formally determined by graph isomorphism [33], [34]. BioNetGen and Kappa provide a syntax for building graphs of molecules and complexes using alphanumeric strings. Here, we will demonstrate the BioNetGen syntax [24], [25] by using it to systematically build the complex AppBC (shown in Figure 1-1).

The **complex** has three **molecules** A, B, C, and they are represented as a dot-separated list.

```
A.B.C
```

Molecule A has **components** b,c. Molecule B has component a. Molecule C has component a. Components are represented as a comma-separated list enclosed within brackets and placed adjacent to the respective molecule.

```
A(b,c).B(a).C(a)
```

Components b,c have **internal states** named P, which denotes the phosphorylated state. Internal states are prefixed with a ~ symbol and placed next to the component names.

```
A(b~P,c~P).B(a).C(a)
```

The complex is formed as a consequence of two binding interactions, or **bonds**. One of the bonds is between the components A(b) and B(c), and is represented by a ! symbol and a tag 1 placed adjacent to the bonded pair.

```
A(b~P!1,c~P).B(a!1).C(a)
```

The other bond in the complex is between components A(c) and C(a). We use the ! symbol followed by the tag 2 to differentiate it from the other bond.

```
A(b~P!1,c~P!2).B(a!1).C(a!2)
```

The syntax is not affected by the ordering of components within molecules, or the ordering of molecules within a complex, or the ordering of bonds by tag IDs. So, the following strings all represent the same graph and the same chemical species as the one above.

```
A(c~P!2,b~P!1).B(a!1).C(a!2)

A(b~P!1,c~P!2).C(a!2).B(a!1)

A(b~P!2,c~P!1).B(a!2).C(a!1)

B(a!1).A(b~P!1,c~P!2).C(a!2)
```

1.3.2 Patterns

Properties of a graph can be formally extrapolated to supergraphs that contain the graph as a subgraph. In the rule-based framework, the **pattern** is a partial graph of a complex. Multiple complexes may share the same pattern as a subgraph, so the pattern also formally defines a class of complexes with a shared set of structures. This enables the formal, precise and flexible definition of macrostates based on the structures shared by the microstates. For example, the macrostate

{A,A0p,Ap0,App} in Figure 1-1 is defined by the following structural constraint: the species must be composed of one A molecule that is not bound to any other molecule. In BioNetGen syntax, it is sufficient to specify the following pattern:

```
A(b,c)
```

Here, we have not specified the internal states, so the internal states are not used in the matching process. However, we have specified that the components are unbound (no ! tags), so the pattern selects complexes with unbound b and c components, which are as follows (underline emphasizing the shared subgraph):

```
A(b~0,c~0)

A(b~P,c~0)

A(b~0,c~P)

A(b~P,c~P)
```

Another example is the macrostate {Ap0, App, AppC} which selects complexes with an unbound phosphorylated B-binding site on molecule A. In BioNetGen, it is sufficient to specify the pattern

```
A(b~P)
```

This will automatically match the complexes (underline emphasizing the shared subgraph):

```
A(b~P,c~0)

A(b~P,c~P)

A(b~P,c~P!1).B(a!1)
```

By providing a formal way to encode structural constraints, the pattern syntax removes the burden of manually defining the semantics of each micro or macro state. It also removes the burden of manual enumeration and verification. In contrast, any modeler interaction with a reaction network, such as specifying a new macrostate as an output, requires sequential or pairwise manual examination of the state space, which is time-intensive and introduces error and bias.

1.3.3 Reaction Rules

The **reaction rule** is a reaction composed by using patterns as reactants and products. Since patterns map to classes of complexes with shared structures, the reaction rule maps to a class of reactions on those shared structures. The entire reaction class can be parameterized by mapping the reaction rule to a rate law. For example, consider the reaction rule:

```
A(b~P) + B(b) -> A(b~P!1).B(a!1)          k
```

The reactant patterns specify the shared set of structures that enable a species to participate in this reaction class. Here, the pattern A(b~P) specifies that the b component on A must be phosphorylated and unbound, and the pattern B(b) specifies that the a component on B must be unbound. The product patterns represent a graph transformation relative to the reactants. Here, the transformation implemented is the addition of a bond !1 between components A(b) and B(a). By default, the reaction class is assumed to have an elementary rate law and the expression k specifies the rate constant uncorrected for symmetry and multiplicity [24], [25].

By using the patterns to select combinations of species, the entire reaction class can be enumerated. Here, the pattern A(b~P) maps to the species A(b~P,c~0), A(b~P,c~P) and A(b~P,c~P!1).C(a!1), and the pattern B(a) maps to the species B(a). The combinations of these species (3x1) lead to the following three reactions, all of which are parameterized by the rate constant k (underline to emphasize the graph overlap with the reaction rule):

```
A(b~P,c~0) + B(b) -> A(b~P!1,c~0).B(a!1)                    k

A(b~P,c~P) + B(b) -> A(b~P!1,c~P).B(a!1)                    k

A(b~P,c~P!2).C(a!2) + B(b) -> A(b~P!1,c~P).B(a!1).C(a!2)    k
```

In contrast to a reaction network, the reaction rule is explicit about which structures are necessary to define the reaction class. Also, the number of reaction rules needed to model a system depends on the number of such reaction classes with unique kinetics. A small number of modular independent interactions will require a small number of reaction classes to specify them, even if their combinations generate a much larger reaction network [18], [19].

1.3.4 Model Specification and Simulation

The domain-oriented approach makes rule-based models easy to modify and extend, as well as making explicit the contribution of each type of protein structure to each reaction class. When the corresponding reaction network is finite, it can be generated automatically from the rule-based specification [24]. This has enabled building models for systems where a small number of molecule types lead to very large networks, e.g. model of Syk activation in FcεRI receptor signaling (24 rules – 3680 reactions) [35], early events in epidermal growth factor signaling (39 rules – 3749 reactions) [36], ultrasensitivity in multi-site phosphorylation [37], etc.

If the reaction network is infinite or too large to be stored in computational memory, the rule-based specification can still be simulated using network-free approaches [38]–[40], or a hybrid approach between network-based and network-free [34]. This has made the rule-based specification attractive for specifying models with infinite state spaces generated from finite reaction classes, e.g. receptor aggregation by ligand crosslinking [41], Lat crosslinking [42], CaMKII activation [43], etc.

The reaction rule is also a compact and portable unit, which makes reaction rules suitable for cataloging kinetic processes. A number of rule-based knowledge bases have been constructed in recent times, such as for FcεRI receptor signaling [44], T-cell receptor signaling [45], signaling from the ErbB receptor family [46], yeast pheromone signaling [47], etc. Some of these repositories have 100s of rules that will result in a practically infinite network, and therefore cannot be simulated at all using network-based methods.

1.3.5 Outstanding Issues

Recently, a detailed mechanistic model of a whole cell was published [48], in which reaction networks were embedded as submodels. It is anticipated that such whole cell comprehensive models will eventually involve integration of rule-based databases of individual pathways, such as [44]–[46]. Model frameworks are also being built to use rule sets that can be re-combined in modular ways [49], so we expect to see larger rule-based models with 10 or more molecule types. However, currently, there is no automated way to visualize the signaling implicit in the rules as a global regulatory network or pathway diagram, which will be necessary to understand such large models. We address this in Chapter 2.

Another problem inherited from reaction networks is that defining reaction classes with unique kinetic parameters does not constrain the thermodynamics of the system. When there are loops of reaction mechanisms in the system, detailed balance constraints apply (sum of free energies of the reaction around the loop should be zero), but the specific loops present are not immediately obvious when reactions or reaction rules are being built. Manual verification is needed to get the correct model, and this can be resource intensive when there are a number of cooperative processes acting on the same molecule or complex. The lack of constraint on detailed balance is especially noticeable in the very large rule-based models [44], [46], where the varied branching structure of complexes results in many such reaction mechanism loops.

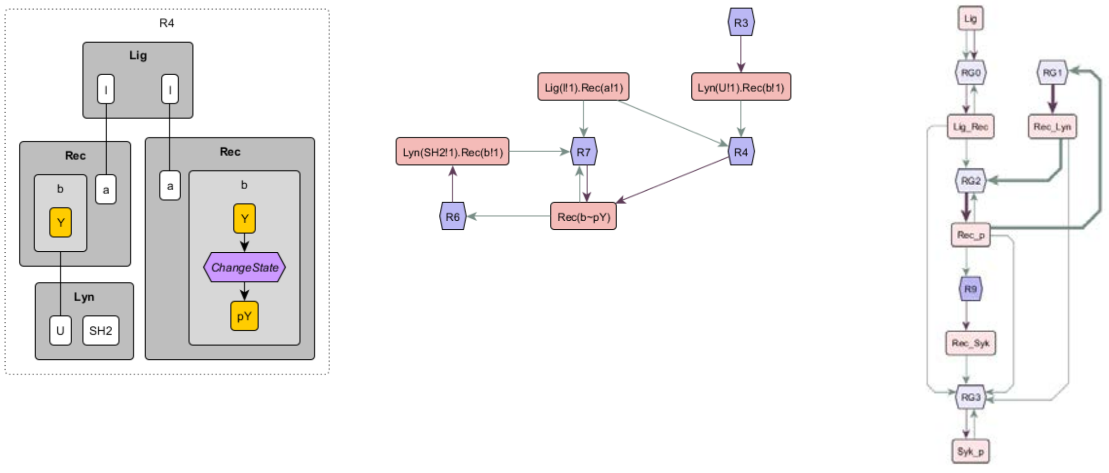
A class of approaches have emerged to address this issue, which we call “energy-based rule-based modeling”. In this specification, graph isomorphism is used specify energy contributions from molecules, sites and binding interactions [50]–[52]. In energy-based BioNetGen, energies of species are measured by counting matches of the species to user-specified energy patterns. Reaction rules are defined with rate laws based on the Arrhenius equation, called “energy rules” [50], and reactions generated from these rules are guaranteed to satisfy detailed balance. However, calculating the energy-based reaction rate for a reaction depends on both reactants and products, and this makes it incompatible with network-free simulation, where information on the products is not available until the reaction event has fired. We address this problem in Chapter 3.

2.0 VISUALIZATION OF RULE-BASED MODELS

2.1 SYNOPSIS

Visualizing biochemical interactions has a long history of being conveyed through symbolic, pictorial and graphical representations. Typically, these systems contain many types of physical entities (such as molecules and complexes) and processes (such as reactions) and relationships defined between them. The primary goal of this chapter is to develop automated visualizations for rule-based models that convey encoded information at the level of individual processes (local level) as well as at the level of the model itself (global level). Figure 2-1 summarizes the contributions made in this chapter.

Visualizations are hard to evaluate using exact metrics and attributes, however, there are broad aspects using which they can be compared against each other. In **Section 2.2**, I list some of the concerns applicable to visualization of rule based models. In **Section 2.3**, I discuss some of the standard diagramming procedures that have been developed for the signal transduction modeling community. In **Section 2.4**, I discuss automated methods that have been developed for reaction networks and the different rule-based frameworks, and their advantages and disadvantages. In **Section 2.5**, I provide formal graph abstractions that I have developed for rule-based models which enables generation of local and global visualizations. In **Section 2.6,** I apply these methods to rule-based models in the literature.



(A) (B) (C)

Figure 2-1 Summary of contributions to visualization of rule-based models. Panel A shows a compact visualization of a single rule, in which the change implemented by the rule is shown as a graph transformation (ChangeState). Panel B shows a visualization of regulatory interactions between a small number of rules (R3, R4, R6, R7), called the regulatory graph. Dark edges show production and consumption of states, and light edges indicate a regulatory influence from a state to a rule. In this chapter, I provide theory and implementation that generates a regulatory graph from rule syntax. Panel C shows a visualization of a rule-based model as a regulatory graph, similar to a pathway diagram. The nodes on this graph represent groupings of rules and states. In this chapter, I provide a systematic pruning, grouping and coarse-graining procedure to generate compact pathway diagrams from the aggregated regulatory graphs of rules.

2.2 VISUAL CONCERNS

2.2.1 Content versus Intent

The word *model* is widely used, but it can mean a variety of things. In the context of modeling biochemistry, a *model* can imply a set of statements in the mind of the modeler representing biochemical knowledge, i.e. a *mind model*. The modeler translates these statements into machine-readable mathematical symbols and relationships, resulting in a *formal model*. When either of these representations is translated into visual objects in a visual medium, the resulting model is a *visual model*.

In this chapter, I will use *model* to refer to the formal model in a particular mathematical framework such as rule-based modeling or reaction network modeling. The specific mathematical statements about entities and processes used in the model, as well as any systematic transformation thereof, will be referred to as the *content* of a model. The biochemical statements in the mind of the modeler that were used to create the model will be referred to as the *intent* of the model. A mapping of either of these representations to a set of visual objects and relationships will be referred to as a *visualization*.

The separation of these concepts allows us to rationally treat the problem of interconversion. The intent of a model is limited only by the vernacular of biochemistry, and can be considered unrestricted for all practical purposes. The content of a model is restricted by the formal definitions of the mathematical framework used. The visualization of both model intent and content is restricted at a formal level by the choice of visual objects and notations, and at a practical level by the cognitive and aesthetic appeal of the generated diagram. The methods that I develop in this chapter will use the content of a rule-based model to generate the visual representation, but will aim to convey the intent of the model and appeal to intuition.

2.2.2 Local versus Global

Both model intent and content are typically composed of individual statements about small numbers of entities, for example, a reaction with two reactant species and one product species representing a particular binding interaction. A visualization examining one such statement is considered to be *local*. A visualization of the whole model, composed from many such statements, is considered to be *global*. At the local level, the focus is on detail and the visualization is tailored to present the maximum amount of detail that is possible. At the global level, the focus is on identifying higher-order motifs and trends in the model, and a certain level of coarse-graining may be needed to uncover these. A comparison to real-world maps is applicable here: maps of larger geographic areas have to approximate features found in detailed maps of smaller areas in order to be useful. In this chapter, we will specifically distinguish between local and global visualizations of rule-based models and we will define appropriate coarse-graining procedures during the generation of global visualizations.

2.2.3 Flow versus Adjacency

An important visualization objective in diagrams of biochemistry is to emphasize the temporal order of events, particularly sequences and cycles [53]. Delineating specific paths, such as pathways, feedback loops and feed-forward loops, is important for visual comprehension and I will collectively refer to them as *signal flows*. Encoding the temporal order along a graphical dimension, i.e. aligning signal flows top-down or left-right in the diagram can drastically improve visual comprehension [54]. However, a high density of edges can preclude a good visual alignment of flows, in which case force-directed layouts are efficient. However, these layouts emphasize adjacency relationships [55] and have poor visual comprehension. For a generated diagram to be useful, its size and edge density needs to be sufficiently sparse to align signal flows in an optimal manner.

2.2.4 Art versus Automation

There are many aspects to producing a diagram: synthesizing the content, defining the notation and attribute mappings, drawing the actual elements on some visual media, laying out the visual elements in an optimal manner. When every one of these aspects is left to the discretion of the diagrammer, then the diagram becomes a one-off art project that requires a heavy investment of time. It would be preferable to automate as many of these steps as possible. However, artistic and aesthetic considerations do play a role in the usefulness of a diagram [54], so a balance is necessary between automation and artistic discretion. In this work, we focus on automated generation of the content of a diagram, and coarse-graining of diagrams with minimal user input. The development of automated layout algorithms is beyond the scope of this work.

2.2.5 Abstraction versus Enumeration

When defining a naming system or notation for a set of objects, there is a tendency to enumerate all possible states of those objects and to assign labels or features to every possibility indiscriminately. I call this the *enumerative* approach. Manual cataloguing, annotating and diagramming approaches are typically of this type. Another way of defining a system or notation is to generalize over the set of objects and create a few carefully defined types or classes of objects, then map the real-world possibilities as instances of those classes. I call this the *abstract* approach. Mathematical modeling frameworks are typically of this form. Enumeration is superficially the most direct approach, but it inevitable encounters a level of complexity that hinders automation and comprehension. Abstraction is much harder to do, but the right abstraction for a task can drastically reduce the complexity involved, improve clarity and enable automation.

2.3 VISUAL STANDARDS

A set of approaches exist whose goal is to standardize drawing schemes, notations and semantics for biochemical diagrams. In general, there are two types of maps: maps that emphasize the structural components of the system and binding interactions, which I call *contact maps*, and maps that emphasize temporal order and signal flow, which I call *flow maps*.

Contact maps typically show one instance each of every type of entity in the system and use edges to show binding interactions, processes and influences of entities on processes. These maps are typically edge-dense and laid out to emphasize structural connectivity and adjacency. Contact maps present a global structure-centric view, but they are not very useful for local and detailed representations of processes. Also, large and complex contact maps make it difficult to delineate signal flows.

Flow maps typically have one or two major types of nodes indicating entities and/or processes and are laid out such that the major signal flows are aligned along the top-down or left-right dimension. These are typically edge-sparse and have improved visual comprehension, but structural relationships may not be evident from the map. Flow maps can be useful at both local and global levels, i.e. to represent individual processes in detail as well as a network of signal flows.

Here I review the standardized diagramming approaches, point out whether they are contact maps or flow maps, and discuss their positives and drawbacks.

2.3.1 The Molecular Interaction Map (MIM)

The Molecular Interaction Map [56]–[58] was an early standard diagramming procedure that falls in the class of contact maps. It provided a simple and flexible abstract visual notation that allowed representation of domains, motifs, covalent modifications and binding interactions as visual objects and processes as edges between entities. The MIM introduces the notion of *contingency relations*, which are influences of entities on processes and these were also represented as edges. Five basic contingencies are defined: stimulation, requirement, inhibition, absolute inhibition and catalysis. The MIM developers also provide an XML-based schema that can be used to generate machine-readable diagram. The MIM suffers from the same problems as all contact maps, such as high edge-density and de-emphasis of signal flows.

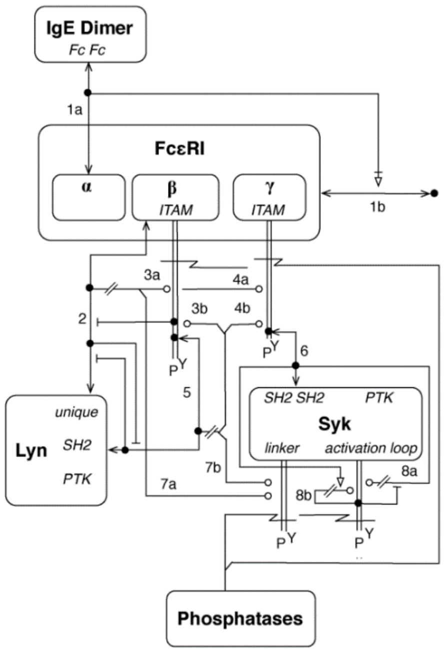


Figure 2-2. Molecular Interaction Map of the Faeder et al. model of signaling from the FcεRI receptor. Model is from Faeder et al [35] and diagram is from Chylek et al [59]. Boxes and labels represent molecules and domains. Edges show covalent modifications (double lines) and binding interactions (bidirectional arrows). Edges with special arrowheads show contingency relations such as catalysis (circles), cleavage (zigzag), stimulation (triangle) and inhibition (flat arrowhead).

2.3.2 The Systems Biology Graphical Notation (SBGN)

SBGN [11] was developed as a successor to MIMs with the goal of making it comprehensive with regard to biochemical representation. Recognizing that different visualization needs exist, it includes three different categories of maps: the process description map, the entity relationship map, and the activity flow map. SBGN also comes with XML-based schema to enable software support, with the schema being actively developed with community guidance [60]. There are a number of software that now support SBGN, such as SBGNViz [61], Paxtools [62], CellDesigner [63], Cytoscape [64], and Rxncon [65].

2.3.2.1 SBGN Process Description diagram

The SBGN Process Description falls in the category of flow maps and is targeted towards visualizing individual reactions in detail. Physical entities can be represented as pools of other entities, enabling the representation of detailed molecular structures: complexes with many molecules and molecules with domains and covalent modifications. Processes are represented by a different type of node, and directed edges indicate whether an entity is consumed or produced by a process or whether an entity influences a process. The Process Description diagram is useful for representing reactions where information is available about the internal structure of participating species, e.g. a reaction where a receptor is phosphorylated by a recruited kinase. Technically, the Process Description diagram can be aggregated from individual reactions to a global visualization, however its visual comprehension scales poorly with number of reactions, primarily due to the amount of detail represented. In comparison to the other SGBN diagrams, the Process Description diagram has the most concise abstraction and semantics and is more applicable for visualizing model content rather than intent.

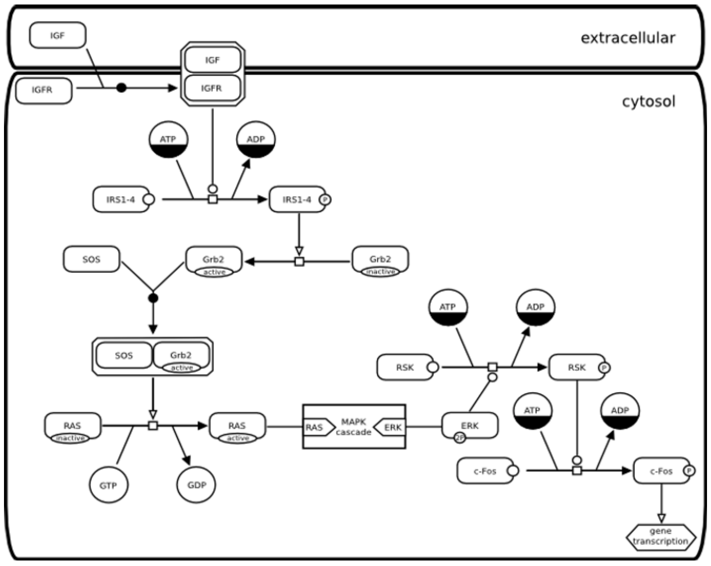


Figure 2-3. SBGN Process Description diagram of MAPK signaling from the insulin growth factor receptor. Diagram is from sbgn.org. Glyphs on edges represent the type of interaction: binding (dark circles), conversions (empty squares). Edges with arrowheads represent requirements (empty circles and triangles).

2.3.2.2 SBGN Entity Relationship diagram

The SBGN Entity Relationship diagram falls in the class of contact maps and therefore suffers from the limitations of this class, namely high edge density and lack of alignment of signal flows. However, in comparison to MIMs, the notation used here is more comprehensive with regards to biochemistry. Entity classes are derived from the Systems Biology Ontology [12] and includes support for many types of material entities (genes, proteins, small molecules, sugars), functional entities (gene start site, etc.) and covalent modifications (phosphorylation, acetylation, etc.). Edges describe binding and other processes, as well as influences of entities on processes, with each type of edge requiring a distinct arrowhead. Unlike the Process Description, the approach used here is enumerative rather than abstract, so the diagrams become visually cluttered very quickly from the different entity node shapes and arrowheads. The semantics of entities and arrowheads are standardized [11], and designed to represent model intent.

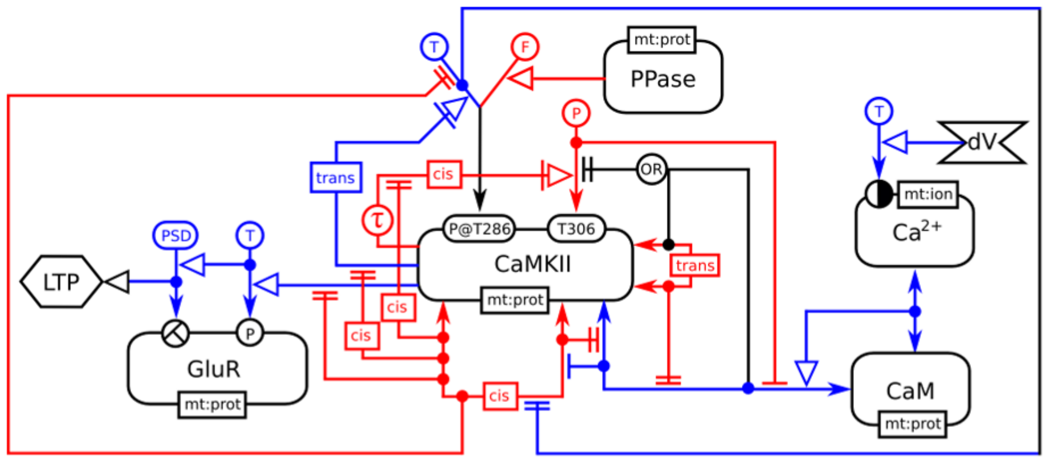


Figure 2-4. SBGN Entity Relationship diagram of CAMKII signaling. Diagram is from sbgn.org and Le Novere et al. [11]. Entities such as molecules and domains are represented as boxes with labels. Edges with special arrowheads indicate a variety of processes and influences of entities on processes.

2.3.2.3 SBGN Activity Flow

The SBGN Activity Flow diagram is a flow map constructed using only activity nodes and influence edges. Activity nodes are vaguer than process or entity nodes, and so the activity flow diagram is the most poorly defined of all three SBGN maps [11]. An activity is typically some combination of process and entity, e.g. a phosphorylation process at a phospho-motif site, or a catalytic process that generates a particular metabolite. However, because of its simplicity, it is similar to hand-drawn diagrams by biologists, and it is useful to show summaries of signal flow in model. Similar to methods we describe later in the chapter, model content in the form of process descriptions may be used to generate the activity flows using a coarse-graining method provided by Vogt et al. [66].

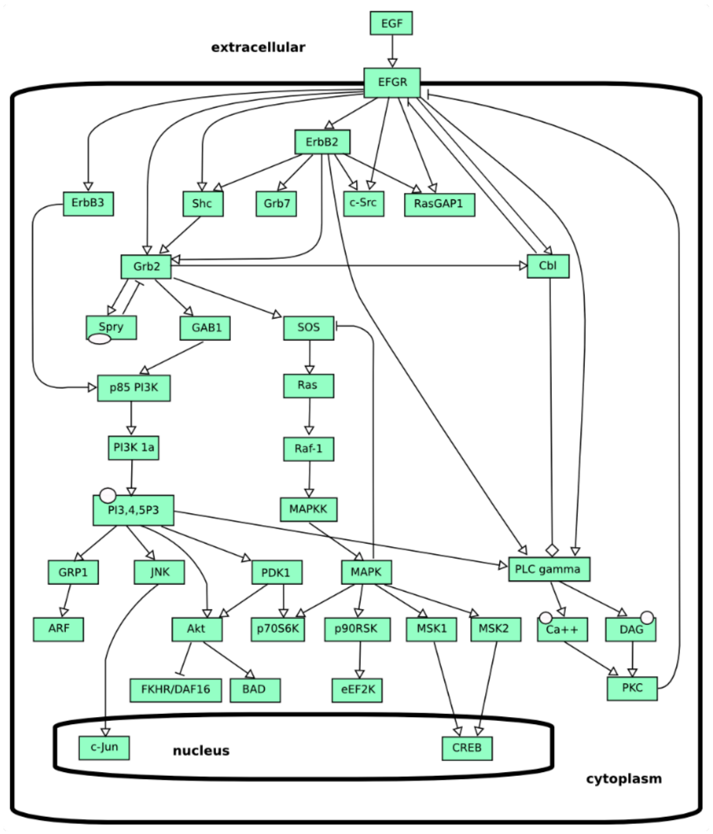


Figure 2-5. Activity flow diagram of the epidermal growth factor signaling pathway. Diagram is from sbgn.org and a previous version was published in Le Novere et al. [11]

2.3.3 The Extended Contact Map

The Extended Contact Map [59] is conceptually similar to the SBGN Entity Relationship and the MIM. It borrows aspects from both maps and applies them to rule-based models. Additionally, it provides a comprehensive and enumerative description of biochemical processes. Domain structure is represented as nested nodes, which allows more flexibility than the entity relationship diagram in representing molecular structure. The number of arrowheads used is minimized: contingencies are limited to stimulation, catalysis that is transformative (such as phosphorylation) and catalysis that is destructive (such as proteolysis). The map is designed to showcase model intent rather than content, and is paired with a model guide where the model content is described in detail. Edges on the map are annotated with index entries in the model guide.

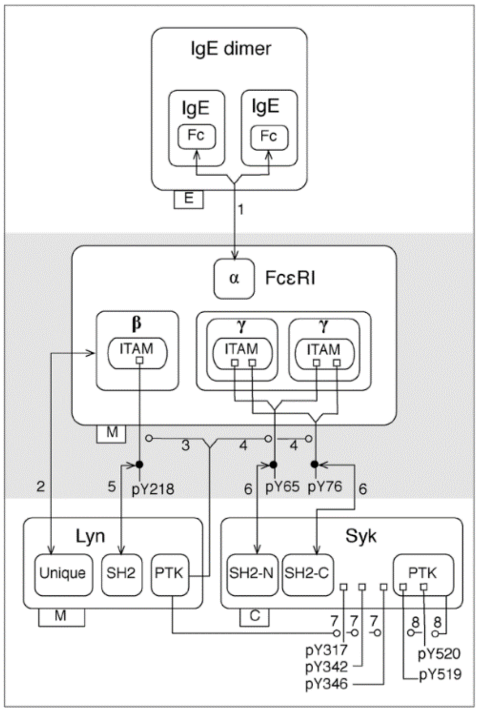


Figure 2-6. Extended contact map of FcεRI signaling modeled in Faeder et al. Model is from [35] and digaram is from Chylek et al. [59].

While the limitations of the contact map format are not completely resolved, Chylek et al. do provide guidelines on how to arrange molecule types in a top-down hierarchy that is best reflective of signal flow [59]. The major limitation of this map is the level of human effort required to build and lay out the map, build the model guide and annotate the map using the model guide. Also, commercial software are needed to build the map and there is no standard schema underlying the map that can be used to extend software support.

2.4 VISUALIZATION OF RULE-BASED MODELS – STATE OF THE ART

In this section, I discuss graph-based abstractions that have been used to visualize rule-based models and reaction networks. These are conceptually different from the standards in the previous section in that they have a solid mathematical definition, therefore, with appropriate mappings between the mathematical objects involved, these maps can be generated automatically from model content with relative ease. In this section, I will describe and evaluate formal abstractions currently used in the rule-based software frameworks such as BioNetGen [67], Kappa [68], Simmune [30], VCell [69] and Rxncon [65]. The reader is advised to be familiar with the basic concepts of reaction networks and rule-based models outlined in Chapter 1.

2.4.1 The Reaction Petri Net

The Petri Net is a directed bipartite graph with one node type for entities or states and another node type for processes. For chemical reaction networks, entities are chemical species and processes are reactions, and edge direction indicates whether a reaction consumes a species or produces a species. In Figure 2-7, we show a Petri net composed of the following reactions:

```
Rxn1: A + B -> AB

Rxn2: AB -> ABp

Rxn3: ABp + C -> ABpC
```

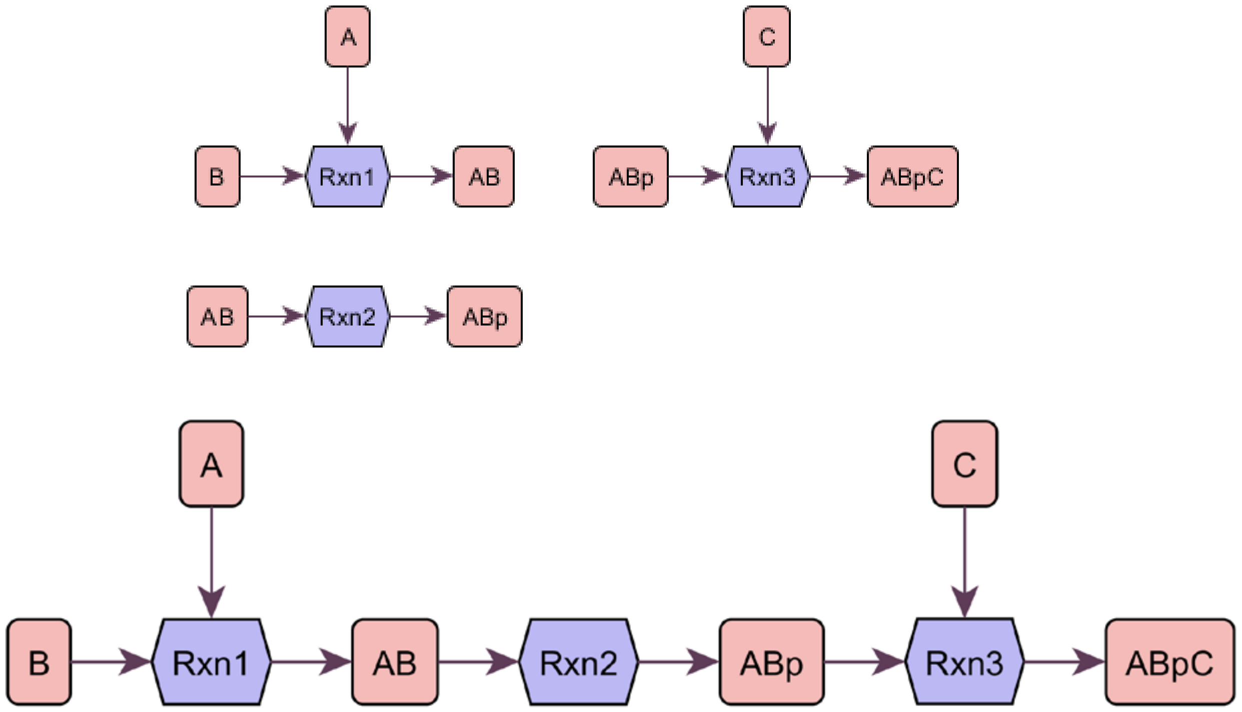


Figure 2-7. Petri net of a model of three reactions. The reactions are Rxn1:A+B->AB, Rxn2:AB->ABp, Rxn3:ABp+C->ABpC. The Petri net of each reaction is shown above and the combined Petri net is shown below. Aggregation of Petri nets to show signal flow is trivial because each chemical species (A, B, C, AB, ABp, ABpC) is produced or consumed in its entirety by individual reactions.

As mentioned previously, reaction networks can be automatically generated from rule-based models, and frameworks such as VCell [69] and BioUML (biouml.org) have provided tools to generate and visualize the corresponding Petri nets. However, because of combinatorial complexity, even small rule-based models are capable of generating much larger reaction networks [35], [70], so the reaction network Petri net is not scalable for use with rule-based models.

2.4.2 The Site Graph

The site graph [28] is a nested graph used to represent patterns. Component nodes are nested within molecules and internal state nodes are nested within components. Undirected edges between components represent bonds. Variations on the site graph been used for visualizing patterns in most of the current literature on rule-based models. The site graph is similar to a pool of entities in SBGN Process Description, but it has additional information in the form of bonds between molecular components.

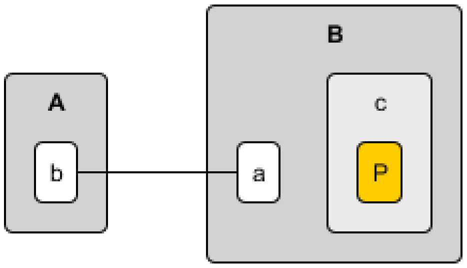


Figure 2-8. Site graph of a pattern A(b!1).B(a!1,c~P). Molecule A has component b, molecule B has components a and c, component c has internal state P, and components a and b are linked by a bond.

2.4.3 The Rule Petri Net

A reaction rule is composed of reactant and product patterns, similar to how a reaction is composed of reactant and product species. So, similar to reactions, a reaction rule can also be visualized as a Petri net. Additionally, since patterns have site graph visualizations, these site graphs can be embedded in the entity nodes of the Petri net. This provides an explicit and detailed local visualization of the rule as is present in the model, and this visualization is also compatible with SBGN Process Description. In Figure 2-9, we show Petri net visualizations of the following rules:

R1: A(b) + B(a) -> A(b!1).B(a!1)

R2: A(b!1).B(a!1,c~0) -> A(b!1).B(a!1,c~P)

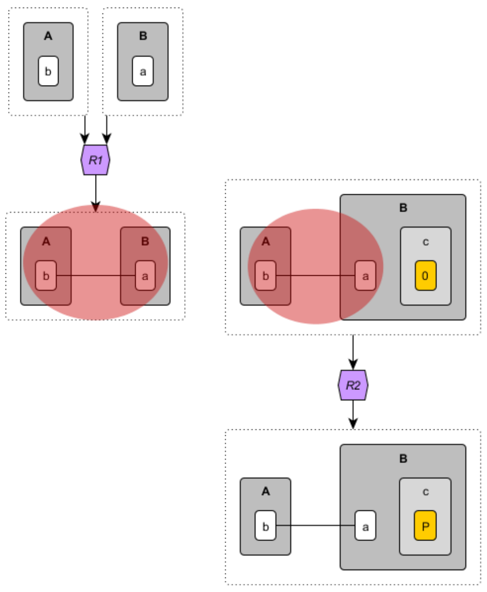


Figure 2-9. Rule Petri nets and partial overlaps. Shown here are rules R1 and R2. R1 produces a bond between sites A(b) and B(a). R2 phosphorylates site B(c) (note 0 to P) when in the A-B complex. Note that the bond is shared between the rules (red overlay), being formed in R1 and used as context in R2, but it is only a subgraph of the reactant pattern in R2. Because of such partial overlaps, rule Petri nets cannot be aggregated into a combined Petri net.

There are two problems with this approach. The first problem arises because a rule represents the necessary conditions to implement a graph transformation (Section 1.3.3), which means that some parts of the reactants are modified to generate the products, whereas some parts remain unmodified and are considered context for the implemented modification. However, representing reactants and products separately, as in a Petri net, does not necessarily convey quickly which parts are modified and which parts are not, and a viewer would have to manually compare the graph structures of reactants and products to arrive at this information. When the rule has sufficiently detailed context, repeating these structures on both sides of the rule can obscure the parts of the rule that are modified, which results in poor visual comprehension.

The second problem arises from partial overlaps between rules, which is not encountered in reaction networks. In a reaction network, a whole discrete chemical species is produced by one reaction and consumed by another. Petri nets of the individual reactions can be aggregated into a Petri net of the full model on which this flow of information is obvious. On the other hand, in a rule-based model, it is not necessary that the whole pattern produced in one rule be consumed by another to constitute a signal flow. It is sufficient that the overlap is partial, i.e. some subgraph of the product pattern of one rule which was modified by that rule is now present as a subgraph of the reactant pattern of another rule (see Figure 2-9). A simple Petri net representation of multiple rules would not resolve these partial overlaps and would not be a useful global visualization.

2.4.4 The Formal Contact Map

The formal contact map is a site graph that shows one instance each of structures defined in the model (molecular, component, internal state) and one instance each of types of bonds present. It is the simplest map in the class of contact maps, and provides a concise summary of the structural relationships in a model, but does not show signal flow. The formal contact map can be generated automatically in Kappa [28], [71] and Rulebender, the BioNetGen GUI [26].

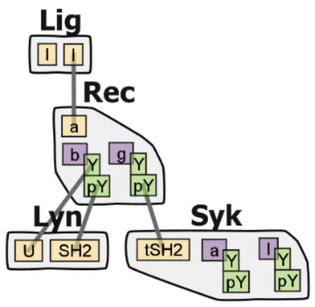


Figure 2-10. Contact map of FcεRI signaling modeled in Faeder et al. [35]showing types of molecules, components, bonds and internal states [35]. Diagram generated in Rulebender, the BioNetGen graphical user interface [26].

2.4.5 The Rule Influence Diagram

Both Kappa (http://kappa-dev.github.io) and Rulebender, the BioNetGen GUI [26], enable a visualization of every pairwise interaction within a set of rules, called the rule influence diagram. Generating this diagram involves characterizing every partial overlap between every pair of rules such as the one in Figure 2-9. While this can provide a picture of signal flow for a small set of rules, the number of overlaps that need to be examined grows as the square of the number of rules, which makes it infeasible for large rule-based models (10's – 100's of rules).

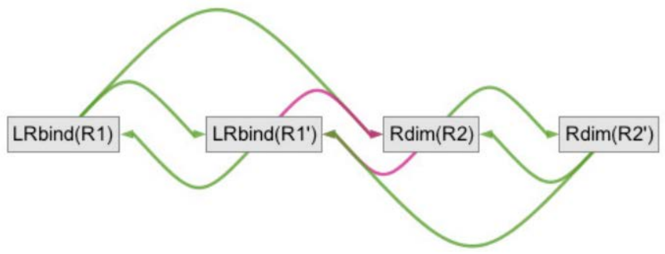


Figure 2-11. Rule influence diagram. Each rule is represented as a node and edges represent activation (green) or inhibition (purple) influences. Diagram is from Smith et al., generated in Rulebender, the BioNetGen GUI [26].

2.4.6 The Kappa Story

Kappa provides a visualization called the story by tracing and compressing the order in which simulation events happen in order to generate an observable state of interest [28]. The Kappa story is a good representation of signal flow, but computing it requires specific parameter choices and simulation to generate the traces, which can be infeasible for models with many rules. Also, the Kappa story is not strictly a global or complete visualization of the model, because each story is targeted at a specific endpoint state.

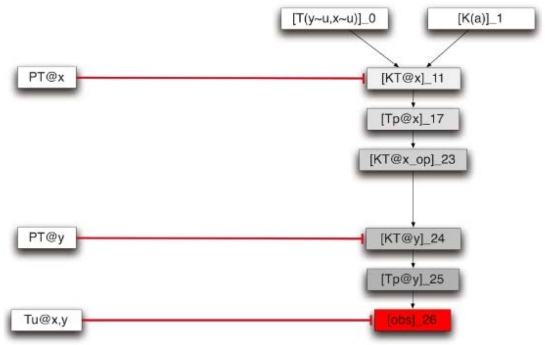


Figure 2-12. Kappa story showing a causal sequence of rules. The top two nodes are starting states. The other nodes are firings of rules showing the name of the rule and the simulation step. Diagram is from Danos et al. [28].

2.4.7 Simmune Network Viewer

To obtain a visual representation of reaction rules that can be aggregated into a global picture, the Simmune Network Viewer [32] first coarse-grains reaction rules to their fundamental molecule stoichiometries. For example, consider the two reaction rules (underline for emphasis):

```
A(b!1,x~on).B(a!1) -> A(b,x~on) + B(a!1)

A(b!1,y~on).B(a!1) -> A(b,y~on) + B(a!1)
```

Although they differ in their internal state specifications, they would both be coarse grained as:

```
A.B -> A + B
```

This form can then be converted into a Petri net of the rule (see Figure 2-13) and these rule Petri nets are aggregable unlike those in Section 2.4.3. Each entity node on this Petri net represents a particular arrangement of molecules and bonds in a complex.

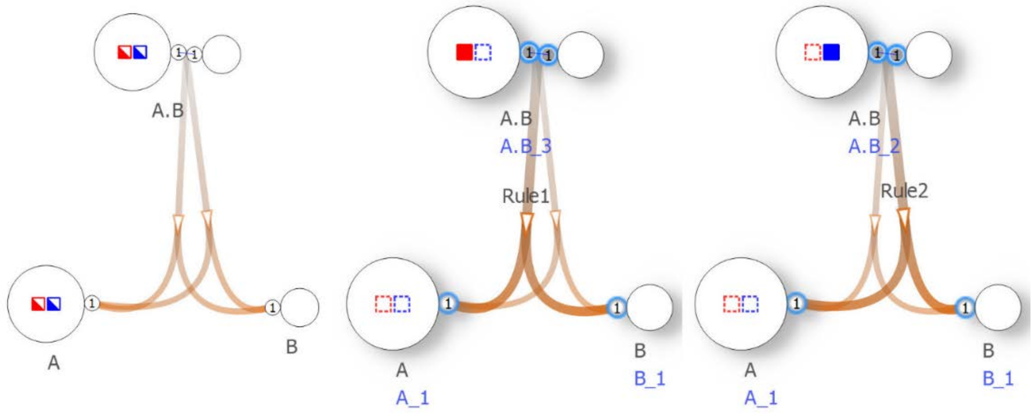


Figure 2-13. Simmune Network Viewer coarse-grains rules to molecule stoichiometry to enable a Petri net or bipartite graph representation [32]. In the first panel, we show two rules with stoichiometry A.B -> A + B. When neither rule is selected by clicking, the red and blue ‘features’ are in the “don’t care” state (half filled). In the second panel, Rule1 is selected and the features specific to Rule1 are displayed (red feature is “on”, i.e. filled). Similarly, in the third panel, Rule2 is selected and the features specific to Rule2 are displayed (blue feature is “on”).

The problem with this approach is that signal flows that are mediated through internal state changes are not represented on the visualization. For example, consider a molecule X with three sites that are sequentially activated. The rule set would be:

```
X(a~off) -> X(a~on)

X(a~on,b~off) -> X(a~on,b~on)
```

```
X(b~on,c~off) -> X(b~on,c~on)
```

From observing the rules, we can infer that activation of X(a) leads to activation of X(b), which in turn leads to activation of X(c). However, since all patterns here have the same molecule stoichiometry X, the Simmune Network Viewer cannot show the signal flow represented by these rules (see Figure 2-14).

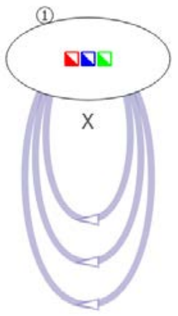


Figure 2-14. Simmune Network Viewer cannot show signal flow mediated through 'features' (internal states in BioNetGen). This diagram was generated from a sequential model where the red feature activates spontaneously, then red activates blue and blue activates green. In the Simmune Network Viewer, the different states of the molecule are represented by the same node X.

2.4.8 Rxncon Regulatory Graph

Rxncon [65] is a relatively new rule-based framework that emphasizes convenient model building and visualization. The Rxncon specification marries the enumerative approach of visual standards with the abstract approaches of rule-based frameworks such as BioNetGen, Kappa and Simmune. In Rxncon, entities can be "elemental states", such as a single bond or a single phosphorylated state, or they can be simple combinations of these states using Boolean operators such as OR and AND. Processes are drawn from a manually enumerated list of commonly encountered processes such as binding, phosphorylation, acetylation, etc. Then, the influence of entities on processes is modeled as *contingency* relations, and the list of defined contingencies closely adheres to those used in visualization: requirement, stimulation, inhibition, etc. If a kinetic model is desired, pre-defined rule-based templates are used to transform the Rxncon specification into either a rule-based model or reaction network or Boolean model. An example of an Rxncon specification is shown below:

```
A_ppi_B

A_P+_B ; ! A--B
```

Here, ppi and P+ are drawn from the Rxncon list of processes denoting protein-protein interaction and phosphorylation respectively. The symbol ! represents the contingency relation 'requires' and A--B represents the bound AB state.

The Rxncon specification has a few advantages. First, contingency relations can be expressed in the form of English-like statements (process-contingency-entity = subject-verb-object) and easily scaled to tables and databases. Second, the enumerated lists of processes and contingencies closely mirror that of visual standards such SBGN diagrams, which can be automatically generated from the specification [65]. Third, it enables a new visualization called the regulatory graph (Figure 2-15). The regulatory graph is a bipartite graph or Petri net on elemental states and processes, with edges showing effects of processes and contingency relations. It is a compact and useful abstraction for conveying signal flow, and there is a 1:1 map between the Rxncon specification and the regulatory graph.

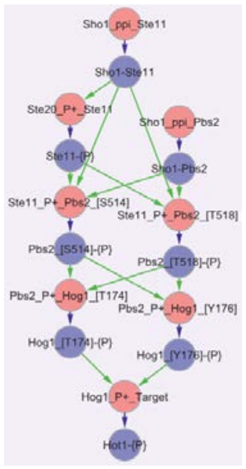


Figure 2-15. Regulatory graph from the Rxncon specification. Diagram is from Tiger2012. Red nodes represent processes such as binding (ppi) and phosphorylation (P+), whereas blue nodes represent states created or modified by processes and which influence processes in the form of contingency relations.

However, the convenience comes at a significant loss, especially in comparison to other rule-based modeling frameworks such as BioNetGen, Kappa and Simmune. First, biochemical signaling complexes are known to be of arbitrary and heterogenous composition [18]. This motivates other rule-based languages to use graphs to represent complexes, such as shown in Figure 2-16. However, Rxncon uses simple Boolean relations and does not have a high enough resolution to model arbitrary complexes [65]. Second, the rule-based templates and the rule-generation process encode strong assumptions about how sites interact, which are not applicable to all biochemical systems. Rxncon applies the rule-generation logic to all systems uniformly, which will lead to incorrect mechanistic assumptions in many cases. Third, the enumerated list of processes and contingencies unnecessarily restrict the model specification. Because of the expanding nature of biochemical knowledge, it may not be possible to ever have a complete list of all relevant processes and contingencies. For example, Rxncon does not include conformational change as a possible reaction type, and even if it did, it would have to enumerate what types of conformational changes are commonly encountered. On the other hand, BioNetGen implements the abstract notion of internal states on components and internal state changes in rules, which allows modeling of any process involving molecular attributes, such as phosphorylation (~0 to ~P), acetylation (~0 to ~Ac), conformational change (say, ~open to ~closed), or any other uncommon process. In general, any approach that builds mechanistic kinetic models from human-readable maps, such as Path2Models [72] or Rxncon [65], is bound to be approximate or incorrect because of these reasons. The recommended strategy is to separate the formal and visual specification, so that the model itself is exact and uses abstract entities, but visualization and human understanding are supplemented with annotations drawn from an enumerated list or database. Annotation frameworks for rule-based models are already being developed for this purpose [73].

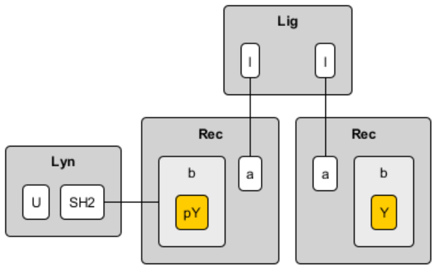


Figure 2-16. Example of a real complex that cannot be represented in Rxncon. Shown is a receptor dimer formed by crosslinking due to a bivalent ligand. One receptor is phosphorylated and bound to Lyn. The other receptor is not bound to anything else and is unphosphorylated. In Rxncon, there is no Boolean combination of the elemental states Lig--Rec, Rec--Lyn, and Rec-{P} that can replicate this complex.

2.5 GRAPH ABSTRACTIONS FOR RULE-BASED VISUALIZATIONS

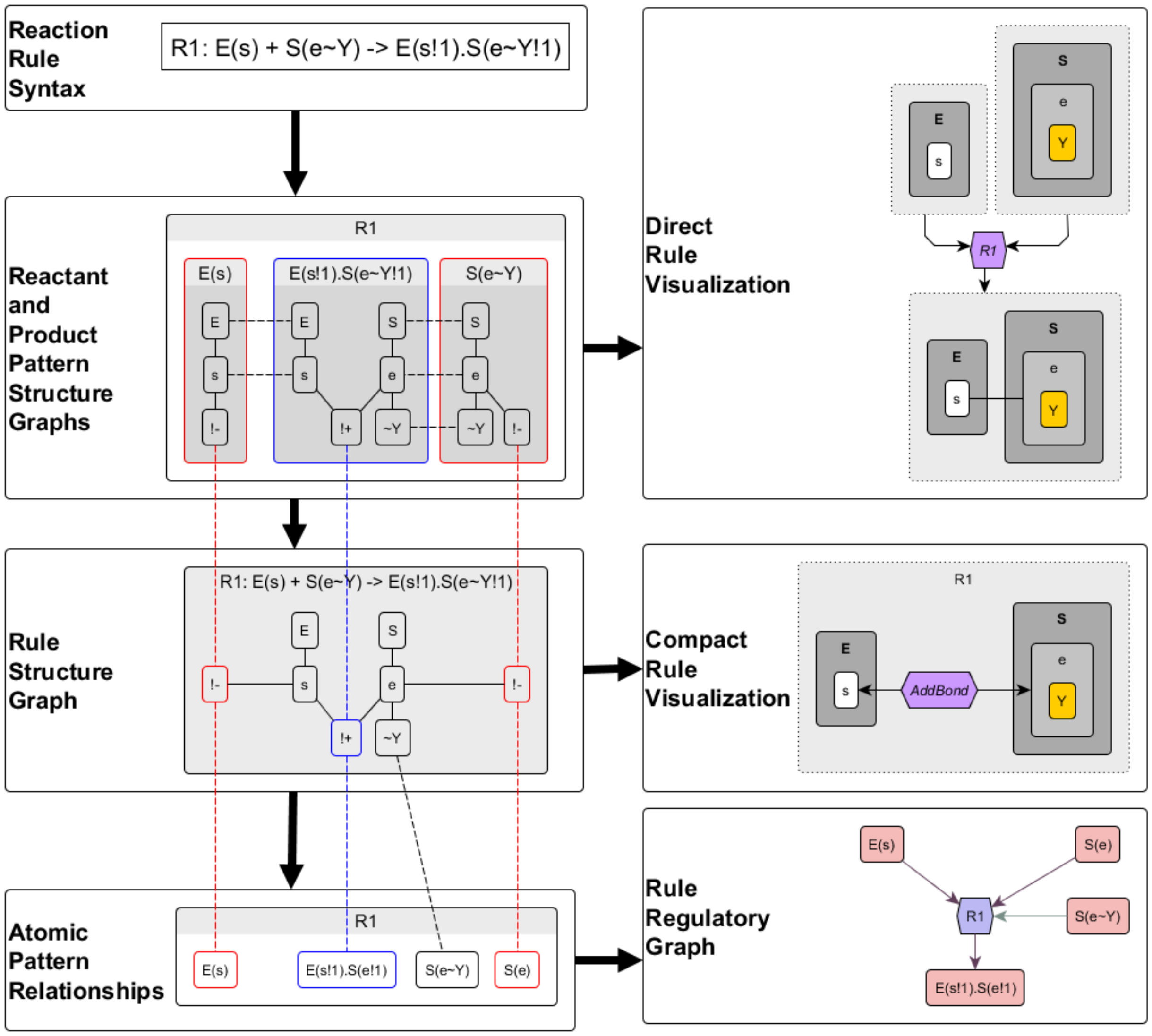
In this section, I define new graph-based abstractions for rule-based models that address two primary visual objectives for rule-based models: (i) visualizing individual rules in a compact manner to convey the modeled mechanisms, and (ii) visualizing a set of rules as a network of regulatory interactions. To the second objective, an addendum can be added: the network visualization must be compact, scalable and useful, even for large numbers of rules. The first and second objectives address visualization of model content as defined in the model specification, but the addendum addresses the problem of being able to generate useful diagrams that showcase model intent, even when the model content is too large or too complex.

This section is intended to hold the formalisms underlying the generation of visualizations from a rule-based model. These visualizations address the objectives defined above: visualizing individual rules, inferring a network of interactions, and tuning the complexity of the generated network. As such, it is dense with abstract definitions and terminology, so the casual reader is advised to skip to **Section 2.6**, where the methods described here have been applied to specific models and systems. In this synopsis, I present a brief outline, summarized in Figure 2-17.

In BioNetGen, the pattern (a graph representing sites on a complex) and the reaction rule (a graph transformation representing a kinetic process) are the fundamental mathematical abstractions used. Hogg et al. (Supplement) [34] provide standard definitions which I will adhere to. In **Section 2.5.1 Preliminary Definitions**, I define node-labeled edge-labeled graphs and elemental graph operations. In **Section 2.5.2 Structure Graphs**, I define pattern structure graphs which are used to generate site graph and rule Petri net visualizations, and then define rule structure graphs which are used to generate compact rule visualizations. These visualizations address the first visual objective. In **Section 2.5.3 Atomic Patterns**, I define objects called atomic patterns that capture the notion of a class of sites in the model that can be acted upon, such as a free binding site, or a bond, or a phosphorylated state. In **Section 2.5.4 Regulatory Graphs**, I provide a systematic approach to generate a regulatory network in the form of a bipartite edge-labeled graph from a reaction rule, and show how to aggregate such graphs from individual rules into a flow-based representation of the whole model. This addresses the second objective. Figure 2-17 also shows an outline on how these two objectives are achieved. Prior to this work, it was not possible to generate global visualizations of rule-based models by aggregating individual rules. The regulatory graph formalism enables the coarse-graining of the rule-based specification into a simpler aggregable representation.

I also define automated coarse-graining procedures (pruning, grouping, collapsing) that use minimal external input to transform the generated model regulatory graphs into compact pathway diagrams. This addresses the addendum to the second objective. As I will demonstrate in Section 2.5.7, these coarse-graining approaches can result in a dramatic reduction of complexity, generating useful visual representations even for very large rule-based models.

In **Section 2.5.5 Complexity Analysis**, I discuss scalability concerns of the methods used, and **in Section 2.5.6 Comparisons to Other Approaches**, I discuss how compact rule visualization and regulatory graphs compare with existing methods, such as those described in **Sections 2.3 and 2.4.**



Reaction Rule Syntax
R1: E(s) + S(e~Y) -> E(s!1).S(e~Y!1)
Reactant and Product Pattern Structure Graphs
R1
E(s)
E(s!1).S(e~Y!1)
S(e~Y)
Direct Rule Visualization
R1
AddBond
Rule Structure Graph
R1: E(s) + S(e~Y) -> E(s!1).S(e~Y!1)
Compact Rule Visualization
Atomic Pattern Relationships
E(s)
E(s!1).S(e!1)
S(e~Y)
S(e)
Rule Regulatory Graph

Figure 2-17 Outline of the rule visualization methods described in this chapter. Patterns from the reaction rule are represented as pattern structure graphs which are merged into the rule structure graph. These are used to generate the direct and compact rule visualizations. From the rule structure graph, atomic patterns and their relationship to the rule are inferred. These are then visualized as the rule regulatory graph.

2.5.1 Preliminary Definitions

Here I define node-labeled edge-labeled graphs, consistency properties of node and edge labels, and simple graph operations. They form the basis for specific types of these graphs that will be defined in later parts of the chapter, and which will be used to represent objects in rule-based models and visualizations.

2.5.1.1 Label prototypes and applications

Definition 2.5-1

A **prototype**, say *f*, is an arbitrarily defined type of labeling function.

Definition 2.5-2

An **application** $f_X \coloneqq X \rightarrow A^*$ is an instance of a prototype *f* applied to a particular domain X. Here A^* represents the set of words from alphanumeric characters plus the symbols $\{(,), \sim, !, +, -, \emptyset\}$. The relationship between application, prototype and domain is denoted $f \longmapsto f_X$. The prototype is multi-valued in the sense that more than one application can be defined over the same domain derived from the same prototype, i.e. $f \longmapsto f_{X,1}, f_{X,2} \dots$

Definition 2.5-3

A **prototype set** is a set of prototypes. This is denoted $F \coloneqq \{f^i\}$, where *i* indexes the set.

Definition 2.5-4

An **application set** is a set of applications derived from a prototype set applied to the same domain.

$$F \coloneqq \{f^i\}$$

$$F_X \coloneqq \{f_X^i \coloneqq X \rightarrow A^* \mid f^i \longmapsto f_X^i \forall f^i \in F\}$$

The relationship between the prototype set, application set and domain is denoted $F \longmapsto F_X$.

Definition 2.5-5

Two applications are **consistent** (denoted ≡) if they are derived from the same prototype and map identical labels to identical elements.

$$f_{X,1} \equiv f_{X,2} \iff$$

$$f \longmapsto f_{X,1}, f_{X,2}\,,$$

$$f_{X,1}(x) = f_{X,2}(x)\ \forall x \in X$$

Applications defined on two domains are consistent if they are consistent over the domain of intersection.

$$f_X \equiv f_Y \iff$$

$$f \longmapsto f_X, f_Y\,,$$

$$f_X(x) = f_Y(x)\ \forall x \in X \cap Y$$

Definition 2.5-6

Two application sets are **consistent** if they are derived from the same prototype set and each pair of applications derived from each prototype is consistent.

$$F_{X,1} \equiv F_{X,2} \iff$$

$$F \longmapsto F_{X,1}, F_{X,1}$$

$$f_{X,1}^i \equiv f_{X,2}^i\ \forall f^i \in F$$

Corollary 2.5-7

An application (or application set) defined over a domain implies the existence of a consistent application defined over any subset of the domain.

Proof: Given $f \longmapsto f_X, Y \subset X$, let $f_Y \coloneqq f_X(x) \forall x \in Y$, then $f \longmapsto f_X$and $f_Y \equiv f_X$ by construction. By extending to every $f_X^i \in F_X$ and by Definition 2.5-6, exists $F_Y \equiv F_X$.

Corollary 2.5-8

Two consistent applications (or application sets) defined over two domains implies the existence of a consistent application over the union of the two domains.

Proof: Given $X, Y, f \longmapsto f_X, f \longmapsto f_Y, f_X \equiv f_Y$,

by Corollary 2.5-7, exists $f_{X\cap Y} \equiv f_X \equiv f_Y, f_{X-Y} \equiv f_X, f_{Y-X} \equiv f_Y$,

now let $Z = X \cup Y = (X - Y) \cup (X \cap Y) \cup (Y - X)$, and $f_Z(x) = \begin{cases} f_{X-Y}(x) & x \in X - Y \\ f_{X\cap Y}(x) & x \in X \cap Y \\ f_{Y-X}(x) & x \in Y - X \end{cases}$

By construction $f_Z \equiv f_X \equiv f_Y$.

By extending to every $f_X^i \in F_X$ and by Definition 2.5-6, exists $F_Z \equiv F_X \equiv F_Y$.

Definition 2.5-9

Given a set X and a function g, $g \odot X$ is defined as the image of g in X for elements where such an image exists, i.e. $g \odot X = \{g(x), \forall x \in X | \exists g(x)\}$

2.5.1.2 Graphs

Definition 2.5-10

A **graph type** is defined by the tuple (Λ, Σ), where Λ is an arbitrarily defined set of node label prototypes and Σ is an arbitrarily defined set of edge label prototypes.

Definition 2.5-11

A **graph instance**, say G, of graph type (Λ, Σ), is defined by a set of nodes $V \subset A^*$, a set of edges $E \subset V \times V$, a particular node application $\Lambda_V \coloneqq V \to \{A^*\}, \Lambda \longmapsto \Lambda_V$ and a particular edge application $\Sigma_E \coloneqq E \to \{A^*\}, \Sigma \longmapsto \Sigma_E$, i.e. $G \coloneqq (V, E, \Lambda_V, \Sigma_E)$.

Henceforth, "given G" will be taken to mean "given $G \coloneqq (V, E, \Lambda_V, \Sigma_E)$", with graph type assumed to be obvious. Similarly, "given G_i" will be taken to mean "given $G_i \coloneqq \left(V_i, E_i, \Lambda_{V_i}, \Sigma_{E_i}\right)$" for any index i.

Definition 2.5-12

Two graphs are **consistent** (denoted $\equiv$) if their node and edge applications are consistent, i.e. given G_1, G_2, if $\Lambda_{V_1} \equiv \Lambda_{V_2}, \Sigma_{E_1} \equiv \Sigma_{E_2}$, then $G_1 \equiv G_2$.

Definition 2.5-13

A graph is a **subgraph** of another graph (denoted $\sqsubset$), if the two graphs are consistent and the nodes and edges of the first graph are subsets of the nodes and edges of the second graph, i.e. given G_1, G_2, if $V_1 \subset V_2, E_1 \subset E_2, G_1 \equiv G_2$ then $G_1 \sqsubset G_2$.

2.5.1.3 Common Graph Operations

Definition 2.5-14

Given two consistent graphs, a **trivial merge** (denoted $\sqcup$) of two graphs is the graph composed from the union of the node and edge sets, i.e. given $G_1, G_2, G_1 \equiv G_2$,

then $G_1 \sqcup G_2 \coloneqq (V_1 \cup V_2, E_1 \cup E_2, \Lambda_{V_1 \cup V_2}, \Sigma_{E_1 \cup E_2})$

Definition 2.5-15

A **node remap** operation $\mathcal{VM}(v_0 \to v_1)$ copies edges incident on a node v_0 to another node v_1, i.e. given G and nodes $v_0, v_1 \in V$,

let $E_0 = \{(v, v') \in E \mid v_0 \in \{v, v'\}\}$ and $g(v) = \begin{cases} v_1 & v = v_0 \\ v & v \neq v_0 \end{cases}$ and $h(v, v') = (g(v), g(v'))$,

let $E_1 = \{h(v, v'), \forall\ (v, v') \in E_0\ \}$ and

and let $\Sigma \longmapsto \Sigma_{E_1} s.t. \forall f \in \Sigma, f_{E_1}(h(v, v')) = f_{E_0}((v, v')) \forall (v, v') \in E_0$

then $\mathcal{VM}(v_0 \to v_1) \circ G \coloneqq (V, E \cup E_1, \Lambda_V, \Sigma_{E \cup E_1})$

Definition 2.5-16

A **node remap sequence** $Map(Y)$ performs a sequence of remap operations on a graph G using some set of ordered pairs $Y \coloneqq \{(v, v') \mid v, v' \in V\}$.

$$Map(Y) = \left\{ \prod_{(v,v')\in Y} \mathcal{VM}(v \rightarrow v') \right\} \circ G$$

In this work, we will ensure by construction that the order of remaps does not matter, which can be achieved by having the sources and targets for the remap be disjoint sets,

i.e. $Src \cap Tgt = \{\}$ where $v \in Src, v' \in Tgt, \forall (v, v') \in Y$

Definition 2.5-17

A **node delete** operation $\mathcal{VD}(v_0)$ removes a node $v_0$and any edges incident to v_0,

i.e. given $G = (V, E, \Lambda_V, \Sigma_V)$ and node $v_0 \in V$,

let $E_0 = \{(v, v') \in E \mid v_0 \in \{v, v'\}\}$ and $V_0 = \{v_0\}$ the singleton

then by construction $\mathcal{VD}(v_0 \rightarrow v_1) \circ G \coloneqq (V - V_0, E - E_0, \Lambda_{V-V_0}, \Sigma_{E-E_0})$

Definition 2.5-18

A **node delete sequence** $Del(Y)$ performs a series of node delete operations on a graph $G = (V, E, \dots)$ using some subset of nodes $Y \subset V$, i.e.

$$Del(Y) = \left\{ \prod_{v\in Y} \mathcal{VD}(v_0) \right\} \circ G$$

Definition 2.5-19

A **filter** operation $\mathcal{F}(f, y) \circ X$ creates a subset of the domain X that maps to value y under the application of the prototype f, i.e.

$$\mathcal{F}(f, y) \circ X = \{x \in X \mid f_X(x) = y\}$$

Definition 2.5-20

A **filter sequence** $Fil(Y) \circ X$ creates a subset of the domain X by sequentially filtering using each pair of prototypes and values in $Y \coloneqq \{(f, y) \mid f \in F, y \in A^*\}$.

$$Fil(Y) \circ X = \left\{ \prod_{(f,y) \in Y} \mathcal{F}(f,y) \right\} \circ X$$

2.5.2 Structure Graphs

Here, I define graphs called structure graphs that are useful to represent structural objects such as molecules, components, internal states and bonds, and relationships between those objects. The pattern (described in Section 1.3.2) is represented as the pattern structure graph (2.5.2.1), and this is converted into a visual object called the pattern site graph (2.5.2.2). The reaction rule (described in Section 1.3.3) is composed from pattern structure graphs (2.5.2.3) and has a classical visualization as a Petri net (2.5.2.4). The information in the rule is condensed into a single rule structure graph (2.5.2.5), which in turn is converted into a visual object called the compact rule visualization (2.5.2.6). The compact rule visualization is a new contribution to the visualization of rule-based models.

2.5.2.1 Pattern Structure Graph

In BioNetGen, patterns are used to represent complexes and subgraphs of complexes. The pattern structure graph has one node each for every molecule, component, internal state and binding state in the pattern. Pattern structure graphs are based on hierarchical graphs defined in Lemons et al. [74], with the difference being that hierarchical graphs use an edge to represent a bond rather than a separate node. The pattern structure graph is also equivalent to the pattern defined in Hogg et al. [34], where it is defined as a generic data structure rather than a strict graph.

Definition 2.5-21

The **pattern structure graph** is a graph with unlabeled edges and labeled nodes defined by the node labeling prototype $\Lambda = \{T, \Omega\}$,

where T is a node-type function mapping to the codomain $\{\text{mol}, \text{comp}, \text{is}, \text{bs}\}$, where mol=molecule, comp=component, is=internal state and bs=bond state, and Ω is a name function mapping to the set of all words A^*. Additionally, from Hogg REF, restrictions are placed on adjacency relationships, which are defined in the table below:

Table 2-1. Adjacency restrictions for different node types on the structure graph.

Node type	Adjacent node type			
	Molecule	Component	Internal State	Bond State
Molecule	0	Defined by molecule type	0	0
Component	1	0	1	1
Internal State	0	1	0	0
Bond State	0	1 or 2	0	0

Additionally, bond state labels are restricted to {!+,!-,!?} respectively, denoting bond, unbound state and unspecified state respectively. A bond state labeled !+ is called a fully specified bond if it has two adjacent component nodes, or a bond wildcard if it has one adjacent component node.

Let P^*denote the set of all pattern structure graphs present in a model.

2.5.2.2 Pattern Site Graph

To convert the pattern structure graph into a pattern site graph visualization: (i) nest each component node within the adjacent molecule node, (ii) nest each internal state node within the adjacent component node, (iii) remove unbound state nodes, (iv) if a bond has two parents, replace bond node by an edge between the parent components.

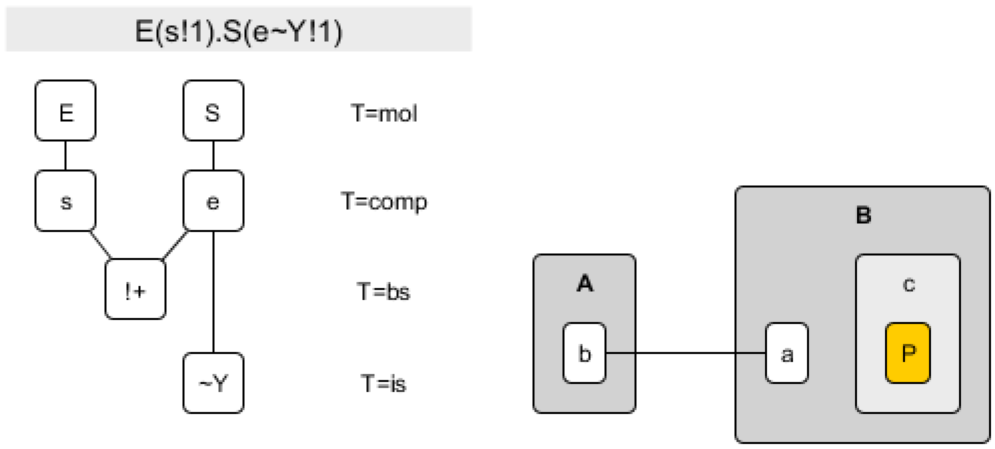


Figure 2-18. Pattern structure graph and site graph. The first panel shows a pattern structure graph (defined in **Definition 2.5-21**), constructed from the pattern E(s!1).S(e~Y!1). Each node has a node-type (T) defined to be molecule (mol), component (comp), internal state (is) or bond state (bs). Each node also has a name **Ω** which is shown as the node label. The bond state has two adjacent components and name !+, indicating it is a fully specified bond. The second panel shows a pattern site graph generated from the pattern structure graph (see Section 2.5.2.2).

2.5.2.3 Reaction Rule

Here, I provide a definition for the reaction rule that is compatible with previous definitions in this thesis as well as Hogg et al. [34].

Definition 2.5-22

A reaction rule $r \coloneqq (rname, Re, Pr)$, where $rname$ is a unique name assigned to the rule and $Re, Pr \subset P^*$are sets of reactant and product patterns respectively.

Definition 2.5-23

Let R_A^* be the set of all rule names, and R^*be the set of all rules r.

2.5.2.4 Rule Petri Net

To obtain a rule Petri net (as in Section 2.4.3) from the rule as defined in Definition 2.5-23, (i) draw a node to represent the rule and label with $rname$, (ii) draw a node to represent each pattern $p \in Re \cup Pr$, (iii) within each node, embed the site graph of the respective pattern, (iv) draw a directed edge from each reactant pattern to the rule node, and (v) draw a directed edge from the rule node to each product pattern. As mentioned previously (Section 2.4.3), rule Petri nets cannot be aggregated from local visualizations of rules into a global one because of partial overlaps between rules.

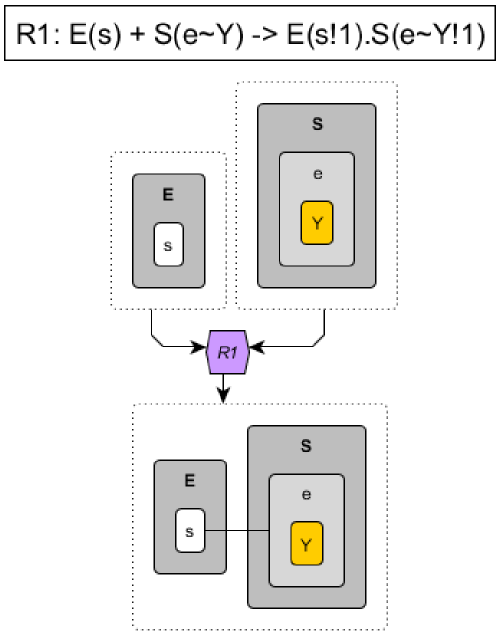


Figure 2-19. Petri net of Rule R1 showing binding of enzyme and unphosphorylated substrate. The reactant patterns and product patterns are visualized separately as site graphs (Section 2.5.2.2) and embedded in the entity nodes of the Petri net. Note that other than the bond that is formed, each structure is repeated twice, once on the reactant side and once on the product side.

2.5.2.5 Rule Structure Graph

The rule Petri net shows each reactant and product separately, and structures that are not modified by the rule are represented twice, once on the reactant side and once on the product side. The rule structure graph is a systematic merging of reactant and product pattern structure graphs of the rule that eliminates redundancies.

Definition 2.5-24

The **rule structure graph** is a node-labeled, edge-unlabeled graph defined by the tuple of node prototypes $\Lambda = \{T, \Omega, side\}$, where T, Ω are as defined in Definition 2.5-21 and $side$ maps each node to one of $\{l, r, lr\}$ indicating left, right or both respectively.

To build a rule structure graph from a reaction rule:

1. Given a rule $r \coloneqq (rname, Re, Pr)$ where $Re, Pr \subset P^*$, first we trivially merge the reactant and product sides separately to give 'left' and 'right' structure graphs, $G^l$and G^r respectively.

$$G^l \coloneqq \bigcup_{G_i \in Re} G_i = \left(V^l, E^l, \Lambda_{V^l}\right)$$

$$G^l \coloneqq \bigcup_{G_j \in Pr} G_j = (V^r, E^r, \Lambda_{V^r})$$

2. BioNetGen computes a partial one-to-one map $\phi: V^l \nrightarrow V^r$ between reactant and product patterns that identifies the unmodified structures. Let dom and img be the subsets of V^l and V^r respectively in which this map is one-to-one onto.

$$img = \phi \ \odot V^l$$

$$dom = \phi^{-1} \odot img$$

The structures mapped by ϕ are present on both sides of the rule. We refer to the nodes in dom as 'original' and the nodes in img as duplicate respectively.

3. Now, we merge the left and right sides of the rule and remove the duplicate nodes.

$$W = \{(\phi(v), v), \forall v \in dom\}$$

$$G = Del(img) \circ Map(W) \circ (G^l \sqcup G^r)$$

4. We compute $side$ depending on which side of the rule a node originates from:

$$side_V(v) = \begin{cases} l & v \in V^l - dom \\ r & v \in V^r - img \\ lr & v \in dom \end{cases}$$

5. The rule structure graph is given by G as computed in step 3, with $side_V$ as computed in step 4 being included in the application set Λ_V.

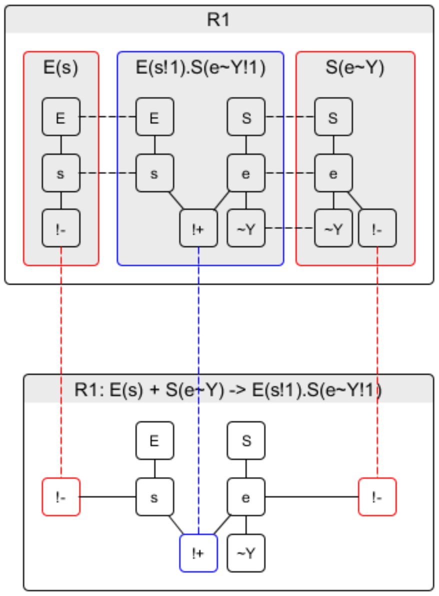


Figure 2-20. Synthesizing the rule structure graph from a reaction rule. At the top, we show rule R1 which has two reactant patterns (red) and one product pattern (blue), and they are shown as pattern structure graphs according to **Definition 2.5-21**. BioNetGen computes a partial map between reactants and products that identifies the unmodified structures (dotted black line). By merging the graphs and removing redundant nodes, we generate the rule structure graph as seen at the bottom. During the process, we keep track of the origins of the nodes, whether they are from the product side (blue), reactant side (red) or both (black).

Definition 2.5-25

Let $r' = (rname, G)$ be the new definition of a rule, where G is the rule structure graph derived from $r = (rname, Re, Pr)$ as shown above. Let R^*_{sg} be the set of all rules r' as defined here.

2.5.2.6 Compact Rule Visualization

Using the rule structure graph, we can generate a compact visualization of the rule by identifying the modified structures and adding graph operation nodes. These are special nodes whose labels indicate what kind of graph transformation is being carried out by the rule.

Given a rule structure graph as in Definition 2.5-24, (i) for each molecule node with side l, draw a graph operation node labeled **DeleteMol** and draw a directed edge from the molecule node to the graph operation node, (ii) for each molecule node with side r, draw a graph operation node labeled **AddMol** and draw a directed edge to the molecule node from the graph operation node, (iii) for each component with side lr, whose adjacent internal states have side l or r, draw a graph operation node labeled **ChangeState**, draw a directed edge from the adjacent internal state with side l to the graph operation node, and draw a directed edge to the adjacent internal state with side r from the graph operation node, (iv) for each bond with side l, replace bond node with graph operation node labeled **DeleteBond**, add edge direction away from graph operation node on incident edges, (v) for each bond with side r, replace bond node with graph operation node labeled **AddBond**, add edge direction away from graph operation node on incident edges, (vi) render remaining nodes according to site graph conventions in Section 2.5.2.2, (vii) nest ChangeState operation nodes within adjacent component nodes.

The compact rule visualization is a compact and useful local visualization of rules that is explicit about which parts of the participating complexes are modified and which parts are context for the interaction. However, parts modified in one rule can remain unmodified in another rule, so compact rule visualizations cannot be strictly aggregated into a global visualization.

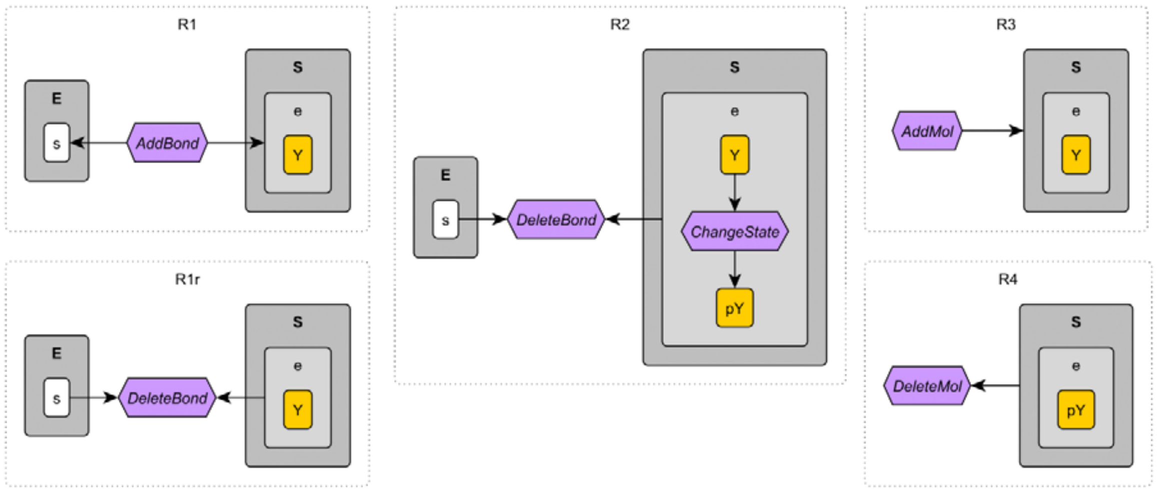


Figure 2-21. Compact rule visualization. Instead of showing reactants and products separately, each structure present in the rule is shown only once. The changes implemented by the rule are shown as graph operation nodes of the following types: AddBond, DeleteBond, ChangeState, AddMol, DeleteMol. Shown here are five rules: R1, R1r, R2, R3 and R4.

2.5.3 Atomic Patterns

Patterns represent specific combinations of “actionable sites” that are relevant to a particular reaction rule. For example, a pattern such as EGFR(Y1068~P!1).Grb2(SH2!1) has the following “actionable sites”, a bond EGFR(Y1068!1).Grb2(SH2!1) that could be formed or broken between the binding sites EGFR(Y1068~P) and Grb2(SH2), and a phosphorylated state EGFR(Y1068~P) that could be dephosphorylated. However, the common understanding of signal flow is not through specific combinations, but through individual types of sites. For example, a statement such as “the phosphorylated state of Y1068 affects binding of Grb2” requires treating the classes of sites

EGFR(Y1068~P) and EGFR(Y1068!1).Grb2(SH2!1) separately with a causal influence from the first to the second. Here, I call actionable sites on patterns as **atomic patterns**, and provide a formalism where such sites can be identified on any pattern. Atomic patterns are analogous to elemental states used by Tiger et al. [65].

2.5.3.1 Definitions and Interpretations

Definition 2.5-26

A node on a pattern structure graph is **well-defined** if (i) it is a molecule node, (ii) it is a component and its adjacent molecule is well-defined, (iii) it is a bond state node that is bound (!+) or unbound (!-) and its adjacent component(s) are well defined, (iv) it is an internal state other than the default state and its adjacent component is well-defined.

Definition 2.5-27

An **atomic pattern** is a connected pattern structure graph in which the graph has at most one well-defined bond state or internal state node, but not both at the same time, and there is at least one bond state or internal state per component.

Practically, atomic patterns take five different forms which can be represented using BioNetGen syntax: (i) **molecule** such as A, (ii) **free binding site** such as A(b), (iii) **bond** such as A(b!1).B(b!1), (iv) **bond wildcard** such as A(b!+), and (v) **internal state** such as A(b~0!?). For internal state atomic patterns, we will use shorthand A(b~0) with the understanding that as long as it is treated as an internal state atomic pattern, only its internal state is relevant. The syntax form allows atomic patterns to be represented as compact alphanumeric labels.

Definition 2.5-28

The **atomic pattern equivalence class** is defined by an atomic pattern structure graph and represents the class of isomorphic subgraphs on all patterns. Henceforth, all references to atomic patterns will be to the corresponding equivalence classes.

Let P_{ap}^* be the set of all atomic patterns encountered in a model. Since atomic patterns have syntax equivalents as mentioned in Definition 2.5-27, $P_{ap}^* \subset A^*$.

By defining them as equivalence classes, atomic patterns are functionally equivalent to elemental states in Rxncon. Also, the number of atomic patterns in a model is bounded and much smaller than the number of patterns. This is because an atomic pattern describes an equivalence class for a single site or feature, whereas a pattern describes an arrangement of sites or features in a complex, which is subject to combinatorial complexity [18].

2.5.3.2 Determining Atomic Patterns

Atomic patterns are not explicitly defined by the user, but instead, we provide a procedure by which they can be determined from pattern structure graphs automatically.

Definition 2.5-29

The **atomic pattern map** $\theta: V \to A^*$ is a map from nodes of a pattern structure graph $G \coloneqq (V, E, \ldots)$ to a particular atomic pattern equivalence class, or none by default ($\emptyset$).

To build this map, subgraphs isomorphic to atomic patterns are determined heuristically by examining each node and its neighboring nodes (see Figure 2-22).

2.5.4 Regulatory Graphs

Regulatory graphs defined in this section are visually similar to the flow-based regulatory graphs shown in Section 2.4.8 as part of the Rxncon framework [65]. The inference of a regulatory graph from a reaction rule (2.5.4.1) is a novel contribution of this work. Prior to this work, it was not possible to generate a global visualization of signal flow, but in this section we show how this is possible by aggregating regulatory graphs of rules (2.5.4.3). We also show how to reduce the complexity of the generated diagram by removing redundant nodes (2.5.4.4), by using a combination of automated and user-seeded approaches to group nodes (2.5.4.5). Atomic patterns on the graph are grouped manually, and this information is used by an automated algorithm to group reaction rules. Following this, the complexity of the graph can be greatly reduced by collapsing groups of nodes to single representative nodes (2.5.4.6)

Definition 2.5-30

The **regulatory graph type** is defined by the tuple (Λ, Σ), where $\Lambda \coloneqq (T, \Omega, Gr)$ and $\Sigma \coloneqq \{R, P, C\}$. Here T is a node type function prototype mapping to the codomain $\{ap, r\}$, denoting atomic pattern equivalence class and reaction rule respectively, Ω is a node subtype function prototype mapping to the codomain $\{mol, fbs, wc, b, is, g, \emptyset\}$, where $\{mol, fbs, wc, b, is\}$ refer to the five types of atomic patterns (molecule, free binding site, bond wildcard, bond, internal state respectively), g refers to 'group' and $\emptyset$ is a default value. Gr maps each node to some group name in A^* or to $\emptyset$ by default. $\{R, P, C\}$ are binary function protoypes (i.e. with codomain $\{0,1\}$) used to identify whether an edge relation can be a reactant, a product, or a context relation respectively. A single edge can be assigned more than one relation.

2.5.4.1 Rule Regulatory Graph

1. Starting from a rule structure graph $r = \{rname, G^{sg}\}$ as in Definition 2.5-25, where $G^{sg} \coloneqq (V^{sg}, E^{sg}, \Lambda_{V^{sg}})$, where $\Lambda_{V^{sg}} = \{T_{V^{sg}}, \Omega_{V^{sg}}, side_{V^{sg}}\}$, we first compute an atomic pattern map $\theta_{V^{sg}}: V^{sg} \rightarrow P_{ap}^* \cup \{\emptyset\}$ as defined in Definition 2.5-29.

2. We identify which atomic pattern subgraphs are consumed, produced or left unchanged by querying the $side$ property. We exclude molecule atomic patterns that are not consumed or produced to minimize the complexity of the generated graph.

$$V^l = \theta_{V^{sg}} \odot \mathcal{F}(side, l) \circ V^{sg}$$

$$V^r = \theta_{V^{sg}} \odot \mathcal{F}(side, r) \circ V^{sg}$$

$$V^{lr} = \theta_{V^{sg}} \odot (\mathcal{F}(side, lr) \circ V^{sg} - \mathcal{F}(T, mol) \circ V^{sg})$$

3. The node set for the regulatory graph is $V = V^l \cup V^r \cup V^{lr} \cup \{rname\}$. Here, V^l, V^r, V^{lr} are atomic pattern nodes, and $rname$ is the rule node.

4. The edge set maps the rule node to each atomic pattern node, i.e. $E = \{rule\} \times (V^l \cup V^r \cup V^{lr})$

5. The node applications are assigned depending on whether the node is an atomic pattern or the rule name.

$$T_V = \begin{cases} ap & v \in V^l \cup V^r \cup V^{lr} \\ r & v = rname \end{cases}$$

The subtype function Ω_V is assigned from $\{mol, fbs, is, b, wc\}$ depending on the type of atomic pattern or to $\emptyset$ if $rname$.

6. Let $\mathbb{1}_x(v) = \begin{cases} 1 & v \in V^x \\ 0 & v \notin V^x \end{cases}$, where v is a node, and $x \in \{l, r, lr\}$. The edge applications derived from prototypes R, P, C are populated by querying the membership of nodes within each of these sets. Note that this allows the same edge to be assigned 1 for multiple applications.

$$\forall v \in V,$$

$$R_E\big((rname, v)\big) = \mathbb{1}_l(v),$$

$$P_E\big((rname, v)\big) = \mathbb{1}_r(v),$$

$$C_E\big((rname, v)\big) = \mathbb{1}_{lr}(v)$$

7. The remaining node application Gr_V is populated with default values, i.e. $Gr_V \coloneqq V \to \{\emptyset\}$

8. The graph $G \coloneqq (V, E, \Lambda_V, \Sigma_E)$, where $\Lambda_V = \{T_V, \Omega_V, Gr_V\}, \Sigma_E = \{R_E, P_E, C_E\}$ is the **rule regulatory graph**.

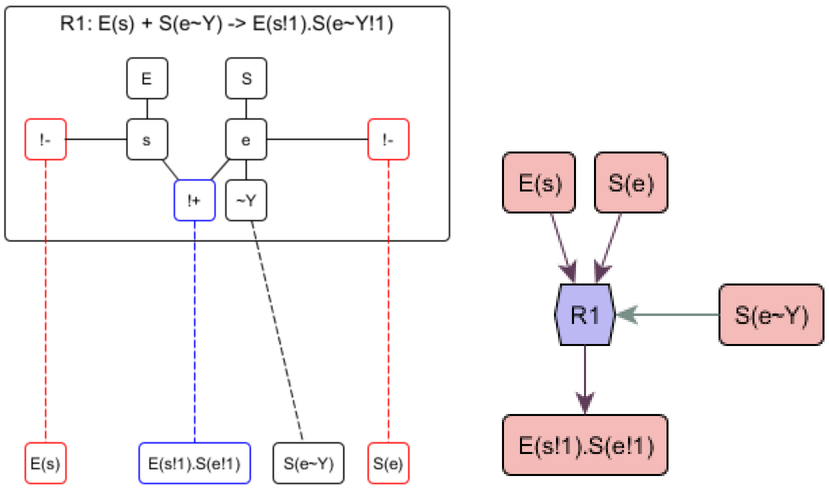


Figure 2-22. Atomic Patterns and the Regulatory Graph. The first panel shows the atomic pattern map built from a rule structure graph by querying neighborhoods of nodes and identify subgraphs compatible with **Definition 2.5-27**. During the process, we also identify the relationships of each atomic pattern to the rule: reactant (red), product (blue) and/or context (black). In the second panel, these relationships are visualized as the regulatory graph, with one node representing the rule, other nodes representing the atomic patterns, dark edges indicating reactant and product relationships and light edges indicating context relationships.

2.5.4.2 Resolving Wildcards

In BioNetGen, a wildcard bond such as A(b!+) can match one or more fully specified bonds such as A(b!1).B(a!1) and A(b!1).C(a!1). For the regulatory graph to be complete, these relationships need to be represented on the graph. BioNetGen can compute relationships between wildcard bonds and fully specified bonds by examining their syntax forms.

$$E_{wc} = \{(v_{wc}, v_b) | v_b, v_{wc} \in V, \Omega_V(v_b) = b, \Omega_V(v_{wc}) = wc\}$$

$$V_{wc} = \{v \; \forall v \in V, \Omega_V(v) = wc\}$$

The wildcards are then "resolved", by mapping context edges on wildcards onto matching bond nodes and then deleting the wildcard nodes. The resultant graph is

$$G' = Del(V_{wc}) \circ Map(E_{wc}) \circ G$$

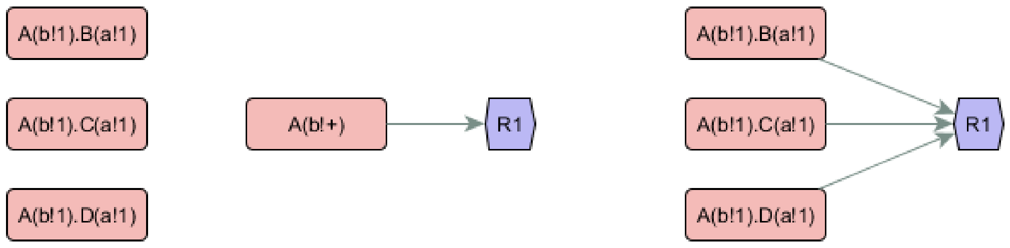


Figure 2-23. Resolving Wildcards. The first panel shows a graph with a wildcard node A(b!+) and three matching bond nodes. When resolving wildcards as defined in Section 2.5.4.2, context edges from wildcards are remapped to matching bond nodes and the wildcard nodes are deleted. The second panel shows the resultant graph.

2.5.4.3 Model Regulatory Graph

1. The model regulatory graph can be aggregated from individual rule regulatory graphs using a trivial merge, i.e. given rule set $M = \{r, r \coloneqq (rname_r, G_r)\}$, the merged regulatory graph is

$$G = \bigcup_{r \in M} G_r = (V, E, \Lambda_V, \Sigma_E)$$

2. When the graph is aggregated as above, there may be some wildcard bonds used in some rules, but whose matching bonds were generated in other rules. These are resolved according to the procedure in Section 2.5.4.2. Let V_{wc} be the set of nodes removed and E_{wc} be the set of context edges added. The model regulatory graph is given by

$$G^{model} \coloneqq (V - V_{wc}, E \cup E_{wc}, \Lambda_{V-V_{wc}}, \Sigma_{E \cup E_{wc}}).$$

The model regulatory graph is a flow-based bipartite representation of the model that is derived directly from model content (reaction rules).

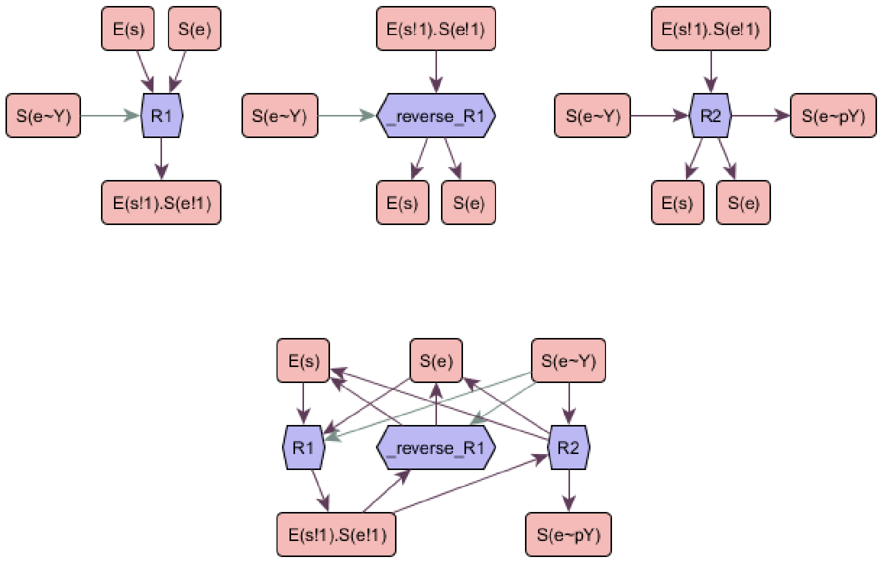


Figure 2-24. Merging Regulatory Graphs. The top panel shows regulatory graphs of three rules, R1 and _reverse_R1 modeling reversible binding of kinase to substrate and R2 modeling simultaneous phosphorylation and dissociation, The bottom panel shows the regulatory graph of the model which is built by a simple merge of the regulatory graphs of individual rules.

2.5.4.4 Pruned Regulatory Graph

A subset of atomic patterns can be considered redundant to the representation of signal flow, for example, the bond pattern A(b!1).B(a!1) makes it obvious that the corresponding free binding sites are A(b) and B(a). Removing redundant nodes, which we call **background nodes**, can simplify the graph and clarify the signal flow. What specific nodes should be considered background can vary with each model although we provide common and useful heuristics.

Given a model regulatory graph $G^{model} = (V, E, \dots)$, an arbitrary node application partitioning the node set $f_V^{bkg}: V \rightarrow \{0,1\}$, where the value 1 implies that the node is included in the background and the value 0 implies that it excluded from the background, the set of background nodes is $V_{bkg} = \left\{v \in V \middle| f_V^{bkg}(v) = 1\right\}$ and the **pruned regulatory graph** without background is

$$G^{simplified} = Del(V_{bkg}) \circ G^{model}$$

There are different heuristics that can be applied to identify the background. Ranked in order of increasingly strong assumptions made about the model: (i) remove atomic patterns in the order in which they are encountered in the rules, (ii) remove all free binding sites and first encountered internal states on components, (iii) remove all reverses of bidirectional rules, (iv) remove all rules that do not produce non-background states. Currently, the implementation enables the first heuristic and allows for user input to override the selections made regarding background inclusions and exclusions. In the future, it might be useful to provide the user a flexible choice with regard to these heuristics in addition to the manual overrides.

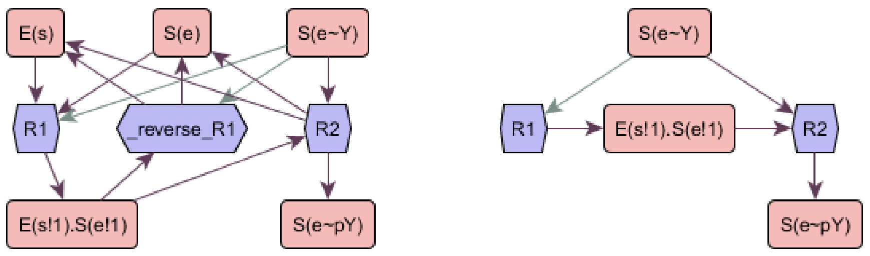


Figure 2-25. Removing redundant nodes (called background) from the model regulatory graph clarifies the signal flow. The first panel shows a regulatory graph of a model with reversible binding of kinase to substrate (R1, _reverse_R1) and phosphorylation (R2). Assigning the free binding sites E(s) and S(e) and the reverse rule R1(reverse) to background and removing them results in the graph in the second panel. On this graph, we see the signal flow that is pertinent to signaling: R1 causes the bound state which enables R2 which causes the phosphorylated state.

2.5.4.5 Grouped Regulatory Graph

A key reason for the predominance of manually drawn diagrams for biochemical models is that the principles of organization that “make sense” for a particular model are often arbitrarily defined and not generalizable to all biochemical models or even all rule-based models. There are some principles that are generally applicable, for example, when multiple rules model the same kinetic process under different local conditions, it could be useful to refer to that group of rules as a collective process. On the other hand, when a molecule has multiple phosphorylation sites, in some models it might be useful to refer to all of them together as a collective phosphorylated state, whereas in other models it might be necessary to consider functional differences between them such as activating or downregulating phosphorylated states. On the regulatory graph, I provide an organizing scheme which seamlessly incorporates both formal analysis and user-driven choices. First, the modeler provides an organization of atomic patterns into groups or classes. Second, an automated algorithm compares the reactant/product relationships of individual rules, incorporates the user-defined grouping of atomic patterns, and sorts rules into functionally similar rule groups.

1. Given a regulatory graph of the model, $G^{model} = (V, E, \dots)$, let $V_{ap}, V_{rule} \subset V$ be the sets of atomic pattern nodes and rule nodes respectively.
2. For each rule node $v \in V_{rule}$, we can compute sets of reactants and products depending on their adjacent atomic pattern nodes.

$$\forall v \in V_{rule},$$

$$\mathcal{R}(v) = \left\{ v' \in V_{ap} | R_E\big((v, v')\big) = 1 \right\}$$

$$\mathcal{P}(v) = \left\{ v' \in V_{ap} | P_E\big((v, v')\big) = 1 \right\}$$

3. Given a seed grouping of atomic patterns, $seed: V_{ap} \to A^*$, where an atomic pattern is assigned to a group with a label or to the default ungrouped state $\emptyset$, we first construct a function that maps each element to itself or to its group name:

$$\psi(v) = \begin{cases} v & seed(v) = \emptyset \\ seed(v) & seed(v) \neq \emptyset \end{cases}$$

When no $seed$ is provided, $\psi(v)$ is simply the identity function.

4. For each rule on the regulatory graph, we compute the transformed reactant-product tuple which will be used to assign membership to rules:

$$RP(v) = (\{\psi(v), \forall v \in \mathcal{R}(v)\}, \{\psi(v), \forall\ v \in \mathcal{P}(v)\})$$

Let $RP^* = \{RP(v) | v \in V_{rule}\}$ be the set of such tuples and let $group: RP^* \to A^*$ define a set of unique labels applied to each unique element in RP^*.

5. Now, we have enough information to construct a grouping on all nodes:

$$Gr_V(v) = \begin{cases} seed(v) & v \in V_{ap} \\ group\big(RP(v)\big) & v \in V_{rule} \end{cases}$$

Gr_V is included with the other node applications on the model regulatory graph in order to make it the **grouped regulatory graph**.

2.5.4.6 Collapsed Regulatory Graph

Given a grouped regulatory graph, a simplifying assumption can be made by rendering individual members of groups indistinguishable from each other. This allows a drastic compression of the graph, and a compact interpretation of the regulatory interactions in terms of groups of atomic patterns and sites, rather than individual ones. Since each group is collapsed to a single node representing the group, we call this the **collapsed regulatory graph.**

1. Given a grouped regulatory graph $G^{model} = (V, E, \ldots)$, we partition the set of nodes V into two subsets: one set of nodes that have groups assigned to them, and the other with no groups assigned to them.

$$V_{grouped} = \{v \in V | Gr_V(v) \neq \emptyset\}, V_{ungrouped} = V - V_{grouped}$$

2. Then, we define new nodes representing groups, i.e.

$$V_{groups} = \big\{Gr_V(v) \big| v \in V_{grouped}\big\}.$$

We also define appropriate node applications:

$$T_{groups}(v) = \{T(v') | v = Gr_V(v'), v \in V_{grouped}\}$$

$$\Omega_{groups}(v) = \{g, v \in V_{groups}\}$$

We then create a temporary graph involving only the group nodes

$$G' = (V_{groups}, \{\quad\}, T_{groups}, \Omega_{groups})$$

3. Then we merge the two graphs, remap grouped nodes to group nodes and delete the grouped nodes to get the **collapsed regulatory graph**.

$$\text{Let } Y = \{(v, Gr_V(v)), \forall v \text{ } in \text{ } V_{grouped}\}$$

$$G^{collapsed} = Del(V_{grouped}) \circ Map(Y) \circ (G^{model} \sqcup G')$$

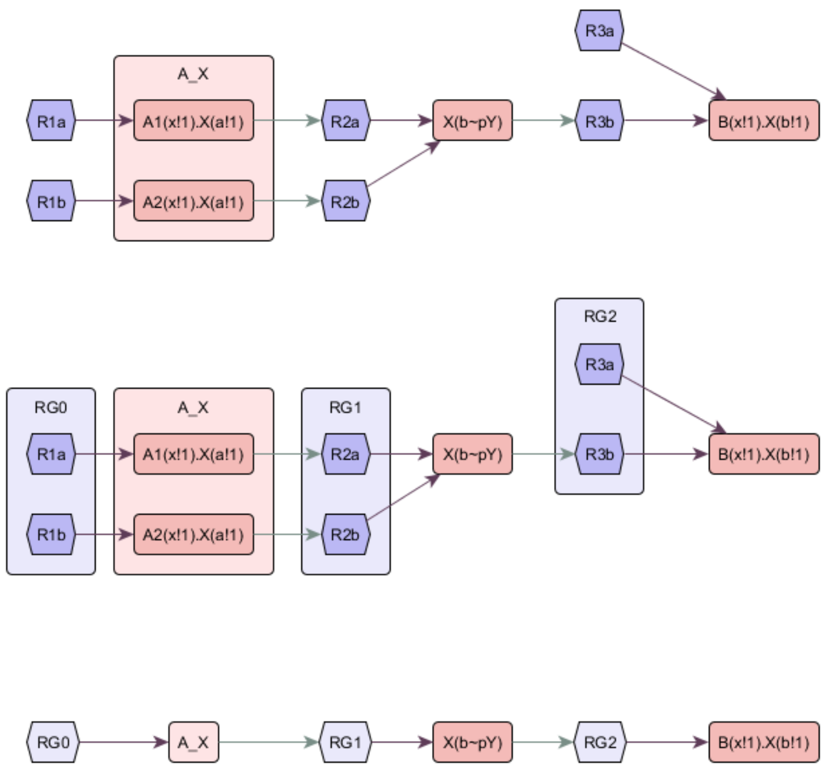


Figure 2-26. Grouping and Collapsing nodes on the regulatory graph. The first panel shows a regulatory graph of a model where kinases A1 and A2 bind a scaffold X resulting in A1(x!1).X(a!1) and A2(x!1).X(a!1) respectively, which leads to the phosphorylated state X(b~pY), which in turn leads to binding of B to X, i.e.

B(x!1).X(b!1). Also shown in the first panel is a grouping of A(x!1).X(a!1) and A2(x!1).X(a!1) bound states under A_X. The second panel shows automated rule-grouping using the first graph as a seed. Rule groups RG1 and RG2 group rules that implement phosphorylation and B-binding respectively irrespective of context edges. Since the user defined a group on A(x!1).X(a!1) and A2(x!1).X(a!1), rules R1a and R1b, which produce A(x!1).X(a!1) and A2(x!1).X(a!1) respectively, are considered similar and grouped under RG0.

2.5.5 Complexity Analysis

Determining the atomic pattern map for every rule in the model as in Section 2.5.3.1 has a time complexity of $\mathcal{O}\left(n_{rule} * n_{\frac{nodes}{rule}} * n_{\frac{neighbors}{node}}\right)$, where n_{rule} is the number of rules, $n_{\frac{nodes}{rule}}$ is a bound on the number of nodes per rule structure graph, and $n_{\frac{neighbors}{node}}$ is a bound on the size of the local subgraph that needs to be explored to construct the atomic pattern. The stoichiometry constraints defined in Definition 2.5-21 and the additional restrictions used in Definition 2.5-27 ensure that $n_{\frac{neighbors}{node}}$ has an upper bound of 4. The size of the largest rule in the model places an upper bound on $n_{\frac{nodes}{rule}}$ since rules are intended to be finite local descriptions of kinetic interactions. Thus, determining the atomic pattern map, and therefore the rule regulatory graph and the model regulatory graph has a time complexity of $\mathcal{O}(n_{rule})$.

To generate the grouped and collapsed regulatory graphs requires additional work in determining local reactant/product relationships on the regulatory graph, sorting them and determining unique combinations, as detailed in Section 2.5.4.5. The time complexity of the grouping step is $\mathcal{O}\left(n_{rule} * n_{\frac{edges}{rule}} \log n_{\frac{edges}{rule}}\right)$, where n_{rule} is the number of rules and $n_{\frac{edges}{rule}}$ denotes the number of reactant/product relationships of each rule to the set of atomic patterns. The worst case scenario is when a number of molecule addition or deletion transformations are employed in the same rule, which will result in a high value for $n_{\frac{edges}{rule}}$. Such pathological rules are rarely encountered and are typically a sign of poor model construction if encountered frequently within the same rule-based model. The average case encountered is where the rule implements at most one or two transformations involving AddBond, DeleteBond and ChangeState, and in this case, $n_{\frac{edges}{rule}}$ has a low upper bound that is below 10. Thus, for the average rule-based model, the time complexity for the grouping step is $\mathcal{O}(n_{rule})$.

2.5.6 Comparisons to Other Approaches

SBGN Process Description (Section 2.3.2.1) provides a local visualization that can be adapted for the rule Petri net and it suffers from the same limitations as the rule Petri net, i.e. it cannot be aggregated over partial overlaps and there is significant repetition involved in showing reactants and products separately. The compact rule visualization is more compact than the rule Petri net (Section 2.4.3), and is tailored specifically for communicating graph transformations in reaction rules.

In comparison to the rule influence diagram (Section 2.4.5) which has $\mathcal{O}(n_{rule}^2)$ complexity, the regulatory graph requires an $\mathcal{O}(n_{rule})$ time on average to be constructed from a rule-based model. This is because building the regulatory graph does not involve explicit comparisons of whole patterns in rules. This also makes the regulatory graph have fewer edges per node than the rule influence diagram, which improves visual comprehension. The regulatory graph also has a similar scalability advantage over the Kappa story (Section 2.4.6), which requires sampling combinations of rules that form causal sequences.

Additionally, in comparison to the Kappa story (Section 2.4.6), the regulatory graph is a truly global visualization, i.e. it does not require observables to be defined beforehand for the model graph to be synthesized. On the other hand, the Kappa story has the advantage of defining arbitrarily complex observables, whereas, the regulatory graph only uses atomic patterns. Future work in this direction would involve extending the regulatory graph abstraction to accommodate such observables.

In comparison to the Simmune Network Viewer (Section 2.4.7), the regulatory graph enables identifying signal flows through internal states such as phosphorylated states. Additionally, the graph is complete, unlike Simmune Network Viewer, which does not resolve patterns involving wildcard bonds. Finally, the grouping approach provided on the regulatory graph is flexible and allows deploying different grouping strategies. This ability to systematically tune the resolution of the graph is not present in Simmune Network Viewer, which opts for a fixed two-layer view.

The collapsed regulatory graph contains contextual information for processes at the same level of resolution as the molecular interaction map (Section 2.3.1), the SBGN Entity Relationship diagram (Section 2.3.2.2) and the Extended Contact Map (Section 2.3.3). It may seem that these maps contain more information than the collapsed regulatory graph because of the variety of contingency relations that are allowed to be represented. However, extracting these contingencies from a formal model is not trivial or generalizable, and typically these diagramming frameworks require the modeler to manually interpret the model in terms of the diagrammatic elements. However, since SBGN Process Description is also a bipartite abstraction, there is potential for the regulatory graph to be automatically converted to an SBGN PD diagram.

It is fairly obvious in most quantitative fields that information encoded in a high-resolution format can be systematically degraded into a lower-resolution format, and that the converse is not possible without making strong assumptions about the nature of the encoded information. For example, high-resolution images can be easily converted to low-resolution images that encode less information, but the reverse is not possible in the general case. In a similar vein, mechanistic models of biochemistry require graph-based formalisms such as that of BioNetGen in order to explicitly represent site-based interactions, and human-readable pathway diagrams have to necessarily be achieved by coarse-graining these explicit models. The Rxncon framework fails to recognize this disparity in resolution, and unlike the BioNetGen specification, the Rxncon specification cannot represent the full spectrum of site-based kinetic interactions [65]. In this work I have provided the opposite approach to Rxncon: begin with a BioNetGen kinetic specification of arbitrary complexity, and then apply systematic transformations and coarse-graining procedures to generate a simple human-readable diagram.

2.5.7 Implementation

The methods developed here have been implemented and packaged with BioNetGen 2.2.6 (bionetgen.org). In a BioNetGen model, "actions" are calls to methods that analyze or simulate models. The visualization methods can be accessed with the visualize() action.

For compact rule visualization, the following action is used:

```
visualize( { type=>"ruleviz_operation"} )
```

For generating complete regulatory graphs from reaction rules, the following action is used:

```
visualize( { type=>"regulatory", background=>1 } )
```

In both cases, to generate separate files for each rule, the flag `each=>1` can be used.

To generate a text file that contains all the rules, atomic patterns and relationships between them, use:

```
visualize( { type=>”regulatory”, background=>1, textonly=>1 } )
```

To turn on the pruning heuristic, `background=>0` is used. This is also the default setting.

To enable automated grouping of rules, use:

```
visualize( { type=>”regulatory”,groups=>1} )
```

To enable collapsing of the generated group structures, use:

```
visualize( { type=>”regulatory”,groups=>1,collapse=>1} )
```

Options files can be provided to the visualize action to modify the background assignment and to provide groups of atomic patterns as seed for the rule-grouping algorithm:

```
visualize( { type=>”regulatory”,opts=>[“opts1.txt”, “opts2.txt”] } )
```

In the options file, background options are specified as follows:

```
begin background

        begin include

                <whitespace-separated list of atomic patterns>

        end include

        begin exclude

                <whitespace-separated list of atomic patterns>

        end exclude

end background
```

Atomic patterns in the include and exclude sections are populated from the text file generated by `textonly=>1`.Atomic patterns in the include section are included in the background, and those in the exclude section are excluded from the background, with both inclusions and exclusions overriding the pruning heuristic.

In the options file, groups of atomic patterns are specified as follows:

```
begin classes

    begin groupname1

        <whitespace-separated list of atomic patterns>

    end groupname1

    begin groupname2

        <whitespace-separated list of atomic patterns>

    end groupname2

end background
```

These groups are used to seed the automated rule-grouping algorithm. Names for rule groups are assigned automatically. A detailed and updated version of the documentation can be found at http://bionetgen.org/index.php/Visualization.

2.6 VISUALIZATION CASE STUDIES

In this section, I demonstrate the effectiveness of the visualizations defined in Section 2.5 for local and global views of rule-based models, and I will do so by applying the methods developed here on previously published models. As explained in Chapter 1, each rule models an explicit reaction mechanism, with patterns explicitly showing the participating sites in detail. The local perspective in a rule-based model is conveying the requirements and graph transformations in each reaction rule, and the global perspective is the network of signal flows between rules and sites in the model. In this work, for the first time, we demonstrate how to achieve a global visualization of signal flow from a rule-based model, as well as, how to simplify automatically generated diagrams of large models into compact pathway diagrams. Finally, we show how regulatory graphs are useful for identifying cascades and feedback loops in a model which improve the understanding of a model.

2.6.1 Visualizing Mechanisms in Detail

In the BioNetGen specification, each reaction rule has reactant and product patterns, with the reactants encoding the site configuration that drives the process (the "before" state), and the products encoding the transformation relative to the reactants (the "after" state). Compact rule visualization (Section 2.5.2.6) was designed to explicitly show the graph transformation modeled in a rule, instead of the implicit "before/after" representation used in the syntax. Here, I will use four rules from the Faeder et al. model of signaling from the FcεRI receptor in mast cells [35] to demonstrate the utility of the compact rule visualization in conveying these mechanisms. In the model, rules R3 and R6 are as follows (underline for comparison):

```
R3: Rec(b~Y)  + Lyn(U,SH2) <->  Rec(b~Y!1).Lyn(U!1,SH2)

R6: Rec(b~pY) + Lyn(U,SH2) <->  Rec(b~pY!1).Lyn(U,SH2!1)
```

Both rules model binding of cytoplasmic Lyn with the β domain (denoted 'b') of the receptor under slightly different local conditions. A bond is indicated by !1 placed next to the two components forming the bond. In both R3 and R6, the Lyn kinase has to have both U and SH2 domains unbound. In R3, the binding site on the receptor is unphosphorylated (~Y), and the binding site on Lyn is the 'unique' domain U. In R6, the binding site on the receptor is phosphorylated (~pY), and the binding site on Lyn is the SH2 domain.

In biochemical parlance, rule R3 models constitutive recruitment of cytoplasmic Lyn to the receptor, and rule R6 models activated recruitment since it requires the phosphorylated state of the β domain. Rules R3 and R6 are examples of explicit reaction mechanisms: the sites driving the process, the sites affected by the process, the local arrangement of these sites within complexes, the relevant internal states (unphosphorylated/phosphorylated) and binding states (bound/unbound), are all explicitly specified. As shown in Figure 2-27, compact rule visualization shows the molecules and sites involved and differentiates between the modified and unmodified sites using graph operation nodes such as AddBond. It also enables a side-by-side display and comparison of the two mechanisms.

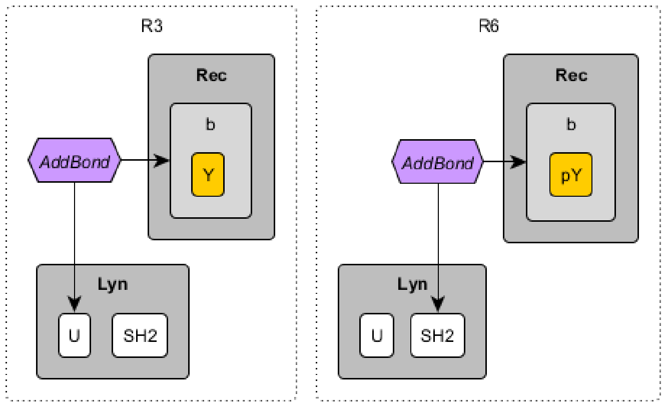


Figure 2-27. Compact rule visualization of rules R3 and R6 from Faeder et al. [35] modeling recruitment of cytoplasmic Lyn kinase to the $\boldsymbol{\beta}$ domain ('b') of the Fc$\boldsymbol{\varepsilon}$RI receptor (Rec). In R3, recruitment is constitutive, i.e. Lyn binds via its U domain to an unphosphorylated $\boldsymbol{\beta}$ domain (b~Y), whereas in R6, recruitment is activated, i.e. Lyn binds via its SH2 domain to a phosphorylated $\boldsymbol{\beta}$ domain (b~pY). The action of each rule is made explicit by using a graph operation node (AddBond), and the rules can be compared side-by-side.

Now consider the rules R4 and R7, as shown below (underline for comparison).

```
        R4: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)

        R7: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)
```

Note that the patterns used in R4 and R7 are larger than the ones in R3 and R6, and involve more molecules, components and bonds (!1, !2, !3, etc.). Although the syntax is precise and flexible for building arbitrarily large graphs, the content of these large graphs are not immediately obvious to the human eye. However, this information can be easily conveyed using compact rule visualization, as in Figure 2-28. From this figure, we can see that both rules R4 and R7 require a receptor dimer which is formed when a bivalent ligand crosslinks two receptors. In both R4 and R7, one of the receptors in the dimer is bound to a Lyn kinase through its β domain (denoted 'b'), and the other receptor has an unbound β domain that undergoes phosphorylation, as indicated by a ChangeState operation from Y to pY. The only difference between the two rules is the manner of recruitment of Lyn to the first receptor: in R4, Lyn is bound via the U domain to the unphosphorylated β domain, whereas in R7, Lyn is bound via SH2 domain to the phosphorylated βdomain.

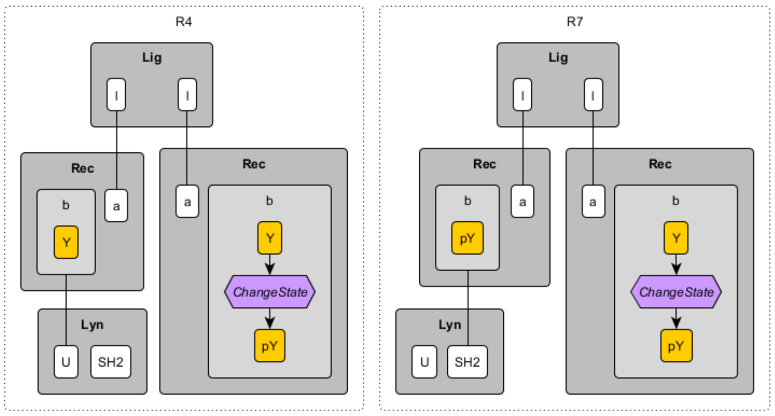


Figure 2-28. Compact rule visualization of rules R4 and R7 from Faeder et al. [35]. In both rules, a receptor dimer is required which is formed due to crosslinking by a bivalent ligand. One half of the dimer binds Lyn kinase via its $\boldsymbol{\beta}$ domain, which results in phosphorylation of the $\boldsymbol{\beta}$ domain on the other half of the dimer. In R4, Lyn is recruited via its U domain to the unphosphorylated $\boldsymbol{\beta}$ domain whereas in R7, Lyn is recruited via its SH2 domain to the phosphorylated $\boldsymbol{\beta}$ domain.

2.6.2 Visualizing Interactions of Mechanisms

Signal flow in a rule-based model constitutes of effects of individual rules on sites, which can subsequently influence other rules. For example, consider the four rules shown in Figure 2-27 and Figure 2-28. R4 requires the constitutively bound state of Lyn to receptor, which is produced by R3. R7 requires the actively bound state of Lyn to receptor, which is produced by R6. R6 requires the phosphorylated state on the β domain which is produced by rules R4 and R7. The compact rule visualization, while useful for understanding individual rules, cannot provide a global picture of signal flows. On the other hand, the regulatory graph, as developed in 2.5.4.1, was designed to show relationships between rules and their sites of action. Figure 2-29 is a pruned regulatory graph (Section 2.5.4.4), on which we can see that constitutive binding via R3 leads to phosphorylation via R4, and then there is a positive feedback loop between activated recruitment via R6 and phosphorylation via R7.

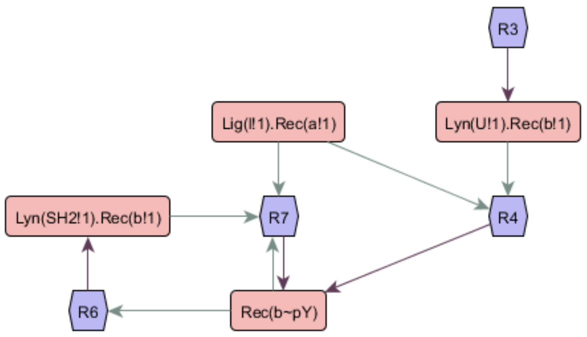


Figure 2-29. Regulatory graph of rules R3, R4, R6 and R7 that were visualized in **Figure 2-27** and **Figure 2-28**. The graph has two types of nodes: rules and atomic patterns. Atomic patterns represent distinct classes of structural features, with bonds such as Lyn(U!1).Rec(b!1) and phosphorylated states such as Rec(b~pY). Atomic patterns representing free binding sites such as Lyn(U) and unphosphorylated states such as Rec(b~Y) are not shown. The regulatory graph enables identifying the positive feedback loop between R6 and R7.

2.6.3 Visualizing Models as Pathway Diagrams

The model regulatory graph (Section 2.5.4.3) is a complete, automated, flow-based visual representation of the model that is aggregated from individual rules. Even though it has a better scaling performance than the rule influence diagram or the Kappa story (see Section 2.5.5), the visual quality of these graphs was still found to degrade rapidly as the number of rules increases, which is expected behavior for the general class of node-link visualizations [75]. To usefully visualize rule-based models of any size, we needed to be able to control the size and complexity of the generated diagrams. So, we developed a series of coarse-graining procedures that incorporate both user input and graph analysis to transform the automatically generated model regulatory graph into a smaller pathway diagram (Sections 2.5.4.4, 2.5.4.5, 2.5.4.6). We demonstrate this using the Faeder et al. model of mast cell signaling from the FcεRI receptor [35].

The Faeder et al. model uses four molecule types (Lig, Rec, Lyn, Syk), as seen in the formal contact map in Figure 2-30, and requires 24 reaction rules to specify very detailed kinetic interactions involving specific sites on these molecules [35]. The ligand has two symmetric receptor binding sites, and this leads to the formation of a crosslinked receptor dimer. The receptor has intracellular domains β and γ that can be phosphorylated. Both phosphorylated and unphosphorylated forms of the β domain bind Lyn, whereas only the phosphorylated form of the γ domain binds Syk. When one of these kinases is recruited to one side of the dimer, it phosphorylates substrates found on the other side of the dimer, a phenomenon called trans-phosphorylation. Lyn trans-phosphorylates both Lyn and Syk binding sites, as well as the linker region on recruited Syk. Syk trans-phosphorylates the activation loop on recruited Syk, which also enhances Syk's kinase activity.

Table 1.2-1 summarizes the 24 rules and the kinetic mechanisms they represent. The complete model regulatory graph, automatically generated from the model specification in BioNetGen, is shown in Figure 2-31. The graph has 45 nodes and 125 edges, and although it provides a flow-based representation, it is still too large and complex to be useful as a human-readable pathway diagram. We then implemented the following steps to coarse-grain the graph into a simpler version that conveys model intent.

The first step is to remove rules and atomic patterns that can be considered redundant (Section 2.5.4.4). On this graph, we removed the dephosphorylation rules, the reverses of binding rules, and the unphosphorylated states and free binding sites on the receptor and kinases. The background assignment was performed with a heuristic, but it can also be provided by the user in the form of an options file. This results in the pruned regulatory graph in Figure 2-32, which has 22 nodes and 45 edges. On this graph, it is easier to track the signal flow: ligand binding to Lyn recruitment to receptor phosphorylation to Syk recruitment and phosphorylation. The second step is to impose an organization of atomic patterns and rules into groups. For this model, user input was provided to group atomic patterns as follows: Lig = {Lig(l)}, Lig_Rec={Lig(l!1).Rec(a!1)}, Rec_p = {Rec(b~pY), Rec(g~pY)}, Rec_Syk = {Rec(g!1).Syk(tSH2!1)}. The automated rule-grouping algorithm (Section 2.5.4.5) was then used to analyze the graph and identify groups of rules that have similar effects on atomic patterns. Figure 2-33 shows the grouped regulatory graph. The final step is to reduce groups of nodes to a single node each (Section 2.5.4.6). The collapsed regulatory graph, as shown in Figure 2-34, is much more compact than previous versions (15 nodes, 25 edges) and shows the signal flows as one would on a simple pathway diagram, using broad classes of sites and processes.

The graph in Figure 2-34 can also be used to identify feedbacks in the model: the positive feedback between activated Lyn recruitment and receptor phosphorylation, and the self-enhancing effects of receptor phosphorylation and Syk activation loop phosphorylation. This was not possible with any previous automated global visualization for rule-based models.

When atomic patterns are left ungrouped, the regulatory graph makes it possible to discern the finer aspects of regulation involving specific atomic patterns. For example, in Figure 2-34, one can distinguish between processes that phosphorylate Syk at two different sites and we can see that one of them is Lyn-dependent, whereas the other is not. Similarly, one can distinguish that activated recruitment of Lyn participates in a positive feedback loop, but constitutive recruitment does not. It is possible to coarse-grain the model regulatory graph to different extents, by using a different grouping strategy. For example, in addition to the grouping specified above, if we also specify Rec_Lyn = {Lyn(U!1).Rec(b!1), Lyn(SH2!1).Rec(b!1)} and Syk_p = {Syk(a~pY), Syk(l~pY)}, followed by automated rule grouping and collapsing, this results in the graph in Figure 2-35, which has 11 nodes and 18 edges. However, the regulatory description in this figure is more coarse-grained than previous ones, because Syk phosphorylation and Lyn recruitment are treated generically. By choosing the specific number and size of each group of atomic patterns, the modeler is able to modulate the complexity of the resultant diagram and the degree to which the details of the model regulatory graph are coarse-grained. This allows the modeler to tailor the generated diagram for a specific purpose or audience.

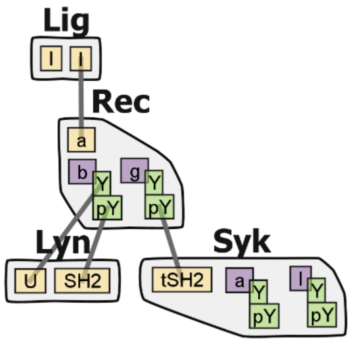


Figure 2-30. Formal contact map of the Faeder et al. model [35]. The model has four molecule types. The ligand has two symmetric binding sites for the receptor. The receptor has $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ domains that bind Lyn and Syk respectively, with Lyn being able to use U or SH2 domains to bind the receptor. The receptor $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ domains as well as the activation loop (denoted ‘a’) and the linker region (denoted ‘l’) on Syk can be in unphosphorylated (Y) or phosphorylated (pY) states.

Table 2-2. Descriptions of mechanisms modeled as reaction rules in the Faeder et al. model [35]. Rule names beginning with '_reverse_' are reverses of binding rules. Multiple rules are required when there are kinetic contributions from variations in molecular states or binding interactions, e.g. trans-phosphorylation of $\boldsymbol{\beta}$ by Lyn is assumed to have different rates when the kinase is recruited to the unphosphorylated site (rule R4) or the phosphorylated site (rule R7).

Rule Names	Kinetic Interaction Modeled
R1, _reverse_R1	Binding of free ligand to receptor.
R2, _reverse_R2	Crosslinking of ligand-bound receptor with another receptor.
R3, _reverse_R3	Binding of Lyn to unphosphorylated β domain of receptor.
R6, _reverse_R6	Binding of Lyn to phosphorylated β domain of receptor.
R9, _reverse_R9	Binding of Syk to phosphorylated γ domain of receptor.
R4, R7	Trans-phosphorylation of β by Lyn.
R5, R8	Trans-phosphorylation of γ by Lyn.
R10, R11	Trans-phosphorylation of Syk linker region by Lyn.
R12, R13	Trans-phosphorylation of Syk activation loop by Syk.
R14	Dephosphorylation of β domain of receptor.
R15	Dephosphorylation of γ domain of receptor.
R16, R18	Dephosphorylation of linker region of Syk.
R17, R19	Dephosphorylation of activation loop of Syk.

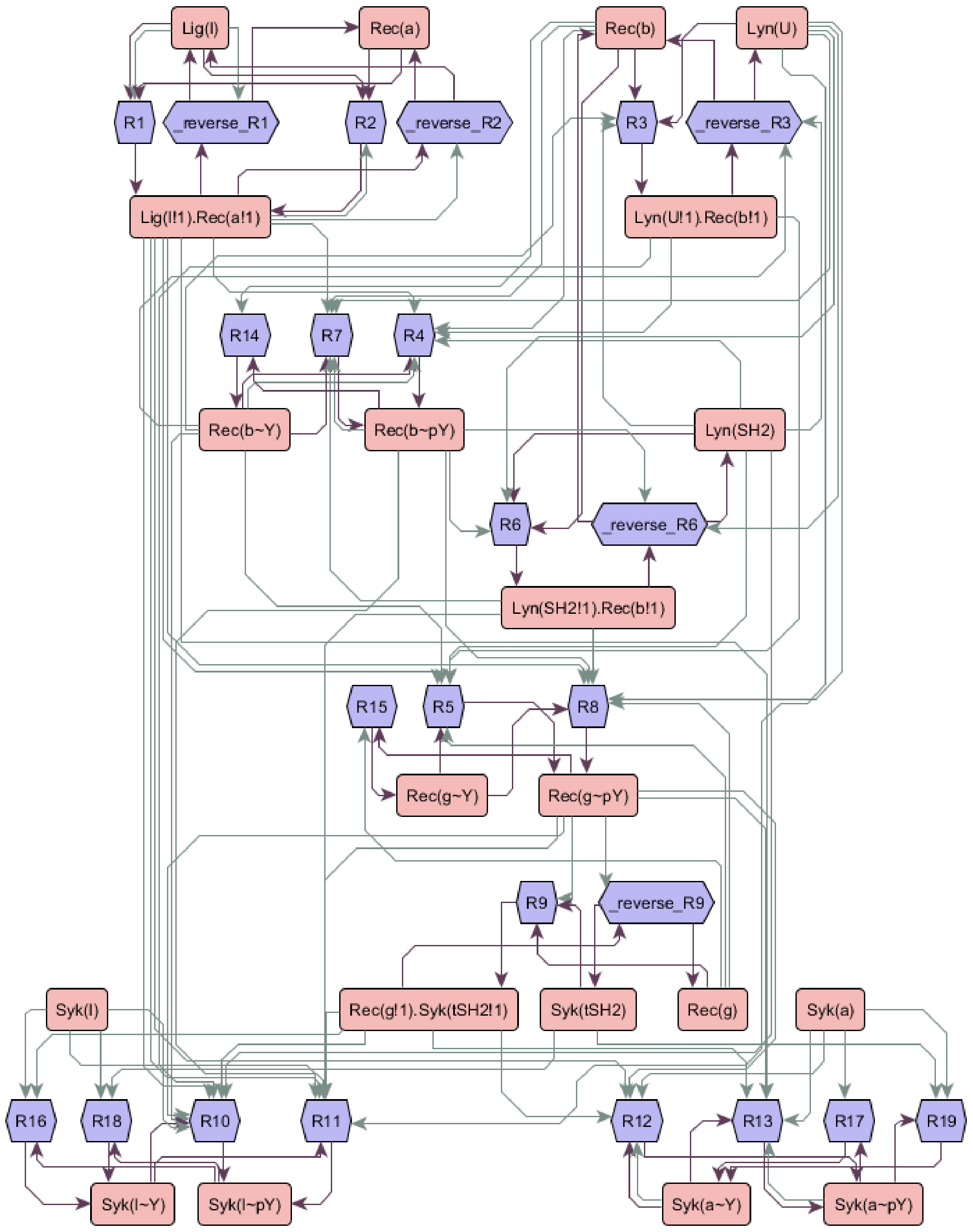


Figure 2-31. Complete regulatory graph of the Faeder et al. model of FcεRI signaling [35], automatically generated from the BioNetGen specification. Dark edges indicate consumption and production and light edges indicate context influence.

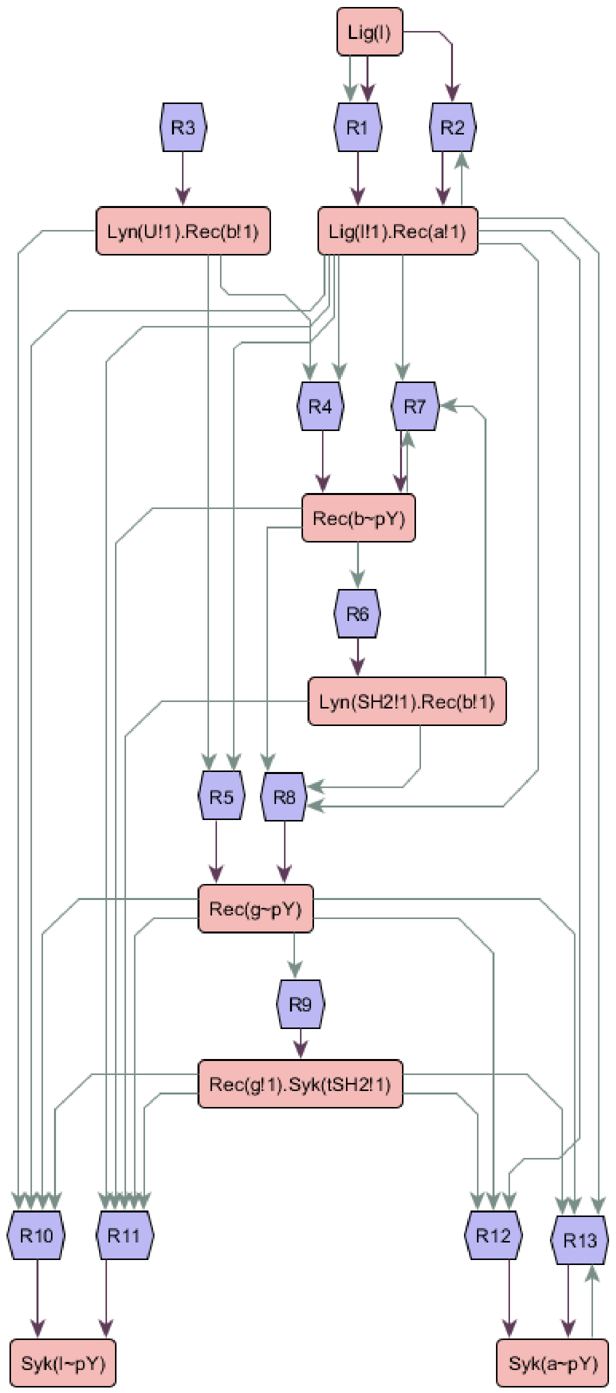


Figure 2-32. Pruned regulatory graph of the Faeder et al. model generated from the model regulatory graph in **Figure 2-31** by applying a heuristic and removing a few rules (reverses of binding rules, dephosphorylating rules) and atomic patterns (free binding sites and and unphosphorylated states on Rec, Lyn and Syk).

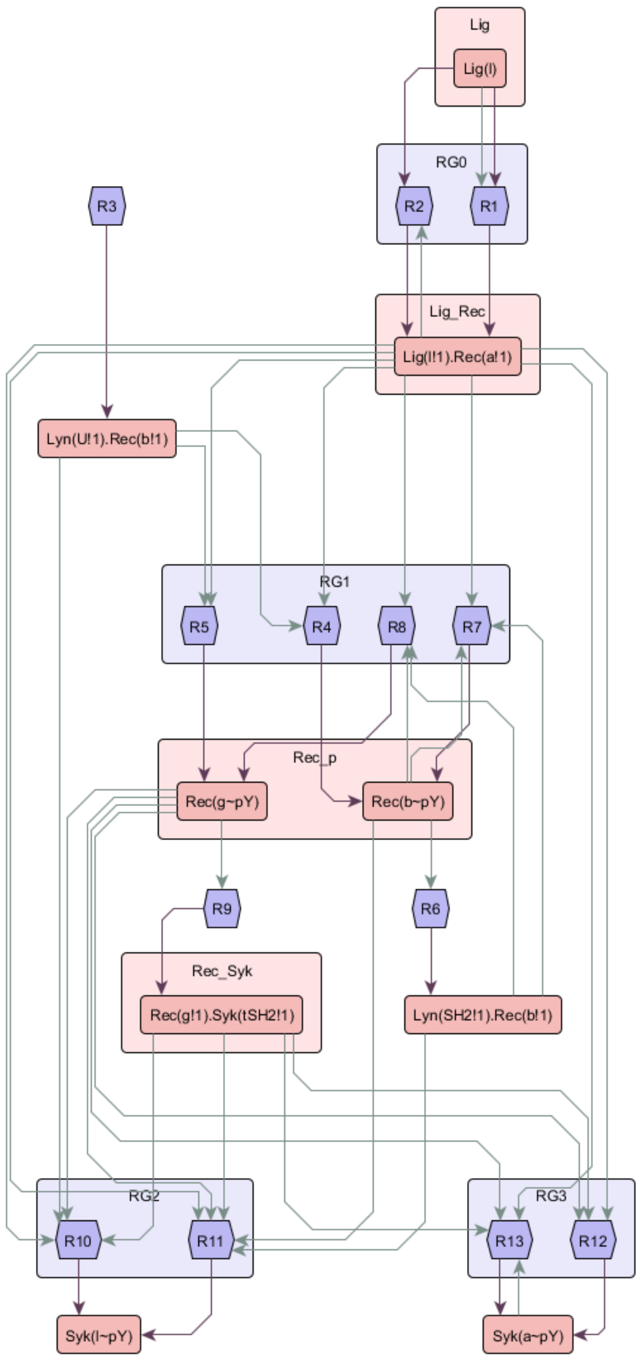


Figure 2-33. Grouped regulatory graph of the Faeder et al model generated from the pruned graph in **Figure 2-32**. We used user input that grouped atomic patterns as follows: Lig = {Lig(l)}, Lig_Rec={Lig(l!1).Rec(a!1)}, Rec_p = {Rec(b~pY), Rec(g~pY)}, Rec_Syk = {Rec(g!1).Syk(tSH2!1)}. The rule groups were automatically determined using this information, such that each group of rules produces the same effect on the sites involved.

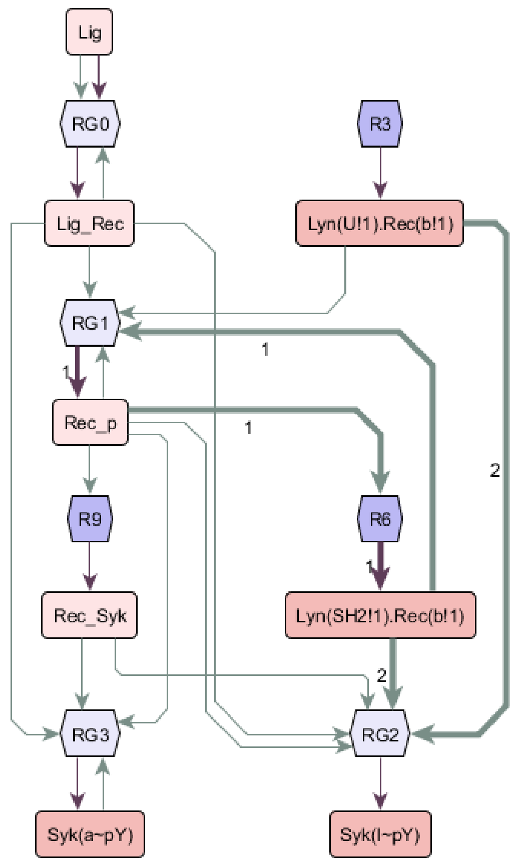


Figure 2-34. Collapsed regulatory graph of the Faeder et al. model generated from **Figure 2-33** by automatically reducing groups of nodes to single representative nodes. At this level of resolution, the signal flow in the system is much more clarified and resembles a simple pathway diagram. Ligand binding and crosslinking (RG0) and constitutive Lyn recruitment (R3) initiate the cascade, which culminates in phosphorylation of Syk linker region (RG2) and activation loop (RG3). Signaling motifs can be identified on this graph, such as the positive feedback loop involving constitutive Lyn recruitment and receptor phosphorylation {R6, Lyn(SH2!1).Rec(b!1), RG1, Rec_p} and the self-enhancing effects of receptor phosphorylation {RG1. Rec_p} and Syk activation loop phosphorylation {RG3, Syk(a~pY)}. The ungrouped atomic patterns enable discerning the finer features of regulation (thick edges), e.g. Syk linker region phosphorylation (RG2) is Lyn-dependent (edges labeled 2), but Syk activation loop phosphorylation (RG3) is not, and that activated Lyn recruitment (R6) is part of a positive feedback loop, whereas constitutive recruitment (R3) is not (edges labeled 1).

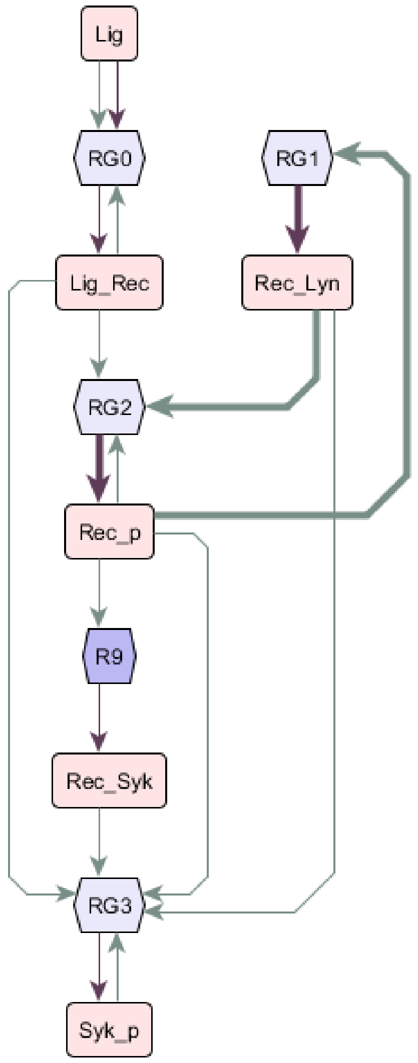


Figure 2-35. Collapsed regulatory graph of Faeder et al. model, with an alternative grouping of atomic patterns provided by the user. In addition to the groups of atomic patterns used in **Figure 2-33**, here we used the additional groups Syk_p = {Syk(a~pY),Syk(l~pY)} and Rec_Lyn = {Lyn(U!1).Rec(b!1), Lyn(SH2!1).Rec(b!1)}. The resultant collapsed graph, as shown here, is simpler than **Figure 2-34**: the two Syk phosphorylated states have been merged into one, and the two Lyn-recruited states have been merged into one. As a consequence, Lyn recruitment and Syk phosphorylation are treated generically, and the finer regulatory features that were discerned in **Figure 2-34** cannot be identified in this figure. For example, the feedback loop is routed through the generic Lyn-recruited state Rec_Lyn (thick edges).

2.6.4 Visualizing Large Libraries of Rules

Many recent publications in the literature involve building a repository of reaction rules that comprehensively document all site-based interactions in a particular biochemical system, typically a signaling pathway initiated from a specific family of cell surface receptors. Examples of modeled receptor families include the ErbB family of receptor tyrosine kinases [46], of which the epidermal growth factor receptor is a prominent member [25], the high affinity IgE receptor FcεRI [44], the T-cell receptor [45], and yeast pheromone receptor Ste2 [47]. These models typically have 10s to 100s of rules, but by applying commonly used as well as model-specific grouping strategies to the model regulatory graphs (Sections 2.5.4.5, 2.5.4.6), we were able to generate relatively compact signal flow diagrams. Importantly, we were able to use these diagrams to identify signaling motifs such as cascades and feedback loops, which was not possible using previous automated global diagrams of rule-based models. Here we showcase regulatory graphs of Creamer et al. [46] and Chylek et al. [44], and discuss the strategies used in complexity reduction.

The Creamer et al.[46] model, whose regulatory graph is shown in Figure 2-36, has 19 molecule types, of which four are receptors from the ErbB family (EGFR, ErbB2, ErbB3, and ErbB4) and two are ligands that bind these receptors (EGF, HRG). In the model, each receptor can bind a ligand molecule (except ErbB2), which leads to dimerization of receptors on the membrane, which in turn activates an intracellular kinase domain on these receptors, which in turn phosphorylates the receptor tails. Adaptors and signal mediators bind to specific phospho-motifs in the receptor tails and initiate the MAP kinase pathway and PIP3/Akt1 pathway, whose primary outputs are activated Erk2 and activated Akt1 respectively. The model also includes internalization of ligand-bound receptors, feedback from Erk2 and Akt1, as well as crosstalk between the two pathways [46].

The Creamer et al.[46] model faces severe combinatorial complexity, even when modeled as reaction rules, because of dimerization between receptor types: every receptor type can homo-dimerize with another receptor of the same type or hetero-dimerize with a receptor of a different type, leading to 16 different dimer configurations (4x4), each of which requires ligand-binding, dimerization, and phosphorylation rules. The model requires 625 reaction rules in all and is currently the largest rule-based model in existence and is a challenge to visualize [46]. The complete model regulatory graph has 939 nodes and 5110 edges. The following grouping strategy was applied: the ligand-bound states were grouped together under Lig_Rec, and the dimer states were grouped under Dimer. Multiple phosphorylated states on a molecule were grouped into a collective phosphorylated state, and this was done for each molecule type. For kinases where binding to the kinase active-site cleft was explicitly modeled, the convention A_on_B was used to show the binding of a substrate B to the active-site of a kinase A. After pruning, grouping and collapsing, the resultant graph, shown in Figure 2-36 has 85 nodes and 157 edges and was laid out to visually emphasize the different modules in the system: surface interactions, internalization, adaptor binding, KRas activation, MAPK pathway, PI3K activation, Akt1 pathway, Erk2 feedback and Akt1 feedback.

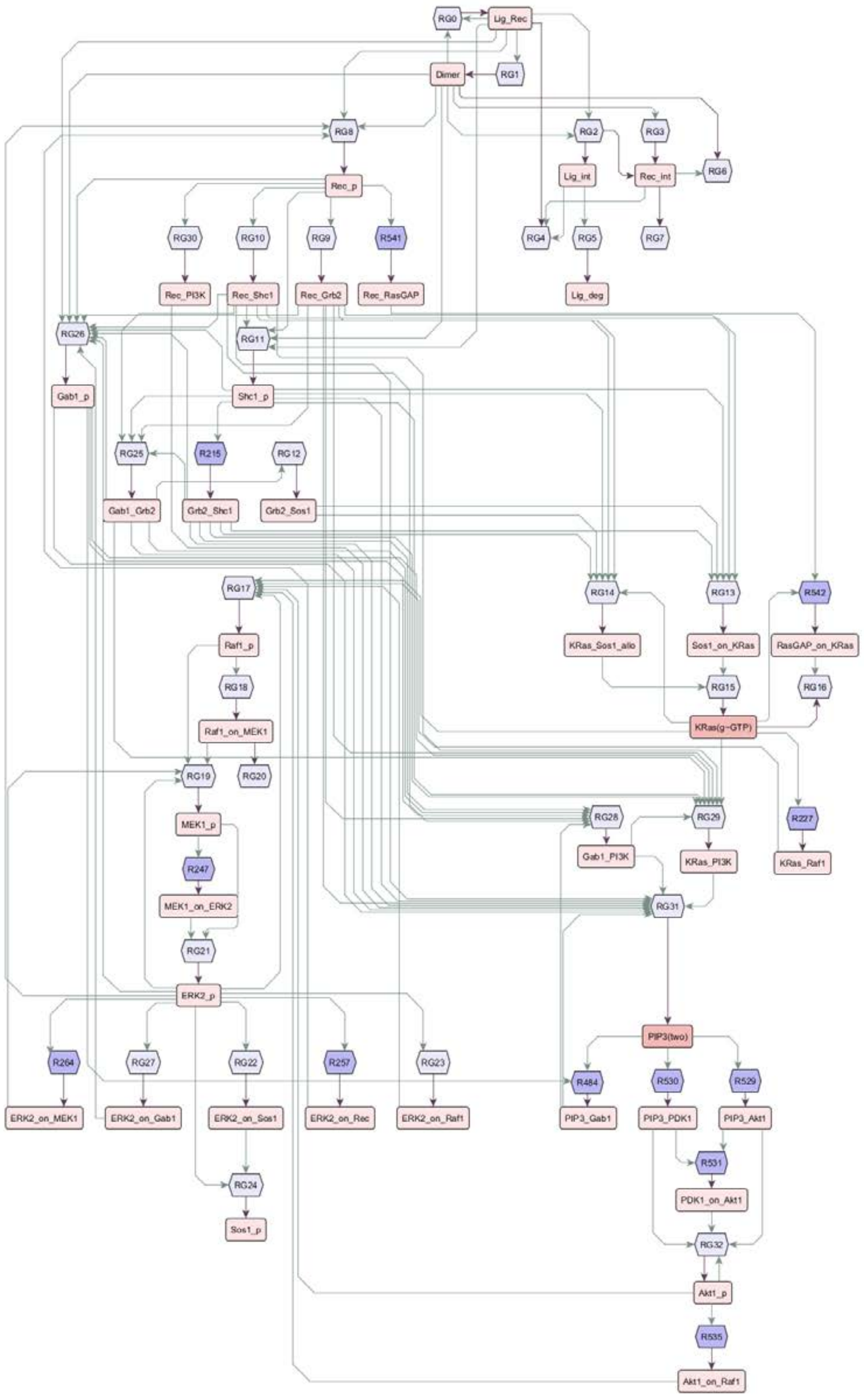


Figure 2-36 Regulatory graph visualization of the Creamer et al. model of signaling from ErbB family of receptors [46]. The grouping of atomic patterns was provided by the user and rule-grouping was automated.

Chylek et al. model signaling from FcεRI receptor, which plays an important role in mast cell signaling [44]. The ligand is assumed to be a multivalent entity that crosslinks receptors, which initiates trans-phosphorylation events by cytoplasmic kinases (Lyn, Fyn, Csk, Syk, BTK) recruited to the receptor, and binding of adaptors (Grb2, Grap2). Also in the model are scaffolds (Pag1, Lat) that bring together many kinases and their substrates. This results in activation of multiple enzymatic processes that regulate phosphoinositide levels, including PI3K, Inpp5d and Plcg1 (with PTEN as a background modulator). Chylek et al. [44] actually provide two different models. The first model is larger, encompasses the whole system, and represents phosphoinositide entities as whole molecules, such as PIP2() and PIP3(). The second model is smaller, but focuses on the interaction of enzymes PI3K, Inpp5d and Plcg1 with phosphoinositide entities in high detail, with the phosphoinositides modeled as structured molecule named PI with sites OH3, OH4 and OH5 representing the 3’,4’ and 5’ hydroxyl positions, internal state ~P representing phosphates at those positions, and the site ‘head’ representing the phosphoinositide head group [44]. We merged both models into a single model, replacing phosphoinositide-modifying rules in the first model with the detailed interactions from the second model. The resulting model has 17 molecule types and requires 178 reaction rules. The complete model regulatory graph has 313 nodes and 1084 edges. In general, the grouping strategy mirrored the one used for Creamer et al.[46]: pairs of binding molecules were grouped together, and phosphorylation sites on the same molecule were grouped together (with the exception of Lyn and Fyn which I will discuss shortly). After pruning, grouping and compressing, the resultant graph is shown in Figure 2-37, and has 70 nodes and 129 edges. The graph was laid out to emphasize the different parts of the system: the interactions governing Lyn/Fyn regulation, the Syk-Lat cascade beginning from ligand-binding, the initiation of the three arms of the phosphoinositide pathway, and the feedbacks involved. In Figure 2-38 and Figure 2-39, I show different subsets of the graph in Figure 2-37, and I will use these to highlight the model-specific grouping strategies involved, as well as the ability to identify signaling motifs such as feedback and feed-forward loops.

In Figure 2-38, I show a subset of the graph in Figure 2-37 that highlights detailed and complex interactions between the receptor, the cytoplasmic kinases Lyn, Fyn and Syk, and the scaffold Pag1. Lyn and Fyn are structurally and functionally homologous Src kinases, so we lumped functionally similar sites on Lyn and Fyn within the same group, as if they belong to a generic SrcKinase. Multiple domains on Lyn and Fyn act in concert to either bind receptor, or the scaffold Pag1 or to bind in an intramolecular fashion and exist in a self-inhibited state. These domains were grouped under SrcKinaseBindingGroup. Phosphorylation sites on Lyn and Fyn could be distinctly classified as those with positive or negative effects on Lyn/Fyn activity, and so these were grouped under two different collectives: SrcKinaseActivation_p and SrcKinaseInhibition_p. In Figure 2-38, the thick edges outline the canonical signal flow, and the boxes were added to emphasize positive feedback loops. The initial cascade causes activation of Lyn and Fyn: Lyn/Fyn bind receptor (RG1), phosphorylate Lyn/Fyn binding sites on the receptor (RG3) in a positive feedback loop, then are phosphorylated themselves on their activating phospho-motifs (RG5). Activated Lyn and Fyn are important components of the Syk/Lat activation cascade that is not shown in this subset. Following Lyn/Fyn activation is a cascade that causes downregulation: activated Lyn/Fyn bind Pag1 and phosphorylate Pag1 in a similar positive feedback loop (RG7, RG8), which brings them in proximity to Csk kinase that phosphorylates inhibiting phospho-motifs (RG9), which in turn leads to the self-bound self-inhibited state (RG2), and attenuation of the signal from Lyn/Fyn. Having conveyed the architecture of the system using this diagram, it is much easier to discuss functional aspects of this architecture, for example, Barua et al. [76] discuss how a similar architecture in B cells can result in a pulse-like signal from the Src kinases and how modulating concentrations of the Src kinases can lead to oscillatory behaviors.

In Figure 2-39, we show subsets of the graph in Figure 2-37 that highlight the signaling motifs underlying phosphoinositide regulation. Note that the phosphoinositides are modeled as structured entities, so processes involving PIP2, PIP3 and other phosphoinositides are represented on this graph as processes producing or consuming the OH3, OH5 or the 'head' component. 3'-phosphorylation by R152 represents PIP2 to PIP3 conversion by PI3K. Similarly, 5'-dephosphorylation by R95 represents PIP3 consumption by Inpp5d, and head removal by R159 represents PIP2 hydrolysis by Plcg1. Edges were annotated with labels to refer to different cascades in Figure 2-39. Cascade 1 shows signal flow from phosphorylated Lat (Lat_p) leading to PI3K activation (R152), which results in BTK recruitment to PIP3 at the membrane (R153). Cascade 2 also begins at Lat_p and shows Plcg1 recruitment to the membrane by Lat_p (R132). The two cascades 1 and 2 meet and synergize to cause BTK-dependent activation of Plcg1 (R154, R156), and this signaling motif is called a coherent feed-forward loop. Cascade 3 shows Plcg1 enzyme activity (R159) following activation by BTK. Cascade 4 begins from the receptor phosphorylated state (Rec_b_p) and shows Gab2 phosphorylation by Fyn on the receptor complex (RG15), which leads to PI3K activity (R152), which increases PIP3 levels. Cascade 5 also begins from the phosphorylated receptor and shows Inpp5d recruitment (RG16), which results in Inpp5d activity (R95) and reduces PIP3 levels. The two cascades 4 and 5 originate and diverge from the same point (Rec_b_p), but have opposing effects on PIP3 levels, and this is an example of an incoherent feed-forward loop. Cascade 6, in addition to cascade 4, follows a positive feedback loop involving PIP3-dependent recruitment of Gab2 to membrane (R146) which leads to increased phosphorylation of Gab2 (RG15) and increased PI3K activity (R152). Notably, Chylek et al.

include in their paper manually drawn visualizations that highlight these motifs [44], and in this work, we were able to trace these motifs on a diagram that was automatically generated from the formal BioNetGen specification and a user-specified organization of sites.

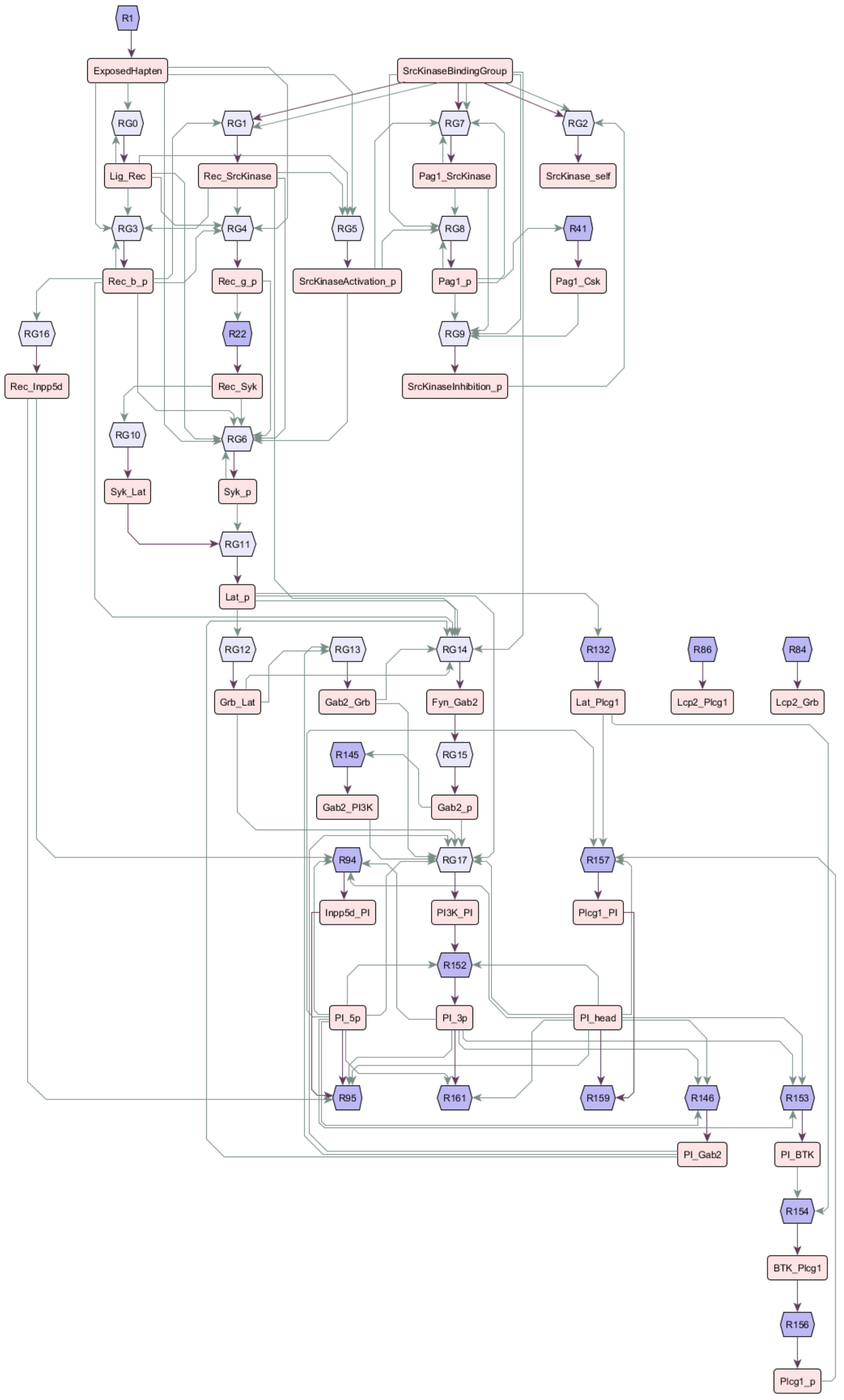


Figure 2-37. Regulatory graph visualization of the Chylek et al. model of signaling from the FcεRI receptor [77]. The grouping of atomic patterns was provided by the user and rule-grouping was automated.

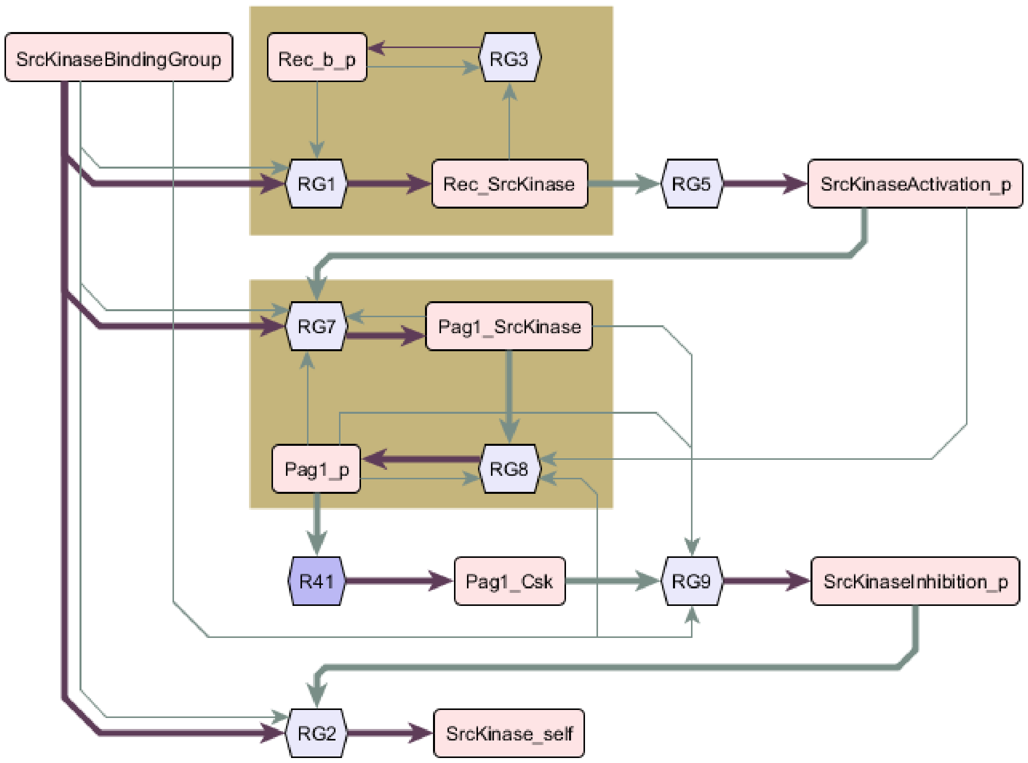


Figure 2-38. Subset of the Chylek et al. regulatory graph, showing interactions of the SrcKinases (Lyn and Fyn) with the FcεRI receptor (full graph in **Figure 2-37**). Thick edges highlight the canonical flow of signal in the model: first, there is initiation of signaling by binding of the SrcKinaseBindingGroup (a group of domains on Lyn and Fyn) with the receptor (RG1). Then, there is a positive signaling cascade where receptor binds SrcKinase (RG1) leading to activation of the SrcKinase by trans-phosphorylation (RG5). Then, there is a negative cascade, which involves SrcKinase binding the scaffold Pag1 (RG7) leading to Csk-dependent phosphorylation of inhibitory sites on the SrcKinase (RG9) which in turn promotest the formation of a self-bound self-inhibited state of the SrcKinase (RG2). Boxes were added to highlight the positive feedback loops between receptor binding to SrcKinase (RG1) and receptor phosphorylation (RG9), and between Pag1 binding to SrcKinase (RG7) and Pag1 phosphorylation (RG8).

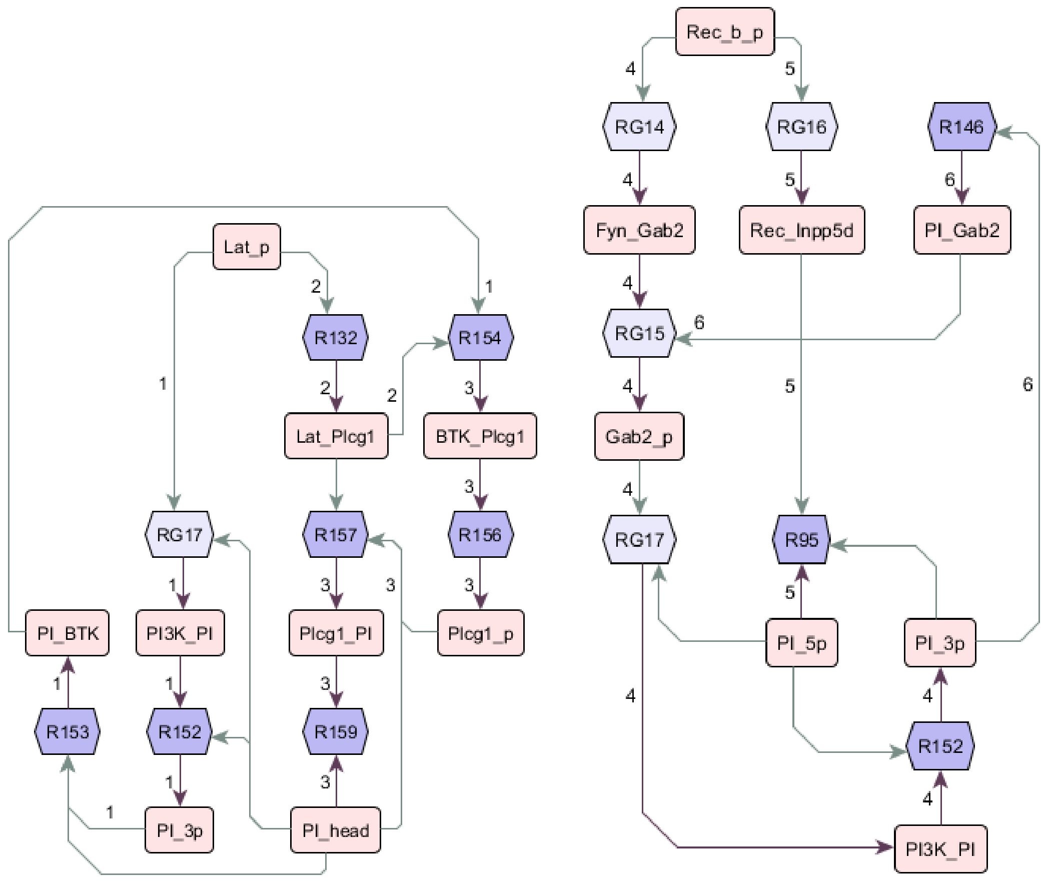


Figure 2-39. Subsets of the Chylek et al. regulatory graph, highlighting other signaling motifs (full graph in **Figure 2-37**). In the graph on the left, the edges labeled 1 follow a path from phosphorylated Lat to PI3K activity (R152) to BTK recruitment (R153), and edges labeled 2 follow a path from phosphorylated Lat to Plcg1 recruitment (R132). Both paths converge onto BTK-Plcg1 binding (R154) and Plcg1 activation (R156). This is an example of a coherent feed-forward loop, where multiple paths converge and synergize. The edges labeled 3 follow a path that results in a negative feedback: Plcg1 activation (R156) leading to PIP2 cleavage that consumes the phosphoinositol head group (PI_head), which in turn is required context for RG17 in branch 1. On the graph on the right, the edges labeled 4 follow a path from phosphorylated receptor to Gab2 phosphorylation (RG15) which leads to PI3K activity (R152) that produces the 3'-phosphate (PI3P), increasing PIP3 levels, and edges labeled 5 follow a path from phosphorylated receptor to phosphatase recruitment (RG16) and phosphatase activity (R95) that consumes the 5'-phosphate, decreasing PIP3 levels. This is an example of an incoherent feed-forward loop, where converging paths have opposing effects. Edges labeled 6 indicate a positive feedback loop involving PI3K activity (R153) and Gab2 recruitment via PIP3 (R146) and Gab2 phosphorylation (RG15).

2.6.5 Comparison of Visualization Size and Complexity

Following Ghoniem et al. [75], the readability of graphs decays with graph size and edge density. Here, we took eight rule-based models of varying sizes and generated all possible automated visualizations so that they can be compared with these metrics. The chart in Figure 2-40 summarizes these results.

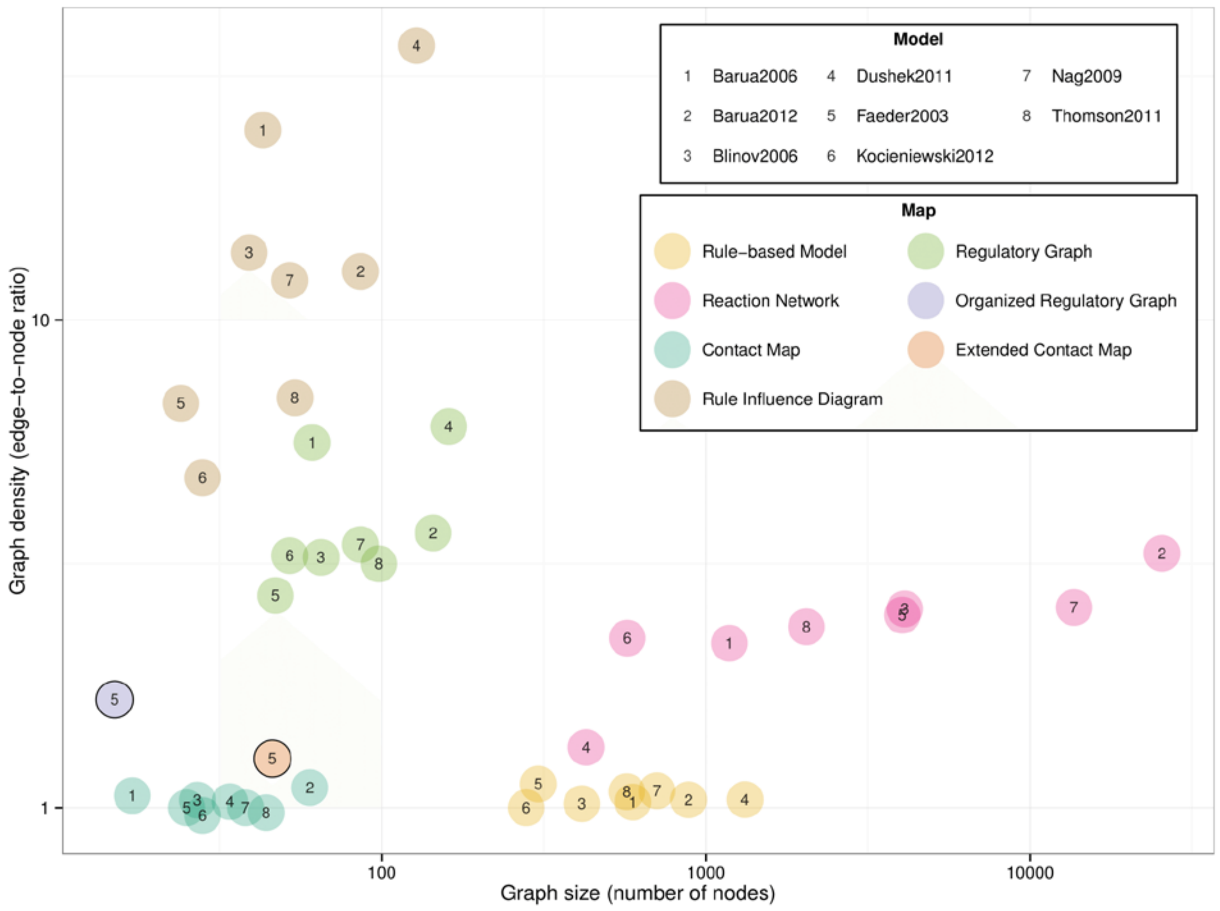


Figure 2-40 Analysis of graph size and complexity for different visualizations. We have used eight rule-based models of various sizes from the literature [35], [37], [42], [47], [76], [78]–[80], and compared the different automated visualizations. X-axis shows size of the graph as number of nodes. Y-axis shows density of the graph as number of edges per node.

In Figure 2-40, note that the contact maps are always the most compact, because they only show structural relationships. Reaction networks can be many times larger and denser than the corresponding rule-based models. The rule influence diagram usually has fewer nodes than regulatory graph, but can have an extremely large number of edges. The full regulatory graph is moderately larger than the rule influence diagram, but with much lower visual complexity. Also shown for the Faeder et al. model (marked 5 in Figure 2-40), are the extended contact map and the organized regulatory graph (i.e. after pruning, grouping and collapsing). The extended contact map is fully manual and is therefore very compactly designed. However, the organized regulatory graph which uses minimal user input to compress the regulatory graph is only marginally larger, and has similar number of edges per node.

2.7 CONCLUDING REMARKS

The work in this chapter was geared towards solving two important visualization problems for rule-based models: visualizing individual rules to understand the modeled mechanisms, and visualizing sets of rules to understand the underlying regulatory network. As shown in the case studies, the regulatory graph can be used to extract and visualize information from a BioNetGen model specification [25], [35] and external input in the form of groups of sites can be systematically used to generate a compressed representation. The abstractions provided are also general enough to be applicable for other rule-based frameworks such as Kappa [68] and Simmune [30], and the development of a common standard (SBML-multi, sbml.org) will open up a wide range of models that can be visualized in this way. Also, unlike sets of rules, which can only be displayed as a list, the regulatory graph data structure is amenable to interactive display and exploration. We expect this to be useful for the increasingly large number and size of rule-based models that is anticipated in the immediate future. In fact, there is already significant movement in the broader modeling community towards collaborative and comprehensive models, such as the whole cell model of *Mycoplasma genitalium* [48]. There are many directions in which the regulatory graph can be extended, for example, node attributes can be paired with numeric values to represent simulation fluxes [81], and a hierarchical multi-layered grouping strategy could be useful for organizing very large model graphs [82]. A more immediate direction to encourage wider adoption is to synthesize automated SBGN diagrams from the regulatory graph [60].

3.0 ENERGY-BASED MODELS AND NETWORK-FREE SIMULATION

3.1 SYNOPSIS

Network-free simulation and energy-based rule-based modeling are recent advances in rule-based modeling. Network-free simulation involves simulating a chemical system without having to generate the reaction network. This is accomplished by mapping patterns in a rule-based model directly to lists of agents in the simulation, calculating the reaction probabilities and firing reaction events. Energy-based rule-based modeling involves specifying a model such the free energy of a reaction can be computed by counting matches of “energy patterns” into reactant and product. In contrast to the classical rule-based specification which does not constrain the energetics of reactions, the energy-based specification ensures detailed balance is always satisfied leading to thermodynamically correct models. Also, it enables kinetics to be specified in terms of cooperativity parameters. Currently, it is not possible to perform a network-free simulation of an energy-based rule-based model. This is because calculating the rate of a reaction from its free energy requires knowing both reactants and products of the reaction event. While this can be accomplished very easily in a network-based context, this information is not available to a network-free simulation until after the reaction event has fired. I explain the problem using a simple example in **Section 3.2**.

In **Section 3.3**, I synthesize a common framework that brings together the rule-based specification, the reaction network, network generation from a rule-based model, the energy-based specification, and the network-based and network-free forms of simulation. The material for this section summarizes methods found in the rule-based literature [24], [34], [39], [50]. The casual reader is advised to skip this section.

In **Section 3.4,** I provide a procedure to synthesize a set of rules compatible with network-free simulation by expanding the energy rules in an energy-based rule-based specification. Then I demonstrate the method using two worked out examples.

3.2 MOTIVATING EXAMPLE

3.2.1 Energy-based Model

Consider the following model, where a molecule A has two states A0 and Ap and binds molecule B under both conditions:

$$\begin{tikzcd}
A_0 \arrow[r, shift left, "k_{fP1}"] \arrow[d, shift right, "k_{fAB1}"'] & A_p \arrow[l, shift left, "k_{rP1}"] \arrow[d, shift right, "k_{fAB2}"'] \\
A_0B \arrow[u, shift right, "k_{rAB1}"'] \arrow[r, shift left, "k_{fP2}"] & A_pB \arrow[l, shift left, "k_{rP2}"] \arrow[u, shift right, "k_{rAB2}"']
\end{tikzcd}$$

Rephrasing the model in terms of equilibrium constants, let

$$K_{P1} = \frac{k_{fP1}}{k_{rP1}}, K_{P2} = \frac{k_{fP2}}{k_{rP2}}$$

$$K_{AB1} = \frac{k_{fAB1}}{k_{rAB1}}, K_{AB2} = \frac{k_{fAB2}}{k_{rAB2}}$$

The model becomes

$$\begin{tikzcd}
A_0 \arrow[r, leftrightarrow, "K_{P1}"] \arrow[d, leftrightarrow, "K_{AB1}"] & A_p \arrow[d, leftrightarrow, "K_{AB2}"] \\
A_0B \arrow[r, leftrightarrow, "K_{P2}"] & A_pB
\end{tikzcd}$$

Each equilibrium constant can be related to the Gibbs free energy of the reaction thus:

$$K_{AB1} = \exp\left(-\frac{G_{AB1}}{RT}\right), K_{AB2} = \exp\left(-\frac{G_{AB2}}{RT}\right)$$

$$K_{P1} = \exp\left(-\frac{G_{P1}}{RT}\right), K_{P2} = \exp\left(-\frac{G_{P2}}{RT}\right)$$

By the principle of detailed balance, the sum of the free energies around the loop should be zero, i.e.

$$G_{AB1} + G_{P2} - G_{AB2} - G_{P1} = 0$$

This places constraints on the equilibrium constants:

$$K_{AB1}K_{P2}\left(\frac{1}{K_{AB2}K_{P1}}\right) = 1 \Longrightarrow \frac{K_{P2}}{K_{P1}} = \frac{K_{AB2}}{K_{AB1}}$$

To simplify the notation, let $\alpha = \frac{K_{P2}}{K_{P1}} = \frac{K_{AB2}}{K_{AB1}}$ and $K_P = K_{P1} = \frac{k_{fP}}{k_{rP}}, K_{AB} = K_{AB1} = \frac{k_{fAB}}{k_{rAB}}$. Then,

$$\begin{tikzcd}
A_0 \arrow[r, leftrightarrow, "K_P"] \arrow[d, leftrightarrow, "K_{AB}"] & A_p \arrow[d, leftrightarrow, "\alpha K_{AB}"] \\
A_0B \arrow[r, leftrightarrow, "\alpha K_P"] & A_pB
\end{tikzcd}$$

Now, consider the problem of representing this system in BioNetGen using the molecule types A(b,t~0~P) and B(a), where t~0 and t~P represent the two states of A.

If we assume that state change of A is independent of binding to B and vice versa, then $\alpha = 1$, and we only need to use two reversible rules to model the system:

```
A(b) + B(a) <-> A(b!1).B(a!1)         kfAB,krAB


A(t~0) <-> A(t~P)      kfP,krP
```

This is because the independence between the two processes enables each process to be modeled separately.

On the other hand, if $\alpha \neq 1$, and if we assume that $\frac{\alpha_1}{\alpha_2} = \frac{\alpha_3}{\alpha_4} = \alpha$, then we need four reversible rules to model the system:

```
A(b,t~0) + B(a) <-> A(b!1,t~0).B(a!1)          kfAB,krAB

A(b,t~P) + B(a) <-> A(b!1,t~P).B(a!1)          alpha1*kfAB,alpha2*krAB

A(t~0,b) <-> A(t~P,b)                          kfP,krP

A(t~0,b!1).B(a!1) <-> A(t~P,b!1).B(a!1)        alpha3*kfP,alpha4*krP
```

This is because the cooperativity between the two processes results in four unique reversible reaction classes instead of two.

In the classical rule-based modeling framework, if there was cooperativity between binding and state change, the modeler would have to specify four rules, as well as specify the relationships between $\alpha_1, \alpha_2, \alpha_3, \alpha_4$. This was a source of error, because an inappropriate definition would break detailed balance. Also, the number of reaction classes was dependent on the number of cooperativities present in the system, making the model hard to edit by simply adding and removing cooperativity terms.

Energy-based rule-based modeling was developed so that cooperativity terms can be defined in terms of pattern matches and free energy accounting can be performed automatically. The modeler only has to specify the minimum number of reaction rules with the most essential context, i.e.

```
A(b) + B(a) <-> A(b!1).B(a!1)

A(t~0) <-> A(t~P)
```

The kinetics is specified in terms of energy pattern matches. So, in the case where there is no cooperativity between binding and state change, the modeler would specify

```
begin energy patterns

        A(t~P)              G_P

        A(b!1).B(a!1)       G_AB

end energy patterns
```

In the case where there is cooperativity, the modeler would specify

```
begin energy patterns

        A(t~P)              G_P

        A(b!1).B(a!1)       G_AB

        A(b!1,t~P).B(a!1)   G_alpha

end energy patterns
```

In both cases, all four reactions are generated automatically from the two rules:

```
A(b,t~0) + B(a) <-> A(b!1,t~0).B(a!1)

A(b,t~P) + B(a) <-> A(b!1,t~P).B(a!1)

A(t~0,b) <-> A(t~P,b)

A(t~0,b!1).B(a!1) <-> A(t~P,b!1).B(a!1)
```

The energies of each reaction are computed depending on the energy pattern definition used. For the first case (no cooperativity), the energies are calculated as:

$$\begin{tikzcd} A_0 \arrow[r, leftrightarrow, "G_P"] \arrow[d, leftrightarrow, "G_{AB}"] & A_p \arrow[d, leftrightarrow, "G_{AB}"] \\ A_0B \arrow[r, leftrightarrow, "G_P"] & A_pB \end{tikzcd}$$

In the second case (cooperativity), the energies are calculated as:

$$\begin{tikzcd} A_0 \arrow[r, leftrightarrow, "G_P"] \arrow[d, leftrightarrow, "G_{AB}"] & A_p \arrow[d, leftrightarrow, "G_{AB} + G_\alpha"] \\ A_0B \arrow[r, leftrightarrow, "G_P+G_\alpha"] & A_pB \end{tikzcd}$$

Thus, in the energy-based specification, the model does not have to be specified in terms of reaction classes with unique kinetics, and when the reaction network is generated, the kinetic specification is calculated automatically from the calculated free energies of reactants and products.

3.2.2 Network-free Simulation

Consider a system with the binding reaction rule (with rate constant k):

```
A(b) + B(a) <-> A(b!1).B(a!1)          k
```

Now suppose we instantiate a simulation system with three particles, Particle1 having structure A(b,t~0), Particle2 having structure A(b,t~P) and Particle3 having structure B(a).

The system has two underlying reactions:

```
Rxn1:  A(b,t~0) + B(a) <-> A(b!1,t~0).B(a!1)        k

Rxn2:  A(b,t~P) + B(a) <-> A(b!1,t~P).B(a!1)        k
```

If we know the underlying form of the species and reactions, then we can use simple labels to refer to the species: A0, Ap, to refer to the states of A, B to refer to the molecule B, and A0-B and Ap-

B to refer to the two A-B complexes respectively. We can also represent the simulation system of three particles as a set of populations of these species:

{ A0 = 1, Ap = 1, B = 1, A0-B = 0, and Ap-B = 0 }

The relative propensity of each reaction is calculated by the product of the rate constant with the populations of the reactant species:

$$rate(Rxn1) = k * A_0 * B = k(1)(1) = k$$

$$rate(Rxn2) = k * A_p * B = k(1)(1) = k$$

The time of the next reaction event depends on the sum of the rates, i.e. $2k$. Which reaction is selected to fire depends on their relative propensity which is identically $\frac{k}{2k} = \frac{1}{2}$. Say Rxn1 fires, then the species populations are updated:

{ A0 = 0, Ap = 1, B = 0, A0-B = 1, and Ap-B = 0 }

Therefore, as long as the identity of species and reactions are known, the propensities of each reaction can be calculated explicitly and the reaction events can be sampled.

However, it is possible to simulate this system even without knowing the species and reactions, i.e. as a network-free simulation. Given a system of three particles as above, Particle1 having A(b,t~0), Particle2 having structure A(b,t~P) and Particle3 having structure B(a), and given a single reaction rule with rate constant k

```
Rule1: A(b) + B(a) <-> A(b!1).B(a!1)          k
```

We keep track of matches of patterns to particles:

```
A(b) – Particle1, Particle2

B(a) – Particle3
```

Then we can use the size of these lists to calculate the propensity of the reaction class

$$rate(Rule1) = k * |A(b)| * |B(a)| = k(2)(1) = 2k$$

So, although we used a single reaction class in this case rather than the two reactions above, the time of the next reaction event will be sampled correctly, i.e. with rate $2k$. When the event fires, particles are selected randomly from the lists of reactant patterns, i.e. Particle1 or Particle2 will be selected with equal probability from the list matching $A(b)$. Therefore, the network-free simulation is considered exactly equivalent to the network-based simulation, as long as the propensities can be calculated from the current set of particles.

Say Particle1 is selected for $A(b)$ and Particle3 for $B(a)$, then the firing of the reaction event results in the forming of a bond between the b component in Particle1 and the a component in Particle3. After the reaction event has fired, we can identify the effect of the reaction: Particle1 had structure A(b,t~0), Particle3 had B(a), so the resultant must be a particle with structure A(b!1,t~0).B(a!1), which can be called Particle4.

3.2.3 Network-free Simulation with Energy-based Rules

Note that in the energy-based specification, the identity of both reactant and product species was essential to compute the free-energy of the reaction and hence the rate. This posed no problems in the network-based simulation because all possible reactions and species are known prior to instantiating the simulation system. However, in the network-free simulation, the rate needs to be computed only from the current set of particles, and the identity and structure of the product species can be known only after the reaction event has fired. Thus, energy rules cannot be used as reaction classes in a network-free simulation. The work in this chapter was geared towards synthesizing a set of rules that are equivalent to the energy-based specification, but can be simulated in a network-free manner.

3.3 BIONETGEN THEORY FOR MODEL SPECIFICATION AND SIMULATION

In this section, I define formalisms that summarize and unify the specification and simulation of models in the rule-based framework [24], [34], [39], [50], and establish the basis for energy-based network-free simulation. In Sections 3.3.1, 3.3.2 and 3.3.3, I establish the basic concepts of embeddings involving patterns and reaction rules. In Section 3.3.4, I define rule-based models and reaction networks, and simulation systems that can be equivalently set up under both frameworks. Section 3.3.5 summarizes the conversion of a rule-based model into a reaction network, and Section 3.3.6 discusses stochastic simulation using network-based methods. Sections 3.3.6, 3.3.7 and 3.3.8 address the recent advance of energy-based rule-based modeling and simulation of energy-based models using network-based methods. Corollary 3.3-43 is a novel result that enables the work in **Section 3.4.** Section 3.3.9 discusses network-free simulation for the classical rule-based model.

3.3.1 Patterns

Definition 3.3-1

A **molecule type definition** is a set of molecule names M, a set of component names C, a set of internal state labels S, stoichiometry constraints for components in molecules $N_{MC}: M \times C \rightarrow \mathbb{N}$, and internal state label constraints for components $N_{MCS}: M \times C \times S \rightarrow \{0,1\}$.

Definition 3.3-2

A **pattern** is a graph $p \coloneqq (V, E, name, type)$ with a set of nodes V, a set of undirected edges $E \subset V \times V$, and node labeling functions $name, type$ where $type: V \rightarrow \{mol, comp, is, bs\}$ indicates whether a node is molecule (mol), component (comp), internal state (is) or bond state (bs), and

$name\colon V \to A^* \cup \{!+, !-, !?, \emptyset\}$, where A^* is the set of alphanumeric labels allowed by BioNetGen, $\{!+, !-, !?\}$ are labels indicating bound, unbound, and unspecified bond labels respectively, and $\emptyset$ indicates default internal state label respectively. Additional restrictions on the graph are: (i) a component must be adjacent to a single molecule, a single internal state and a single bond state, (ii) an internal state or a bond state with labels $\{!-, !?\}$ must be adjacent to a single component, (iii) a bond state with label !+ must be adjacent to one or two components. Given a molecule type definition $(M, C, S, N_{MC}, N_{MCS})$, additional model-specific restrictions are: if n_{mc} denotes the number of edges in E whose node pairs are a molecule named m, and a component named c, and if n_{mcs} denotes the number of connected 3-node subgraphs in (V, E) with a molecule named m, a component named c, and an internal state named s:

$$\forall m \in M, c \in C, n_{mc} \leq N_{MC}\big((m, c)\big)$$

$$\forall m \in M, c \in C, s \in S, n_{mcs} = N_{MCS}\big((m, c, s)\big)$$

A pattern has a **canonical label**, i.e. a unique label that can be generated by ordering molecules, components and bonds.

Definition 3.3-3

A **stream** is a path $v = (v_1, v_2 \dots v_n)$ on a pattern $p = (V, E \dots)$ such that the $type$ attribute satisfies the descending order $mol > comp > is, bs$.

Definition 3.3-4

A **local view** of a node is the subgraph composed from all streams passing through the node in the pattern.

Definition 3.3-5

A **pattern tuple** is a tuple of patterns. Given a pattern tuple, there also exists a **pattern tuple graph** formed by a trivial merge. We will refer to pattern tuples and pattern tuple graphs interchangeably and use the same notation (p).

3.3.2 Pattern Embeddings

Definition 3.3-6

A pattern $p = (V_p, E_p, \ldots)$ **embeds** in a pattern $q = (V_q, E_q, \ldots)$, denoted $p \lhd q$, if there exists a total injective map $\phi_{p,q} = V_p \to V_q$ that preserves name and type attributes and edge relationships. If a number of embeddings exist, then this is denoted by the set $\{\phi_{p,q}\}$. A pattern tuple has a **canonical order**, which can be achieved by ordering the canonical labels of the individual patterns. All pattern tuples considered hereafter will be assumed to be in their canonical order.

Definition 3.3-7

A pattern tuple (p) **embeds** in a pattern tuple (q), denoted $(p) \lhd (q)$, if there exists an injective map pairing every pattern p in (p) with some pattern q in (q), and there exists an embedding $\phi_{p,q}$ for every pair (p, q). The tuple of embeddings $(\phi_{p,q})$ that define a pattern tuple embedding is denoted $\Phi_{(p),(q)}$, and if a number of such embeddings can exist, then the set of pattern tuple embeddings is denoted $\{\Phi_{(p),(q)}\}$. Nominally, a pattern tuple embedding can be treated as a simple embedding between the corresponding pattern tuple graphs.

Definition 3.3-8

A **restricted embedding** $\phi_{p,q}|V$ is derived from an embedding between patterns $\phi_{p,q}: V_p \to V_q$, by restricting the domain to the subset $V \cap V_p$ and the image to the subset $V \cap V_q$. By extension, a restricted embedding can also be derived from a pattern tuple embedding and is denoted $\Phi_{(p),(q)}|V$.

Corollary 3.3-9

Embeddings are transitive, because the composition of two injections is injective.

$$p \lhd q, q \lhd r \implies \exists \phi_{p,q}, \phi_{q,r} \implies \exists \phi_{p,r} | \phi_{p,r} = \phi_{p,q} \circ \phi_{q,r} \implies p \lhd r.$$

Definition 3.3-10

A **correspondence map** between patterns or pattern tuples is a generalization of the embedding by allowing it to be partial in the domain, and allowing components to map without requiring a map between adjacent binding or internal states. A correspondence map is valid only if merging correspondent nodes still preserves the stoichiometry constraints outlined in the molecule type definition and pattern definition.

Definition 3.3-11

A pattern p, is **isomorphic** to another pattern q, denoted $p \sim q$, if $p \lhd q$ and $q \lhd p$.

Definition 3.3-12

A pattern tuple (p), is **isomorphic** to another pattern tuple (q), denoted $(p) \sim (q)$, if $(p) \lhd (q)$ and $(q) \lhd (p)$.

Definition 3.3-13

An embedding between two identical pattern tuples is called an **automorphism**, denoted $\Phi_{(p),(p)}$.

Definition 3.3-14

The **trivial automorphism** is a special case of the automorphism, where every node maps to itself, denoted $\Gamma_{p,p}$ for patterns and $\Gamma_{(p),(p)}$ for pattern tuples.

3.3.3 Reaction Rules

Definition 3.3-15

The **reaction rule** $r \coloneqq \left(R, P, \Psi_{R,P}\right)$ is defined by a tuple of reactant patterns R, a tuple of product patterns P, and a correspondence map between them $\Psi_{R,P}$ (with the additional restriction that there is no unmatched component in R). BioNetGen can compute the correspondence map heuristically, so it is sufficient to define a rule as $r \coloneqq (R, P)$. The rule as defined here excludes the BioNetGen features such as include/exclude reactants, include/exclude products, rules on dot-connected patterns, and species deletion.

Definition 3.3-16

The **site of action** of a rule is the subset of nodes not in the correspondence map.

Definition 3.3-17

The **reaction center** of a rule is the union of local views of the site of action.

Definition 3.3-18

The **reaction context** is the union of local views of the complement of the reaction center.

Definition 3.3-19

A reaction rule $r = (R, P, \Psi_{R,P})$ **embeds** in another reaction rule $r' = (R', P', \Psi_{R',P'})$, denoted $r \lhd r'$, if $R \lhd R', P \lhd P'$ and the following loop commutes:

$$\begin{tikzcd} R \arrow[r, "\Psi_{R,P}"] \arrow[d, "\Phi_{R,R'}"'] & P \arrow[d, "\Phi_{P,P'}"] \\ R' \arrow[r, "\Psi_{R',P'}"] & P' \end{tikzcd}$$

Definition 3.3-20

The **restriction to the reaction center** $r|RC$ of a reaction rule $r = (R, P, \Psi_{R,P})$ is given by $r|RC = (R|RC, P|RC, \Psi_{R|RC,P|RC})$ where $R|RC, P|RC$ are restrictions of the reactant and product patterns to the reaction center.

Corollary 3.3-21

$r|RC \lhd r$ because the correspondence maps $\Psi_{R,P}, \Psi_{R|RC,P|RC}$ and self-embeddings $\Gamma_{R,R|RC}, \Gamma_{P,P|RC}$ commute.

Definition 3.3-22

Given pattern or pattern tuple graphs A and B, a correspondence map $\Psi_{A,B}$, V denoting some set of nodes, and E denoting some set of vertex pairs, we define the following graph operations

$DN(V) \circ A$	Removes the nodes V and all downstream nodes from A.
$RN(\Psi_{A,B}) \circ A$	Replaces each node in A with its image in B, if such an image exists.
$AN(V) \circ A$	Adds nodes V to A.
$AE(E) \circ A$	Adds edges E between pre-existing pairs of nodes in A.

Definition 3.3-23

A **reaction rule** $r = (R, P, \Psi_{R,P})$ can also be represented as a **graph rewriting** $R \overset{r}{\Rightarrow} P$, which is equivalent to the following sequence of operations on the pattern tuple graph R:

$$P = AE(E_{img^C}) \circ AN(img^C) \circ RN(\Psi_{R,P}) \circ DN(dom^C) \circ R$$

where (i) dom, dom^C are partitions induced by the partial map $\Psi_{R,P}$ in R and P respectively such that the map is total in $dom \rightarrow img$, (ii) $E_{img^C} \subset E_P$ is the subset of edges on the product pattern tuple graph P with at least one node in img^C.

Definition 3.3-24

The **action of a rule** $R \overset{r}{\Rightarrow} P$ on pattern tuple R', given a pattern tuple embedding $\Phi_{R,R'}$, is given by

$$P' = r \circ RN(\Phi_{R,R'}^{-1}) \circ R'$$

The action of a rule is denoted as $P' = r[R', \Phi_{R,R'}]$.

Given the action of a rule on pattern tuple R', there exists an $\Phi_{P,P'}$ that commutes as follows:

$$\begin{tikzcd} R \arrow[r, Rightarrow, "r"] \arrow[d, "\Phi_{R,R'}"'] & P \arrow[d, "\Phi_{P,P'}"] \\ R' \arrow[r, Rightarrow, "{r[R',\Phi_{R,R'}]}"] & P' \end{tikzcd}$$

Definition 3.3-25

A **reversible rule** is a pair of rules (r, r_{rev}) with complementary reactant and product pattern tuples, i.e. if $r = (R, P)$, then $r_{rev} = (P, R)$.

Definition 3.3-26

The **statistical factor** ρ_r of the reaction rule $r = \left(R, P, \Psi_{R,P}\right)$ is a function of how many symmetries are present on reactant side of the reaction center and how many of them are broken when they are converted to products. By convention, BioNetGen uses three terms to calculate the statistical factor. The rule group term RG counts how many automorphisms in R induce an automorphism in P under $\Psi_{R,P}$. The reaction center stabilizer term $Stab$ counts how many of the induced automorphisms are identity morphisms on the reaction center. The context reactant graph terms CRG counts permutations on reactant patterns that do not form part of the reaction center.

$$RG = \left\{\phi \in \Phi_{R,R}, \Psi_{R,P} \circ \phi \circ \Psi_{R,P}^{-1} \in \Phi_{P,P}\right\}$$

$$Stab = \{\phi \in RG, \phi(x) = x \;\forall\; x \in R|RC\}$$

$$R_{con} = \{p \in R, p \notin dom_{\Psi}, p \nsim q \forall q \in dom_{\Psi}\}, R_{con}^{\sim} = \{q |\; q \sim p, [\in R_{con}\}$$

$$|CRG| = Perm(R_{con}, R_{con}) = \prod_{q \in R_{con}^{\sim}} |\{p \sim q, p \in R_{con}\}|!$$

$$\rho = \frac{1}{(|RG|/|Stab|) * |CRG|}$$

3.3.4 Ensembles, Models and Rate Constants

Definition 3.3-27

A **complex** x, also known as a **particle**, is an instance of a pattern, as defined in Definition 3.3-2, with additional constraints: (i) no bond state or internal state is unspecified, (ii) $n_{mc} = N_{MC}\left((m, c)\right) \forall m, c$, (iii) all bonds named '!+' are adjacent to two components, (iv) all components are adjacent to exactly one bond state and one internal state. A particle is analogous to a freely diffusing chemical entity in a simulation system.

Definition 3.3-28

The **particle ensemble,** denoted $Ens = \{x\}$, is a set containing complex instances (defined in Definition 3.3-27).

Definition 3.3-29

The **pattern embedding class in an ensemble**, denoted $\mathbb{P}_p^{Ens}$ is the set of embeddings from pattern p to a particle ensemble Ens, i.e. $\mathbb{P}_p^{Ens} = \{\phi_{p,x}, x \in Ens\}$.

Definition 3.3-30

The **species**, denoted s, is a pattern that is isomorphic to particles.

Definition 3.3-31

The **species observable class in an ensemble**, denoted $\mathbb{O}_s^{Ens}$ is the set of particles in a particle ensemble isomorphic to species s, i.e. $\mathbb{O}_s^{Ens} = \{s \sim x, s \in Ens\}$.

Definition 3.3-32

The **species space**, denoted $S = \{s\}$, is a set of unique species. For a model, S denotes the space of all possible species.

Definition 3.3-33

The **species ensemble**, denoted $Sp(Ens)$, is a set of species mapped to population counters and can be used to represent the particle ensemble Ens, i.e. $Sp(Ens) = \{(s, n_s), s \in S, n_S = |\mathbb{O}_S^{Ens}|\}$.

Definition 3.3-34

The **reaction** $\mu = (R_s, P_s)$ is a transformation from a reactant tuple of species R_s to a product tuple of species P_s. Similar to those for a reaction rule, the following can be defined for a reaction also: a correspondence map computable in BioNetGen (Definition 3.3-15), the site of action (Definition

3.3-16), the restriction to the reaction center (Definition 3.3-20), a reformulation using graph rewriting (Definition 3.3-23), and reversible reactions (Definition 3.3-25).

Definition 3.3-35

The reaction $\mu = (R_s, P_s)$ can also be equivalently defined as a set if tuples $\mu = \{(s, n_s^{R,\mu}, n_s^{P,\mu})\}$, where $n_s^{R,\mu}$ and $n_s^{P,\mu}$ represent stoichiometries of species s in R_s and P_s respectively from Definition 3.3-34.

Definition 3.3-36

The **reaction rule rate constant** k_r is a function describing the kinetics of the reaction class mapped by the rule r. The rate law can be decomposed as follows:

$$k_r = \rho_r * k_r^{const} * k_r^{Ens}(Ens) * k_r^{\mu}(r \lhd \mu)$$

Where ρ_r is the statistical factor computed according to Definition 3.3-26, k_r^{const} is a function of user-defined variables, k_r^{Ens} is a function of the current simulation ensemble, and k_r^{μ} is a function of the map from the rule to the reaction or reaction instance. k_r^{Ens} is composed of **global functions** which compute sums of species or particles in the ensemble. In current BioNetGen, k_r^{μ} is restricted to be a function of individual **local functions** specific to reactant species, i.e.

$$k_r^{\mu}(r \lhd \mu) = fn(k_1(p_1), k_2(p_2), \ldots), \forall p_i \in R_\mu$$

We shall therefore denote $k_r^{\mu}(r \lhd \mu)$, as a function of the reactant species $k_r^{\mu}(R_\mu)$.

Definition 3.3-37

The **rule-based model**, $Model \coloneqq \{MolDef, \{r\}, Ens, \{r\} \to \{k(r, Ens)\}\}$, is a molecule type definition $MolDef$, a set of rules $\{r\}$, a particle ensemble Ens, and a reaction rule rate constant k_r mapped to each rule.

Definition 3.3-38

A **reaction network**, $Rxnnet \coloneqq \{S, \{\mu\}, \{\mu\} \rightarrow \{k_\mu\}, Sp(Ens)\}$, is a set of possible species S, a set of reactions $\{\mu\}$, a function mapping each reaction μ to a symmetry-independent rate constant k_μ, and a species ensemble $Sp(Ens)$. Note that the equivalent particle ensemble Ens does not need to be specified, since it is sufficient to distinguish species from each other, without needing to distinguish particles.

3.3.5 Network Generation from a Rule-based Model

Network generation is a procedure to generate a reaction network (Definition 3.3-38) from a rule-based model (Definition 3.3-37).

1) Start with a rule-based model rule-based model $Model \coloneqq \{MolDef, \{r\}, Ens, \{r\} \rightarrow \{k_r\}\}$, $\{r\}$ being a set of rules $\{k_r\}$ being a set of rate laws, Ens a particle ensemble, and $Sp(Ens)$ the equivalent species ensemble.
2) Let P^* be the set of all patterns from the reactants and products of all rules. Also define empty sets $S_{new}, S_{current}$ and for each rule r, empty sets Emb_r, Emb_r^{new}.
3) $S_{new} \leftarrow S$
4) **Mapping:** $\forall p \in P^*, s \in S_{new}$ Compute $\Phi_{p,s} = \{\phi_{p,s}\}$.
5) $\Phi^* \leftarrow \Phi^* \cup \Phi_{p,s} \forall p \in P^*, s \in S_{new}$
6) $S_{current} \leftarrow S_{new}, S_{new} \leftarrow \{ \quad \}$
7) For each reaction rule $r = (R, P)$,
    a) For each tuple $R_s = (s)$ drawn with repeats from $S_{current}$ and not canonically ordered,
        i) Compute $\{\Phi_{R,R_s}\}$ where $\Phi_{R,R_s} = (\phi)$, where $\phi \in \Phi^*$
        ii) For each $\Phi_r^i \in \{\Phi_{R,R_s}\}, \Phi_r^i \notin Emb_r$,

(1) compute $\mu_r^i = (R_s, P_s)$, where i indexes the set of maps, r indicates the rule generating the reaction, and $P_s = r\left[R_s, \Phi^i\right]$.

(2) Perform additional checks on whether (R_s, P_s) is a satisfactory reaction.

(3) $S_{new} \leftarrow S_{new} \cup \{S_{current} - P_s\}, Emb_r^{new} \leftarrow \left\{\Phi_r^i\right\}$

iii) $\mathrm{M} \leftarrow \mathrm{M} \cup \left\{\mu_r^i\right\}$

8) If $|S_{new}| > 0$ or $\exists r, |Emb_r^{new}| > 0$

a) $S \leftarrow S \cup S_{new}$

b) $\forall r, Emb_r \leftarrow Emb_r \cup Emb_r^{new}$

c) Goto Step 3

9) **Lumping:** Let $M^{\sim} = \{\mu \sim \mu' \forall \mu' \in M\}$. For each $\mu \in M^{\sim}$, let $M_{\mu,r}$ be the subset of M isomorphic to μ generated from reaction rule r. The rate constant of the reaction is computed as follows:

$$k_\mu = \sum_{\{r\}} \sum_{M_{\mu,r}} k_r$$

Here, k_r is the rate law function of the rule. From Definition 3.3-36,

$$k_r = \rho_r * k_r^{const} * k_r^{Ens}(Ens) * k_r^{\mu}\left(R_\mu\right)$$

Here, the statistical factor ρ_r, the function of user-defined variables k_r^{const} are available from the rule-based specification. $k_r^{Ens}(Ens)$ is translated into a function of **global functions** in the species space, i.e. some $fn\big(Sp(Ens)\big)$. $k_r^{\mu}\left(R_\mu\right)$ is evaluated concretely since the form of the reaction μ is now available. Together, k_μ is a function of numeric constants and the species ensemble.

10) The reaction network is given by $Rxnnet \coloneqq \left\{S, \{\mu\}, \{\mu\} \rightarrow \left\{k_\mu\right\}, Sp(Ens)\right\}$

The network generation process will be denoted $NetGen(Model)$ for the general case presented here. When a specific set of seed species and rules are used to expand the network, without reference to a species ensemble or global or local functions, it will be denoted $NetGen(SeedSp, Rules)$.

3.3.6 Network-based Stochastic Simulation

Consider a reaction network is generated from a rule-based model as in Section 3.3.5. Using the form of the reaction provided in Definition 3.3-35, each reaction μ is a set of tuples $\left\{\left(s, n_s^{R_\mu}, n_s^{P_\mu}\right)\right\}$. Given a species ensemble $Sp(Ens) = \{(s, n_s), s \in S, n_s \in \mathbb{N}\}$, the rate a_μ of a reaction $\mu = (R_s, P_s)$ is computed as:

$$a_\mu = k_\mu \prod_{s \in R_\mu} (n_s)^\wedge(n_s^{R_\mu})$$

From the next reaction method of stochastic simulation [7], the time at which the next reaction fires is inversely proportional to $\Sigma_\mu a_\mu$, and the relative probability of a particular reaction firing is computed as $\frac{a_\mu}{\Sigma_\mu a_\mu}$.

1. Given $Sp(Ens)^{t_0} = \left\{\left(s, n_s^{t_0}\right)\right\}, t \leftarrow t_0,$
2. Compute $a_\mu(t) \forall \mu$ from $Sp(Ens)^t$.
3. Compute Δt from $\Sigma_\mu a_\mu(t)$.
4. Select μ with relative probability $\frac{a_\mu(t)}{\Sigma_\mu a_\mu(t)}$.
5. Update species ensemble: $\forall s, n_s^{t+\Delta t} \leftarrow n_s^t - n_s^{R_\mu} + n_s^{P_\mu}$.
6. Update time. $t \leftarrow t + \Delta t$.

7. If $t < t_{end}$, go to Step 2.

The memory cost of the simulation scales with $\mathcal{O}\left(n_{reactions} + n_{species}\right)$ and is independent of the number of rules. Therefore, even if a small number of rules are sufficient to describe the kinetics of the system, it is possible that the equivalent reaction network is too large to be simulated or possibly even infinite in size.

3.3.7 Energy-based Rule-based Formulation

Definition 3.3-39

An **energy pattern** e, is a pattern which is assigned the numeric energy value G_e.

Definition 3.3-40

An **energy definition** is the tuple (E, G_E), where $E = (e)$ is a tuple of energy patterns, and $G_E\colon E \to \mathbb{N}$ is a function mapping energy patterns to numeric energy values.

Definition 3.3-41

Given an energy definition, the **free energy of formation** of a species s, denoted G_s, is the sum of matches of energy patterns to the species, weighted by the respective energies.

$$G_s = \sum_{e \in E} \left|\{\phi_{e,s}\}\right| G_e$$

Definition 3.3-42

Given an energy definition, the **free energy of a reaction** $\mu = (R^\mu, P^\mu)$, denoted G_μ, is the sum of the free energy of the product species minus the sum of the free energy of the reactant species.

$$G_\mu = -\sum_{s \in R^\mu} G_s + \sum_{s \in P^\mu} G_s$$

From Definition 3.3-41,

$$G_\mu = -\sum_{s\in R^\mu}\sum_{e\in E}\left|\{\phi_{e,s}\}\right|G_e + \sum_{s\in P^\mu}\sum_{e\in E}\left|\{\phi_{e,s}\}\right|G_e$$

$$= \sum_{e\in E}\left(-\sum_{s\in R^\mu}\left|\{\phi_{e,s}\}\right| + \sum_{s\in P^\mu}\left|\{\phi_{e,s}\}\right|\right)G_e$$

Corollary 3.3-43

The free energy of a reaction can be computed from only the energy patterns overlapping with the site of action.

Proof: Let $\Psi_{R,P}$ be the correspondence map of reaction $\mu = (R, P)$, defined according to Definition 3.3-34. Let V_R, V_P be the sets of nodes on the reactant and product tuple graphs respectively. The site of action (Definition 3.3-16) induces a partition V_R^{SA}, V_R^{Ψ} on V_R (V_R^{SA}- nodes in the site of action, V_R^{Ψ}- nodes in the correspondence map), and a similar partition V_P^{SA}, V_P^{Ψ} on V_P. Let $V^\Psi = V_R^\Psi \cup V_P^\Psi$. Now, V^Ψ induces a partition in the set $\{\phi_{e,s}\}$ of embeddings from any energy pattern e to any species s in R or P, where $\{\phi_{e,s}|\Psi\}$ is the set of embeddings completely in V^Ψ and $\{\phi_{e,s}|\mathrm{SA}\}$ is the complement of $\{\phi_{e,s}|\Psi\}$. Substituting for $\{\phi_{e,s}\}$ in Definition 3.3-42,

$$G_\mu = \sum_{e\in E}\left(-\sum_{s\in R^\mu}\left|\{\phi_{e,s}\}\right| + \sum_{s\in P^\mu}\left|\{\phi_{e,s}\}\right|\right)G_e$$

$$= \sum_{e\in E}\left(-\sum_{s\in R^\mu}\left|\{\phi_{e,s}|\Psi\}\cup\{\phi_{e,s}|\mathrm{SA}\}\right| + \sum_{s\in P^\mu}\left|\{\phi_{e,s}|\Psi\}\cup\{\phi_{e,s}|\mathrm{SA}\}\right|\right)G_e$$

Since $\{\phi_{e,s}|\Psi\}\cap\{\phi_{e,s}|\mathrm{SA}\} = \{\}$ by construction,

$$G_\mu = \sum_{e\in E}\left(-\sum_{s\in R^\mu}\left|\{\phi_{e,s}|\Psi\}\right| - \sum_{s\in R^\mu}\left|\{\phi_{e,s}|\mathrm{SA}\}\right| + \sum_{s\in P^\mu}\left|\{\phi_{e,s}|\Psi\}\right| + \sum_{s\in P^\mu}\left|\{\phi_{e,s}|\Psi\}\right|\right)G_e$$

$$= \sum_{e\in E}\left(-\sum_{s\in R^\mu}\left|\{\phi_{e,s}|\Psi\}\right| + \sum_{s\in P^\mu}\left|\{\phi_{e,s}|\Psi\}\right| - \sum_{s\in R^\mu}\left|\{\phi_{e,s}|\mathrm{SA}\}\right| + \sum_{s\in P^\mu}\left|\{\phi_{e,s}|\mathrm{SA}\}\right|\right)G_e$$

By construction, $\Psi_{R,P}$ is total and invertible in $V_{\mathrm{R}}^{\Psi} \to V_{P}^{\Psi}$, any embedding completely in V_{R}^{Ψ} has an equivalent embedding in V_{P}^{Ψ} and *vice versa*, so $\forall e, \sum_{s \in R^{\mu}} \left|\{\phi_{e,s}|\Psi\}\right| = \sum_{s \in P^{\mu}} \left|\{\phi_{e,s}|\Psi\}\right|$.

Substituting in the equation for G_{μ} above,

$$G_{\mu} = \sum_{e \in E} \left(-\sum_{s \in R^{\mu}} \left|\{\phi_{e,s}|\mathrm{SA}\}\right| + \sum_{s \in P^{\mu}} \left|\{\phi_{e,s}|\mathrm{SA}\}\right| \right) G_{e}$$

By construction, $\{\phi_{e,s}|\mathrm{SA}\}$ is the set of embeddings that involve at least one node in the site of action. Hence proved.

Definition 3.3-44

Given a reversible reaction (μ_{+}, μ_{-}) where $\mu_{+} = (R, P), \mu_{-} = (P, R)$, free energy of formation G_{μ} in the forward direction, activation energy G_{μ}^{EA}, and distribution parameter ϕ_{μ}, the forward and reverse rate constants $k_{\mu+}, k_{\mu-}$, termed **energy-based rate constants,** are computed as follows:

$$k_{\mu+} = \exp\left(-\frac{G_{\mu+}}{RT}\right)$$

$$k_{\mu-} = \exp\left(-\frac{G_{\mu-}}{RT}\right)$$

Where $G_{\mu+} = G_{\mu}^{EA} + \phi_{\mu} G_{\mu}$ and $G_{\mu-} = G_{\mu}^{EA} + (\phi_{\mu} - 1) G_{\mu}$

Here, $G_{\mu+}$ and $G_{\mu-}$ are energy terms that describe the kinetics in terms of forward and reverse activation energies, following from linear transition state theory (REF) and Arrhenius equation. The distribution term ϕ_{μ} indicates how the free energy of reaction G_{μ} is distributed in the forward and reverse directions, and the G_{μ}^{EA} term captures all contributions to the kinetics that are independent of G_{μ}. This definition of forward and reverse rate constants also satisfies the definition of the equilibrium constant in terms of the free energy of the reaction.

$$K_{\mu}^{eq} = \frac{k_{\mu+}}{k_{\mu-}}$$

$$= \frac{\exp\left(-\frac{G_{\mu+}}{RT}\right)}{\exp\left(-\frac{G_{\mu-}}{RT}\right)} = \exp\left(-\frac{G_{\mu+} - G_{\mu-}}{RT}\right)$$

$$= \exp\left(-\frac{G_{\mu}^{EA} + \phi_{\mu} G_{\mu} - G_{\mu}^{EA} - (\phi_{\mu} - 1) G_{\mu}}{RT}\right)$$

$$K_{\mu}^{eq} = \exp\left(-\frac{G_{\mu}}{RT}\right)$$

Definition 3.3-45

The **energy-based rule-based model** $eModel \coloneqq \{MolDef, EnergyDef, \{(r_+, r_-)\}, \{(r_+, r_-)\} \rightarrow \{k_{Arr}(G_r^{EA}, \phi_r)\}, Ens\}$, has a molecule type definition $MolDef$, an energy definition $EnergyDef$, a set of reversible rules $\{(r_+, r_-)\}$, each of which is mapped to an Arrhenius rate law requiring an activation energy term G_r^{EA} and a distribution term ϕ_r, and a particle ensemble Ens.

3.3.8 Energy-based Network Generation and Simulation

Since the rules in an energy-based model have the same form as rules in a rule-based model, the network generation described in Section 3.3.5 is applicable for energy-based rule-based models also. However, in Step 6 of Section 3.3.5, after identifying the set of reactions up to isomorphism ($M^{\sim}$), the rate constants are not calculated by lumping. Instead, for each reaction $\mu = (R_s, P_s)$, the free energy of the reaction G_μ is calculated according to Definition 3.3-42, and then the forward and reverse rate constants $k_{\mu+}$and $k_{\mu-}$ are calculated according to the Arrhenius rate law in Definition 3.3-44. Network-based stochastic simulation of the resultant reaction network is the same as outlined in Section 3.3.6.

3.3.9 Network-free Stochastic Simulation

Network-free simulation involves simulating a rule-based model (Definition 3.3-37) without generating the corresponding reaction network. Given a particle ensemble Ens and a set of patterns P^* from the rule-based model, the algorithm keeps track of the pattern embedding class in the ensemble for each pattern, i.e. $\mathbb{P}_p^{Ens}$ (Definition 3.3-29).

The reaction probability a_r is calculated for the entire reaction class defined by the rule r as follows:

$$a_r = k_r^{const} * k_r^{Ens}\left(\left\{\mathbb{P}_p^{Ens}\right\}\right) * \rho_r * \sum_{r \lhd \mu} k_r^{\mu}$$

Each term is explained in Definition 3.3-36. Here μ is the individual reaction instance on particles R_x, rather than a reaction on species. In NFsim [39], the fast network-free simulator for BioNetGen, k_r^{μ} is only allowed to be composed of products of local functions on individual reactant patterns:

$$k_r^{\mu} = \prod_{p \in R, x \in R_x} k_p(x)$$

This allows for efficient simulation, because the sum of products is decomposed as a product of the sums, and there is no need to track individual combinations of pattern matches:

$$\sum_{r \lhd \mu} \prod_{p \in R, x \in R_x} k_p(x) = \prod_{p \in R} \sum_{x \in \mathbb{P}_p^{Ens}} k_p(x)$$

Substituting in the term for local functions, the rate of the reaction class is given by

$$a_r = k_r^{const} * k_r^{Ens}\left(\left\{\mathbb{P}_p^{Ens}\right\}\right) * \rho_r * \prod_{p \in R} \sum_{x \in \mathbb{P}_p^{Ens}} k_p(x)$$

When no local functions are used, the simulation is even more efficient, because the rate calculation resolves to:

$$a_r = k_r^{const} * k_r^{Ens}\left(\left\{\mathbb{P}_p^{Ens}\right\}\right) * \rho_r * \prod_{p \in R} \left|\mathbb{P}_p^{Ens}\right|$$

The time at which the next reaction instance fires is proportional to $\Sigma_r a_r$, and the relative probability of each reaction class being chosen to fire is $\frac{a_r}{\Sigma_r a_r}$. Within each reaction class, a pattern match is selected randomly for each reactant pattern, with the probability weighted by any local function defined on that pattern match.

1. Given $Ens^{t_0} = \{x\}, t \leftarrow t_0$,
2. Compute $a_r(t) \forall r$ from Ens^t.
3. Compute Δt from $\Sigma_r a_r(t)$.
4. Select r with relative probability $\frac{a_r(t)}{\Sigma_r a_r(t)}$.
5. For each $p \in R$, select $\phi_{p,x}$ with relative probability $\frac{k_p(x)}{\Sigma_x k_p(x)}$
6. Determine reaction instance, i.e. the particle tuple $R_x \subset Ens$, and the corresponding pattern tuple embedding Φ_{R,R_x}. Determine $P_x = r\left[R_x, \Phi_{R,R_x}\right]$.
7. Perform checks on whether this is a valid reaction.
8. Update particle ensemble: $Ens^{t+\Delta t} \leftarrow Ens^t - R_x + P_x$.
9. Update time. $t \leftarrow t + \Delta t$.
10. If $t < t_{end}$, go to Step 2.

3.4 ENERGY-BASED NETWORK-FREE SIMULATION

3.4.1 The Problem

The rule-based model (Definition 3.3-37) is a set of reaction rules mapped to rate laws. By expanding each reaction rule into a class of reactions, a reaction network can be constructed which resolves all possible chemical species and all possible reactions between those species (Section 3.3.5). In the energy-based formulation (Section 3.3.7), after building the network, the reactant and product species are analyzed to compute the free energy change of each reaction (Definition 3.3-42), which in turn is used to compute the rate constant for that reaction (Definition 3.3-44). The reaction network can then be used to simulate a species ensemble, with each species is assigned a population number which is tracked across time (Section 3.3.6). Because the identity of reactants and products are known for each reaction are known prior to firing a reaction event, the energy-based formulation works seamlessly with network generation and network-based simulation.

In contrast to network-based simulation, network-free simulation involves simulating a particle ensemble directly using reaction rules (Section 3.3.9). Here, the system keeps track, not of populations of types of particles, but each particle individually, and the embeddings of patterns into particles (Definition 3.3-29). Reaction events are selected by analyzing the current ensemble (i.e. the potential reactants for each reaction event), and the full specification of the product particles are known only after the reaction event fires. This makes simulation very efficient, but places restrictions on the types of rate laws that are possible, for example, rate laws with local functions can only use attributes of the reactants (Definition 3.3-36) and not the products. Since the energy-based rate laws require knowledge of both reactants and products to calculate the reaction rate, the energy-based rule-based model specification cannot be directly simulated using a truly network-free algorithm. These concepts are explained with an example in **Section 3.2.**

Rule refinement is a general procedure where patterns in a rule are expanded by adding molecules, components, internal states and bonds. The refined rule now only matches a subset of the reaction class matched by the original rule because of the additional match conditions. A rule refinement approach typically tries to organize the reaction class matched by a reaction rule into many such subsets, and generate a unique reaction rule for each subset. For example, in network generation (Section 3.3.5), the goal is to resolve reaction rules to reactions, so that each reaction produces and consumes distinct chemical species. Another example is the rule-refinement performed in the hybrid particle-population simulator [34], wherein some particles are treated individually and some are lumped using population counters. The goal of rule refinement here was to resolve reaction classes such that reactant/product patterns matched unlumped or lumped particles, but not both at the same time.

For energy-based network-free simulation, the goal is to generate reaction classes with distinct free energy change values. By Corollary 3.3-43, we know that it is sufficient to count overlaps of energy patterns with the reaction center. Therefore, the patterns in an energy rule only need to be resolved locally and not necessarily over the whole particle. Since the energy patterns are themselves finite, the depth to which configurations local to the reaction center have to be explored is also finite.

The goal of the energy-based specification is to simplify the representation of cooperative interactions. Theoretically it is possible to use very complex rules as energy-based rules, such as rules with more than two reactant patterns, binding rules that form loops within a complex. However, the utility of the energy-based specification for those circumstances has not been clearly determined. The most straightforward application of energy rules is for simple elementary reaction mechanisms where the complexity of variations in local context can be abstracted away in the form of energy pattern matches. For the rest of the chapter, we limit the discussion to exactly two types of energy rules: bimolecular binding reactions, and unimolecular state change reactions.

3.4.2 Rule Expansion Strategy

The overall strategy for refining the rule is as follows: (i) compose molecule type specific to the local context of the reaction center of the energy rule, (ii) incorporate tagging of the reaction center on the molecule types, (iii) identify how far patterns need to be expanded in order to capture unique combinations of energy pattern matches, (iii) use network generation to expand the local context upto those unique combinations, (iv) use network generation to expand the energy rule onto the generated set of patterns, and (v) use energy matches to compute the rate for each reaction rule in the expanded set.

1) Given molecule type definition $MolDef = (M, C, S, N_{MC}, N_{MCS})$, bond definition N_{MCMC}, energy patterns E, and a reversible energy rule (r_+, r_-) where $r_+ = (R, P)$.
2) Compute $MolDefNew = (M^{new}, C^{new}, S^{new}, N_{MC}^{new}, N_{MCS}^{new}, N_{MCMC}^{new})$ which excludes molecules, components, internal states and bonds that are not relevant to compute energies.
3) Add new molecule types with tagged reaction centers and update $MolDefNew$.
4) Synthesize a set of seed species Sp and a set of expander rules Ru from $MolDefNew$.
5) Expand context and collect "species": $(Sp', \ldots) = NetGen(Sp, Ru)$
6) Expand energy rule using generated species: $(Sp'', Ru^{exp}) = NetGen(Sp', \{r_+, r_-\})$
7) Evaluate Arrhenius rate laws for $r \in RU^{exp}$.

3.4.3 Building Molecule and Bond Types for Local Context

Given a molecule type definition $(M, C, S, N_{MC}, N_{MCS})$ and a bond type definition N_{MCMC}, a reversible energy rule (r_+, r_-), where $r_+ = (R, P)$, an energy definition involving a set of energy patterns E, and a map from energy patterns to energies of formation $E \to G_E$, we first reduce the rule to its reaction center: $r^\circ = (R^\circ, P^\circ)$ as in Definition 3.3-20. Then, we use embeddings from R° and P° to select "relevant" energy patterns (these are the only energy pattern embeddings that matter, according to Corollary 3.3-43).

$$E^r = \left\{p \in E \middle| \exists \phi_{p,q}, q \in R^\circ \cup P^\circ\right\}$$

Then we build an extended molecule type definition $(M^{new}, C^{new}, S^{new}, N_{MC}^{new}, N_{MCS}^{new}, N_{MCMC}^{new})$, that captures the space of local configurations around the reaction center. M^{new} and C^{new} contain all molecule types and component types present in the patterns in R, P, E^r, and $N_{MC}^{new}$is populated from N_{MC} for those molecule and component types. For every component type c in molecule type m, if at least one pattern in R, P, E^r has an internal state, then S^{new} and N_{MCS}^{new} are populated with all internal states available to that component type in S and N_{MCS}. $N_{MCMC}^{new}$is populated with all pairs of doublets $(m_1, c_1), (m_2, c_2)$ that are assigned 1 under N_{MCMC} where at least one doublet (m_1, c_1) has the value 1 under N_{MC}^{new}. If any new molecule types or component types were added in this step, $M^{new}, C^{new}, N_{MC}^{new}$ are updated.

3.4.4 Tagging the Reaction Center

Next, we build a set of modified molecule types where molecules and components participating in the reaction center are tagged. For example, suppose the reaction center r° is

```
A(b) + B(a) -> A(b!1).B(a!1)
```

We first build the renaming maps

$$ren_M = \{A \to A_{mod}, B \to B_{mod}\}$$

$$ren_C = \{(A,b) \to (A_{mod}, b_{mod}), (B,a) \to (B_{mod}, a_{mod})\}$$

Then, using the templates in $(M^{new}, C^{new}, S^{new}, N^{new}_{MC}, N^{new}_{MCS}, N^{new}_{MCMC})$, we create additional molecule types A_{mod}, B_{mod}, which are renamed versions of A and B in which one component of type b is replaced with one of type b_{mod}, and one of type a is replaced with one of type a_{mod}. The new molecule types and component types are added to M^{new} and C^{new} respectively. N^{new}_{MC} is updated with new stoichiometries, whereas internal state and bond state definitions in N^{new}_{MCS} and N^{new}_{MCMC} are copied over to the new component types. For example, if the template molecule type was $A(b,b,c)$, then the new molecule type is $A_{mod}(b_{mod}, b, c)$, and the following updates are made:

$$\forall s, N^{new}_{MCS}(A,b,s) = N^{new}_{MCS}(A_{mod}, b_{mod}, s)$$

$$\forall m, c\ N^{new}_{MCMC}\big((A,b),(m,c)\big) = N^{new}_{MCMC}\big((A_{mod}, b_{mod}),(m,c)\big)$$

The only exception to the N^{new}_{MCMC} assignment is an asymmetric tagging for the specific pair that was tagged in the first place: i.e. $A_{mod}(b_{mod}!1).B_{mod}(a_{mod}!1)$ is allowed, but not $A_{mod}(b_{mod}!1).B(a!1)$ or $A(b!1).B_{mod}(a_{mod}!1)$.

3.4.5 Topology Constraints

Here, we compose a set of constraints that determine how deep patterns must be expanded to properly account for all energy contributions. For each energy pattern e, we use tagging-by-renaming to generate variants $\{e_i\}$ that have exactly one reaction center. Then, the distance of non-tagged molecules from the reaction center is calculated. For example, if $A(b!1,b,c).B(a!1)$ is an energy pattern, it results in the variants $A_{mod}(b_{mod}!1,b,c).B_{mod}(a_{mod}!1)$ and $A_{mod}(b!1,b_{mod},c).B(b!1)$. In the first variant, there is no molecule that is not part of the reaction center. In the second variant, the molecule B is at a distance 1 from the reaction center. The maximum distance for each non-tagged molecule type is computed over all variants of all energy patterns. This results in a map from each molecule type to a non-negative number, i.e. $M^{new} \to \mathbb{N}_0$.

3.4.6 Context Expansion

Here, we use network generation to expand the patterns such that the context surrounding the reaction center is unambiguously resolved in terms of energy pattern matches. Given new molecule and bond types $MolDefNew = (M^{new}, C^{new}, S^{new}, N_{MC}^{new}, N_{MCS}^{new}, N_{MCMC}^{new})$, and topology constraints, $Topol: M^{new} \to \mathbb{N}_0$, we build a set of seed species $Sp = Seed(MolDefNew)$ in which each pattern has one molecule only, which is an instance of one molecule type, and in which each component is instantiated with the maximum allowed stoichiometry in N_{MC}^{new}, and with an internal state specified by N_{MCS}^{new} (none if unspecified, randomly chosen if multiple internal states are specified). Then, we build a set of expander rules $Ru = Rules(MolDefNew)$, wherein for each m_1, m_2, c_1, c_2 such that $N_{MCMC}^{new}\big((m_1, c_1), (m_2, c_2)\big) = 1$, there exists a rule:

```
m1(c1) + m2(c2) <-> m1(c1!1).m2(c2!1)
```

Similarly, for every m, c, s_1, s_2 such that $N_{MCS}^{new}(m, c, s_1) = N_{MCS}^{new}(m, c, s_2) = 1$, there exists a rule

```
m(c~s1!?) <-> m(c~s2!?)
```

Then network generation is performed using these seed species and rules, i.e. $NetGen(Sp, Ru)$ with the following constraints on species: (i) at least one species in each reaction must have a reaction center tag, (ii) no species generated can have more than one reaction center tag, (iii) the distance of any molecule to the reaction center tag is less than or equal to the limit computed from energy patterns, i.e. from the set of topology constraints $Topol$. Note that the species in this network are true species only under the new molecule type definition, but not under the original one.

3.4.7 Energy Rule Expansion

Here, we use network generation to expand the energy rule into normal reaction rules. The input rule set comprises only the forward and reverse directions of the energy rule, and with the reaction center tagged. The reactions are limited to use the species generated during context expansion. After expansion, the tags are removed and the resultant set of "reactions" constitute the expanded rule set. For each rule in this set, a unique free energy can be calculated by counting energy pattern matches (as in Definition 3.3-42) and this can be used to compute the numeric rate from the Arrhenius rate law (as in Definition 3.3-44).

3.4.8 Scalability Concerns

When there are no repeated components within molecules, the number of rules generated from an energy rule scale with the size of the largest energy pattern that overlaps with each reaction center, which is typically a low bound. However, when there are repeated components, the number of rules generated from the energy rule has the potential to grow combinatorially. Nevertheless, there is a finite limit on the number of rules that are possible since finite-sized energy patterns and finite-sized molecule types are used. Also, even in the worst case scenario, it is the pre-processing of the model that is inefficient and there is no reduction in the efficiency of the simulation algorithm.

3.4.9 Alternate Strategies

Prior to this approach, I considered two other strategies. These were rejected because the inefficiency was not resolved prior to simulation.

In the first strategy, based on rejection sampling [83], I proposed to treat energy rules as reaction classes and oversample them at a constant high rate. After the reaction event has fired, the identity of the products can be determined, which in turn can be used to determine the free energy of the just-fired reaction event, and consequently the difference between the correct rate and the oversampled rate. Following this, the reaction event can be probabilistically accepted or rejected resulting in exact kinetics. However, this strategy has two issues even if an upper bound on the reaction rate can be calculated for a given energy rule: (i) when there are repeated components, the bound may not be tight enough to enable efficient simulation, e.g. if a molecule has 3 components of the same type, then any energy patterns overlapping with a component of that type contribute three times the energy to the calculation of the bound, (ii) if there is at least one reaction generated by the rule that is much faster than the other reactions, then rejection sampling of the other reactions will be very inefficient.

In the second strategy, I proposed to use the current state of the simulation system to compute the immediate future possibilities and sample them exactly. However, this strategy was discarded because it required computing a unique energy term for every potential reaction center present in the system, which is inefficient for bimolecular reactions.

3.4.10 Implementation Issues

In order to implement the given procedure in BioNetGen, the network generation algorithm in the BioNetGen code needs to be modularized to perform generic rule refinement. First, the current code executes on the current model object that is persistent during runtime, but we require it to be called multiple times on arbitrary sets of molecule types, rules and species (e.g. Sections 3.4.6 and 3.4.7), and not necessarily on the current model as a whole. Second, the checks on generated species and reactions, as well as modifications done to them such as lumping of statistical factors, are currently only those that are required for building the reaction network specification. These procedures must be imported as modules, so that tasks specific to the current refinement goal can be flexibly performed, such as arbitrary checks on species and reactions (Section 3.4.5), turning lumping off, evaluating pattern matches and rate laws, preventing new species from being formed (Section 3.4.7), etc. For each of the examples below, the molecule type construction (Section 3.4.3), and reaction center tagging (Section 3.4.4) was done manually. Then, two different BNGL files were constructed manually, each calling the network generation code for respectively expanding context and energy rule. Parameters passed to network generation were tuned to be similar to the topology constraints (Section 3.4.6).

3.4.11 Example 1: Bivalent Ligand Bivalent Receptor

Consider an energy-based rule-based model with a bivalent ligand that binds a bivalent receptor. The molecule types are

```
Lig(R,R)

Rec(L,L)
```

There is only one type of bond

```
Lig(R!1).Rec(L!1)
```

Consider the energy rule

```
Lig(R)+Rec(L) <-> Lig(R!1).Rec(L!1)
```

Here, the single rule can generate an infinite state space of species because the combination of two sites on the receptor and two sites on the ligand leads to a chain that can grow indefinitely. First, because there is no other context in the model except unbound and bond states, we can skip Section 3.4.3 and reuse the same molecule types.

Next, we tag the reaction center on the rule,

```
Lig_RC(R_RC)+Rec_RC(L_RC) <-> Lig_RC(R_RC!1).Rec(L_RC!1)
```

Then we create the tagged molecule types where one component has been tagged that indicates participation in the reaction center.

```
Lig_RC(R_RC,R)

Rec_RC(L_RC,L)
```

Now, consider the following set of energy patterns:

```
Rec(L,L)

Lig(R,R)

Rec(L!1).Lig(R!1)
```

The tagged variants of these energy patterns are:

```
Rec_RC(L_RC,L)

Lig_RC(R_RC,R)
```

```
Rec_RC(L_RC!1).Lig_RC(R_RC!1)

Rec_RC(L!1).Lig(R!1)

Rec(L!1).Lig_RC(R!1)
```

Note that in the last two patterns, one of the molecules have been tagged, but the components have not been tagged. This is valid because both receptor and ligand have multiple sites for each other. The new molecule types are

```
Rec_RC(L_RC,L)

Lig_RC(R_RC,R)
```

The new bond types are

```
Rec_RC(L_RC!1).Lig_RC(R_RC!1)

Rec_RC(L!1).Lig_RC(R!1)
```

The updated list of molecule types is

```
Rec(L,L)

Lig(R,R)

Rec_RC(L_RC,L)

Lig_RC(R_RC,R)
```

The updated list of bond types is

```
Rec(L!1).Lig(R!1)

Rec_RC(L_RC!1).Lig_RC(R_RC!1)

Rec_RC(L!1).Lig_RC(R!1)
```

From the tagged variants of the energy patterns, we compute topology constraints.

```
Rec->1, Lig->1
```

Using the new molecule types, we can compose the set of seed species (Sp):

```
Rec(L,L)

Lig(R,R,R)

Rec_RC(L_RC,L)

Lig_RC(R_RC,R)
```

Using the new molecule types and bond types, we can compose the set of expander rules (Ru):

```
Lig(R) + Rec(L) <-> Lig(R!1).Rec(L!1)

Lig_RC(R) + Rec(L) <-> Lig_RC(R!1).Rec(L!1)

Lig(R) + Rec_RC(L) <-> Lig(R!1).Rec_RC(L!1)

Lig_RC(R_RC) + Rec_RC(L_RC) <-> Lig_RC(R_RC!1).Rec_RC(L_RC!1)
```

Performing network generation using the seed species and expander rules, i.e. $NetGen(Sp, Ru)$, we get the following species:

```
Rec(L,L)

Lig(R,R)

Rec_RC(L,L_RC)

Lig_RC(R,R_RC)

Lig_RC(R!1,R_RC).Rec(L!1,L)

Lig(R!1,R).Rec_RC(L!1,L_RC)
```

```
 Lig_RC(R,R_RC!1).Rec_RC(L,L_RC!1)

 Lig(R!1,R).Lig_RC(R!2,R_RC).Rec(L!2,L!1)

Lig(R!1,R!2).Rec(L!2,L).Rec_RC(L!1,L_RC)

Lig(R!1,R!2).Lig_RC(R!3,R_RC).Rec(L!3,L!2).Rec_RC(L!1,L_RC)

Lig_RC(R!1,R_RC!2).Rec(L!1,L).Rec_RC(L,L_RC!2)

Lig(R!1,R).Lig_RC(R,R_RC!2).Rec_RC(L!1,L_RC!2)

Lig(R!1,R).Lig_RC(R!2,R_RC!3).Rec(L!2,L).Rec_RC(L!1,L_RC!3)

Lig(R!1,R).Lig_RC(R!2,R_RC!3).Rec(L!2,L!1).Rec_RC(L,L_RC!3)

Lig(R!1,R!2).Lig_RC(R,R_RC!3).Rec(L!2,L).Rec_RC(L!1,L_RC!3)

Lig(R!1,R).Rec(L!1,L)
```

Using the energy rule to generate reactions drawn from this set of species and then removing the reaction center tags, we get the following rule variants:

```
1. Lig(R,R) + Rec_RC(L,L) -> Lig_RC(R,R!1).Rec(L,L!1)

2. Lig(R,R) + Lig(R!1,R).Rec(L!1,L) -> Lig(R!1,R).Lig(R,R!2).Rec(L!1,L!2)

3. Lig(R,R) + Lig(R!1,R!2).Rec(L!2,L).Rec(L!1,L) ->
   Lig(R!1,R!2).Lig(R,R!3).Rec(L!2,L).Rec(L!1,L!3)

4. Lig(R!1,R).Rec(L!1,L) + Rec(L,L) -> Lig(R!1,R!2).Rec(L!1,L).Rec(L,L!2)

5. Lig(R!1,R).Lig(R!2,R).Rec(L!2,L!1) + Rec(L,L) ->
   Lig(R!1,R).Lig(R!2,R!3).Rec(L!2,L!1).Rec(L,L!3)
```

```
6. Lig(R!1,R).Rec(L!1,L) + Lig(R!1,R).Rec(L!1,L) ->
   Lig(R!1,R).Lig(R!2,R!3).Rec(L!2,L).Rec(L!1,L!3)
```

By counting matches of energy patterns, as in Definition 3.3-42, the free energy of these rule variants are computed, and they result in the following expressions respectively:

```
1. (-E_R)+(-E_L)+E_RL

2. (-E_L)+E_RL

3. (-E_L)+E_RL

4. (-E_R)+E_RL

5. E_RL

6. E_RL
```

3.4.12 Example 2: EGFR-Grb2-Shc

Consider an energy-based rule-based model with the following molecule types

```
EGF(r)

EGFR(l,d,Y992~0~P,Y1068~0~P,Y1148~0~P)

Grb2(SH2,SH3)

Shc(PTB,Y317~0)
```

Consider the energy rule:

```
EGFR(Y1068~P) + Grb2(SH2) <-> EGFR(Y1068~P!1).Grb2(SH2!1)
```

The context-free version of this rule is

```
        EGFR(Y1068) + Grb2(SH2) <-> EGFR(Y1068!1).Grb2(SH2!1)
```

Energy patterns that are relevant to the rule have to completely embed at least one of these patterns.

Suppose the set of relevant energy patterns (with matching energies) was:

EGFR(Y1068!1).Grb2(SH2!1)	E_RG1
EGFR(Y1068!1,Y1148).Grb2(SH2!1)	E_RG2
EGFR(Y1068,Y1148!1).Shc(PTB!1)	E_RS
EGFR(Y1068!1,Y1148!2).Grb2(SH2!1).Shc(PTB!2)	E_RGS
Grb2(SH2,SH3!1).Sos(dom!1)	E_GS

Given these energy patterns and the rule patterns, the implied molecule type definition is:

```
        EGFR(Y1068~0~P,Y1148)

        Grb2(SH2,SH3)

        Shc(PTB)

        Sos(dom)
```

Also, from the model, the bond definition is given by

```
        EGFR(Y1068!1).Grb2(SH2!1)

        EGFR(Y1148!1).Shc(PTB!1)

        Grb2(SH3!1).Sos(dom!1)
```

Now, we take the context-free energy rule and add reaction center tags:

```
        EGFR_RC(Y1068_RC) + Grb2_RC(SH2_RC) <->

EGFR_RC(Y1068_RC!1).Grb2(SH2_RC!1)
```

We create variants of the energy patterns using tagged molecule and component names.

```
EGFR_RC(Y1068_RC!1).Grb2_RC(SH2_RC!1)

EGFR_RC(Y1068_RC!1,Y1148).Grb2_RC(SH2_RC!1)

EGFR_RC(Y1068_RC,Y1148!1).Shc(PTB!1)

EGFR_RC(Y1068_RC!1,Y1148!2).Grb2(SH2_RC!1).Shc(PTB!2)

Grb2_RC(SH2_RC,SH3!1).Sos(dom!1)
```

This results in new molecule types and bond types. The new molecule types are:

```
EGFR_RC(Y1068_RC~0~P,Y1148)

Grb2_RC(SH2_RC,SH3)
```

The new bond types are:

```
EGFR_RC(Y1068_RC!1).Grb2_RC(SH2_RC!1)

EGFR_RC(Y1148!1).Shc(PTB!1)

Grb2_RC(SH3!1).Sos(dom!1)
```

The updated list of molecule types is

```
EGFR(Y1068~0~P,Y1148)

Grb2(SH2,SH3)

Shc(PTB)

Sos(dom)

EGFR_RC(Y1068_RC~0~P,Y1148)

Grb2_RC(SH2_RC,SH3)
```

The updated list of bond types is:

```
EGFR(Y1068!1).Grb2(SH2!1)

EGFR(Y1148!1).Shc(PTB!1)

Grb2(SH3!1).Sos(dom!1)

EGFR_RC(Y1068_RC!1).Grb2_RC(SH2_RC!1)

EGFR_RC(Y1148!1).Shc(PTB!1)

Grb2_RC(SH3!1).Sos(dom!1)
```

Using the variant energy patterns, we can compute maximum distance of each molecule from a reaction center:

```
EGFR -> 0, Grb2-> 0, Shc -> 1, Sos -> 1
```

Using the updated molecule types, we can compose a set of seed "species" (Sp):

```
EGFR(Y1068~0,Y1148)

Grb2(SH2,SH3)

Shc(PTB)

Sos(dom)

EGFR_RC(Y1068_RC~0,Y1148)

Grb2_RC(SH2_RC,SH3)
```

Using the updated molecule types and bond types, we can compose expander rules Ru:

```
EGFR(Y1068~0!?) <-> EGFR(Y1068~P!?)

EGFR_RC(Y1068_RC~0!?) <-> EGFR_RC(Y1068_RC~P!?)
```

```
EGFR(Y1068)+Grb2(SH2) <-> EGFR(Y1068!1).Grb2(SH2!1)

EGFR(Y1148) + Shc(PTB) <-> EGFR(Y1148!1).Shc(PTB!1)

Grb2(SH3) + Sos(dom) <-> Grb2(SH3!1).Sos(dom!1)

EGFR_RC(Y1068_RC) + Grb2_RC(SH2_RC) <->
EGFR_RC (Y1068_RC!1).Grb2_RC(SH2_RC!1)

EGFR_RC(Y1148) + Shc(PTB) <-> EGFR_RC(Y1148!1).Shc(PTB!1)

Grb2_RC(SH3) + Sos(dom) <-> Grb2_RC(SH3!1).Sos(dom!1)
```

Performing network generation using the seed species and expander rules, i.e. $NetGen(Sp, Ru)$, and using the constraints mentioned in Section 3.4.6, we get the following state space of species (Sp'):

```
EGFR(Y1068~0,Y1148)

Grb2(SH2,SH3)

Shc(PTB)

Sos(dom)

EGFR_RC(Y1068_RC~0,Y1148)

Grb2_RC(SH2_RC,SH3)

EGFR_RC(Y1068_RC~P,Y1148)

EGFR_RC(Y1068_RC~0!1,Y1148).Grb2_RC(SH2_RC!1,SH3)

EGFR_RC(Y1068_RC~0,Y1148!1).Shc(PTB!1)

Grb2_RC(SH2_RC,SH3!1).Sos(dom!1)
```

```
        EGFR_RC(Y1068_RC~P!1,Y1148).Grb2_RC(SH2_RC!1,SH3)

        EGFR_RC(Y1068_RC~P,Y1148!1).Shc(PTB!1)

        EGFR_RC(Y1068_RC~0!1,Y1148).Grb2_RC(SH2_RC!1,SH3!2).Sos(dom!2)

        EGFR_RC(Y1068_RC~P!1,Y1148).Grb2_RC(SH2_RC!1,SH3!2).Sos(dom!2)

        EGFR_RC(Y1068_RC~0!1,Y1148!2).Grb2_RC(SH2_RC!1,SH3).Shc(PTB!2)

EGFR_RC(Y1068_RC~0!1,Y1148!2).Grb2_RC(SH2_RC!1,SH3!3).Shc(PTB!2).Sos(dom!3)

        EGFR_RC(Y1068_RC~P!1,Y1148!2).Grb2_RC(SH2_RC!1,SH3).Shc(PTB!2)

EGFR_RC(Y1068_RC~P!1,Y1148!2).Grb2_RC(SH2_RC!1,SH3!3).Shc(PTB!2).Sos(dom!3)
```

Then we expand the energy rule using only these species, and then remove the reaction center tags. This results in the four variants:

1. `EGFR(Y1068~P,Y1148) + Grb2(SH2,SH3) ->`

   `EGFR(Y1068~P!1,Y1148).Grb2(SH2!1,SH3)`

2. `EGFR_RC(Y1068~P,Y1148) + Grb2(SH2,SH3!1).Sos(dom!1) ->`

   `EGFR(Y1068~P!1,Y1148).Grb2(SH2!1,SH3!2).Sos(dom!2)`

3. `EGFR(Y1068~P,Y1148!1).Shc(PTB!1) + Grb2(SH2,SH3) ->`

   `EGFR(Y1068~P!1,Y1148!2).Grb2(SH2!1,SH3).Shc(PTB!2)`

4. `EGFR(Y1068~P,Y1148!1).Shc(PTB!1) + Grb2(SH2_RC,SH3!1).Sos(dom!1) ->`

   `EGFR(Y1068~P!1,Y1148!2).Grb2_RC(SH2!1,SH3!3).Shc(PTB!2).Sos(dom!3)`

Computing the energy of formation for each of these rules using energy pattern matches (using Definition 3.3-42) results in a unique free energy change expression for each rule. The free energy changes in the forward direction for each reaction are as follows:

1. E_RG1+E_RG2
2. E_RG1+E_RG2+(-E_GS)
3. E_RG1+(-E_RS)+E_RGS
4. E_RG1+(-E_RS)+E_RGS+(-E_GS)

3.5 CONCLUDING REMARKS

The work in this chapter was geared towards bridging two recent advances in rule-based modeling: the energy-based rule-based model specification [34] and the network-free simulation algorithm [39]. First I showed that only overlaps of energy patterns with a reaction center need to be considered. Then I showed that when energy rules are limited to a single transformation, pre-processing the rules to expand them is sufficient to generate reaction classes that can be used for network-free simulation. The procedure also requires a finite number of pre-processing steps, and does not require modifying the network-free simulation algorithm.

4.0 MODEL CONSTRUCTION USING REACTION RULE MODULES

4.1 SYNOPSIS

Rule-based models are being built with increasingly large sizes, as evidenced by some of the models referenced in this thesis [44]–[47]. Therefore, improving and scaling up rule-based model construction is an active area of research [65], [84]. In **Section** Error! Reference source not found. Error! Reference source not found., I detail some of the issues that hinder model construction. **Section 4.3 CURRENT APPROACHES**, I review some of the current strategies used in building larger rule-based models. In **Section** Error! Reference source not found. Error! Reference source not found., I provide an additional approach that complements the current approaches. I demonstrate this approach using an example of Ras mutants in which there are many possible models for explaining the behavior.

4.2 SOURCES OF COMPLEXITY IN MODEL CONSTRUCTION

4.2.1 Contextual Complexity

The same kinetic process can occur with different rates under different local conditions. Enumerating the combinations of local conditions that lead to the different kinetic variants can be a time consuming process. This is especially true, when there are cooperative interactions that occur on the same molecule. By the principle of detailed balance, the equilibrium constants of a loop of reaction mechanisms have to be constrained, as demonstrated in **Section** Error! Reference source not found. Error! Reference source not found.**.** Ignoring these constraints lead to too many unconstrained parameters and incorrect models. However, identifying such loops in reaction mechanisms is not trivial, e.g. in a model with 4 molecule types and 24 rules, Faeder et al. identify 4 different loops that need to be constrained [35].

4.2.2 Model Hypotheses

Usually, multiple hypotheses are available regarding how the kinetic classes are structured. These variations can be simple changes to rule parameters, e.g. say in Hypothesis 1, A binds B at a certain rate:

```
A(b) + B(a) -> A(b!1).B(a!1)          k
```

In Hypothesis 2, let the rate be reduced by a certain factor.

```
A(b) + B(a) -> A(b!1).B(a!1)          f*k
```

Now, these are specific hypotheses about a single rule. A model hypothesis could involve many combinations of these hypotheses over the entire rule set, and each combination would be a valid model. Enumerating the combinations manually can be a cumbersome process.

4.2.3 Variant Molecule Types

A common experimental procedure is to modify the participating molecules (such as by truncating a domain or removing a phosphorylation site), and examining the changes in behavior, which will lead to functional hypotheses about the individual molecules as well as the role they play in the larger system. However, when building a rule-based model with many similar molecule types, reaction rules involving each molecule type have to be replicated for all the variants. When performed manually, this can consume a lot of time and resources.

For example, in a model of ErbB family signaling [46], the general paradigm is that receptors can recruit ligands, and also form homo- and hetero- dimers, and there is cooperativity between the two processes. The species in the model are monomers (say, R), homodimers, say (RR), heterodimers (say RR'), and each of these species can bind one or two ligand molecules. Given that the model has two ligand types and four receptor types, enumerating the combinations that lead to ligand-binding and dimerization reaction classes comprises a large fraction of the 625 rules in the models. Not only so, each combination of cooperative interactions that results in a loop has to be cross-checked for detailed balance.

4.3 CURRENT APPROACHES

4.3.1 The Macro Approach

PySB [84] is a recent modular construction framework that was developed for rule-based models. Here, Python methods that build reaction rules are used to build and aggregate rule sets. For example, say a method was defined:

```
def catalyze([<moleculetypes>]):

        ….
```

Say the catalysis method performs a simple of operation of taking two patterns, and building a catalysis rule set, such as:

```
E(s) + S(e~Y) <-> E(s!1).S(e~Y!1)   k1,k2

E(s!1).S(e~Y!1) -> E(s) + S(e~pY)     kcat
```

If there are many combinations of enzymes and substrates, then the method can be called repeatedly on each combination. For example, passing the method the enzyme E1 and substrate S1, we can generate the rules,

```
E1(s) + S1(e~Y) <-> E1(s!1).S1(e~Y!1)   k1,k2

E1(s!1).S1(e~Y!1) -> E1(s) + S1(e~pY)        kcat
```

The final model is constructed by aggregating calls to rule construction methods into a single program.

4.3.2 The Typing Approach

In MetaKappa [85], the language enables the systematic variation of molecule types. For example, suppose we consider both MEK1 and MEK2 variants in a model, then, first we define a generic molecule type (Kappa syntax – %gen = generic molecule type):

```
%gen: MEK(S~Y,ST~Y)
```

Then we introduce concrete variations of the molecule type (Kappa syntax – %conc = concrete molecule type):

```
%conc: MEK1 = MEK[S\{S218} ST\{S222}]

%conc: MEK2 = MEK[S\{S222} ST\{S226}]
```

Here component types from the generic type (S,ST) are replaced with concrete variants (S218, etc.) Now we are allowed to define rules on both generic and concrete types:

```
MEK(S~pY) -> MEK(S~Y)

MEK1(S218~pY) -> MEK(S218~Y)
```

When compiling the model, rules constructed on generic types are expanded automatically to corresponding rules on the concrete types.

4.3.3 The Energy-based Approach

Energy based approaches [50]–[52] handle the combinatorial complexity that arises from allosteric and cooperative interactions. In energy-based BioNetGen, energy patterns are first specified by the user and assigned energy values:

```
A(b!1).B(a!1) G_ab
```

```
B(a,x~0)        G_b0

B(a,x~1)        G_b1
```

Then, the model is specified using “energy-rules”,

```
A(b) + B(a) -> A(b!1).B(a!1)        Arrhenius(phi, E_ab)
```

When specified in this way, the free energy values (and thereby the reaction rate parameters) of reactions generated from the rule can be calculated by counting energy pattern matches to reactant and product sides. For example, the rule generates the following reactions with the following energies.

```
A(b) + B(a,x~0) -> A(b!1).B(a!1,x~0)        -G_b0 +G_ab

A(b) + B(a,x~1) -> A(b!1).B(a!1,x~1)        -G_b1 +G_ab
```

This provides a high degree of compression for cooperative processes, because it is not necessary for the modeler to provide a separate rule and rate constant for each reaction class. Providing a set of energy rules and energy patterns ensures that all combinations of patterns that lead to contextual variants of the rule can be enumerated.

4.4 REACTION RULE MODULES

In the macro approach, the aggregable model object is the call to the Python rule-building method. However, it is not useful to create variations of pre-existing rule-sets. On the other hand, MetaKappa and energy-based rules provide a significant compression by identifying higher-order abstractions such as types of molecule types and energy patterns. Here, I propose the idea of a reaction module that complements the above approaches. Using a prototype implementation of reaction modules for BioNetGen, I demonstrate the utility of the approach.

4.4.1 Motivating Example

Consider the molecule Ras, which is a small GTPase that is important for growth factor signaling. Ras mutations are widespread in cancer, and are therefore a target of intense experimental scrutiny. The Ras molecule has a nucleotide binding site, which binds GTP or GDP. The GTP-bound state is called ‘active’, and the Ras molecule can then bind effector molecules and activate them. GTP-bound Ras slowly hydrolyzes GTP into GDP, which is the inactive form. Other molecule types regulate Ras activation and inactivation, e.g. RasGEF which promotes the release of GDP from the inactive form leading to its subsequent activation, and RasGAP which increase the endogenous GTP-ase activity of Ras, leading to its subsequent inactivation. In a study by Stites et al [86], multiple hypothesis are suggested for how Ras mutants can increase Ras activation: reduced GTPase activity, GAP insensitivity, increased effect affinity, etc. Here, we use this system as an example where model building is hindered by multiple overlapping model hypotheses, and show how a reaction-module approach can be useful.

4.4.2 Basic Model

We first build the wildtype model. Ras nucleotide binding and hydrolysis is represented as:

```
Ras(nuc~u,bs) <-> Ras(nuc~gdp,bs)          kf_gdp, kr_gdp

Ras(nuc~u,bs) <-> Ras(nuc~gtp,bs)          kf_gtp, kr_gtp
Ras(nuc~gtp,bs) -> Ras(nuc~gdp,bs)          k_hyd
```

RasGAP activity, which is an enhancement of GTP-hydrolysis, is modeled as

```
Ras(nuc~gtp,bs) + GAP(ras) <-> Ras(nuc~gtp,bs!0).GAP(ras!0) kf_GAP,kr_GAP

Ras(nuc~gtp,bs!0).GAP(ras!0) -> Ras(nuc~gdp,bs) + GAP(ras)              kcat_GAP
```

RasGEF activity, which is enhanced release of bound nucleotide, is modeled as

```
Ras(nuc~gdp,bs) + GEF(ras) <-> Ras(nuc~gdp,bs!0).GEF(ras!0)  kf_GEF_0,kr_GEF_0

Ras(nuc~gdp,bs!0).GEF(ras!0) -> Ras(nuc~u,bs) + GEF(ras)       kcat_GEF1

Ras(nuc~gtp,bs) + GEF(ras) <-> Ras(nuc~gtp,bs!0).GEF(ras!0)  kf_GEF_1,kr_GEF_1

Ras(nuc~gtp,bs!0).GEF(ras!0) -> Ras(nuc~u,bs) + GEF(ras)       kcat_GEF1
```

Activated Ras binds effector molecules which lead to subsequent signaling events.

```
Ras(nuc~gtp,bs) + Eff(ras) <-> Ras(nuc~gtp,bs!0).Eff(ras!0)      kf_Eff, kr_Eff
```

Let K_gdp, K_gtp, K_GAP, K_GEF1, K_GEF2 and K_eff be the corresponding equilibrium binding constants of each binding event respectively.

4.4.3 Mechanistic Hypotheses

Stites et al. [86] experimentally explore multiple mechanistic hypotheses about Ras mutations. We show how these hypotheses affect the modeled rules.

4.4.3.1 Reduced GTPase activity (RGA)

A Ras mutant could slow down the endogenous GTPase activity, leading to persistence of the activated state. In the model, this assumption would affect the k_hyd parameter, which would be reduced by a factor, say f_RGA*k_hyd.

4.4.3.2 GAP Insensitivity (GI)

A Ras mutant could be insensitive to GAP-induced acceleration of hydrolysis. Thus, even when GAP binds, inactivation proceeds only at the endogenous rate. In the model, this assumption would involve replacing the k_cat_GAP parameter with the endogenous hydrolysis rate k_hyd.

4.4.3.3 Increased Effector Affinity (IEA)

A Ras mutant could bind the effector stronger than wildtype Ras, which would lead to increased downstream activation. In the model, this would mean that the effector binding affinity K_eff is multiplied by a factor f_IEA.

4.4.4 Combinatorial Complexity in the Hypotheses Space

Note that the hypotheses above do not have to be mutually exclusive. Some combination of them could realistically be present in a Ras mutant. Also, the mechanisms can influence each other as well. For example, GAP insensitivity results in the GAP-bound catalytic rate being equal to the endogenous rate of hydrolysis, but the endogenous rate itself can be modified by the reduced GTPase activity hypotheses. Thus every combination of hypotheses is a different model.

Consider a set of Boolean variables (which take values 0 or 1) which can be used to formulate the combinations. Let b_mut be the variable that indicates whether a mutant is present or not, b_GI indicates GAP insensitivity, b_RGA indicates reduced GTPase activity and b_IEA indicates increased effector affinity. In total there are 1 wild type + 2^3 mutant models, i.e. 9 in all. Here we demonstrate how each combination of hypotheses can be built using reaction modules.

4.4.5 Building Reaction Modules

Here, we define a reaction module as a set of rules and associated rate expressions and parameters to which systematic modification operations can be defined. For demonstration, we use a module that models Ras interaction with GAP:

```
Module Ras-GAP-basic

    Rule[1] = Ras(nuc~gtp,bs) + GAP(ras) <-> Ras(nuc~gtp,bs!0).GAP(ras!0)

    Rate[1] = kf_gap, kr_gap

    Rule[2] = Ras(nuc~gtp,bs!0).GAP(ras!0) -> Ras(nuc~gdp,bs) + GAP(ras)

    Rate[2] = kcat_gap

    Param = kf_gap, kr_gap, kcat_gap
```

Now a modification can be defined where we add the context Ras(t~wt)

```
Module Ras-GAP-wt gets Ras-GAP-basic

    AddContext Ras(t~wt)
```

Similarly, we can define a module with the added context Ras(t~mut), indicating mutant.

```
Module Ras-GAP-mut gets Ras-GAP-basic

    AddContext Ras(t~mut)
```

Now, we can define variations of the mutant module under each hypothesis. Here, we define a modification that replaces one parameter expression with another.

```
Module Ras-GAP-mut-GI gets Ras-GAP-mut

        SwapParam kcat_gap k_hyd

Module Ras-GAP-mut-GI-RGA gets Ras-GAP-mut

        SwapParam kcat_gap f_RGI*k_hyd
```

Now, the logic for building rules and rate expressions for each combination of hypotheses is embedded within each module.

4.4.6 Aggregating Modules using Boolean Variables

The RasGAP portion of the model with its alternate variants can be specified with Boolean variables:

```
if(b_mut==0)

        load ras-GAP

else

        load ras-GAP-wt

        if(b_GI==1)

                if(b_RGA==1)

                        load ras-GAP-mut-GI-RGA

                else
```

```
                        load ras-GAP-mut-GI

        else

                    load ras-GAP-mut
```

From the specified logic, the expanded rules and rate expressions are automatically calculated. For example, the model will have the rule:

```
    Ras(nuc~gtp,bs!0,t~mut).GAP(ras!0) -> Ras(nuc~gdp,bs,t~mut) + GAP(ras)
```

The rate expression for the rule will be compiled as

```
    b_GI*(b_RGA*f_RGI*k_hyd + (1-b_RGA)*k_hyd) + (1-b_GI)*kcat_GAP
```

Evaluating the expression for different settings of b_RGA and b_GI will result in the right parameter expressions under those assumptions:

```
    b_GI=0      b_RGA=0     kcat_GAP

    b_GI=0      b_RGA=1     kcat_GAP

    b_GI=1      b_RGA=0     k_hyd

    b_GI=1      b_RGA=1     f_RGI*k_hyd
```

4.4.7 Comparison with Current Approaches

Rule sets are not distinct objects with attributes and methods in the macro approach. To build this model with macros, we would have to define three separate rule-building methods for each hypotheses combination and then aggregate them. Here, there is an added layer of abstraction in the form of the Rule Module object, and defining standard procedures to modify them minimizes the effort involved in building individual rules.

If we were to build this model using the typing approach, we would define two concrete variants of the generic Ras, but the compiled expression for the different combinations of hypotheses has to be constructed manually. Adding a Rule Module layer can therefore complement the typing approach.

The energy-based specification can also benefit from the module strategy, by defining Energy Pattern Modules in addition to Rule Modules. Energies for pattern assignments can then be compiled using Boolean variables that depend on the model hypotheses, e.g.

```
G_patt = b_1*G_1 + (1-b_1)*b_2*G_2
```

This enables building of models where the complexity of assumptions can be tuned. For example, cooperative energy terms can be added sequentially for 2-molecule patterns only under one assumption, for 2-molecule and 3-molecule patterns under another assumption, and so on.

A caveat for the rule module approach is complex rules with many instances of the same type would require complex modification strategies. For example, in the case demonstrated here, there is only one Ras molecule per rule which was easily modified by adding context. However, suppose there was a rule with 2 Ras molecules, and *n* Ras variants, then *n^2* variants would need to be generated systematically. This could be co-ordinated with the macro approach.

Macros, typing, energy rules and reaction modules all modularize different aspects of rule-based modeling and therefore can be used to complement each other. It is possible that a rule-based specification of the future would be able to use all four strategies in a seamless manner.

5.0 CONCLUSION

Rule-based modeling is a graph-based approach for specifying biochemical kinetics[4], [25]. In a rule-based model, structured graphs called patterns specify parts of molecules and complexes, and a reaction rule specifies graph operations on a combinations of patterns. A reaction rule can be used to represent a class of many reactions by specifying the modifications common to the class (say, binding or phosphorylation), and the minimal configuration of sites necessary for that class (say, a phosphorylated state, or a particular arrangement of non-covalent bonds) [24], [33], [34]. Mapping such reaction classes to rate laws enables an explicit site-based kinetic specification that is typically more compact than the equivalent reaction network because of sparse dependence relations between sites. The equivalent reaction network and the corresponding combinatorially complex state space of molecules and complexes can be generated automatically from the rule based model if such a finite network exists [24], [33], [34]. Compact rule-based models have been constructed for systems with combinatorially complex reaction networks [35], [79], and the reaction rule has been used as a portable data object to build databases of kinetic interactions [44], [46]. In this work, I provided new advances in the following aspects of the rule-based framework: building models, visualizing model content and simulating models.

In Chapter 2, I first reviewed current diagramming approaches, both manual and automated, for rule-based models. Multiple standards exist for diagrams constructed by hand [11], [57], [59]. A few automated tools exist for rule-based models, but they are typically not global [28], not complete as a visual representation of signaling [32], or use a limited model-building framework that is favorable for visualization[65]. Here, I made two contributions: a new method for visualizing individual rules (called compact rule visualization), and a set of procedures for generating a global visualization of signal flow and regulation from a given set of reaction rules (called the regulatory graph). I also provided coarse-graining procedures that can compress the automatically generated regulatory graph into compact pathway diagrams. I demonstrated the method’s scalability with regard to number of rules and usefulness in identifying cascades and feedback loops using case studies [35], [44], [46].

In Chapter 3, I addressed a discrepancy between two recent advances in rule-based modeling and simulation. The first advance is the network-free simulation algorithm, which enables simulation of a rule-based model without having to generate the corresponding reaction network [39], [40]. This is accomplished by treating each reaction rule as a reaction class whose rate can be calculated by counting matches of the reactant patterns into the simulation system[39]. The second advance is the energy-based rule-based specification, which allows compact specification of cooperative processes without breaking detailed balance[50]–[52]. This is accomplished by specifying free energies of molecules and complexes as a sum of pattern matches weighted by energy values and computing rates of reactions from reaction free energies[50]. The discrepancy arises when one attempts to do a network-free simulation of an energy-based rule-based model. The rate at which a reaction fires depends on the free energy of the reaction which in turn depends on both reactants and products, but the identity and structure of the products are not available in a network-free simulation until after the reaction has fired. The solution provided here was to expand the energy rule into distinct “normal” rules whose reaction free energies can be computed uniquely, and to use the generated rules in a network-free simulation.

In Chapter 4, I discussed common approaches in the rule-based literature for improving and modularizing model construction and provided an additional method to supplement the current approaches. The current approaches include the macro approach[84], where a model is built using programming calls to methods that build rules, the typing approach[85], where molecule types are themselves arranged in a hierarchy of types and rules can be defined at any level on the hierarchy, and the energy-based approach[50], where pattern matches are mapped to energy values and free-energy accounting is automated for different combinations of matches. The approach I provided involves treating sets of rules as rule modules on which systematic modifications can be specified. Model construction can then be performed as Boolean combinations of rule modules, enabling automated compilation of rate expressions when different rates are supplied under different Boolean settings. This is useful when the same set of reaction rules can be parameterized in different ways depending on combinations of model hypotheses.

In recent years, attention has been paid to the fact that detailed mechanistic models are necessary to be able to understand, predict and perturb biochemical systems. Outside the rule-based framework, reaction networks of specific systems have tended to increase in size over time (e.g. [14], [87], [88]). A model of a whole unicellular organism has been built in which reaction networks have been used as submodels[48]. Rule-based modelers have improved on model-building efforts, using the reaction rule as a modular unit in building increasingly large databases of kinetic interactions(e.g. [35], [44], [46], [47], [79]). I expect the regulatory graph abstraction developed in Chapter 2, paired with sophisticated grouping algorithms [89], to be useful for interactively navigating and visualizing these databases. The regulatory graph, by virtue of being a reduced model representation, will also be useful in other contexts, such as visualizing simulation fluxes [81].

Another major problem with large sets of rules is the manual verification of detailed balance and the complexity involved in specifying cooperative processes. For example, the large models of Creamer et al. [46] and Chylek et al. [44] have many loops of binding reaction mechanisms that need to be manually enumerated and verified for satisfying detailed balance. This is addressed elegantly by the energy-based rule-based specification: energy rules specify minimal context, energy patterns specify cooperativity parameters, and variations of processes due to cooperative interactions can be enumerated automatically while preserving detailed balance [50]–[52]. Therefore, I expect the energy-based specification to play a bigger role in the development of rule databases in the future, and it should be the default way to encode reversible non-covalent binding interactions. The methods developed in Chapter 3 should then be important for the simulation of models built from these databases.

Finally, mechanistic models of biochemistry typically have a large number of free parameters, and experimental data rarely constrain all model parameters to narrow bands of values. Studies show that for a single model fitted to a particular set of data, different directions in parameter space are often constrained to different extents [90], although it is still possible to extract falsifiable predictions from such models [91]. This phenomenon can be exploited too, for example, it is possible to select over the space of model perturbations to design experiments, optimizing for how well the results of the experiment are expected to constrain parameter values[92], [93]. When multiple model hypotheses are involved, models can be selected using probabilistic fitness measures for how well the model explains the data (e.g. [94], [95]), such as the Bayes factor ratio [96]. The methods developed in Chapter 4 will be useful for building nested models based on modular hypotheses about kinetic interactions in the system, which can then be subjected to probabilistic model selection as above.

BIBLIOGRAPHY

[1] H. Kitano, “Systems biology: a brief overview.,” *Science*, vol. 295, no. 5560, pp. 1662–4, Mar. 2002.

[2] S. Ghosh, Y. Matsuoka, Y. Asai, K.-Y. Hsin, and H. Kitano, “Software for systems biology: from tools to integrated platforms,” *Nat. Rev. Genet.*, vol. 12, no. 12, pp. 821–832, Nov. 2011.

[3] D. L. Nelson, M. M. Cox, and A. L. Lehninger, *Lehninger Principles of Biochemistry*, 6th ed. New York: W.H. Freeman, 2013.

[4] L. A. Chylek, L. A. Harris, C. S. Tung, J. R. Faeder, C. F. Lopez, and W. S. Hlavacek, “Rule-based modeling: A computational approach for studying biomolecular site dynamics in cell signaling systems,” *Wiley Interdisciplinary Reviews: Systems Biology and Medicine*, vol. 6, no. 1. pp. 13–36, 2014.

[5] A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, and C. S. Woodward, “SUNDIALS: Suite of Nonlinear and Differential/Algebraic Equation Solvers,” *ACM Trans. Math. Softw.*, vol. 31, no. 3, pp. 363–396, Sep. 2005.

[6] D. T. Gillespie, “Exact stochastic simulation of coupled chemical reactions,” *J. Phys. Chem.*, vol. 81, no. 25, pp. 2340–2361, Dec. 1977.

[7] M. A. Gibson and J. Bruck, “Efficient Exact Stochastic Simulation of Chemical Systems with Many Species and Many Channels,” *J. Phys. Chem. A*, vol. 104, no. 9, pp. 1876–1889, Mar. 2000.

[8] M. Hucka, a. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, a. P. Arkin, B. J. Bornstein, D. Bray, a. Cornish-Bowden, a. a. Cuellar, S. Dronov, E. D. Gilles, M. Ginkel,

V. Gor, I. I. Goryanin, W. J. Hedley, T. C. Hodgman, J.-H. Hofmeyr, P. J. Hunter, N. S. Juty, J. L. Kasberger, a. Kremling, U. Kummer, N. Le Novere, L. M. Loew, D. Lucio, P. Mendes, E. Minch, E. D. Mjolsness, Y. Nakayama, M. R. Nelson, P. F. Nielsen, T. Sakurada, J. C. Schaff, B. E. Shapiro, T. S. Shimizu, H. D. Spence, J. Stelling, K. Takahashi, M. Tomita, J. Wagner, and J. Wang, “The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models,” *Bioinformatics*, vol. 19, no. 4, pp. 524–531, Mar. 2003.

[9] N. Le Novere, A. Finney, M. Hucka, U. S. Bhalla, F. Campagne, J. Collado-Vides, E. J. Crampin, M. Halstead, E. Klipp, P. Mendes, P. Nielsen, H. Sauro, B. Shapiro, J. L. Snoep, H. D. Spence, and B. L. Wanner, “{M}inimum information requested in the annotation of biochemical models ({M}{I}{R}{I}{A}{M}),” *Nat. Biotechnol.*, vol. 23, no. 12, pp. 1509–1515, Dec. 2005.

[10] D. Waltemath, R. Adams, F. T. Bergmann, M. Hucka, F. Kolpakov, A. K. Miller, I. I. Moraru, D. Nickerson, S. Sahle, J. L. Snoep, and N. Le Novere, “{R}eproducible computational biology experiments with {S}{E}{D}-{M}{L}--the {S}imulation {E}xperiment {D}escription {M}arkup {L}anguage,” *BMC Syst Biol*, vol. 5, p. 198, 2011.

[11] N. Le Novère, M. Hucka, H. Mi, S. Moodie, F. Schreiber, A. Sorokin, E. Demir, K. Wegner, M. I. Aladjem, S. M. Wimalaratne, F. T. Bergman, R. Gauges, P. Ghazal, H. Kawaji, L. Li, Y. Matsuoka, A. Villéger, S. E. Boyd, L. Calzone, M. Courtot, U. Dogrusoz, T. C. Freeman, A. Funahashi, S. Ghosh, A. Jouraku, S. Kim, F. Kolpakov, A. Luna, S. Sahle, E. Schmidt, S. Watterson, G. Wu, I. Goryanin, D. B. Kell, C. Sander, H. Sauro, J. L. Snoep, K. Kohn, and H. Kitano, “The Systems Biology Graphical Notation.,” *Nat. Biotechnol.*, vol. 27, no. 8, pp. 735–41, Aug. 2009.

[12] M. Courtot, N. Juty, C. Knüpfer, D. Waltemath, A. Zhukova, A. Dräger, M. Dumontier, A. Finney, M. Golebiewski, J. Hastings, S. Hoops, S. Keating, D. B. Kell, S. Kerrien, J. Lawson, A. Lister, J. Lu, R. Machne, P. Mendes, M. Pocock, N. Rodriguez, A. Villeger, D. J. Wilkinson, S. Wimalaratne, C. Laibe, M. Hucka, and N. Le Novère, "Controlled vocabularies and semantics in systems biology.," *Mol. Syst. Biol.*, vol. 7, no. 543, p. 543, Jan. 2011.

[13] C. Li, 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, "BioModels Database: An enhanced, curated and annotated resource for published quantitative kinetic models.," *BMC Syst. Biol.*, vol. 4, p. 92, Jun. 2010.

[14] W. W. Chen, B. Schoeberl, P. J. Jasper, M. Niepel, U. B. Nielsen, D. a Lauffenburger, and P. K. Sorger, "Input-output behavior of ErbB signaling pathways as revealed by a mass action model trained against dynamic data.," *Mol. Syst. Biol.*, vol. 5, no. 239, p. 239, Jan. 2009.

[15] R. Randhawa, C. a Shaffer, and J. J. Tyson, "Model composition for macromolecular regulatory networks.," *IEEE/ACM Trans. Comput. Biol. Bioinform.*, vol. 7, no. 2, pp. 278–87, 2010.

[16] M. Schulz, F. Krause, N. Le Novère, E. Klipp, and W. Liebermeister, "Retrieval, alignment, and clustering of computational models based on semantic annotations.," *Mol. Syst. Biol.*, vol. 7, no. 512, p. 512, Jan. 2011.

[17] J. H. Gennari, M. L. Neal, M. Galdzicki, and D. L. Cook, "Multiple ontologies in action: Composite annotations for biosimulation models," *J. Biomed. Inform.*, vol. 44, no. 1, pp. 146–154, Feb. 2011.

[18] W. S. Hlavacek, J. R. Faeder, M. L. Blinov, A. S. Perelson, and B. Goldstein, “The complexity of complexes in signal transduction.,” *Biotechnol. Bioeng.*, vol. 84, no. 7, pp. 783–94, Dec. 2003.

[19] N. M. Borisov, N. I. Markevich, J. B. Hoek, and B. N. Kholodenko, “Signaling through receptors and scaffolds: independent interactions reduce combinatorial complexity.,” *Biophys. J.*, vol. 89, no. 2, pp. 951–66, Aug. 2005.

[20] O. S. Soyer and S. Bonhoeffer, “Evolution of complexity in signaling pathways.,” *Proc. Natl. Acad. Sci. U. S. A.*, vol. 103, no. 44, pp. 16337–42, 2006.

[21] A. Steinacher and O. S. Soyer, “Evolutionary Principles Underlying Structure and Response Dynamics of Cellular Networks,” in *Evolutionary Systems Biology*, vol. 751, O. S. Soyer, Ed. New York, NY: Springer New York, 2012, pp. 225–247.

[22] C. A. Shaffer, J. J. Tyson, and R. Randhawa, “Model Composition and Aggregation in Macromolecular Regulatory Networks Ranjit Randhawa Model Composition and Aggregation in Macromolecular Regulatory Networks Ranjit Randhawa Abstract,” *Syntax*, 2008.

[23] W. S. Hlavacek, J. R. Faeder, M. L. Blinov, R. G. Posner, M. Hucka, and W. Fontana, “Rules for modeling signal-transduction systems.,” *Sci. STKE*, vol. 2006, p. re6, 2006.

[24] J. R. Faeder, M. L. Blinov, and W. S. Hlavacek, “Rule-based modeling of biochemical systems with BioNetGen.,” *Methods Mol. Biol.*, vol. 500, no. 2, pp. 113–67, Jan. 2009.

[25] J. A. P. Sekar and J. R. Faeder, “Rule-based modeling of signal transduction: a primer.,” *Methods Mol. Biol.*, vol. 880, pp. 139–218, Jan. 2012.

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

Bioinformatics, vol. 13 Suppl 8, p. S3, Jan. 2012.

[27] V. Danos and C. Laneve, “Formal molecular biology,” *Theor. Comput. Sci.*, vol. 325, no. 1, pp. 69–110, Sep. 2004.

[28] V. Danos, J. Feret, W. Fontana, R. Harmer, and J. Krivine, “Rule-based Modelling of Cellular Signaling,” in *CONCUR 2007 - Concurrency Theory, Lecture Notes in Computer Science*, vol. 4703, 2007, pp. 17–41.

[29] V. Danos, J. Feret, W. Fontana, and J. Krivine, “Scalable Simulation of Cellular Signaling Networks,” in *Programming Languages and Systems, Lecture Notes in Computer Science*, vol. 4807, 2007, pp. 139–157.

[30] M. Meier-Schellersheim, X. Xu, B. Angermann, E. J. Kunkel, T. Jin, and R. N. Germain, “Key role of local regulation in chemosensing revealed by a new molecular interaction-based modeling method,” *PLoS Comput. Biol.*, vol. 2, pp. 0710–0724, 2006.

[31] F. Zhang, B. R. Angermann, and M. Meier-Schellersheim, “The Simmune Modeler visual interface for creating signaling networks based on bi-molecular interactions.,” *Bioinformatics*, vol. 29, no. 9, pp. 1229–30, May 2013.

[32] H.-C. Cheng, B. R. Angermann, F. Zhang, and M. Meier-Schellersheim, “NetworkViewer: visualizing biochemical reaction networks with embedded rendering of molecular interaction rules.,” *BMC Syst. Biol.*, vol. 8, p. 70, Jan. 2014.

[33] M. Blinov, J. Yang, J. Faeder, and W. S. Hlavacek, “Graph Theory for Rule-Based Modeling of Biochemical Networks,” *Trans. Comput. Syst. Biol. VII*, vol. 4230, pp. 89–106, 2006.

[34] J. S. Hogg, L. A. Harris, L. J. Stover, N. S. Nair, and J. R. Faeder, “Exact hybrid particle/population simulation of rule-based models of biochemical systems.,” *PLoS*

Comput. Biol., vol. 10, no. 4, p. e1003544, Apr. 2014.

[35] J. R. Faeder, W. S. Hlavacek, I. Reischl, M. L. Blinov, H. Metzger, A. Redondo, C. Wofsy, and B. Goldstein, “Investigation of early events in Fc epsilon RI-mediated signaling using a detailed mathematical model.,” *J. Immunol.*, vol. 170, no. 7, pp. 3769–81, Apr. 2003.

[36] M. L. Blinov, J. R. Faeder, B. Goldstein, and W. S. Hlavacek, “A network model of early events in epidermal growth factor receptor signaling that accounts for combinatorial complexity,” in *BioSystems*, 2006, vol. 83, no. 2–3 SPEC. ISS., pp. 136–151.

[37] O. Dushek, P. A. Van Der Merwe, and V. Shahrezaei, “Ultrasensitivity in multisite phosphorylation of membrane-anchored proteins,” *Biophys. J.*, vol. 100, no. 5, pp. 1189–1197, 2011.

[38] J. Colvin, M. I. Monine, J. R. Faeder, W. S. Hlavacek, D. D. Von Hoff, and R. G. Posner, “Simulation of large-scale rule-based models.,” *Bioinformatics*, vol. 25, no. 7, pp. 910–7, Apr. 2009.

[39] M. W. Sneddon, J. R. Faeder, and T. Emonet, “Efficient modeling, simulation and coarse-graining of biological complexity with NFsim.,” *Nat. Methods*, vol. 8, no. 2, Dec. 2010.

[40] J. Yang, M. Monine, J. Faeder, and W. Hlavacek, “Kinetic Monte Carlo method for rule-based modeling of biochemical networks,” *Phys. Rev. E*, vol. 78, no. 3, pp. 1–7, Sep. 2008.

[41] M. I. Monine, R. G. Posner, P. B. Savage, J. R. Faeder, and W. S. Hlavacek, “Modeling multivalent ligand-receptor interactions with steric constraints on configurations of cell-surface receptor aggregates.,” *Biophys. J.*, vol. 98, no. 1, pp. 48–56, Jan. 2010.

[42] A. Nag, M. I. Monine, J. R. Faeder, and B. Goldstein, “Aggregation of membrane proteins by cytosolic cross-linkers: theory and simulation of the LAT-Grb2-SOS1 system.,” *Biophys. J.*, vol. 96, no. 7, pp. 2604–23, Apr. 2009.

[43] P. J. Michalski and L. M. Loew, "CaMKII activation and dynamics are independent of the holoenzyme structure: an infinite subunit holoenzyme approximation.," *Phys. Biol.*, vol. 9, no. 3, p. 036010, Jun. 2012.

[44] L. A. Chylek, D. A. Holowka, B. A. Baird, and W. S. Hlavacek, "An interaction library for the FcεRI signaling network," *Front. Immunol.*, vol. 5, no. APR, p. 172, Jan. 2014.

[45] L. A. Chylek, V. Akimov, J. Dengjel, K. T. G. Rigbolt, B. Hu, W. S. Hlavacek, and B. Blagoev, "Phosphorylation site dynamics of early T-cell receptor signaling.," *PLoS One*, vol. 9, no. 8, p. e104240, 2014.

[46] M. S. Creamer, 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, and R. G. Posner, "Specification, annotation, visualization and simulation of a large rule-based model for ERBB receptor signaling.," *BMC Syst. Biol.*, vol. 6, no. 1, p. 107, Jan. 2012.

[47] T. M. Thomson, K. R. Benjamin, A. Bush, T. Love, D. Pincus, O. Resnekov, R. C. Yu, A. Gordon, A. Colman-Lerner, D. Endy, and R. Brent, "Scaffold number in yeast signaling system sets tradeoff between system output and dynamic range," *Proc. Natl. Acad. Sci.*, vol. 108, no. 50, pp. 20265–20270, 2011.

[48] J. R. Karr, J. C. Sanghvi, D. N. Macklin, M. V Gutschow, J. M. Jacobs, B. Bolival, N. Assad-Garcia, J. I. Glass, and M. W. Covert, "A whole-cell computational model predicts phenotype from genotype.," *Cell*, vol. 150, no. 2, pp. 389–401, Jul. 2012.

[49] C. F. Lopez, J. L. Muhlich, J. A. Bachman, and P. K. Sorger, "Programming biological models in Python using PySB," *Mol. Syst. Biol.*, vol. 9, Feb. 2013.

[50] J. Hogg, "Advances in rule-based modeling: Compartments, energy, and hybrid simulation, with application to sepsis and cell signaling.," 2013.

[51] V. Danos, R. Harmer, and R. Honorato-Zimmer, “Thermodynamic graph-rewriting,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2013, vol. 8052 LNCS, pp. 380–394.

[52] J. F. Ollivier, V. Shahrezaei, and P. S. Swain, “Scalable rule-based modelling of allosteric proteins and biochemical networks.,” *PLoS Comput. Biol.*, vol. 6, no. 11, p. e1000975, Jan. 2010.

[53] P. Saraiya, C. North, and K. Duca, “Visualizing biological pathways: requirements analysis, systems evaluation and research agenda,” *Inf. Vis.*, vol. 4, no. 3, pp. 191–205, 2005.

[54] E. R. Tufte, *The Visual Display of Quantitative Information*. Graphics Press, 1986.

[55] H. Gibson, J. Faith, and P. Vickers, “A survey of two-dimensional graph layout techniques for information visualisation,” *Inf. Vis.*, vol. 12, no. 3–4, pp. 324–357, 2012.

[56] K. W. Kohn, “Molecular interaction map of the mammalian cell cycle control and DNA repair systems.,” *Mol. Biol. Cell*, vol. 10, no. 8, pp. 2703–34, Aug. 1999.

[57] K. W. Kohn, “Molecular interaction maps as information organizers and simulation guides.,” *Chaos*, vol. 11, no. 1, pp. 84–97, Mar. 2001.

[58] K. W. Kohn, M. I. Aladjem, S. Kim, J. N. Weinstein, and Y. Pommier, “Depicting combinatorial complexity with the molecular interaction map notation.,” *Mol. Syst. Biol.*, vol. 2, p. 51, Jan. 2006.

[59] L. a Chylek, B. Hu, M. L. Blinov, T. Emonet, J. R. Faeder, B. Goldstein, R. N. Gutenkunst, J. M. Haugh, T. Lipniacki, R. G. Posner, J. Yang, and W. S. Hlavacek, “Guidelines for visualizing and annotating rule-based models.,” *Mol. Biosyst.*, vol. 7, no. 10, pp. 2779–2795, Oct. 2011.

[60] M. P. Van Iersel, A. C. Villéger, T. Czauderna, S. E. Boyd, F. T. Bergmann, A. Luna, E.

Demir, A. Sorokin, U. Dogrusoz, Y. Matsuoka, A. Funahashi, M. I. Aladjem, H. Mi, S. L. Moodie, H. Kitano, N. Le novère, and F. Schreiber, "Software support for SBGN maps: SBGN-ML and LibSBGN," *Bioinformatics*, vol. 28, no. 15, pp. 2016–2021, 2012.

[61] M. Sari, I. Bahceci, U. Dogrusoz, S. O. Sumer, B. A. Aksoy, Ö. Babur, and E. Demir, "SBGNViz: A Tool for Visualization and Complexity Management of SBGN Process Description Maps.," *PLoS One*, vol. 10, no. 6, p. e0128985, 2015.

[62] E. Demir, Ö. Babur, I. Rodchenkov, B. A. Aksoy, K. I. Fukuda, B. Gross, O. S. Sümer, G. D. Bader, and C. Sander, "Using Biological Pathway Data with Paxtools," *PLoS Comput. Biol.*, vol. 9, no. 9, pp. 1–5, 2013.

[63] Y. Matsuoka, A. Funahashi, S. Ghosh, and H. Kitano, "Modeling and simulation using CellDesigner.," *Methods Mol. Biol.*, vol. 1164, pp. 121–45, 2014.

[64] E. Gonçalves, M. van Iersel, and J. Saez-Rodriguez, "CySBGN: a Cytoscape plug-in to integrate SBGN maps.," *BMC Bioinformatics*, vol. 14, p. 17, 2013.

[65] C.-F. Tiger, F. Krause, G. Cedersund, R. Palmér, E. Klipp, S. Hohmann, H. Kitano, and M. Krantz, "A framework for mapping, visualisation and automatic model creation of signal-transduction networks.," *Mol. Syst. Biol.*, vol. 8, no. 578, p. 578, Jan. 2012.

[66] T. Vogt, T. Czauderna, and F. Schreiber, "Translation of SBGN maps: Process Description to Activity Flow.," *BMC Syst. Biol.*, vol. 7, p. 115, Jan. 2013.

[67] M. L. Blinov, J. R. Faeder, B. Goldstein, and W. S. Hlavacek, "BioNetGen: Software for rule-based modeling of signal transduction based on the interactions of molecular domains," *Bioinformatics*, vol. 20, no. 17, pp. 3289–3291, Nov. 2004.

[68] V. Danos and C. Laneve, "Formal molecular biology," in *Theoretical Computer Science*, 2004, vol. 325, pp. 69–110.

[69] I. I. Moraru, J. C. Schaff, B. M. Slepchenko, M. L. Blinov, F. Morgan, A. Lakshminarayana, F. Gao, Y. Li, and L. M. Loew, "Virtual Cell modelling and simulation software environment.," *IET Syst. Biol.*, vol. 2, no. 5, pp. 352–62, Sep. 2008.

[70] M. L. Blinov, J. R. Faeder, B. Goldstein, and W. S. Hlavacek, "A network model of early events in epidermal growth factor receptor signaling that accounts for combinatorial complexity.," *Biosystems.*, vol. 83, no. 2–3, pp. 136–51, 2006.

[71] V. Danos, J. Feret, W. Fontana, R. Harmer, J. Hayman, J. Krivine, C. Thompson-Walsh, and G. Winskel, "Graphs, Rewriting and Pathway Reconstruction for Rule-Based Models," in *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2012)*, 2012, vol. 18, pp. 276–288.

[72] F. Büchel, N. Rodriguez, N. Swainston, C. Wrzodek, T. Czauderna, R. Keller, F. Mittag, M. Schubert, M. Glont, M. Golebiewski, M. van Iersel, S. Keating, M. Rall, M. Wybrow, H. Hermjakob, M. Hucka, D. B. Kell, W. Müller, P. Mendes, A. Zell, C. Chaouiya, J. Saez-Rodriguez, F. Schreiber, C. Laibe, A. Dräger, and N. Le Novère, "Path2Models: large-scale generation of computational models from biochemical pathway maps.," *BMC Syst. Biol.*, vol. 7, p. 116, 2013.

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

[74] N. W. Lemons, B. Hu, and W. S. Hlavacek, "Hierarchical graphs for rule-based modeling of biochemical systems.," *BMC Bioinformatics*, vol. 12, p. 45, 2011.

[75] M. Ghoniem, J.-D. Fekete, and P. Castagliola, "On the readability of graphs using node-link and matrix-based representations: a controlled experiment and statistical analysis," *Inf.*

Vis., vol. 4, no. 2, pp. 114–135, 2005.

[76] D. Barua, W. S. Hlavacek, and T. Lipniacki, “A computational model for early events in B cell antigen receptor signaling: analysis of the roles of Lyn and Fyn.,” *J. Immunol.*, vol. 189, no. 2, pp. 646–58, 2012.

[77] L. A. Chylek, B. S. Wilson, and W. S. Hlavacek, “Modeling Biomolecular Site Dynamics in Immunoreceptor Signaling Systems,” in *A Systems Biology Approach to Blood*, M. Orešič and A. Vidal-Puig, Eds. Cham: Springer International Publishing, 2014, pp. 245–262.

[78] D. Barua, J. R. Faeder, and J. M. Haugh, “Structure-based kinetic models of modular signaling protein function: focus on Shp2.,” *Biophys. J.*, vol. 92, no. 7, pp. 2290–300, Apr. 2007.

[79] M. L. Blinov, J. R. Faeder, B. Goldstein, and W. S. Hlavacek, “A network model of early events in epidermal growth factor receptor signaling that accounts for combinatorial complexity.,” *Biosystems.*, vol. 83, no. 2–3, pp. 136–51, 2006.

[80] P. Kocieniewski, J. R. Faeder, and T. Lipniacki, “The interplay of double phosphorylation and scaffolding in MAPK pathways.,” *J. Theor. Biol.*, vol. 295, pp. 116–24, Feb. 2012.

[81] M. König and H.-G. Holzhütter, “Fluxviz - Cytoscape plug-in for visualization of flux distributions in networks.,” *Genome Inform.*, vol. 24, pp. 96–103, 2010.

[82] Z. Hu, Y. C. Chang, Y. Wang, C. L. Huang, Y. Liu, F. Tian, B. Granger, and C. Delisi, “VisANT 4.0: Integrative network platform to connect genes, drugs, diseases and therapies.,” *Nucleic Acids Res.*, vol. 41, no. Web Server issue, 2013.

[83] J. Von Neumann, “13. Various Techniques Used in Connection With Random Digits,” 1951.

[84] C. F. Lopez, J. L. Muhlich, J. A. Bachman, and P. K. Sorger, “Programming biological

models in Python using PySB.," *Mol. Syst. Biol.*, vol. 9, p. 646, Jan. 2013.

[85] V. Danos, J. Feret, W. Fontana, R. Harmer, and J. Krivine, "Rule-Based Modelling and Model Perturbation," in *Transactions on Computational Systems Biology XI*, vol. 5750, C. Priami, R.-J. Back, and I. Petre, Eds. Springer Berlin / Heidelberg, 2009, pp. 116–137.

[86] E. C. Stites, P. C. Trampont, Z. Ma, and K. S. Ravichandran, "Network analysis of oncogenic Ras activation in cancer.," *Science*, vol. 318, no. 5849, pp. 463–7, Oct. 2007.

[87] B. N. Kholodenko, O. V Demin, G. Moehren, and J. B. Hoek, "Quantification of short term signaling by the epidermal growth factor receptor.," *J. Biol. Chem.*, vol. 274, no. 42, pp. 30169–81, Oct. 1999.

[88] B. Schoeberl, C. Eichler-Jonsson, E. D. Gilles, and G. Müller, "Computational modeling of the dynamics of the MAP kinase cascade activated by surface and internalized EGF receptors.," *Nat. Biotechnol.*, vol. 20, no. 4, pp. 370–5, Apr. 2002.

[89] C. Vehlow, F. Beck, and D. Weiskopf, "The State of the Art in Visualizing Group Structures in Graphs," in *Eurographics Conference on Visualization (EuroVis)*, 2015.

[90] R. N. Gutenkunst, J. J. Waterfall, F. P. Casey, K. S. Brown, C. R. Myers, and J. P. Sethna, "Universally sloppy parameter sensitivities in systems biology models.," *PLoS Comput. Biol.*, vol. 3, no. 10, pp. 1871–78, Oct. 2007.

[91] R. N. Gutenkunst, F. P. Casey, J. J. Waterfall, C. R. Myers, and J. P. Sethna, "Extracting falsifiable predictions from sloppy models.," *Ann. N. Y. Acad. Sci.*, vol. 1115, pp. 203–11, Dec. 2007.

[92] F. P. Casey, D. Baird, Q. Feng, R. N. Gutenkunst, J. J. Waterfall, C. R. Myers, K. S. Brown, R. A. Cerione, and J. P. Sethna, "Optimal experimental design in an epidermal growth factor receptor signalling and down-regulation model," *IET Syst. Biol.*, vol. 1, no. 3, p. 190, 2007.

[93] J. F. Apgar, D. K. Witmer, F. M. White, and B. Tidor, “Sloppy models, parameter uncertainty, and the role of experimental design.,” *Mol. Biosyst.*, vol. 6, no. 10, pp. 1890–900, Oct. 2010.

[94] D. J. Klinke, “An empirical Bayesian approach for model-based inference of cellular signaling networks.,” *BMC Bioinformatics*, vol. 10, p. 371, Jan. 2009.

[95] H. Eydgahi, W. W. Chen, J. L. Muhlich, D. Vitkup, J. N. Tsitsiklis, and P. K. Sorger, “Properties of cell death models calibrated and compared using Bayesian approaches.,” *Mol. Syst. Biol.*, vol. 9, p. 644, Feb. 2013.

[96] R. E. Kass and A. E. Raftery, “Bayes factors,” *J. Am. Stat. Assoc.*, vol. 90, no. 430, pp. 773–795, 1995.