Extending Regulatory Network Modeling with Multistate Species

Umme Juka Mobassera

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Masters
in
Computer Science and Applications

Clifford A. Shaffer, Chair
John J. Tyson, Co-Chair
Yang Cao

October 31, 2011
Blacksburg, Virginia

Keywords: Computational Systems Biology, Multistate Species, Rule Based Modeling,
SBML, Modeling Tool, Software, JigCell

Extending Regulatory Network Modeling with Multistate Species

Umme Juka Mobassera

(ABSTRACT)

By increasing the level of abstraction in the representation of regulatory network models, we can hope to allow modelers to create models that are beyond the threshold of what can currently be expressed reliably. As hundreds of reactions are difficult to understand, maintain, and extend, thousands of reactions become next to impossible without any automation or aid. Using the multistate-species concept we can reduce the number of reactions needed to represent certain systems and thus, lessen the cognitive load on modelers.

A multistate species is an entity with a defined range for state variables, which refers to a group of different forms for a specific species. A multistate reaction involves one or more multistate species and compactly represents a group of similar single reactions. In this work, we have extended JCMB (the JigCell Model Builder) to comply with multistate species and reactions modeling and presented a proposal for enhancing SBML (the Systems Biology Markup Language) standards to support multistate models.

Acknowledgments

I would like to express my appreciation for people who are, directly or indirectly, part of this work.

First of all, I would like to give thanks to Dr. Clifford A. Shaffer, the chair of my committee, for his invaluable guidance all through this work. Specifically, I want to mention about his suggestions in the writing process. He reviewed the drafts thoroughly each time and helped me to correct even the small mistakes. I would like to convey my gratitude to Dr. John Tyson, the co-chair of my committee, for being the caring boss. He managed time for me when required from his busy schedule and made me understand the biological models and the vision of the project. I am grateful to Dr. Yang Cao for the advice and suggestions he gave time to time for the improvement of this work.

Beside my committee, I would like to thank Dr. Debashis Barik for providing me with introductory materials regarding multistate modeling and for helping me to collect the perspectives of modelers for the user interface design. Dr. Kathy Chen helped me to understand the features of the existing software and tools. Whenever I went to her, even for silly problems, she spent from her busy time to explain that to me with patience. Thank you Kathy. I also like to give thanks to all the members of the Tyson Lab for being so supportive colleagues.

I want to mention about the financial support I received for this work from the NIH under Grant R01GM078989.

I also want to take the opportunity here to express my gratitude for my parents and other family members who have supported me with positive inspiration all through my life. I want to mention my grand father who gave me encouragement for pursuing higher studies. And last but not the least, my gratitude to the Bangladeshi students and families in Blacksburg who became a family away from family for me through their endless support.

Dedication

For the most precious gift I ever got - ***Maheer***

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The physiology of a living cell is governed by underlying networks of interacting macromolecules such as genes, mRNAs and proteins. Modelers of biochemical systems often work with hundreds of reactions representing the interactions of dozens of different species and parameters. Many tools have been developed to make the task of modeling and editing faster and less error prone. These tools allow modelers to create models with hundreds of reactions, which is beyond their ability to construct by hand. By varying the level of abstraction, we hope that such models can be expressed in ways that preserve the information, while reducing cognitive load on the modeler.

A common theme in intra-cellular regulatory networks is multi-state species, such as, a protein with multiple states. A multistate species can change its binding and reacting properties based on changes in one or many of its binding sites by phosphorylation or methylation, for example. Multi-state species are frequently present in biological systems, so modeling multistate species properly is important for those biochemical models.

A multistate species with n distinct binding sites can have 2^n different states if every combination of bindings is a separate state, so, a species with 5 binding sites may exhibit $2^5 = 32$ different binding combinations. Depending on the number of sites and the number of states or values these binding sites can take, the possible number of species' states can become very large and impractical to represent explicitly. Additional complexity results when multiple multistate species in a model combine in many different ways to make complexes. Defining and maintaining such a large system gives the modelers more work and increases the complexity of the resulting simulation. For example, in the tyrosine kinase receptor EGFR multiple (atleast nine) tyrosines are phosphorylated during signaling [13]. As a result, there are possibly 2^9 or 512 different forms of EGFR and more than 100K distinct combinations or dimers. Ofcourse the actual number of relevant phospho-forms of EGFR is much lower, however, the whole set of forms may create interest for researchers.

A different issue comes into the picture when some states appear often in reactions and some do not appear at all. This might happen when some of the possible phosphorylation sites of a multistate species are not used. Obviously modelers would like to get a way to effectively represent the used states when required with less labor and time. This indicates

the necessity for defining the species compactly but using enumeration for referring to a set of states when needed.

Modeling a system is not a one-time task. Models require frequent editing, and ideally will be reused in the future. The bigger that models become, the more complex and difficult they are to view, handle and use. By managing as much of the bookkeeping as possible, automated tools can make modeling easier and faster for modelers.

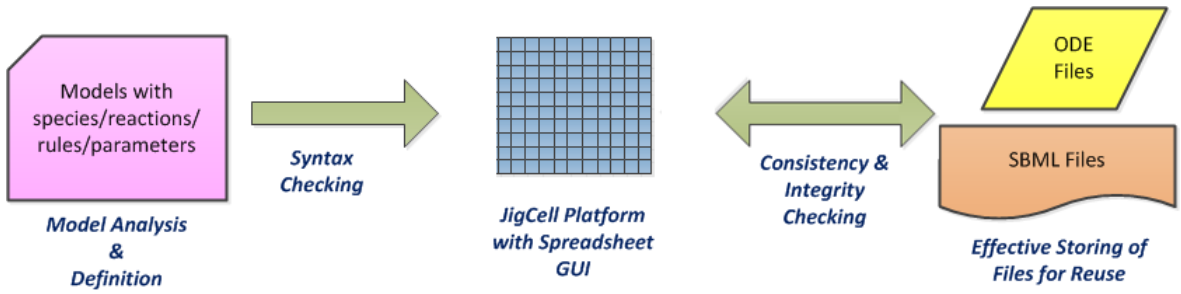

Figure 1.1: Modeling of multistate reactions with JigCell and SBML

JigCell [12] is a collection of software tools for modeling biochemical networks. We seek to enhance the JigCell ModelBuilder (JCMB) to support the multistate modeling aspect of the SBML [22] standard. The Systems Biology Markup Language or SBML is a XML-based format for storing and representing biochemical models. SBML comprises many different constructs or structures to represent different biological elements, and it allows storing and communicating among computational models of biological processes with free, open, and widespread software support and a community of users and developers.

We have added to JCMB a convenient way to define multistate species and their interactions, and tested our efforts with real examples from the published literature, with the expectation to have an insight into the core modeling mechanism (Figure 1.1). We have revised the existing SBML proposals for multistate species modeling which we amend to support a more general representation. In this work, we consider four aspects of multistate species and reaction modeling:

- Study and analyze real biological models involving multistate reactions
- Standardize the syntax for expressing and storing these models
- Check the consistency and integrity of the multistate part in overall modeling
- Create a software platform for multistate models and propose a way to reuse them in existing standard tools

In Chapter 2, multistate species and reactions are presented using a small model to show the impact and advantages of multistate modeling. In Chapter 3, we analyze the literature, standards and tools related to multistate modeling. Chapter 4 presents the syntax for defining multistate species and reactions. Chapter 5 gives an example of multistate modeling with a full model. We identify and discuss the multistate features of the model and how we can practically represent them in a compact way. In Chapter 6, the JCMB environment and

SBML standards for multistate modeling are discussed. Chapter 7 discusses future plans, describing limitations of the current work and ways to overcome those.

Chapter 2

A Small Example of a Multistate System

In this chapter a small model (part of the model discussed in [1]) is presented. It illustrates the multistate reaction concept and our modeling approach. Figure 2.1 shows the core reactions of the model, representing mutual antagonism between Clb2 and Cdh1. In this model, dependence of Cdh1 on the ratio of Clb2 kinase vs Cdc14 phosphatase activity is crucial for the phosphorylation stages.

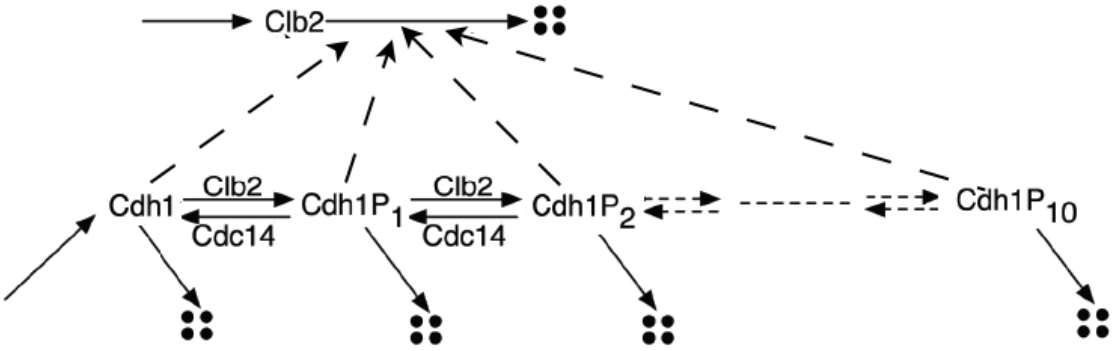

Figure 2.1: The Clb2-Cdh1 *Antagonism Model*

In this model, we have a total of 11 forms or states for Cdh1: a non-phosphorylated form (Cdh1) and 10 phosphorylated forms denoted as $Cdh1P_1$, $Cdh1P_2$, ..., $Cdh1P_{10}$. These forms take part in the phosphorylation and dephosphorylation reactions as reactant or product. All of the eleven forms of Cdh1 take part in the degradation of Clb2. In this model there are 34 single-state reactions in total if each phosphorylation state is considered as a distinct species (Figure 2.2).

We can define Cdh1 as a multistate species with a state range represented by indices, such as, Cdh1P{i} with $0 \leq i \leq 10$. Using this compact notation, we can reduce the number of

SL	Reactions	Rate law	Equation
1	$\rightarrow$ Clb2	Mass action	k_s
2	Clb2 $\rightarrow$	Mass action	Clb2.Xa
3	Cdh1+Clb2 $\rightarrow$ Clb2+Cdh1P$_1$	Mass action	k_p.Clb2.Cdh1
4	Cdh1P$_1$+Clb2 $\rightarrow$ Clb2+Cdh1P$_2$	Mass action	k_p.Clb2.Cdh1P$_1$
5	Cdh1P$_2$+Clb2 $\rightarrow$ Clb2+Cdh1P$_3$	Mass action	k_p.Clb2.Cdh1P$_2$
6	Cdh1P$_3$+Clb2 $\rightarrow$ Clb2+Cdh1P$_4$	Mass action	k_p.Clb2.Cdh1P$_3$
7	Cdh1P$_4$+Clb2 $\rightarrow$ Clb2+Cdh1P$_5$	Mass action	k_p.Clb2.Cdh1P$_4$
8	Cdh1P$_5$+Clb2 $\rightarrow$ Clb2+Cdh1P$_6$	Mass action	k_p.Clb2.Cdh1P$_5$
9	Cdh1P$_6$+Clb2 $\rightarrow$ Clb2+Cdh1P$_7$	Mass action	k_p.Clb2.Cdh1P$_6$
10	Cdh1P$_7$+Clb2 $\rightarrow$ Clb2+Cdh1P$_8$	Mass action	k_p.Clb2.Cdh1P$_7$
11	Cdh1P$_8$+Clb2 $\rightarrow$ Clb2+Cdh1P$_9$	Mass action	k_p.Clb2.Cdh1P$_8$
12	Cdh1P$_9$+Clb2 $\rightarrow$ Clb2+Cdh1P$_{10}$	Mass action	k_p.Clb2.Cdh1P$_9$
13	Cdh1P$_1$+Cdc14 $\rightarrow$ Cdc14+Cdh1	Mass action	k_h.Cdc14.Cdh1P$_1$
14	Cdh1P$_2$+Cdc14 $\rightarrow$ Cdc14+Cdh1P$_1$	Mass action	k_h.Cdc14.Cdh1P$_2$
15	Cdh1P$_3$+Cdc14 $\rightarrow$ Cdc14+Cdh1P$_2$	Mass action	k_h.Cdc14.Cdh1P$_3$
16	Cdh1P$_4$+Cdc14 $\rightarrow$ Cdc14+Cdh1P$_3$	Mass action	k_h.Cdc14.Cdh1P$_4$
17	Cdh1P$_5$+Cdc14 $\rightarrow$ Cdc14+Cdh1P$_4$	Mass action	k_h.Cdc14.Cdh1P$_5$
18	Cdh1P$_6$+Cdc14 $\rightarrow$ Cdc14+Cdh1P$_5$	Mass action	k_h.Cdc14.Cdh1P$_6$
19	Cdh1P$_7$+Cdc14 $\rightarrow$ Cdc14+Cdh1P$_6$	Mass action	k_h.Cdc14.Cdh1P$_7$
20	Cdh1P$_8$+Cdc14 $\rightarrow$ Cdc14+Cdh1P$_7$	Mass action	k_h.Cdc14.Cdh1P$_8$
21	Cdh1P$_9$+Cdc14 $\rightarrow$ Cdc14+Cdh1P$_8$	Mass action	k_h.Cdc14.Cdh1P$_9$
22	Cdh1P$_{10}$+Cdc14 $\rightarrow$ Cdc14+Cdh1P$_9$	Mass action	k_h.Cdc14.Cdh1P$_{10}$
23	Cdh1 $\rightarrow$	Mass action	k_d.Cdh1
24	Cdh1P$_1$ $\rightarrow$	Mass action	k_d.Cdh1P$_1$
25	Cdh1P$_2$ $\rightarrow$	Mass action	k_d.Cdh1P$_2$
26	Cdh1P$_3$ $\rightarrow$	Mass action	k_d.Cdh1P$_3$
27	Cdh1P$_4$ $\rightarrow$	Mass action	k_d.Cdh1P$_4$
28	Cdh1P$_5$ $\rightarrow$	Mass action	k_d.Cdh1P$_5$
29	Cdh1P$_6$ $\rightarrow$	Mass action	k_d.Cdh1P$_6$
30	Cdh1P$_7$ $\rightarrow$	Mass action	k_d.Cdh1P$_7$
31	Cdh1P$_8$ $\rightarrow$	Mass action	k_d.Cdh1P$_8$
32	Cdh1P$_9$ $\rightarrow$	Mass action	k_d.Cdh1P$_9$
33	Cdh1P$_{10}$ $\rightarrow$	Mass action	k_d.Cdh1P$_{10}$
34	$\rightarrow$ Cdh1	Mass action	k_{s1}

Figure 2.2: List of reactions in the *Antagonism Model* using single state species and reactions

#	Reaction	Type	Equation
1	-> Clb2	Mass Action	ks
2	Clb2 ->	Mass Action	Xa * Clb2
3	Cdh1P{i} + Clb2 -> Clb2 + Cdh1P{i+1} ; 0 <= i <= 9	multistate - Ordered - Mass Action	kp * Cdh1P{i} * Clb2
4	Cdh1P{i} + Cdc14 -> Cdh1P{i-1} + Cdc14 ; 1 <= i <= 10	multistate - Ordered - Mass Action	kh * Cdh1P{i} * Cdc14
5	Cdh1P{i} -> ; 0 <= i <= 10	multistate - Mass Action	kd * Cdh1P{i}
6	-> Cdh1P{0}	Mass Action	ks1

Figure 2.3: Reactions in the compact multistate format in the *Antagonism Model*

reactions to 5 by expressing the reactants and products as multistate species (Figure 2.3). In the compact format, Reaction 3 represents original reactions 3-12 (phosphorylation of Cdh1), Reaction 4 represents original reactions 13-22 (dephosphorylation of Cdh1), and Reaction 5 represents original reactions 23-33 (degradation of Cdh1). Reaction 6 represents the synthesis of the unphosphorylated form of Cdh1 (original reaction 34).

Variable	Type	Equation
Xa	Assignment	K_a * Cdh1 + K_i *($Cdh1P_1$ + $Cdh1P_2$ + $Cdh1P_3$ + $Cdh1P_4$ + $Cdh1P_5$ + $Cdh1P_6$ + $Cdh1P_7$ + $Cdh1P_8$ + $Cdh1P_9$ + $Cdh1P_{10}$)

Figure 2.4: The variable *Xa* is assigned the expression for activity of Cdh1 in the *Antagonism Model*

In the model (Reaction 2 in Figure 2.2), *Xa* is the activity of Cdh1 expressed as the weighted sum of its phosphorylated and un-phosphorylated forms. Figure 2.4 shows the expression assigned to *Xa*. We need a way to represent this sort of expression in the multistate format. Equation 2.1 shows how we can use the summation operator to make the assignment expression easy to view, manipulate and edit. In Chapters 4 and 6, we discuss the represention of the *sum()* operator in SBML and how it is used that in JCMB, respectively.

$$X_a = k_a * Cdh1 + k_i * \sum_{i=1}^{10} Cdh1P_i \tag{2.1}$$

At present, there is no standard mechanism or tool that uses the multistate format directly, for example in simulation. SBML also does not provide a way to express multistate reactions that is acceptable and usable by all related tools. JCMB therefore provides a translator to expand the compact (multistate) model into a single-state species model for use with other SBML-compliant tools.

Chapter 3

Multistate Modeling: Background and Related Tools

In this chapter we survey the literature for prior efforts to represent multistate reactions.

3.1 The SBML Standard and the *multi* package

The SBML Level 3 core specification [23] is based on a modular approach that allows features and proposals supplemented to the core to be supported as additional packages. In SBML Level 3, multistate and multi-compartment species are described using the package *multi*[24] which , at present, is in the proposal phase. Multi also supports creation of complexes made up of different components.

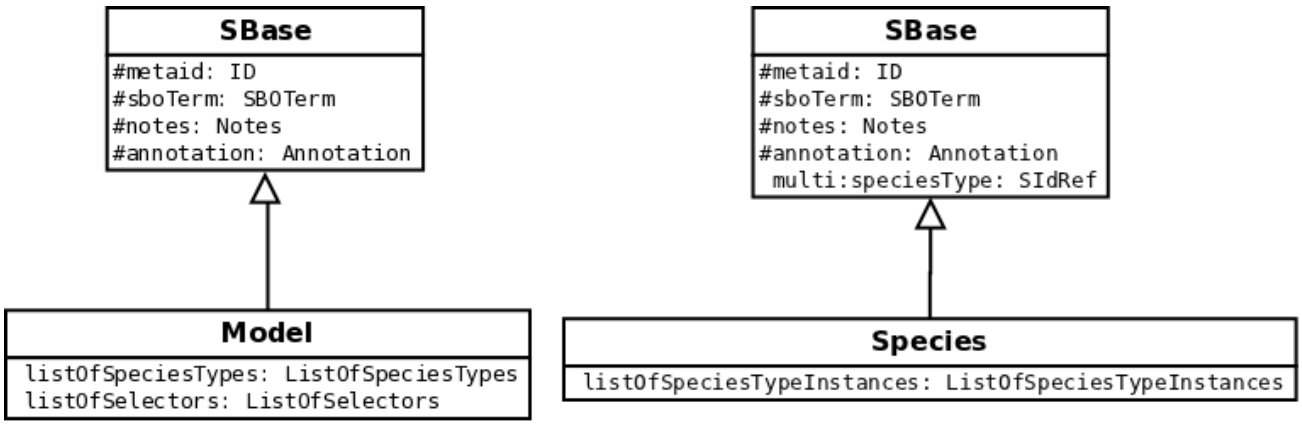

Figure 3.1: The SBML <Model> and <Species> constructs in the *multi* extension

The proposed SBML level-3 *multi* package, comprising of new <SpeciesType> and <Selector> constructs, is supposed to capture and represent the multistate modeling concept. Figure 3.1, 3.2, and 3.3 show the major constructs and modifications proposed in this

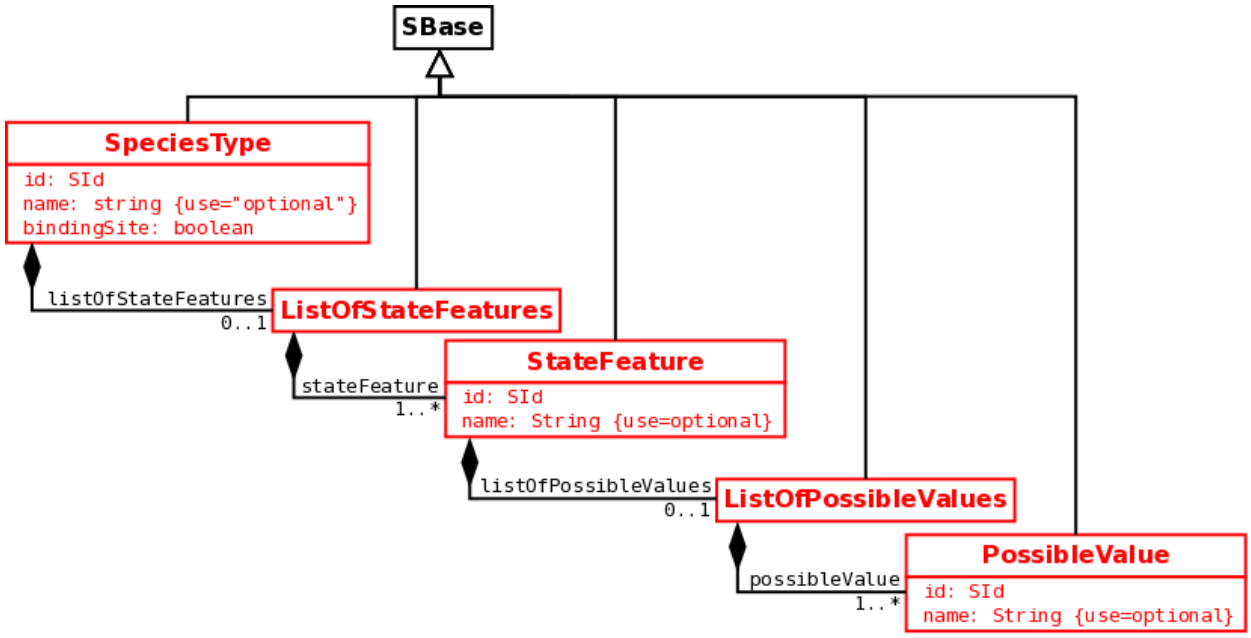

Figure 3.2: The SBML <SpeciesType> construct in the *multi* extension

package. In addition, *multi* has proposed modifications to some existing constructs of the SBML core, such as, <Species>, <SimpleSpeciesReference>, <Rule>, <Reaction> etc.

A <SpeciesType> can represent a number of state features or sites under one list of state features. Each <StateFeature> is identified by a *SId* and an optional name, and contains a list of <PossibleValue>. The values are possibly strings expressing the values or conditions of the state (for example, possible values of a state can be something like bound, free, or closed). A <SpeciesType> also has a boolean attribute <bindingSite> to describe multi-compartment entities.

A <Selector> is a mask describing the rules that a species or any other entity has to pass in order to be used or rejected in a set of reactions. A selector contains one or more <SpeciesTypeState>. A <SpeciesTypeState> contains optional links to a list of <StateFeatureInstances> and a list of <ContainedSpeciesTypes>. From Figure 3.3 we can see that each <StateFeatureInstances> contains a list of allowed possible values as <StateFeatureValue>.

Another important construct in the SBML core is the <SimpleSpeciesReference> which contains a <SIdRef> referring to the corresponding species. The <Reaction> construct represents the reactants and products using <SimpleSpeciesReference>. In multistate and multi-compartment modeling, a species reference instance needs to know the set of species instances it should contain, based on certain restriction and selection criteria. To get this additional effect, in *multi* the simple species reference construct has been modified to carry a link to the list of <SpeciesTypeRestriction>, which eventually points to an instance of the

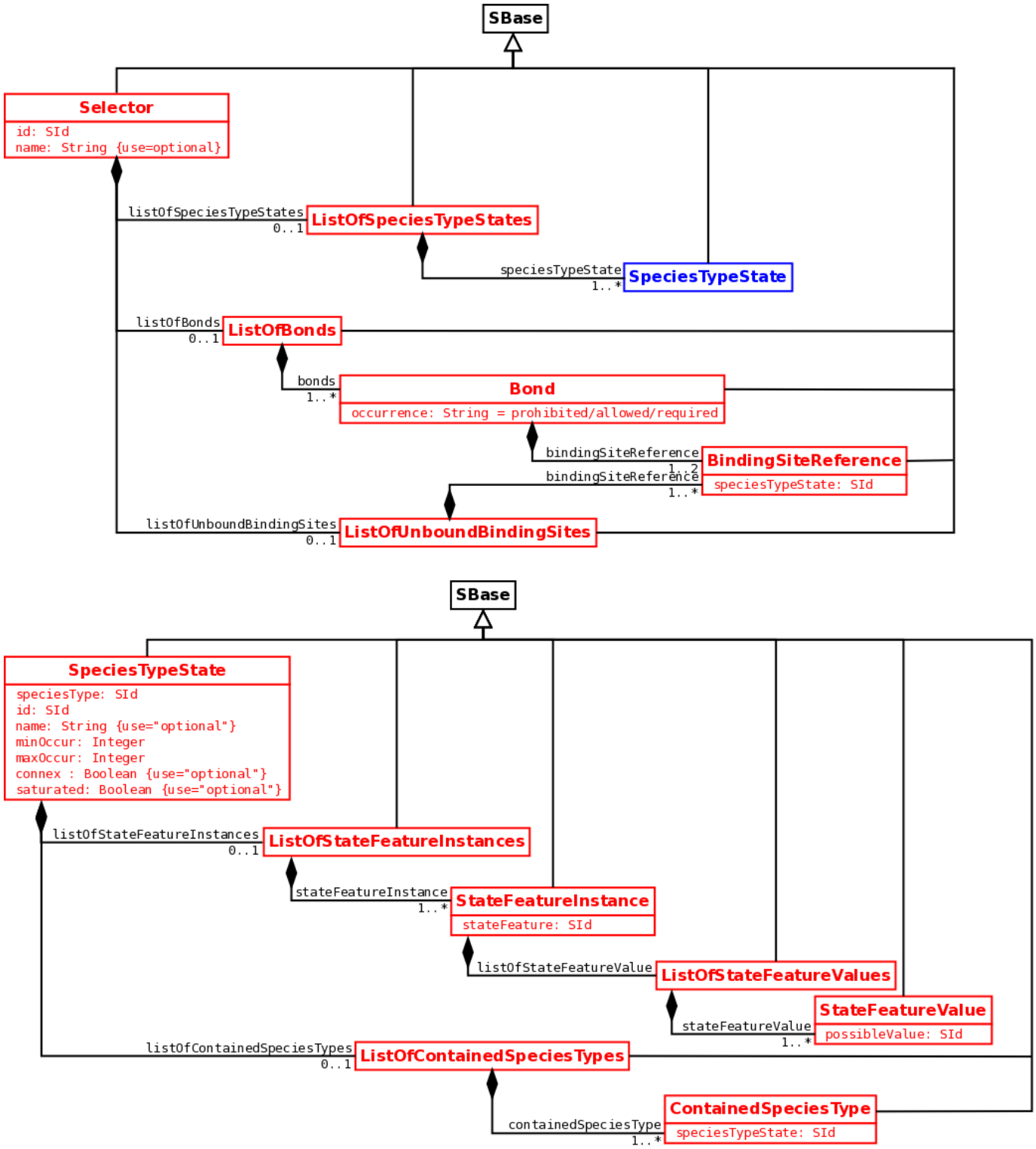

Figure 3.3: The SBML <Selector> and <SpeciesTypeState> constructs in the *multi* extension

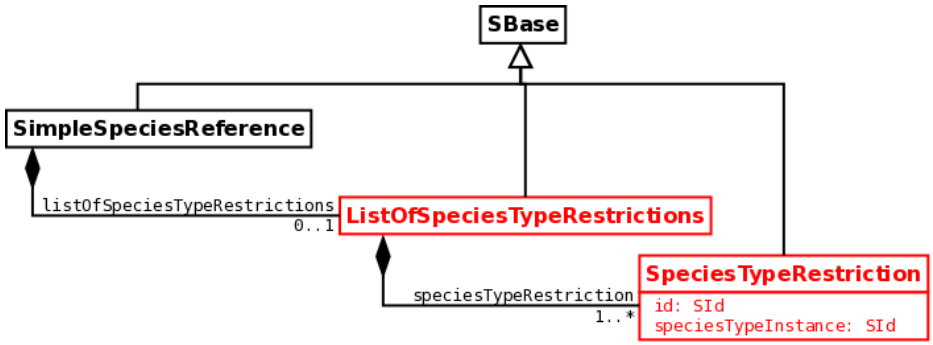

Figure 3.4: The SBML <SimpleSpeciesReference> construct in the *multi* extension

<SpeciesType> (Figure 3.4). Similarly, according to this proposal, the <Rule> construct contains an additional link to one <SpeciesTypeInstance>.

Other modifications proposed in *multi* include the <Bond>, <BindingSiteReference> and <ReactionRule>. Most of the remaining proposal supports multi-compartment models. In this work we mainly focus on multistate modeling. So we do not discuss those constructs here.

The multistate modeling constructs in *multi* enable the model to express different state features and binding sites for a species and its related reactions, but the states should be explicitly defined. Multi does not show a way to effectively represent a species list with enumeration. As we saw in Chapter 2, the ability to make state variables with a range is important. In Chapter 6 we discuss in detail how it is possible to modify the proposed standard so that it can include the ordering information of states and also can comply with the existing requirements.

For storing the properties and parameters of a multistate model in standard SBML format, some additions and modifications are needed in the proposed *multi* package. Since these features are not part of the current SBML standard, we can use SBML <annotations> to convey this information. According to the SBML definition, an annotation can be part of any <SBase> element (for example, <Species> in SBML Level 2). So, it is desirable to define multistate species as a special type of <Species> with state variables included in annotations. Similarly, other related constructs like <Reaction>, <SpeciesReferenece>, <Rule> etc. should be modified to include state information. Multi-dimensional arrays can be a suitable structure to represent the multistate species concept, if implemented with SBML standards.

3.2 Rule-Based Modeling and BioNetGen

In the rule-based modeling (RBM), the molecular interactions in a system are defined as rules. Rules provide conditions and outcomes for reactions and rate laws to guide the transformations. A rule either can express one simple chemical reaction or can define a large class of reactions, involving a common transformation rate law. Thus using rules it is possible to generate a list of reactions.

BioNetGen [2] was developed at Los Alamos National Lab by M.L. Blinov, J.R. Faeder and W.S. Hlavacek. It represents protein-protein interactions (including signal transduction, metabolic, and genetic regulatory networks) compactly and precisely using reaction rules, and processes those rules to obtain a chemical reaction network in a form suitable for simulation. BioNetGen follows the SBML standard and takes input in BioNetGen language (BNGL). Using BNGL it is possible to input site-specific details of protein-protein interactions of a model in BioNetGen.

In BNGL a model is constructed using some building blocks such as parameters, molecules (optional), species or seed species, reaction rules, observables etc [5]. BNGL allows patterns in the *rules* block to generate reaction rules using the set of species defined in the *species* block. For example, a species XP can be defined as $XP(bs1)$ where $bs1$ is the binding site for the species. Subsequently, it is possible to use $XP(bs1)$ for referring to the unbound forms, $XP(bs1!+)$ for referring to the bound forms, and $XP()$ for referring to all forms of XP including the complex forms. BNGL can generate definition of complex forms created using the pattern and reaction rules, even if that is not defined explicitely in the species block.

RuleBender [21] is a tool for creating, debugging, simulating and analyzing rule-based biological models. It is a free tool with an integrated development environment for creating models in BNGL. BioNetGen simulator commands can be applied to the models loaded or created in the RuleBender. We used the *RuleBender* interface to represent the small model of Chapter 2 in the BioNetGen language. Figure 3.5 shows BNGL code for the antagonism model.

In BNGL a multistate species XP should be represented as XP(*site1* $\sim$ *value1* $\sim$ *value2* $\sim$ ... *valueN, site2, ...*) where each site is defined with explicit list of state values (separated by a $\sim$). In Figure 3.5 we see the multistate species Cdh1 (which has one site for phosphorylation with 11 possible states) is defined as Cdh1($P \sim 0 \sim 1 \sim 2 \sim 3 \sim 4 \sim 5 \sim 6 \sim 7 \sim 8 \sim 9 \sim 10$). Although BNGL supports the use of patterns and mappings to avoid explicit description of all the states of the bindings that are involved in a reaction, it requires iterative definition of rules (for phosphorylation and dephosphorylation reactions) for each state of the multistate species (Cdh1) for generating our example multistate model. Because, BNGL can not automatically interpret the ordering (for example, next state or previous state) between the different states of a binding site from the species definition and can not recognize the relative order of states of the species in the reaction unless we define the reaction rules explicitly for each state of the multistate species. But such an ordering is required commonly in multistate reactions. Hence, modelers seek a platform to create multistate models with

```
begin model
begin parameters
.....
end parameters

begin molecule types
Trash()
Cdh1(P~0~1~2~3~4~5~6~7~8~9~10)
Clb2
Cdc14
end molecule types

begin reaction rules
#Clb2
Clb2 -> Trash()  Xa
Trash() -> Clb2  ks

#Phosphorylation
Cdh1(P~0) + Clb2 -> Clb2 + Cdh1(P~1)   kp
Cdh1(P~1) + Clb2 -> Clb2 + Cdh1(P~2)   kp
Cdh1(P~2) + Clb2 -> Clb2 + Cdh1(P~3)   kp
Cdh1(P~3) + Clb2 -> Clb2 + Cdh1(P~4)   kp
Cdh1(P~4) + Clb2 -> Clb2 + Cdh1(P~5)   kp
Cdh1(P~5) + Clb2 -> Clb2 + Cdh1(P~6)   kp
Cdh1(P~6) + Clb2 -> Clb2 + Cdh1(P~7)   kp
Cdh1(P~7) + Clb2 -> Clb2 + Cdh1(P~8)   kp
Cdh1(P~8) + Clb2 -> Clb2 + Cdh1(P~9)   kp
Cdh1(P~9) + Clb2 -> Clb2 + Cdh1(P~10)    kp

#Dephosphorylation
Cdh1(P~1) + Cdc14 -> Cdc14 + Cdh1(P~0) kh
Cdh1(P~2) + Cdc14 -> Cdc14 + Cdh1(P~1) kh
Cdh1(P~3) + Cdc14 -> Cdc14 + Cdh1(P~2) kh
Cdh1(P~4) + Cdc14 -> Cdc14 + Cdh1(P~3) kh
Cdh1(P~5) + Cdc14 -> Cdc14 + Cdh1(P~4) kh
Cdh1(P~6) + Cdc14 -> Cdc14 + Cdh1(P~5) kh
Cdh1(P~7) + Cdc14 -> Cdc14 + Cdh1(P~6) kh
Cdh1(P~8) + Cdc14 -> Cdc14 + Cdh1(P~7) kh
Cdh1(P~9) + Cdc14 -> Cdc14 + Cdh1(P~8) kh
Cdh1(P~10) + Cdc14 -> Cdc14 + Cdh1(P~9)   kh

#Degradation
Cdh1(P~0) -> Trash() kd
Cdh1(P~1) -> Trash() kd
Cdh1(P~2) -> Trash() kd
Cdh1(P~3) -> Trash() kd
Cdh1(P~4) -> Trash() kd
Cdh1(P~5) -> Trash() kd
Cdh1(P~6) -> Trash() kd
Cdh1(P~7) -> Trash() kd
Cdh1(P~8) -> Trash() kd
Cdh1(P~9) -> Trash() kd
Cdh1(P~10) -> Trash()   kd

#Synthesis
Trash() -> Cdh1(P~0) ks1
end reaction rules
end model
```

Figure 3.5: The *Antagonism Model* expressed in BNGL

less effort and more efficiency.

3.3 Other Software

There are other tools available for modeling and simulation of similar type of reaction networks. For example, Virtual Cell (a web-based computational environment for modeling and simulation of cell biology models) and NFSim (a simulation platform that simulates straightforward BNGL specification of models) are two such tools. Both the tools are based on BioNetGen and SBML standards, and lack the features of multistate modeling.

COPASI is a tool for modeling, simulation, and analysis of biochemical networks and their dynamics. COPASI, based on SBML standards and its own internal structure, also lacks features to support multi-state modeling. The JigCell and COPASI teams are working to develop an integrated modeling and simulation platform that includes the multistate modeling concept.

Chapter 4

Regulatory Network Modeling with Multistate Reactions

In this chapter, proposals for representating multistate species, reactions, and rules are presented.

4.1 Multistate Species

A multistate species has one or more multi-valued sites or features. Such a set of features can be represented with state or index variables. For example $X\{i, j\}$ represents a multistate species with 2 index variables (sites) i and j, each of which may have multiple values defined by specific lower and upper limits. In the model discussed in Chapter 2, Cdh1 is a multistate species that could be represented as CdhP{i} with one index variable, i.

The range of values for each variable of a multistate species may be different in different reactions but they must all fall within the pre-defined range of the species index. In expanded form, every possible combination of states could be represented by a separate species, if that particular state is expressed or used in any of the reactions. For example, $X\{i, j\}$ with $0 \leq i \leq 5$ and $0 \leq j \leq 2$ could be represented by a total of 6*3 single state species. Reaction parameters such as rate constants may depend on the value of the state variables.

We present a consistent syntax with which to represent multistate species. To keep similarity with standard single state species naming syntax, a pair of curly braces {} is used with comma separated index variables inside. For example *Y{0:5}* is a multistate species with 1 site, representing a set of six variants of the species. In contrast, *Y{0}* is a single state species. *Y{0:2,0:5}* represents a species with 2 sites, the first with three states and the second with six states. *Y{index1}* represents a species with one site referred to as $index1$ and the range for $index1$ (perhaps, $0 \leq index1 \leq 10$) needs to be defined separately.

Table 4.1: Multistate reaction types

Index	Reaction Type	Reaction Format	Rate Law Equation
1	Phosphorylation	$XP\{i\} \rightarrow XP\{i+1\}$ or, $XP\{i\} + E \rightarrow E + XP\{i+1\}$	$k_i * XP\{i\}$ or, $k'_i * E * XP\{i\}$
2	De-phosphorylation	$XP\{i+1\} \rightarrow XP\{i\}$ or $XP\{i+1\} + E \rightarrow E + XP\{i\}$	$k_i * XP\{i\}$ or $k'_i * E * XP\{i\}$
3	Degradation	$XP\{i\} \rightarrow .$ or $XP\{i\} + E \rightarrow E$	$k_i * XP\{i\}$ or $k'_i * E * XP\{i\}$
4	Complex Binding 1	a) $XP\{i\} + Y \rightarrow C\{i\}$ b) $XP\{i\} + YP\{j\} \rightarrow XY\{i,j\}$	a) $k_i * XP\{i\} * Y$ b) $k_i * XP\{i\} * YP\{j\}$
5	Complex Binding 2	a) $C\{i\} \rightarrow XP\{i\} + Y$ b) $XY\{i,j\} \rightarrow XP\{i\} + YP\{j\}$	$k_i * C\{i\}$
6	Degradation of complex form 1	a) $C\{i\} \rightarrow Y$; degrades $XP\{i\}$ b) $C\{i\} \rightarrow XP\{i\}$; degrades Y (with $C\{i\}$ a complex of Y and $XP\{i\}$)	$k_i * C\{i\}$
7	Degradation of complex form 2	a) $XY\{i,j\} \rightarrow YP\{j\}$; degrades $XP\{i\}$ b) $XY\{i,j\} \rightarrow XP\{i\}$; degrades $YP\{j\}$ (with $XY\{i,j\}$ a complex of $XP\{i\}$ and $YP\{j\}$)	$k_{ij} * XY\{i,j\}$

4.2 Multistate Reaction Format

Multistate reactions are similar to other reactions but have multistate species as reactants and/or products. We want to express appropriate sets of multistate reactions as a single reaction. The compact structure reduces the cognitive load on the modelers to a meaningful extent for large models. The other concern here is to keep the integrity of the model so that it can be easily converted for use in other tools that do not recognize our multistate representation.

Reactions with multistate properties are involved in a number of different behaviors. For example, some reactions result in modification of proteins or formation of heterogeneous protein complexes with enzymes and substrates. Others involve the targeted degradation of protein complexes. Understanding the nature of these reactions is important for modeling the dynamics of protein-protein interaction networks. These interactions are seen at different levels of protein sites with binding and catalytic activities. Some sample reactions have been listed in Table 4.1.

Based on the number of phosphorylation or dephosphorylation events that occur due to the binding of an enzyme to the target protein, phosphorylation or dephosphorylation reactions can be categorized as either processive or distributive. A phosphorylation (or

dephosphorylation) reaction is called processive if all or a specific number of the available sites of a substrate protein get phosphorylated (or dephosphorylated) when a kinase (or phosphatase) binds to it, with the effect being done in a single binding event (i.e., before the kinase or phosphatase gets dissociated from the binding site). But in a distributive reaction at most one phosphorylation or dephosphorylation event occurs on a single binding of the enzyme (i.e., kinase or phosphatase) to the substrate.

Phosphorylation and dephosphorylation reactions can be dependent on (or independent of) how many sites are already phosphorylated. The mechanism may depend on the sequence of sites involved in phosphorylation or dephosphorylation reactions. For the ordered case, the reaction events involve the sites in a specific sequence (i.e., the sites are phosphorylated in a specific sequence and dephosphorylated in the reverse sequence) and the rate of the reactions does not depend on the number of free or occupied sites. For the disordered case, the reactions can occur at arbitrary sites and the rate of phosphorylation or dephosphorylation is proportional to the number of free or occupied sites.

When the order of the species states is important for a series of reactions, it is required to express relations such as predecessor or successor among reactants and products. For example, $XP\{i\} \rightarrow XP\{i-1\}$ or $XP\{i\} \rightarrow XP\{i+1\}$. Here direct reference to previous or next state has been used where i gives the starting state value. This fashion of expression is especially useful where individual state identity is less significant. For some reactions it might be required to refer to a specific set of states (such as $XP\{[0,1,5]\}$) or a set of states defined with a function (such as, the odd states for $XP\{0:11\}$ can be defined as $XP\{2*i+1\}$ with a defined range for the value of i). Arithmetic expressions can be used to represent such relations with assigned numerical values for states or a defined range of numerical values for the states as a whole.

The syntax for multistate reactions should include these reaction parameters:

- A range for each site of each multistate species involved in the reaction
- Reaction Mechanism
  - Progressive or distributive
  - Ordered or disordered (except degradation and complexation reactions)
- Rate law and related parameters
  - State dependent or independent parameters
  - (If dependent) specific function (i.e., stepwise, exponential, or custom)

4.3 Rate Laws

Rate laws play an important role in the mechanism of a reaction. Rate law equation and parameters can impact significantly the nature of the multistate reaction. Some rate law equations for multistate reactions are shown in Table 4.2. A model editor should allow users to add customized rate law equations, or to choose from common ones. We have mainly focused on distributive models, specifically models with distributive and multistate phosphorylation/dephosphorylation reactions. We have studied and implemented details for

two known rate laws (*Mass Action* and *Michaelis-Menten*) and provided option for defining local rate law for the reaction (*Local*).

We have noticed that the reaction mechanism (ordered or disordered) has an impact on phosphorylation and dephosphorylation reactions, but not on synthesis, degradation or complexation reactions. It is expected that multistate modeling software will be intelligent enough to distinguish among reactions, for example, to distinguish multistate phosphorylation and dephosphorylation (or ubiquitylation and deubiquitylation) reactions from multistate reactions like degradation or complex formation. Based on the reaction type, different reaction mechanisms can be applied.

Another important parameter in multistate modeling is the reaction rate constant. If the rate constant (say k) is state independent then it is assigned one value for all the reactions involving the species states. When the rate constant depends on the state of the system, it is assigned values following some function. For example:

- **Stepwise**: The value of the rate constant is different for each state. Then we need to define a set of rate constants, one for each state. We write it as $k\{i\}$.
- **Exponential**: For the exponential case, the rate constant of subsequent states depends exponentially on the first step, for example: $k\{i\} = k_0 r^i$.
- **Local Function**: The custom function allows a user to create any function required by the model. For example, $k\{i\} = kp(1 + k_0 r^{i+1})$ can be a custom function for the rate constant.

Table 4.2 shows some rate law equations used for multistate reactions using k_p, k_h, k_d, and k_c as rate constants.

4.4 Other Constructs

There may be parameters or rules that involve specific states of a multistate species, defined with a range of one or more index variables. For example, the activity of the active or inactive forms of XP (a multistate species which has one unphosphosphorylated and some phosphorylated states) can be expressed like Equation 4.1 using the summation of concentration of involved states with some weights. In this equation, a_i is the weight for active species, and L, M and N are numerical variables to define range of index, i.

$$A_{active} = \sum_{i=L}^{M} a_i * XP_i; 0 \leq L \leq M \leq N \tag{4.1}$$

A mathematical implementation of the summation operator is required to represent such expressions. Other constructs required for multistate modeling are *Array* and *Set*. An array is a structure to represent a group of objects in a defined order. It can be used to reperesent the range of indices of a multistate species. A set is a collection of distinct objects. We can use this construct to define the list of some species states defined by a specific function as described in Section 4.3. As we are using SBML ultimately to store models and SBML follows

Table 4.2: Rate law equations for multistate phosphorylation, dephosphorylation, degradation and complexation reactions, with state-independent rate constants

Index	Reaction Example	Reaction Type	Reaction Mechanism	Rate Law	Equation
1	$XP\{i\} + E \rightarrow XP\{i+1\} + E$ $0 \leq i \leq N-1$	Phospho-rylation	Ordered	Mass Action	$k_p * E * XP\{i\}$
2				Michaelis -Menten	$(k_p * E * XP\{i\}/K_{Mi})/$ $(1 + \sum_{i=0}^{N-1} XP\{i\}/K_{Mi})$
3			Disordered	Mass Action	$(N-i) * k_p * E * XP\{i\}$
4				Michaelis -Menten	$((N-i) * k_p * E * XP\{i\}/K_{Mi})/$ $(1 + \sum_{i=0}^{N-1} XP\{i\}/K_{Mi})$
5	$XP\{i\} + E \rightarrow XP\{i-1\} + E$ $1 \leq i \leq N$	Dephospho-rylation	Ordered	Mass Action	$k_h * E * XP\{i\}$
6				Michaelis -Menten	$(k_h * E * XP\{i\}/K_{Mi})/$ $(1 + \sum_{i=1}^{N} XP\{i\}/K_{Mi})$
7			Disordered	Mass Action	$i * k_h * E * XP\{i\}$
8				Michaelis -Menten	$(i * k_h * E * XP\{i\}/K_{Mi})/$ $(1 + \sum_{i=1}^{N} XP\{i\}/K_{Mi})$
9	$XP\{i\} + E$ $\rightarrow E$ $0 \leq i \leq N$	Degradation	N/A	Mass Action	$k_d * E * XP\{i\}$
10				Michaelis -Menten	$(k_d * E * XP\{i\}/K_{Mi})/$ $(1 + \sum_{i=0}^{N} XP\{i\}/K_{Mi})$
11	$XP\{i\} + Y$ $\rightarrow C\{i\}$ $0 \leq i \leq N$	Complexation Type 1	N/A	Mass Action	$k_c * XP\{i\} * Y$
12	$XP\{i\} + YP\{j\}$ $\rightarrow C\{i,j\}$ $0 \leq i \leq N$ $0 \leq j \leq M$	Complexation Type 2	N/A	Mass Action	$k_c * XP\{i\} * YP\{j\}$

MathML constructs for mathematical operators, we have implemented these operators and constructs using MathML in JCMB.

MathML supports two forms of summation:

1. $\sum_{i=i_{min}}^{i_{max}} f(X_i)$. Here summation operands are lowlimit i_{min}, uplimit i_{max} and a function of X_i.

2. $\sum_{i \in B} f(X_i)$. Here summation operands are a set B and a function of X_i; the summation operation is carried out on $f(X_i)$ over the selected values of i as defined in B.

So, it is possible to use the summation operator either over a range of indices or a set of indices as required by the model.

Chapter 5

A Complete Model

Any realistic model for the molecular mechanism of the budding yeast cell-cycle control system naturally includes multistate species and reactions. In this chapter, a version of the model of Barik et al [1]. is discussed in terms of its multistate features.

5.1 Biological Model

Figure 5.1 shows a schematic diagram of Barik's complete model, a part of which appeared as the small model in Chapter 2. In Figure 5.1, solid arrows represent chemical reactions, dotted arrows represent enzymatic activity, dashed arrows represent multistate phosphorylation chains, and T-shaped arrows with balls on the cross bar indicate reversible binding reactions. This diagram shows the major components of the model. To simplify the view, the synthesis and degradation reactions of Whi5, SBF, Cdh1, Net1, Hbf, Hi5 and Ht1 are not shown here. We assume that the synthesis and degradation reactions follow simple birth-death processes. The diagram includes synthesis and degradation of MbS, which is the mRNA for ClbS, the only regulated mRNA species in the model. Hi5, Hbf and Ht1 are three unregulated phosphatases, which oppose cyclin-dependent kinases on the Whi5, SBF and Net1 phosphorylation chains, respectively. The number of molecules of ClbM plays an important role in the model by triggering the event of cell division. At a certain threshold of ClbM the cell divides, so cell volume becomes half of the previous volume and a new cell cycle begins.

5.2 The Multistate Model

In the schematic diagram (Figure 5.1), there are six multistate species (see Table 5.1) and 17 non-multistate or single species. All of the multistate species are one dimensional (i.e., consisting of one phosphorylation sequence), though the number of possible phosphorylation states differs from species to species.

In Figure 5.1 the synthesis and degradation of the four proteins (Net, Cdh1, SBF and

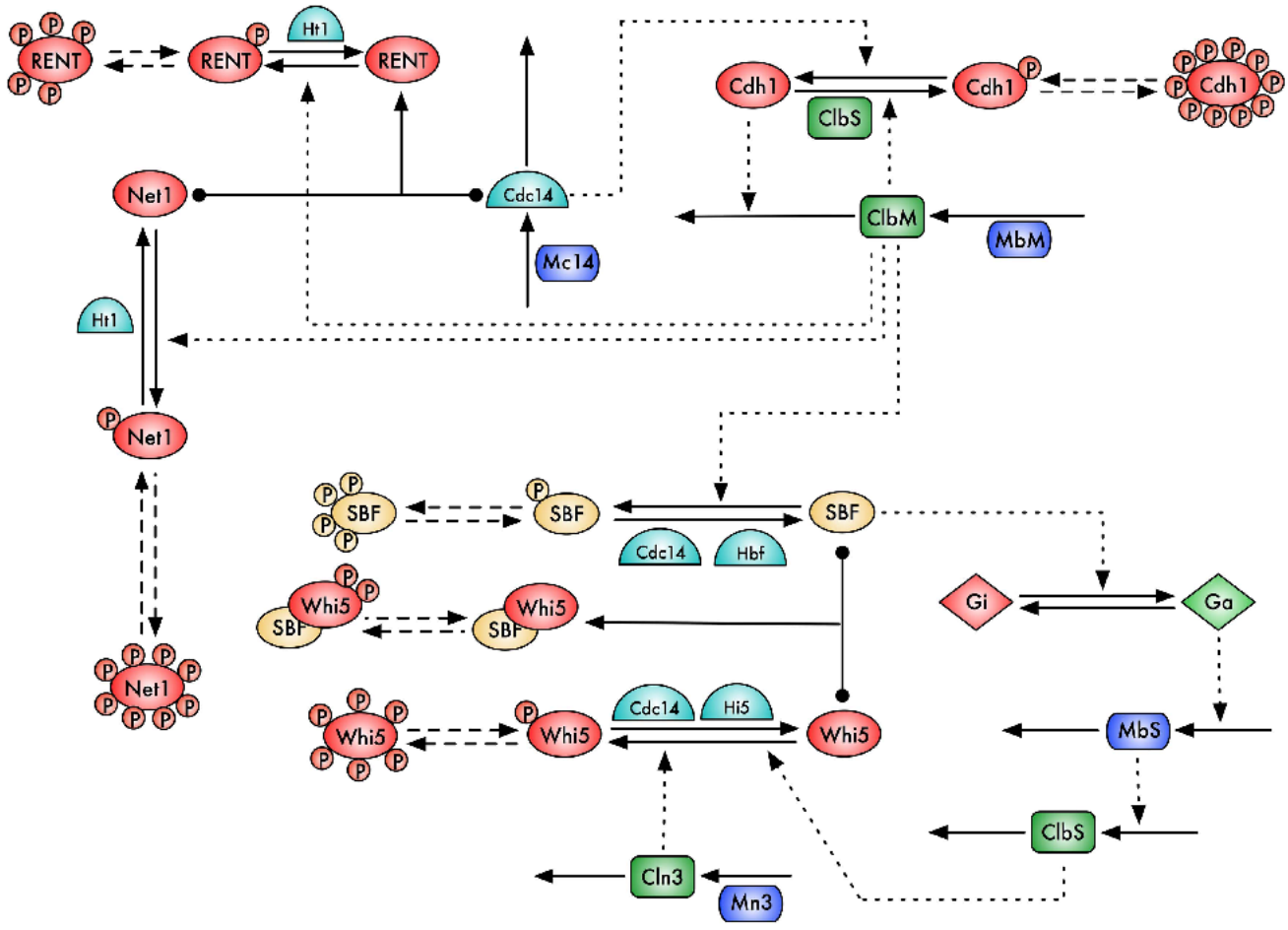

Figure 5.1: The *Cell Cycle Control Model* of the budding yeast [1]

Table 5.1: Multistate species list in the *Cell Cycle Control Model* of the budding yeast

Index	Species Name	Phosphorylation States	Representation as Multistate Species
1	sbf	5	sbf[0:4]
2	whi5	7	whi5[0:6]
3	cmp	3	cmp[0:2]
4	cdh1	11	cdh1[0:10]
5	net1	9	net1[0:8]
6	rnt	6	rnt[0:5]

Whi5) has not been shown. Also the details of the creation and destruction of the complexes they form (i.e., Cmp and RNT) are missing. There are a total of 220 single-state reactions in this model, including all the synthesis and degradation reactions missing in the diagram as mentioned above. Using multistate reactions, we can reduce that to 73 (see Figures 5.2 and 5.3).

Name	Reaction	Type	Equation
1	-> clbm	Local	ksbm * (cell / PI) * mbm
2	cdh1{0} + clbm -> cdh1{0}	Local	kdbma * cdh1{0} * clbm / cell
3	clbm ->	Local	kdbmi * cdh1pT * clbm / cell
4	clbm ->	Mass Action	gdbm * clbm
5	-> cdh1{0}	Local	ksh1 * (cell / PI) * mh1
6	cdh1{i} -> cdh1{i-1} ; 1 <= i <= 10	multistate - Ordered - Mass Action	kdh1p * cdh1{i}
7	cdh1{i} + cdc14 -> cdh1{i-1} + cdc14 ; 1 <= i <= 10	multistate - Local	kdh1 * cdh1{i} * cdc14 / cell
8	cdh1{i} -> ; 0 <= i <= 10	multistate - Mass Action	gdh1 * cdh1{i}
9	cdh1{i} + clbs -> cdh1{i+1} + clbs ; 0 <= i <= 9	multistate - Local	kph1 * cdh1{i} * clbs / cell
10	cdh1{i} + clbm -> cdh1{i+1} + clbm ; 0 <= i <= 9	multistate - Local	kph1p * cdh1{i} * clbm / cell
11	-> mbm	Local	ksmbm * PI
12	mbm ->	Mass Action	gdmbm * mbm
13	-> mh1	Local	ksmh1 * PI
14	mh1 ->	Mass Action	gdmh1 * mh1
15	mn3 -> cln3 + mn3	Local	ksn3 * (cell / PI) * (cell / PI) * mn3
16	cln3 ->	Mass Action	gdn3 * cln3
17	-> clbs	Local	ksbs * (cell / PI) * mbs
18	clbs ->	Mass Action	gdbs * clbs
19	-> cdc14	Local	ksc14 * (cell / PI) * mc14
20	cdc14 ->	Mass Action	gd14 * cdc14
21	net1{i} + cdc14 -> rnt{i} ; 0 <= i <= 5	multistate - Local	kar * cdc14 * net1{i} / cell
22	rnt{i} -> cdc14 ; 0 <= i <= 5	multistate - Mass Action	gdt1 * rnt{i}
23	rnt{i} -> net1{i} + cdc14 ; 0 <= i <= 5	multistate - Mass Action	kdr * rnt{i}
24	-> sbf{0}	Mass Action	ksbf * cell
25	sbf{0} + whi5{i} -> cmp{i} ; 0 <= i <= 2	multistate - Local	kac * sbf{0} * whi5{i} / cell
26	cmp{i} -> sbf{0} + whi5{i} ; 0 <= i <= 2	multistate - Mass Action	kdc * cmp{i}
27	cmp{i} -> sbf{0} ; 0 <= i <= 2	multistate - Mass Action	gdi5 * cmp{i}
28	sbf{i} + hbf -> sbf{i-1} + hbf ; 1 <= i <= 4	multistate - Local	kdbf * sbf{i} * hbf / cell
29	sbf{i} + cdc14 -> sbf{i-1} + cdc14 ; 1 <= i <= 4	multistate - Local	kdbfp * sbf{i} * cdc14 / cell
30	sbf{i} -> ; 0 <= i <= 4	multistate - Mass Action	gdbf * sbf{i}
31	sbf{i} + clbm -> sbf{i+1} + clbm ; 0 <= i <= 3	multistate - Local	kpbf * sbf{i} * clbm / cell
32	-> hbf	Local	kshbf * (cell / PI) * mhbf
33	hbf ->	Mass Action	gdhbf * hbf
34	whi5{i} + cdc14 -> whi5{i-1} + cdc14 ; 1 <= i <= 6	multistate - Local	kdi5p * whi5{i} * cdc14 / cell
35	whi5{i} + hi5 -> whi5{i-1} + hi5 ; 1 <= i <= 6	multistate - Local	kdi5 * whi5{i} * hi5 / cell
36	whi5{i} + cln3 -> whi5{i+1} + cln3 ; 0 <= i <= 5	multistate - Local	kpi5 * whi5{i} * cln3 / cell
37	whi5{i} + clbs -> whi5{i+1} + clbs ; 0 <= i <= 5	multistate - Local	kpi5p * whi5{i} * clbs / cell

Figure 5.2: Multistate reactions in the *Cell Cycle Control Model* of the budding yeast (part1)

Most of the kinetics for the reactions follows the simple mass action rate law, and the multistate reactions use the ordered mechanism. A few of the reactions are not mass action, and the rate of reaction is affected by species that are not necessarily a reactant. There are more than 100 parameters and variables used in this model. In the reactions we used the

term *cell* which represents the volume of the cell. *PI* refers to the ploidy of the cell, which in this model of the yeast cell is one.

Name	Reaction	Type	Equation
38	-> whi5{0}	Local	ksi5 * (cell / PI) * mi5
39	whi5{i} -> ; 0 <= i <= 6	multistate - Mass Action	gdi5 * whi5{i}
40	cmp{i} -> whi5{i} ; 0 <= i <= 2	multistate - Mass Action	gdbf * cmp{i}
41	-> hi5	Local	kshi5 * (cell / PI) * mhi5
42	hi5 ->	Mass Action	gdhi5 * hi5
43	-> net1{0}	Local	kst1 * (cell / PI) * mt1
44	net1{i} + ht1 -> net1{i-1} + ht1 ; 1 <= i <= 8	multistate - Local	kdt1 * net1{i} * ht1 / cell
45	net1{i} + clbm -> net1{i+1} + clbm ; 0 <= i <= 7	multistate - Local	kpt1 * net1{i} * clbm / cell
46	net1{i} -> ; 0 <= i <= 8	multistate - Mass Action	gdt1 * net1{i}
47	rnt{i} -> net1{i} ; 0 <= i <= 5	multistate - Mass Action	gd14 * rnt{i}
48	rnt{i} + clbm -> rnt{i+1} + clbm ; 0 <= i <= 4	multistate - Local	kpnt * rnt{i} * clbm / cell
49	rnt{i} + ht1 -> rnt{i-1} + ht1 ; 1 <= i <= 5	multistate - Local	kdnt * rnt{i} * ht1 / cell
50	-> ht1	Local	ksht1 * (cell / PI) * mht1
51	ht1 ->	Mass Action	gdht1 * ht1
52	cmp{i} + cdc14 -> cmp{i-1} + cdc14 ; 1 <= i <= 2	multistate - Local	kdcmp * cmp{i} * cdc14 / cell
53	cmp{i} + hi5 -> cmp{i-1} + hi5 ; 1 <= i <= 2	multistate - Local	kdcm * cmp{i} * hi5 / cell
54	cmp{i} + clbs -> cmp{i+1} + clbs ; 0 <= i <= 1	multistate - Local	kpcmp * cmp{i} * clbs / cell
55	cmp{i} + cln3 -> cmp{i+1} + cln3 ; 0 <= i <= 1	multistate - Local	kpcm * cmp{i} * cln3 / cell
56	-> mn3	Local	ksmn3 * PI
57	mn3 ->	Mass Action	gdmn3 * mn3
58	sbf{0} -> ga + sbf{0}	Local	kag * (sbf{0} / cell) * (PI - ga)
59	ga ->	Mass Action	kdg * ga
60	ga -> mbs + ga	Mass Action	ksmbs * ga
61	mbs ->	Mass Action	gdmbs * mbs
62	-> mc14	Local	ksmc14 * PI
63	mc14 ->	Mass Action	gdmc14 * mc14
64	-> mhbf	Local	ksmhbf * PI
65	mhbf ->	Mass Action	gdmhbf * mhbf
66	-> mi5	Local	ksmi5 * PI
67	mi5 ->	Mass Action	gdmi5 * mi5
68	-> mhi5	Local	ksmhi5 * PI
69	mhi5 ->	Mass Action	gdmhi5 * mhi5
70	-> mt1	Local	ksmt1 * PI
71	mt1 ->	Mass Action	gdmt1 * mt1
72	-> mht1	Local	ksmht1 * PI
73	mht1 ->	Mass Action	gdmht1 * mht1

Figure 5.3: Multistate reactions in the *Cell Cycle Control Model* of the budding yeast (part2)

In Reaction 3 of Figure 5.2 we used the variable $cdh1pT$. In this model $Cdh1\{0\}$ (the unphosphorylated form of Cdh1) is the high activity form (with rate constant kdbma = 0.025 $fL\ molec^{-1}\ min^{-1}$). $cdh1pT$ represents the summation of the low activity (phosphorylated) forms of Cdh1 (with rate constant kdbmi = 1.7 * 10^{-5} $fL\ molec^{-1}\ min^{-1}$). We used the summation operator to define $cdh1pT$ as $sum(cdh1\{i\}; 1; 10)$.

Chapter 6

Multistate Modeling with the JigCell ModelBuilder and SBML

In Chapter 3, we saw the BioNetGen (and related tools such as NFSim) does not define relations of species (as reactants and products) involved in a reaction as successor or predecessor (or with other arithmetic expressions) in a straight-forward manner. Although BioNetGen can define states of species using patterns and mapping, there is a limitation of expressing relations in a group of reactions. For example, consider the phosphorylation reaction of a multistate species XP defined as $XP\{i\}+E \rightarrow XP\{i+1\}+E$, where the value of i indicates the number of phosphate groups attached to XP. In this reaction $XP\{i+1\}$ is the successor state of $XP\{i\}$ as it contains one more phosphate group than $XP\{i\}$ does. In BioNetGen, expressions for successor or predecessor such as $i \rightarrow i+1$ or $i \rightarrow i-1$ in species with indices are allowed only after providing explicit definitions for the state order and reaction rules (as we saw in Figure 3.5). But the order of species states plays an important role in the multistate reactions and demands a meaningful way of expression.

Multistate phosphorylation reactions appear frequently in living cells. In some cases, number of states for a substrate involved in the reaction (i.e. the number of phosphate groups attached to the species in a specific site or number of phosphorylated sites) appears to be more important than the individual states (i.e., the resulting phosphorylated residues). In a cascade of phosphorylation reactions with multiple states, the option to identify a state by index or position is required often. For example, multistate phosphorylation of the yeast cyclin dependent kinase inhibitor Sic1 has a threshold of six sites (six or more must be phosphorylated out of the nine possible phosphorylation sites of Sic1) to promote binding to the ubiquitin ligase (SCF^{Cdc4}) for Sic1 degradation regardless of which six are bound [17]. Providing the explicit definition of the order of states (which is the only option with tools such as BioNetGen) is cumbersome for users, we need a way to create models with less effort and a conversion mechanism for using the models with other simulation tools such as BioNetGen.

6.1 JCMB Spreadsheet

We have extended JCMB to support multistate modeling. In this section, we use the antagonism model of Chapter 2 to describe the modified JigCell GUI. Details for how JigCell is used to model single state reactions can be found in [25].

6.1.1 Species

For modeling a system with JCMB, defining the species is an important step. In JCMB,

- It is possible to define the species directly in the *Species* spreadsheet before entering the reactions as described in Figure 6.1.
- Alternatively, while the reactions are entered, JCMB can identify and automatically create the species involved by asking the required information (Figure 6.2).

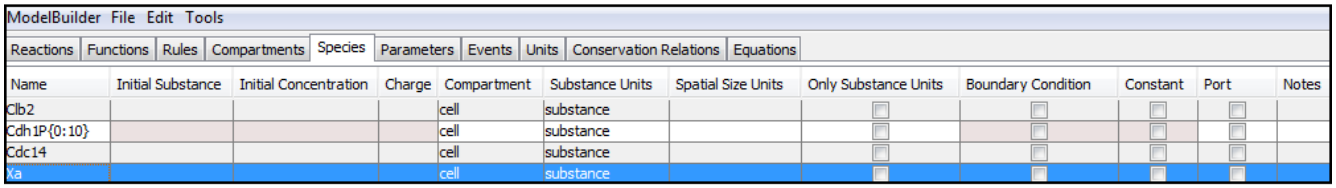

Figure 6.1: Species spreadsheet in JCMB

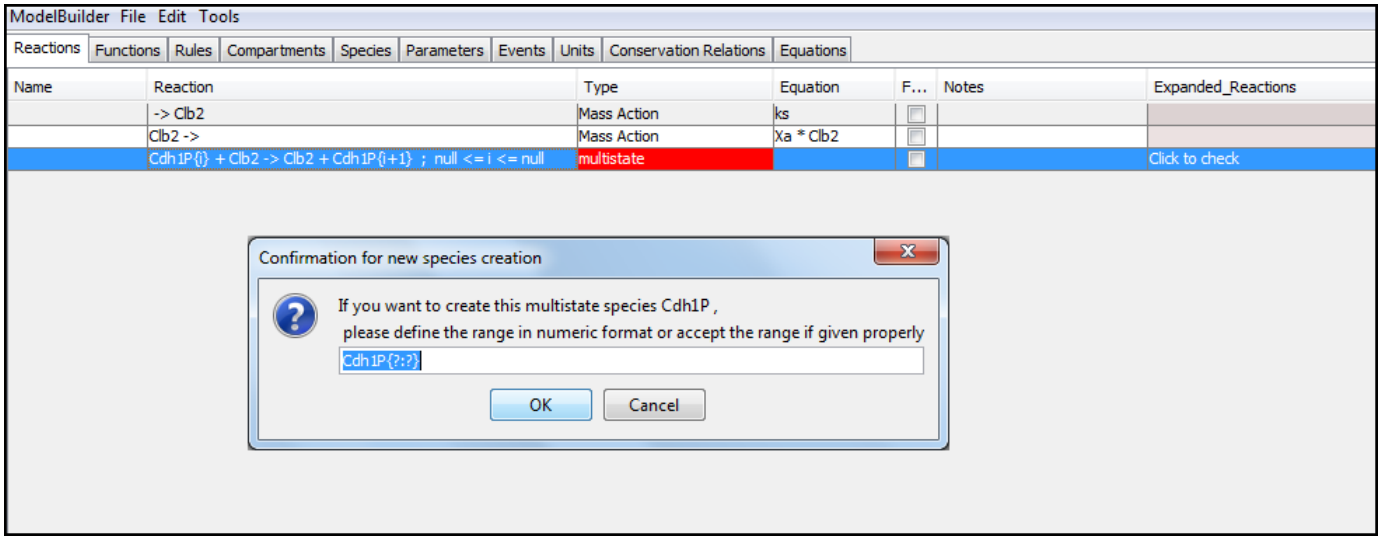

Figure 6.2: A multistate species creation dialog box in JCMB

Before accepting a reaction in the reaction spreadsheet, JCMB will ensure that all involved species are defined appropriately. In other words, it will check that there is no conflict

between single state and multistate species naming, and the multistate species are defined properly with state ranges defined. For example, in Figure 6.2, Cdh1P has been identified as a one dimensional (with one site) multistate species and that acceptable values for *'?:?'* inside the curly braces are numbers. Cdh1P{0:10} is the correct definition for this model.

6.1.2 Rules

The *Rules* spreadsheet accepts assignment rules to create variables for use in reactions. Besides other existing arithmetic operators, enhanced JCMB offers the summation operator as *sum(expression, lowerlimit, upperlimit)*. For multistate reactions, this operator is used to express activity of a multistate species. The range defined with lowerlimit and upperlimit automatically selects the related states of the species identified by index value. Figure 6.3 shows how we can define the *Xa* variable for activity of Cdh1P.

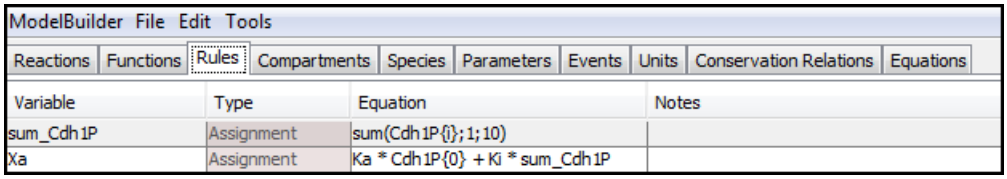

Figure 6.3: Rule spreadsheet in JCMB showing use of the *sum()* operator

6.1.3 Reactions

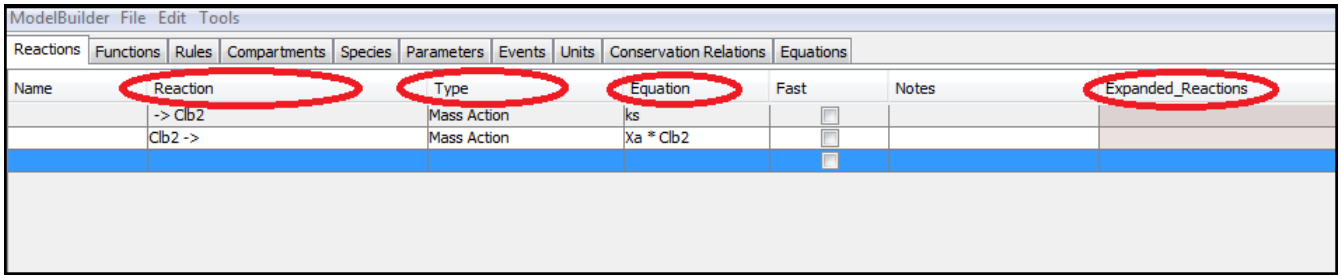

Figure 6.4: Reaction spreadsheet in JCMB: *Reaction*, *Type*, *Equation*, and *Expanded Reactions* columns for defining multistate reactions

The *Reaction* spreadsheet is the main input area for reactions and related parameters in JCMB. There are four columns in this spreadsheet that are most important for multistate reactions as shown in Figure 6.4. A click in the *Reaction* column brings out the reaction entry dialog box (Figure 6.5) which takes reactions as $Reactant(s) \rightarrow Product(s)$. A multistate

reactant/product must be entered with some index variables (for example, $XP\{i\}$), whereas one specific state of the multistate species is treated as a normal species and can be entered directly with the state value (for example, $XP\{0\} \rightarrow$ is a normal single state reaction). The reaction entered in the dialog box will be accepted and shown in the *Reaction* field of that row (as in Figure 6.6) only after a successful pass through syntax checking and the species creation steps. The indices of multistate species will initially be shown as undefined or null.

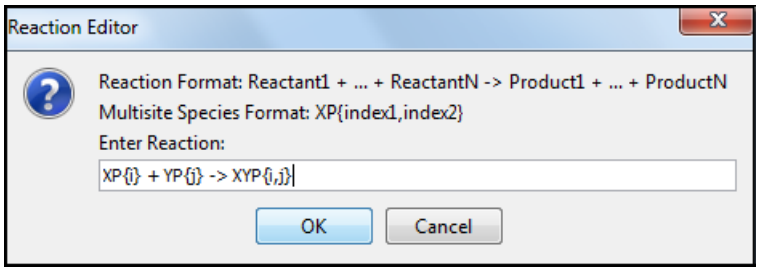

Figure 6.5: Reaction entry dialog box

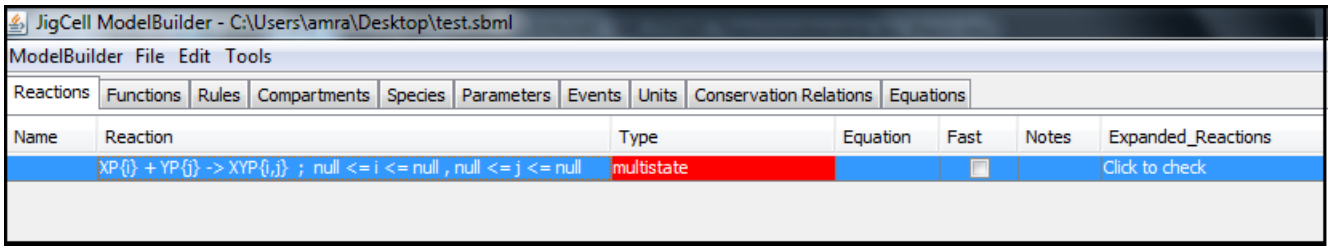

Figure 6.6: An accepted multistate reaction in the *Reaction* column with undefined range for indices

The second input column is named *Type*, highlighted in red in Figure 6.6 to indicate it is the next field that needs to be filled out. For normal reactions, this field will offer a simple dropdown box to select the rate law. The *Multistate Reaction Parameter Editor* dialog box will appear after clicking on this field if the reaction involves one or more multistate species. This dialog box, as shown in Figure 6.7, takes information for some parameters, as follows,

- Based on the reaction type, the *RateLaw* field shows some relevant rate laws in its dropdown selection box. The *Local* rate law gives options to customize the rate equation later in the *Equation* column. Any predefined rate law (defined in the *Functions* spreadsheet, discussed in the next section) will also appear here.
- The next area is for index variables used in this reaction. An acceptable range (with lower and upper limit) should be given for each index.
- Based on the type of the reaction, the reactions can be defined as ordered or disordered in the *Reaction Mechanism* area. In this reaction the option is disabled as, according

to definition, complex binding reactions do not have this sort of mechanism.

- The fourth choice is for state dependency of the rate constant, k. The rate constant can be independent of the state value or dependent on it. For the later case, k can be evaluated by a known function (e.g., *Exponential*, *Stepwise*) or some custom equation which can be given in a textbox by selecting *Local*. Details of the possible values of this field has been discussed under Chapter 4.3 .

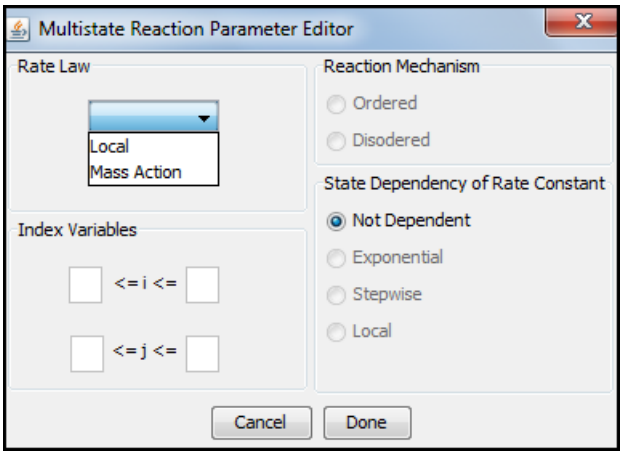

Figure 6.7: Multistate reaction parameter editor dialog box

The *Equation* column takes the rate equation based on the rate law and parameters defined in the *Type* field as shown in Figure 6.8. The *Expanded_Reactions* field becomes enabled only for the multistate reactions to show the expanded view of the compact multistate reaction. It is helpful for verifying that the compact form represents the set of reactions correctly. Figure 6.9 shows a view of this feature with all the reactions of the *Antagonism* model in multistate format.

Color has been used to assist modelers in filling out the required information in this spreadsheet. After a reaction is given in the *Reactions* column, the column that should be filled out next (in this case, *Type*) becomes red. As for multistate reactions, the *Type* field takes multiple parameters, and yellow highlighting is used to indicate incomplete information. In Figure 6.10, rate law information is not provided but the range for the index i is already given, so the field is highlighted yellow. If information is given completely and correctly then the *Equation* column becomes red, because it is the next column to be filled out.

6.1.4 Functions

This spreadsheet helps to define custom, model-specific rate laws and also lists the rate laws and functions used in the model so far. For multistate reactions we may need to define some custom rate laws as listed in Table 4.2. Figure 6.11 shows the rate laws used in the

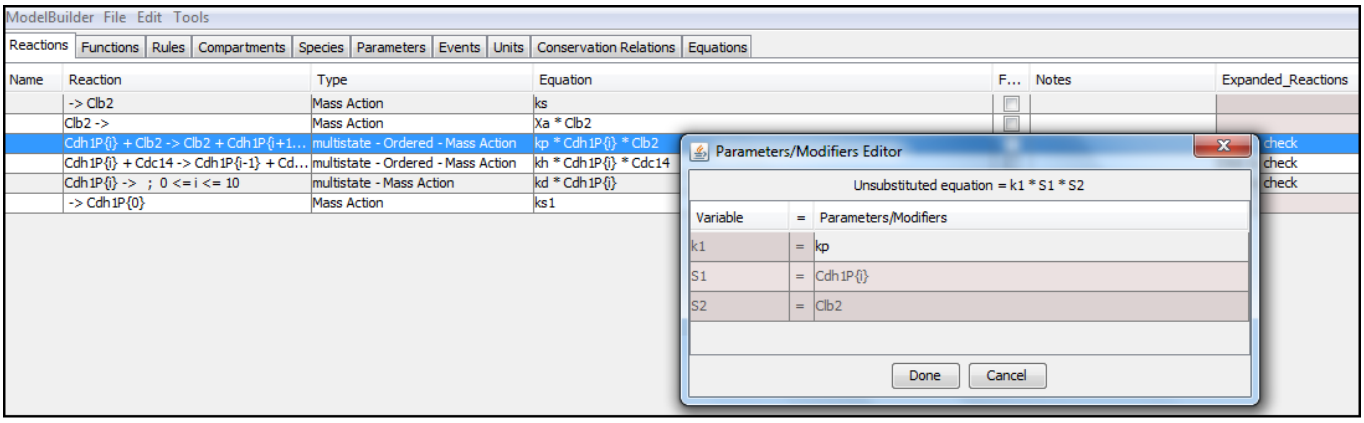

Figure 6.8: *Equation* column in the Reaction spreadsheet

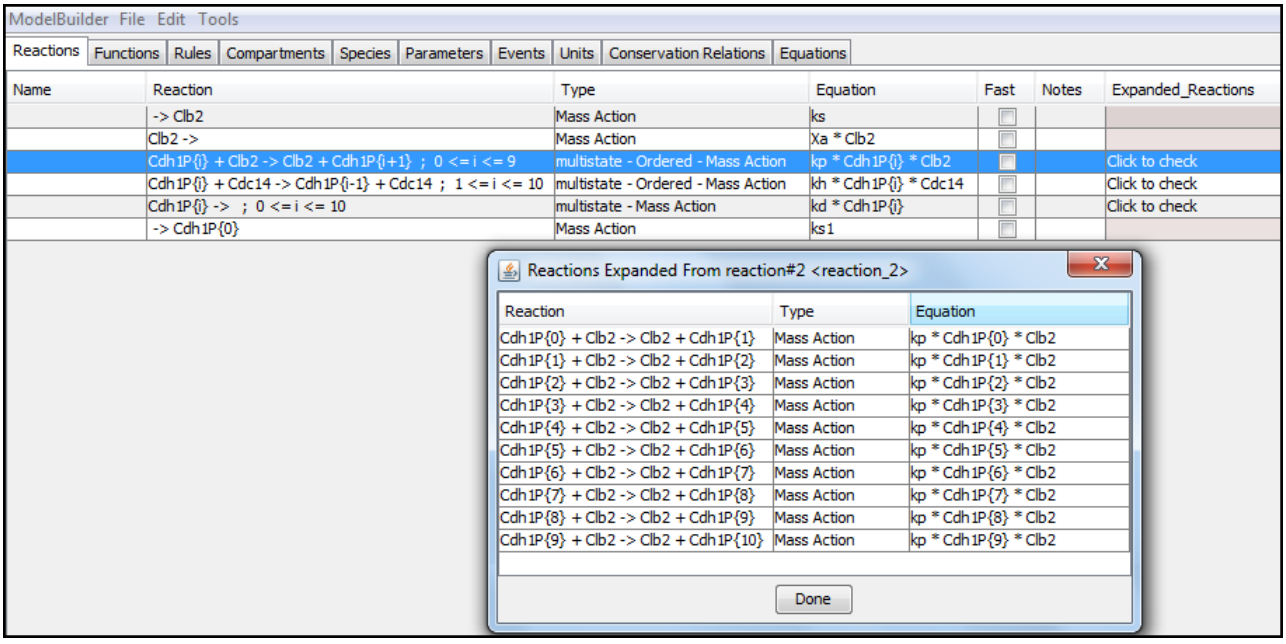

Figure 6.9: *Expanded Reactions* dialog box showing multistate reactions in the *Antagonism Model*

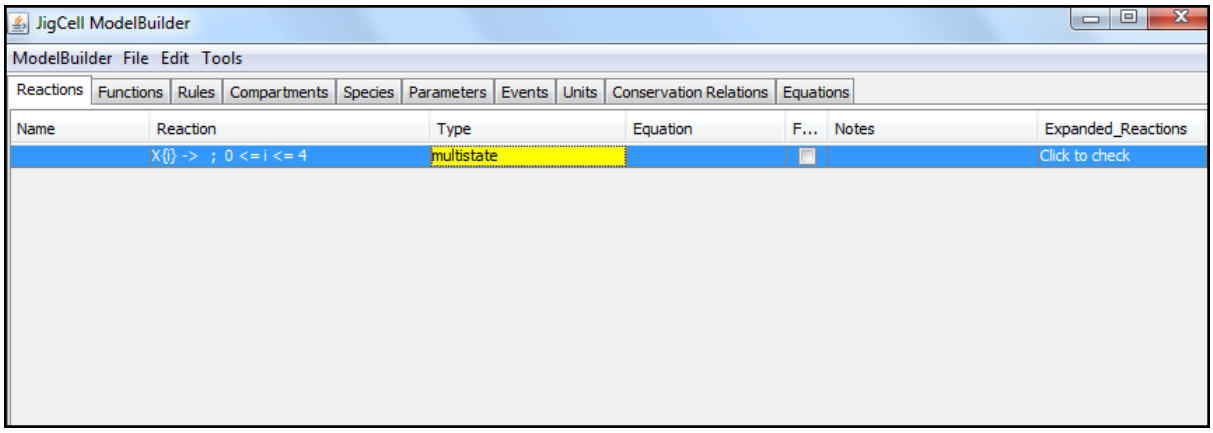

Figure 6.10: Highlighting columns with red or yellow for guiding in building the model

Antagonism model, for example, *Mass_Action_2_Ordered* is the name of the rate law for ordered phosphorylation reaction with two substrates.

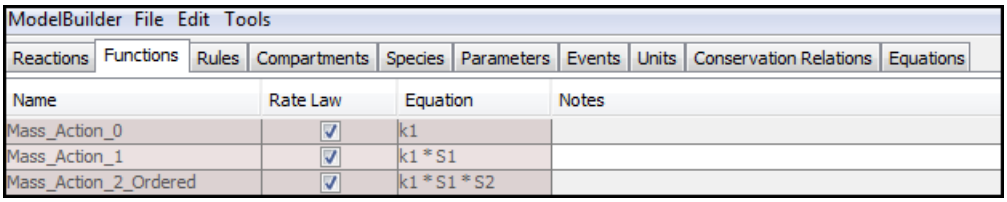

Figure 6.11: Functions spreadsheet

6.1.5 Parameters

This spreadsheet is used for its original purpose (i.e., for definition and assignment of parameters) as for single state models. Figure 6.12 shows the list of parameters of the *Antagonism* model.

6.1.6 Equations

In multistate modeling, the *Equations* spreadsheet (Figure 6.13) can provide a means to check that everything is going as expected. With flattening of the model it is possible to get the list of ODEs with single state species to use in a simulation or in other tools.

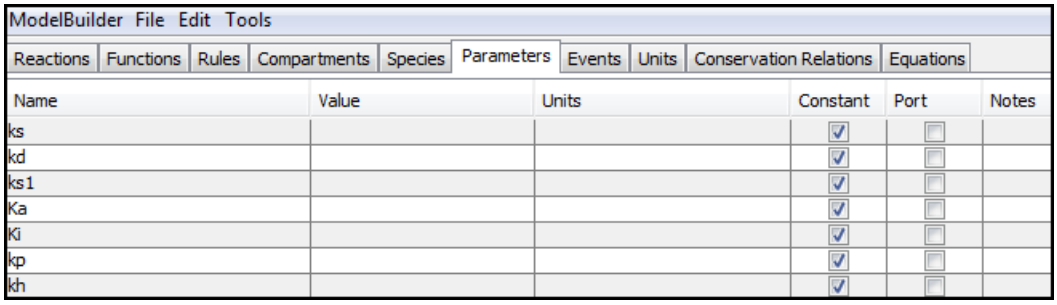

Figure 6.12: Parameters spreadsheet showing parameters used in the *Antagonism Model*

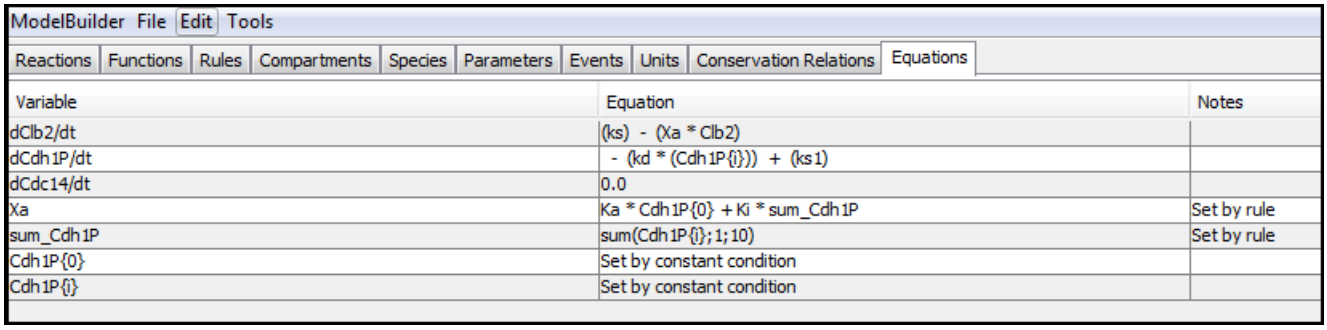

Figure 6.13: Equations (ODEs) shown in multistate format

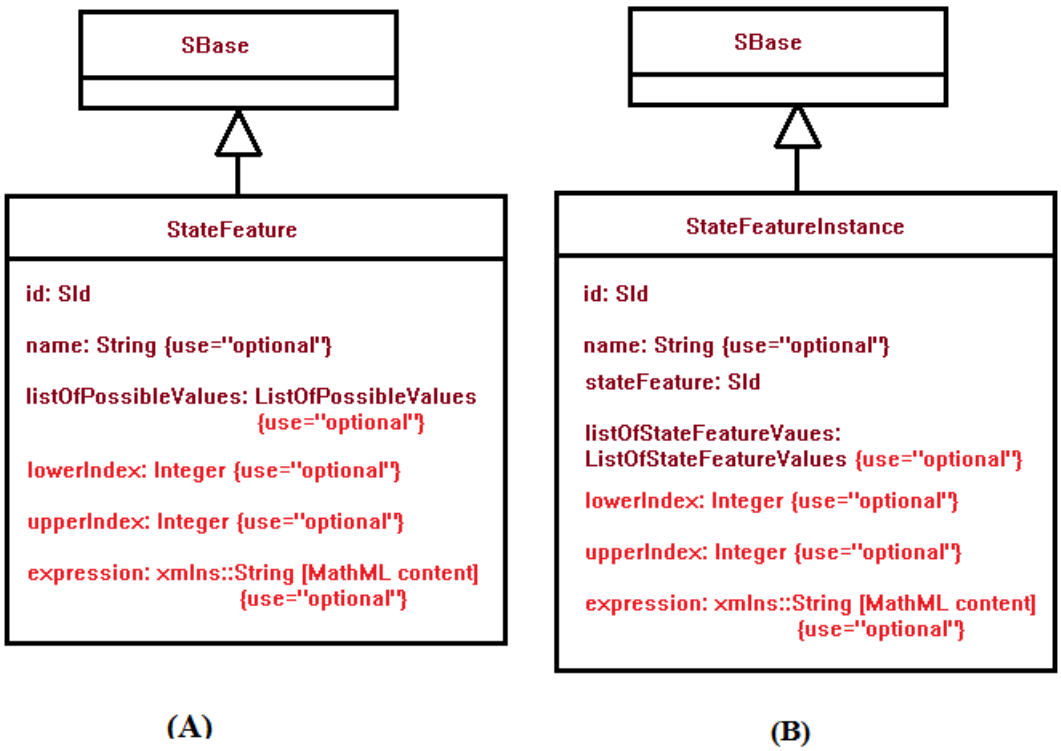

Figure 6.14: Modification proposed for the <*StateFeature*> constructs of the SBML *multi* package (A) StateFeature (B) StateFeatureInstance

6.2 Multistate Modeling in SBML

A goal of JCMB is to store and reuse regulatory network models in a standard way. It uses the standard SBML format for its underlying structure. We have already discussed different kinds of multistate reactions in Chapter 4. In Chapter 3, we discussed the SBML level 3 *multi* package which is still under review by the SBML community. Here we will discuss how we can modify *multi* to include the definition of multistate models containing species with range of indices.

From Figure 3.2 we can see, in the current proposal of *multi* package, each <*SpeciesType*> has <*ListOfStateFeatures*> and each <*StateFeature*> can have a number of <*PossibleValue*>. This is a set of explicitly defined values. We propose to modify the <*StateFeature*> to make option to include a range of values or mathematical expression (Figure 6.14 (A)). Here, the existing <*ListOfPossibleValues*> and the newly added {<*LowerIndex*>, <*UpperIndex*>} or <Expression> constructs can not be used at the same time on one element. <*LowerIndex*> and <*UpperIndex*> both should be present with proper value if either one is present. The <Expression> construct is particularly useful for reactions (we will talk about it shortly).

<*Selector*> is the other construct that needs modification. <*StateFeatureInstance*> under the selector construct represents a list of state feature values which is a list of allowed values for applying restriction related to the reaction or rule. We propose to add {<*LowerIndex*>, <*UpperIndex*>} and <Expression> constructs in addition to the <ListOfStateFeatureValues> (Figure 6.14 (B)). Conditions and constraints are similar to the <*StateFeature*> construct.

The modification in <*StateFeature*> value also has impact on <*SimpleSpeciesReference*> under the <*Reaction*> construct. From Figure 3.4, we can see that <*SimpleSpeciesReference*> eventually links to a <*SpeciesTypeInstance*>, which contains the allowed range of species states to define conditional species reference for the corresponding reaction. According to our proposal (Figure 6.14), a <SpeciesType> can take mathematical expressions which are useful to define reactions similar to $XP\{i\}+E \rightarrow XP\{i+1\}+E$ or $X\{i\}+Y\{j\} \rightarrow XY\{i,j\}$. Here, the variables i, j are considered properties of the reactions and are used to define the allowed range or set of states for reactants and products. In this context, we propose to modify the <*Reaction*> construct (Figure 6.15). Here each <*Reaction*> has an optional <*ListOfLocalVariable*> and each <*LocalVariable*> defines either a range (with {<*LowerLimit*>, <*UpperLimit*>}) or a set (by <*Values*> which is a MathML *Set* construct) of values for that variable. The variable is called local as the scope of it is inside the corresponding reaction only.

We have implemented the multistate enhancement in JCMB using <*annotations*>. The representation for Cdh1 (the multistate species in the Antagonism model) using annotation is shown in Figure 6.16. Similarly, annotations for <*SpeciesReference*> and <*Reactions*> have been used to store index ranges of reactants/products and other reaction related parameters like mechanism or state dependency (Figure 6.17). The resulting SBML code for the summation rule has been shown in Figure 6.18.

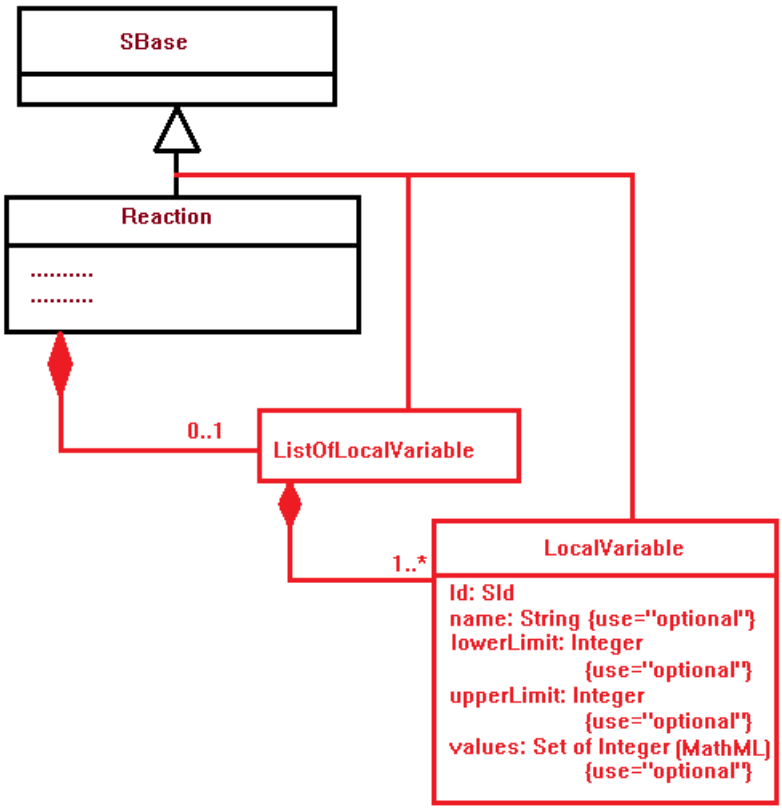

Figure 6.15: Modification proposed for the <*Reaction*> construct of the SBML for *multi* package

```
<listOfSpecies>
  <species id="Clb2_1" name="Clb2" compartment="cell_1" />
  <species id="Cdh1P_1" name="Cdh1P" compartment="cell_1">
   <annotation
    xmlns:jigcell="http://www.sbml.org/2001/ns/jigcell">
    <jigcell:index jigcell:lowerindex="0"
     jigcell:upperindex="10"></jigcell:index>
   </annotation>
  </species>
  <species id="Cdc14_1" name="Cdc14" compartment="cell_1" />
  <species id="Xa_1" name="Xa" compartment="cell_1" />
</listOfSpecies>
```

Figure 6.16: Multistate species definition expressed in SBML using annotation (the *Antagonism Model*)

```
<reaction id="reaction_2" name="reaction_2" fast="false"
 reversible="false">
 <annotation
     xmlns:jigcell="http://www.sbml.org/2001/ns/jigcell" >
     <jigcell:ratelaw jigcell:name="Mass_Action_2_Ordered"
     jigcell:nameset="false"></jigcell:ratelaw>
     <jigcell:rateparams jigcell:o_d="Ordered" jigcell:k_dep=""
     jigcell:nameset="false"></jigcell:rateparams>
     <jigcell:variable jigcell:name="i" jigcell:lowerindex="0"
     jigcell:upperindex="9"></jigcell:variable>
 </annotation>
 <listOfReactants>
  <speciesReference species="Cdh1P_1">
   <annotation
     xmlns:jigcell="http://www.sbml.org/2001/ns/jigcell">
     <jigcell:index jigcell:expression="i"></jigcell:index>
   </annotation>
   </speciesReference>
   <speciesReference species="Clb2_1" />
 </listOfReactants>
 <listOfProducts>
   <speciesReference species="Clb2_1" />
   <speciesReference species="Cdh1P_1">
    <annotation
     xmlns:jigcell="http://www.sbml.org/2001/ns/jigcell">
     <jigcell:index jigcell:expression="i+1"></jigcell:index>
    </annotation>
   </speciesReference>
 </listOfProducts>
 <kineticLaw>
  ... ... ...
 </kineticLaw>
</reaction>
```

Figure 6.17: Multistate reaction definition expressed in SBML using annotation (the *Antagonism Model*)

```
<listOfRules>
  <assignmentRule variable="sum_Cdh1P_1">
       <math:math>
        <math:apply>
         <math:sum/>
             <math:ci>Cdh1Pi_1</math:ci>
               <math:apply>
                <math:lowlimit/><math:cn>1.0</math:cn>
               </math:apply>
               <math:apply>
                <math:uplimit/><math:cn>10.0</math:cn>
               </math:apply>
             <math:ci>Cdh1Pi_1</math:ci>
        </math:apply>
      </math:math>
  </assignmentRule>
  <assignmentRule variable="Xa_1">
    ……
  </assignmentRule>
</listOfRules>
```

Figure 6.18: Summation rule expressed in SBML using MathML and annotation (the *Antagonism Model*)

6.3 Model Flattening

JCMB provides options to save models in multistate format for future editing. It also allows conversion of a multistate model into standard SBML format (Figure 6.19). The process is referred to as flattening the multistate model. As SBML constructs are interrelated, we need to take care of all dependencies. Fortunately, there is a particular order which allows flattening with no dependency violation. Figure 6.19 presents the flow diagram of this process.

The flattening process follows the steps described below to keep the integrity of the flattened model:

1. A standard SBML model is created with non-conflicting constructs like *compartments*, *units* etc.

2. The functions for rate laws generated from the multistate model are converted. If a multistate phosphorylation reaction uses the mass action-ordered rate law, a separate function with identifiable function name and specific equation format is generated in the multistate model. So in the conversion process, name and SId of the function is recreated, the equation is resolved and an entry is given in the function map for identifying and replacing the function in the following steps.

3. Each multistate species is expanded into a list of single state species. Appropriate and distinguishable names are given and a unique SId is created for each single state species in the list. All species from the multistate model are added to the new model.

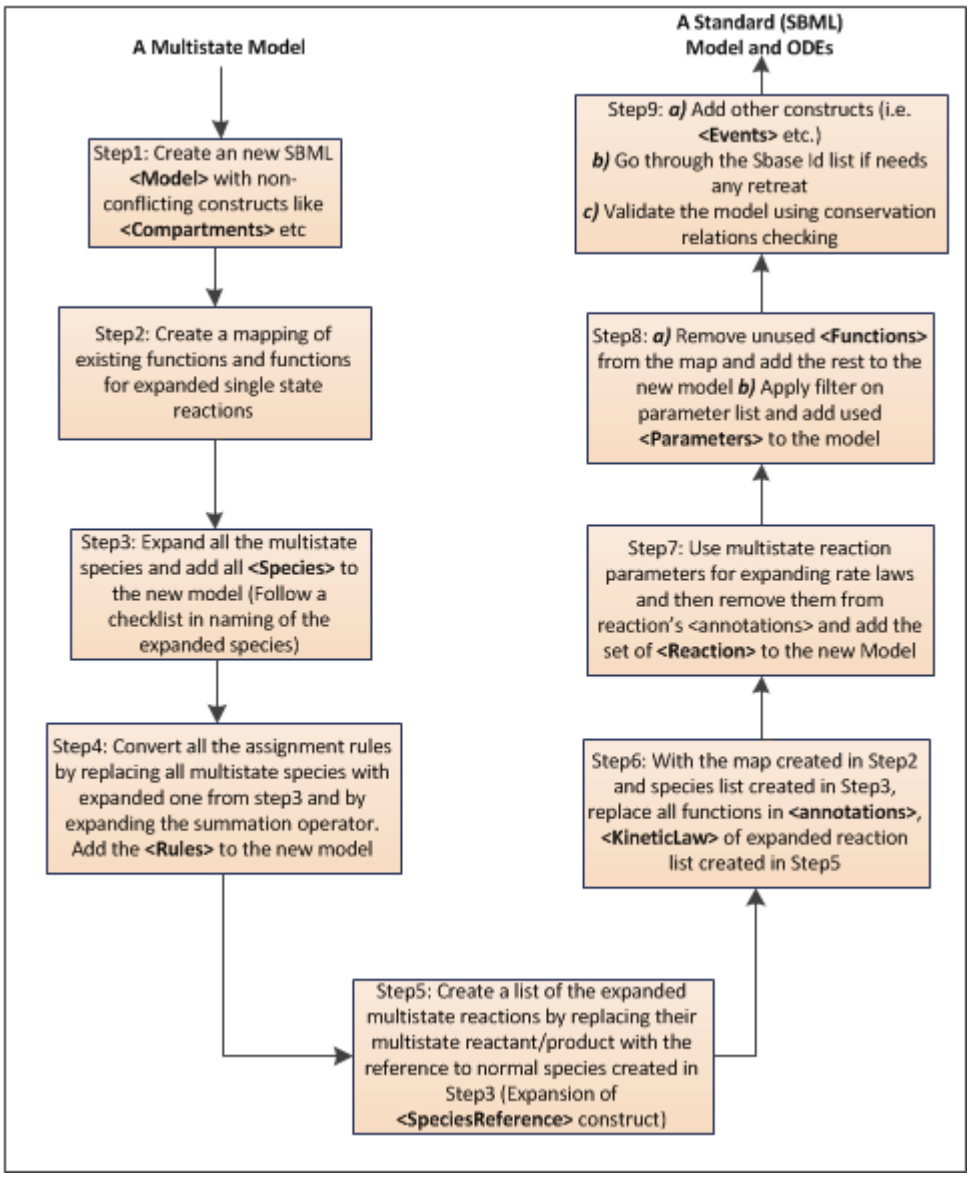

Figure 6.19: The flattening process for a multistate model into a standard SBML model

4. In a multistate model, there might be some rules with a summation operator. In Step 4 the rules are expanded as required and also multistate species are replaced with single state species created in Step 3. For example, the assignment rule $sum\_Cdh1P = sum(Cdh1P\{i\}; 1; 10)$ (as in Figure 6.3) becomes $sum\_Cdh1P = Cdh1P\{1\} + Cdh1P\{2\} + Cdh1P\{3\} + Cdh1P\{4\} + Cdh1P\{5\} + Cdh1P\{6\} + Cdh1P\{7\} + Cdh1P\{8\} + Cdh1P\{9\} + Cdh1P\{10\}$ after the expansion of the summation operator. Then each multistate species $Cdh1P\{1\}$, ... ,$Cdh1P\{10\}$ is replaced with corresponding single state species $Cdh1P\_1$, ..., $Cdh1P\_10$.

5. Each multistate reaction is expanded with replacement of each multistate reactant and product by corresponding single state species. For each reactant/product, the $SpeciesReference$ construct is updated at this step.

6. Function definitions in *annotation* and *kineticlaw* are converted using the map created in Step 2 and species list from Step 3.

7. In multistate modeling parameters are used for various purposes, for instance, for expressing reaction mechanism (ordered or disordered), state dependency (stepwise, exponential or local) etc. In JCMB we store these parameters in the *annotation* field. In this step, these parameters are used to expand the model. For example, the state dependency parameter is used to expand the rate law equation. If a multistate reaction contains an exponential state dependent rate constant $k\{i\} = k_0 r^i$, then using the value of the state variable i the rate constant is evaluated for each single state reaction of the group. Finally those parameters are removed from the *Reaction* construct in the flattened model.

8. Using the map created in Step 2, functions are inserted into the model. Unused parameters are detected from the multistate model and used parameters are inserted in the new model.

9. The rest of the constructs, such as *events* etc., are added, the conservation relations are checked and the overall model integrity is validated.

10. A standard SBML model is generated and list of ODEs can be exported to other simulation tools.

After conversion the new SBML model contains the list of species shown in Figure 6.20, the list of rules shown in Figure 6.21 and the list of reactions as shown previously in Figure 2.2. JCMB also produces a list of ODEs (Figure 6.22) that can be used in simulation tools like *XPPAUT* or *OSCILL8*.

6.4 Compliance with Other Tools and Simulators

To check the integrity and compliance of the flattened SBML model, we loaded the flattened *Antagonism Model* into COPASI and checked the details of the model constructs. It shows

ModelBuilder File Edit Tools

Reactions | Functions | Rules | Compartments | Species | Parameters | Events | Units | Conservation Relations | Equations

Name	Initial Substance	Initial Concentration	Charge	Compartment	Substance Units	Spatial Size Units	Only Substance Units	Boundary Condition	Constant	Port	Notes
Clb2				cell	substance						
Cdh1P_0				cell	substance						
Cdh1P_1				cell	substance						
Cdh1P_2				cell	substance						
Cdh1P_3				cell	substance						
Cdh1P_4				cell	substance						
Cdh1P_5				cell	substance						
Cdh1P_6				cell	substance						
Cdh1P_7				cell	substance						
Cdh1P_8				cell	substance						
Cdh1P_9				cell	substance						
Cdh1P_10				cell	substance						
Cdc14				cell	substance						
Xa				cell	substance						
sum_Cdh1P				cell	substance						

Figure 6.20: List of species after flattening (the *Antagonism Model*)

ModelBuilder File Edit Tools

Reactions | Functions | Rules | Compartments | Species | Parameters | Events | Units | Conservation Relations | Equations

Variable	Type	Equation	Notes
sum_Cdh1P	Assignment	Cdh1P_1 + Cdh1P_2 + Cdh1P_3 + Cdh1P_4 + Cdh1P_5 + Cdh1P_6 + Cdh1P_7 + Cdh1P_8 + Cdh1P_9 + Cdh1P_10	
Xa	Assignment	Ka * Cdh1P_0 + Ki * sum_Cdh1P	

Figure 6.21: List of rules after flattening (the *Antagonism Model*)

ModelBuilder File Edit Tools

Reactions | Functions | Rules | Compartments | Species | Parameters | Events | Units | Conservation Relations | Equations

Variable	Equation	Notes
dClb2/dt	(ks) - (Xa * Clb2)	
dCdh1P_0/dt	- (kp * (Cdh1P_0) * Clb2) + (kh * (Cdh1P_1) * Cdc14) - (kd * (Cdh1P_0)) + (ks1)	
dCdh1P_1/dt	(kp * (Cdh1P_0) * Clb2) - (kp * (Cdh1P_1) * Clb2) - (kh * (Cdh1P_1) * Cdc14) + (kh * (Cdh1P_2) * Cdc14) - (kd * (Cdh1P_1))	
dCdh1P_2/dt	(kp * (Cdh1P_1) * Clb2) - (kp * (Cdh1P_2) * Clb2) - (kh * (Cdh1P_2) * Cdc14) + (kh * (Cdh1P_3) * Cdc14) - (kd * (Cdh1P_2))	
dCdh1P_3/dt	(kp * (Cdh1P_2) * Clb2) - (kp * (Cdh1P_3) * Clb2) - (kh * (Cdh1P_3) * Cdc14) + (kh * (Cdh1P_4) * Cdc14) - (kd * (Cdh1P_3))	
dCdh1P_4/dt	(kp * (Cdh1P_3) * Clb2) - (kp * (Cdh1P_4) * Clb2) - (kh * (Cdh1P_4) * Cdc14) + (kh * (Cdh1P_5) * Cdc14) - (kd * (Cdh1P_4))	
dCdh1P_5/dt	(kp * (Cdh1P_4) * Clb2) - (kp * (Cdh1P_5) * Clb2) - (kh * (Cdh1P_5) * Cdc14) + (kh * (Cdh1P_6) * Cdc14) - (kd * (Cdh1P_5))	
dCdh1P_6/dt	(kp * (Cdh1P_5) * Clb2) - (kp * (Cdh1P_6) * Clb2) - (kh * (Cdh1P_6) * Cdc14) + (kh * (Cdh1P_7) * Cdc14) - (kd * (Cdh1P_6))	
dCdh1P_7/dt	(kp * (Cdh1P_6) * Clb2) - (kp * (Cdh1P_7) * Clb2) - (kh * (Cdh1P_7) * Cdc14) + (kh * (Cdh1P_8) * Cdc14) - (kd * (Cdh1P_7))	
dCdh1P_8/dt	(kp * (Cdh1P_7) * Clb2) - (kp * (Cdh1P_8) * Clb2) - (kh * (Cdh1P_8) * Cdc14) + (kh * (Cdh1P_9) * Cdc14) - (kd * (Cdh1P_8))	
dCdh1P_9/dt	(kp * (Cdh1P_8) * Clb2) - (kp * (Cdh1P_9) * Clb2) - (kh * (Cdh1P_9) * Cdc14) + (kh * (Cdh1P_10) * Cdc14) - (kd * (Cdh1P_9))	
dCdh1P_10/dt	(kp * (Cdh1P_9) * Clb2) - (kh * (Cdh1P_10) * Cdc14) - (kd * (Cdh1P_10))	
dCdc14/dt	0.0	
Xa	Ka * Cdh1P_0 + Ki * sum_Cdh1P	Set by rule
sum_Cdh1P	Cdh1P_1 + Cdh1P_2 + Cdh1P_3 + Cdh1P_4 + Cdh1P_5 + Cdh1P_6 + Cdh1P_7 + Cdh1P_8 + Cdh1P_9 + Cdh1P_10	Set by rule

Figure 6.22: List of ODEs after conversion (the *Antagonism Model*)

that we can use the generated (flattened) SBML files with other standard tools and perform simulation tasks. A snapshot of the COPASI reactions is shown in Figure 6.23.

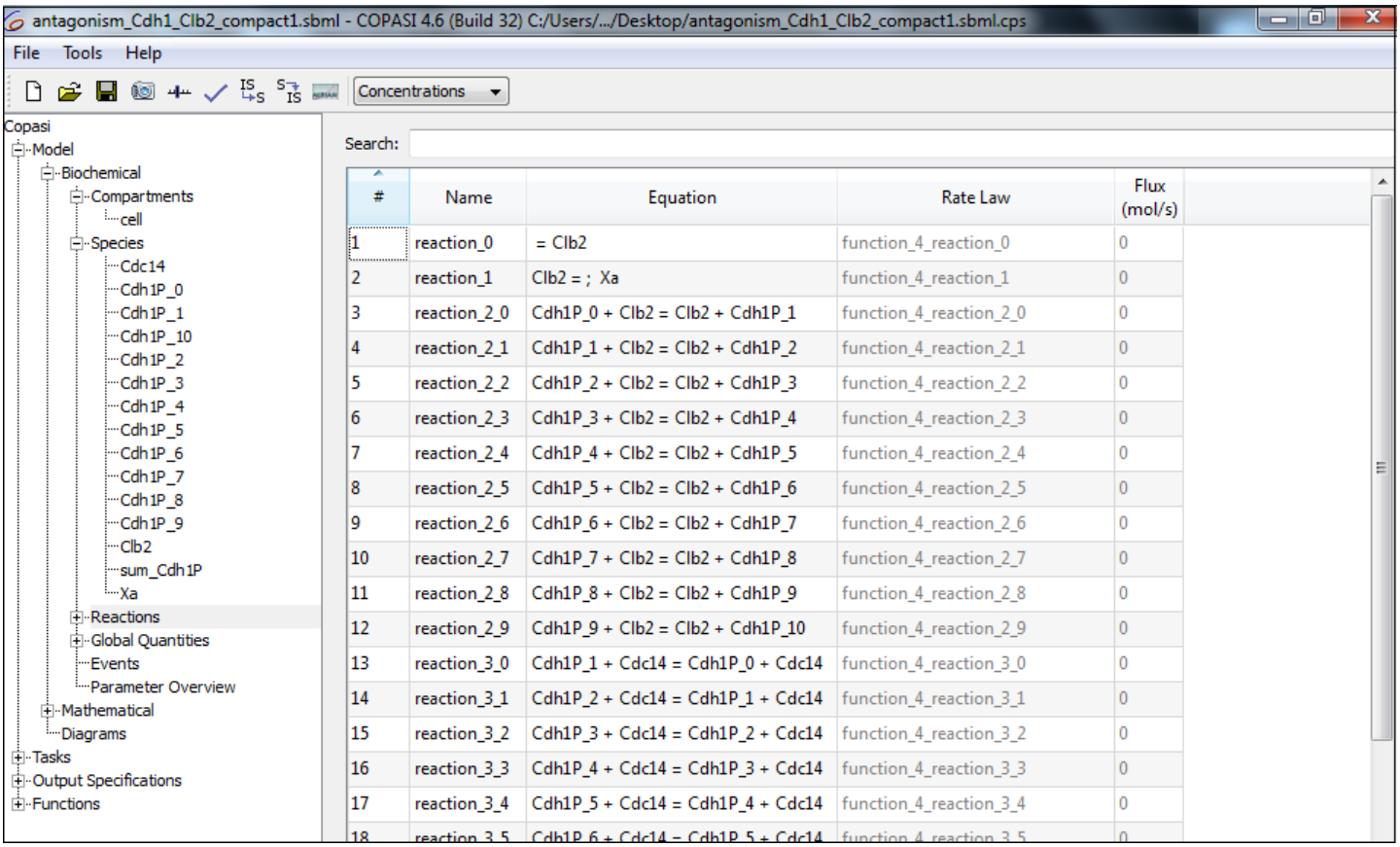

Figure 6.23: Reactions of the *Antagonism Model* shown in the COPASI reaction window

In Chapter 5 we showed the multistate *Cell Cycle Control Model* of the budding yeast (Figures 5.2 and 5.3). Using JCMB we flattened that model as well and generated the list of ODEs (shown in Appendix). We then reproduced the deterministic time course simulation of the ClbM-Cdh1 antagonism part of the model. In this simulation we considered the ODEs generated from JCMB and used the parameter values and initial concentrations of species as described in the paper and supplementary materials of [1]. We considered the exponential growth of cell volume (i.e. *cell*) using the ODE $\frac{dcell}{dt} = \alpha * cell$ and also considered the cell division constraint of the model (i.e. cell is divided when concentration of ClbM goes below 12.5 nM). We then compared our simulation (Figure 6.24 b) with the plot (Figure 6.24 a) from [1] to demonstrate the correctness of the generated ODE list.

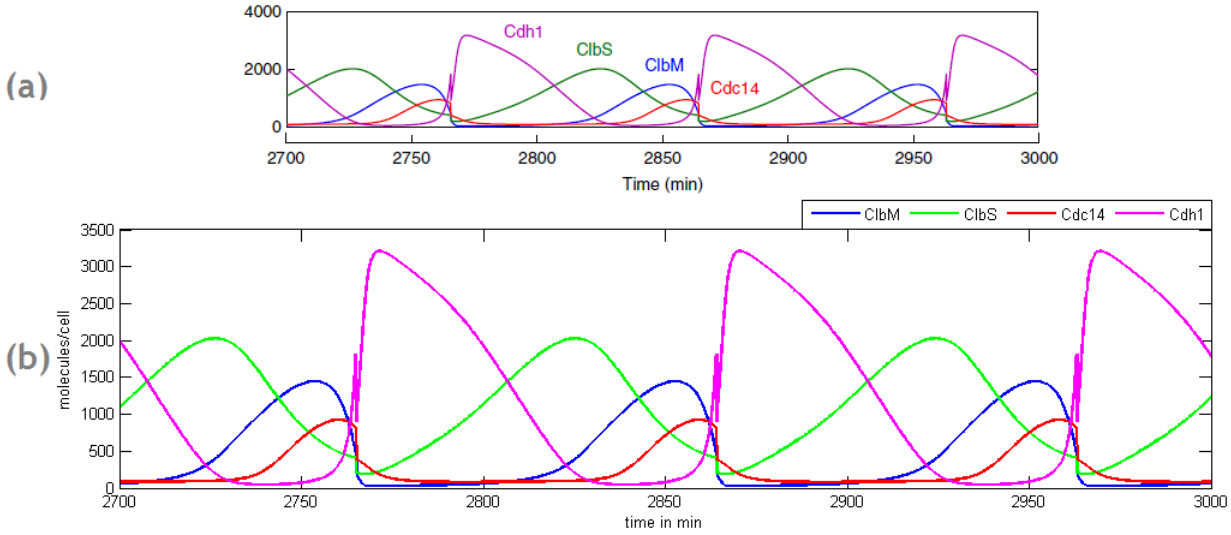

Figure 6.24: Deterministic time course simulation of some key regulatory species (Cdh1, ClbM, ClbS, and Cdc14) of the *Cell Cycle Control Model* of the budding yeast. (a) Plot taken from Figure 3 of [1] (b) Plot generated from the flattened model

Chapter 7

Conclusions and Future Work

In this thesis, we have discussed a representation for multistate species and reactions in regulatory network models. We have analyzed how the approach can be made more compliant and standardized with respect to existing constructs and the requirements of current research papers.

We have implemented a platform for multistate modeling by enhancing the JigCell software tools. Although we have attempted to collect and consider all possible types of multistate reactions, we could not implement all the features of those reactions. For example, we have implemented and tested only the distributive form of phosphorylation reactions in the current tool. But with the extendable structure of the tool it will be easy to implement more reaction types in future when needed. Apart from this, we have not yet tested our models with the concept of set instead of a range of states and also for species with more than two sites. This tool is a means to aid the modelers with current models and to give a platform for future enhancements.

As an extention to the multistate modeling efforts, we aim to integrate the multistate constructs with the concept of model aggregation, proposed by Randhawa et al. [20]. We are looking forward to enhancing aggregation proposal based on the analysis of large realistic models and the current demands of modelers. Our ultimate goal is to fuse the multistate and aggregation concepts together and provide an integrated graphical user interface for modelers where they can create models (single or multistate) in a hierarchical approach.

Bibliography

[1] D. Barik, W. T. Baumann, M. R. Paul, B. Novak, and J. J. Tyson. A model of yeast cell-cycle regulation based on multisite phosphorylation. *Molecular Systems Biology*, 6(405), 2010.

[2] BioNetGen. Accessible at `http://bionetgen.org`.

[3] COmplex PAthway SImulator (COPASI). Accessible at `http://www.copasi.org`.

[4] V. Danos, J. Feret, W. Fontana, and J. Krivine. Scalable simulation of cellular signaling networks. *Lecture Notes in Computer Science*, 4807:139–157, 2007.

[5] J. Faeder, M. Blinov, and W. Hlavacek. Rule-based modeling of biochemical systems with BioNetGen. *Molecular Biology*, 500(2):167–157, 2009.

[6] A. Finney. *Computational Methods in Systems Biology*, volume 3082 of *Lecture Notes in Computer Science*, chapter Developing SBML Beyond Level 2: Proposals for Development, pages 242–247. 2005.

[7] J. Gunawardena. Distributivity and Processivity in Multisite Phosphorylation Can Be Distinguished through Steady-State Invariants. *Biophysical Journal*, 93:3828–3834, 11 2007.

[8] W. Hlavacek, J. Faeder, M. Blinov, R. Posner, M. Hucka, and W. Fontana. Rules for modeling signal-transduction systems. *Science STKE*, 2006.

[9] S. Hoops, S. Sahle, R. Gauges, C. Lee, J. Pahle, N. Simus, M. Singhal, L. Xu, P. Mendes, and U. Kummer. COPASI - a COmplex PAthway SImulator. *Bioinformatics*, 22:3067–74, 2006.

[10] M. Hucka, A. Finney, B. J. Bornstein, S. M. Keating, B. E. Shapiro, J. Matthews, B. L. Kovitz, M. J. Schilstra, A. Funahashi, J. C. Doyle, and H. Kitano. Evolving a lingua franca and associated software infrastructure for computational systems biology: the Systems Biology Markup Language (SBML) project. *IEEE Systems Biology*, 1(1):41–53, 2004.

[11] M Hucka, A Finney, HM Sauro, H Bolouri, JC Doyle, H Kitano, AP Arkin, BJ Bornstein, D Bray, A Cornish-Bowden, AA Cuellar, S Dronov, ED Gilles, M Ginkel, V Gor, II Goryanin, WJ Hedley, TC Hodgman, JH Hofmeyr, PJ Hunter, NS Juty, JL Kasberger, A Kremling, U Kummer, N Le Novre, LM Loew, D Lucio, P Mendes, E Minch, ED Mjolsness, Y Nakayama, MR Nelson, PF Nielsen, T Sakurada, JC Schaff, BE Shapiro, TS Shimizu, HD 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*, 9(4):524–531, 2003.

[12] JigCell. Accessible at `http://jigcell.cs.vt.edu`.

[13] R. N. Jorissen, F. Walker, N. Pouliot, T. P. J. Garrett, C. W. Ward, and A. W. Burgess. Epidermal growth factor receptor: Mechanisms of activation and signalling. *Experimental Cell Research*, 284:31–53, 2003.

[14] S. Marjan Varedi K, Alejandra C. Ventura, Sofia D. Merajver, and Xiaoxia Nina Lin. Multisite Phosphorylation Provides an Effective and Flexible Mechanism for Switch-Like Protein Degradation. *PLoS One*, 5(12)(e14029), 12 2010.

[15] O. Kapuy, D. Barik, M. R. D. Sananes, J. J. Tyson, and B. Novak. Bistability by Multiple Phosphorylation of Regulatory Proteins. *Progress in Biophysics and Molecular Biology*, 100(1-3):47–56, 2009.

[16] A. Mallavarapu, M. Thomson, B. Ullian, and J. Gunawardena. Programming with models: modularity and abstraction provide powerful capabilities for systems biology. *Journal of the Royal Society Interface*, 6:257–270, 2009.

[17] P. Nash, X. Tang, S. Orlicky, Q. Chen, FB Gertler, MD Mendenhall, F. Sicheri, T. Pawson, and M. Tyers. Multisite phosphorylation of a CDK inhibitor sets a threshold for the onset of DNA replication. *Nature*, 414(6863):514–521, 2001.

[18] Oscill8. Accessible at `http://oscill8.sourceforge.net`.

[19] P. Patwardhan and W. T. Miller. Processive phosphorylation: mechanism and biological importance. *Cell Signal*, 19:2218–2226, 11 2007.

[20] R. Randhawa, C. A. Shaffer, and J. J. Tyson. Model aggregation: a building-block approach to creating large macromolecular regulatory networks. *Bioinformatics*, 25:3289–3295, 2009.

[21] RuleBender: Rule Based Modeling and Simulation. Accessible at `http://rulebender.cs.pitt.edu/`.

[22] SBML: The Systems Biology Markup Language. Accessible at `http://sbml.org/Main_Page`.

[23] SBML Level 3 Version 1 Core. Accessible at `http://sbml.org/Documents/Specifications#SBML_Level_3_Version_1_Core`.

[24] SBML Level 3 Multi Package Proposal. Accessible at `http://sbml.org/Community/Wiki/SBML_Level_3_Proposals/Multistate_and_Multicomponent_Species_Proposal`.

[25] C. A. Shaffer, J. W. Zwolak, R. Randhawa, and J. J. Tyson. Modeling Molecular Regulatory Networks with JigCell and PET. *Systems Biology*, 500(4), 2009.

[26] D. G. Shin, L. Liu, L. M. Loew, and J. Schaff. Virtual Cell: a general framework for simulating and visualizing cellular physiology. In *4th IFIP 2.6 Working Conference on Visual Database Systems*, L'Aquila, Italy, 1998.

[27] M. Sneddon, W. Pontius, J. Faeder, and T. Emonet. NFsim: Managing complexity in stochastic simulation of reaction networks. In *2nd Q-Bio Conference*, Santa Fe, NM, 2008.

[28] X windows Phase Plane plus AUTo (XPPAUT). Accessible at `http://www.math.pitt.edu/~bard/xpp/xpp.html`.

[29] J. Yang, M. Monine, J. Faeder, and W. Hlavacek. Kinetic monte carlo method for rule-based modeling of biochemical networks. *Physical Review*, 78(031910), 2008.

Appendix

This appendix lists the set of ODEs generated using JCMB from the *Cell Cycle Control Model* of budding yeast.

$$
\begin{aligned}
cdh1pT &= cdh1\_1 + cdh1\_2 + cdh1\_3 + cdh1\_4 + cdh1\_5 \\
&\quad + cdh1\_6 + cdh1\_7 + cdh1\_8 + cdh1\_9 + cdh1\_10 \\
\frac{dclbm}{dt} &= ksbm * (cell/PI) * mbm - (kdbma * cdh1\_0 * clbm)/cell \\
&\quad - (kdbmi * cdh1pT * clbm)/cell - gdbm * clbm \\
\frac{dcdh1\_0}{dt} &= (ksh1 * (cell/PI) * mh1) + kdh1p * cdh1\_1 + (kdh1 * cdh1\_1 * cdc14)/cell \\
&\quad - gdh1 * cdh1\_0 - (kph1 * cdh1\_0 * clbs)/cell - (kph1p * cdh1\_0 * clbm)/cell \\
\frac{dcdh1\_1}{dt} &= -kdh1p * cdh1\_1 + kdh1p * cdh1\_2 - (kdh1 * cdh1\_1 * cdc14)/cell \\
&\quad + (kdh1 * cdh1\_2 * cdc14)/cell - gdh1 * cdh1\_1 + (kph1 * cdh1\_0 * clbs)/cell \\
&\quad - (kph1 * cdh1\_1 * clbs)/cell + (kph1p * cdh1\_0 * clbm)/cell \\
&\quad - (kph1p * cdh1\_1 * clbm)/cell \\
\frac{dcdh1\_2}{dt} &= -kdh1p * cdh1\_2 + kdh1p * cdh1\_3 - (kdh1 * cdh1\_2 * cdc14)/cell \\
&\quad + (kdh1 * cdh1\_3 * cdc14)/cell - gdh1 * cdh1\_2 + (kph1 * cdh1\_1 * clbs)/cell \\
&\quad - (kph1 * cdh1\_2 * clbs)/cell + (kph1p * cdh1\_1 * clbm)/cell \\
&\quad - (kph1p * cdh1\_2 * clbm)/cell \\
\frac{dcdh1\_3}{dt} &= -kdh1p * cdh1\_3 + kdh1p * cdh1\_4 - (kdh1 * cdh1\_3 * cdc14)/cell \\
&\quad + (kdh1 * cdh1\_4 * cdc14)/cell - gdh1 * cdh1\_3 + (kph1 * cdh1\_2 * clbs)/cell \\
&\quad - (kph1 * cdh1\_3 * clbs)/cell + (kph1p * cdh1\_2 * clbm)/cell \\
&\quad - (kph1p * cdh1\_3 * clbm)/cell \\
\frac{dcdh1\_4}{dt} &= -kdh1p * cdh1\_4 + kdh1p * cdh1\_5 - (kdh1 * cdh1\_4 * cdc14)/cell \\
&\quad + (kdh1 * cdh1\_5 * cdc14)/cell - gdh1 * cdh1\_4 + (kph1 * cdh1\_3 * clbs)/cell \\
&\quad - (kph1 * cdh1\_4 * clbs)/cell + (kph1p * cdh1\_3 * clbm)/cell
\end{aligned}
$$

$$
\begin{aligned}
& -(kph1p * cdh1\_4 * clbm)/cell \\
\frac{dcdh1\_5}{dt} &= -kdh1p * cdh1\_5 + kdh1p * cdh1\_6 - (kdh1 * cdh1\_5 * cdc14)/cell \\
& +(kdh1 * cdh1\_6 * cdc14)/cell - gdh1 * cdh1\_5 + (kph1 * cdh1\_4 * clbs)/cell \\
& -(kph1 * cdh1\_5 * clbs)/cell + (kph1p * cdh1\_4 * clbm)/cell \\
& -(kph1p * cdh1\_5 * clbm)/cell \\
\frac{dcdh1\_6}{dt} &= -kdh1p * cdh1_6 + kdh1p * cdh1\_7 - (kdh1 * cdh1_6 * cdc14)/cell \\
& +(kdh1 * cdh1\_7 * cdc14)/cell - gdh1 * cdh1_6 + (kph1 * cdh1\_5 * clbs)/cell \\
& -(kph1 * cdh1\_6 * clbs)/cell + (kph1p * cdh1\_5 * clbm)/cell \\
& -(kph1p * cdh1\_6 * clbm)/cell \\
\frac{dcdh1\_7}{dt} &= -(kdh1p * cdh1\_7) + (kdh1p * cdh1\_8) - (kdh1 * cdh1\_7 * cdc14)/cell \\
& +(kdh1 * cdh1\_8 * cdc14)/cell - (gdh1 * cdh1\_7) + (kph1 * cdh1\_6 * clbs)/cell \\
& -(kph1 * cdh1\_7 * clbs)/cell + (kph1p * cdh1\_6 * clbm)/cell \\
& -(kph1p * cdh1\_7 * clbm)/cell \\
\frac{dcdh1\_8}{dt} &= -(kdh1p * cdh1\_8) + (kdh1p * cdh1\_9) - (kdh1 * cdh1\_8 * cdc14)/cell \\
& +(kdh1 * cdh1\_9 * cdc14)/cell - (gdh1 * cdh1\_8) + (kph1 * cdh1\_7 * clbs)/cell \\
& -(kph1 * cdh1\_8 * clbs)/cell + (kph1p * cdh1\_7 * clbm)/cell \\
& -(kph1p * cdh1\_8 * clbm)/cell \\
\frac{dcdh1\_9}{dt} &= -(kdh1p * cdh1\_9) + (kdh1p * cdh1\_10) - (kdh1 * cdh1\_9 * cdc14)/cell \\
& +(kdh1 * cdh1\_10 * cdc14)/cell - (gdh1 * cdh1\_9) + (kph1 * cdh1\_8 * clbs)/cell \\
& -(kph1 * cdh1\_9 * clbs)/cell + (kph1p * cdh1\_8 * clbm)/cell \\
& -(kph1p * cdh1\_9 * clbm)/cell \\
\frac{dcdh1\_10}{dt} &= -(kdh1p * cdh1\_10) - (kdh1 * cdh1\_10 * cdc14)/cell - (gdh1 * cdh1\_10) \\
& +(kph1 * cdh1\_9 * clbs)/cell + (kph1p * cdh1\_9 * clbm)/cell \\
\frac{dcdc14}{dt} &= ((ksc14 * (cell/PI) * mc14) - (gd14 * cdc14) - (kar * cdc14 * net1\_0)/cell \\
& -(kar * cdc14 * net1\_1)/cell - (kar * cdc14 * net1\_2)/cell - (kar * cdc14 * net1\_3)/cell \\
& -(kar * cdc14 * net1\_4)/cell - (kar * cdc14 * net1\_5)/cell + (gdt1 * rnt\_0) \\
& +(gdt1 * rnt\_1) + (gdt1 * rnt\_2) + (gdt1 * rnt\_3) + (gdt1 * rnt\_4) + (gdt1 * rnt\_5) \\
& +(kdr * rnt\_0) + (kdr * rnt\_1) + (kdr * rnt\_2) + (kdr * rnt\_3) \\
& +(kdr * rnt\_4) + (kdr * rnt\_5) \\
\frac{dclbs}{dt} &= ((ksbs * (cell/PI) * mbs) - (gdbs * clbs)
\end{aligned}
$$

$$
\begin{aligned}
\frac{dmbm}{dt} &= (ksmbm * PI) - (gdmbm * mbm) \\
\frac{dmh1}{dt} &= (ksmh1 * PI) - (gdmh1 * mh1) \\
\frac{dmn3}{dt} &= (ksmn3 * PI) - (gdmn3 * mn3) \\
\frac{dcln3}{dt} &= (ksn3 * (cell/PI) * (cell/PI) * mn3) - (gdn3 * cln3) \\
\frac{dnet1\_0}{dt} &= -(kar * cdc14 * net1\_0)/cell + (kdr * rnt\_0) + ((kst1 * (cell/PI) * mt1) \\
&\quad +(kdt1 * net1\_1 * ht1)/cell - (kpt1 * net1\_0 * clbm)/cell \\
&\quad -(gdt1 * net1\_0) + (gd14 * rnt\_0) \\
\frac{dnet1\_1}{dt} &= -(kar * cdc14 * net1\_1)/cell + (kdr * rnt\_1) - (kdt1 * net1\_1 * ht1)/cell \\
&\quad +(kdt1 * net1\_2 * ht1)/cell + (kpt1 * net1\_0 * clbm)/cell \\
&\quad -(kpt1 * net1\_1 * clbm)/cell - (gdt1 * net1\_1) + (gd14 * rnt\_1) \\
\frac{dnet1\_2}{dt} &= -(kar * cdc14 * net1\_2)/cell + (kdr * rnt\_2) - (kdt1 * net1\_2 * ht1)/cell \\
&\quad +(kdt1 * net1\_3 * ht1)/cell + (kpt1 * net1\_1 * clbm)/cell \\
&\quad -(kpt1 * net1\_2 * clbm)/cell - (gdt1 * net1\_2) + (gd14 * rnt\_2) \\
\frac{dnet1\_3}{dt} &= -(kar * cdc14 * net1\_3)/cell + (kdr * rnt\_3) - (kdt1 * net1\_3 * ht1)/cell \\
&\quad +(kdt1 * net1\_4 * ht1)/cell + (kpt1 * net1\_2 * clbm)/cell \\
&\quad -(kpt1 * net1\_3 * clbm)/cell - (gdt1 * net1\_3) + (gd14 * rnt\_3) \\
\frac{dnet1\_4}{dt} &= -(kar * cdc14 * net1\_4)/cell + (kdr * rnt\_4) - (kdt1 * net1\_4 * ht1)/cell \\
&\quad +(kdt1 * net1\_5 * ht1)/cell + (kpt1 * net1\_3 * clbm)/cell \\
&\quad -(kpt1 * net1\_4 * clbm)/cell - (gdt1 * net1\_4) + (gd14 * rnt\_4) \\
\frac{dnet1\_5}{dt} &= -(kar * cdc14 * net1\_5)/cell + (kdr * rnt\_5) - (kdt1 * net1\_5 * ht1)/cell \\
&\quad +(kdt1 * net1\_6 * ht1)/cell + (kpt1 * net1\_4 * clbm)/cell \\
&\quad -(kpt1 * net1\_5 * clbm)/cell - (gdt1 * net1\_5) + (gd14 * rnt\_5) \\
\frac{dnet1\_6}{dt} &= -(kdt1 * net1\_6 * ht1)/cell + (kdt1 * net1\_7 * ht1)/cell \\
&\quad +(kpt1 * net1\_5 * clbm)/cell - (kpt1 * net1\_6 * clbm)/cell - (gdt1 * net1\_6) \\
\frac{dnet1\_7}{dt} &= -(kdt1 * net1\_7 * ht1)/cell + (kdt1 * net1\_8 * ht1)/cell \\
&\quad +(kpt1 * net1\_6 * clbm)/cell - (kpt1 * net1\_7 * clbm)/cell - (gdt1 * net1\_7) \\
\frac{dnet1\_8}{dt} &= -(kdt1 * net1\_8 * ht1)/cell
\end{aligned}
$$

$$
\begin{aligned}
& +(kpt1 * net1\_7 * clbm)/cell - (gdt1 * net1\_8) \\
\frac{drnt\_0}{dt} &= (kar * cdc14 * net1\_0)/cell - (gdt1 * rnt\_0) - (kdr * rnt\_0) \\
& -(gd14 * rnt\_0) - (kpnt * rnt\_0 * clbm)/cell + (kdnt * rnt\_1 * ht1)/cell \\
\frac{drnt\_1}{dt} &= (kar * cdc14 * net1\_1)/cell - (gdt1 * rnt\_1) - (kdr * rnt\_1) \\
& -(gd14 * rnt\_1) + (kpnt * rnt\_0 * clbm)/cell - (kpnt * rnt\_1 * clbm)/cell \\
& -(kdnt * rnt\_1 * ht1)/cell + (kdnt * rnt\_2 * ht1)/cell \\
\frac{drnt\_2}{dt} &= (kar * cdc14 * net1\_2)/cell - (gdt1 * rnt\_2) - (kdr * rnt\_2) \\
& -(gd14 * rnt\_2) + (kpnt * rnt\_1 * clbm)/cell - (kpnt * rnt\_2 * clbm)/cell \\
& -(kdnt * rnt\_2 * ht1)/cell + (kdnt * rnt\_3 * ht1)/cell \\
\frac{drnt\_3}{dt} &= (kar * cdc14 * net1\_3)/cell - (gdt1 * rnt\_3) - (kdr * rnt\_3) \\
& -(gd14 * rnt\_3) + (kpnt * rnt\_2 * clbm)/cell - (kpnt * rnt\_3 * clbm)/cell \\
& -(kdnt * rnt\_3 * ht1)/cell + (kdnt * rnt\_4 * ht1)/cell \\
\frac{drnt\_4}{dt} &= (kar * cdc14 * net1\_4)/cell - (gdt1 * rnt\_4) - (kdr * rnt\_4) \\
& -(gd14 * rnt\_4) + (kpnt * rnt\_3 * clbm)/cell - (kpnt * rnt\_4 * clbm)/cell \\
& -(kdnt * rnt\_4 * ht1)/cell + (kdnt * rnt\_5 * ht1)/cell \\
\frac{drnt\_5}{dt} &= (kar * cdc14 * net1\_5)/cell - (gdt1 * rnt\_5) - (kdr * rnt\_5) \\
& -(gd14 * rnt\_5) + (kpnt * rnt\_4 * clbm)/cell - (kdnt * rnt\_5 * ht1)/cell \\
\frac{dsbf\_0}{dt} &= (ksbf * cell - (kac * sbf\_0 * whi5\_0)/cell - (kac * sbf\_0 * whi5\_1)/cell \\
& -(kac * sbf\_0 * whi5\_2)/cell + (kdc * cmp\_0) + (kdc * cmp\_1) \\
& +(kdc * cmp\_2) + (gdi5 * cmp\_0) + (gdi5 * cmp\_1) + (gdi5 * cmp\_2) \\
& +(kdbf * sbf\_1 * hbf)/cell + (kdbfp * sbf\_1 * cdc14)/cell - (gdbf * sbf\_0) \\
& -(kpbf * sbf\_0 * clbm)/cell - (kag * (sbf\_0/cell * (PI - ga))) \\
& +((kag * (sbf\_0/cell * (PI - ga)) \\
\frac{dsbf\_1}{dt} &= -(kdbf * sbf\_1 * hbf)/cell + (kdbf * sbf\_2 * hbf)/cell - (kdbfp * sbf\_1 * cdc14)/cell \\
& +(kdbfp * sbf\_2 * cdc14)/cell - (gdbf * sbf\_1) \\
& +(kpbf * sbf\_0 * clbm)/cell - (kpbf * sbf\_1 * clbm)/cell \\
\frac{dsbf\_2}{dt} &= -(kdbf * sbf\_2 * hbf)/cell + (kdbf * sbf\_3 * hbf)/cell - (kdbfp * sbf\_2 * cdc14)/cell \\
& +(kdbfp * sbf\_3 * cdc14)/cell - (gdbf * sbf\_2) \\
& +(kpbf * sbf\_1 * clbm)/cell - (kpbf * sbf\_2 * clbm)/cell
\end{aligned}
$$

$$
\begin{aligned}
\frac{dsbf\_3}{dt} &= -(kdbf * sbf\_3 * hbf)/cell + (kdbf * sbf\_4 * hbf)/cell - (kdbfp * sbf\_3 * cdc14)/cell \\
&\quad +(kdbfp * sbf\_4 * cdc14)/cell - (gdbf * sbf\_3) \\
&\quad +(kpbf * sbf\_2 * clbm)/cell - (kpbf * sbf\_3 * clbm)/cell \\
\frac{dsbf\_4}{dt} &= -(kdbf * sbf\_4 * hbf)/cell - (kdbfp * sbf\_4 * cdc14)/cell \\
&\quad -(gdbf * sbf\_4) + (kpbf * sbf\_3 * clbm)/cell \\
\frac{dwhi5\_0}{dt} &= -(kac * sbf\_0 * whi5\_0)/cell + (kdc * cmp\_0) + (kdi5p * whi5\_1 * cdc14)/cell \\
&\quad +(kdi5 * whi5\_1 * hi5)/cell - (kpi5 * whi5\_0 * cln3)/cell \\
&\quad -(kpi5p * whi5\_0 * clbs)/cell + ((ksi5 * (cell/PI) * mi5) \\
&\quad -(gdi5 * whi5\_0) + (gdbf * cmp\_0) \\
\frac{dwhi5\_1}{dt} &= -(kac * sbf\_0 * whi5\_1)/cell + (kdc * cmp\_1) - (kdi5p * whi5\_1 * cdc14)/cell \\
&\quad +(kdi5p * whi5\_2 * cdc14)/cell - (kdi5 * whi5\_1 * hi5)/cell + (kdi5 * whi5\_2 * hi5)/cell \\
&\quad +(kpi5 * whi5\_0 * cln3)/cell - (kpi5 * whi5\_1 * cln3)/cell + (kpi5p * whi5\_0 * clbs)/cell \\
&\quad -(kpi5p * whi5\_1 * clbs)/cell - (gdi5 * whi5\_1) + (gdbf * cmp\_1) \\
\frac{dwhi5\_2}{dt} &= -(kac * sbf\_0 * whi5\_2)/cell + (kdc * cmp\_2) - (kdi5p * whi5\_2 * cdc14)/cell \\
&\quad +(kdi5p * whi5\_3 * cdc14)/cell - (kdi5 * whi5\_2 * hi5)/cell + (kdi5 * whi5\_3 * hi5)/cell \\
&\quad +(kpi5 * whi5\_1 * cln3)/cell - (kpi5 * whi5\_2 * cln3)/cell + (kpi5p * whi5\_1 * clbs)/cell \\
&\quad -(kpi5p * whi5\_2 * clbs)/cell - (gdi5 * whi5\_2) + (gdbf * cmp\_2) \\
\frac{dwhi5\_3}{dt} &= -(kdi5p * whi5\_3 * cdc14)/cell + (kdi5p * whi5\_4 * cdc14)/cell \\
&\quad -(kdi5 * whi5\_3 * hi5)/cell + (kdi5 * whi5\_4 * hi5)/cell + (kpi5 * whi5\_2 * cln3)/cell \\
&\quad -(kpi5 * whi5\_3 * cln3)/cell + (kpi5p * whi5\_2 * clbs)/cell \\
&\quad -(kpi5p * whi5\_3 * clbs)/cell - (gdi5 * whi5\_3) \\
\frac{dwhi5\_4}{dt} &= -(kdi5p * whi5\_4 * cdc14)/cell + (kdi5p * whi5\_5 * cdc14)/cell \\
&\quad -(kdi5 * whi5\_4 * hi5)/cell + (kdi5 * whi5\_5 * hi5)/cell \\
&\quad +(kpi5 * whi5\_3 * cln3)/cell - (kpi5 * whi5\_4 * cln3)/cell \\
&\quad +(kpi5p * whi5\_3 * clbs)/cell - (kpi5p * whi5\_4 * clbs)/cell - (gdi5 * whi5\_4) \\
\frac{dwhi5\_5}{dt} &= -(kdi5p * whi5\_5 * cdc14)/cell + (kdi5p * whi5\_6 * cdc14)/cell \\
&\quad -(kdi5 * whi5\_5 * hi5)/cell + (kdi5 * whi5\_6 * hi5)/cell + (kpi5 * whi5\_4 * cln3)/cell \\
&\quad -(kpi5 * whi5\_5 * cln3)/cell + (kpi5p * whi5\_4 * clbs)/cell \\
&\quad -(kpi5p * whi5\_5 * clbs)/cell - (gdi5 * whi5\_5) \\
\frac{dwhi5\_6}{dt} &= -(kdi5p * whi5\_6 * cdc14)/cell - (kdi5 * whi5\_6 * hi5)/cell
\end{aligned}
$$

$$
\begin{aligned}
& & & +(kpi5 * whi5\_5 * cln3)/cell + (kpi5p * whi5\_5 * clbs)/cell - (gdi5 * whi5\_6) \\
\frac{dcmp\_0}{dt} &= & & (kac * sbf\_0 * whi5\_0)/cell - (kdc * cmp\_0) - (gdi5 * cmp\_0) - (gdbf * cmp\_0) \\
& & & +(kdcmp * cmp\_1 * cdc14)/cell + (kdcm * cmp\_1 * hi5)/cell \\
& & & -(kpcmp * cmp\_0 * clbs)/cell - (kpcm * cmp\_0 * cln3)/cell \\
\frac{dcmp\_1}{dt} &= & & (kac * sbf\_0 * whi5\_1)/cell - kdc * cmp\_1 - gdi5 * cmp\_1 - gdbf * cmp\_1 \\
& & & -(kdcmp * cmp\_1 * cdc14)/cell + (kdcmp * cmp\_2 * cdc14)/cell \\
& & & -(kdcm * cmp\_1 * hi5)/cell + (kdcm * cmp\_2 * hi5)/cell \\
& & & +(kpcmp * cmp\_0 * clbs)/cell - (kpcmp * cmp\_1 * clbs)/cell \\
& & & +(kpcm * cmp\_0 * cln3)/cell - (kpcm * cmp\_1 * cln3)/cell \\
\frac{dcmp\_2}{dt} &= & & (kac * sbf\_0 * whi5\_2)/cell - kdc * cmp\_2 - gdi5 * cmp\_2 - gdbf * cmp\_2 \\
& & & -(kdcmp * cmp\_2 * cdc14)/cell - (kdcm * cmp\_2 * hi5)/cell \\
& & & +(kpcmp * cmp\_1 * clbs)/cell + (kpcm * cmp\_1 * cln3)/cell \\
\frac{dhbf}{dt} &= & & kshbf * (cell/PI) * mhbf - gdhbf * hbf \\
\frac{dhi5}{dt} &= & & kshi5 * (cell/PI) * mhi5 - gdhi5 * hi5) \\
\frac{dht1}{dt} &= & & ksht1 * (cell/PI) * mht1 - gdht1 * ht1 \\
\frac{dga}{dt} &= & & kag * (sbf_0/cell) * (PI - ga) - kdg * ga \\
\frac{dmbs}{dt} &= & & ksmbs * ga - gdmbs * mbs \\
\frac{dmc14}{dt} &= & & ksmc14 * PI - gdmc14 * mc14 \\
\frac{dmhbf}{dt} &= & & ksmhbf * PI - gdmhbf * mhbf \\
\frac{dmi5}{dt} &= & & ksmi5 * PI - gdmi5 * mi5 \\
\frac{dmhi5}{dt} &= & & ksmhi5 * PI - gdmhi5 * mhi5 \\
\frac{dmt1}{dt} &= & & ksmt1 * PI - gdmt1 * mt1 \\
\frac{dmht1}{dt} &= & & ksmht1 * PI - gdmht1 * mht1
\end{aligned}
$$