MODEL ABSTRACTION AND TEMPORAL BEHAVIOR ANALYSIS OF GENETIC REGULATORY NETWORKS

by

Hiroyuki Kuwahara

A dissertation submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

School of Computing

The University of Utah

December 2007

THE UNIVERSITY OF UTAH GRADUATE SCHOOL

SUPERVISORY COMMITTEE APPROVAL

of a dissertation submitted by

Hiroyuki Kuwahara

This dissertation has been read by each member of the following supervisory committee and by majority vote has been found to be satisfactory.

	Chair: Chris J. Myers
	James Keener
	Gary Lindstrom
	Michael Samoilov
	Konrad Slind

THE UNIVERSITY OF UTAH GRADUATE SCHOOL

FINAL READING APPROVAL

To the Graduate Council of the University of Utah:

I have read the dissertation of Hiroyuki Kuwahara in its final form and have found that (1) its format, citations, and bibliographic style are consistent and acceptable; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the Supervisory Committee and is ready for submission to The Graduate School.

Date

Chris J. Myers
Chair: Supervisory Committee

Approved for the Major Department

Martin Berzins
Chair/Director

Approved for the Graduate Council

David S. Chapman
Dean of The Graduate School

ABSTRACT

With advances in technologies such as high throughput data collection and genome sequencing methods, the Human Genome Project which, among other things, determined the human DNA sequence and identified all the genes in human DNA was completed at least two years ahead of time. As these technologies are becoming more accurate, efficient, and cost effective and a massive amount of genomic and proteomic data are becoming available at a rapid pace, we are now in the position to face the challenge to understand how these genes coupled with environmental stimuli orchestrate the regulation of cell-level behaviors. However, understanding such complex systems is very expensive and is most likely impractical with wet-lab experiments alone as the amount and the complexity of data substantially increase, requiring the integration of computational methods to make the process more efficient.

To allow for substantial acceleration in computational analysis, this dissertation develops a model abstraction methodology for biochemical systems which systematically performs various model abstractions to reduce the complexity of computational biochemical models. Our methodology is particularly useful for systems with small molecular counts that require the discrete and stochastic representation and thus demand substantial computational requirements. As a case study, this dissertation illustrates the application of individual abstraction methods to such systems. Furthermore, it demonstrates the application of collective abstraction methods at various accuracy levels to temporal behavior analysis of several genetic regulatory networks. This dissertation shows that analysis time of biologically relevant properties of such genetic regulatory networks can be improved from days of work to minutes of work using our methodology while maintaining reasonable accuracy.

To Aki and Reina

CONTENTS

APPENDICES

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor, Professor Chris J. Myers, for his support and direction, which significantly improved the quality of my research. I am thankful to Dr. Cynthia Thompson for giving me the opportunity to work as an undergraduate research assistant, which helped me decide to go to graduate school. Professor Konrad Slind and Professor Gary Lindstrom supported me tremendously to survive the initial phase of my graduate school, and I would like to thank them for that. I would like to thank Professor James Keener for his expertise in the field of Mathematical Physiology and numerous stimulating discussions on my work. I enjoyed and learned a lot from my collaboration with Dr. Michael Samoilov. I am indebted to him for his guidance and knowledge that substantially enhanced the quality of my work.

Numerous individuals also provided invaluable support. I would like to thank my officemates and the members of Chris' bio group (Nathan Barker, Curtis Madsen, and Nam Nguyen). I am grateful to Karen Feinauer for her administrative help that allowed me to better concentrate on my research. I am very thankful to the Beechams (Bill, Fumi, Kevin, and Theresa) for their support from the beginning of my life in Salt Lake City.

Finally, I would like to thank my family, Aki and Reina. Aki has put up with me throughout my years in graduate school. Without her love and support, my work would not have been the same. Reina taught me how to stay up through the nights. Her smile made it possible to write this dissertation much faster.

CHAPTER 1

INTRODUCTION

Traditional molecular biology typically focuses on identifying individual genes and proteins and studying their specific functions in isolation. While this approach provides great details of components of biological systems, it has limitations in explaining how a biological system operates through interactions and networks of such components. Thus, it is difficult to predict with this approach, for example, how a cell would respond to perturbations in some genes or proteins.

Systems biology, on the other hand, examines the dynamics and interactions of biological components, and focuses on understanding of systems-level biological properties. A systems biology approach streamlines understanding of a biological system at the *systems level* by connecting the interdependent, *component-level* knowledge of the system such as the properties of genes and proteins, while such a systems-level understanding cannot be reductionistically deduced from the collection of components in the system alone. Thus, this approach can lead to better insights as to how living systems operate, and in turn, it can potentially have significant impacts on understanding how to control and engineer living systems to do useful things. However, living systems are very complex and understanding such complex systems is very expensive and is most likely impractical with *wet-lab* experiments alone, requiring the integration with computational approaches to reduce the number of experiments. Computational methods are used in systems biology, for example, to collect a large amount of data from a biological system, to infer computational models that can explain the data produced by the biological system, and to quantitatively analyze the temporal behavior of such computational models.

Computational modeling and simulation methodologies can facilitate building concepts of complex biological systems mathematically and validating them efficiently. Dynamics of a biological system represented in a computational model can be tested and validated against the corresponding empirical observations via simulation. Thus, tightly integrating computational approaches into the process of analyzing biological systems can help scientists further biological insights of such dynamical systems [67]. Figure 1.1 depicts the idealized workflow of this integrative systems biology approach. First, the initial knowledge and assumptions of a biological system are represented by various computational models. These models are then utilized to obtain computational results via simulation or other computational analysis methods. Only those models whose computational results are found to be consistent with the experimental facts are then further considered and utilized for more extensive system analysis. The successive experiments are those that eliminate the inadequate models. Thus, the successful models are assumed to be consistent with the current experimental knowledge. This process can be cycled again and again to explore and corroborate new biological hypotheses, gaining deeper understanding of biological systems [67, 68].

Thanks to advances in technologies, in genetic regulatory networks—where, for instance, high-throughput gene expression analysis methods are available and a vast amount of quantitative data has been collected—the information required for building quantitative models of genetic regulatory networks can be obtained, making the systems biology approach very promising.

1.1 Computational Modeling and Analysis

Numerous methods have been proposed for modeling genetic regulatory networks [64, 12]. Traditionally, biochemical systems are modeled and analyzed within the continuous-deterministic, *classical chemical kinetics* (CCK) framework based on the *law of mass action* where the dynamics of a well-stirred system are described by a set of ordinary differential equations (ODEs). However, the limitations of the CCK analysis have been broadly accepted [9, 40, 97, 102, 103]. In particular,

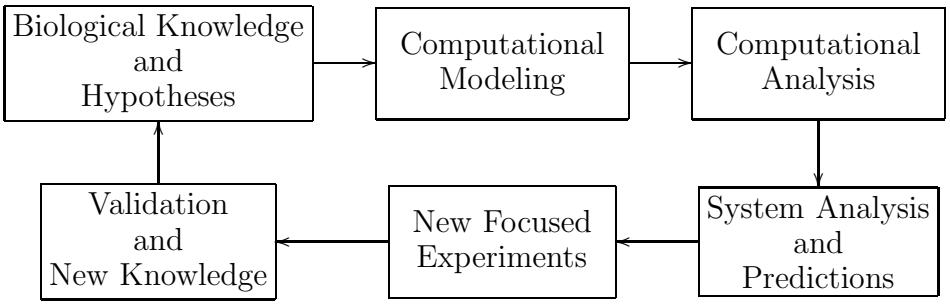


Figure 1.1. Idealized systems biology workflow. First, the initial knowledge and hypotheses of a biological system are conceptualized using various computational models. These models are then utilized to obtain computational temporal behavior results via simulation. Only those models whose computational results are found to be consistent with the experimental facts are then further considered for more extensive system analysis. The successful models are assumed to be consistent with the current experimental knowledge. In order to further test the possible models of the system and eliminate the inadequate models, new focused experiments are applied to the system to obtain more biological knowledge. This process can be cycled again and again to explore and corroborate new biological hypotheses, gaining deeper understanding of biological systems.

given the same initial condition, the CCK analysis of biochemical systems always produces the same results as it neglects fluctuations. Such treatment, nevertheless, can be justified when the molecular populations are very large, and hence a CCK analysis may provide the most efficient approach to determine the time evolution of a system in such cases. However, many regulatory components in biological systems can be present in amounts too small to simply neglect the effects of inherent fluctuations [80, 58, 92, 24, 86]. Moreover, if a system being analyzed has multiple steady states, the traditional ODE approach may not be able to provide an accurate time evolution of a system since it cannot capture spontaneous transitions between steady states [54, 50].

In order to more accurately predict the temporal behavior of biochemical systems without acquiring more information on a biological system such as the positions and the velocities of every molecule, the *stochastic chemical kinetics* (SCK) framework can be used [57]. SCK describes the time evolution of a well-stirred

biochemical system as a discrete-state jump Markov process that is analytically governed by the *chemical master equation* (CME) [55]. Assuming that the system is spatially homogeneous, this SCK approach describes the time evolution of a biochemical system at the individual reaction level by exactly tracking the quantities of each molecular species and by treating each reaction as a separate random event. However, directly obtaining the solution of the CME of any realistic system, either analytically or numerically, is not feasible due to its intrinsic complexity. Since the solution of CME is rarely available in realistic biochemical systems, the time evolution of moments is also generally infeasible to compute from the CME.

To overcome this, several methods have been introduced to approximate the time evolution of moments of the process without solving the CME [54, 6]. Such approximations are very useful to efficiently understand the mean behavior, standard deviation, skewness, etc., as well as to potentially characterize the time evolution of the asymptotic probability distribution of the system states. However, utilizing such methods alone may come across difficulties in quantitative analyses of some biologically relevant properties based on stochastic competition such as probabilistic analysis of lysis/lysogeny developmental pathways in bacteriophage λ-infected *Escherichia coli* [9]. Furthermore, since the complexity of the moment evolution equations may significantly increase as the size of the system increases [54], such approaches may be unwieldy for large-scale biological systems.

Instead of attempting to solve the CME, exact discrete-stochastic numerical realizations of a system dynamics via Gillespie's *stochastic simulation algorithm* (SSA) [52], which is derived from the same premise as the CME, are often used to infer the temporal system behavior with a much smaller memory footprint. This Monte Carlo simulation approach is useful to intuitively observe the trend of system dynamics, which may be possible with as few as tens of numerical realizations. Furthermore, *in silico* experiments via Monte Carlo simulation come with potentially unlimited controlling capabilities and abilities to capture virtually any dynamical properties of the system, making a number of qualitative and quantitative analyses which cannot be done in wet-lab experiments possible. Unfortunately, the compu-

tational requirements of the SSA can be substantial due largely to the fact that it not only requires a potentially large number of simulation runs in order to estimate the system behavior at a reasonable degree of statistical confidence, but it also requires every single reaction event to be simulated one at a time.

1.2 Abstraction

Ultimately, given the substantial computational requirements of stochastic simulations and comparative complexities of *in situ* genetic regulatory networks, abstraction is absolutely essential for efficient computational analysis. This abstraction can be achieved either during the simulation or the modeling stage.

Simulation abstraction approximates the exact SSA to accelerate the simulation process while the complexity of a model is left unchanged. This approach typically involves runtime identification of reaction events that can be skipped without significant effects on the system behavior, and the usage of an approximated simulation procedure that accelerates the simulation process by sacrificing the exactness. An example of this simulation abstraction is Gillespie's explicit τ-leaping method [56]. This method approximates the number of firings of each reaction in a predefined interval rather than executing each reaction individually. Although this simulation abstraction is very promising for many applications, the strict CME-level model requirements may not be suitable for a large, systems-level model as the underlying system complexity does not change with this approach.

Model abstraction transforms a low-level model to a higher-level model, making computational analysis more efficient and the complexity of the system lower. While the detailed reaction-level representations of biomolecular networks allow for very comprehensive descriptions of biological systems, such low-level models may lead to substantial computational costs and may obscure the understanding of the overall system structure and interdependency of the components. Thus, going to a higher-level representation and abstracting away dynamically insignificant reactions or species in order to reduce the complexity of the system can help make the overall systems biology analysis more efficient, as well as make crucial components

and interactions of a system more intuitive. This could be accomplished through a variety of techniques depending on the structure of the system and what the assumptions are. Although many model abstractions have long been in wide use individually, their traditionally manual transformation becomes increasingly more tedious and demanding as multiple methods are collectively applied to a particular biological system. The problem becomes even more acute as the size of the network increases, eventually rendering it intractable and potentially leading to significant errors in large model transformations.

1.3 Systematic Model Abstraction

To address these issues surrounding model abstraction of complex systems, this dissertation presents a generalized model abstraction methodology that systematically reduces the small-scale complexity found in biochemical systems represented by *REaction-Based* (REB) models (i.e., models composed of a set of chemical reactions) while broadly preserving the large-scale system behavior. Thus, this approach alleviates the abstraction problems by systematically testing network patterns and characteristics to determine which abstraction methods are applicable [73, 74]. Furthermore, this approach allows one to scan through the effective levels of abstraction and to optimize model transformation for *efficiency-versus-accuracy* by first adjusting the various precision criteria in individual abstraction methods and then performing transformation accordingly.

Our methodology—outlined in Figure 1.2—begins with a REB model, which could be simulated via the SSA or one of its variants though at a substantial computational cost. To reduce the cost of computational analysis, the original REB model is simplified by applying abstraction methods that mainly attempt to reduce the number of reactions and species based on the structure of the model and the abstraction criteria. The result is an abstracted REB model with fewer reactions and species, which substantially lowers the cost of stochastic simulation. To further reduce the complexity of the system as well as analysis time, this abstracted REB model can be automatically translated into a *finite state system* (FSS) model by

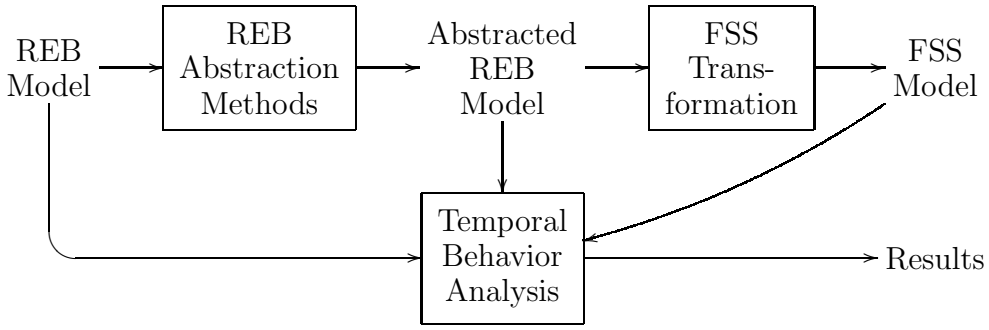


Figure 1.2. Automated model abstraction tool flow.

representing the dynamics of the system states (i.e., molecular population levels in the system) by a finite state graph. This model can then be efficiently analyzed, for example, using a Markov chain analysis method.

Whereas we believe that our model abstraction methodology can, in theory, be applied to any biochemical networks, this dissertation concentrates on genetic regulatory networks as a proof of concept to evaluate our methodology, which includes abstraction methods tailored to reduce the computation time of temporal behavior analysis of such networks. This dissertation therefore exemplifies the applicability of our model abstraction methodology to genetic regulatory networks as case studies.

1.4 Contributions

The major contributions of this dissertation are the development of a systematic model abstraction methodology for genetic regulatory networks and the development of an automated modeling and temporal behavior analysis tool for biochemical systems. The modeling and analysis tool, which we call REB2SAC, implements the systematic model abstraction methodology along with a variety of efficient computational analysis methods that can take advantage of the simplified model [73, 74]. Thus, REB2SAC streamlines the computational analysis of biochemical systems and, in turn, the process of systems biology research. Furthermore, REB2SAC takes as an input a REB model formated in the *systems biology markup language* (SBML), an emerging standard computer-readable format for representing

models of biochemical reaction networks [42]. Thus, genetic regulatory network models which are built using SBML-compliant modeling tools such as BioSPICE's PathwayBuilder [16] and CellDesginer [44] can be easily applied to construct models that the tool can analyze. Also, REB2SAC has been integrated into a graphical user interface tool called BioSim [2], bringing more user friendliness which is crucial for biologists to concentrate on biology problems without dealing with computer science problems.

In addition, contributions are made by analyzing real biological systems using REB2SAC. Doing so allows us to not only evaluate the model abstraction methodology in terms of accuracy and speedup by comparing original, detailed models with corresponding abstracted models but also to provide the computational biology community with genetic regulatory network models in SBML format, together with analysis results.

1.5 Dissertation Outline

This dissertation is organized as follows. First, it presents the background information on modeling and analysis of genetic regulatory networks. Chapter 2 overviews genetic regulatory networks. It introduces several key features of genetic regulatory networks that are acknowledged in later chapters. In Chapter 3, an overview of chemical kinetics is described. The CCK model and the SCK model are first discussed, and then the derivation of the SSA along with several approximation methods of the discrete-stochastic simulations are presented in this chapter.

Next, this dissertation presents our model abstraction methods. The REB model abstraction methods are described in Chapter 4. This chapter first formally defines the REB model, and then presents derivations as well as algorithms of several major REB model abstraction methods. This dissertation then describes the FSS model transformation in Chapter 5. This chapter first formally defines the FSS model, and then describes several temporal behavior analysis methods using the FSS model. Furthermore, it presents two abstraction methods to transform a REB model into a FSS model.

Case studies for our methodology are presented next. Chapter 6 illustrates the applications of individual REB model abstraction methods, and presents the simulation results. Chapter 7 presents more comprehensive analysis of our methodology by collectively applying the abstraction methods to produce various levels of abstracted models of a couple of genetic regulatory networks. Biologically relevant properties of such genetic regulatory networks are computationally obtained from each resolution of the models, and the results are compared with those from the experimental methods.

Finally, this dissertation concludes in Chapter 8 by summarizing the work and presenting the potential future work.

CHAPTER 2

GENETIC REGULATORY NETWORKS

Cells are the fundamental building blocks of all living organisms, providing structures and specialized functions. Genetic regulatory networks control the levels of gene expression and protein synthesis via interactions of DNA, RNA, and proteins, as well as other constitutive molecules, and play crucial roles in development and maintenance of cells' actions and properties. Thus, understanding of the structures as well as the dynamics of genetic regulatory networks can lead to significant insights as to how a living system operates under various conditions. This chapter is intended to provide a brief overview of genetic regulatory networks. Section 2.1 overviews the flow of genetic information, explaining the basic elements and the steps required to express or turn on genes. Section 2.2 describes the mechanisms of gene regulation. Section 2.3 then overviews analysis of genetic regulatory networks generated using, among other things, a vast amount of gene expression data that have been generated with high throughput data collection methods.

2.1 The Flow of Genetic Information

Proteins carry out most of the work in cells and are required for the structures and functions of cells. Thus, direct actions and properties of a cell are mainly determined by the proteins it contains. Proteins are differentiated according to their large range of functions in a cell. For example, some proteins called *enzymes* accelerate chemical reactions, while some other proteins provide structure and support for cells. However, one thing proteins generally cannot do is to reproduce themselves. When a cell needs more proteins to maintain or change its phenotype, it uses the hereditary material known as *deoxyribonucleic acid* (DNA).

Most of the instructions needed to direct synthesis of proteins is encoded in DNA. DNA is the blueprint of life for all living organisms where nearly every cell in a living system has the same information. DNA is a nucleic acid molecule that encodes genetic information in the sequences of four chemical units (bases): *adenine* (A), *guanine* (G), *cytosine* (C), and *thymine* (T). DNA is usually double stranded whereby the sequences of base pairs are arranged so that A is always paired with T and G is always paired with C. DNA contains a number of segments called *genes* that encode instructions to produce single-stranded nucleic acid molecules called *ribonucleic acids* (RNAs). The type of RNA that encodes the instructions to make a protein is called *messenger RNA* (mRNA).

The process of the production of RNA and proteins from DNA is known as *gene expression*. The framework for the flow of genetic information is specified by the *central dogma of molecular biology* [34]. In particular, the flow of genetic information is from DNA to mRNA, and then from mRNA to proteins. These two steps in gene expression are known as *transcription* and *translation*. In prokaryotic cells in which the genetic material is in the *cytoplasm*, both transcription and translation take place in the cytoplasm. In the case of an *eukaryotic* cell in which the genetic information is contained inside a membrane-bound nucleus, transcription takes place inside the nucleus while translation takes place in the *cytoplasm*.

In transcription, one of two DNA strands is used as a template to create a complementary strand of RNA. DNA is unwound and genes along one strand of DNA are transcribed into RNA molecules, which is directed by an enzyme called *RNA polymerase* (RNAP). The process of transcription starts with the binding of RNAP to a region of DNA located near the beginning of a gene called a *promoter* as shown in Figure 2.1. RNAP binds and unbinds to promoters rapidly where the RNAP binding rate is mainly determined by the promoter affinity strength. When RNAP binds to a promoter site, the DNA is still double-stranded ("closed"). Thus, this structure of RNAP and wound DNA is known as a *closed complex*. RNAP then unwinds double-stranded DNA to make genetic information of a gene available to transcribe. This complex of RNAP and unwound DNA is known as an *open complex*.

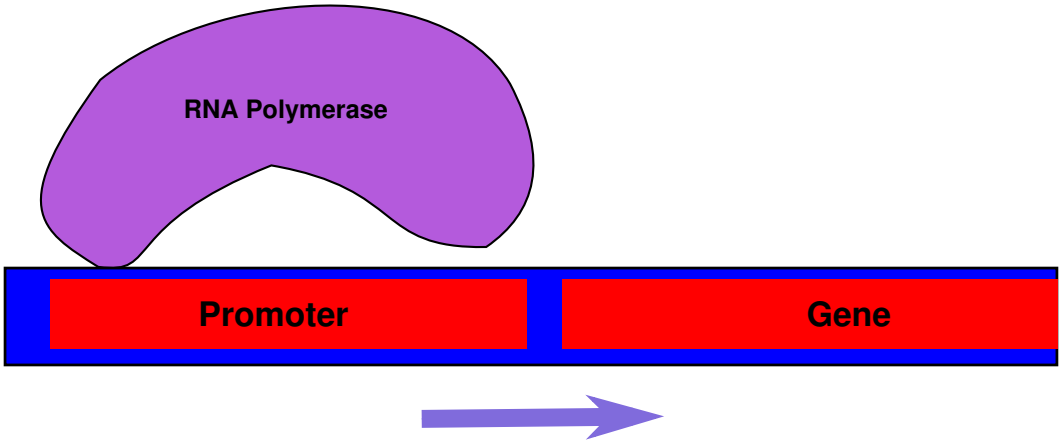


Figure 2.1. Transcription initiation. RNAP binds to a promoter to initiate transcription of the downstream gene.

By forming an open complex, RNAP becomes ready to read information encoded on one side of the DNA to generate a complementary strand of RNA. Following this initiation, as depicted in Figure 2.2, RNAP travels along the gene and synthesizes RNA as it moves until it sees the stop signal encoded in a segment of DNA called a *terminator*. At the terminator, RNAP falls off from the DNA and releases the RNA.

Following transcription, mRNA then moves to interact with a protein-RNA molecular complex called a *ribosome*, which produces proteins using *amino acids* delivered by *transport RNA* (tRNA) in the cytoplasm. Each sequence of three bases in mRNA, called a *codon*, codes for a specific amino acid, and a chain of amino acids is assembled to produce a protein.

2.2 Regulation of Gene Expression

With recent advances in technology and sciences, whole *genome* (all of an organism's genetic material) sequences of many organisms have been revealed. For example, the completion of the Human Genome Project has determined the sequences of 3 billion base pairs and identified around 25,000 genes in the human genome [13, 111]. While all the cells in the human body have largely the same DNA, their structures and functionalities can be diverse. For example, the properties of

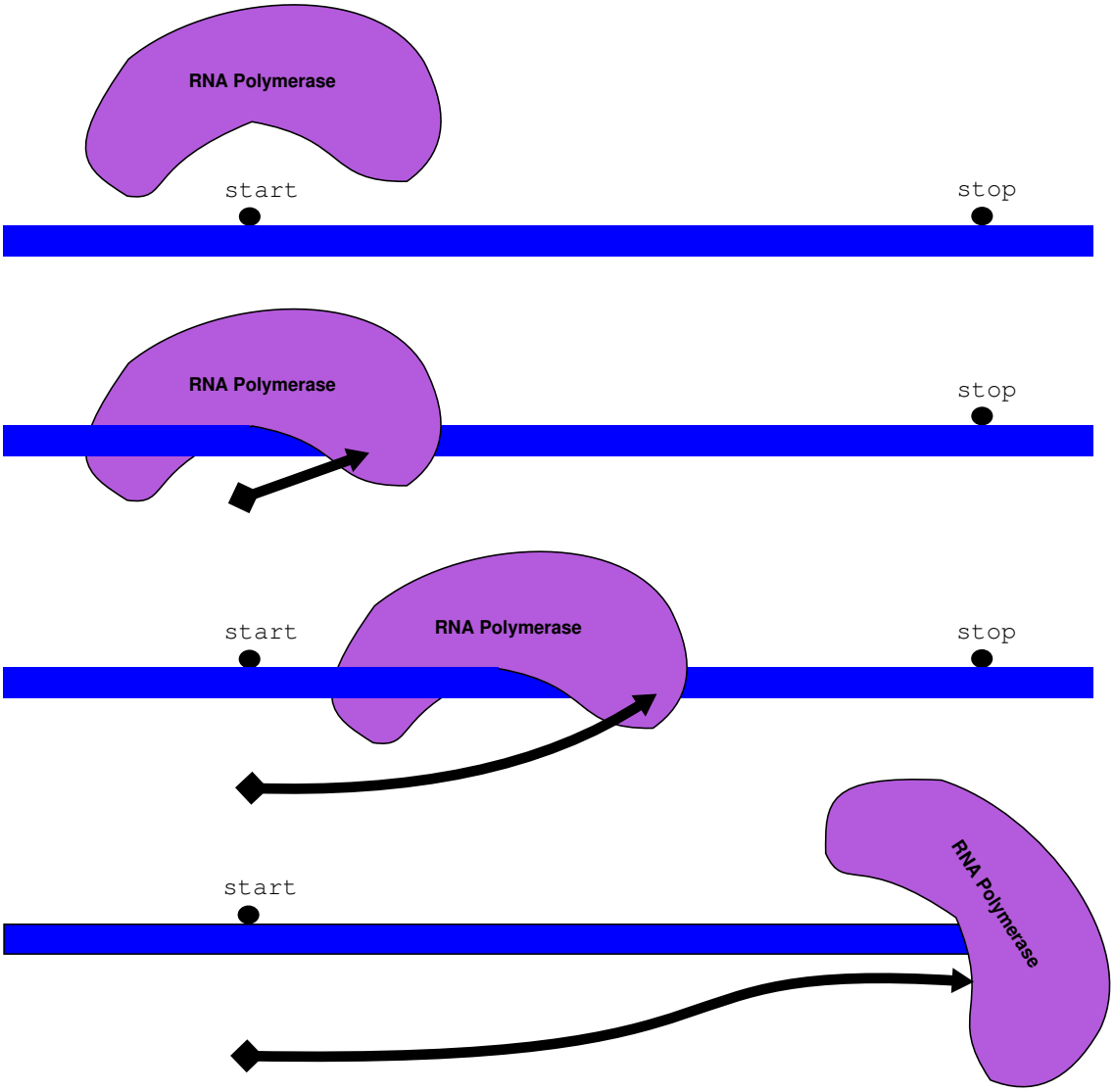


Figure 2.2. Synthesis of mRNA. RNAP travels along the gene and synthesizes RNA as it moves until it sees the terminator sequence, and RNAP falls off the DNA and releases the RNA at the terminator.

a skin cell are very different from those of a brain cell. This is because a skin cell produces only the proteins appropriate for that cell by precisely expressing a subset of the 25,000 genes that is different from a subset of genes expressed in a brain cell. However, considering about 20,000 genes that even a simple organism like *Caenorhabditis elegans* (roundworm) has [100] and the number of genes in the human genome, the number of genes alone does not determine the structural

and functional complexities of organisms. It has been suggested that the key to higher morphological and behavioral complexity is greater elaborate regulation of gene expression [76]. It has been experimentally shown that the process of gene expression can be highly nondeterministic in living cells [40]. Although such fluctuations limit the precision of gene expression, they can play a critical role in cell-to-cell variation where some species exploit the variations via nondeterministic process of gene expression when they are advantageous for their survival [81, 9].

Although gene expression can be regulated at each step of transcription and translation, the heart of regulation to, for example, adapt phenotypes in response to environmental stimuli comes from the transcription initiation where *transcription factors* and *cis-regulatory* DNA elements control when and how genes are transcribed and in turn proteins are synthesized [37, 77, 31, 36, 63]. Transcription factors are largely regulatory proteins that can control the rate of transcription by occupying *cis*-regulatory elements on DNA. Negative transcription factors called *repressors* prevent transcription of genes upon binding to the corresponding *cis*-regulatory elements, while positive transcription factors called *activators* enhance transcription of genes. *Cis*-regulatory elements on DNA include promoters and operators. Operators are regulatory sequences of DNA that are usually located near the corresponding promoters of genes to which transcription factors can bind to repress or activate transcription. These critical transcription regulatory components can be present in very low counts in a cell [60], contributing to the nondeterministic effects in gene expression [80, 40, 103].

Figure 2.3 shows a relatively simple two-gene system to illustrate the mechanism of genetic regulatory networks. In this network, a piece of DNA contains two genes: a and b. Suppose proteins A and B, the products from genes a and b, are not present in the system, and the promoter for gene a has a higher affinity to RNAP binding than the promoter for gene b at the basal rate. Transcription of gene a is then initiated much more frequently than that of gene b, causing gene a to be expressed and protein A to be synthesized more often at this configuration. Protein A is an activator of transcription of gene b and it can occupy the operator site of gene b

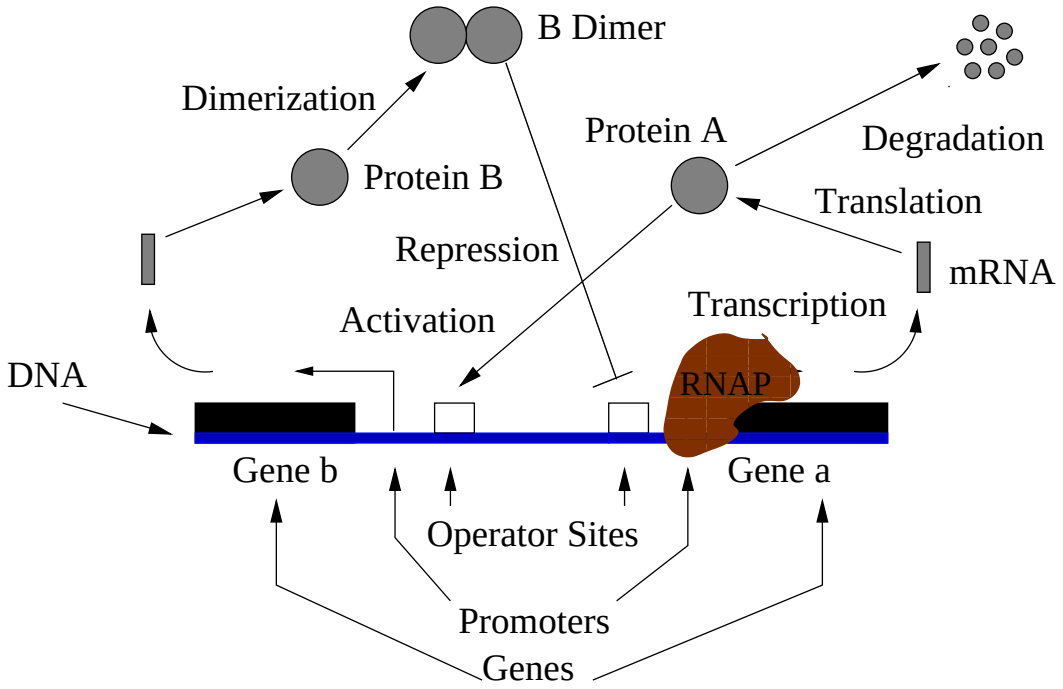


Figure 2.3. Two-gene system to illustrate the mechanism of gene expression regulation. Protein A activates expression of gene b, while protein B represses expression of gene a.

to increase the expression rate of gene b. Thus, protein B can be synthesized at a higher level. Two copies of protein B can *dimerize* and form a B_2 molecule, and this B *dimer* can act as a repressor of transcription of gene a by occupying the operator site of gene a. Consequently, at high levels of protein B, protein A is rarely produced, and with degradation, the level of protien A becomes so low that protein A can no longer effectively occupy the operator site of gene b to activate the expression of gene b. Thus, with limited production at its basal rate, protein B degrades, allowing gene a to be expressed once again.

2.3 Network Analysis

Advances in technologies such as DNA sequencing and gene expression profiling methods [79, 104] contribute to increase throughput of generation of data required for systematic approaches to understand genetic regulatory networks. A vast amount of data collected by such technologies can be used to catalog a list of

components of a genetic regulatory network and infer the structure of that network. However, elucidating genetic regulatory networks only via wet-lab experiments can be a daunting task, and complexity of this task substantially increases as the amount of collected data increases and as the network being analyzed becomes more complex. For example, just to find out the causal relationship between genes a and b of the genetic regulatory network in Figure 2.3 may require many experiments.

In order to infer that protein A activates the expression of gene b, gene a can be mutated or knocked out [113] and the expression level of gene a can be compared between the wild-type cell and gene-a-mutated one. If the expression level of gene b is higher in the wild-type cell, then it can be concluded that the participation of protein A increases the expression of gene b, which, in turn, allows one to infer that protein A directly activates the expression of gene b. However, other hypotheses can be made to explain the causality. For example, one hypothesis would be that protein A activates the expression of another gene, say gene c, and protein C activates the expression of gene b. Another alternative hypothesis would be that protein D binds to the operator site of gene b to repress the transcription and A binds to protein D at a high affinity rate to prevent protein D from binding to the operator site of gene b, causing an increase in expression of gene b by the presence of protein A. To test these hypotheses, for example, genes c and d can be knocked out, and expression of gene b can be profiled for each hypothesis. However, testing and (in)validating every single hypothesis via wet-lab experiments in a brute-force fashion is extremely time consuming and expensive. This is especially true for large-scale or systems-level biological networks where it is impossible to generate and test all possible hypotheses manually in the wet-lab.

Computational modeling and analysis can be applied to generate and screen hypotheses, which can stimulate the development of new experiments and effectively reduce the number of experiments to test hypotheses [67, 33]. The first step of such a computational systems biology approach to understanding of genetic regulatory networks is to construct computational models encapsulating hypotheses and explaining experimental facts such as gene expression data. This can be

done using various machine learning and data mining techniques [14, 43, 117]. Once quantitative computational models are constructed, they can be utilized to analyze the temporal behavior via simulation, allowing the hypothesis and assumptions encapusulated in each model to be analyzed and screened. Furthermore, a computational modeling and analysis approach comes with potentially unlimited controlling capabilities and abilities to capture virtually any dynamical properties of the system, making possible a number of qualitative and quantitative analyses which cannot be done in wet-lab experiments. Since quantitative data required to support systematic construction of such computational models are now becoming available via high-throughput molecular biology methods, this computational approach is now becoming possible and is able to provide useful biological insights [20, 38, 9, 116]. Furthermore, it can be used, for example, to apply an engineering approach to more efficiently and effectively analyze how a genetic regulatory network can be controlled and designed to achieve specific functions [21, 8]. Therefore, computational modeling and analysis can revolutionalize the way biological systems are analyzed and contribute to further understanding of such systems.

CHAPTER 3

CHEMICAL KINETICS

In order to describe the time evolution of a biochemical system quantitatively, the use of a mathematical model is essential. This chapter presents an overview of *chemical kinetics* models. Section 3.1 describes the chemical reactions that are the fundamental processes to transition a biochemical system's states. Section 3.2 presents the traditional classical chemical kinetics (CCK) model to describe the dynamics of a well-stirred biochemical system continuously and deterministically. Section 3.3 describes the stochastic chemical kinetics (SCK) approach, which describes the time evolution of a well-stirred biochemical system as a discrete-stochastic process. Finally, Section 3.4 presents various approximated simulation methods to facilitate efficient analysis of SCK models.

3.1 Chemical Reactions

A chemical reaction is a process in which a subset of species in a system is chemically changed into another subset of species in the system. For example, the chemical reaction:

$$2s_1 + s_2 \rightarrow s_3 \tag{3.1}$$

consumes two molecules of s_1 and one molecule of s_2 and produces one molecule of s_3. In chemical reactions, species that are consumed (i.e., species that are shown on the left side of the arrow) are called *reactants*, while species that are produced (i.e., species on the right side of the arrow) are called *products*. Thus, in Reaction 3.1, species s_1 and s_2 are the reactants and species s_3 is the product. A number that appears in front of a species such as 2 in species s_1 in Reaction 3.1 is the *stoichiometry*, which represents the quantitative relationships between the reactants

and products in chemical reactions. If a species is not preceeded by a number in a chemical reaction such as s_2 and s_3 in Reaction 3.1, then the stoichiometry of that species is implicitly understood as 1 in that reaction.

Chemical reactions can be either *elementary reactions* which cannot be broken down into smaller reaction steps or a collection of elementary reactions. Since Reaction 3.1 is a *trimolecular reaction* which involves three molecules as the reactants and requires a reactive collision of three molecules, it is highly unlikely, if not impossible, to occur in one step physically, and thus it is not strictly an elementary reaction. However, such reactions are sometimes approximated as elementary reactions.

Unlike a trimolecular reaction, a *bimolecular reaction*, which occurs as a consequence of a reactive collision of two molecules, is an elementary reaction. For example, a chemical reaction of the form:

$$2s_1 \rightleftharpoons s_2 \tag{3.2}$$

is a bimolecular reaction which takes two molecules of s_1 and produces one molecule of s_2, and vice versa. Bimolecular reactions such as Reaction 3.2 are known as *dimerization reactions* since two molecules of the same species produce a dimer form. Reaction 3.2 has a double arrow indicating that it is a *reversible* reaction in that both the forward and backward reactions are possible. Essentially, in most chemical reactions, reversible reactions are the norm and irreversible reactions can be viewed as special cases of reversible reactions in which the forward directional reaction dominates that of the reverse direction.

Unimolecular reactions are also elementary reactions. A unimolecular reaction converts one molecule of a species into one or more molecules of (an)other species. For example, a reaction of the form:

$$s_1 \rightarrow s_2 \tag{3.3}$$

is a unimolecular reaction which converts a single molecule of species s_1 into a single molecule of species s_2.

In general, by defining v_{ij}^r as:

$$v_{ij}^r \equiv \text{ the stoichiometry of species } s_i \text{ as a reactant in reaction } r_j, \tag{3.4}$$

and v_{ij}^p as:

$$v_{ij}^p \equiv \text{ the stoichiometry of species } s_i \text{ as a product in reaction } r_j, \tag{3.5}$$

reaction r_j in a chemically reacting system of N species $\{s_1, \ldots s_N\}$ has the form:

$$v_{1j}^r s_1 + v_{2j}^r s_2 + \cdots + v_{Nj}^r s_N \rightleftharpoons v_{1j}^p s_1 + v_{2j}^p s_2 + \cdots + v_{Nj}^p s_N. \tag{3.6}$$

If species s_i does not participate in reaction r_j as a reactant, then v_{1j}^r is set to 0. Similarly, if s_i is not produced by reaction r_j, then v_{1j}^p is set to 0.

3.2 Classical Chemical Kinetics

In order to quantitatively analyze the dynamics of a chemically reacting system, each state transition must be quantitatively specified, and thus, chemical kinetics must be specified. CCK considers each reaction as an event that proceeds continuously in a well-stirred system where a large number of molecules are involved, and, in turn, the reaction rate is viewed as the speed of changes in the states of the participating species per unit time. Traditionally, the time evolution of a well-stirred biochemical system is described by a set of coupled, ordinary differential equations (ODEs) which are based on CCK. This approach can thus take advantage of the well-established theory in numerical solutions of the initial value problem of ODEs to analyze the dynamics of biochemical systems [94, 66].

3.2.1 The Law of Mass Action

The foundation of CCK is the *law of mass action*. The law of mass action states that the time-rate (i.e., the speed) of a chemical reaction is proportional to the product of the *concentrations* (i.e., number of molecules divided by the volume) of the reactant molecules in that reaction. In other words, according to the law of mass action, there exists a proportionality constant such that the product of

such a constant and the concentrations of the reactant molecules form the reaction rate. Note that even though it is called a "law," the law of mass action is a useful model, rather than a law, that can provide a good approximation for a well-stirred system, the dynamics of which can be relatively well-estimated without knowing the position and the velocity of each molecule in the system [65, 41]. Also, the law of mass action is only applicable to elementary reactions. Thus, a very detailed-level system description is required to construct mass action kinetics models.

With the law of mass action, Reaction 3.1 can identify its reaction rate V to be

$$V = k_1 {x_1}^2 x_2 \tag{3.7}$$

where x_i is the concentration of species s_i and the constant k_1 is called a *rate constant*. To show that a reaction is treated as an elementary reaction and the mass action kinetics is applied to it, the corresponding rate constant is usually given with the directional arrows. For example, Reaction 3.1 with mass action kinetics is shown as:

$$2s_1 + s_2 \xrightarrow{k_1} s_3. \tag{3.8}$$

Reaction rates of CCK have the units of concentration per unit time. Thus, the rate constants of trimolecular reactions—such as Reaction 3.8—have the unit (concentration^2 $\times$ unit time)$^{-1}$, while the rate constants of bimolecular reactions have the unit (concentration $\times$ unit time)$^{-1}$. In general, the rate constant of a reaction with n molecules of reactants has the unit (concentration^{n-1} $\times$ unit time)$^{-1}$.

Reversible reactions have two rate constants; one is for the forward rate constant and the other is for the backward rate constant. For example, Reaction 3.2 with the forward rate constant k_2 and the backward rate constant k_{-2} is depicted by

$$2s_1 \underset{k_{-2}}{\overset{k_2}{\rightleftharpoons}} s_2. \tag{3.9}$$

The reaction rate V of Reaction 3.9 is then derived via the law of mass action to be:

$$V = k_2 {x_1}^2 - k_{-2} x_2. \tag{3.10}$$

In general, suppose reaction r_j has the form of Reaction 3.6 and reaction r_j is an elementary reaction with k_+ as the forward rate constant and k_- as the backward rate constant. Then, using the law of mass action, the reaction rate V_j of reaction r_j is described by:

$$V_j = k_+ \prod_{i=1}^{N} {x_i}^{v_{ij}^r} - k_- \prod_{i=1}^{N} {x_i}^{v_{ij}^p} \tag{3.11}$$

where x_i is the concentration of species s_i. If reaction r_j is an irreversible reaction, then k_- is set to 0, making the second term on the right hand side in Equation 3.11 vanish.

3.2.2 Ordinary Differential Equation Model

Suppose Reactions 3.8 and 3.9 are the only reactions that occur within a chemically reacting system. Then, since two molecules of species s_1 are consumed by Reaction 3.8, and two molecules of species s_1 are consumed by the forward reaction of Reaction 3.9 and two molecules of species s_1 are produced by the backward reaction of Reaction 3.9, the change of the concentration of s_1 in this system over unit time is specified as:

$$\frac{dx_1}{dt} = -2k_1{x_1}^2x_2 - 2k_2{x_1}^2 + 2k_{-2}x_2. \tag{3.12}$$

Similarly, one molecule of species s_2 is consumed by Reaction 3.8, and one molecule of species s_2 is both produced and consumed by Reaction 3.9. Thus, the differential equation for the change of the concentration of species s_2 with respect to time is

$$\frac{dx_2}{dt} = -k_1{x_1}^2x_2 - k_2{x_1}^2 + k_{-2}x_2. \tag{3.13}$$

Since species s_2 only participates in Reaction 3.8 as a reactant, and one molecule of species s_2 is consumed by it, the differential equation of the concentration of species s_3 becomes

$$\frac{dx_3}{dt} = k_1{x_1}^2x_2. \tag{3.14}$$

Therefore, the ODE model that describes the dynamics of the system of species s_1, s_2, s_3 with Reactions 3.8 and 3.9 consists of the set of Equations 3.12, 3.13, and 3.14.

In general, suppose N chemical species $\{s_1, \ldots, s_N\}$ in a chemically reacting system interact via M reaction channels $\{r_1, \ldots, r_M\}$ where each r_j has the form of Reaction 3.6. Then, by defining v_{ij} as:

$$v_{ij} \equiv v_{ij}^p - v_{ij}^r \tag{3.15}$$

where v_{ij}^r and v_{ij}^p are from Equations 3.4 and 3.5, respectively, the change of the state of species s_i by reaction r_j per unit time can be quantified by $v_{ij}V_j$ where V_j is the reaction rate of reaction r_j. Therefore, by denoting x_i to be the concentration of species s_i, the ODE model of this system can be specified as:

$$\frac{dx_i}{dt} = \sum_{j=1}^{M} v_{ij}V_j, \quad 1 \leq i \leq N. \tag{3.16}$$

3.2.3 Limitations of CCK

Although biochemical systems have traditionally been analyzed via CCK, there are several limitations in this approach that may be critical to some biochemical systems such as genetic regulatory networks. The implication of CCK is a continuous-deterministic process description of the time evolution of a biochemical system. Representing each concentration of species in continuum states can be justified when the molecular populations of the species are very large. In such cases, the relative change of molecular populations by each reaction is so small that it can be viewed as a continuous change. Also, when this large-molecular-population assumption holds in a biochemical system, the relative fluctuation of each molecular population—which can be empirically estimated roughly as the inverse square root of its mean population—becomes so small that it can be safely neglected. Thus, the dynamics of the system can be described by a deterministic process and CCK analysis may provide the most efficient approach to determine the average time evolution of a system under such conditions.

However, if a biochemical system includes some species with small molecular counts, then the fundamental assumption of CCK is violated and the computational analyses may not be able to capture the true dynamic behavior of such systems

[109, 83, 52]. Furthermore, if a biochemical system with a highly nonlinear behavior has multiple steady states such as the bistable *Schlögl reactions* [105], the CCK approach may not be able to provide an accurate system description since it cannot capture spontaneous transitions between steady states [54, 114, 50, 47].

3.3 Stochastic Chemical Kinetics

Unlike CCK, SCK describes the time evolution of a chemically reacting system by a discrete-stochastic process. SCK considers N chemical species $\{s_1, \ldots, s_N\}$ which interact through M *irreversible*, elementary-reactions $\{r_1, \ldots, r_M\}$ inside a well-stirred, chemically reacting system with a constant volume Ω in thermal equilibrium at some constant temperature. With these assumptions, it describes the time evolution of the system within the discrete-stochastic framework in continuous time. This section presents an overview of SCK, which is based on the work of Gillespie [52, 53, 55, 57].

3.3.1 System State

The most exact way to simulate the dynamics of a molecular system is *molecular dynamics* where movements of every molecule in the system are tracked [57]. The system state of molecular dynamics is the positions and velocities of every molecule in the system where the dynamics of the system state are described by capturing every movement and every collision of molecules in the system. While this approach can show the time evolution of species' populations as well as the spatial distribution of each species, acquiring such detailed knowledge and performing such computationally expensive simulations may be infeasible.

By making the well-stirred assumption, SCK as well as CCK can greatly simplify the complexity of models. With this assumption, the spatial property of a system is ignored and the system state of SCK is defined to be simply the populations of species in the system. Thus, by denoting $\mathbf{X}(t) \equiv (X_1(t), \ldots, X_N(t))$, the system state vector that represents the number of molecules of each s_i, the evolution of

$\mathbf{X}(t)$, given that $\mathbf{X}(t_0) = \mathbf{x_0}$ (for $t \geq t_0$) can be probabilistically described rigorously for each reaction event.

3.3.2 Propensity Function

In SCK, each reaction r_j is viewed as a discrete random event that changes the system state by $\mathbf{v_j} \equiv (v_{1j}, \ldots, v_{Nj})$, called the *state change vector*, whose i^{th} element v_{ij} is defined in Equation 3.15. Thus, given the system is in state $\mathbf{x} \equiv (x_1, \ldots x_N)$ where x_i is the molecular population of species s_i, the system jumps to state $\mathbf{x} + \mathbf{v_j}$ as a consequence of a single r_j reaction event. The time that the next event of reaction r_j occurs is governed by function a_j, which is called the *propensity function* of reaction r_j, and it is defined as follows:

$$\begin{aligned} a_j(\mathbf{x})dt \equiv &\text{ the probability that, given } \mathbf{X}(t) = \mathbf{x}, \text{ reaction } r_j \text{ will occur} \\ &\text{inside } \Omega \text{ in the next infinitesimal time interval } [t, t + dt) \end{aligned} \tag{3.17}$$

where the infinitesimal time dt is taken to be so small that at most one reaction event occurs within the interval. Thus, the strict requirement of SCK is that each reaction r_j shall be an elementary reaction. Since the propensity function is the basis of the discrete-stochastic process of SCK, it may be said to be the *fundamental premise of stochastic chemical kinetics*. The propensity function of each reaction r_j is quantified by first defining a *specific probability rate constant* c_j such that:

$$\begin{aligned} c_j dt \equiv &\text{ the probability that a randomly chosen combination of} \\ &\text{reactant molecules of } r_j \text{ inside } \Omega \text{ at time } t \text{ will transform} \\ &\text{via } r_j \text{ within the next infinitesimal time } dt. \end{aligned} \tag{3.18}$$

Suppose reaction r_j is a unimolecular reaction and in the form of Reaction 3.3. Then, from Definition 3.18, the propensity function is defined as $a_j(\mathbf{x}) = c_j x_1$. The value of c_j for a unimolecular reaction r_j turns out to be the same as the reaction rate constant from CCK.

Suppose reaction r_j is a bimolecular reaction of the form $s_1 + s_2 \xrightarrow{c_j} \cdots$. Then, by assuming that the system is well-stirred or spatial-homogeneous, the propensity function takes the form of $c_j x_1 x_2$. In this case, c_j is numerically the same as k_j/Ω, and thus, the propensity function equals the reaction rates via CCK.

If, however, reaction r_j is a dimerization reaction of the form $2s_1 \xrightarrow{c_j} \cdots$, the propensity function becomes $a_j(\mathbf{x}) = \frac{c_j}{2!}x_1(x_1-1)$ and the relationship between the specific probability rate constant and the classical reaction rate constant becomes $c_j = 2k_j/\Omega$. Also, if reaction r_j is a trimerization reaction $3s_1 \xrightarrow{c_j} \cdots$ and if it is assumed to be an elementary reaction, then the propensity function of reaction r_j becomes $a_j(\mathbf{x}) = \frac{c_j}{3!}x_1(x_1-1)(x_1-2)$ where $c_j = 3!k_j/\Omega^2$. In general, the propensity function for an n-merization reaction: $ns_1 \xrightarrow{c_j} \cdots$ becomes $a_j(\mathbf{x}) = \frac{c_j}{n!}\frac{x_1!}{(x_1-n)!}$, and the relationship between k_j and c_j becomes $c_j = n!k_j/\Omega^{n-1}$. Thus, in these n-merization reactions, while the reaction rates via CCK are not equal to the corresponding propensity functions, they approximate the propensity functions very well. This is especially true when the molecular counts are relatively high. This type of approximation is commonly applied to propensity functions, allowing biochemical system models to be numerically analyzed via both CCK and SCK approaches conveniently.

3.3.3 Chemical Master Equation

By assuming that a system is well-stirred—through either external force or many *non-reactive* collisions in the system—after each reaction event, the current molecular population is the only information that matters in the propensity functions. This means that a system transition depends on the current state of the system and does not depend on the history of the system. In other words, $\mathbf{X}(t)$ is a Markov process. Moreover, since each propensity function does not explicitly depend on time—by assuming Ω and the system temperature are constant throughout the system evolution—the stochastic process $\mathbf{X}(t)$ is said to be *temporally homogeneous*. The implication is that the time evolution of $\mathbf{X}(t)$ can be described by a temporally homogeneous jump Markov process.

To analytically describe the very Markov process defined by SCK, let $P(\mathbf{x}, t \mid \mathbf{x_0}, t_0)$ be the probability such that:

$$P(\mathbf{x}, t \mid \mathbf{x_0}, t_0) \equiv \text{the probability that } \mathbf{X}(t) = \mathbf{x} \text{ given } \mathbf{X}(t_0) = \mathbf{x_0}. \tag{3.19}$$

Then, the evolution of the probability $P(\mathbf{x}, t \mid \mathbf{x_0}, t_0)$ characterizes the temporal behavior of a biochemical system via SCK.

To express how $P(\mathbf{x}, t \mid \mathbf{x_0}, t_0)$ evolves, two cases are considered as depicted in Figure 3.1. If $\mathbf{X}(t) = \mathbf{x}$, then, as shown in Figure 3.1(a), the probability of the state transitioning to state $\mathbf{x} + \mathbf{v_i}$ in the next infinitesimal time dt is $a_j(\mathbf{x})dt$ for all $j \in [1, M]$. In other words, the probability that, given $\mathbf{X}(t) = \mathbf{x}$, $\mathbf{X}(t + dt) = \mathbf{x}$ is $1 - \sum_{j=1}^{M} a_j(\mathbf{x})dt$. On the other hand, if $\mathbf{X}(t) = \mathbf{x} - \mathbf{v_i}$, then the probability of the system transitioning to state $\mathbf{x}$ at time $t + dt$ is $a_j(\mathbf{x} - \mathbf{v_i})dt$ as shown in Figure 3.1(b). Therefore, the conditioned probability $P(\mathbf{x}, t + dt \mid \mathbf{x_0}, t_0)$ can be expressed as:

$$\begin{aligned} P(\mathbf{x}, t + dt \mid \mathbf{x_0}, t_0) = P(\mathbf{x}, t \mid \mathbf{x_0}, t_0) \left[1 - \sum_{j=1}^{M} a_j(\mathbf{x})dt\right] \\ + \sum_{j=1}^{M} [P(\mathbf{x} - \mathbf{v_j}, t \mid \mathbf{x_0}, t_0) a_j(\mathbf{x} - \mathbf{v_j})dt]. \end{aligned} \tag{3.20}$$

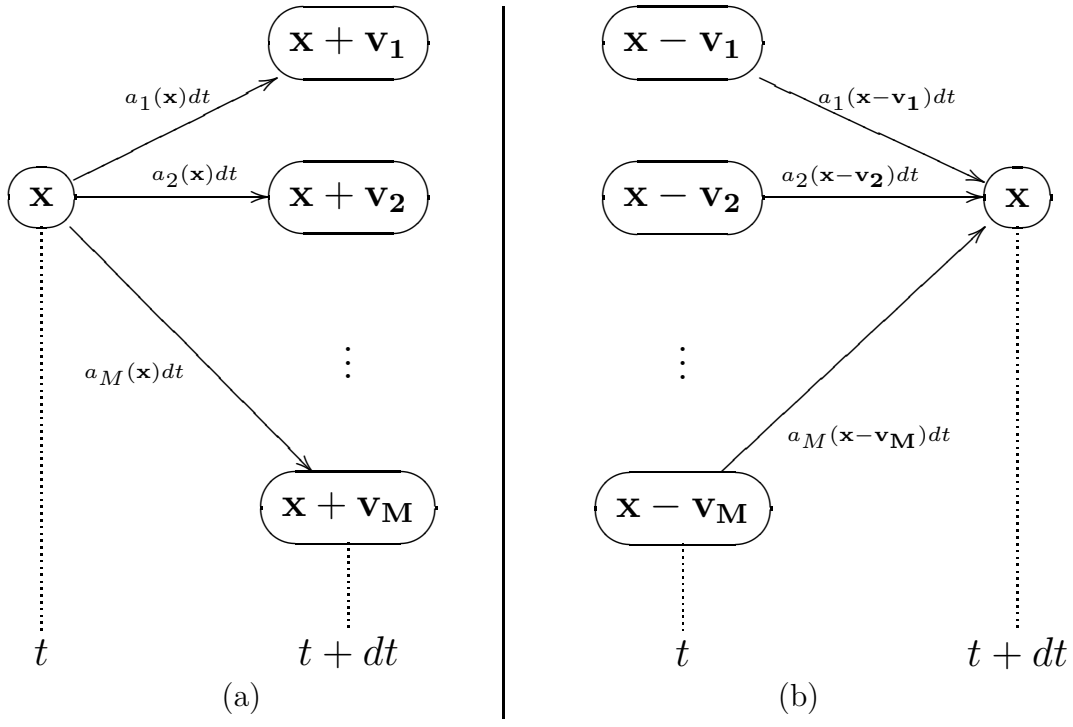


Figure 3.1. (a) Transitions from state $\mathbf{x}$ and (b) transitions to state $\mathbf{x}$.

In this equation, the first term of the right hand side is the probability of the system staying in state $\mathbf{x}$ in the next infinitesimal time, and the second term is the probability of the system moving to state $\mathbf{x}$ in the next infinitesimal time. Taking the limit: $dt \to 0$ and rearranging Equation 3.20 gives the forward *chemical master equation* (CME):

$$\frac{\partial P(\mathbf{x}, t \mid \mathbf{x_0}, t_0)}{\partial t} = \sum_{j=1}^{M} [P(\mathbf{x} - \mathbf{v_j}, t \mid \mathbf{x_0}, t_0) a_j(\mathbf{x} - \mathbf{v_j}) - P(\mathbf{x}, t \mid \mathbf{x_0}, t_0) a_j(\mathbf{x})]. \tag{3.21}$$

Although the integral of the CME gives the probability $P(\mathbf{x}, t \mid \mathbf{x_0}, t_0)$ that captures the evolution of a biochemical system, directly obtaining the solution of the CME of most realistic systems, either analytically or numerically, is not feasible [114, 47]. This is because Equation 3.21 is actually a set of coupled, ordinary differential equations for each system state, and the system-state-space is usually very large, if not infinite, for realistic systems. For example, if a simple biochemical system contains 10 species, each of whose maximum molecular count can be determined to be 99, then the state space of this system is at most 10^{20}, and it is too large to obtain $P(\mathbf{x}, t \mid \mathbf{x_0}, t_0)$ either analytically or numerically.

3.3.4 Stochastic Simulation Algorithm

Due to its intrinsic complexity, the CME itself may not be particularly useful for analyzing the temporal behavior of biochemical systems. In 1976, Gillespie introduced a Monte Carlo simulation algorithm called the *stochastic simulation algorithm* (SSA) to numerically simulate the temporally homogeneous jump Markov process that represents a well-stirred chemically reacting system [52, 53]. The SSA is said to be *exact* in the sense that it does not approximate infinitesimal time increments dt by small but finite time steps Δt, and that it faithfully corresponds to the probability distribution function described by the SCK.

Although the SSA neither tries to solve the CME nor utilizes the CME for numerical simulation, it is derived from the same premise from which the CME is derived—i.e., the propensity functions and the state change vectors. It defines a

probability density function $p(\tau, j \mid \mathbf{x}, t)$ such that

$$p(\tau, j \mid \mathbf{x}, t)d\tau \equiv \text{the probability that, given } \mathbf{X}(t) = \mathbf{x}, \text{ the next reaction in } \Omega \text{ will occur in the infinitesimal time interval } [t+\tau, t+\tau+d\tau), \text{ and it will be } r_j. \tag{3.22}$$

Thus, to generate an exact Monte Carlo procedure (i.e., the SSA), samples of the random variable that are distributed according to the probability density function $p(\tau, j \mid \mathbf{x}, t)$ must be accurately generated. One way to do this is to describe $p(\tau, j \mid \mathbf{x}, t)$ as the product of two probability density functions. Since τ and j in $p(\tau, j \mid \mathbf{x}, t)$ are independent in a temporally homogeneous process, this joint probability density function can be expressed as:

$$p(\tau, j \mid \mathbf{x}, t) = p_1(\tau \mid \mathbf{x}, t) \times p_2(j \mid \mathbf{x}, t), \tag{3.23}$$

where $p_1(\tau \mid \mathbf{x}, t)$ is a probability density function that can be used to answer "when the next reaction will occur" and $p_2(j \mid \mathbf{x}, t)$ is a probability density function that can be used to answer "what the next reaction will be" based on the definition of Equation 3.22. Here, since $\mathbf{X}(t)$ is a temporally homogeneous jump Markov process, the probability density function $p_1(\tau \mid \mathbf{x}, t)$ must be exponentially distributed with the decay constant $a_0(\mathbf{x})$ where:

$$a_0(\mathbf{x}) \equiv \sum_{j=1}^{M} a_j(\mathbf{x}), \tag{3.24}$$

and the probability density function $p_2(j \mid \mathbf{x}, t)$ becomes statistically independent on the next reaction time, and simply a probability of r_j being chosen out of all the reactions. Therefore, these functions can be expressed as:

$$p_1(\tau \mid \mathbf{x}, t) = a_0(\mathbf{x}) \exp\left(-a_0(\mathbf{x})\tau\right), \tag{3.25}$$

$$p_2(j \mid \mathbf{x}, t) = \frac{a_j(\mathbf{x})}{a_0(\mathbf{x})}, \tag{3.26}$$

and the Monte Carlo procedure of using Equations 3.25 and 3.26 is called the *direct method* [52]. The direct method generates samples of the random variable τ which is distributed according to Equation 3.25 via

$$\tau = \frac{-\ln(n_1)}{a_0(\mathbf{x})} \tag{3.27}$$

and it then generates samples of the random variable j which is distributed according to Equation 3.26 via

$$j = \text{smallest integer satisfying } \sum_{\mu=1}^{j} a_\mu(\mathbf{x}) \geq n_2 a_0(\mathbf{x}) \tag{3.28}$$

where n_1 and n_2 are random numbers from the standard unit distribution.

The algorithm shown in Figure 3.2 outlines the algorithm for the direct method. First, Algorithm 3.3.1 initializes the time and state (line 1). Then, it repeats the sequence of evaluating all the propensity functions for the current state, calculating $a_0(\mathbf{x})$ (line 3), generating both τ and j according to Equations 3.27 and 3.28, respectively (lines 4-6), and updating the time and state based on τ and $\mathbf{v_j}$ (line 7). This loop is run until the termination condition such as a time limit is satisfied (line 8).

Although Algorithm 3.3.1 is simple and easy to implement, the temporal behavior analyses of biochemical systems via the SSA may be very expensive. This is because the SSA can only solve the time evolution of the Markov state density function $P(\mathbf{x}, t \mid \mathbf{x_0}, t_0)$ statistically, and it may require a potentially large number of simulation runs in order to estimate the system behavior to a reasonable degree of statistical confidence. Furthermore, the SSA requires every single reaction event to be simulated one at a time. Therefore, each simulation run may require a significant amount of time, especially for realistic biochemical systems.

Algorithm 3.3.1 *Gillespie's direct method*

1: *initialize:* $t \leftarrow t_0$, $\mathbf{x} \leftarrow \mathbf{x_0}$
2: ***repeat***
3: *evaluate all propensity functions and calculate* $a_0(\mathbf{x})$
4: *pick* 2 *unit uniform random numbers* n_1 *and* n_2
5: *set* $\tau \leftarrow -\ln(n_1)/a_0(\mathbf{x})$
6: *set* $j \leftarrow$ *smallest integer satisfying* $\sum_{\mu=1}^{j} a_\mu(\mathbf{x}) \geq n_2 a_0(\mathbf{x})$
7: *update:* $t \leftarrow t + \tau$, $\mathbf{x} \leftarrow \mathbf{x} + \mathbf{v_j}$
8: ***until*** *simulation termination condition is met*

Figure 3.2. Algorithm for Gillespie's direct method.

To alleviate the runtime of the stochastic simulation, Gibson and Bruck have introduced a streamlined implementation of the SSA known as the *next reaction method*, which requires only one standard unit random number asymptotically per state transition, as opposed to two in the direct method, and uses clever data structures so that the propensity functions are re-evaluated only when required [49]. However, it has been shown that the optimized direct method is more efficient than the next reaction method for most realistic systems due to the high cost of maintaining the data structure of the next reaction method [29].

3.4 Approximated Discrete-Stochastic Simulations

Due to substantially high computational demands of the SSA, approximation—which can accelerate the simulation process by sacrificing the exactness of the SSA—is absolutely essential for efficient and effective temporal behavior analyses of realistic biochemical systems. This section presents such approximation methods.

3.4.1 The Bunker et al. Method

Bunker et al. introduced a discrete simulation method in 1974 [23]. Although their method was introduced prior to the introduction of the SSA, it can be viewed as an approximation of the SSA. The core difference between the Bunker et al. method and the SSA is that the former calculates the mean τ instead of sampling τ according to the probability density function $p(\tau, j \mid \mathbf{x}, t)$. Hence, it can avoid the random number generation for sampling τ, requiring generation of only one random number per state transition as opposed to two in the direct method of the SSA.

The mean τ is calculated according to:

$$\langle \tau \rangle = \int_0^\infty \tau' a_0(\mathbf{x}) \exp(-a_0(\mathbf{x})\tau') d\tau' = \frac{1}{a_0(\mathbf{x})}. \tag{3.29}$$

Thus, computing $\langle \tau \rangle$ is easier and faster than computing a sample value of τ. Even though this Bunker et al. method should be faster than the SSA, it still simulates every reaction event one at a time. Thus, the computational cost of its simulation can be very expensive. Especially when the value of $a_0(\mathbf{x})$ is very large due to, for

example, inclusion of fast reactions in a system, the mean τ becomes so small that a substantial number of state transitions is required for simulation of such systems via the Bunker et al. method.

3.4.2 The τ-leaping Methods

To overcome the inefficiency of simulating reaction events one at a time, Gillespie has introduced an approximation method known as the τ-leaping method [56]. The basic idea of the τ-leaping method is to approximate the number of firings of each reaction in a preselected time interval τ rather than individually executing each reaction event. Thus, if τ is selected to be large enough to leap many reaction events, then the simulation process accelerates drastically. However, in order for the τ-leaping method to approximate the SSA well, the leaping time τ must be chosen so that the following *Leap Condition* is satisfied.

Definition 3.1 (Leap Condition) *A condition that requires τ to be so small that changes in the values of the propensity functions of each reaction in the interval $[t, t+\tau]$ are kept minimal.*

With this condition being satisfied, the system advancement from time t to time $t+\tau$ can be well-approximated by the equation:

$$\mathbf{X}(t+\tau) = \mathbf{x} + \sum_{j=1}^{M} \mathbf{v_j} \mathcal{P}_j(a_j(\mathbf{x}), \tau), \tag{3.30}$$

where $\mathcal{P}_j(a_j(\mathbf{x}), \tau)$ is a Poisson-distributed random variable which gives the number of the r_j reaction events that fire in the time interval $[t, t+\tau)$ using $a_j(\mathbf{x})dt$ as the probability of an r_j reaction event to fire in any infinitesimal time dt. In the limit: $\tau \to 0$, this approach becomes equivalent to the SSA. However, in that limit—or near that limit where τ is approximated by a very small but finite Δt in numerical simulations—the τ-leaping method would be painfully slow and even slower than the SSA because most reactions do not occur in such a small time interval. Therefore, the τ-leaping method should not be used if τ is found to be less than a few multiples of $1/a_0(\mathbf{x})$ [57].

The core of the τ-leaping method is the selection of τ. Hence, there have been a number of techniques introduced to improve the original τ-selection to improve the leaping method itself. For example, Gillespie and Petzold have introduced a method to estimate the largest value of τ by choosing an *accuracy control* ε to bound the change in the value of the propensity function by $\varepsilon a_0(\mathbf{x})$ where $\varepsilon \in (0, 1]$ [51]. Cao et al. have further improved the τ-selection by uniformly bounding the relative changes in the values of the propensity functions [28].

There are other variants of τ-leaping methods. For example, several τ-leaping methods are introduced to avoid having a molecular population go negative, which may happen in the original τ-leaping method [112, 32, 26]. The implicit τ-leaping method is introduced to better accommodate systems with stiff conditions where reactions with widely different time scales are present [98]. The trapezoidal τ-leaping method [30] is, then, proposed to have a better accuracy and stiff stability properties than the explicit and the implicit τ-leaping methods by adapting the trapezoidal rule [10] for solving continuous-deterministic ODEs.

While the τ-leaping methods are very promising for some systems, they may not perform well for systems with fast reactions driven by species present in very small counts. This is because, in such systems, the leaping time τ which satisfies the Leaping Condition is so small that leaping many reaction events is not feasible. In such cases, the exact SSA usually performs better than the τ-leaping methods.

3.4.3 Slow-Scale SSA

In simulation of biochemical systems with very large time scale differences such as those often found in complex regulatory systems, some reactions take place much less frequently than some other reactions. Furthermore, it is often the case that firings of the *slow* reactions have greater impact on the system's behavior than the *fast* reactions. Thus, in order to observe interesting temporal behavior of systems essentially controlled by firings of the slow reactions via the SSA, much of the computational time must be spent executing the fast yet less important reactions. In order to accelerate this inefficient simulation process of the SSA in

such situations, Cao et al. introduced an approximate simulation method called the *slow-scale SSA* (ssSSA) [27]. The main idea of the ssSSA is to skip over the expensive fast reactions and simulate only the slow reactions.

The ssSSA partitions a system into a fast subsystem and a slow subsystem. This is done by first partitioning M reactions in a system to a set of the fast reactions: $\mathbf{R^f} \equiv \{r_1^f, \ldots, r_{M_f}^f\}$ and a set of the slow reactions: $\mathbf{R^s} \equiv \{r_1^s, \ldots, r_{M_s}^s\}$ where $M_f + M_s = M$. It then partitions N species in the system into a set of the fast species: $\mathbf{S^f} \equiv \{s_1^f, \ldots, s_{N_f}^f\}$ and a set of the slow species: $\mathbf{S^s} \equiv \{s_1^s, \ldots, s_{N_s}^s\}$ where $N_f + N_s = N$. A fast species $s \in \mathbf{S^f}$ is defined in such a way that there exists a reaction $r \in \mathbf{R^f}$ such that the state of species s is changed by reaction r. Conversely, a slow species $s \in \mathbf{S^s}$ is defined so that, for all reactions $R \in \mathbf{R^f}$, the state of species s is *not* changed by reaction r. From the definition of $\mathbf{S^f}$ and $\mathbf{S^s}$, therefore, a Markov process $\mathbf{X}(t)$ can be partitioned into the fast subsystem $\mathbf{X^f}(t)$ and the slow subsystem $\mathbf{X^s}(t)$.

Since $\mathbf{X^f}(t)$ and $\mathbf{X^s}(t)$ are usually coupled and thus non-Markovian processes, working with these subsystems is notoriously difficult. To approximate $\mathbf{X^f}(t)$ by a Markov process, a *virtual fast process* $\hat{\mathbf{X}}^\mathbf{f}(t)$ is introduced. The process $\hat{\mathbf{X}}^\mathbf{f}(t)$ is composed of the fast species and the fast reactions. In other words, all the slow reactions are ignored in $\hat{\mathbf{X}}^\mathbf{f}(t)$. Thus, $\hat{\mathbf{X}}^\mathbf{f}(t)$ is a Markov process because it does not depend on slow reactions for state transitions and because the state variables for slow species in every propensity function of the fast reactions become constant with the constraint.

Suppose the system is in state $(\mathbf{x^f}, \mathbf{x^s})$ at time t. Then, the propensity function of a slow reaction r_j^s is specified as $a_j^s(\mathbf{x^f}, \mathbf{x^s})$. Now, since $\hat{\mathbf{X}}^\mathbf{f}(t)$ is a fast process by definition, the probability distribution of the fast species can be assumed to move to the stationary distribution $\hat{P}(\mathbf{x^{f'}}, \infty \mid \mathbf{x^f}, \mathbf{x^s})$ before the next slow reaction event fires. Thus, by letting $d_s t$ be the infinitesimal time on the time scale of the slow reactions but very large compared to that of the fast reactions, the probability that one slow reaction r_j^s event occurs in $[t, t + d_s t)$ can be well approximated by $\bar{a}_j^s(\mathbf{x^f}, \mathbf{x^s}) d_s t$ where the *slow-scale propensity function* $\bar{a}_j^s(\mathbf{x^f}, \mathbf{x^s})$ is defined as:

$$\bar{a}_j^s(\mathbf{x^f}, \mathbf{x^s}) \equiv \sum_{\mathbf{x^{f'}}} \hat{P}(\mathbf{x^{f'}}, \infty \mid \mathbf{x^f}, \mathbf{x^s}) a_j^s(\mathbf{x^{f'}}, \mathbf{x^s}). \tag{3.31}$$

Therefore, the slow-scale propensity functions can be used to predict when and which slow reaction event fires next, and update $\mathbf{x^f}$ and $\mathbf{x^s}$ accordingly. Furthermore, a sample of $\hat{P}(\mathbf{x^{f'}}, \infty \mid \mathbf{x^f}, \mathbf{x^s})$ can be used to update $\mathbf{x^f}$ to approximate the changes from the fast reactions.

Although the ssSSA can efficiently approximate the stochastic simulation of some systems with large time scale differences, it has several limitations. For example, it is not feasible to compute $\hat{P}(\mathbf{x^{f'}}, \infty \mid \mathbf{x^f}, \mathbf{x^s})$ for most systems, and thus the stationary distribution usually has to be computed approximately. Also, since the propensity functions of some reactions can change drastically in each simulation, computationally expensive partitioning of reactions and species may need to be performed frequently in such situations. Furthermore, because the number of species and reactions is not reduced by this method, the complex procedure of the ssSSA may have difficulty in simulating large-scale biochemical systems.

CHAPTER 4

REACTION-BASED ABSTRACTION

In well-stirred chemical and biological molecular systems, including genetic regulatory networks which this dissertation focuses on, reaction-based (REB) representations typically provide the most detailed level of specification for the underlying system structure and dynamics [15]. REB abstraction methods are used to reduce a REB model's size by merging reactions, removing irrelevant reactions, etc. Our tool REB2SAC includes several such techniques to facilitate efficient analysis of biochemical systems [73, 74]. Each REB abstraction method traverses the graph structure of the REB model and applies transformations to it when the respective conditions are satisfied. The result is a new REB model with fewer reactions and species. This chapter first defines the REB model formally in Section 4.1. Section 4.2 then describes the modeling assumptions that are made so that the algorithms of our abstraction methods can be presented without having to deal with all the possible modeling scenarios. Finally, this chapter presents the main REB abstraction methods that we have implemented.

4.1 Reaction-Based Model

The REB model is a bipartite weighted directed graph that connects species based on the interactions that they can have via a set of reaction channels in a system. Thus, it can describe both the CCK model and the SCK model. A REB model can be encoded in an emerging standard, the *Systems Biology Markup Language* (SBML) [42]. Thus, REB models can be conveniently constructed using SBML-compliant modeling tools such as PathwayBuilder [16] and CellDesigner [44]. The use of this standardized format has the advantage of allowing for easy exchange of computational models by researchers as well as the ability to analyze models by

a variety of SBML-compliant analysis tools. The REB model is formally defined as follows.

Definition 4.1 (REB model) *A REB model is specified with* $\langle \mathbf{S}, \mathbf{R}, \mathbf{R_{rev}}, \mathbf{E}, \mathbf{K} \rangle$ *where* $\mathbf{S}$ *is the set of species nodes,* $\mathbf{R}$ *is the set of reaction nodes,* $\mathbf{R_{rev}} \subseteq \mathbf{R}$ *is the set of reversible reactions,* $\mathbf{E} : ((\mathbf{S} \times \mathbf{R}) \cup (\mathbf{R} \times \mathbf{S})) \rightarrow \mathbb{N}$ *is a function that returns the stoichiometry of the species with respect to a reaction, and* $\mathbf{K} : \mathbf{R} \rightarrow (\mathbb{R}^{|\mathbf{S}|} \rightarrow \mathbb{R})$ *are the kinetic rate laws for the reactions.*

For example, a REB model with two reactions r_1 and r_2 of the form:

$$r_1 : \; s_1 + s_2 \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} s_2 + s_3$$

$$r_2 : \; s_1 + s_3 \xrightarrow{k_2} 2s_1 + s_2$$

contains the following sets:

$$\begin{aligned}
\mathbf{S} &= \{s_1, s_2, s_3\}, \\
\mathbf{R} &= \{r_1, r_2\}, \\
\mathbf{R_{rev}} &= \{r_1\}, \\
\mathbf{E} &= \{((s_1, r_1), 1), ((s_2, r_1), 1), ((s_3, r_1), 0), \\
&\quad ((r_1, s_1), 0), ((r_1, s_2), 1), ((r_1, s_3), 1), \\
&\quad ((s_1, r_2), 1), ((s_2, r_2), 0), ((s_3, r_2), 1), \\
&\quad ((r_2, s_1), 2), ((r_2, s_2), 1), ((r_2, s_3), 0)\}, \\
\mathbf{K} &= \{(r_1 \rightarrow ((|s_1|, |s_2|, |s_3|) \rightarrow (k_1 |s_1| |s_2| - k_{-1} |s_2| |s_3|))), \\
&\quad (r_2 \rightarrow ((|s_1|, |s_2|, |s_3|) \rightarrow (k_2 |s_1| |s_3|)))\}.
\end{aligned}$$

In the kinetic rate law, $|s|$ is a variable that represents the state of species s. In other words, in CCK analysis, $|s|$ usually represents the concentration of species s, while in SCK analysis, $|s|$ represents the molecular count of species s. Similarly, the units for the rate constants, k_j, are adjusted based on the analysis. Note that $|s|_0$ is used to denote the initial state of species s. Also note that the user can specify a set of *interesting species* (i.e., $\mathbf{S_i} \subseteq \mathbf{S}$), which should never be abstracted so that they can always be analyzed.

To simplify the description of the algorithms in this dissertation, this section introduces several sets. $\mathbf{R_s^r}$, $\mathbf{R_s^p}$, and $\mathbf{R_s^m}$ are the sets of reactions in which species s appears as a reactant, product, and *modifier*, respectively. Here, a reactant is a species that is consumed by a reaction, a product is a species that is produced by a reaction, and a modifier is a species that is neither produced nor consumed by a reaction, as defined in the SBML standard [42]. Similarly, $\mathbf{S_r^r}$, $\mathbf{S_r^p}$, and $\mathbf{S_r^m}$, are the sets of species that appear in reaction r as a reactant, product, and modifier, respectively. These sets are defined formally below:

$$\begin{aligned}
\mathbf{R_s^r} &= \{r \in \mathbf{R} \mid \mathbf{E}(s,r) > \mathbf{E}(r,s)\},\\
\mathbf{R_s^p} &= \{r \in \mathbf{R} \mid \mathbf{E}(r,s) > \mathbf{E}(s,r)\},\\
\mathbf{R_s^m} &= \{r \in \mathbf{R} \mid \mathbf{E}(s,r) > 0 \wedge \mathbf{E}(s,r) = \mathbf{E}(r,s)\},\\
\mathbf{S_r^r} &= \{s \in \mathbf{S} \mid \mathbf{E}(s,r) > \mathbf{E}(r,s)\},\\
\mathbf{S_r^p} &= \{s \in \mathbf{S} \mid \mathbf{E}(r,s) > \mathbf{E}(s,r)\},\\
\mathbf{S_r^m} &= \{s \in \mathbf{S} \mid \mathbf{E}(s,r) > 0 \wedge \mathbf{E}(s,r) = \mathbf{E}(r,s)\}.
\end{aligned}$$

In these definitions, a species is considered a reactant of a reaction only if the net change of the state of that species by a reaction is negative. Similarly, it is a product only if the net change of the state of that species by a reaction is positive. Finally, it is considered a modifier if it is used in a reaction but the state of that species is not changed by that reaction. Thus, the REB model for our example includes the following nonempty sets:

$$\begin{aligned}
&\mathbf{R_{s_1}^r} = \{r_1\}, \quad \mathbf{R_{s_1}^p} = \{r_2\}, \quad \mathbf{R_{s_2}^p} = \{r_2\},\\
&\mathbf{R_{s_2}^m} = \{r_1\}, \quad \mathbf{R_{s_3}^r} = \{r_2\}, \quad \mathbf{R_{s_3}^p} = \{r_1\},\\
&\mathbf{S_{r_1}^r} = \{s_1\}, \quad \mathbf{S_{r_1}^p} = \{s_3\}, \quad \mathbf{S_{r_1}^m} = \{s_2\},\\
&\mathbf{S_{r_2}^r} = \{s_3\}, \quad \mathbf{S_{r_2}^p} = \{s_1, s_2\}.
\end{aligned}$$

4.2 Modeling Assumptions for Abstraction

To simplify the description of the algorithms of several abstraction methods, a couple of assumptions are made to restrict the presentation of an input REB model.

The first assumption is that the kinetic law expression of an n-merization reaction of the reactant s always has the term $|s|^n$ even when a REB model is used for SCK analysis. For example, a REB model with a reaction r of the form:

$$2s_1 \xrightarrow{k_1} s_2 \tag{4.1}$$

contains the following kinetic law set:

$$\mathbf{K} \;=\; \{(r \rightarrow ((|s_1|, |s_2|) \rightarrow (k_1 |s_1|^2)))\}.$$

In the case of SCK analysis, a stricter requirement is that the kinetic law expression of rection r be $k_1 |s_1| (|s_1| - 1)$. However, by assuming that $|s_1|$ is relatively high, the relative error between $k_1 |s_1| (|s_1| - 1)$ and $k_1 |s_1|^2$:

$$\epsilon = \frac{|s_1|^2 - |s_1| (|s_1| - 1)}{|s_1| (|s_1| - 1)} \tag{4.2}$$

becomes very small. For example, even when $|s_1|$ is as low as 10, the relative error ϵ becomes 0.11, and if $|s_1|$ is 100, then ϵ becomes 0.01. Thus, we make this simplification throughout this dissertation to keep several model abstractions of a REB model interchangeable between the CCK analysis and the SCK analysis. Note that, for the SCK model, a stricter model can be generated by simply replacing $|s|^n$ with $\prod_{i=0}^{n-1} (|s| - i)$.

The second assumption is that reactions are modeled as reversible reactions whenever possible. From the definition of the REB model, the same biochemical process can be represented by different REB models. However, it is usually the case that only a handful of such models biochemically make sense. For example, the following reversible reaction system:

$$s_1 + s_2 \rightleftharpoons s_3 \tag{4.3}$$

can be represented by three different REB models. The first way is to represent this system by a model with a reversible reaction with species s_1 and s_2 as the reactants (i.e., $\mathbf{R_{rev}} = \{r_1\}, \mathbf{S^r_{r_1}} = \{s_1, s_2\}, \mathbf{S^p_{r_1}} = \{s_3\}$). The second way is to represent the reaction system by a model with a reversible reaction with species

s_1 and s_2 as the products (i.e., $\mathbf{R_{rev}} = \{r_1\}, \mathbf{S^r_{r_1}} = \{s_3\}, \mathbf{S^p_{r_1}} = \{s_1, s_2\}$). The third way is to represent the system by a model with a pair of two reactions (i.e., $\mathbf{R} = \{r_1, r_2\}, \mathbf{R_{rev}} = \emptyset$). However, if the modeler's intention is to treat s_1 and s_2 as the reactants and s_3 as the product, then it is typically the case that the modeler chooses the first REB model to represent this reaction system.

Modeling biochemical systems using $\mathbf{R_{rev}}$ with the species in the correct order has the advantage of allowing our abstraction methods to correctly identify which ones should be the reactants and which ones should be the products based on the flow of the overall reaction process that the user has intended to model. Thus, our abstraction methods assume that reversible reactions are modeled using $\mathbf{R_{rev}}$ with correctly specifying the forward reactions and the backward reactions. This means that, if there exist irreversible reactions r_f and r_b such that

$$|\mathbf{S^r_{r_f}}| > 0 \wedge |\mathbf{S^r_{r_b}}| > 0 \wedge \mathbf{S^r_{r_f}} = \mathbf{S^p_{r_b}} \wedge \mathbf{S^p_{r_f}} = \mathbf{S^r_{r_b}} \wedge \mathbf{S^m_{r_f}} = \mathbf{S^m_{r_b}},$$

then these reactions are assumed to be modeled by combining the forward reaction, r_f, and the backward reaction, r_b, to form a reversible reaction.

Owing to this constraint, while the dynamics of each species $s \in \mathbf{S}$ in the corresponding ODE model can be simply described as:

$$\frac{d\,|s|}{dt} = \sum_{r \in \mathbf{R}} (\mathbf{E}(r, s) - \mathbf{E}(s, r))\,\mathbf{K}(r), \tag{4.4}$$

each reversible reaction must first be split into a pair of forward and backward reactions in order to analyze the temporal behavior of a REB model via the SSA. This can be achieved by performing the algorithm shown in Figure 4.1. Algorithm 4.2.1 iterates on each reversible reaction r. It first creates a backward reaction r' (line 2). Based on the reactants, products, and modifiers of reaction r, then, the products, reactants, and modifiers of reaction r' are generated (lines 3-5). This algorithm assumes that the kinetic law of each reaction r is in the form of "$expression_f - expression_b$". It then splits the kinetic law expression into the forward kinetic law expression and the backward kinetic law expression to put them into reactions r and r', respectively (lines 6-8). Finally, it sets $\mathbf{R_{rev}}$ to be an empty set and returns the new model (lines 10-11).

Algorithm 4.2.1 *Split reversible reactions*
Model SplitReversible(Model M)

1: ***for all*** $r \in \mathbf{R_{rev}}$ ***do***
2: $M \leftarrow addReaction(M, r')$
3: $\forall s \in \mathbf{S_r^r}$. $M \leftarrow addProduct(M, s, r', \mathbf{E}(s, r))$
4: $\forall s \in \mathbf{S_r^p}$. $M \leftarrow addReactant(M, s, r', \mathbf{E}(r, s))$
5: $\forall s \in \mathbf{S_r^m}$. $M \leftarrow addModifier(M, s, r', \mathbf{E}(s, r))$
6: $\text{“}expression_f - expression_b\text{”} \leftarrow \mathbf{K}(r)$
7: $\mathbf{K}(r) \leftarrow \text{“}expression_f\text{”}$
8: $\mathbf{K}(r') \leftarrow \text{“}expression_b\text{”}$
9: ***end for***
10: $\mathbf{R_{rev}} \leftarrow \emptyset$
11: ***return*** M

Figure 4.1. Algorithm to split reversible reactions.

Using the REB model with $\mathbf{R_{rev}}$ being empty, the SSA can be more efficiently performed than the original direct method [52]. This is because the structure of the REB model enables one to easily determine which species' states are affected by the firing of a reaction event. Thus, similar to the data structure used in the next reaction method [48, 49], it can minimize the number of evaluations to update the value of each propensity function for each step. Figure 4.2 shows the algorithm for a more efficient direct method using the REB model where $\mathbf{S} \equiv \{s_1, \ldots, s_N\}$ and $\mathbf{R} \equiv \{r_1, \ldots, r_M\}$. Algorithm 4.2.2 first initializes the time variable and state variables of each species (line 1). It then evaluates all the propensity functions based on the current species' state, puts the value of the propensity function of r_j into a_j, and calculates the sum of all the propensity functions a_0 (lines 2-3). Next, it picks two unit uniform random numbers, and uses them to find the next reaction time and which reaction is the next reaction (lines 5-7). Based on the selected reaction $r_{j'}$, the state of each species that is used as either a reactant or product is updated (lines 8-10). At this point, if the termination condition is satisfied, then the simulation ends here (lines 11-12). If not, then the kinetic law is updated for each reaction a_j that uses as a reactant, product, or modifier any species whose state is just changed, and the values of a_0 and each a_j are also updated (lines 13-22). It then

Algorithm 4.2.2 *Direct method using the REB model*

1: $t \leftarrow t_0$, $\forall i \in [1, N]$. $|s_i| \leftarrow |s_i|_0$
2: *evaluate* $\mathbf{K}(r_j)$ *for all* $j \in [1, M]$ *and put the value in* a_j
3: $a_0 \leftarrow \sum_j a_j$
4: *pick* 2 *unit uniform random numbers* n_1 *and* n_2
5: $\tau \leftarrow -\ln(n_1)/a_0$
6: $j' \leftarrow$ *smallest integer satisfying* $\sum_{j=1}^{j'} a_j \geq n_2 a_0$
7: $t \leftarrow t + \tau$
8: ***for all*** $s \in (\mathbf{S}^{\mathbf{r}}_{\mathbf{r_{j'}}} \cup \mathbf{S}^{\mathbf{p}}_{\mathbf{r_{j'}}})$ ***do***
9: $\quad |s| \leftarrow |s| + \mathbf{E}(r_{j'}, s) - \mathbf{E}(s, r_{j'})$
10: ***end for***
11: ***if*** *simulation termination condition is met* ***then***
12: $\quad$ ***exit***
13: ***else***
14: $\quad \mathbf{Q} \leftarrow \emptyset$
15: $\quad$ ***for all*** $s \in (\mathbf{S}^{\mathbf{r}}_{\mathbf{r_{j'}}} \cup \mathbf{S}^{\mathbf{p}}_{\mathbf{r_{j'}}})$ ***do***
16: $\quad\quad \mathbf{Q} \leftarrow \mathbf{Q} \cup \mathbf{R}^{\mathbf{r}}_{\mathbf{s}} \cup \mathbf{R}^{\mathbf{p}}_{\mathbf{s}} \cup \mathbf{R}^{\mathbf{m}}_{\mathbf{s}}$
17: $\quad$ ***end for***
18: $\quad$ ***for all*** j *such that* $r_j \in \mathbf{Q}$ ***do***
19: $\quad\quad$ *evaluate* $\mathbf{K}(r_j)$ *and put the value in* a'
20: $\quad\quad a_0 \leftarrow a_0 + (a' - a_j)$
21: $\quad\quad a_j \leftarrow a'$
22: $\quad$ ***end for***
23: $\quad$ ***go to*** *line 4*
24: ***end if***

Figure 4.2. Algorithm for a more efficient direct method using the REB model.

goes back to line 4 and repeats the process (line 23). This simulation algorithm is implemented in REB2SAC to accommodate an optimized version of the SSA for the REB model data structure. This simulation method is used for efficient analyses of various biochemical systems [74, 75, 72, 14, 87].

4.3 Michaelis-Menten Approximation

This section considers the following bimolecular enzymatic reaction scheme:

$$E + S \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} C \xrightarrow{k_2} E + P \tag{4.5}$$

where E, S, C, and P represent an enzyme, a substrate, an enzyme-substrate complex, and a product, respectively. This enzymatic reaction scheme specifies the

transformation of a substrate, S, into a product, P, catalyzed by an enzyme, E, where E has one active site to which S can bind to form C. Because characteristics of such enzymatic reactions can be ubiquitously found in biochemical systems, understanding of such mechanism may be crucial to elucidate how component-level dynamics play a role in an overall system behavior. Unfortunately, it is often the case that these enzymatic reactions come with very rapid complex formation and complex breakup reactions compared with the production formation, resulting in very large time scale differences which can significantly contribute to higher computational costs for both CCK and SCK analyses. Therefore, abstracting away such expensive enzymatic reactions is essential for efficient analysis of biochemical systems.

Since Reaction 4.5 has four species, the CCK model of this reaction scheme has four differential equations. However, by assuming that the states of the species in Reaction 4.5 are not changed by other reactions, the following mass conservations can be established to reduce the number of equations:

$$E_{tot} = |E|(t) + |C|(t), \tag{4.6}$$

$$S_{tot} = |S|(t) + |C|(t) + |P|(t) \tag{4.7}$$

for all $t \geq 0$ where E_{tot} and S_{tot} are constants. Thus, using these mass conservation equations, the CCK model of Reaction 4.5 is governed by the following two ordinary differential equations:

$$\frac{d|C|}{dt} = k_1(E_{tot} - |C|)(S_{tot} - |C| - |P|) - (k_{-1} + k_2)|C|, \tag{4.8}$$

$$\frac{d|P|}{dt} = k_2|C|. \tag{4.9}$$

The relationship between an enzyme E and a substrate S in this enzymatic reaction scheme was first proposed and its CCK model was developed by Henri in the early 20th century [62]. Due to the analytical difficulty of following the dynamic behavior of the enzymatic reaction, Michaelis and Menten then studied the enzyme action by measuring the initial reaction rate, instead [84]. This Michaelis and

Menten (MM) reaction eliminates the formation of complex C, and its kinetics describes the speed of the product formation solely based on S as:

$$\frac{d\,|P|}{dt} = \frac{V_{max}\,|S|}{K + |S|} \tag{4.10}$$

where $V_{max} \equiv k_2 E_{tot}$ is a constant specifying the maximum speed and K is another constant. Hence, it can facilitate better analytical understanding of the dynamic behavior of an enzymatic reaction by eliminating the intermediate species and by reducing the dimensionality of the system. Furthermore, unlike the parameters of Reaction 4.5: k_1 and k_{-1}, the parameters, K and V_{max}, can actually be measured experimentally. Thus, a MM reaction can be constructed and simulated even when full knowledge of the underlying enzymatic reaction is not available and the enzymatic reaction cannot be analyzed quantitatively at that level of detail.

Briggs and Haldane introduced the theoretical basis of the MM reaction by assuming that the changes in $|C|$ over time is minimal on the time scales of interest (i.e., $\frac{d|C|}{dt} \approx 0$) [22]. This approximation is known as the *quasi-steady-state approximation* (QSSA), and it generates Equation 4.10 by replacing K with K_M where $K_M \equiv (k_{-1}+k_2)/k_1$ which is known as the MM constant. The application of the QSSA in Reaction 4.5 can be justified with *singular perturbation theory* which states that the error from the QSSA estimate is small when the condition

$$\frac{E_{tot}}{|S|_0 + K_M} \ll 1 \tag{4.11}$$

is satisfied [107, 65].

In SCK, this type of approximation had not been given much consideration until recently. However, this has changed due to the substantial computational demands of the SSA. Aside from the advantage of reducing dimensionality of the system, one major advantage of the stochastic version of QSSA is that it can substantially reduce the simulation time by removing the fast reactions (i.e., the complex formation and the complex breakup). Thus, to facilitate more efficient temporal behavior analysis, extended MM approximations have been applied to several biochemical systems [7, 9, 116]. Also, the mathematical justification for the application of the QSSA

within the SCK framework has been investigated to establish a theoretical basis to illustrate how the QSSA can be applied to the SSA [96, 101]. Since the dimension of the model can be reduced using Equations 4.6 and 4.7, the CME of Reaction 4.5 becomes:

$$\begin{aligned}\frac{\partial Pr(c,p;t)}{\partial t} &= k_1(E_{tot}-c+1)(S_{tot}-c-p+1)Pr(c-1,p;t)\\ &+ k_{-1}(c+1)Pr(c+1,p;t) + k_2(c+1)Pr(c+1,p-1;t)\\ &- \left[k_1(E_{tot}-c)(S_{tot}-c-p) + (k_{-1}+k_2)c\right]Pr(c,p;t),\end{aligned} \tag{4.12}$$

where $Pr(c,p;t)$ is the probability that $|C|(t) = c$ and $|P|(t) = p$.

The derivation of the stochastic QSSA model of Reaction 4.5 begins by assuming that the net rate change of the probability distribution of $|C|$ becomes approximately zero for the time scales of interest. From this assumption, the reversible reaction to form and break up complex C can be removed, and the kinetic law for the production reaction can be approximated by replacing $|C|$ by its average $\langle|C|\rangle$ to form: $k_2\langle|C|\rangle$. It turns out that Condition 4.11 is still valid for the condition of the stochastic QSSA [96, 101]. And in that situation, the average $|C|$ can be well approximated as:

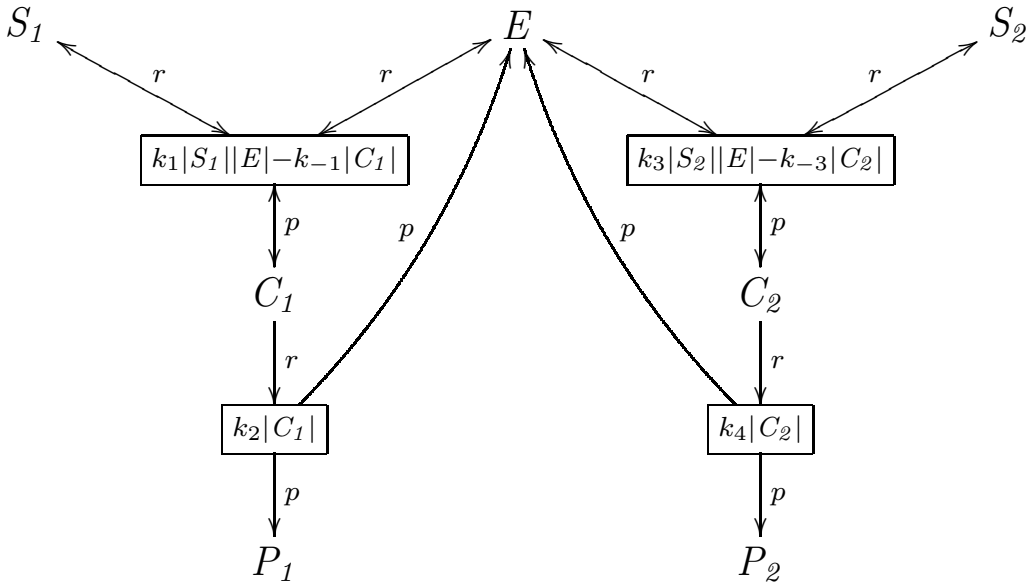
$$\langle|C|\rangle \approx \frac{E_{tot}|S|}{K_M + |S|}. \tag{4.13}$$

Thus, Equation 4.12 can be approximated by the QSSA to be:

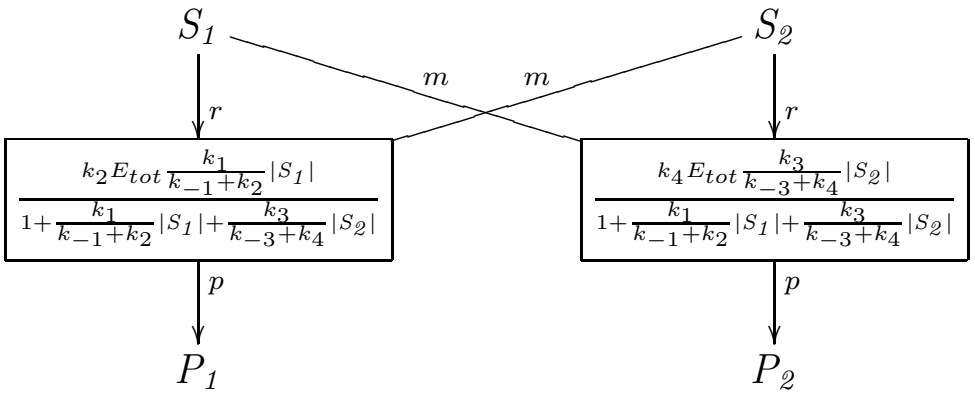
$$\frac{\partial Pr(s,p;t)}{\partial t} = \frac{k_2E_{tot}(s+1)}{K_M+(s+1)}Pr(s+1,p-1;t) - \frac{k_2E_{tot}s}{K_M+s}Pr(s,p;t), \tag{4.14}$$

where $Pr(s,p;t)$ is the probability that $|S|(t) = s$ and $|P|(t) = p$. Therefore, conveniently, the QSSA can be applied to both the CCK model and the SCK model to reduce the model complexity.

Figure 4.3 shows a graphical representation of a more complex competitive enzymatic reaction to illustrate the application of the QSSA. In this graphical representation, a reaction that is connected to a species with a double arrow is a shorthand to show a reversible reaction, and species connected to a reaction with letters, r, p, and m are a reactant, a product, and a modifier for that reaction, respectively. In Figure 4.3(a), the substrates S_1 and S_2 compete to bind to the



(a)



(b)

Figure 4.3. Quasi-steady-state approximation: (a) the original model, and (b) the abstracted model.

enzyme E to produce products P_1 and P_2 by first forming complexes C_1 and C_2, respectively. An extended form of the QSSA can be applied to this network to remove this enzyme, its complex forms, and the reactions that form these complexes. Importantly, this also clarifies the essential biological meaning of the process by

removing intermediate steps, which may otherwise obscure the functional logic of the mechanism. The resulting abstracted reaction model is shown in Figure 4.3(b).

The algorithms shown in Figure 4.4 implement the QSSA for multiple alternative substrate systems [73, 74] which is a generalization of the complete characterization of enzyme-substrate and enzyme-substrate-competitor reactions [106]. First, Algorithm 4.3.1 considers each species, s, as a potential enzyme. Each species is checked using Algorithm 4.3.2. If s is an interesting species or does not occur as a reactant in any reaction, then s is not considered further (line 2). Otherwise, each reaction, r_1, in which s is a reactant is considered in turn. If r_1 is not reversible, does not have two reactants, or does not have a rate law of the right form (i.e., $k_1 |s| |s_1| - k_{-1} |s_c|$), then again s is not considered further (line 4). Reaction r_1 combines s and s_1 into a complex s_c. If the initial molecule count of this complex is not 0, s_c is an interesting species, s_c does not occur as a reactant or product in exactly one reaction, or occurs as a modifier in any, then again this approximation is terminated for s (lines 5-6). The reaction r_2 converts s_c into a product and releases the enzyme s. If this reaction is reversible, does not have exactly one reactant and no modifiers, does not have s as a product, has more than two products, or does not have a rate law of the form, $k_2 |s_c|$, then s is not considered further (lines 7-9). Finally, it checks the validity of the quasi-steady-state assumption by comparing the ratio of $|s|$ and the sum of $|s_1|$ and K_M to the predefined threshold constant T_1 (line 10). For each reaction, a configuration is formed that includes the substrate s_1, complex s_c, constant $k_1/(k_{-1} + k_2)$, production rate constant k_2, complex forming reaction r_1, and product forming reaction r_2 (line 11). If Algorithm 4.3.2 terminates successfully (i.e., returns a nonempty set of configurations, $\mathbf{C}$), then Algorithm 4.3.3 is called to apply the transformation to the REB model. First, it loops through the set of configurations to form an expression that is used in the denominator in each new rate law as well as forming a list of all the substrates that bind to the enzyme s (lines 1-6). Next, for each configuration $(s_1, s_c, K_1, k_2, r_1, r_2)$, it makes the substrate s_1 a reactant for r_2, makes all other substrates modifiers for r_2, creates

Algorithm 4.3.1 *Standard quasi-steady-state approximation*
Model StandardQSSA(Model M)

1: ***for all*** $s \in \mathbf{S}$ ***do***
2: $\quad \mathbf{C} \leftarrow QSSAConditionSatisfied(M, s)$
3: $\quad$ ***if*** $\mathbf{C} \neq \emptyset$ ***then*** $M \leftarrow QSSATransform(M, s, \mathbf{C})$
4: ***end for***
5: ***return*** M

Algorithm 4.3.2 *Check the conditions for QSSA*
Configs QSSAConditionSatisfied(Model M, Species s)

1: $\mathbf{C} \leftarrow \emptyset$
2: ***if*** $(s \in \mathbf{S_i}) \vee (|\mathbf{R^r_s}| = 0)$ ***then return*** $\emptyset$
3: ***for all*** $r_1 \in \mathbf{R^r_s}$ ***do***
4: $\quad$ ***if*** $(r_1 \notin \mathbf{R_{rev}}) \vee (|\mathbf{S^r_{r_1}}| \neq 2) \vee (\mathbf{K}(r_1) \neq$ "$k_1 |s| |s_1| - k_{-1} |s_c|$ ") ***then return*** $\emptyset$
5: $\quad$ ***if*** $(|s_c|_0 \neq 0) \vee (s_c \in \mathbf{S_i})$ ***then return*** $\emptyset$
6: $\quad$ ***if*** $(|\mathbf{R^r_{s_c}}| \neq 1) \vee (|\mathbf{R^m_{s_c}}| \neq 0) \vee (\mathbf{R^p_{s_c}}| \neq 1)$ ***then return*** $\emptyset$
7: $\quad \{r_2\} \leftarrow \mathbf{R^r_{s_c}}$
8: $\quad$ ***if*** $r_2 \in \mathbf{R_{rev}}) \vee (|\mathbf{S^r_{r_2}}| \neq 1) \vee (|\mathbf{S^m_{r_2}}| \neq 0)$ ***then return*** $\emptyset$
9: $\quad$ ***if*** $(s \notin \mathbf{S^p_{r_2}}) \vee (|\mathbf{S^p_{r_2}}| \notin \{1, 2\}) \vee (\mathbf{K}(r_2) \neq$ "$k_2 |s_c|$ ") ***then return*** $\emptyset$
10: $\quad$ ***if*** $|s|_0/(|s_1|_0 + (k_{-1} + k_2)/k_1) > T_1$ ***then return*** $\emptyset$
11: $\quad \mathbf{C} \leftarrow \mathbf{C} \cup \{(s_1, s_c, k_1/(k_{-1} + k_2), k_2, r_1, r_2)\}$
12: ***end for***
13: ***return*** $\mathbf{C}$

Algorithm 4.3.3 *Perform the QSSA transformation*
Model QSSATransform(Model M, Species s, Configs $\mathbf{C}$*)*

1: *Exp* $Z \leftarrow 1$
2: $\mathbf{L} \leftarrow \emptyset$
3: ***for all*** $(s_1, s_c, K_1, k_2, r_1, r_2) \in \mathbf{C}$ ***do***
4: $\quad Z \leftarrow Z + (K_1 * |s_1|)$
5: $\quad \mathbf{L} \leftarrow \mathbf{L} \cup \{s_1\}$
6: ***end for***
7: ***for all*** $(s_1, s_c, K_1, k_2, r_1, r_2) \in \mathbf{C}$ ***do***
8: $\quad M \leftarrow addReactant(M, s_1, r_2, \mathbf{E}(s_1, r_1))$
9: $\quad \forall m \in \mathbf{L} \setminus \{s_1\}.\ M \leftarrow addModifier(M, m, r_2, 1)$
10: $\quad \mathbf{K}(r_2) \leftarrow (k_2 * |s|_0 * k_e * |s_1|)/Z$
11: $\quad M \leftarrow removeSpecies(M, s_c)$
12: $\quad M \leftarrow removeReaction(M, r_1)$
13: ***end for***
14: $M \leftarrow removeSpecies(M, s)$
15: ***return*** M

Figure 4.4. Algorithms to perform the quasi-steady-state approximation.

a new rate law for r_2, and removes species s_c and reaction r_1 (lines 7-13). Finally, this algorithm removes the enzyme, s (line 14).

If the complex C dissociates into E and S much faster than it is converted into the product P (i.e., $k_{-1} >> k_2$), then the formation of the complex can be seen to equilibrate with the breakup of the complex on the time scales of the production reaction. Thus, a more aggressive approximation known as the *rapid equilibrium approximation* can be applied to further reduce the complexity of the kinetic law in such situations. This approximation transforms an enzymatic one-substrate reaction to the MM form with $K \equiv k_{-1}/k_1$ in Equation 4.10.

4.4 Production-Passage-Time Approximation

While the QSSA and the rapid equilibrium approximation can significantly reduce the complexity of a computational model, these approximation methods may not perform well in terms of accuracy when the underlying hypotheses are violated. To better accommodate such cases, we have developed an approximation method called *production-passage-time approximation* (PPTA) which can reduce the simulation time of enzymatic reaction models while maintaining better accuracy in more general settings [72]. Our new approach approximates the passage time of the complex C which is destined to turn into the product P, and only tracks such instances of C. Thus, the PPTA eliminates the substrate-complex loop by removing the fast complex breakup reaction, allowing a substantial acceleration in stochastic simulations of enzymatic reactions.

To describe the PPTA method, this section introduces several notations and assumptions. Let irreversible reactions r_1, r_{-1}, and r_2 be the complex formation reaction, the complex breakup reaction, and the production reaction in Reaction 4.5, respectively. Then, the enzymatic reaction scheme contains the following propensity functions for each reaction r_i:

$$\mathbf{K}(r_1)(\mathbf{X}) \equiv a_1(\mathbf{X}) = k_1 \left|E\right| \left|S\right| \tag{4.15}$$

$$\mathbf{K}(r_{-1})(\mathbf{X}) \equiv a_{-1}(\mathbf{X}) = k_{-1} \left|C\right| \tag{4.16}$$

$$\mathbf{K}(r_2)(\mathbf{X}) \equiv a_2(\mathbf{X}) = k_2 \left|C\right| \tag{4.17}$$

where $\mathbf{X} = (\left|E\right|, \left|S\right|, \left|C\right|, \left|P\right|)$. Reaction 4.5 is considered to have the initial condition: $\mathbf{X}(t_0) = \mathbf{x_{t_0}}$, where $\mathbf{x_{t_0}} = (E_{tot}, S_{tot}, 0, 0)$, $E_{tot} \geq 1$, and $S_{tot} \geq 1$. Let $\mathbf{x}_\infty = (E_{tot}, 0, 0, S_{tot})$, then the probability that $\mathbf{X}(t) = \mathbf{x}_\infty$ given $\mathbf{X}(t_0) = \mathbf{x_{t_0}}$ approaches 1, as $t \to \infty$. In other words, in any simulation run, the enzymatic reaction process always reaches $\mathbf{x}_\infty$ eventually. In order for each numerical realization of $\mathbf{X}(t)$ to transition from $\mathbf{x_{t_0}}$ to $\mathbf{x}_\infty$, S must be transformed into C at least S_{tot} times and C must be converted into P exactly S_{tot} times. Thus, let $\mathbf{x}^{(i)}(t)$ be the i-th sample trajectory of $\mathbf{X}(t)$ given that $\mathbf{X}(t_0) = \mathbf{x_{t_0}}$ and $\mathbf{T_i}$ be a set of time instances such that each time instance, t^i_j, represents the time point where the j-th reaction event occurs in $\mathbf{x}^{(i)}(t)$. Then, the statement $\forall i.\ |\mathbf{T_i}| \in [2S_{tot}, \infty)$ must be true. In other words, it takes at least $2S_{tot}$ reaction events for $\mathbf{X}$ to transition from $\mathbf{x_{t_0}}$ to $\mathbf{x}_\infty$ in any simulation run. Intuitively, if $k_{-1} \ll k_2$, then C tends to be consumed by r_2 rather than r_{-1}, making the size of each $\mathbf{T_i}$ close to the lower bound $2S_{tot}$. On the other hand, if $k_{-1} \gg k_2$, then C is more likely to be consumed by r_{-1}, and in consequence each $|\mathbf{T_i}|$ is very likely much greater than $2S_{tot}$, making the computational cost of simulations significantly higher.

Our new PPTA approach minimizes the number of reaction events that fire through the passage of each $\mathbf{x}^{(i)}(t)$ to $\mathbf{x}_\infty$ by preventing each $\mathbf{x}^{(i)}(t)$ from revisiting the same state. Thus, it guarantees that $\forall i.\ |\mathbf{T_i}| = 2S_{tot}$. This is achieved by eliminating r_{-1} and approximating transitions of each $\mathbf{x}^{(i)}(t)$ using only complex-formation and production reactions. In other words, the PPTA approximates the passage time of the formation of each C molecule which leads to a production of P, and only keeps track of such instances of formation of C, rather than explicitly also simulating the formation of C molecules that are destined to dissociate into E and S molecules. Therefore, the PPTA can accelerate the stochastic simulations

of Reaction 4.5, especially when $k_{-1} \gg k_2$ where the reduction in each $|\mathbf{T_i}|$ by this new approach is substantial.

Let us first consider the special case where the total molecular count of the enzyme is 1 (i.e., $E_{tot} = 1$), and describe the derivation of the PPTA model. This section then extends this special case to more general cases where the total molecular count of the enzyme is greater than 1 (i.e., $E_{tot} > 1$).

When E_{tot} is 1, the enzyme state for all $t \geq 0$ is defined by $|E|(t) = 1 - |C|(t)$. Also, r_1 is only enabled when E is active (i.e., $|E|(t) = 1$), and r_{-1} and r_2 are only enabled when C is active (i.e., $|C|(t) = 1$). In this case, $\mathbf{X}(t)$ can be seen as a temporal-homogeneous birth-death Markov process $\mathbf{Y}(t)$ with $2S_{tot} + 1$ states as shown in Figure 4.5. Each state $\sigma \in [0, 2S_{tot}]$ of $\mathbf{Y}(t)$ can then be mapped onto a system state $\mathbf{x}_\sigma$ of $\mathbf{X}(t)$ by the relationship:

$$\mathbf{x}_\sigma \equiv \left((\sigma + 1) \bmod 2, S_{tot} - \left\lceil \frac{\sigma}{2} \right\rceil, \sigma \bmod 2, \left\lfloor \frac{\sigma}{2} \right\rfloor \right). \tag{4.18}$$

Thus, if S_{tot} is 100, for example, there are 201 states in this process, and state 11 represents the vector $\mathbf{x_{11}} = (0, 94, 1, 5)$. For all $t > t_0$, the probability that $\mathbf{Y}(t) = \sigma$ given that $\mathbf{Y}(t_0) = 0$ is the same as the probability that $\mathbf{X}(t) = \mathbf{x}_\sigma$ given that $\mathbf{X}(t) = \mathbf{x_{t_0}}$, and with the initial condition $\mathbf{X}(t_0) = \mathbf{x_{t_0}}$, each simulation run of $\mathbf{Y}(t)$ starts in state 0, and eventually ends up in state $2S_{tot}$. Since E is active only in even numbered states in this process, r_1 can fire only in these states except in state $2S_{tot}$. Similarly, C is active only in odd numbered states, so r_{-1}

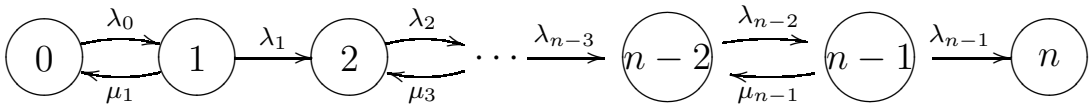


Figure 4.5. The state graph of the birth-death process of Reaction 4.5 when $E_{tot} = 1$. This birth-death process has $n + 1$ states where $n = 2S_{tot}$, and each state σ can be mapped onto a system state of $\mathbf{X}(t)$ by the relationship $\mathbf{x}_\sigma \equiv ((\sigma + 1) \bmod 2, S_{tot} - \lceil \sigma/2 \rceil, \sigma \bmod 2, \lfloor \sigma/2 \rfloor)$. Transition rate λ_σ is $a_1(\mathbf{x}_\sigma)$ if σ is an even number, and $a_2(\mathbf{x}_\sigma)$ if σ is an odd number. Transition rate μ_σ is $a_{-1}(\mathbf{x}_\sigma)$ if σ is an odd number and 0 otherwise.

and r_2 can fire in these states. Thus, let $\mathbf{\Sigma_e}$ be a set of even numbered states $\{2m \mid 0 \leq m \leq S_{tot}\}$, $\mathbf{\Sigma_{e'}}$ be a set of states $\mathbf{\Sigma_e} \setminus \{2S_{tot}\}$, and $\mathbf{\Sigma_o}$ be a set of odd numbered states $\{2m+1 \mid 0 \leq m < S_{tot}\}$. Then, the $\sigma \rightarrow \sigma + 1$ transition rate λ_σ becomes:

$$\lambda_\sigma = \begin{cases} a_1(\mathbf{x}_\sigma) & \text{if } \sigma \in \mathbf{\Sigma_{e'}}, \\ a_2(\mathbf{x}_\sigma) & \text{if } \sigma \in \mathbf{\Sigma_o}, \end{cases} \tag{4.19}$$

whereas the $\sigma \rightarrow \sigma - 1$ transition rate μ_σ becomes:

$$\mu_\sigma = \begin{cases} a_{-1}(\mathbf{x}_\sigma) & \text{if } \sigma \in \mathbf{\Sigma_o}, \\ 0 & \text{if } \sigma \in \mathbf{\Sigma_e}. \end{cases} \tag{4.20}$$

Suppose $\mathbf{Y}(t)$ starts in state σ_0 where $\sigma_0 \in \mathbf{\Sigma_{e'}}$. Then, the average waiting time that $\mathbf{Y}(t)$ spends in states σ_0 and $\sigma_0 + 1$ for each simulation run is equivalent to $t(\sigma_0; \sigma_0 \rightarrow \sigma_0 + 2)$ and $t(\sigma_0 + 1; \sigma_0 \rightarrow \sigma_0 + 2)$, respectively, where $t(\sigma_j; \sigma_i \rightarrow \sigma_k)$ is the mean time that $\mathbf{Y}(t)$ spends in state σ_j in the course of a (first) passage from σ_i to σ_k. In other words, using the variable $t(\sigma_j; \sigma_i \rightarrow \sigma_k)$,

$$t(\sigma_0; \sigma_0 \rightarrow \sigma_0 + 2) \equiv t(\sigma_0; 0 \rightarrow 2S_{tot}), \tag{4.21}$$

$$t(\sigma_0 + 1; \sigma_0 \rightarrow \sigma_0 + 2) \equiv t(\sigma_0 + 1; 0 \rightarrow 2S_{tot}), \tag{4.22}$$

since the transitions: $\sigma_0 \rightarrow \sigma_0 - 1$ and $\sigma_0 + 2 \rightarrow \sigma_0 + 1$ are not allowed in $\mathbf{Y}(t)$. To find out the mean waiting times in states σ_0 and $\sigma_0 + 1$ using the *pedestrian approach* [54], then, variables: $v(\sigma)$ and $v_+(\sigma)$ are defined. The variable $v(\sigma)$ is defined as the average number of visits by $\mathbf{Y}(t)$ to state σ in the course of a first passage from state 0 to state $2S_{tot}$ while $v_+(\sigma)$ is defined as the average number of transitions $\sigma \rightarrow \sigma + 1$ taken by $\mathbf{Y}(t)$ in the course of a first passage from state 0 to state $2S_{tot}$. Using these variables, the probability that $\mathbf{Y}(t)$ moves to state $\sigma_0 + 2$ from state $\sigma_0 + 1$ at the very next jump can be expressed as $v_+(\sigma_0 + 1)/v(\sigma_0 + 1)$. Since this probability can also be expressed as $\lambda_{\sigma_0+1}/(\lambda_{\sigma_0+1} + \mu_{\sigma_0+1})$, and since $v_+(\sigma_0 + 1)$ is 1, the following equation holds:

$$v(\sigma_0 + 1) = \frac{(\lambda_{\sigma_0+1} + \mu_{\sigma_0+1})}{\lambda_{\sigma_0+1}}. \tag{4.23}$$

Because state σ_0+1 can only be visited from state σ_0 in $\mathbf{Y}(t)$, $v_+(\sigma_0)$ must be equal to $v(\sigma_0+1)$. Furthermore, since the transition from state σ_0 to state σ_0-1 cannot occur in $\mathbf{Y}(t)$, $v(\sigma_0)$ must be equivalent to $v(\sigma_0+1)$. Therefore,

$$v(\sigma_0) = \frac{(\lambda_{\sigma_0+1} + \mu_{\sigma_0+1})}{\lambda_{\sigma_0+1}}. \tag{4.24}$$

Now, let $T(\sigma)$ be a random variable which represents the pausing time in state σ in $\mathbf{Y}(t)$. Then, since $\mathbf{Y}(t)$ is a temporally homogeneous birth-death Markov process, $T(\sigma)$ must be a random variable which is necessarily exponentially distributed with parameter $(\lambda_\sigma + \mu_\sigma)$. Then, the mean pausing times in states σ_0 and σ_0+1 can be expressed, respectively, as:

$$\langle T(\sigma_0)\rangle = \int_0^\infty t\lambda_{\sigma_0} \exp(-\lambda_{\sigma_0} t)dt = \frac{1}{\lambda_{\sigma_0}}, \tag{4.25}$$

$$\begin{aligned}\langle T(\sigma_0+1)\rangle &= \int_0^\infty t\left(\lambda_{\sigma_0+1} + \mu_{\sigma_0+1}\right) \exp\left(-\left(\lambda_{\sigma_0+1} + \mu_{\sigma_0+1}\right) t\right) dt \\ &= \frac{1}{\lambda_{\sigma_0+1} + \mu_{\sigma_0+1}}.\end{aligned} \tag{4.26}$$

Since $t(\sigma_j; \sigma_i \rightarrow \sigma_k)$ can be formulated as the product of $\langle T(\sigma_j)\rangle$ and $v(\sigma_j)$, the mean waiting times that $\mathbf{Y}(t)$ spends in states σ_0 and σ_0+1 can be expressed as:

$$t(\sigma_0; 0 \rightarrow 2S_{tot}) = \frac{\lambda_{s_0+1} + \mu_{\sigma_0+1}}{\lambda_{\sigma_0+1}\lambda_{\sigma_0}} = \frac{a_2(\mathbf{x}_{\boldsymbol{\sigma_0+1}}) + a_{-1}(\mathbf{x}_{\boldsymbol{\sigma_0+1}})}{a_2(\mathbf{x}_{\boldsymbol{\sigma_0+1}})a_1(\mathbf{x}_{\boldsymbol{\sigma_0}})}, \tag{4.27}$$

$$t(\sigma_0+1; 0 \rightarrow 2S_{tot}) = \frac{1}{\lambda_{\sigma_0+1}} = \frac{1}{a_2(\mathbf{x}_{\boldsymbol{\sigma_0+1}})}. \tag{4.28}$$

Using this information, $\mathbf{Y}(t)$ can be approximated by creating a temporally homogeneous birth Markov process $\mathbf{Y}'(t)$ with the same state space where the mean waiting time in each state σ is $t(\sigma; 0 \rightarrow 2S_{tot})$ derived from $\mathbf{Y}(t)$. Figure 4.6 shows the state graph of $\mathbf{Y}'(t)$. Since the waiting time in each state σ in $\mathbf{Y}'(t)$ is exponentially distributed, the $\sigma \rightarrow \sigma+1$ transition rate λ'_σ is the reciprocal of $t(\sigma; 0 \rightarrow 2S_{tot})$. Thus, λ'_σ is $a_1(\mathbf{x}_\sigma)a_2(\mathbf{x}_{\sigma+\mathbf{1}})/(a_{-1}(\mathbf{x}_{\sigma+\mathbf{1}}) + a_2(\mathbf{x}_{\sigma+\mathbf{1}}))$ if $\sigma \in \boldsymbol{\Sigma}_\mathbf{o}$ and $a_2(\mathbf{x}_\sigma)$ if $\sigma \in \boldsymbol{\Sigma}_{\mathbf{e}'}$. Therefore, using the PPTA, Reaction 4.5 with E_{tot} being 1 is approximated by a new reaction scheme:

$$E + S \xrightarrow{k_{1'}} C \xrightarrow{k_2} E + P \tag{4.29}$$

where $k_{1'} = k_1k_2/(k_{-1} + k_2)$.

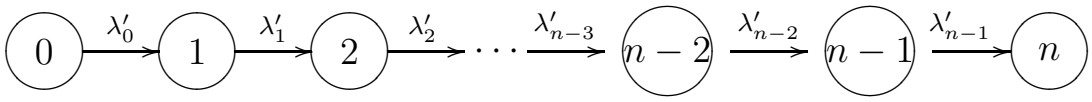


Figure 4.6. The state graph of the pure birth process of the PPTA model when $E_{tot} = 1$. This birth process has the same state space as the birth-death process in Figure 4.5.

When $E_{tot} > 1$, Reaction 4.5 is considered as a set of enzymatic reactions as follows:

$$E_i + S \underset{k_{-1}}{\overset{k_1}{\rightleftarrows}} C_i \xrightarrow{k_2} E_i + P, \quad 1 \leq i \leq E_{tot} \tag{4.30}$$

where $|E_i|(t_0) = 1$ and $|C_i|(t_0) = 0$ for each i. Although simulations of this process are definitely slower than that of $\mathbf{X}(t)$, this transformation itself does not require any approximation since $k_1 |E| |S| \equiv \sum_{i=1}^{E_{tot}} k_1 |E_i| |S|$, $k_{-1} |C| \equiv \sum_{i=1}^{E_{tot}} k_{-1} |C_i|$, and $k_2 |C| \equiv \sum_{i=1}^{E_{tot}} k_2 |C_i|$. Thus, by applying the PPTA to each of the transformed enzymatic reactions, Reaction 4.5 can be approximated by

$$E_i + S \xrightarrow{k_{1'}} C_i \xrightarrow{k_2} E_i + P, \quad 1 \leq i \leq E_{tot}, \tag{4.31}$$

which can now be represented using Reaction 4.29. This implies that the accuracy of the PPTA for the $E_{tot} > 1$ case is based on that of the PPTA of the $E_{tot} = 1$ case, and that the PPTA model provides the most accurate results if $E_{tot} = 1$.

The two parameters in a PPTA model: $k_{1'}$ and k_2 can be derived from K_M, and V_{max} as follows:

$$k_{1'} = \frac{V_{max}}{K_M e_{tot}} \quad \text{and} \quad k_2 = \frac{V_{max}}{e_{tot}}. \tag{4.32}$$

Unlike the parameters k_1 and k_{-1}, the parameters K_M and V_{max} can actually be measured experimentally. Thus, a PPTA model can be constructed and simulated even when full knowledge of the underlying enzymatic reaction is not available and the enzymatic reaction cannot be analyzed quantitatively at that level of detail. This is also true for a QSSA model as its MM form only requires K_M and V_{max} parameters; however, since a PPTA model does not assume that the intermediate

species is in quasi-steady state, a PPTA model may perform better than a QSSA model in terms of accuracy, especially in the pre-steady state phase.

Figure 4.7 shows a graphical representation of the PPTA model of the competitive enzymatic reaction shown in Figure 4.3 to illustrate the application of the PPTA. By applying the extended form of the PPTA to this competitive enzymatic reaction, the reactions to break up complexes C_1 and C_2 are removed and the two complex formation reactions are approximated.

Figure 4.8 shows the algorithm to perform the PPTA on bimolecular enzymatic reactions. First, Algorithm 4.4.1 considers each species, s, as a potential enzyme (line 1). Each species is checked using Algorithm 4.4.2, and if Algorithm 4.4.2 terminates successfully (i.e., returns a nonempty set of configurations, **C**), then Algorithm 4.4.3 is called to apply the transformation to the REB model (lines 2-3). Algorithm 4.4.1 ends by returning a new model M (line 5). Algorithm 4.4.2 first checks to see if s is an interesting species or does not occur as a reactant in any

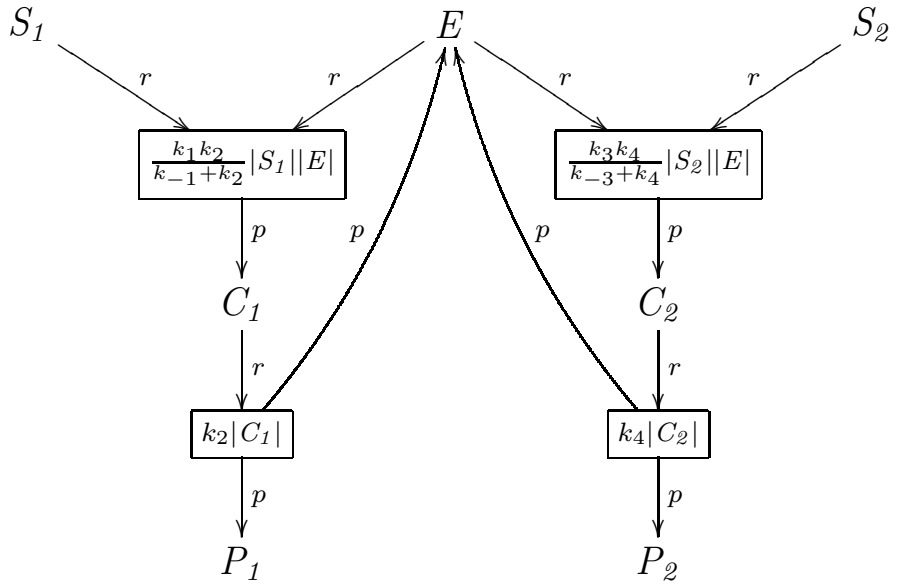


Figure 4.7. Production-passage-time approximation of a competitive enzymatic reaction.

Algorithm 4.4.1 *PPTA for bimolecular enzymatic reactions*
Model PPTAMethod(Model M)

1: ***for all*** $s \in \mathbf{S}$ ***do***
2: $\quad \mathbf{C} \leftarrow PPTAConditionSatisfied(M, s)$
3: $\quad$ ***if*** $\mathbf{C} \neq \emptyset$ ***then*** $M \leftarrow PPTATransform(M, s, \mathbf{C})$
4: ***end for***
5: ***return*** M

Algorithm 4.4.2 *Check the pattern for PPTA*
Configs PPTAConditionSatisfied(Model M, Species s)

1: $\mathbf{C} \leftarrow \emptyset$
2: ***if*** $(s \in \mathbf{S_i}) \vee (|\mathbf{R_s^r}| = 0)$ ***then return*** $\emptyset$
3: ***for all*** $r \in \mathbf{R_s^r}$ ***do***
4: $\quad$ ***if*** $(r \notin \mathbf{R_{rev}}) \vee (|\mathbf{S_r^r}| \neq 2) \vee (\mathbf{K}(r) \neq$ "$k_1 |s| |s_1| - k_{-1} |s_c|$ ") ***then return*** $\emptyset$
5: $\quad$ ***if*** $(|s_c|_0 \neq 0) \vee (s_c \in \mathbf{S_i})$ ***then return*** $\emptyset$
6: $\quad$ ***if*** $(|\mathbf{R_{s_c}^r}| \neq 1) \vee (|\mathbf{R_{s_c}^m}| \neq 0) \vee (\mathbf{R_{s_c}^p}| \neq 1)$ ***then return*** $\emptyset$
7: $\quad \{r_p\} \leftarrow \mathbf{R_{s_c}^r}$
8: $\quad$ ***if*** $(r_p \in \mathbf{R_{rev}}) \vee (|\mathbf{S_{r_p}^r}| \neq 1) \vee (|\mathbf{S_{r_p}^m}| \neq 0)$ ***then return*** $\emptyset$
9: $\quad$ ***if*** $(s \notin \mathbf{S_{r_2}^p}) \vee (|\mathbf{S_{r_p}^p}| \notin \{1, 2\}) \vee (\mathbf{K}(r_p) \neq$ "$k_2 |s_c|$ ") ***then return*** $\emptyset$
10: $\quad \mathbf{C} \leftarrow \mathbf{C} \cup \{(s_1, k_1 k_2/(k_{-1} + k_2), r)\}$
11: ***end for***
12: ***return*** $\mathbf{C}$

Algorithm 4.4.3 *Perform the PPTA model transformation*
Model PPTATransform(Model M, Species s, Configs $\mathbf{C}$*)*

1: ***for all*** $(s_1, k_{1'}, r) \in \mathbf{C}$ ***do***
2: $\quad \mathbf{R_{rev}} \leftarrow \mathbf{R_{rev}} \setminus \{r\}$
3: $\quad \mathbf{K}(r) \leftarrow k_{1'} |s| |s_1|$
4: ***end for***
5: ***return*** M

Figure 4.8. Algorithms to perform production-passage-time approximation for bimolecular enzymatic reactions.

reaction; then s is not considered further (line 2). Otherwise, each reaction, r, in which s is a reactant is considered in turn. If r is not reversible, does not have two reactants, or does not have a rate law of the right form (i.e., $k_1 |s| |s_1| - k_{-1} |s_c|$), then again s is not considered further (line 4). Reaction r combines s and s_1 into a complex s_c. If the initial molecule count of this complex is not 0, s_c is an interesting species, s_c does not occur as a reactant or product in exactly one reaction, or occurs

as a modifier in any, then again this approximation is terminated for s (lines 5-6). The reaction r_p converts s_c into a product and releases the enzyme s. If this reaction is reversible, does not have exactly one reactant and no modifiers, does not have s as a product, has more than two products, or does not have a rate law of the form, $k_2 |s_c|$, then s is not considered further (lines 7-9). For each reaction, a configuration is formed that includes the substrate s_1, complex s_c, constant $k_1 k_2/(k_{-1} + k_2)$, complex forming reaction r, and product forming reaction r_p (line 10). After the successful loop of complex formation reaction, it returns the set of configurations $\mathbf{C}$ (line 12). Algorithm 4.4.3 loops through the set of configurations to modify the complex formation reactions. For each configuration $(s_1, s_c, k_{1'}, r)$, it removes reaction r from the set of reversible reactions, and changes the kinetic law of r according to the PPTA (lines 1-4). It then returns the new model M (line 5).

4.5 Operator Site Reduction

REB models of genetic networks generally include multiple operator sites which transcription factors may occupy. It is often the case that the rates at which transcription factors bind and unbind to these operator sites are rapid with respect to the rate of *open complex formation* (i.e., initiation of transcription). It is also typically the case that the number of operator sites is much smaller than the number of *RNA polymerase* (RNAP) and transcription factor molecules. Therefore, a method similar to the QSSA and the rapid equilibrium approximation called *operator site reduction* can be used to systematically merge reactions and remove operator sites and their complexes from REB models. Note that this method may also be applicable to other molecular scaffolding systems such as those found in signal transduction networks.

The first step in this transformation is to identify operators within the REB model. This is done by assuming that an operator is a species small in number that is neither produced nor degraded. Suppose our algorithm has identified an operator O, and there are $N + 1$ configurations in which transcription factors and RNAP can bind to it. Let O_i, K_i, and X_i with $i \in [1, N]$, be the i-th bound complex

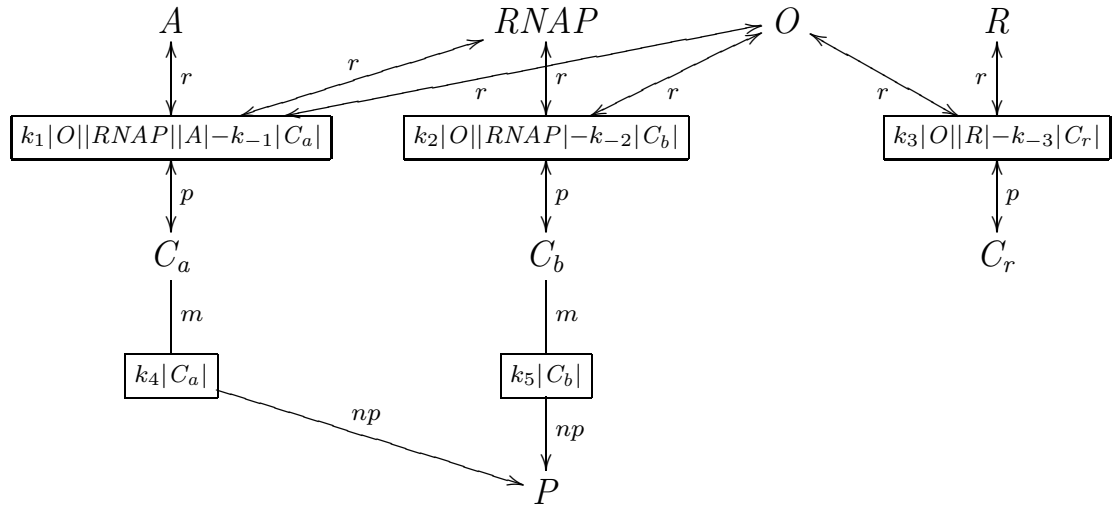
of the operator O, the equilibrium constant for forming this configuration—which is the ratio of the forward rate constant and the backward rate constant—and the product of the states of the substrates for each component of the complex in this configuration, respectively. Let O_0 be the operator in free form (i.e., not bound to anything). Let C_i with $i \in [0, N]$ be each of the operator configurations. Then, assuming rapid equilibrium, the probability of this operator being in each configuration is:

$$Pr(C_i) = \begin{cases} \frac{1}{Z} & \text{if } i = 0 \\ \frac{K_i \cdot X_i}{Z} & \text{if } 1 \leq i \leq N \end{cases}$$

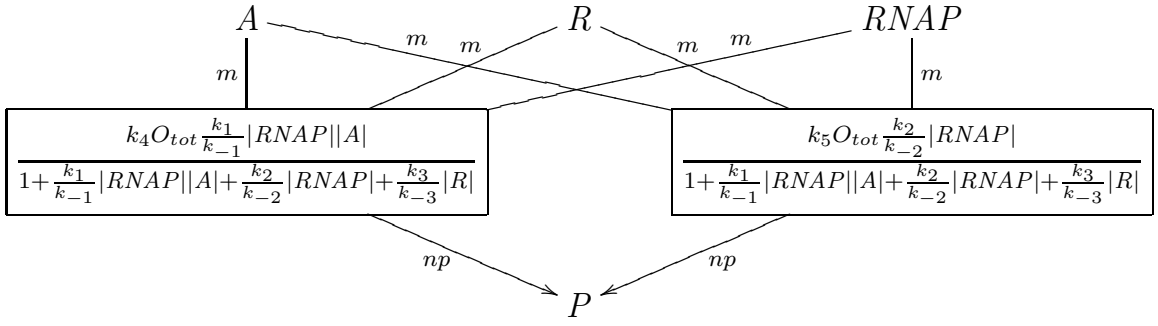
where $Z = 1 + \sum_{j=1}^{N} K_j \cdot X_j$. This probability is the same as the equilibrium statistical thermodynamic model when $K_i = exp(\Delta G_i / RT)$ where ΔG_i is the relative free energies for the i-th configuration, R is the gas constant, and T is the absolute temperature [7]. Assuming that $O_{tot} = |O_0|_0$, then $|O_i| = Pr(C_i)O_{tot}$ is the fraction of operators in the i-th configuration.

Figure 4.9(a) shows the graphical representation of a detailed REB model which describes transcriptional gene regulation to produce protein P based on the configurations of operator site O bindings. The top three reversible reactions involve the binding of $RNAP$, an activator A, and repressor R to O while the bottom two irreversible reactions result in the production of n molecules of the protein P. In this example, there are 4 configurations of the operator, namely, O, C_a, C_b, and C_r. This network has eight species and eight irreversible reactions. Assuming that the operator-binding and unbinding rates are much faster than those of open complex formation, our method can apply operator site reduction. Figure 4.9(b) is the result of applying this abstraction method to Figure 4.9(a). The result has only three species and two reactions. The transformed model represents the probability of O being in a configuration that results in production of P instead of modeling every binding and unbinding of transcription factors and $RNAP$ to the promoter precisely.

The algorithms shown in Figure 4.10 implement operator site reduction. First, Algorithm 4.5.1 considers each species, s, as a potential operator site. Each species



(a)



(b)

Figure 4.9. Operator site reduction: (a) original model and (b) abstracted model.

Algorithm 4.5.1 *Operator site reduction*
Model OpSiteReduction(Model M)

1: ***for all*** $s \in \mathbf{S}$ ***do***
2: $\quad \mathbf{C} \leftarrow OpSiteConditionSatisfied(M, s)$
3: $\quad$ ***if*** $\mathbf{C} \neq \emptyset$ ***then*** $M \leftarrow OpSiteTransform(M, s, \mathbf{C})$
4: ***end for***
5: ***return*** M

Algorithm 4.5.2 *Check the conditions for operator site reduction*
Configs OpSiteConditionSatisfied(Model M, Species s)

1: $\mathbf{C} \leftarrow \emptyset$
2: ***if*** $|s|_0 > maxOperatorThreshold$ ***then return*** $\emptyset$
3: ***if*** $(s \in \mathbf{S_i}) \vee (|\mathbf{R_s^p}| \neq 0)$ ***then return*** $\emptyset$
4: ***for all*** $r_1 \in \mathbf{R_s^r}$ ***do***
5: $\quad$ ***if*** $(r_1 \notin \mathbf{R_{rev}}) \vee (|\mathbf{S_{r_1}^r}| < 2) \vee (|\mathbf{S_{r_1}^p}| \neq 1)$ ***then return*** $\emptyset$
6: $\quad$ ***if*** $(\mathbf{K}(r_1) \neq$ "$k_f \prod_{s' \in \mathbf{S_{r_1}^r}} |s'|^{\mathbf{E}(s', r_1)} - k_r |s_c|$"$)$ ***then return*** $\emptyset$
7: $\quad$ ***if*** $(s_c \in \mathbf{S_i}) \vee (|\mathbf{R_{s_c}^p}| \neq 1) \vee (|\mathbf{R_{s_c}^r}| \neq 0)$ ***then return*** $\emptyset$
8: $\quad$ ***for all*** $r_2 \in \mathbf{R_{s_c}^m}$ ***do***
9: $\quad\quad$ ***if*** $(|\mathbf{S_{r_2}^r}| \neq 0) \vee (|\mathbf{S_{r_2}^m}| \neq 1) \vee (|\mathbf{S_{r_2}^p}| \neq 1) \vee (\mathbf{K}(r_2) \neq k_2 |s_c|)$ ***then return*** $\emptyset$
10: $\quad$ ***end for***
11: $\quad e \leftarrow (k_f / k_r) \prod_{s' \in (\mathbf{S_{r_1}^r} \setminus \{s\})} |s'|^{\mathbf{E}(s', r_1)}$
12: $\quad \mathbf{C} \leftarrow \mathbf{C} \cup \{(s_c, e, r_1)\}$
13: ***end for***
14: ***return*** $\mathbf{C}$

Algorithm 4.5.3 *Perform transformation for operator site reduction*
Model OpSiteTransform(Model M, Species s, Configs $\mathbf{C}$*)*

1: *Exp* $Z \leftarrow 1$
2: $\mathbf{L_1} \leftarrow \emptyset$, $\mathbf{L_2} \leftarrow \emptyset$
3: ***for all*** $(s_c, e, r_1) \in \mathbf{C}$ ***do***
4: $\quad Z \leftarrow Z + e$
5: $\quad \mathbf{L_1} \leftarrow \mathbf{L_1} \cup \mathbf{S_{r_1}^r}$, $\mathbf{L_2} \leftarrow \mathbf{L_2} \cup \{r_1\}$
6: ***end for***
7: ***for all*** $(s_c, e, r_1) \in \mathbf{C}$ ***do***
8: $\quad$ ***for all*** $r_2 \in \mathbf{R_{s_c}^m}$ *with* $\mathbf{K}(r_2) = k_2 |s_c|$ ***do***
9: $\quad\quad \forall m \in \mathbf{L_1}.\ M \leftarrow addModifier(M, m, r_2, \max_{r \in \mathbf{L_2}} (\mathbf{E}(m, r)))$
10: $\quad\quad \mathbf{K}(r_2) \leftarrow (k_2 * |s|_0 * e)/Z$
11: $\quad$ ***end for***
12: ***end for***
13: $\forall (s_c, e, r_1) \in \mathbf{C}.\ M \leftarrow removeReaction(M, r_1)$
14: $\forall (s_c, e, r_1) \in \mathbf{C}.\ M \leftarrow removeSpecies(M, s_c)$
15: $M \leftarrow removeSpecies(M, s)$
16: ***return*** M

Figure 4.10. Algorithms for operator site reduction.

is checked using Algorithm 4.5.2. First, it is assumed that the molecule count of operator sites is small, so if s has an initial molecule count greater than a given threshold, then it is assumed not to be an operator site (line 2). Next, if s is an interesting species or occurs as a product in any reaction, then s is not considered further (line 3). Otherwise, each reaction, r_1, in which s is a reactant is considered in turn. If r_1 is not reversible, has less than two reactants, does not have exactly one product, or does not have a rate law of the right form, then again s is not considered further (lines 5-6). Each reaction, r_1, combines the potential operator site, s, with RNAP and/or transcription factors forming a complex, s_c. If s_c is an interesting species, s_c does not occur as a product in exactly one reaction, or occurs as a reactant in any, then again this approximation is terminated for s (line 7). The species s_c may appear as a modifier in any number of reactions that result in the transcription and translation of proteins. Each of these reactions, r_2, is checked that it has no reactants, only one modifier, only one product, and a rate law of the right form (lines 8-10). For each complex, s_c, a configuration is formed that includes the complex s_c, an equilibrium expression for this configuration, and the complex forming reaction r_1 (lines 11-12). If Algorithm 4.5.2 terminates successfully, then Algorithm 4.5.3 is called to apply the transformation to the REB model. This algorithm is very similar to the one for the QSSA. Again, it loops through the set of configurations to form an expression that is used in the denominator in each new rate law as well as forming lists of all the transcription factors that bind to the operator site s and all the binding reactions (lines 1-6). Next, it considers each configuration, (s_c, e, r_1). For each reaction r_2 in which s_c appears as a modifier, it adds all the transcription factors as modifiers and creates a new rate law for r_2 (lines 8-12). It then removes all the reactions r_1 and species s_c in the configuration from the model (lines 13 and 14). Finally, at the end, this algorithm removes the operator site, s (line 15).

4.6 Dimerization Reduction

Dimerization is another type of reaction that often involves only regulatory molecules and could thus frequently proceed very rapidly compared to the rate of transcription initiation. Therefore, it might also be useful to abstract away these reactions whenever possible by using a version of rapid-equilibrium constraints. Let us consider a reversible reaction that forms a dimer s_d from two molecules of species s_m as shown in Figure 4.11. Then, the dimerization reduction method is used to remove this fast reaction and to express dimer and monomer forms of species in terms of a state variable $|s_t|$ which represents the total number/concentration of the monomer molecules. Thus, the relationship of $|s_t|$, $|s_m|$, and $|s_d|$ can be specified as:

$$|s_t| \;=\; |s_m| + 2\,|s_d|\,. \tag{4.33}$$

In a deterministic process, the dimerization reduction can be achieved by assuming that the forward and backward reactions are in equilibrium and the net change of $|s_m|$ and $|s_d|$ becomes zero. Thus, by assuming that $|s_m|$ and $|s_d|$ are in steady state, the equation:

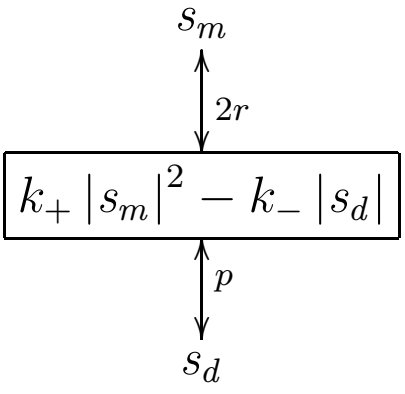


Figure 4.11. Dimerization reaction which forms a dimer s_d from two molecules of species s_m. Constants k_+ and k_- are the forward rate constant and the backward rate constant of the dimerization reaction, respectively.

$$|s_d| \;=\; K_e|s_m|^2 \tag{4.34}$$

where K_e is the equilibrium constant for dimerization (i.e., $K_e = k_+/k_-$) can be established. Using Equations 4.33 and 4.34, the following equation can be derived:

$$K_e|s_t|^2 - (4K_e\,|s_t| + 1)\,|s_d| + 4K_e|s_d|^2 = 0. \tag{4.35}$$

Solving Equation 4.35, $|s_m|$ and $|s_d|$ can be expressed in terms of $|s_t|$ as follows:

$$|s_m| \;=\; \frac{1}{4K_e}\left(\sqrt{8K_e\,|s_t| + 1} - 1\right), \tag{4.36}$$

$$|s_d| \;=\; \frac{|s_t|}{2} - \frac{1}{8K_e}\left(\sqrt{8K_e\,|s_t| + 1} - 1\right). \tag{4.37}$$

In a stochastic process, a stricter requirement of the SCK demands that the propensity function of the forward reaction be $k_+\,|s_m|\,(|s_m| - 1)$. By assuming that $|s_m|$ is practically always greater than 1, however, this can be commonly approximated as $k_+\,|s_m|^2$ as shown in Figure 4.11. The derivation of the dimerization reduction in a SCK model begins with replacing $|s_m|$ and $|s_d|$ with their means, $\langle|s_m|\rangle$ and $\langle|s_d|\rangle$, respectively. Thus, Equation 4.33 for the conservation relationship can be adapted for $|s_t|$, $\langle|s_m|\rangle$, and $\langle|s_d|\rangle$ becomes:

$$|s_t| = \langle|s_m|\rangle + 2\langle|s_d|\rangle. \tag{4.38}$$

Also, the time derivative equation of $\langle|s_d|\rangle$ is obtained as follows:

$$\frac{d\langle|s_d|\rangle}{dt} = k_+\langle|s_m|^2\rangle - k_-\langle|s_d|\rangle. \tag{4.39}$$

Now, since the time scale of the dimerization reaction is typically much faster than that of other reactions which change the states of s_m and s_d, in the time scale of the slow reactions, $\langle|s_m|\rangle$ and $\langle|s_d|\rangle$ rapidly moves to a stationary state where $\frac{d\langle|s_d|\rangle}{dt}$ becomes zero. Thus, by letting $\langle|s_m|_\infty\rangle$ and $\langle|s_d|_\infty\rangle$ be the stationary states of $\langle|s_m|\rangle$ and $\langle|s_d|\rangle$, respectively, a quadratic equation of $\langle|s_d|\rangle$ can be obtained as follows:

$$K_e|s_t|^2 - (4K_e\,|s_t| + 1)\langle|s_d|_\infty\rangle + 4K_e\langle|s_d|_\infty{}^2\rangle = 0. \tag{4.40}$$

In order to obtain an approximate solution of $\langle |s_d|_\infty \rangle$ from Equation 4.40, mean of ${|s_d|_\infty}^2$ is approximated by the square of $\langle |s_d|_\infty \rangle$ (i.e., $\langle {|s_d|_\infty}^2 \rangle \approx \langle |s_d|_\infty \rangle^2$), which gives

$$K_e|s_t|^2 - (4K_e\,|s_t| + 1)\langle |s_d|_\infty \rangle + 4K_e \langle |s_d|_\infty \rangle^2 = 0. \tag{4.41}$$

This can be justified on the grounds for large $|s_t|$ where the standard deviation of $|s_d|$ is very small compared with its average. Therefore, by solving Equation 4.41, $\langle |s_m|_\infty \rangle$ and $\langle |s_d|_\infty \rangle$ are expressed in terms of $|s_t|$ in the same way as the approximate CCK model.

As an example, consider a REB model including a dimerization reaction of species A shown in Figure 4.12(a). This species can only effectively degrade in the monomer form (reaction r_1), but it is transcriptionally active (reactions r_3 and r_4)

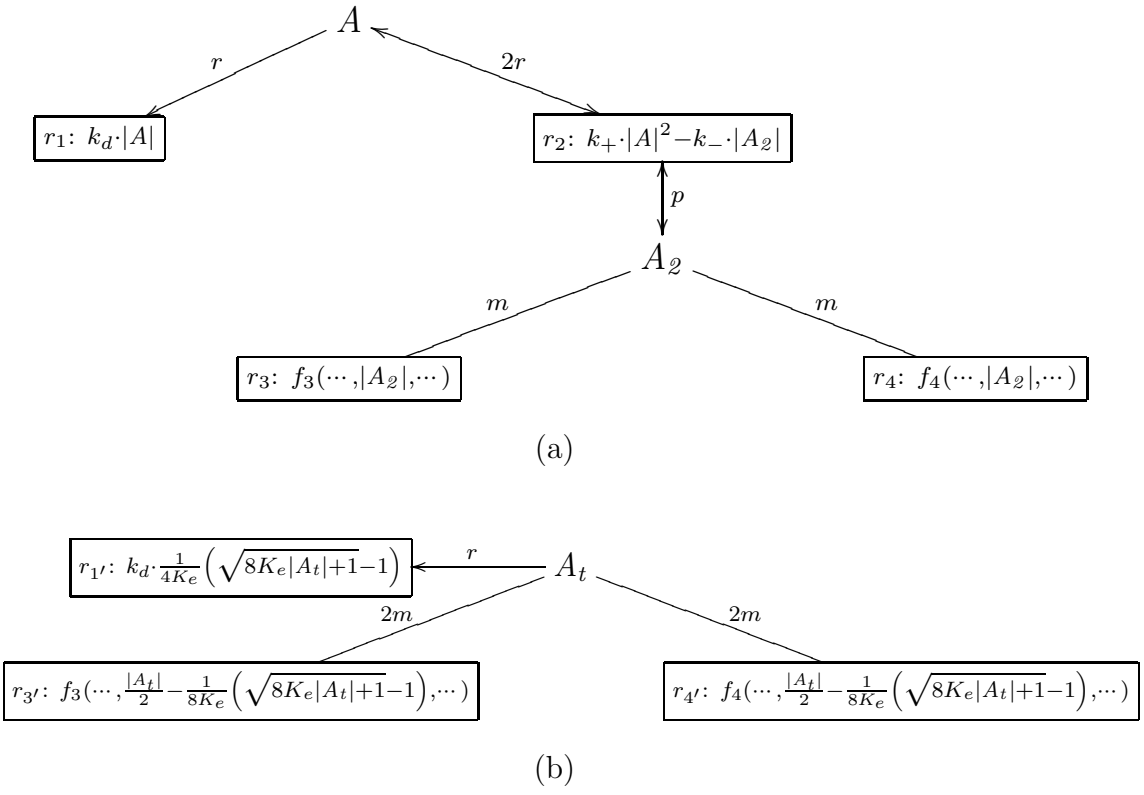


Figure 4.12. Dimerization reduction: (a) original model, and (b) abstracted model.

only as $|A_2|$, its dimer form, (reaction r_2). Using Equations 4.36 and 4.37, the reactions r_1, r_3, and r_4 can be transformed to $r_{1'}$, $r_{3'}$, and $r_{4'}$, respectively, with kinetic laws that are now all expressed in terms of $|A_t|$, the total amount of A, as shown in Figure 4.12(b). Note that the dimerization reaction r_2 is eliminated completely.

The algorithm to perform dimerization reduction is shown in Figure 4.13. First, Algorithm 4.6.1 is used to identify a dimerization reaction. It checks each reversible reaction, r, using Algorithm 4.6.2. A dimerization reaction must include exactly one reactant, one product, and no modifiers (line 1). It must also have a rate law of the right form (line 2). The dimerization reduction also requires that the monomer is never used as a modifier, and that there is only one reaction (this one) which consumes two molecules of the monomer and produces one molecule of the dimer (lines 3-5). If these conditions are met, a record is made of the monomer species s_m, dimer species s_d, and equilibrium constant k_+/k_- (line 6). The transformation is performed by Algorithm 4.6.3. First, a new species s_t is introduced into the model with an initial concentration $|s_m|_0 + 2|s_d|_0$ (lines 1-2). Next, s_m is replaced by s_t in each reaction in which s_m is a reactant, and the rate law is updated as described in Equation 4.36 (lines 3-6). The dimer s_d is also replaced with s_t in the reactions in which it appears as a reactant or modifier, and the rate law is updated using Equation 4.37 (lines 7-11). Finally, the species s_m, the species s_d, and reaction r are all removed from the model (lines 12-14).

4.7 Modifier Constant Propagation

In order to increase the understandability of a REB model as well as the efficiency of its temporal behavior analysis, it is essential to remove all *unimportant* species that do not contribute to the dynamics of a system. To systematically inspect and remove species whose states are statically known to stay unchanged in simulation, this section develops an abstraction method called *modifier constant propagation*. Modifier constant propagation traverses a REB model, and finds a species s which is only used as a modifier. It then substitutes a constant $|s|_0$ for

Algorithm 4.6.1 *Dimerization reduction*
Model DimerReduction(Model M)

1: ***for all*** $r \in \mathbf{R_{rev}}$ ***do***
2: $\quad C \leftarrow DimerConditionSatisfied(M, r)$
3: $\quad$ ***if*** $C \neq$ **nil** ***then*** $M \leftarrow DimerTransform(M, r, C)$
4: ***end for***
5: ***return*** M

Algorithm 4.6.2 *Check the conditions for the dimerization reduction*
Record DimerConditionSatisfied(Model M, Reaction r)

1: ***if*** $(|\mathbf{S_r^r}| \neq 1) \vee (|\mathbf{S_r^p}| \neq 1) \vee (|\mathbf{S_r^m}| \neq 0)$ ***then return*** **nil**
2: ***if*** $(\mathbf{K}(r) \neq$ *"*$k_+ |s_m|^2 - k_- |s_d|$*"*$)$ ***then return*** **nil**
3: $\{s_m\} \leftarrow \mathbf{S_r^r}, \{s_d\} \leftarrow \mathbf{S_r^p}$
4: ***if*** $(\mathbf{E}(s_m, r) \neq 2) \vee (\mathbf{E}(r, s_d) \neq 1)$ ***then return*** **nil**
5: ***if*** $(|\mathbf{R_{s_m}^m}| \neq 0) \vee (|\mathbf{R_{s_d}^p}| \neq 1)$ ***then return*** **nil**
6: ***return*** $\langle s_m, s_d, k_+/k_- \rangle$

Algorithm 4.6.3 *Perform the dimerization reduction transformation*
Model DimerTransform(Model M, Reaction r, Record $\langle s_m, s_d, K_e \rangle$*)*

1: $M \leftarrow addSpecies(M, s_t)$
2: $|s_t|_0 \leftarrow |s_m|_0 + 2|s_d|_0$
3: ***for all*** $r' \in \mathbf{R_{s_m}^r}$ ***do***
4: $\quad M \leftarrow addReactant(M, s_t, r', \mathbf{E}(s_m, r'))$
5: $\quad$ *replace* $|s_m|$ *with* $\frac{1}{4K_e}\left(\sqrt{8K_e|s_t|+1}-1\right)$ *in* $\mathbf{K}(r')$
6: ***end for***
7: ***for all*** $\forall r' \in (\mathbf{R_{s_d}^r} \cup \mathbf{R_{s_d}^m})$ ***do***
8: $\quad$ ***if*** $r' \in \mathbf{R_{s_d}^r}$ ***then*** $M \leftarrow addReactant(M, s_t, r', 2\mathbf{E}(s_d, r'))$
9: $\quad$ ***if*** $r' \in \mathbf{R_{s_d}^m}$ ***then*** $M \leftarrow addModifier(M, s_t, r', 2\mathbf{E}(s_d, r'))$
10: $\quad$ *replace* $|s_d|$ *with* $\frac{|s_t|}{2} - \frac{1}{8K_e}\left(\sqrt{8K_e|s_t|+1}-1\right)$ *in* $\mathbf{K}(r')$
11: ***end for***
12: $M \leftarrow removeSpecies(M, s_m)$
13: $M \leftarrow removeSpecies(M, s_d)$
14: $M \leftarrow removeReaction(M, r)$
15: ***return*** M

Figure 4.13. Algorithm to perform dimerization reduction.

$|s|$ in kinetic law expressions of the reactions that use s as a modifier. Therefore, since $|s|$ is no longer used to influence any kinetic laws, species s is safely removed from a REB model by this method.

Figure 4.14 illustrates an application of modifier constant propagation. In a REB model shown in Figure 4.14(a), species s_1 is used as a modifier in reactions r_1 and r_2. And unlike the other three species in this REB model, species s_1 is not used as a reactant or a product. Thus, by applying modifier constant propagation, this REB model can be transformed to a REB model shown in Figure 4.14(b).

Even when modifier constant propagation cannot reduce the structure of a detailed REB model, it may be used in combination with other abstraction methods to further reduce the complex of an abstracted REB model. For example, as

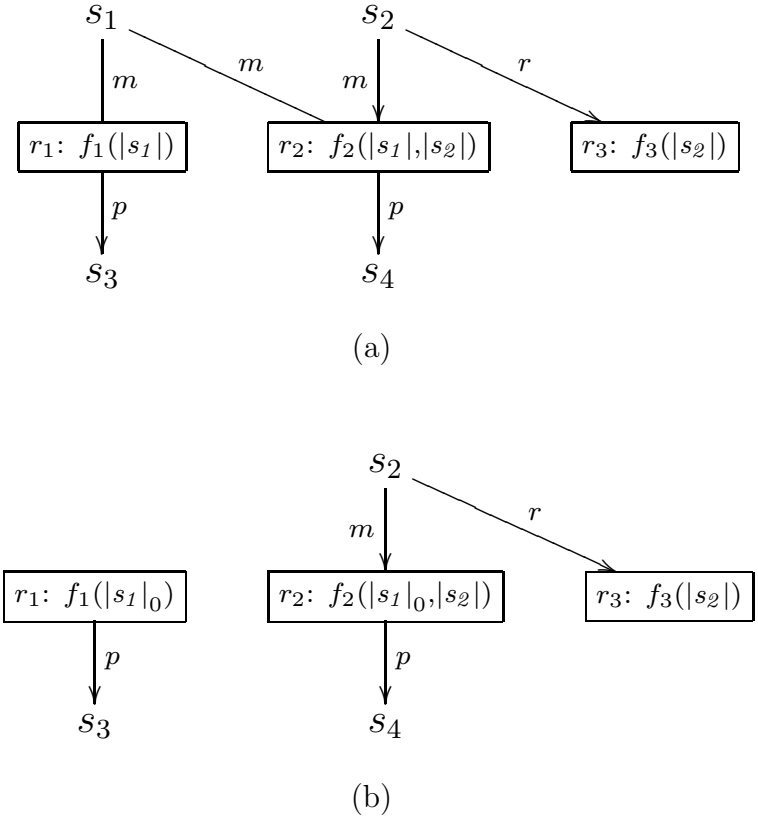


Figure 4.14. Modifier constant propagation: (a) original model with s_1 being used only as a modifier and (b) abstracted model.

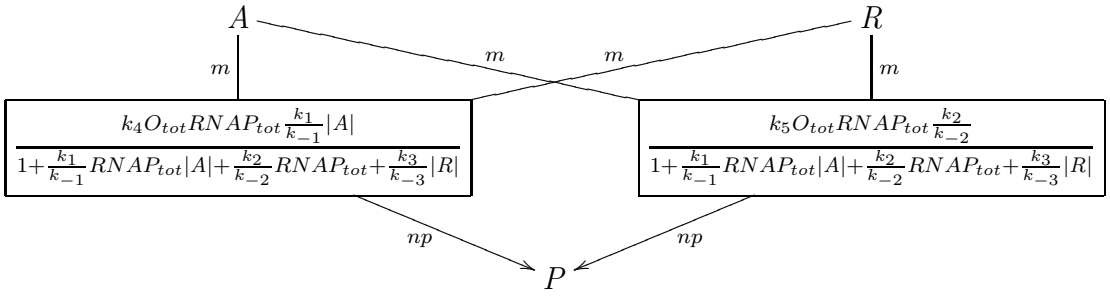


Figure 4.15. A REB model after applying modifier constant propagation to a REB model shown in Figure 4.9(b).

illustrated in a REB model in Figure 4.9(b), after applying operator site reduction, it is often the case that $RNAP$ is only used as a modifier. Thus, by applying modifier constant propagation, $|RNAP|$ can be replaced with a constant $RNAP_{tot}$ where $RNAP_{tot} = |RNAP|_0$. Therefore, as shown in Figure 4.15, a REB model in Figure 4.9(b) can be reduced to three species and two reactions as a result of modifier constant propagation.

The algorithm shown in Figure 4.16 implements modifier constant propagation. Algorithm 4.7.1 first iterates on every species s such that species s is not used as a reactant nor a product in any reaction (line 1). For each s, the kinetic laws for reactions that use s as a modifier to replace $|s|$ with a constant whose value is set as $|s|_0$, and species s is removed from the model (lines 2-3).

Algorithm 4.7.1 *Modifier constant propagation*
Model ModConstProp(Model M)
1: ***for all*** $s \in \{s' | (\mathbf{R}^{\mathbf{r}}_{\mathbf{s}'} = \emptyset) \wedge (\mathbf{R}^{\mathbf{p}}_{\mathbf{s}'} = \emptyset)\}$ ***do***
2: $\quad \forall r \in \mathbf{R}^{\mathbf{m}}_{\mathbf{s}}$. *replace* $|s|$ *with a constant whose value is* $|s|_0$ *in* $\mathbf{K}(r)$
3: $\quad M \leftarrow removeSpecies(M, s)$
4: ***end for***
5: ***return*** M

Figure 4.16. Algorithm for modifier constant propagation.

4.8 Similar Reaction Combination

A REB model may contain multiple reactions whose structures are very similar. Thus, combining such reactions using another abstraction method called *similar reaction combination* can improve the complexity of a REB model by reducing the number of reactions in a REB model. It can also result in a reduction of the computational costs for evaluating kinetic laws by reducing redundant kinetic law expressions. In the context of genetic regulatory networks, an abstracted REB model of transcriptional gene regulation often has protein synthesis mechanisms at a basal rate and enhanced or reduced rates due to transcription factors binding to operator sites. These mechanisms can be represented in structurally similar reactions whose kinetic laws typically contain redundant expressions. Thus, with this method, such protein synthesis mechanisms can be combined into one reaction with a computationally much less expensive kinetic law expression.

Similar reaction combination transforms a REB model by first searching for structurally similar reactions and replaces them with one reaction. Here, reactions r_1 and r_2 are defined to be structurally similar if reactions r_1 and r_2 have the same reactants, products, and modifiers with the same stoichiometries. In other words, if the condition:

$$\forall s \in \mathbf{S}.\ \mathbf{E}(s, r_1) = \mathbf{E}(s, r_2) \wedge \mathbf{E}(r_1, s) = \mathbf{E}(r_2, s) \tag{4.42}$$

is satisfied, then reactions r_1 and r_2 are structurally similar. An implication of Condition 4.42 being satisfied is that firings of both reactions r_1 and reaction r_2 are guaranteed to result in the same state transition of a REB model. Thus, these reactions can be combined to introduce a new reaction r_c such that

$$\forall s \in \mathbf{S}.\ \mathbf{E}(s, r_c) = \mathbf{E}(s, r_1) \wedge \mathbf{E}(r_c, s) = \mathbf{E}(r_1, s) \tag{4.43}$$

$$\mathbf{K}(r_c) = \mathbf{K}(r_1) + \mathbf{K}(r_2). \tag{4.44}$$

With Conditions 4.42 and 4.43, it is implied that

$$\forall s \in \mathbf{S}.\ \mathbf{E}(r_c, s) - \mathbf{E}(s, r_c) = \mathbf{E}(r_1, s) - \mathbf{E}(s, r_1) = \mathbf{E}(r_2, s) - \mathbf{E}(s, r_2). \tag{4.45}$$

Thus, with Conditions 4.44 and 4.45, the ODE model of the combined reactions is identical to the original one. Hence, the similar reaction combination method can be used without making any approximation in the continuous-deterministic analysis case. In the case of SCK analysis via the SSA, suppose reactions r_1 and r_2 are structurally similar. Then, from Condition 4.44, the probability that either reaction r_1 or reaction r_2 is chosen to be the next reaction to fire becomes:

$$Prob(r_1 \text{ or } r_2) = \frac{\mathbf{K}(r_1)}{\sum_{r\in\mathbf{R}}\mathbf{K}(r)} + \frac{\mathbf{K}(r_2)}{\sum_{r\in\mathbf{R}}\mathbf{K}(r)} = \frac{\mathbf{K}(r_c)}{\sum_{r\in\mathbf{R}}\mathbf{K}(r)}. \tag{4.46}$$

Thus, the probability of firing an event of the newly introduced reaction r_c is identical to the probability of firing an event of either reaction r_1 or r_2. Similarly, from Condition 4.44, the computation of the next reaction time τ in the direct method of the SSA does not change before and after the reaction combination as the sum of the propensity functions does not change. Furthermore, from Condition 4.45, the state transitions via the combined reaction r_c are the same as that of reactions r_1 and r_2. Therefore, this method itself does not make any approximation for the SCK analysis as well.

To illustrate an application of similar reaction combination, Figure 4.17 shows a REB model that is abstracted from the REB model in Figure 4.15 by using similar

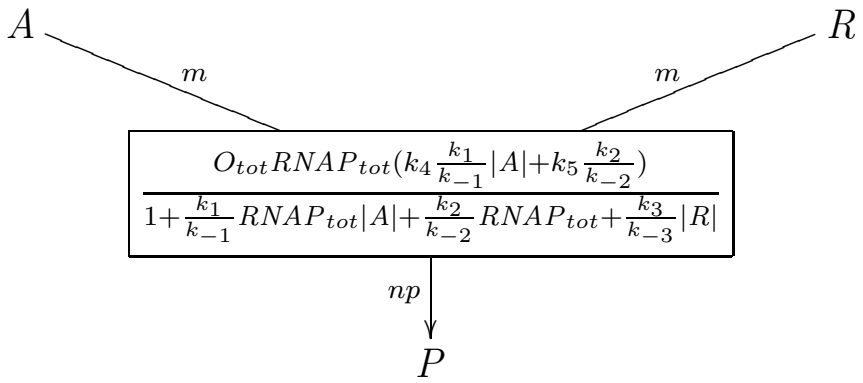


Figure 4.17. A REB model after applying similar reaction combination to a REB model shown in Figure 4.15.

reaction combination. In this reduced REB model, the two reactions to produce n molecules of protein P are combined into one reaction. Since the kinetic laws of the two reactions have the same denominator, the combined reaction is able to simplify its kinetic law, making its evaluation faster than that of the two original kinetic laws.

Figure 4.18 shows the algorithms to implement similar reaction combination for irreversible reactions. Note that, though this algorithm only targets irreversible reactions, it does not present a limitation in its applicability by first transforming all reversible reactions to irreversible reactions using, for example, Algorithm 4.2.1. Algorithm 4.8.1 loops over irreversible reactions. For each irreversible reaction r_1, if r_1 has not been combined by another reaction, then each irreversible reaction r_2 that is different from r_1 and has not already been combined into another reaction is examined to test if Condition 4.42 between reactions r_1 and r_2 is satisfied (lines 1-4). If reactions r_1 and r_2 can be combined, then the kinetic law of r_1 is changed by Algorithm 4.8.2 to reflect the kinetic law of the combined reaction, and r_2 is put into a set of combined reactions (lines 6-7). After the loop, all the substituted reactions are removed from the model and the new model is returned (lines 12-13). Algorithm 4.8.2 examines the structure of the kinetic law expressions e_1 and e_2, and attempts to simplify the expression of the sum of e_1 and e_2. If e_1 and e_2 are both division expressions and they have the common denominator e_d, then their numerators are combined using Algorithm 4.8.2 to form e_n and the expression e_n/e_d is returned (lines 1-3). If e_1 and e_2 are both multiplication expressions and they have the common term e_c, then their other terms are combined using Algorithm 4.8.2 to form e_s and the expression $e_c \times e_s$ is returned (lines 4-6). If the structures of e_1 and e_2 do not satisfy these two cases, then the expression $e_1 + e_2$ is simply returned (lines 7-9).

4.9 Stoichiometry Amplification

One approximation approach to lower the computational costs for stochastic simulation is to advance the system and time faster. This can be achieved by

Algorithm 4.8.1 *Similar reaction combination*
Model SimRxnComb(Model M)

1: $\mathbf{L} \leftarrow \emptyset$
2: ***for all*** $r_1 \in \mathbf{R} \setminus \mathbf{R_{rev}}$ ***do***
3: ***if*** $r_1 \notin \mathbf{L}$ ***then***
4: ***for all*** $r_2 \in (\mathbf{R} \setminus \mathbf{R_{rev}}) \setminus \{r_1\}$ ***do***
5: ***if*** $(r_2 \notin \mathbf{L}) \wedge (\forall s \in \mathbf{S}.\ \mathbf{E}(s, r_1) = \mathbf{E}(s, r_2) \wedge \mathbf{E}(r_1, s) = \mathbf{E}(r_2, s))$ ***then***
6: $\mathbf{K}(r_1) \leftarrow CreateCombExpression(\mathbf{K}(r_1), \mathbf{K}(r_2))$
7: $\mathbf{L} \leftarrow \mathbf{L} \cup \{r_2\}$
8: ***end if***
9: ***end for***
10: ***end if***
11: ***end for***
12: $\forall r \in \mathbf{L}.\ M \leftarrow removeReaction(M, r)$
13: ***return*** M

Algorithm 4.8.2 *Create combined kinetic law expression*
Exp CreateCombExpression(Exp e_1, Exp e_2)

1: ***if*** $\exists\ e_{1n}, e_{2n}, e_d.\ (e_1 =$ "e_{1n} / e_d"$) \wedge (e_2 =$ "e_{2n} / e_d"$)$ ***then***
2: $e_n \leftarrow CreateCombExpression(e_{1n}, e_{2n})$
3: ***return*** e_n / e_d
4: ***else if*** $\exists\ e_{1s}, e_{2s}, e_c.\ (e_1 =$ "$e_c \times e_{1s}$"$) \wedge (e_2 =$ "$e_c \times e_{2s}$"$)$ ***then***
5: $e_s \leftarrow CreateCombExpression(e_{1s}, e_{2s})$
6: ***return*** $e_c \times e_s$
7: ***else***
8: ***return*** $e_1 + e_2$
9: ***end if***

Figure 4.18. Algorithm to perform similar reaction combination for irreversible reactions.

applying an abstraction method called *stoichiometry amplification* which increases $\mathbf{E}(r,s)-\mathbf{E}(s,r)$ for each reaction r and its participant species s while reducing the sum of the values of the propensity functions, $\sum_{r\in\mathbf{R}}\mathbf{K}(r)$ [75]. The stoichiometry amplification with amplification factor of n where n is an integer greater than one transforms a REB model as follows. For each reaction r, $\mathbf{K}(r)$ is reduced by the factor n, and the stoichiometries of each species s in reaction r, $\mathbf{E}(r,s)$ and $\mathbf{E}(s,r)$ are amplified by the factor n. From this definition, this abstraction method does not change the CCK model of a REB model since, for each species s,

$$\sum_{r\in\mathbf{R}}\left(\mathbf{E}(r,s)-\mathbf{E}(s,r)\right)\mathbf{K}(r)=\sum_{r\in\mathbf{R}}\left(n\mathbf{E}(r,s)-n\mathbf{E}(s,r)\right)\frac{\mathbf{K}(r)}{n}. \tag{4.47}$$

In other words, the time derivative of each $|s|$ is identical before and after the application of stoichiometry amplification. Therefore, there is no good reason to apply this abstraction method to a REB model in the case of the CCK analysis.

In the SCK analysis via the SSA, on the other hand, stoichiometry amplification can significantly reduce the computational requirements. To show this, the application of stoichiometry amplification is illustrated in Figure 4.19. Figure 4.19(a) shows a reaction that converts one molecule of species s_1 and two molecules of species s_2 into one molecule of species s_3 with the kinetic law $f(|s_1|,|s_2|)$. By applying the stoichiometry amplification with amplification factor of n, this reaction is abstracted to the reaction shown in Figure 4.19(b) where n molecules of s_1 and $2n$ molecules of s_2 are converted into n molecules of s_3 with the kinetic law $\frac{1}{n}f(|s_1|,|s_2|)$. This implies that it only takes one reaction event for species s_3 in the abstracted model to transition to the state $|s_3|_0+n$ from $|s_3|_0$ while the same transition requires n reaction events in the original model in the SCK analysis via the SSA, provided that $|s_1|_0\geq n$ and $|s_1|_0\geq 2n$. Hence, on average, the application of stoichiometry amplification with amplification factor n can give the stochastic simulation of an abstracted model as much as n-times speedup compared with the simulation of its original model by reducing the number of the kinetic law evaluations n times. Furthermore, the stoichiometry amplification can be used to help reduce the state space of the system [75].

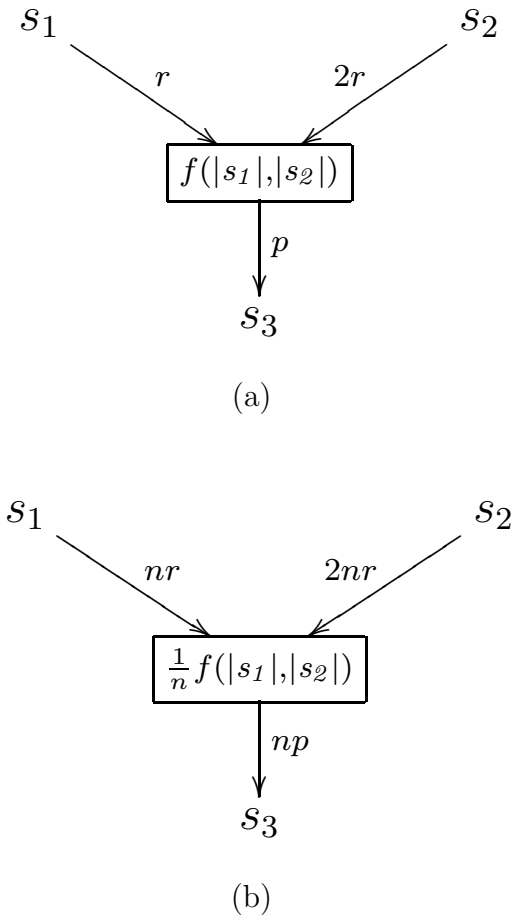


Figure 4.19. Stoichiometry amplification: (a) original model and (b) abstracted model.

For the analysis of the mean temporal behavior, the application of this abstraction method is mathematically justified when the change in the value of each kinetic law via the amplification is not significant. For example, in the reaction model in Figure 4.19(a), suppose the system is in a state where $|s_1| = \sigma_1$, $|s_2| = \sigma_2$, and $|s_3| = \sigma_3$. Then, since the next reaction time is exponentially distributed in the SCK model, by letting $T(s; n_1 \to n_2)$ be the random variable describing the time for species s to transition from state n_1 to state n_2, the average time for species s_3 to transition from σ_3 to $\sigma_3 + n$ can be specified by

$$\langle T(s_{\mathcal{J}}; \sigma_3 \rightarrow \sigma_3 + n) \rangle = \sum_{i=0}^{n-1} \frac{1}{f(\sigma_1 - i, \sigma_2 - 2i)}. \tag{4.48}$$

Here, if the value of $f(\sigma_1, \sigma_2)$ is very close to that of $f(\sigma_1 - i, \sigma_2 - 2i)$ for any i from 1 to $n-1$, $f(\sigma_1, \sigma_2)$ can be safely substituted for $f(\sigma_1 - i, \sigma_2 - 2i)$. Consequently, the average of $T(s_{\mathcal{J}}; \sigma_3 \rightarrow \sigma_3 + n)$ can be well approximated as:

$$\langle T(s_{\mathcal{J}}; \sigma_3 \rightarrow \sigma_3 + n) \rangle \approx \sum_{i=0}^{n-1} \frac{1}{f(\sigma_1, \sigma_2)} = \frac{n}{f(\sigma_1, \sigma_2)}. \tag{4.49}$$

Therefore, using the propensity function that is reciprocal of the approximate $\langle T(s_{\mathcal{J}}; |s_{\mathcal{J}}| \rightarrow |s_{\mathcal{J}}| + n) \rangle$ as obtained via the stoichiometry amplification with amplification factor n, the mean time evolution of the reaction model in Figure 4.19(a) can be well approximated by that in Figure 4.19(b).

The algorithm to apply stoichiometry amplification to all the reactions in a REB model is shown in Figure 4.20. Algorithm 4.9.1 first obtains the amplification factor n (line 1). It then loops over the reactions to transform the REB model. For each reaction r, it reduces the kinetic law expression of reaction r by the factor n, and the stoichiometry of each species in reaction r is amplified by the factor n (lines 2-8). Finally, it returns the updated model (line 9).

Algorithm 4.9.1 *Stoichiometry amplification*
Model StoichAmp(Model M)

```
1: let n be the amplification factor
2: for all r ∈ R do
3:    K(r) ← K(r)/n
4:    for all s ∈ S do
5:       E(r, s) ← nE(r, s)
6:       E(s, r) ← nE(s, r)
7:    end for
8: end for
9: return M
```

Figure 4.20. Algorithm to perform stoichiometry amplification to all the reactions.

4.10 Irrelevant Node Elimination

In a large system, there may be species that do not have significant influence on the species of interest, $\mathbf{S_i}$. This is especially true when a computational model of a biological system is automatically generated from its experimental data, whereby, unlike species in hand-crafted pathway-specific models, some species may appear to be uncoupled. Even when all the species in the original model are coupled, after applying abstractions, a species may no longer influence the species of interest. In such cases, computational performance can be gained by removing such irrelevant species and reactions. *Irrelevant node elimination* performs a reachability analysis on the REB model and detects nodes that do not influence the species in $\mathbf{S_i}$. For example, in Figure 4.21(a), s_6 is the only species in $\mathbf{S_i}$. Therefore, the production

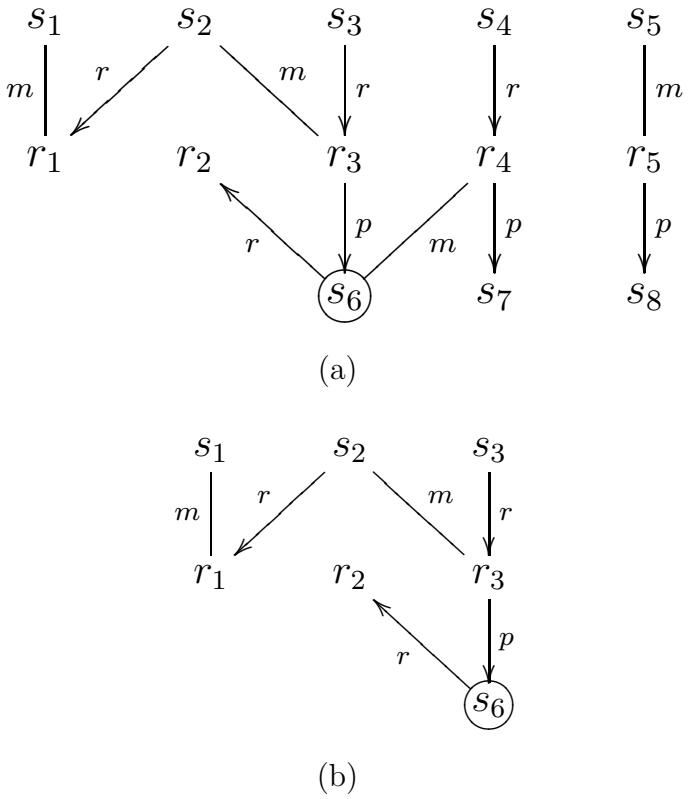


Figure 4.21. Irrelevant node elimination: (a) original model and (b) after reduction.

and degradation reactions of s_6, r_3 and r_2, must be relevant. The reaction r_3 uses s_3 as a reactant and s_2 as a modifier, so these species are relevant too. Since s_2 is relevant, the degradation reaction of s_2, r_1, is also relevant. This reaction uses s_1 as a modifier, so s_1 is relevant. Using these deductions, irrelevant nodes elimination results in the reduced model shown in Figure 4.21(b).

Whereas the irrelevant node elimination guarantees that all the removed nodes are irrelevant to the species in $\mathbf{S_i}$ by statically analyzing the structure of the model, there may still be nodes in the transformed model that can be safely removed without any significant effect on the model. In such cases, a more extensive and expensive dynamic analysis such as sensitivity analysis [35, 59] can be applied to further reduce the model complexity.

Figure 4.22 shows algorithms for irrelevant node elimination. Algorithm 4.10.1 is only concerned with the case where interesting species are specified by the user. Thus, it first checks whether or not $\mathbf{S_i}$ is empty (line 1). The algorithm initializes the set of irrelevant nodes, $\mathbf{V}$, to contain all the species and reactions (line 2). For each interesting species, s, it then removes from $\mathbf{V}$ the relevant nodes by calling Algorithm 4.10.2 (lines 3-5). Finally, it removes irrelevant species and reactions from the model and returns the new model (lines 6-8). Algorithm 4.10.2 considers species s if it is still in $\mathbf{V}$ (i.e., if s is still considered as an irrelevant species) (line 1). It first removes s from $\mathbf{V}$ (line 2). Then, for each reaction r such that r can change the state of s, Algorithm 4.10.3 is called to remove any relevant nodes from $\mathbf{V}$ (lines 3-5). Finally, it returns the updated $\mathbf{V}$ (line 6). Algorithm 4.10.3 acts in a similar fashion as Algorithm 4.10.2. It considers reaction r if it is still in $\mathbf{V}$, and if r is in $\mathbf{V}$, then r is removed from $\mathbf{V}$ (lines 1-2). For each species s such that $|s|$ is used in the kinetic law of reaction r, Algorithm 4.10.2 is called to remove relevant nodes from $\mathbf{V}$ (lines 3-5). Finally, it returns the updated $\mathbf{V}$ (line 6).

Algorithm 4.10.1 *Remove irrelevant nodes*
void IrrelevantNodeElim(Model M)

1: ***if*** $(\mathbf{S_i} = \emptyset)$ ***return*** M
2: $\mathbf{V} \leftarrow (\mathbf{S} \cup \mathbf{R})$
3: ***for all*** $s \in \mathbf{S_i}$ ***do***
4: $\quad \mathbf{V} \leftarrow FindIrrelevantSpecies(M,\ s,\ \mathbf{V})$
5: ***end for***
6: $\forall r \in (\mathbf{V} \cap \mathbf{R}).\ removeReaction(M,\ r)$
7: $\forall s \in (\mathbf{V} \cap \mathbf{S}).\ removeSpecies(M,\ s)$
8: ***return*** M

Algorithm 4.10.2 *Find irrelevant species*
Set FindIrrelevantSpecies(Model M, Species s, Set $\mathbf{V}$ *)*

1: ***if*** $(s \notin \mathbf{V})$ ***return*** $\mathbf{V}$
2: $\mathbf{V} \leftarrow \mathbf{V} \setminus \{s\}$
3: ***for all*** $r \in (\mathbf{R_s^r} \cup \mathbf{R_s^p})$ ***do***
4: $\quad \mathbf{V} \leftarrow FindIrrelevantReaction(M,\ r,\ \mathbf{V})$
5: ***end for***
6: ***return*** $\mathbf{V}$

Algorithm 4.10.3 *Find irrelevant reactions*
Set FindIrrelevantReaction(Model M, Reaction r, Set $\mathbf{V}$ *)*

1: ***if*** $(r \notin \mathbf{V})$ ***return*** $\mathbf{V}$
2: $\mathbf{V} \leftarrow \mathbf{V} \setminus \{r\}$
3: ***for all*** $s \in (\mathbf{S_r^r} \cup \mathbf{S_r^m} \cup \mathbf{S_r^p})$ ***do***
4: $\quad \mathbf{V} \leftarrow FindIrrelevantSpecies(M,\ s,\ \mathbf{V})$
5: ***end for***
6: ***return*** $\mathbf{V}$

Figure 4.22. Algorithms to perform irrelevant node elimination.

CHAPTER 5

STATE-BASED ABSTRACTION

In the SCK analysis via the SSA, the temporal behavior is estimated by generating n sample trajectories of the system as outcomes of n simulation runs. Intuitively, as $n \rightarrow \infty$, this approach gives the best estimate of the temporal behavior. Indeed, at this limit, if the system has a finite variance, the *central limit theorem* guarantees that the distribution of the n-sample average is asymptotically normal, and the *standard error*, S_E, which measures the difference between the estimated mean temporal behavior from the n Monte Carlo simulation runs and the true mean temporal behavior of the system is formulated as $S_E = \hat{\sigma}/\sqrt{n}$ where $\hat{\sigma}$ is the estimated standard deviation. This implies that, as $n \rightarrow \infty$, the numerical estimation reflects the true mean behavior. This also shows that, in order to decrease the uncertainty involved in the numerical estimation of temporal behavior N times, the number of simulation runs must be increased N^2 times. Instead of taking this potentially very expensive approach, the abstracted *chemical master equation* (CME) can be directly solved to estimate the time evolution of the probability distribution [116, 75, 93, 85].

This chapter presents several such abstraction methods to transform a REB model into a state-based model called the *finite state system model* (FSS model) to further improve the numerical analysis time. Section 5.1 formally defines the FSS model, and Section 5.2 describes several analysis methods for the FSS model. Section 5.3 then presents an abstraction method to transform a REB model into a FSS model. Section 5.4 presents another transformation method to further reduce the analysis time by generating a reduced FSS model called the *stochastic asynchronous circuit model* (SAC model).

5.1 Finite State System Model

The FSS model is a state-based, continuous-time discrete-event system where the stochastic state transition of each species is restricted to a finite state space unlike most state transitions in a REB model. By making each species' state space finite, the overall system space of the FSS model becomes also finite. Thus, the state space of a FSS model can be explicitly specified. The FSS model compactly represents a time-homogeneous, discrete-state, Markov process in a finite state space whereby state transitions are decided based on the information on the current state. Therefore, while a system described using the FSS model can be analyzed via stochastic simulation methods such as the SSA, it can also be analyzed using a Markov chain analysis method [110]—albeit possibly requiring a substantial amount of memory to generate all the underlying system states—to directly obtain the solution of an abstracted CME. The FSS model is formally defined as follows.

Definition 5.1 (FSS model) *A FSS model is specified with* $\langle \mathbf{Z}, \mathbf{z_0}, \mathbf{z_{max}}, \mathbf{C} \rangle$ *where* $\mathbf{Z} \equiv (Z_1, \ldots, Z_n)$ *is a vector of non-negative integer random variables,* $\mathbf{z_0}$ *is the vector containing the values of* $\mathbf{Z}$ *at time* 0, $\mathbf{z_{max}}$ *is the vector whose* i*-th element,* z^i_{max}*, specifies the maximum value that random variable* Z_i *can take, and* $\mathbf{C} \equiv \{c_1, \ldots, c_m\}$ *is the set of guarded commands that change the values of the random variables. The system state space of* $\mathbf{Z}$*,* $\mathbf{\Sigma_z}$*, is specified as*

$$\mathbf{\Sigma_z} = \{\mathbf{z} \mid \forall i.\ z_i \in [0, z^i_{max}]\}.$$

$\mathbf{Z}(t)$ *specifies the system state at time* t*. Thus, for each* Z_i*, the probability that* $Z_i(t) > z^i_{max}$ *or* $Z_i(t) < 0$ *is zero for any* $t \geq 0$*. When the system is in state* $\mathbf{z}$*, each guarded command,* c_j*, has a form:*

$$G_j(\mathbf{z}) \xrightarrow{q_j} \mathbf{Z} = \mathbf{z} + \mathbf{u_j}$$

where the function $G_j(\mathbf{z}) : \{0, \ldots, z^i_{max}\}^n \mapsto \{0, 1\}$ *is the guard for* c_j *when the system state is* $\mathbf{z}$*,* q_j *is the transition rate for* c_j*, and* $\mathbf{u_j}$ *is an* n*-dimensional vector whose* i*-th element has the value added to* Z_i *as a result of* c_j*.*

Let [[*bool-exp*]] be an operator that takes a Boolean expression, *bool-exp*, and evaluates to 1 if *bool-exp* is true and 0 otherwise. Then, the guard, $G_j(\mathbf{z})$, of each guarded command, c_j, has the form:

$$G_j(\mathbf{z}) = \prod_{i \in \mathbf{N_j}} [[z_i = v_j^i]] \tag{5.1}$$

where the expression $[[z_i = v_j^i]]$ results in 1 if the current state of Z_i is equal to the value specified by the constant v_j^i otherwise results in 0, and $\mathbf{N_j}$ is a subset of $[1, n]$. Each guarded command, c_j, is required to change the state of $\mathbf{Z}$. Thus, each $\mathbf{N_j}$ must satisfy the condition $|\mathbf{N_j}| > 0 \wedge |\mathbf{N_j}| \leq n$. If the system state is $\mathbf{z}$ at time t (i.e., $\mathbf{Z}(t) = \mathbf{z}$), c_j can be executed if its guard is satisfied (i.e., $G_j(\mathbf{z}) = 1$). The result of executing the guarded command in time step τ is that a new state is reached in which $\mathbf{Z_i}(t + \tau) = \mathbf{z} + \mathbf{u_j}$. Note that $G_j(\mathbf{z})$ can be efficiently encoded in the state graph of a FSS model by using the connection from state $\mathbf{z}$ to state $\mathbf{z} + \mathbf{u_j}$ as an indicator so that $G_j(\mathbf{z})$ is evaluated to 1 if there is a transition edge from state $\mathbf{z}$ to state $\mathbf{z} + \mathbf{u_j}$ otherwise to 0.

From the definition of the FSS model, the probability that, given the system is in state $\mathbf{z}$, c_j is executed and $\mathbf{Z}$ moves to state $\mathbf{z} + \mathbf{u_j}$ within the next infinitesimal time step dt is:

$$P(c_j, dt \mid \mathbf{z}) = G_j(\mathbf{z}) \cdot q_j \cdot dt. \tag{5.2}$$

Consequently, the probability that no transition is taken within the next time step dt is:

$$1 - \left[\sum_{k=1}^{m} G_k(\mathbf{z}) \cdot q_k \cdot dt \right]. \tag{5.3}$$

Thus, from these equations, the time evolution equation of $P(\mathbf{z}, t \mid \mathbf{z_0})$ which describes the probability that $\mathbf{Z}(t) = \mathbf{z}$ given $\mathbf{Z}(0) = \mathbf{z_0}$ for all $t \geq 0$ can be deduced as:

$$\begin{aligned} P(\mathbf{z}, t + dt \mid \mathbf{z_0}) = P(\mathbf{z}, t \mid \mathbf{z_0}) \left[1 - \sum_{j=1}^{m} G_j(\mathbf{z}) q_j dt \right] \\ + \sum_{j=1}^{m} [P(\mathbf{z} - \mathbf{u_j}, t \mid \mathbf{z_0}) G_j(\mathbf{z} - \mathbf{u_j}) q_j dt]. \end{aligned} \tag{5.4}$$

By taking the limit: $dt \rightarrow 0$, and rearranging Equation 5.4, the following abstracted CME represented by a FSS model can be obtained:

$$\frac{\partial P(\mathbf{z}, t \mid \mathbf{z_0})}{\partial t} = \sum_{j=1}^{m} [G_j(\mathbf{z} - \mathbf{u_j}) q_j P(\mathbf{z} - \mathbf{u_j}, t \mid \mathbf{z_0}) - G_j(\mathbf{z}) q_j P(\mathbf{z}, t \mid \mathbf{z_0})]. \quad (5.5)$$

5.2 Finite State System Model Analysis

This section presents several methods to analyze the time evolution of a system described by a FSS model. It first describes a Monte Carlo simulation approach for the FSS model to infer temporal behavior of the system from sample trajectories. It then describes Markov chain analysis methods to directly solve Equation 5.5 to obtain the time evolution of the probability distribution.

5.2.1 Stochastic Simulation of the FSS Model

A stochastic simulation of a process described using a FSS model begins in the state $\mathbf{z_0}$ at time 0 and selects either a guarded command to execute or no guarded commands to execute in a small time step Δt. If a guarded command is executed at time t_1, then the system moves to a new state so that $\mathbf{Z}(t_1) = \mathbf{z_1}$. It then recalculates all the transition probabilities, and continues until terminated. This simulation process is inexact and inefficient since Δt is not a true infinitesimal, yet for a sufficiently small Δt most simulation steps do not result in a state change. Hence, as with a REB model, the exact SSA [53, 54] which skips over the time steps where no state change occurs can be used instead for a more efficient Monte Carlo simulation of a FSS model. The SSA for a FSS model uses the expression $G_j(\mathbf{z})q_j$ as the transition rate for the guarded command, c_j, when the system state is $\mathbf{z}$, which is analogue to the propensity function in the SSA for a REB model.

The algorithm for the stochastic simulation of a FSS model using the direct method of the SSA [52] is described in Figure 5.1. Algorithm 5.2.1 first initializes the time t to 0 and the system state $\mathbf{z}$ to $\mathbf{z_0}$ (line 1). Then, it repeats the sequence of calculating a_0 by summing up $G_k(\mathbf{z})q_k$ for k from 1 to m (line 3), picking two unit uniform random numbers n_1 and n_2 (line 4), computing the next transition time τ and the index of the next guarded command j (lines 5-6), and updating

Algorithm 5.2.1 *Direct method using the FSS model*

1: $t \leftarrow 0$, $\mathbf{z} \leftarrow \mathbf{z_0}$
2: ***repeat***
3: $a_0 \leftarrow \sum_{k=1}^{m} G_k(\mathbf{z}) q_k$
4: *pick* 2 *unit uniform random numbers* n_1 *and* n_2
5: $\tau \leftarrow -\ln(n_1)/a_0$
6: $j \leftarrow$ *smallest integer satisfying* $\sum_{\mu=1}^{j} G_\mu(\mathbf{z}) q_\mu \geq n_2 a_0$
7: $t \leftarrow t + \tau$, $\mathbf{z} \leftarrow \mathbf{z} + \mathbf{u_j}$
8: ***until*** *simulation termination condition is met*

Figure 5.1. Algorithm for the direct method using the FSS model.

the time t and the current state $\mathbf{z}$ (line 7). This loop is run until the termination condition is satisfied (line 8).

5.2.2 Markov Chain Analysis of the FSS Model

In addition to stochastic simulation, a system can be analyzed by solving the abstracted CME that a FSS model represents. This is because, by the definition of the FSS model, Equation 5.5 is a set of ODEs with a finite set of equations and a finite system state space, $|\mathbf{\Sigma_z}|$, making a numerical solution of the abstracted CME possible. Since it is most likely the case that a biochemical system represented by a FSS model is very *sparse* meaning that the majority of $c_j \in \mathbf{C}$ have $G_j(\mathbf{z})$ equal to 0 for each given state $\mathbf{z}$, efficient iterative methods can be applied to the state graph of an underlying continuous-time Markov chain to solve Equation 5.5 [110].

The time evolution of $P(\mathbf{z}, t + \Delta t \mid \mathbf{z_0})$ is generated by approximating the infinitesimal time step dt in Equation 5.4 by finite time step Δt as:

$$\begin{aligned} P(\mathbf{z}, t + \Delta t \mid \mathbf{z_0}) = P(\mathbf{z}, t \mid \mathbf{z_0}) \left[1 - \sum_{j=1}^{m} G_j(\mathbf{z}) q_j \Delta t \right] \\ + \sum_{j=1}^{m} [P(\mathbf{z} - \mathbf{u_j}, t \mid \mathbf{z_0}) G_j(\mathbf{z} - \mathbf{u_j}) q_j \Delta t]. \end{aligned} \tag{5.6}$$

This can be numerically computed using various methods for the initial value problem of ODEs [110, 94]. Figure 5.2 shows the algorithm of a simple iterative method [110] to estimate the time evolution of the probability distribution of a

Algorithm 5.2.2 *Iterative method using the FSS model*

1: $\Delta t \leftarrow \left(\alpha \max_{\mathbf{z}\in\mathbf{\Sigma_z}} \left(\sum_{j=1}^{m} G_j(\mathbf{z}) q_j\right)\right)^{-1}$
2: $t \leftarrow 0$
3: ***for all*** $\mathbf{z} \in \mathbf{\Sigma_z}$ ***do***
4: $\quad P(\mathbf{z}, t) \leftarrow 0$
5: ***end for***
6: $P(\mathbf{z_0}, t) \leftarrow 1$
7: ***repeat***
8: $\quad$ ***for all*** $\mathbf{z} \in \mathbf{\Sigma_z}$ ***do***
9: $\quad\quad P(\mathbf{z}, t + \Delta t) \leftarrow P(\mathbf{z}, t)[1 - \sum_{j=1}^{m} G_j(\mathbf{z}) q_j \Delta t]$
10: $\quad\quad P(\mathbf{z}, t + \Delta t) \leftarrow P(\mathbf{z}, t + \Delta t) + \sum_{j=1}^{m} G_j(\mathbf{z} - \mathbf{u_j}) q_j P(\mathbf{z} - \mathbf{u_j}, t) \Delta t$
11: $\quad$ ***end for***
12: $\quad t \leftarrow t + \Delta t$
13: ***until*** $t \geq t_{max}$

Figure 5.2. Algorithm for a simple iterative method using the FSS model.

system using a FSS model. This generates the time evolution of the probability that $\mathbf{Z}(t) = \mathbf{z}$, $P(\mathbf{z}, t)$, for all $\mathbf{z} \in \mathbf{\Sigma_z}$ given that $P(\mathbf{z_0}, 0) = 1$ up to the time limit, t_{max}. This algorithm is based on the forward Euler method where the time step, Δt, is chosen to be:

$$\Delta t = \frac{1}{\alpha \max_{\mathbf{z}\in\mathbf{\Sigma_z}} \left(\sum_{j=1}^{m} G_j(\mathbf{z}) q_j\right)} \tag{5.7}$$

where $\alpha \geq 1$ is the accuracy factor that can be set by the user. Note that the higher the value of α is, the higher the accuracy becomes for the computation of $P(\mathbf{z}, t)$ albeit with more expensive computational costs. Algorithm 5.2.2 first sets Δt using Equation 5.7 (line 1). The algorithm initializes time t to 0 and $P(\mathbf{z}, t)$ to 1 if $\mathbf{z} = \mathbf{z_0}$ otherwise to 0 (lines 2-6). It then iterates the process of computing the probability distribution of the next time step until t reaches the time limit. For each state $\mathbf{z}$, $P(\mathbf{z}, t + \Delta t)$ is calculated by subtracting from $P(\mathbf{z}, t)$ the probability to move from $\mathbf{z}$ to other states and adding the probability to move to $\mathbf{z}$ from other states (lines 7-11). After the probabilities of all the states are calculated for this iteration, t is updated to be $t + \Delta t$ (line 12).

Sometimes, the required result from the computational analysis of a FSS model is only the estimate of the *stationary probability* of a system, $P(\mathbf{z}, \infty \mid \mathbf{z_0})$, which

describes the probability that $\mathbf{Z}(t) = \mathbf{z}$ as t approaches infinity given that initially $\mathbf{Z}(0) = \mathbf{z_0}$. In the limit $t \to \infty$, the biochemical system equilibrates, and thus the change in the probability over time becomes zero for all system states. Thus, from Equation 5.5, this results in the derivation of the following equation:

$$\sum_{j=1}^{m} [G_j(\mathbf{z} - \mathbf{u_j}) q_j P(\mathbf{z} - \mathbf{u_j}, \infty \mid \mathbf{z_0}) - G_j(\mathbf{z}) q_j P(\mathbf{z}, \infty \mid \mathbf{z_0})] = 0. \tag{5.8}$$

This equation can be once again numerically solved using an iterative method. To do that, let us first introduce a probability density function $P(\mathbf{z}, k \mid \mathbf{z_0})$ which specifies the probability that the system is in state $\mathbf{z}$ in k state transitions given that the system is initially in state $\mathbf{z_0}$. Since a FSS model is a Markov process, the average transition time of c_j when the system is in state $\mathbf{z}$ is $1/\sum_{\mu=1}^{m} G_\mu(\mathbf{z}) q_\mu$. Thus, by replacing each Δt for a c_j event firing in Equation 5.6 with its average transition time, the time evolution equation of $P(\mathbf{z}, k \mid \mathbf{z_0})$ becomes:

$$P(\mathbf{z}, k+1 \mid \mathbf{z_0}) = \sum_{j=1}^{m} \left[P(\mathbf{z} - \mathbf{u_j}, k \mid \mathbf{z_0}) \frac{G_j(\mathbf{z} - \mathbf{u_j}) q_j}{\sum_{\mu=1}^{m} G_\mu(\mathbf{z} - \mathbf{u}_\mu) q_\mu} \right]. \tag{5.9}$$

This equation eliminates the term involving $P(\mathbf{z}, k \mid \mathbf{z_0})$ on the right-hand side as $1 - \sum_{j=1}^{m} \left[G_j(\mathbf{z}) q_j / (\sum_{\mu=1}^{m} G_\mu(\mathbf{z}) q_\mu) \right]$ is zero. This makes sense since, if the system were in state $\mathbf{z}$ after the k-th jump, the system could not be in state $\mathbf{z}$ after the $k+1$-th jump. When, for all $\mathbf{z}$ in $\mathbf{\Sigma_z}$, $P(\mathbf{z}, k+1 \mid \mathbf{z_0}) = P(\mathbf{z}, k \mid \mathbf{z_0})$, the changes in the probability distribution become zero, satisfying Equation 5.8. Therefore, a stationary distribution can be obtained by iterating k until the condition: $\forall \mathbf{z}.\ P(\mathbf{z}, k+1 \mid \mathbf{z_0}) = P(\mathbf{z}, k \mid \mathbf{z_0})$ is satisfied.

Figure 5.3 shows an algorithm to generate a stationary probability distribution of $\mathbf{Z}$ given that $\mathbf{Z}(0) = \mathbf{z_0}$, assuming that there is a stationary probability distribution. Note that, though we have not encountered this situation in analysis of biochemical systems, it is possible that a FSS model does not have a stationary probability distribution for a given initial state. To anticipate such cases, the algorithm can be extended to check if the probability distribution can converge within the maximum transition step, k_{max}. Algorithm 5.2.3 first sets a transition counter k to 0, and initializes the probability distribution so that $P(\mathbf{z}, k \mid \mathbf{z_0})$ is

Algorithm 5.2.3 *A stationary probability distribution*

1: $k \leftarrow 0$
2: ***for all*** $\mathbf{z} \in \mathbf{\Sigma_z}$ ***do***
3: $\quad P(\mathbf{z}, k \mid \mathbf{z_0}) \leftarrow 0$
4: ***end for***
5: $P(\mathbf{z_0}, k \mid \mathbf{z_0}) \leftarrow 1$
6: ***repeat***
7: $\quad$ ***for all*** $\mathbf{z} \in \mathbf{\Sigma_z}$ ***do***
8: $\quad\quad P(\mathbf{z}, k+1 \mid \mathbf{z_0}) \leftarrow \sum_{j=1}^{m} \left[P(\mathbf{z} - \mathbf{u_j}, k \mid \mathbf{z_0}) \frac{G_j(\mathbf{z}-\mathbf{u_j})q_j}{\sum_{\mu=1}^{m} G_\mu(\mathbf{z}-\mathbf{u}_\mu)q_\mu} \right]$
9: $\quad$ ***end for***
10: $\quad k \leftarrow k+1$
11: ***until*** $\forall \mathbf{z} \in \mathbf{\Sigma_z}.\ P(\mathbf{z}, k \mid \mathbf{z_0}) \approx P(\mathbf{z}, k-1 \mid \mathbf{z_0})$

Figure 5.3. Algorithm for a simple iterative method to obtain a stationary probability distribution using the FSS model.

1 if $\mathbf{z} = \mathbf{z_0}$ and 0 otherwise (lines 1-5). It then repeats the process of computing $P(\mathbf{z}, k+1 \mid \mathbf{z_0})$ using Equation 5.9 and incrementing k until $P(\mathbf{z}, k \mid \mathbf{z_0})$ converges (lines 6-11).

5.3 Finite State System Model Transformation

This section presents a technique to transform a REB model, M_R, into a FSS model, M_F. In order to describe this transformation, without loss of generality, let us suppose that M_R has $\mathbf{S} \equiv \{s_1, \ldots, s_n\}$ where the state of each species s_i can be changed by some reaction (i.e., $|\mathbf{R}^{\mathbf{r}}_{\mathbf{s_i}}| + |\mathbf{R}^{\mathbf{p}}_{\mathbf{s_i}}| > 0$), $\mathbf{R} \equiv \{r_1, \ldots, r_{m'}\}$ where each reaction r_j can change the state of some species (i.e., $|\mathbf{S}^{\mathbf{r}}_{\mathbf{r_j}}| + |\mathbf{S}^{\mathbf{p}}_{\mathbf{r_j}}| > 0$), and $\mathbf{R_{rev}} = \emptyset$. Then, M_F has $\mathbf{Z} \equiv \{Z_1, \ldots, Z_n\}$ where each Z_i specifies $|s_i|$, and each z_0^i in $\mathbf{z_0}$ is $|s_i|_0$. Each z_{max}^i in $\mathbf{z_{max}}$ is set by the user input specifying the upper limit molecular count of s_i.

For the generation of the guarded commands, $\mathbf{C} \equiv \{c_1, \ldots, c_m\}$, each reaction r_j first constructs a set of indices, $\mathbf{I_j}$, such that

$$\mathbf{I_j} = \{i \in [1, n] \mid \mathbf{E}(s_i, r_j) > 0 \vee \mathbf{E}(r_j, s_i) > 0\}. \tag{5.10}$$

Hence, the set $\mathbf{I_j}$ contains all the indices of the species that participate in reaction r_j. Here, let $\hat{\mathbf{Z}}_\mathbf{j}$ be the set of random variables $\{Z_{i'} \mid i' \in \mathbf{I_j}\}$, and $\mathbf{\Sigma}_{\hat{\mathbf{z}}_\mathbf{j}}$ be a subset

of $\mathbf{\Sigma_z}$ such that

$$\begin{aligned}\mathbf{\Sigma_{\hat{z}_j}} \equiv \{\mathbf{z} = (&if\ i \in \mathbf{I_j}\ then\ z_i \in [0, z^i_{max}]\ else\ z_i = 0)\} \\ &\cap \{\mathbf{z} \in \mathbf{\Sigma_z} \mid \mathbf{K}(r_j)(\mathbf{z}) > 0\} \\ &\cap \{\mathbf{z} \in \mathbf{\Sigma_z} \mid \forall i.\ z_i \leq z^i_{max} - [\mathbf{E}(r_j, s_i) - \mathbf{E}(s_i, r_j)]\} \\ &\cap \{\mathbf{z} \in \mathbf{\Sigma_z} \mid \forall i.\ z_i \geq [\mathbf{E}(s_i, r_j) - \mathbf{E}(r_j, s_i)]\}.\end{aligned} \tag{5.11}$$

In other words, $\mathbf{\Sigma_{\hat{z}_j}}$ is a subset of $\mathbf{\Sigma_z}$ which has every state $\mathbf{z}$ in which, using the value of each z_i for $|s_i|$, a reaction r_j event can fire with a non-zero transition rate to move to a state where the new value of each Z_i is at most z^i_{max} and at least 0, provided that the states of species that do not participate in reaction r_j are fixed to be 0 (i.e., $z_i = 0$ if $i \notin \mathbf{I_j}$). Then, each reaction r_j generates guarded commands using the information on each state in $\mathbf{\Sigma_{\hat{z}_j}}$, resulting in as many as $|\mathbf{\Sigma_{\hat{z}_j}}|$ guarded commands. Each guarded command, c_μ, from state $\mathbf{z}_\mu \in \mathbf{\Sigma_{\hat{z}_j}}$ is used for the transition event of reaction r_j in states where only the value of $Z_{i'} \in \mathbf{\hat{Z}_j}$ is constrained by the i'-th element of $\mathbf{z}_\mu$, $z^{i'}_\mu$. The guard, $G_\mu(\mathbf{z})$, checks if the condition to enable the transition event, c_μ, is satisfied in state $\mathbf{z}$. This condition can only be satisfied when

$$\forall i' \in \mathbf{I_j}.\ z_{i'} = z^{i'}_\mu \tag{5.12}$$

is true. To generate the form of Equation 5.1, thus, $\mathbf{N}_\mu$ is set so that $\mathbf{N}_\mu$ is equal to $\mathbf{I_j}$, and each v^i_μ is the constant whose value is specified by z^i_μ. The transition rate of c_μ, q_μ, is computed by evaluating $\mathbf{K}(r_j)$ using the value of $\mathbf{z}_\mu$. The increment vector, $\mathbf{u}_\mu$, specifies the increment of the system state as a result of the firing of one c_μ event. Thus $\mathbf{u}_\mu$ is generated so that the i-th element of $\mathbf{u}_\mu$ is specified as:

$$u^i_\mu = \mathbf{E}(r_j, s_i) - \mathbf{E}(s_i, r_j). \tag{5.13}$$

To illustrate the FSS model transformation of a REB model, consider a REB model, M_R, which has the following structure:

$$\mathbf{S} = \{s_1, s_2, s_3\},$$

$$\mathbf{R} = \{r_1, r_2, r_3, r_4\},$$

$$\mathbf{R_{rev}} = \emptyset,$$

$$\mathbf{E} = \{((s_1, r_1), 0), ((s_2, r_1), 0), ((s_3, r_1), 0), ((r_1, s_1), 2), ((r_1, s_2), 0), ((r_1, s_3), 0),$$
$$((s_1, r_2), 1), ((s_2, r_2), 1), ((s_3, r_2), 0), ((r_2, s_1), 0), ((r_2, s_2), 1), ((r_2, s_3), 0),$$
$$((s_1, r_3), 0), ((s_2, r_3), 1), ((s_3, r_3), 0), ((r_3, s_1), 0), ((r_3, s_2), 0), ((r_3, s_3), 0),$$
$$((s_1, r_4), 1), ((s_2, r_4), 0), ((s_3, r_4), 0), ((r_4, s_1), 1), ((r_4, s_2), 0), ((r_4, s_3), 1)\},$$

$$\mathbf{K} = \{(r_1 \rightarrow ((|s_1|, |s_2|, |s_3|) \rightarrow f_1())),$$
$$(r_2 \rightarrow ((|s_1|, |s_2|, |s_3|) \rightarrow f_2(|s_1|, |s_2|))),$$
$$(r_3 \rightarrow ((|s_1|, |s_2|, |s_3|) \rightarrow f_3(|s_2|))),$$
$$(r_4 \rightarrow ((|s_1|, |s_2|, |s_3|) \rightarrow f_4(|s_1|)))\},$$

where $f_1() > 0$, $f_2(|s_1|, |s_2|)$ is greater than 0 if $|s_1| > 0$, otherwise it is 0, $f_3(|s_2|)$ is greater than 0 if $|s_2| > 0$, otherwise it is 0, and $f_4(|s_1|)$ is greater than 0 if $|s_1| > 0$, otherwise it is 0. To better understand the structure of M_R, its graphical representation is shown in Figure 5.4. Since M_R has three species, the transformation of M_R into its corresponding FSS model, M_F, begins by having $\mathbf{Z} = \{Z_1, Z_2, Z_3\}$. In this transformation, suppose the initial states of s_1, s_2, and

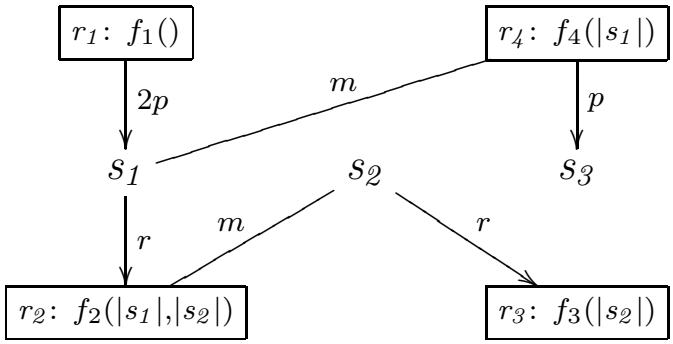


Figure 5.4. The graphical representation of REB model M_R.

s_3 are 0, 10, and 0, respectively, and the upper limit molecular counts of s_1, s_2, and s_3 are all 10. Then, by definition, $\mathbf{z_0} = (0, 10, 0)$ and $\mathbf{z_{max}} = (10, 10, 10)$.

To generate the set of the guarded commands for reaction r_1, $\mathbf{I_1}$ is constructed first. Since reaction r_1 is not influenced by any species in the model, $\mathbf{I_1}$ becomes $\{1\}$. This implies that $\hat{\mathbf{Z}}_\mathbf{1} = \{Z_1\}$. Also, since reaction r_1 only involves species s_1 and $\mathbf{E}(r_1, s_1) - \mathbf{E}(s_1, r_1) = 2$, from Definition 5.11, $\mathbf{\Sigma}_{\hat{\mathbf{z}}_\mathbf{1}}$ includes the states to change the value of Z_1 from 0 to 8. This means that $\mathbf{\Sigma}_{\hat{\mathbf{z}}_\mathbf{1}}$ can be obtained as:

$$\mathbf{\Sigma}_{\hat{\mathbf{z}}_\mathbf{1}} = \{(z_1, 0, 0) \mid z_1 \in [0, 8]\}. \tag{5.14}$$

Thus, since $|\mathbf{\Sigma}_{\hat{\mathbf{z}}_\mathbf{1}}| = 9$, nine guarded commands are generated from reaction r_1, each of which has the transition rate $f_1()$ and the increment vector $(2, 0, 0)$. Therefore, the nine guarded commands have the following structure:

$$\forall k_1 \in [0, 8].\ \ [[z_1 = k_1]] \xrightarrow{f_1()} \mathbf{Z} = \mathbf{z} + (2, 0, 0). \tag{5.15}$$

Reaction r_2 involves two species where s_1 is used as a reactant and s_2 is used as a modifier. Thus, $\mathbf{I_2}$ becomes $\{1, 2\}$, resulting in $\hat{\mathbf{Z}}_\mathbf{2} = \{Z_1, Z_2\}$. Since r_2 changes only the state of species s_1 where one molecule of s_1 is consumed per reaction event (i.e., $\mathbf{E}(s_1, r_2) - \mathbf{E}(r_2, s_1) = 1$), $\mathbf{\Sigma}_{\hat{\mathbf{z}}_\mathbf{2}}$ is described as:

$$\mathbf{\Sigma}_{\hat{\mathbf{z}}_\mathbf{2}} = \{(z_1, z_2, 0) \mid z_1 \in [1, 10] \wedge z_2 \in [0, 10]\}. \tag{5.16}$$

This implies that $|\mathbf{\Sigma}_{\hat{\mathbf{z}}_\mathbf{2}}| = 110$. Hence, reaction r_2 generates $10 \times 11 = 110$ guarded commands, each of which has the increment vector of $(-1, 0, 0)$. Therefore, the 110 guarded commands generated by reaction r_2 have the following structure:

$$\forall k_1 \in [1, 10], k_2 \in [0, 10].\ \ [[z_1 = k_1]] \times [[z_2 = k_2]] \xrightarrow{f_2(k_1, k_2)} \mathbf{Z} = \mathbf{z} + (-1, 0, 0). \tag{5.17}$$

In order to generate the guarded commands from reaction r_3, once again, the set of participant species' indices is first examined. Since reaction r_3 involves only

species s_2, $\mathbf{I_3} = \{2\}$ and $\hat{\mathbf{Z}}_\mathbf{3} = \{Z_2\}$. Furthermore, since at least one molecule of s_2 is needed to fire a reaction r_3 event, $\mathbf{\Sigma}_{\hat{\mathbf{z}}_\mathbf{3}}$ is obtained as:

$$\mathbf{\Sigma}_{\hat{\mathbf{z}}_\mathbf{3}} = \{(0, z_2, 0) \mid z_2 \in [1, 10]\}. \tag{5.18}$$

Thus, there are 10 guarded commands generated by reaction r_3, each of which has the increment vector of $(0, -1, 0)$. Therefore, the 10 guarded commands of r_3 have the following structure:

$$\forall k_2 \in [1, 10].\ \ [[z_2 = k_2]] \xrightarrow{f_3(k_2)} \mathbf{Z} = \mathbf{z} + (0, -1, 0). \tag{5.19}$$

Finally, the guarded commands from reaction r_4 are generated by carrying out the same procedure. First, the sets, $\mathbf{I_3} = \{1, 3\}$, and $\hat{\mathbf{Z}}_\mathbf{3} = \{Z_1, Z_3\}$, are determined because reaction r_4 uses species s_1 as a modifier to produce species s_3. Since $f_4(|s_1|) = 0$ when $|s_1| = 0$, and the upper limit molecular count of s_3 is 10,

$$\mathbf{\Sigma}_{\hat{\mathbf{z}}_\mathbf{4}} = \{(z_1, 0, z_3) \mid z_1 \in [1, 10] \text{ and } z_1 \in [0, 9]\}. \tag{5.20}$$

This implies that reaction r_4 generates $10 \times 10 = 100$ guarded commands where each guarded command has the increment vector of $(-1, 0, 0)$. Therefore, the 100 guarded commands generated have the following structure:

$$\forall k_1 \in [1, 10], k_3 \in [0, 9].\ \ [[z_1 = k_1]] \times [[z_3 = k_3]] \xrightarrow{f_4(k_1)} \mathbf{Z} = \mathbf{z} + (0, 0, 1). \tag{5.21}$$

5.4 N-ary Transformation

While the FSS model transformation method described in the previous section provides a means to analyze the time evolution of biochemical systems by directly solving the CMEs, this method is proven to be inefficient for systems with very large system state space. Even for a system of 10 species where each has an upper limit molecular count of 99, the FSS model transformation can generate up to 10^{20} states. Constructing such a state graph for temporal behavior analysis is infeasible for most computers. Thus, the FSS model transformation method should not be used in such cases.

To more aggressively reduce the state space of a FSS model, this section develops another transformation method called *n-ary transformation*. The n-ary transformation transforms a REB model to a reduced FSS model called the stochastic asynchronous circuit model (SAC model). A SAC model describes the state of each species by n-ary or Boolean levels instead of molecular counts, resulting in further reduction of states per species. Thus, it can further improve the analysis time. For example, suppose a system has 10 species, each of whose states can be qualitatively described as "low," "medium," and "high." Then, with the n-ary transformation, a SAC model with at most 10^3 states can be generated. Therefore, this model can be efficiently analyzed, for example, using Markov chain analysis methods within the asynchronous circuit analysis tool ATACS [1].

Aside from the conditions required for the FSS model transformation, the n-ary transformation requires the REB model to satisfy the property that all reactions should have either one reactant *or* one product, but not both. In other words, the condition:

$$\forall r \in \mathbf{R}.\ |\mathbf{S_r^r}| + |\mathbf{S_r^p}| = 1 \tag{5.22}$$

must be satisfied in order for a REB model to be transformed into a SAC model. Thus, each guarded command in a SAC model, c_j, comes with the following restriction on the increment vector, $\mathbf{u_j}$:

$$\exists i.\ (u_j^i > 0) \wedge (\forall k \neq i.\ u_j^k = 0). \tag{5.23}$$

Thus, the SAC model is a subset of the FSS model whereby each guarded command changes the value of exactly one random variable as a result of its execution. This is often the case after applying the REB abstractions described earlier. If Condition 5.22 does not hold, however, it can be made to hold using *reaction splitization*. One form of reaction splitization is called *single reactant single product reaction splitization*, which splits an irreversible reaction with a single reactant and a single product into an irreversible reaction with no reactant and a single product and an irreversible reaction with a single reactant and no product. In order to

illustrate this transformation, consider the reaction shown in Figure 5.5(a). This reaction converts species s_1 into species s_2 with a rate law $f(|s_1|)$. After splitization, this is transformed into the two reactions shown in Figure 5.5(b). This includes a degradation reaction for s_1 and a production reaction for s_2, with the same rate law. In addition, there are also *multiple reactants reaction splitization* to split a reaction with multiple reactants into multiple reactions with a single reactant and *multiple products reaction splitization* that splits a reaction with multiple products into multiple reactions with a single product, as illustrated in Figure 5.6.

When a REB model satisfies Condition 5.22, it can be transformed into a SAC model with the n-ary transformation. The transformation begins by identifying the states of each species. Let $\mathbf{A_i} \equiv \{A_i^0, A_i^1, \ldots, A_i^{N_i}\}$ be a set with N_i elements that

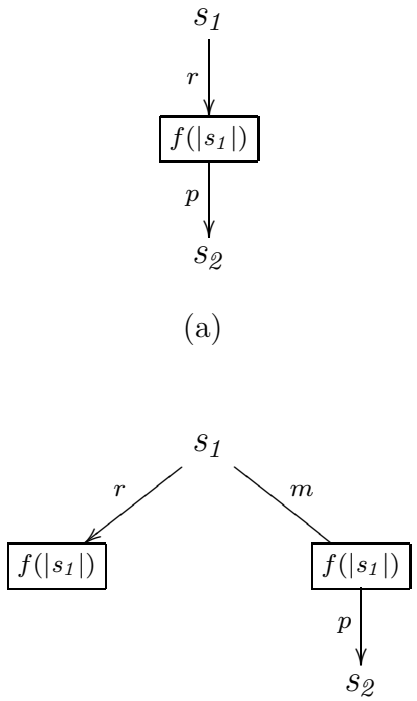


Figure 5.5. Single reactant single product reaction splitization: (a) original reaction and (b) split-up reactions.

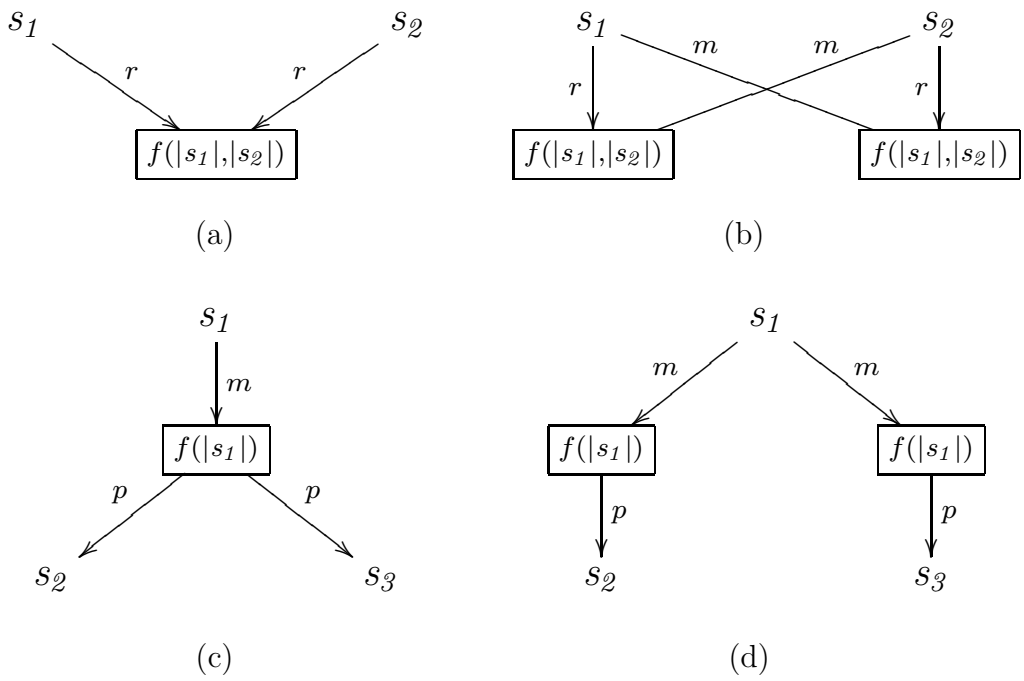


Figure 5.6. Multiple reactants and products reaction splitization. Multiple reactants reaction splitization: (a) original reaction and (b) split-up reactions. Multiple products reaction splitization: (c) original reaction and (d) split-up reactions.

partitions the states of species s_i such that $\forall j.\ A_i^j = [\theta_i^j, \theta_i^{j+1})$ where $\theta_i^0 = 0$, and $\theta_i^{N_i+1} = \infty$. We call $A_i^0, \ldots, A_i^{N_i}$ *critical intervals* and $\theta_i^0, \ldots, \theta_i^{N_i}$ *critical levels* of species s_i. Depending on the nature of the application, the critical levels can be either specified by the user and taken to be model inputs—such as might be the case when our system is utilized by an expert already familiar with the *in situ* behavior of the underlying regulatory network—or estimated automatically from the kinetic rate laws. The SAC model treats each A_i^j as one state. Thus, N_i describes the highest state for species s_i in a SAC model, and thus, in the FSS model notation, N_i is in fact z_{max}^i in that, for all $t \geq 0$, $Z_i(t) \geq 0 \wedge Z_i(t) \leq N_i$. The initial state of Z_i, z_0^i, is determined by examining each critical interval, A_i^j for the condition: $|s_i|_0 \in A_i^j$. Then, z_0^i is set to the index of the critical interval that satisfies this condition.

In $\mathbf{A_i}$, if $\theta_i^{j+1} - \theta_i^j \gg 1$ for some j in $[1, N_i]$, then our method can collapse many states in A_i^j into one state for species s_i, resulting in significant improvement in

analysis time. On the other hand, if $\forall j \in [1, n]$. $\theta_i^{j+1} - \theta_i^j = 1$, then each state of the SAC model describes a molecular count, resulting in the same precision in the state space of species s_i as the FSS model.

In order to identify the critical levels of species s_i, our method first automatically finds all reactions with kinetic rate laws that include a denominator term of the form $K\,|s_i|^n$. For each such reaction, one critical level of s_i is generated with the form $\sqrt[n]{a/(K - aK)}$ where a is an amplifier in the range $[0.5, 1.0)$ selected by the user. Figure 5.7(a) shows two reactions that have kinetic rate laws containing $|s_1|$ terms. Assuming that amplifier a equals 0.5, these two reactions imply the following four critical levels:

$$0, \tfrac{k_{-4}}{k_4}, \tfrac{k_{-2}}{k_2 \cdot RNAP_{tot}}, \text{ and } \tfrac{k_{-3}}{k_3}. \tag{5.24}$$

These levels come from the fact that θ_0 is by definition 0, the denominator of the left reaction rate law in Figure 5.7(a) has the term $k_4/k_{-4}\,|s_1|$, and the denominator of the right reaction rate law has two terms of this form, $k_2/k_{-2}\,|s_1|\,RNAP_{tot}$ and $k_3/k_{-3}\,|s_1|$.

After the critical levels of each species are identified and in turn $\mathbf{z_0}$ and $\mathbf{z_{max}}$ are all determined, the guard, $G_\mu(\mathbf{z})$, for c_μ is generated for each reaction in a similar way as the FSS model transformation. Suppose species s_1 is an activator in reaction r_1 for the production of s_2 as shown in Figure 5.7(b) where its kinetic law, $f(|s_1|)$, is always greater than 0 if $|s_1| \geq 0$. Also, suppose that three critical levels are used for both species s_1 and s_2, that is, the critical levels of s_1 and s_2 are $(0, \theta_1^1, \theta_1^2)$, and $(0, \theta_2^1, \theta_2^2)$, respectively.

In the n-ary transformation, definition of $\mathbf{\Sigma_{\hat{z}_j}}$ is slightly different from the FSS model transformation so that

$$\begin{aligned}
\mathbf{\Sigma_{\hat{z}_j}} \equiv \{\mathbf{z} = (&\mathit{if}\ i \in \mathbf{I_j}\ \mathit{then}\ z_i \in [0, z_{max}^i]\ \mathit{else}\ z_i = 0)\} \\
&\cap \{\mathbf{z} \in \mathbf{\Sigma_z} \mid \mathbf{K}(r_j)(\mathbf{z}) > 0\} \\
&\cap \{\mathbf{z} \in \mathbf{\Sigma_z} \mid \forall i.\ z_i \leq z_{max}^i - [[\mathbf{E}(r_j, s_i) - \mathbf{E}(s_i, r_j) > 0]]\} \\
&\cap \{\mathbf{z} \in \mathbf{\Sigma_z} \mid \forall i.\ z_i \geq [[\mathbf{E}(s_i, r_j) - \mathbf{E}(r_j, s_i) > 0]]\}.
\end{aligned} \tag{5.25}$$

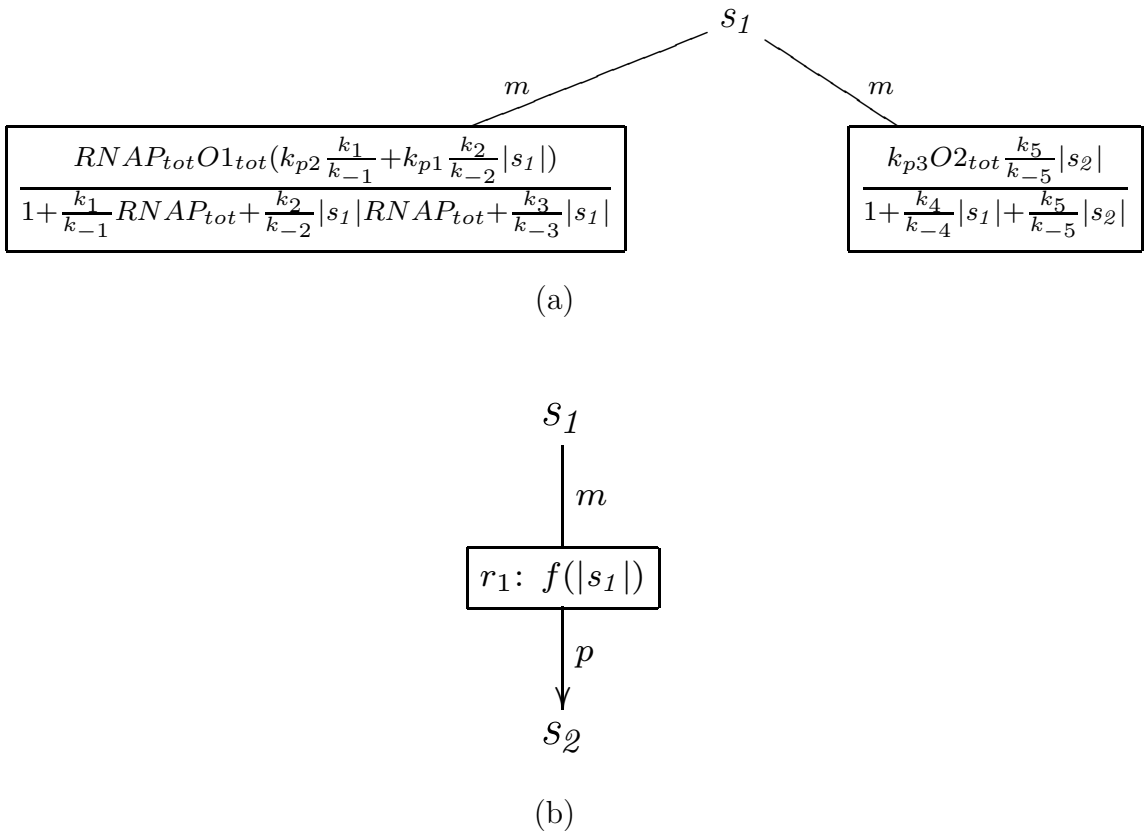


Figure 5.7. (a) Critical level identification. (b) Production of s_2 with activator s_1. $f(|s_1|) > 0$ if $|s_1| \geq 0$.

Thus, since r_1 is a production reaction for species s_2 where $\mathbf{\Sigma_{\hat{z}_1}}$ is obtained as:

$$\mathbf{\Sigma_{\hat{z}_1}} = \{(z_1, z_2, \ldots, z_n) \mid z_1 \in [0, 2] \ and \ \mid z_2 \in [0, 1]\}, \tag{5.26}$$

the legal transition that Z_2 can take are only two: $0 \rightarrow 1$ and $1 \rightarrow 2$ for each possible state of Z_1. Therefore, the guarded commands for r_1 are below:

$$[[z_2 = 0]] \times [[z_1 = 0]] \xrightarrow{q_1} \mathbf{Z} = \mathbf{z} + \mathbf{u}$$
$$[[z_2 = 0]] \times [[z_1 = 1]] \xrightarrow{q_2} \mathbf{Z} = \mathbf{z} + \mathbf{u}$$
$$[[z_2 = 0]] \times [[z_1 = 2]] \xrightarrow{q_3} \mathbf{Z} = \mathbf{z} + \mathbf{u}$$
$$[[z_2 = 1]] \times [[z_1 = 0]] \xrightarrow{q_4} \mathbf{Z} = \mathbf{z} + \mathbf{u}$$
$$[[z_2 = 1]] \times [[z_1 = 1]] \xrightarrow{q_5} \mathbf{Z} = \mathbf{z} + \mathbf{u}$$
$$[[z_2 = 1]] \times [[z_1 = 2]] \xrightarrow{q_6} \mathbf{Z} = \mathbf{z} + \mathbf{u}$$

where $\mathbf{u} \equiv (u_1, \ldots u_n)$ has $u_2 = 1$ and $\forall i \neq 2.\ u_i = 0$.

The final step to generate a SAC model is to assign a transition rate, q_i, to each guarded command. For simplicity, N_i Boolean variables $B_i^1 \ldots B_i^{N_i}$ are introduced for the generation of the rate to change the state of Z_i. The relationship between Z_i and $B_i^1 \ldots B_i^{N_i}$ is:

$$Z_i(t) = z \text{ iff } (\forall j \in [1, z].\ B_i^j(t) = 1) \wedge (\forall j \in [z+1, N_i].\ B_i^j(t) = 0). \tag{5.27}$$

Thus, the time evolution of $|s_i|$ can be approximated using $B_i^1 \ldots B_i^{N_i}$ as

$$|s_i|(t) \approx (\theta_i^{N_i} - \theta_i^{N_i-1})B_i^{N_i}(t) + \cdots + (\theta_i^2 - \theta_i^1)B_i^2(t) + (\theta_i^1 - \theta_i^0)B_i^1(t). \tag{5.28}$$

Taking the derivative of the mean of $|s_i|(t)$ with respect to the mean of $B_i^j(t)$ results in:

$$\frac{\partial\langle |s_i|(t)\rangle}{\partial\langle B_i^j(t)\rangle} \approx (\theta_i^j - \theta_i^{j-1}). \tag{5.29}$$

Using this approximation, the time derivative of $\langle B_i^j \rangle$ is:

$$\frac{d\langle B_i^j(t)\rangle}{dt} = \frac{\partial\langle B_i^j(t)\rangle}{\partial\langle |s_i|(t)\rangle}\frac{d\langle |s_i|(t)\rangle}{dt} \approx \frac{1}{\theta_i^j - \theta_i^{j-1}}\frac{d\langle |s_i|(t)\rangle}{dt}. \tag{5.30}$$

Notice $\langle B_i^j(t)\rangle$ is a continuous variable in the range $[0, 1]$. By letting $\langle B_i^j(t)\rangle$ be the probability that $B_i^j = 1$ at t, our method finds the transition rate functions for B_i^j to move from 0 to 1 and from 1 to 0 from the rate laws of reactions that change

the value of $|s_i|$. The transition rate function of a guarded command changing the value of B_i^j, which is generated from reaction r, is:

$$f = \frac{E \cdot \mathbf{K}(r)}{\theta_i^j - \theta_i^{j-1}} \quad \text{where } E = \begin{cases} \mathbf{E}(s_i, r) & \text{if } s_i \text{ is a reactant of } r, \\ \mathbf{E}(r, s_i) & \text{if } s_i \text{ is a product of } r. \end{cases} \tag{5.31}$$

Finally, our method must evaluate the transition rate functions with appropriate values to generate the transition rates. Suppose reaction r_j uses $|s_i|$ in its kinetic law. Then, to generate the transition rate for the guarded command when $Z_i = z$, our method uses θ_i^z as the value of $|s_i|$ to evaluate $\mathbf{K}(r_j)$. For example, the transition rates of the guarded commands in Figure 5.7(b) are derived from $\mathbf{K}(r_1)$. Since the derived transition rate function is $f(|s_1|)/(\theta_2^\mu - \theta_2^{\mu-1})$ for $\mu \in [1, 2]$, the transition rates for the guarded commands for reaction r_1 are:

$$\begin{aligned} q_1 &= f(0)/\theta_2^1, \\ q_2 &= f(\theta_1^1)/\theta_2^1, \\ q_3 &= f(\theta_1^2)/\theta_2^1, \\ q_4 &= f(0)/(\theta_2^2 - \theta_2^1), \\ q_5 &= f(\theta_1^1)/(\theta_2^2 - \theta_2^1), \\ q_6 &= f(\theta_1^2)/(\theta_2^2 - \theta_2^1). \end{aligned}$$

CHAPTER 6

MODEL ABSTRACTION RESULTS

As a case study, this chapter illustrates the application of our tool to various small systems to facilitate efficient analysis of temporal behavior. Section 6.1 describes an application of our abstraction methods to enzymatic reaction systems. Section 6.2 illustrates an application of the operator site reduction. The results from an application of the dimerization reduction are considered in Section 6.3. Finally, Section 6.4 exemplifies the application of the stoichiometry amplification. Each model used in this chapter is encoded in SBML [42] and simulated for 1,000 runs using an optimized SSA implementation within REB2SAC [74] on a 3GHz Pentium4 with 1GB of memory. Usefulness of model abstractions for each system is reported by comparing the speedup, which is measured based on the runtime of each original model, and the accuracy, which is measured by the mean and standard deviation of the time evolution.

6.1 Enzymatic Reaction Abstraction Results

This section presents the REB model abstraction results from various systems containing an enzymatic reaction scheme. This section first considers three models of the single enzymatic reaction system. It then considers the *enzymatic futile cycle* motif which can be ubiquitously seen in biological systems including GTPase cycles, mitogen-activated protein kinase cascades, and glucose mobilization [102]. Finally, it considers a more complex competitive enzymatic reaction.

6.1.1 Single Enzymatic Reaction

The single enzymatic reaction scheme shown in Figure 6.1 is first used to analyze the results of several enzymatic reaction abstraction methods. As with

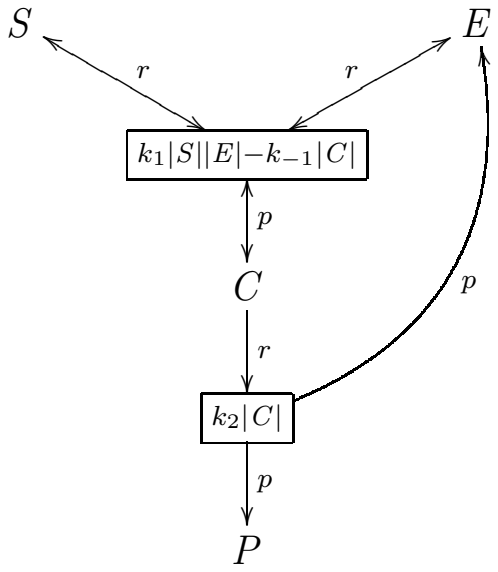


Figure 6.1. The single enzymatic reaction scheme.

most enzymatic reaction systems, this system has the following initial condition:

$$|E|_0 = E_{tot},\ |S|_0 = S_{tot},\ |C|_0 = 0,\ |P|_0 = 0. \tag{6.1}$$

Three models with different parameter value sets are generated from this system, each of which is simulated by applying both the QSSA and the PPTA methods to compare their results with those from the original model. In addition, in order to compare the speedup gained by our abstractions with that by the ssSSA, the first two models are chosen to be the ones that are used to help illustrate the application of the slow-scale SSA in enzymatic reaction systems [25].

The first model of the single enzymatic reaction scheme has the following initial condition and the reaction rate constants:

$$E_{tot} = 220,\ S_{tot} = 3000,\ k_1 = 0.01,\ k_{-1} = 100.0,\ k_2 = 0.01. \tag{6.2}$$

This system is simulated for 20,000 time units and each data point is plotted every 100 time units. Figure 6.2 shows the results from the original model, the QSSA model, the PPTA model, and the QSSA model of this system. The estimated

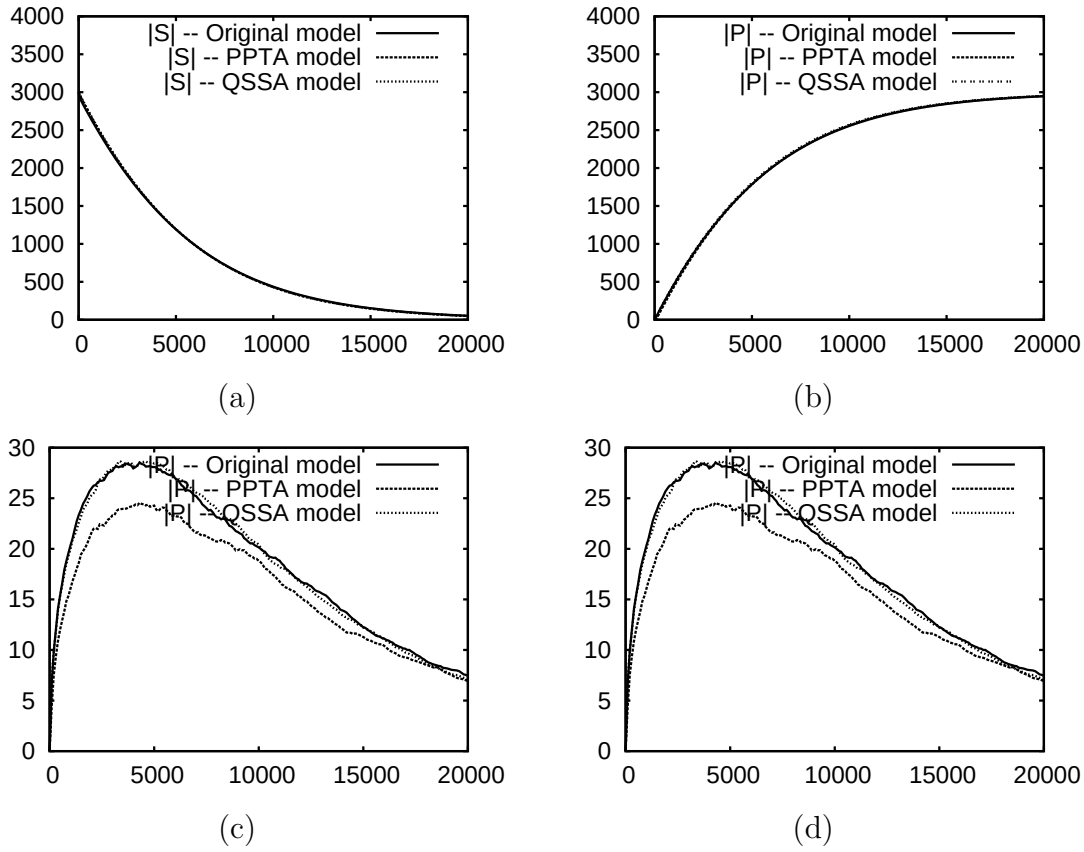


Figure 6.2. Comparison of the original enzymatic reaction model, its PPTA model, and its QSSA model with initial conditions: $E_{tot} = 220$, $S_{tot} = 3000$ and the rate constants: $k_1 = 0.01$, $k_{-1} = 100.0$, $k_2 = 0.01$. (a) Mean of $|S|$. (b) Mean of $|P|$. (c) Standard deviation of $|S|$. (d) Standard deviation of $|P|$.

means $|S|$ and $|P|$ are shown in Figures 6.2(a) and (b), and the estimated standard deviations of $|S|$ and $|P|$ are shown in Figures 6.2(c) and (d), respectively.

The simulation results from the QSSA model are in very close agreement with those from the original model. The average time evolution of the PPTA model is also in very close agreement with that of the original model. The standard deviation produced via the simulation of the PPTA is slightly lower than that of both the original model and the QSSA model throughout; however, considering the ratio of the standard deviations—which are relatively low—and the average molecular counts—which are very high—the results from the PPTA model are still very accurate. Both the QSSA and the PPTA result in substantial speedup as

shown in Table 6.1. While the entire simulation of the original model takes 68.58 hours, that of the PPTA model takes only 22.8 seconds, achieving 10,800 times speedup. The QSSA model produces an even higher speedup. It requires only 9.2 seconds for the simulation, resulting in 26,765 times speedup. Furthermore, since the speedup gained by the ssSSA is 950 on this model, both the PPTA and the QSSA methods are able to outperform the slow-scale SSA by an order of magnitude while maintaining a high degree of accuracy.

The second enzymatic reaction model has a lower total enzyme count, a higher complex dissociation rate constant, and a higher production rate constant as follows:

$$E_{tot} = 10,\ S_{tot} = 3000,\ k_1 = 0.01,\ k_{-1} = 600.0,\ k_2 = 0.1. \tag{6.3}$$

This model illustrates a case where the average of $|C|(t)$ remains less than 1 as the maximum reaction rate of r_1 (i.e., $k_1 E_{tot} S_{tot}$) is less than k_{-1}. This model is simulated for 80,000 time units and each data point is again plotted every 100 time units. Figures 6.3(a) and (b) show the estimated means of $|S|$ and $|P|$, and Figures 6.3(c) and (d) show the estimated standard deviations of $|S|$ and $|P|$, respectively.

Both the means and the standard deviations from the QSSA model as well as the PPTA model track those from the original model very well while, at the same

Table 6.1. Speedup gained by the ssSSA, the QSSA, and the PPTA on various systems involving enzymatic reaction. The results of the ssSSA are from Cao et al. (2005) [25].

		Original	**QSSA**	**PPTA**	**ssSSA**
Single enzymatic reaction 1	**Time**	68.58h	9.2s	22.8s	–
	Speedup	1	26,765	10,800	950
Single enzymatic reaction 2	**Time**	27.63h	8.5s	17.9s	–
	Speedup	1	11,582	5,500	400
Single enzymatic reaction 3	**Time**	34.69s	1.07s	1.72s	–
	Speedup	1	32	20	–
Enzymatic futile cycle	**Time**	17.73h	53.43s	87.51s	–
	Speedup	1	1,194	729	–
Competitive enzymatic reaction	**Time**	65.16m	63.24s	35.78s	–
	Speedup	1	62	109	–

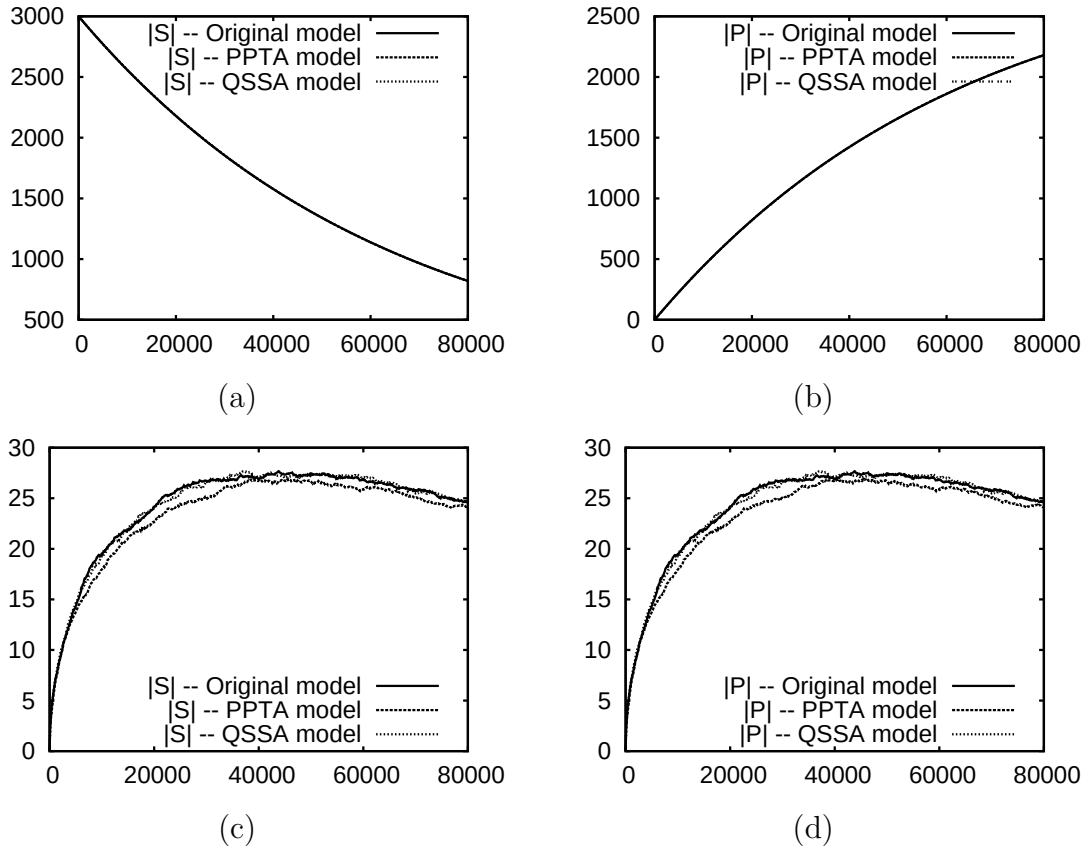


Figure 6.3. Comparison of the original enzymatic reaction model, its PPTA model, and its QSSA model with initial conditions: $E_{tot} = 10$, $S_{tot} = 3000$ and the rate constants: $k_1 = 0.01$, $k_{-1} = 600.0$, $k_2 = 0.1$. (a) Mean of $|S|$. (b) Mean of $|P|$. (c) Standard deviation of $|S|$. (d) Standard deviation of $|P|$.

time, the simulation time of those abstractions are substantially reduced compared with that of the original model as shown in Table 6.1. Whereas the simulation of the original model takes 27.63 hours, that of the QSSA model and the PPTA model takes only 8.5 seconds and 17.9 seconds, respectively. Thus, the QSSA and the PPTA methods are able to improve the computation performance by a factor of 11,582 and 5,500, respectively. Furthermore, since the speedup of the ssSSA is only 400 on this model, both methods are once again able to outperform the ssSSA by an order of magnitude in terms of acceleration. Therefore, for the first two models of the single enzymatic reaction, the QSSA would be the most efficient and effective abstraction as it achieves the highest speedup while maintaining accuracy.

However, since a PPTA model does not assume that the intermediate species is in quasi-steady state, a PPTA model may perform better than a QSSA model in terms of accuracy, especially in a case where the pre-steady state transition is crucial for a prediction of system behavior. For example, suppose a single enzymatic reaction model has the following initial conditions and rate constants:

$$E_{tot} = 25,\ S_{tot} = 50,\ k_1 = 100.0,\ k_{-1} = 10.0,\ k_2 = 0.1. \tag{6.4}$$

Then, since E_{tot} is arguably much smaller than S_{tot}, the QSSA *could* be applied to safely approximate the temporal behavior of the underlying enzymatic reaction. However, in this model, the propagation effects of the pre-steady state dynamics are rather important, making any QSSA-based models unable to describe the temporal behavior well. Therefore, as shown in Figure 6.4, the averages as well as the standard deviations of the time evolution of this single enzymatic reaction system are captured more accurately by the PPTA model than by the QSSA model. While both model abstractions result in an order of magnitude speedup, the QSSA still achieves slightly higher speedup than the PPTA as shown in Table 6.1.

6.1.2 Enzymatic Futile Cycle

The enzymatic futile cycle system consists of two instances of the single enzymatic reaction as shown in Figure 6.5(a). One is to transform S into P catalyzed by E_1, and the other one is to transform P into S catalyzed by E_2. Thus, the original model of the enzymatic futile cycle system has six species and six irreversible reactions. Since this motif is found in many biological systems [102], abstracting away low-level detail of the motif such as unproductive substrate-complex cycles may provide a significant improvement in performance of the overall system behavior analysis. With the PPTA method, unproductive dissociation reactions are removed, transforming the enzymatic futile cycle model into a model with six species and four irreversible reactions shown in Figure 6.5(b). The QSSA further reduces the complexity of the enzymatic futile cycle system by removing the two enzymes and the two intermediate species, resulting in two species and two reactions as shown in Figure 6.5(c).

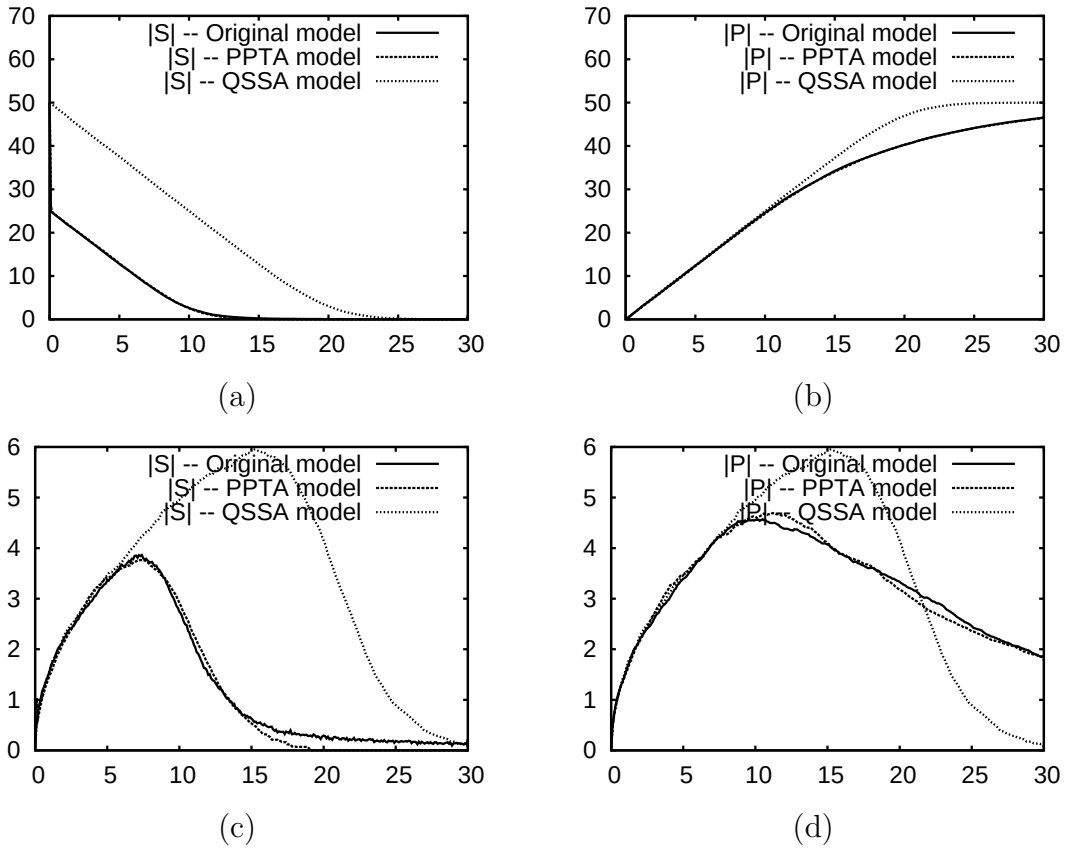
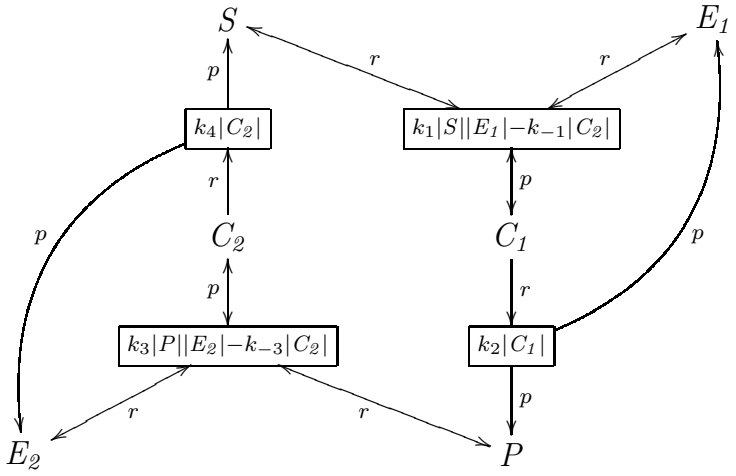
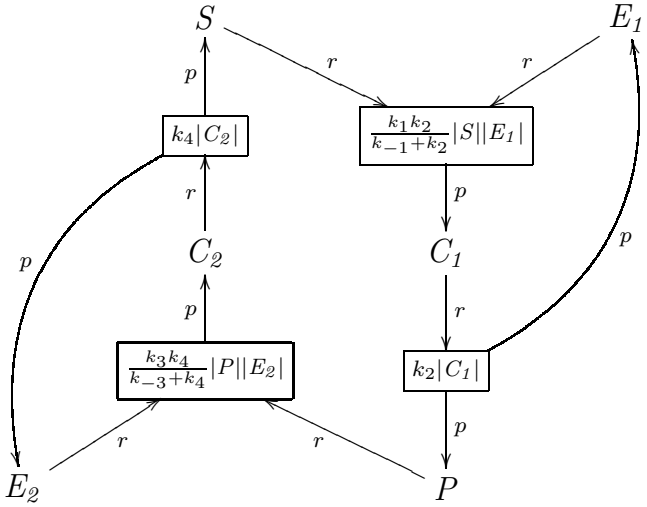


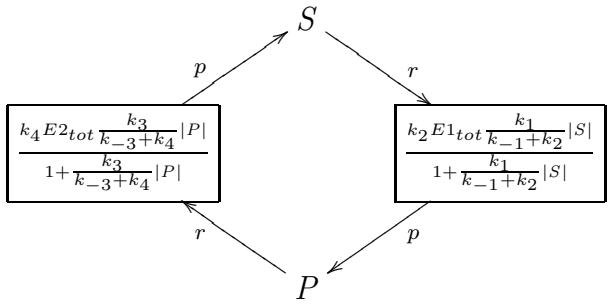
Figure 6.4. Comparison of the original enzymatic reaction model, its PPTA model, and its QSSA model with initial conditions: $E_{tot} = 25$, $S_{tot} = 50$ and the rate constants: $k_1 = 100.0$, $k_{-1} = 10.0$, $k_2 = 0.01$. (a) Mean of $|S|$. (b) Mean of $|P|$. (c) Standard deviation of $|S|$. (d) Standard deviation of $|P|$.



(a)



(b)



(c)

Figure 6.5. The enzymatic futile cycle system. (a) Original model (b) PPTA model (c) QSSA model.

The original enzymatic futile cycle model and the two abstracted models are simulated for 300 time units with one time unit plot-interval to analyze the accuracy as well as the performance gain of the enzymatic reaction model abstractions with the initial conditions:

$$(|S|_0, |P|_0, |E_1|_0, |E_2|_0, |C_1|_0, |C_2|_0) = (0, 100, 10, 20, 0, 0), \tag{6.5}$$

and the rate constants:

$$k_1 = 10^3; k_{-1} = 1.5 \times 10^3; k_2 = 2; k_3 = 10^3; k_{-3} = 5 \times 10^2; \text{ and } k_4 = 1. \tag{6.6}$$

Since each numerical simulation of the two models starts with no copies of S and 10 copies of E_1, this system illustrates a case where substrate is initially lower than the catalyzing enzyme. Furthermore, since this is a closed system, from the conservation law, the condition:

$$|S|(t) + |P|(t) + |C_1|(t) + |C_2|(t) = 100 \tag{6.7}$$

is satisfied for all $t \geq 0$. Thus, this enzymatic futile cycle system illustrates an applicability of the PPTA as well as the QSSA when the numbers of both substrate and enzyme molecules are very low.

Figure 6.6 shows the results from the original model, the QSSA model, and the PPTA model of this enzymatic futile cycle system. The time evolutions of the estimated means of $|S|$ and $|P|$ are shown in Figures 6.6(a) and (b), while the estimated standard deviations of $|S|$ and $|P|$ are shown in Figures 6.6(c) and (d), respectively. From these figures, it is clear that the PPTA model approximates the original enzymatic futile cycle model better than the QSSA model. The QSSA model, however, has a better improvement in the computational performance than the PPTA model as shown in Table 6.1. While the simulation of the original model takes 17.73 hours, that of the PPTA model and the QSSA model takes only 87.51 seconds and 53.43 seconds, which represents speedup factors of more than 729 times and 1,194 times, respectively.

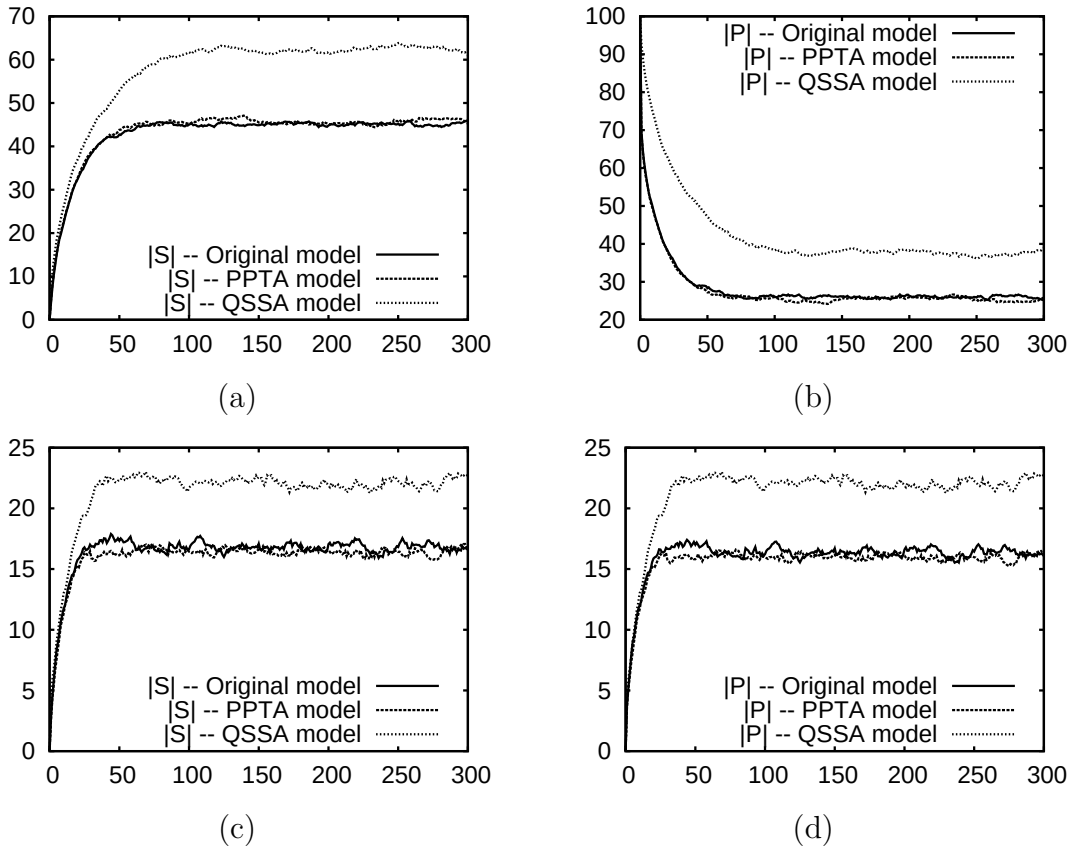
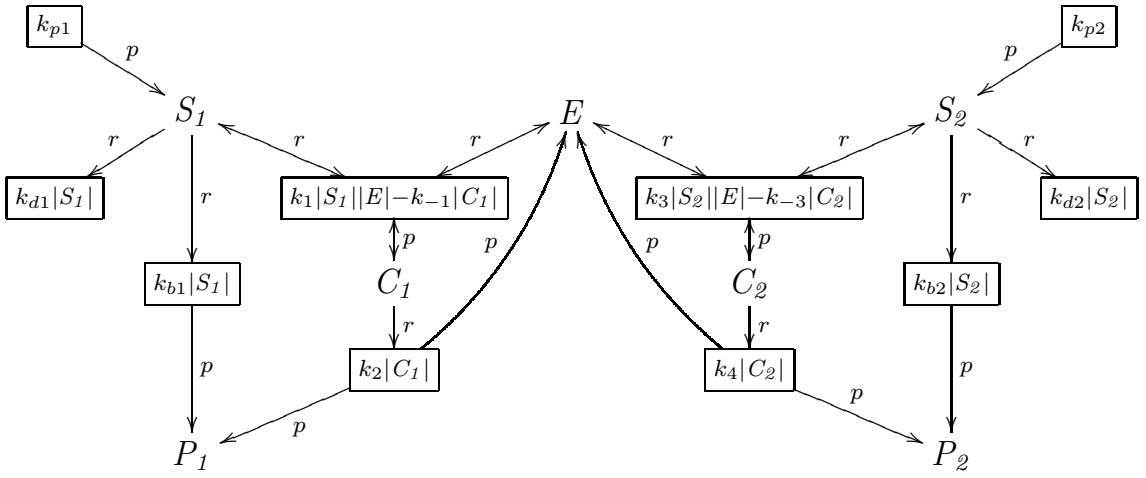


Figure 6.6. Comparison of the original enzymatic futile cycle model, its PPTA model, and its QSSA model. The initial conditions: $|S|_0 = 0$, $|P|_0 = 100$, $|E_1|_0 = 10$, $|E_2|_0 = 20$, $|C_1|_0 = 0$, $|C_2|_0 = 0$, and the rate constants: $k_1 = 1.0 \times 10^3$, $k_{-1} = 1.5 \times 10^3$, $k_2 = 2.0$, $k_3 = 1.0 \times 10^3$, $k_{-3} = 5.0 \times 10^2$, $k_4 = 1.0$ are used. (a) Mean of $|S|$. (b) Mean of $|P|$. (c) Standard deviation of $|S|$. (d) Standard deviation of $|P|$.

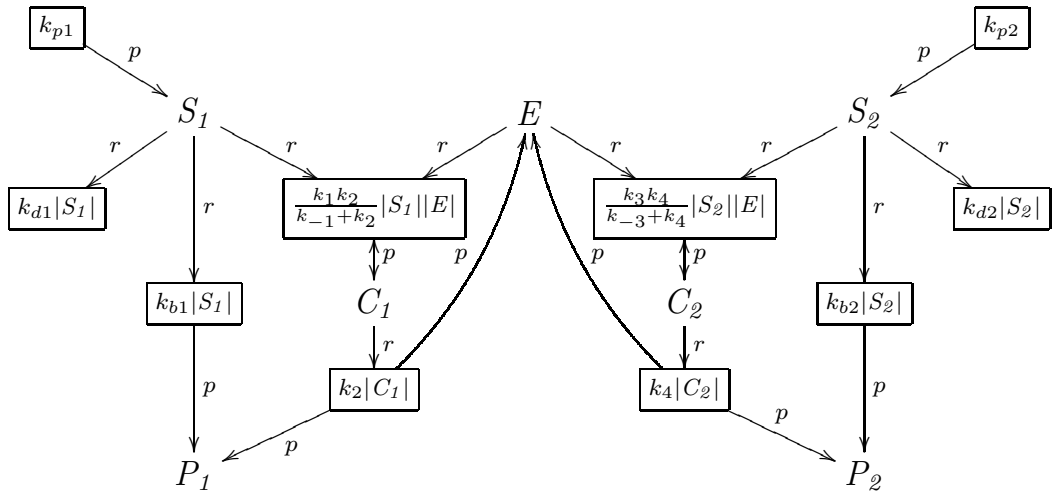
6.1.3 Competitive Enzymatic Reaction

To show the applicability of the abstraction methods in a more complex enzymatic reaction system, a biochemical system with seven species and 12 irreversible reactions whose graphical representation is shown in Figure 6.7(a) is considered.

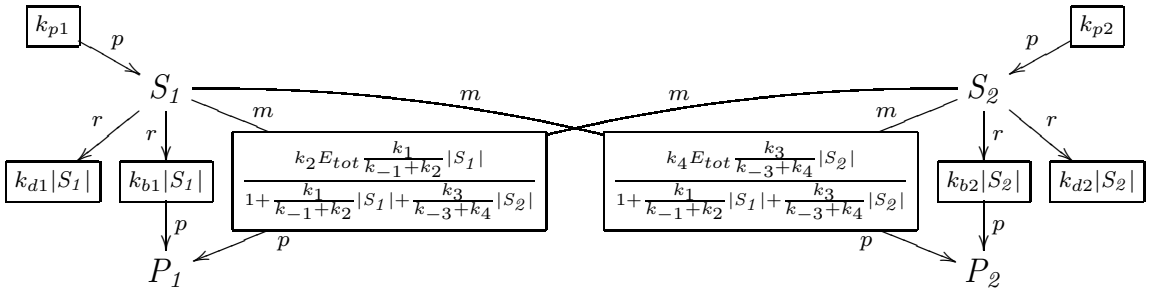
This system includes a competitive enzymatic reaction where the two substrates, S_1, and S_2, compete to bind to enzyme, E, to produce products, P_1, and P_2, respectively. Also, this system contains basal reactions to transform S_1 and S_2 into P_1 and P_2, respectively, without being catalyzed by E. Moreover, since substrates



(a)



(b)



(c)

Figure 6.7. A biochemical system with a competitive enzymatic reaction. (a) Original model (b) PPTA model (c) QSSA model.

S_1 and S_2 are often produced and consumed via various reactions, this system contains reactions to model productions and consumptions of S_1 and S_2. The PPTA method of this system removes the substrate-dissociation reactions from C_1 and C_2 as illustrated in Figure 6.7(b), resulting in a model with seven species and 10 irreversible reactions. The QSSA method further reduces the complexity of the system by removing more species and reactions in the competitive enzymatic reaction as illustrated in Figure 6.7(c), resulting in a model with four species and eight irreversible reactions.

The three models of this system are simulated using the following initial conditions:

$$(|E|_0, |S_1|_0, |S_2|_0, |P_1|_0, |P_2|_0, |C_1|_0, |C_2|_0) = (10, 0, 0, 0, 0, 0, 0), \tag{6.8}$$

and the rate constants:

$$\begin{aligned} &k_{b1} = 2 \cdot 10^{-5}; k_1 = 10^2; k_{-1} = 10^2; k_2 = 0.1; k_{p1} = 10; k_{d1} = 0.2; \\ &k_{b2} = 10^{-5}; k_3 = 200; k_{-3} = 10^2; k_4 = 0.15; k_{p2} = 10; \text{ and } k_{d2} = 0.2. \end{aligned} \tag{6.9}$$

The values of rate constants for the productions and consumptions of S_1 and S_2 are chosen so that the consumption rate constants are relatively high to capture isolation of substrates from binding to the enzyme and so that both substrates are present in low counts throughout the simulations. In other words, for all $t \geq 0$,

$$\langle |S_1|(t) \rangle \leq 100 \text{ and } \langle |S_2|(t) \rangle \leq 100. \tag{6.10}$$

The values of basal transformation rate constants k_{b1} and k_{b2} are chosen so that basal transformation rates are much smaller than those from the catalyzed reactions when the substrates are present in low counts. In other words,

$$k_2 E_{tot} \gg 100 k_{b1} \text{ and } k_4 E_{tot} \gg 100 k_{b2}. \tag{6.11}$$

Figure 6.8 shows the results from the simulations of the three models. The estimated means of $|S_1|$ and $|S_2|$ are shown in Figures 6.8(a) and (b), while the estimated standard deviations of $|S_1|$ and $|S_2|$ are shown in Figures 6.8(c) and (d). In this system, both the means and the standard deviations of $|S_1|$ and $|S_2|$

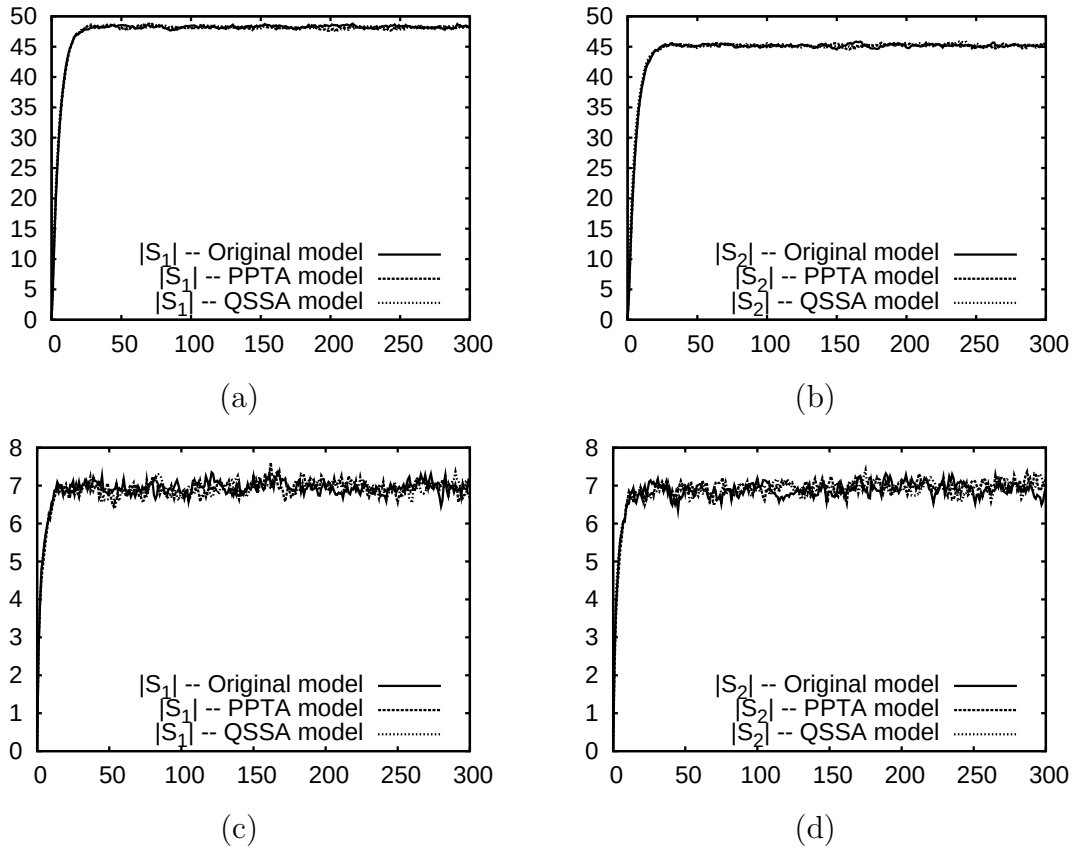


Figure 6.8. Comparison of the original model of competitive enzymatic reaction model, its PPTA model, and its QSSA model. The initial conditions: $|E|_0 = 10$ and 0 molecule for the rest of the species, and the rate constants: $k_{b1} = 2 \cdot 10^{-5}$, $k_1 = 10^2$, $k_{-1} = 10^2$, $k_2 = 0.1$, $k_{p1} = 10$, $k_{d1} = 0.2$, $k_{b2} = 10^{-5}$, $k_3 = 200$, $k_{-3} = 10^2$, $k_4 = 0.15$, $k_{p2} = 10$, $k_{d2} = 0.2$ are used. (a) Mean of $|S_1|$. (b) Mean of $|S_2|$. (c) Standard deviation of $|S_1|$. (d) Standard deviation of $|S_2|$.

from the PPTA model and the QSSA model track those from the original model very well with a substantial improvement in simulation time as shown in Table 6.1. Unlike the other enzymatic reaction systems presented in this section, however, the PPTA model achieves a higher speedup compared with the QSSA model in this system. While the simulation of the the QSSA model takes 63.24 seconds achieving 62 times speedup compared with that of the original model, the simulation of the PPTA model takes only 35.78 seconds, achieving 109 times speedup. Thus, for this competitive enzymatic reaction model, the PPTA is able to outperform the

QSSA model in terms of speedup. One interpretation of this result is that, while the QSSA model can advance the time step further in each reaction event than the PPTA model, the evaluations of the kinetic laws of the reduced competitive enzymatic reactions in the QSSA require higher computational costs due to the complexity of the denominator term, resulting in just enough overhead for the PPTA model to outperform the QSSA model.

6.2 Operator Site Reduction Results

This section illustrates the application of the operator site reduction. For this analysis, a system for a synthesis of protein P based on the binding configuration of the operator site, O, is considered. In this system, the operator site is assumed to have five operator-binding states, $C_i,\ i \in [0, 4]$, each of which specifies whether or not its configuration based on an occupation of two repressors, R_1 and R_2, one activator, A, and $RNAP$ on the operator region can lead to a production of P as depicted in Table 6.2. Here, the operator state C_0 represents the state in which O is not bound to any species. When the operator is in this state, P cannot be synthesized. Binding of A and $RNAP$ to O forms an operator complex C_1 which can lead to a synthesis of P. A synthesis of P is also possible from a complex C_2 which is formed by binding of $RNAP$ to O. The binding of repressor species to O

Table 6.2. Configuration of the operator site for a case study of the operator site reduction. Each state specifies if a given species is bound (i.e., ∘) or not bound (i.e., –). It also specifies if that operator state can lead to a synthesis of protein or cannot lead to any protein production (i.e., ×).

State	A	$RNAP$	R_1	R_2	**Synthesis**
C_0	–	–	–	–	×
C_1	∘	∘	–	–	P
C_2	–	∘	–	–	P
C_3	–	–	∘	–	×
C_4	–	–	–	∘	×

forms complexes which prevent A or $RNAP$ from binding to O. Thus, the operator states, C_3, and C_4, do not lead to a production of P.

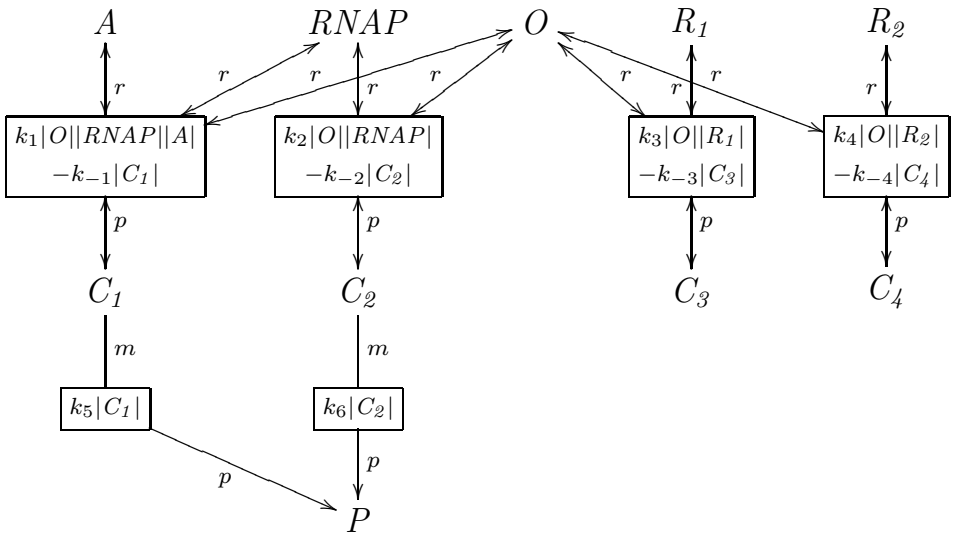
This system is first modeled using only elementary reactions as shown in Figure 6.9(a). This model has 10 species and 10 irreversible reactions. By applying our operator site reduction to this model, the size of the model is then reduced to five species and two irreversible reactions as shown in Figure 6.9(b). These two models are simulated for $1,000$ time units. In order to analyze how the level of the activator influences the protein synthesis, each model is simulated for two different values of $|A|_0$. The first simulation has $|A|_0 = 10$ to examine a case where the activator level is low, while the second one has $|A|_0 = 100$ to examine a case where the activator level is high. For both sets of simulation, the rest of the species are initialized so that:

$$(|O|_0, |R_1|_0, |R_2|_0, |RNAP|_0) = (1, 50, 100, 30), \tag{6.12}$$

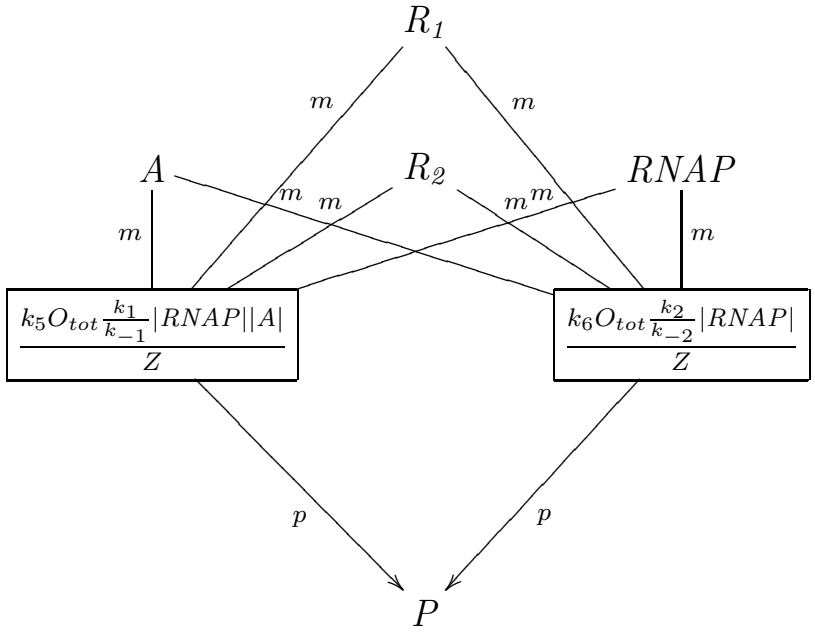
while the other species are initially set to 0. In each simulation, the values of the unbinding rate constants, $k\_i,\ i \in [1, 4]$ are all set to 1.0, while the other rate constants are defined as follows:

$$k_1 = 0.1; k_2 = 0.001; k_3 = 0.2; k_4 = 0.05; k_5 = 0.01; \text{ and } k_6 = 0.001. \tag{6.13}$$

Figure 6.10 shows the results from the simulation of the original model and the abstracted model for the $|A|_0 = 10$ case and the $|A|_0 = 100$ case. Figure 6.10(a) shows the mean time evolution of $|P|$ while Figure 6.10(b) shows the time evolution of the standard deviation of $|P|$. These results indicate that, for both levels of $|A|_0$ in this system, the abstracted model approximates the original model very well. Furthermore, the abstracted model is able to achieve significant acceleration for the simulation. While the entire simulation of the original model takes 25.66 seconds, that of the abstracted model takes only 0.55 seconds, which represents more than 46 times speedup.

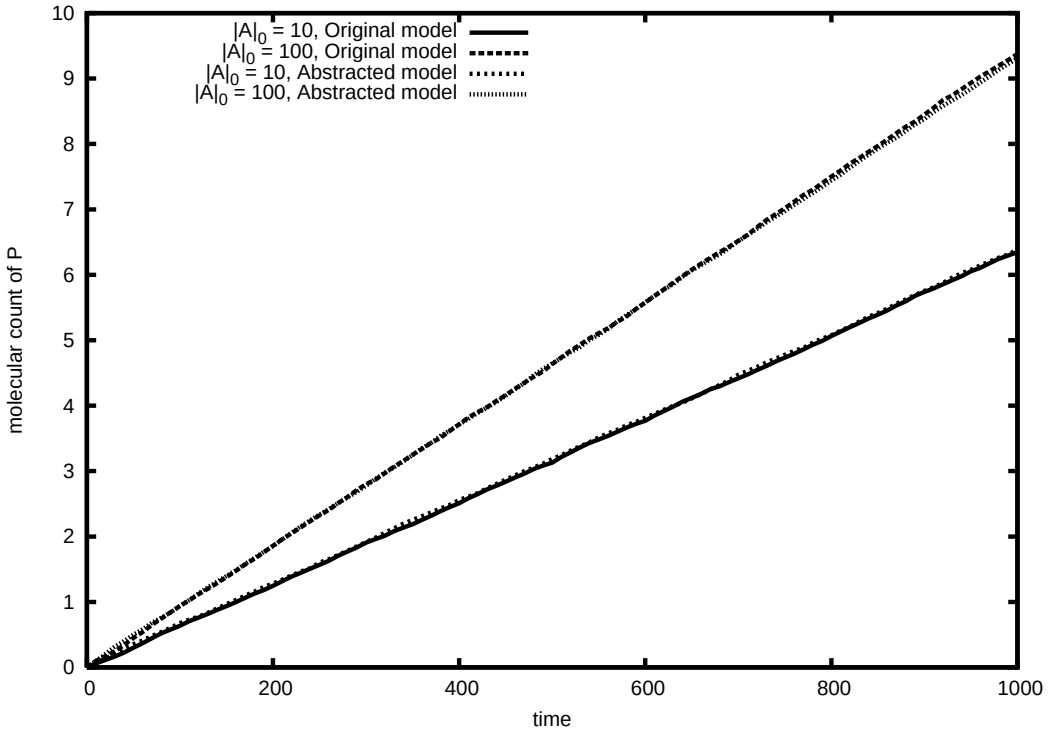


(a)

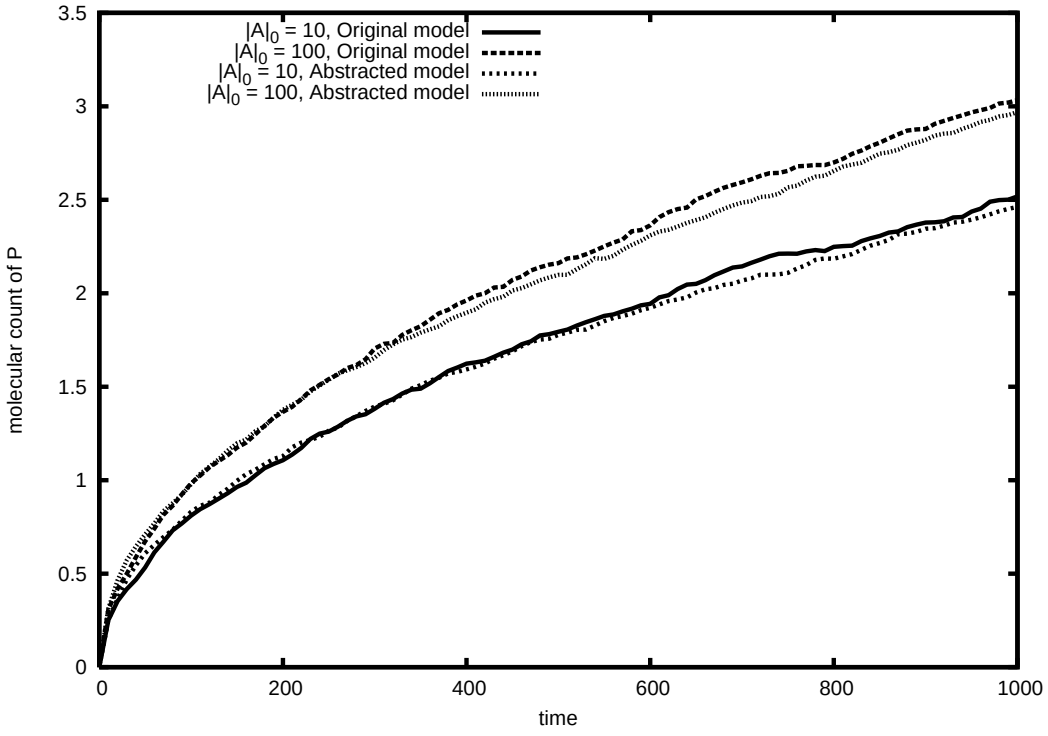


(b)

Figure 6.9. Operator site reduction of the system whose operator configuration is described in Table 6.2. (a) Original model and (b) abstracted model where $Z = 1 + \frac{k_1}{k_{-1}}|RNAP|\,|A| + \frac{k_2}{k_{-2}}|RNAP| + \frac{k_3}{k_{-3}}|R_1| + \frac{k_4}{k_{-4}}|R_2|$.



(a)



(b)

Figure 6.10. Simulation results of the original model and the abstracted model shown in Figure 6.9. (a) Mean of P and (b) standard deviation of P.

6.3 Dimerization Reduction Results

To illustrate the application of the dimerization reduction, consider the model shown in Figure 6.11(a). This model includes a dimerization reaction that produces a dimer, A_2, from a monomer, A, with the forward rate constant, k_f, and the reverse rate constant, k_r. In this model, only the monomer form of A can degrade with rate constant k_d. Also included in this model is a production of species P which uses A_2 as a modifier to enhance the rate of the production. This model has three species and four irreversible reactions. By applying the dimerization reduction, this model can be transformed to the model shown in Figure 6.11(b). This abstracted model expresses the states of the monomer and the dimer in terms of the total molecular counts in order to reduce the complexity of the model to two species and two irreversible reactions.

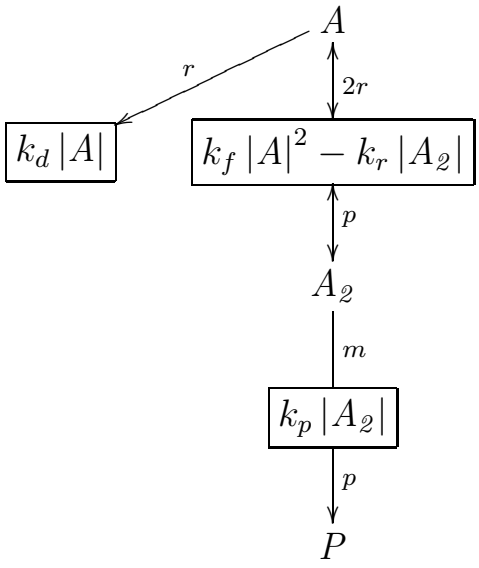
These two models are simulated for 100 time units to obtain the time evolution of P. Each simulation starts with the state where each of the initial molecular counts is given by:

$$(|A|_0, |A_2|_0, |P|_0) = (100, 0, 0). \tag{6.14}$$

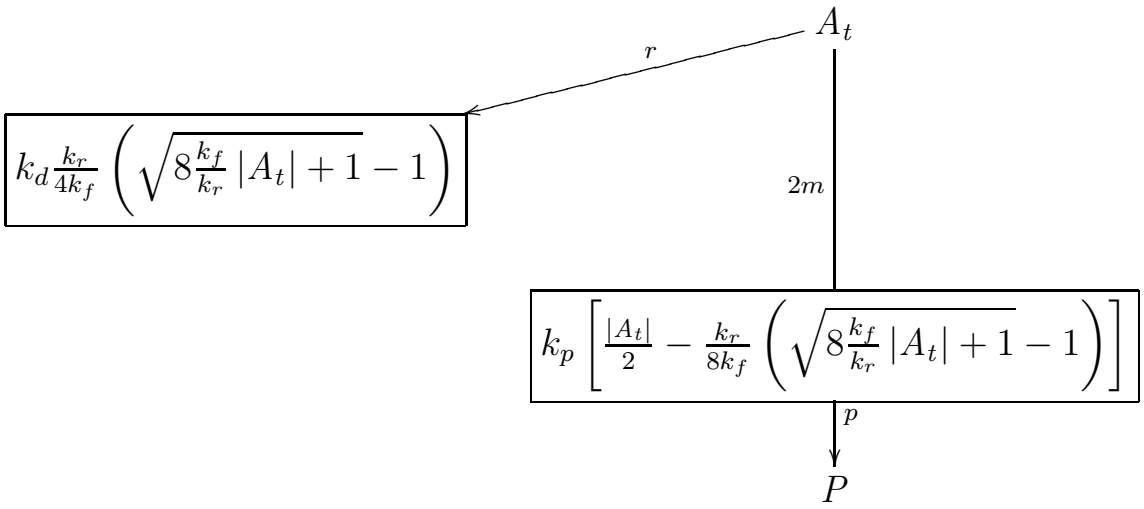
The parameters of the two models are first chosen to be as follows:

$$k_f = 1.0;\ k_r = 10.0;\ k_d = 0.05;\ \text{and}\ k_p = 0.01. \tag{6.15}$$

Then, another set of simulation runs is performed by changing k_f to be 10.0 for both the original model and the abstracted model. These results are shown in Figure 6.12. Figure 6.12(a) represents the time evolution of the average of P, while Figure 6.12(b) represents the mean time evolution of the standard deviation of P. For both values of k_f, the average behavior of P from the abstracted model tracks that from the original model very well. The standard deviation, on the other hand, may seem different. However, the shapes and the trends of the standard deviations obtained from the two models are very similar, and, considering the value of the standard deviations over the averages, the abstracted model approximates the original model relatively well. Furthermore, the speedup gained by the dimerization

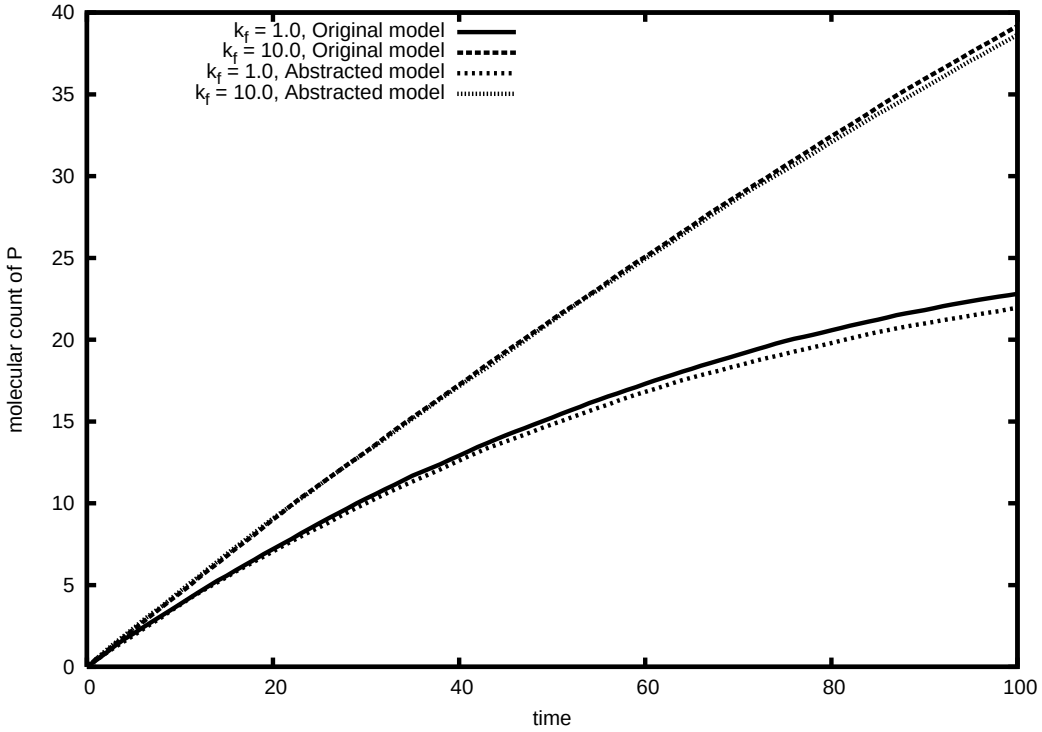


(a)

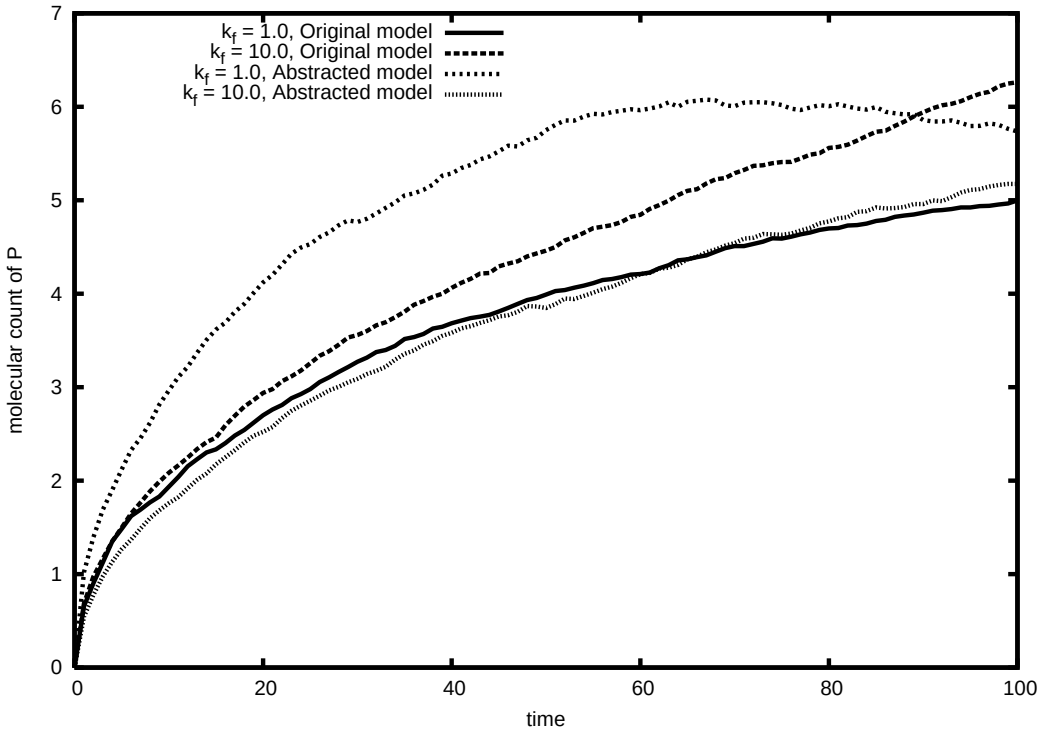


(b)

Figure 6.11. Illustration of the dimerization reduction. (a) Original model and (b) abstracted model.



(a)



(b)

Figure 6.12. Simulation results of the original model and the abstracted model shown in Figure 6.11. (a) Mean of P and (b) standard deviation of P.

reduction in this system is significant. While the entire simulation of the original model takes 675.97 seconds, that of the abstracted model takes only 2.08 seconds, achieving close to 325 times speedup.

6.4 Stoichiometry Amplification Results

This section exemplifies an application of the stoichiometry amplification by using the Lotka model [78] whose graphical representation is depicted in Figure 6.13. The Lotka model, which possesses remarkable dynamical properties, has been used for a case study of the SSA [53]. This model can be interpreted as a predator-prey ecosystem in which the behavior of two competing species is governed [115]. In this predator-prey interpretation, species Y_1 is viewed as a predator, while species Y_2 is viewed as a prey. Hence, the reaction that takes one molecule of Y_1 and one molecule of Y_2 to produce two molecules of Y_2 can be seen as a reproduction of a predator by feeding on a prey. Species $\bar{X}$ represents food for a prey species which is assumed to be controlled to be in a constant amount. Thus, the reaction that takes one molecule of Y_1 and one molecule of $\bar{X}$ to produce two molecules of Y_1 and one molecule of $\bar{X}$ can be seen as a reproduction of a prey by feeding on its food. Species Z represents the number of dead predators by natural causes. Thus, the reaction that takes one molecule of Y_2 to produce one molecule of Z can be viewed as the eventual demise of a predator.

This Lotka model is used to illustrate an application of the stoichiometry amplification, which is indicated by the amplification factor, a, in Figure 6.13. Three different amplification factors, 2, 5, 10, are used to compare the simulation results with the original model (i.e., $a = 1$). Each model is simulated for 0.1 time units from the initial state:

$$(\left|\bar{X}\right|_0, |Y_1|_0, |Y_2|_0, |Z|_0) = (1, 1000, 1000, 0). \tag{6.16}$$

The parameters of each model are set as follows:

$$k_1 = 10.0;\ k_2 = 0.01;\ \text{and}\ k_3 = 10.0. \tag{6.17}$$

The simulation results are shown in Figure 6.14. The time evolutions of the mean

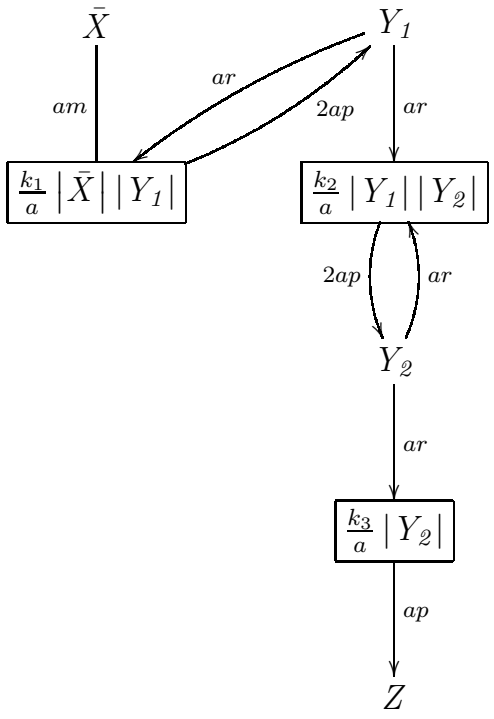
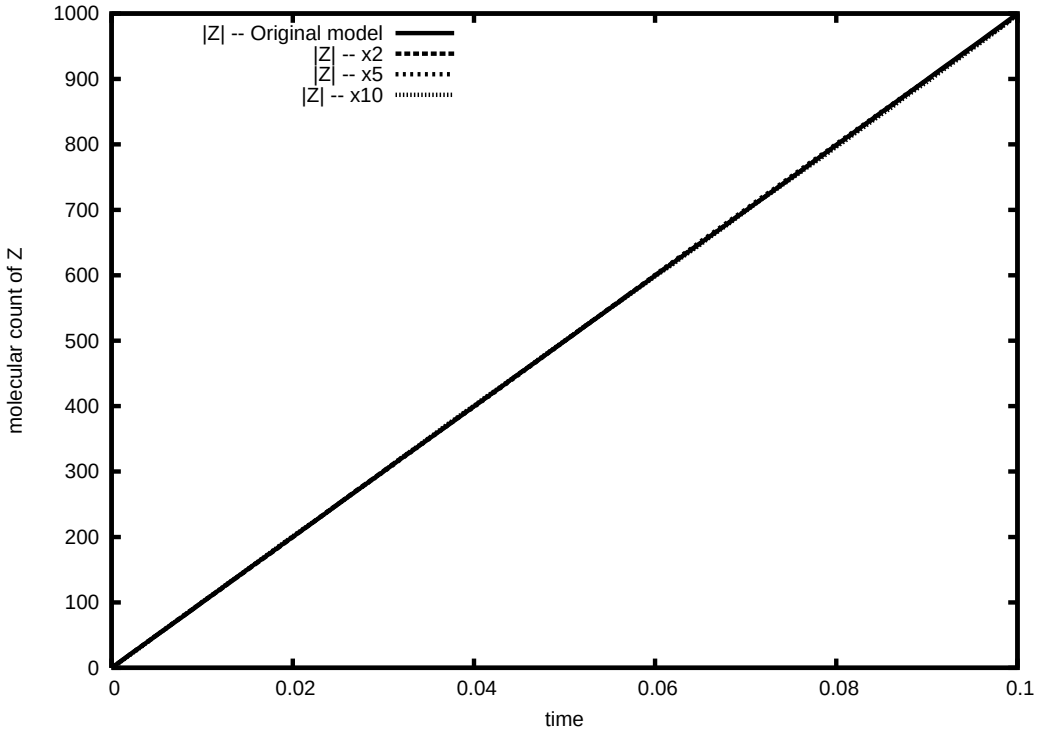
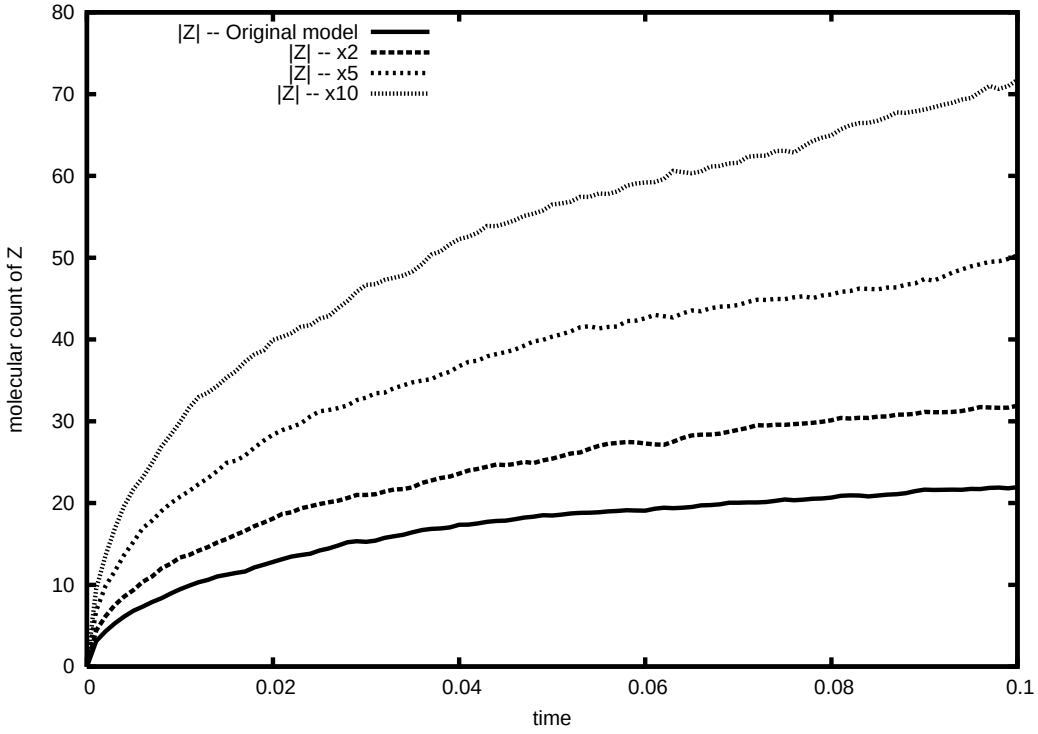


Figure 6.13. Stoichiometry amplification of Lotka model.



(a)



(b)

Figure 6.14. Simulation results of the Lotka model with the stoichiometry amplification. (a) Mean time evolution of Z (b) standard deviation.

and the standard deviation of Z for each model are presented in Figure 6.14(a) and Figure 6.14(b), respectively. It is clear from these figures that the mean time evolution of Z from each amplified model tracks that from the original model very well. The standard deviation of Z, on the other hand, has some perceptible differences between the amplified models and the original model. Figure 6.14(b) shows that, the higher the amplification factor is, the larger the standard deviation tends to be. However, the differences in the the standard deviation are very small considering the value of the mean as shown in Figure 6.15, which presents the ratio of the standard deviation and the mean over time. The computational performance gained by the stoichiometry amplification is significant. The entire simulation of the original model takes 11.11 seconds, while that of the stoichiometry of 2 takes only 5.90 seconds. The stoichiometry of 5 and 10 can further shorten the simulation time to be 2.71 seconds and 1.81 seconds, respectively. Thus, the speedup factor of the stoichiometry amplification on the Lotka model is roughly the same as the amplification factor.

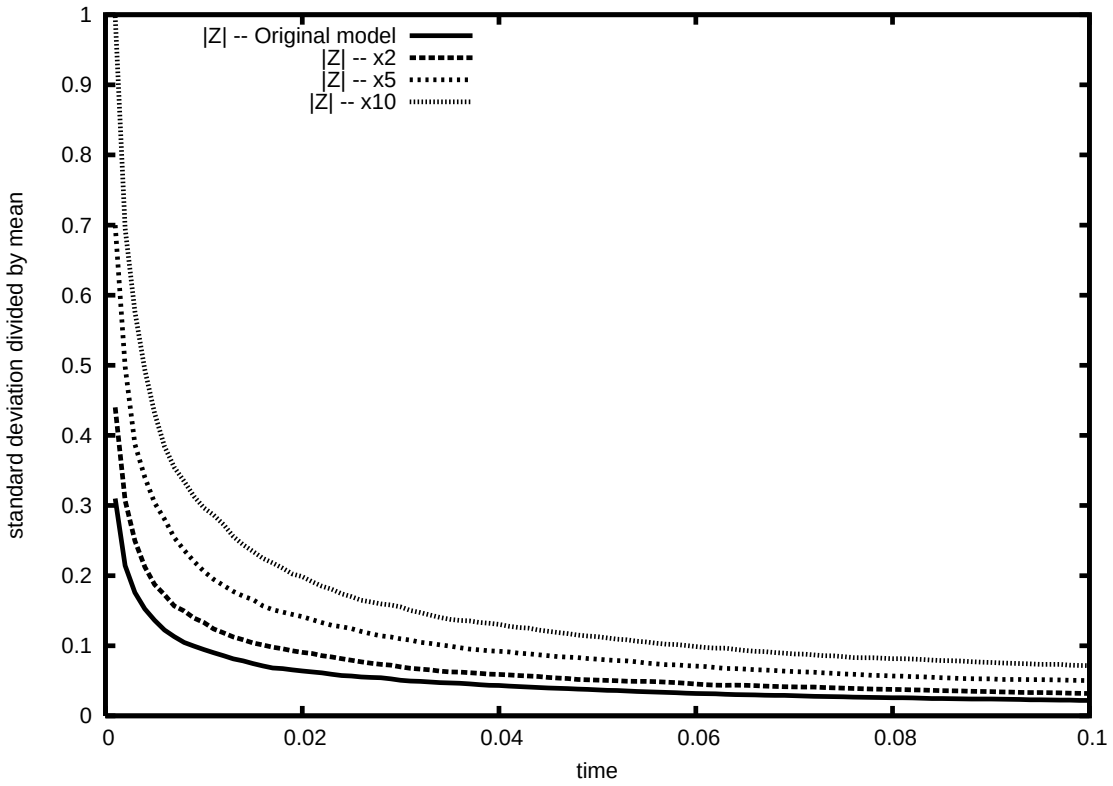


Figure 6.15. The ratio of the standard deviation and the mean of Z for each resolution of the Lotka model.

CHAPTER 7

GENETIC REGULATORY NETWORK ANALYSIS

Both the REB abstraction methods and the FSS abstraction methods coupled with temporal behavior analysis methods are implemented in our automated modeling and analysis tool called REB2SAC [73, 74] which is integrated within our BioSim tool [2] that provides a user-friendly graphical user interface. This chapter presents the application of our tool to the analysis of realistic genetic regulatory networks. Section 7.1 presents the analysis of temperature control in expression of type 1 pili using our model abstraction and compares computational (*in silico*) results with experimental results. Section 7.2 applies our tool to a larger genetic regulatory network model to examine the phage λ lysis/lysogeny decision switch. Models of both systems are available for download at [2].

7.1 Temperature Control in the *E. coli Fim* Switch

Type 1 pili are the foremost virulence factor in Uropathogenic *Escherichia coli* (*E. coli*), which is believed to be responsible for 70-90 percent of urinary tract infections [116]. The pili are 1-2μm long and 7nm-thick helical rods with a 3nm-wide tip, which contains two adapter proteins and adhesins capable of mediating *E. coli* attachment to the mannose-containing receptors found on the surface of many host tissue cells [116, 71]. Type 1 pili are thus thought to aid the infection and colonization process by enhancing the ability of *E. coli* to stick to host cells and by thus enabling them to colonize the bladder. However, while pili provide a means for infection, there are some disadvantages for *E. coli* in being piliated. For example, a highly piliated population leads to preferential activation of the host immune system, which can rapidly clear the infection.

The expression of type 1 pili in *E. coli* is phase variable, with cells randomly switching between *fimbriate* (ON) and *afimbriate* (OFF) phases based on environmental conditions [45, 82, 61]. This phase variation is driven by the inversion of a 314bp chromosomal region known as the fimbriation (*fim*) switch, which contains the promoter for *fimA* and other genes encoding structural pili subunits [5, 39, 82]. The inversion requires either *FimB* or *FimE* site-specific recombinases [69]. Whereas protein *FimB* promotes recombination with little orientational bias, protein *FimE* promotes recombination largely in the ON-to-OFF direction [19].

The empirical observations such as [45] revealed that the inversion of the *fim* switch is controlled by environmental conditions such as temperature and medium. Their analysis showed, among other things, that the wild-type *FimE*-dependent ON-to-OFF switching frequency decreases as the temperature increases in both defined-rich and minimal media, while the *FimB*-promoted switching frequency increases as the temperature increases from 28 °C to 37 °C, and then decreases as the temperature continues to increase from 37 °C to 42 °C in both media. The experimental results also indicate that the wild-type ON-to-OFF switching rate is much faster than *FimB*-promoted switching rate alone, allowing *E. coli* to rapidly undergo afimbriation under appropriate conditions.

Figure 7.1 shows the basic genetic network controlling type 1 fimbriae phase variation. When the switch is in the ON position, as shown in Figure 7.1, the transcription of genes *fimA* through *fimH* can be initiated since the corresponding promoter located within the *fim* invertible element is in the correct orientation. On the other hand, when the switch is in the OFF position, the promoter is in the opposite orientation, and the transcription of the corresponding genes cannot be initiated. Global regulator proteins, *Lrp*, *H-NS*, and *IHF* also play important roles in the regulation of the *fim* switch inversion system by, among other things, acting as sensors of the environmental conditions. For example, *H-NS* acts in a temperature-dependent manner when it binds to the regions containing promoters of genes *fimB* and *fimE* and represses the expression of those genes [88]. Additionally, *Lrp* binds to the 3 *Lrp*-sites and changes *fim* switching rates [17, 46, 99]. Since *H-NS* also down-

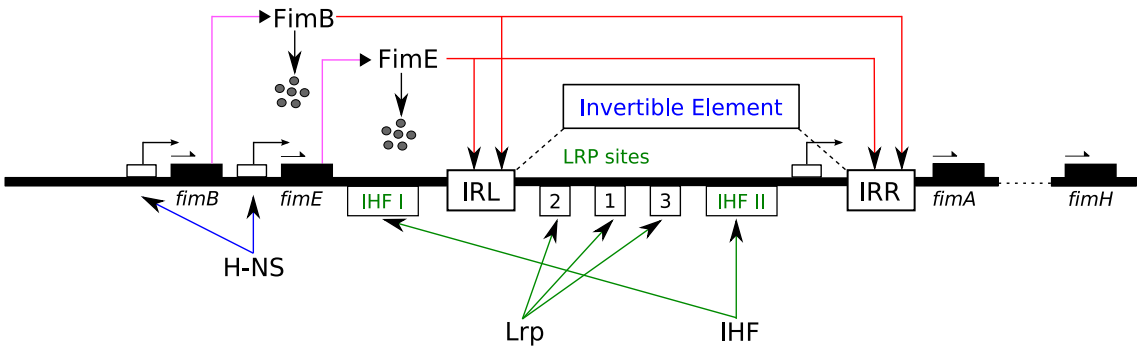


Figure 7.1. Type 1 pili genetic regulatory network (based on [116, 89, 99, 18]). Structural pili subunits are encoded by *fimA* and other downstream genes which are transcribed only when the *fim* switch is in the ON position. Recombinases *FimB* and *FimE* bind to the four adjacent half-sites and invert the switch with different rates. *FimE* is strongly biased in the ON-to-OFF direction, while *FimB* is close to fair. A small protein *H-NS* acts in a temperature-dependent manner and represses the expression of the two recombinases. *Lrp* stimulates and inhibits switching based on its occupancy at 3 *Lrp* sites [99]. It has been further proposed that *IHF* is needed for any observable phase variation as it plays a structural role during switching via the ability to introduce sharp bends into the DNA [18].

regulates the expression of *lrp* [91, 11], *Lrp* also acts effectively in a temperature-dependent manner. Finally, it is shown that *IHF* binds to both *IHF I* and *IHF II* and is required for any observable phase variation in part by playing a structural role in the switching via the ability to introduce sharp bends into the DNA [18].

Wolf and Arkin studied the behavior of the *fim* system and ascertained the importance of discrete and stochastic mechanisms in its dynamics [116]. In particular, the *fim* element inversion events are manifestly discrete and stochastic—randomly promoted by *FimB* and *FimE* bindings to the four adjacent DNA half-sites, *IRL* and *IRR*, and regulated by the corresponding *Lrp* or *IHF* occupancies of *cis*-regulatory genomic elements present in low integer counts. Our computational analysis is interested in examining how different temperature settings (28 °C, 37 °C, and 42 °C) quantitatively affect the wild-type and *fimB*-promoted ON-to-OFF inversion of the *fim* switch in minimal medium. The goal of our analysis is to quantitatively determine the effect of temperature on the ON-to-OFF switching probability both total and *FimB*-driven over one cell generation, and to compare it

with the experimental results in minimal medium [45]. To address the significant computational challenges of the quantitative analysis of this system, REB2SAC is utilized for system modeling and analysis at various levels of abstraction [74]. This analysis expands our previous work [75] which studied the effect of *H-NS* and *Lrp* levels on phase variation rates in *E. coli* using this model abstraction approach, demonstrating significant acceleration in the analysis time.

For our analysis, a detailed kinetic and thermodynamic reaction-level model of the *fim* switch inversion system is first developed. This models the regulation of *FimB* and *FimE* states and the switch inversion based on the configuration of various bindings of its regulatory proteins to the *fim* switch DNA region as described in Appendix A. Our switch inversion model then exactly simulates the discrete-stochastic behavior of this network at various temperature settings *in silico* to compare the results with those derived from empirical observations [45] by comparing the mechanism of ON-to-OFF switching probabilities. Our detailed model contains 31 species and 52 irreversible reactions, and has been simulated for 100,000 runs using our implementation of SSA for various temperature settings (i.e., 28 °C, 37 °C, and 42 °C). Each simulation starts with the switch in the ON position and is run for up to one cell generation of 20 minutes as in [116]. If the switch moves to the OFF position within this time limit, then the simulation is counted as an ON-to-OFF switching event. The ON-to-OFF switching probability is calculated as the number of the ON-to-OFF switching events divided by the total number of simulations with the same initial conditions. Alternatively, this could be viewed as computing the total ON-to-OFF switching probability by summing up the switching events through all possible transition states 3-8, while just the *FimB*-driven events only include transitions for states 4, 7, and 8. Our results are qualitatively and quantitatively consistent with those obtained with the empirical observations [45] (see Table 7.1). The simulation of this model takes 30.5 hours on a 3GHz Pentium4 with 1GB of memory.

To help substantially improve the speed and efficiency of the analysis of the switching probability, the original model can be transformed into an abstracted

Table 7.1. ON-to-OFF switching probability in minimal medium obtained with the empirical method [45] and the computational methods from the detailed model and the abstracted model at different temperatures. The results from the simulations are given for a 95% confidence interval calculated using the binomial distribution.

	probability per cell per cell generation (10^{-5}):		
	28 °C	37 °C	42 °C
Empirical results			
wild type	7,000	1,800	600
FimB-promoted	69 ± 26	110 ± 24	34 ± 28
Detailed model			
wild type	7,305 ± 161	2,002 ± 87	608 ± 48
FimB-promoted	65 ± 16	97 ± 19	40 ± 12
Abstracted model			
wild type	7,242 ± 161	2,000 ± 87	609 ± 48
FimB-promoted	71 ± 17	97 ± 19	49 ± 14

model by collectively using the abstraction methods within REB2SAC. For the abstraction of the *fim* switch inversion model, the abstraction engine in REB2SAC is configured as shown in Figure 7.2. First, the operator site reduction is applied to see if any transcriptional regulator binding/unbinding reactions can be reduced (line 1). With this abstraction method, the production reaction scheme of *FimB* and *FimE* (Figure A.1) as well as the *fim* element configuration subnetwork (Table A.3) can be reduced. Then, modifier constant propagation is applied to the model to see if any species can be removed (line 2). With this method, $|H\text{-}NS|$ and $|RNAP|$ can be replaced with constants whose values are set to the corresponding initial concentrations so that *H-NS* and *RNAP* are removed from the model.

Algorithm 7.1.1 *Model FimAbstractionEngine(Model M)*
1: $M \leftarrow OpSiteReduction(M)$
2: $M \leftarrow ModifierConstantProp(M)$
3: ***return*** M

Figure 7.2. Top level abstraction algorithm for *fim* switch inversion model.

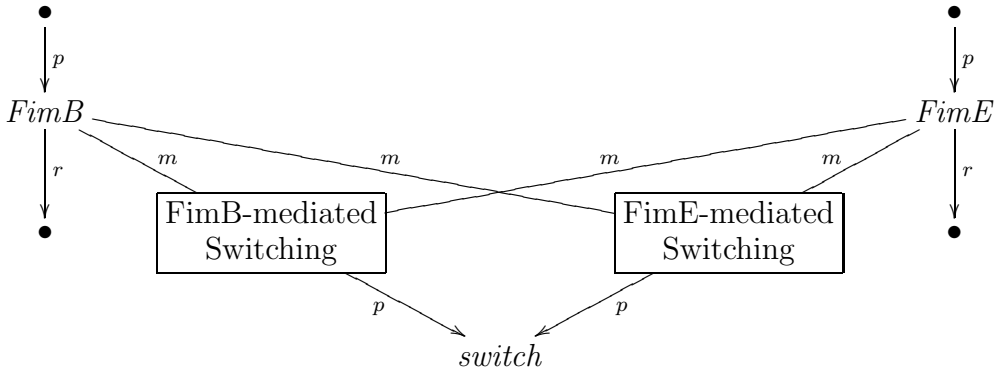


Figure 7.3. Abstracted model of the *fim* switch inversion model. It contains 3 species: *FimB*, *FimE*, and *switch*, and 10 irreversible reactions: 2 for *FimB* regulation, another 2 for *FimE* regulation, 3 reactions for *FimB*-mediated switching, and another 3 for *FimE*-mediated switching.

After applying the abstractions in Figure 7.2, our original model with 31 species and 52 irreversible reactions is transformed to a model with 3 species (i.e., *FimB*, *FimE*, and *switch*), and 10 reactions whose structure is shown in Figure 7.3. This highlights an additional benefit of abstraction in facilitating a higher level view of the network being analyzed, since it removes the low level details such as intermediate species and reactions that involve them. This makes it easier to visualize crucial interactions including identification of the key species that ultimately inhibit and/or activate transcription. The REB2SAC tool can also output the abstracted model as SBML to allow it to be visualized or further analyzed using any SBML compliant tool. Finally, REB2SAC can output the model in presentation MathML to visualize complex rate laws using an XML/HTML browser.

In order to compare the abstracted model with the original one, we have performed the same number of simulations using the same simulator on the same computer, and similarly computed the wild-type and *FimB*-promoted ON-to-OFF switching probabilities for one cell generation in minimal medium. The results from the abstracted model are in close agreement with those from the detailed

one as shown in Table 7.1, and thus are also closely comparable with the results from the empirical analysis [45]. However, the computational gains from the model abstraction are significant. The abstracted model simulation of 100,000 runs takes only 1.85 hours, which is a speedup of about 16 times compared with the runtime of the original model simulation.

The results from both the detailed and the abstracted model are generated via 100,000 simulation runs for each temperature setting, making the standard errors associated with the estimated wild-type ON-to-OFF switching probabilities in the range of 10^{-4} to 10^{-3}. To further improve the computational time of wild-type switching probability analysis without dealing with a statistical uncertainty involved in Monte Carlo simulation approach, a FSS model of the switch inversion system is automatically generated from the abstracted switch inversion model using REB2SAC. To allow for a relatively wide range of fluctuations in the levels of *FimB* and *FimE*, the upper limit molecular counts of the recombinases are set to be double their initial molecular counts (Table A.2). Since species *switch* only has two states, OFF and ON, the upper limit molecular count of this species is set to one. Then, the numbers of system states at the temperature setting of 28 °C, 37 °C, and 42 °C become 59,898, 25,326, and 13,446, respectively. The wild-type switching probability can then be analyzed by iterating the probability distribution for one cell generation time, and calculating the sum of the probabilities of the states where the *fim* switch is ON. Figure 7.4 shows the comparison of the results. The results are in very close agreement with those from the other computational model and, in turn, with the empirical observation from [45]. The entire FSS model analysis of the three temperature settings takes only 3 minutes, achieving a total speedup of more than 600 times compared with the 100,000 stochastic simulation runs of the original model.

7.2 Phage λ Developmental Pathway

Another genetic regulatory network to which we applied REB2SAC is the phage λ lysis/lysogeny developmental pathway in *E. coli*. Phage λ is a virus that infects

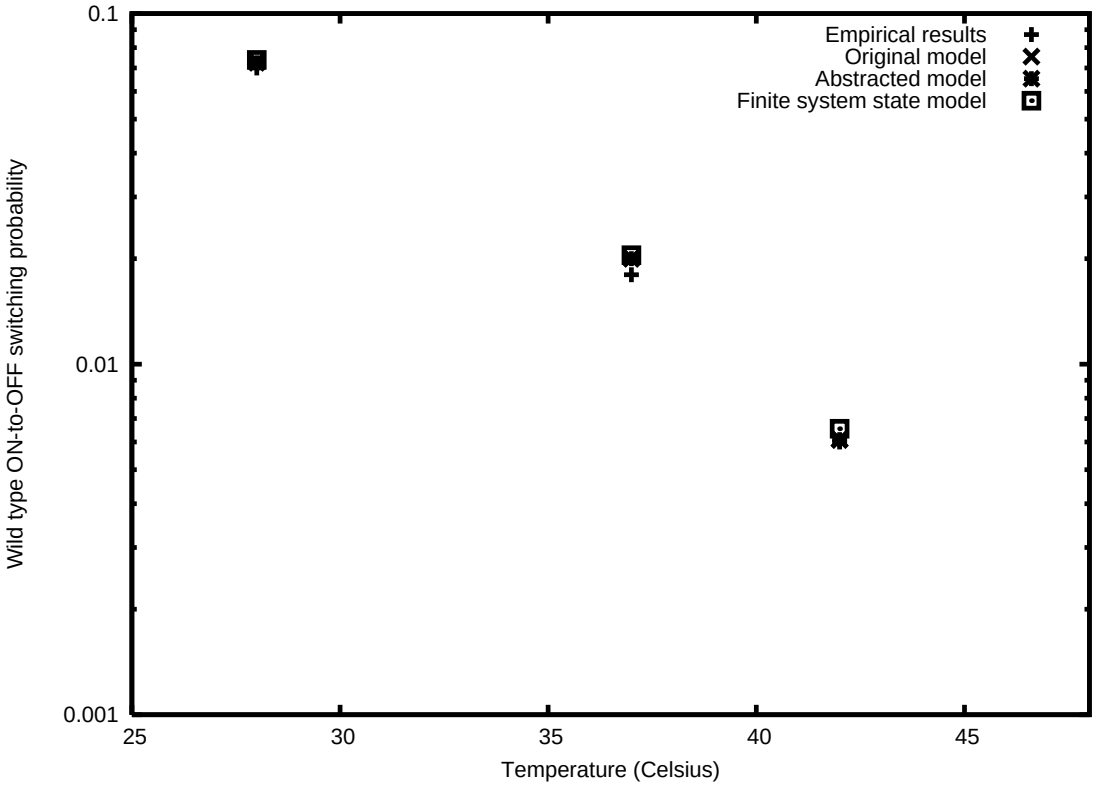


Figure 7.4. ON-to-OFF *fim* switching probability for one cell generation with temperature control. Comparison of empirical results, and computational results from the original model, the abstracted model, and the FSS model.

E. coli cells to multiply itself. There are two strategies that this phage can take to replicate itself as shown in Figure 7.5. One is called *lysis* where the phage creates copies of itself inside the cell and bursts the cell to escape and infect other cells. The other one is a more passive approach called *lysogeny* where the phage integrates its DNA into the host chromosome and replicates its DNA through cell division.

The genetic circuit controlling the phage λ lysis/lysogeny decision is shown in Figure 7.6. The key proteins involved in the phage λ lysis/lysogeny developmental decision are CI, Cro, N, CII, and $CIII$. The lysis/lysogeny decision is a race conditon between CI and Cro. A high concentration of CI leads to the lysogenic pathway, while a high concentration of Cro leads to the lytic pathway. The core component of the genetic circuit is the three operator sites called the λ switch to which CI and Cro dimers can competitively bind to influence the activities of the promoters P_{RM} and P_R [95]. Binding of the CI dimer to the λ switch in wild type setting represses the transcription of the *cro* gene by preventing $RNAP$ from binding to P_R. When the concentration of CI dimer is low, it tends to occupy only one operator site in the λ switch, which turns on the expression of gene cI from P_{RM} only at a very low basal rate. When the concentration of CI dimer is medium, it tends to occupy two operator sites in the λ switch, activating the expression of cI from P_{RM}. However, when the concentration of CI dimer is high, it tends to occupy all three operator sites in the λ switch, which represses the expression of cI from P_{RM} by preventing $RNAP$ from binding to P_{RM}. Binding of Cro dimer to the λ switch in the wild type represses the transcription of cI by preventing $RNAP$ from binding to P_{RM}. When the concentration of Cro dimer is low, it tends to occupy only one operator site in the λ switch; it turns on the expression of *cro* from P_R at a high basal rate. On the other hand, when the concentration of Cro dimer is high, it tends to occupy all three operator sites in the λ switch, which prevents the expression of *cro* from P_R.

Immediately after the infection, there are no CI and Cro molecules in the cell [9]. In this condition, while CI can be synthesized from two promoters, P_{RM}, and P_{RE}, the synthesis of Cro is higher than that of CI since the basal transcription rate

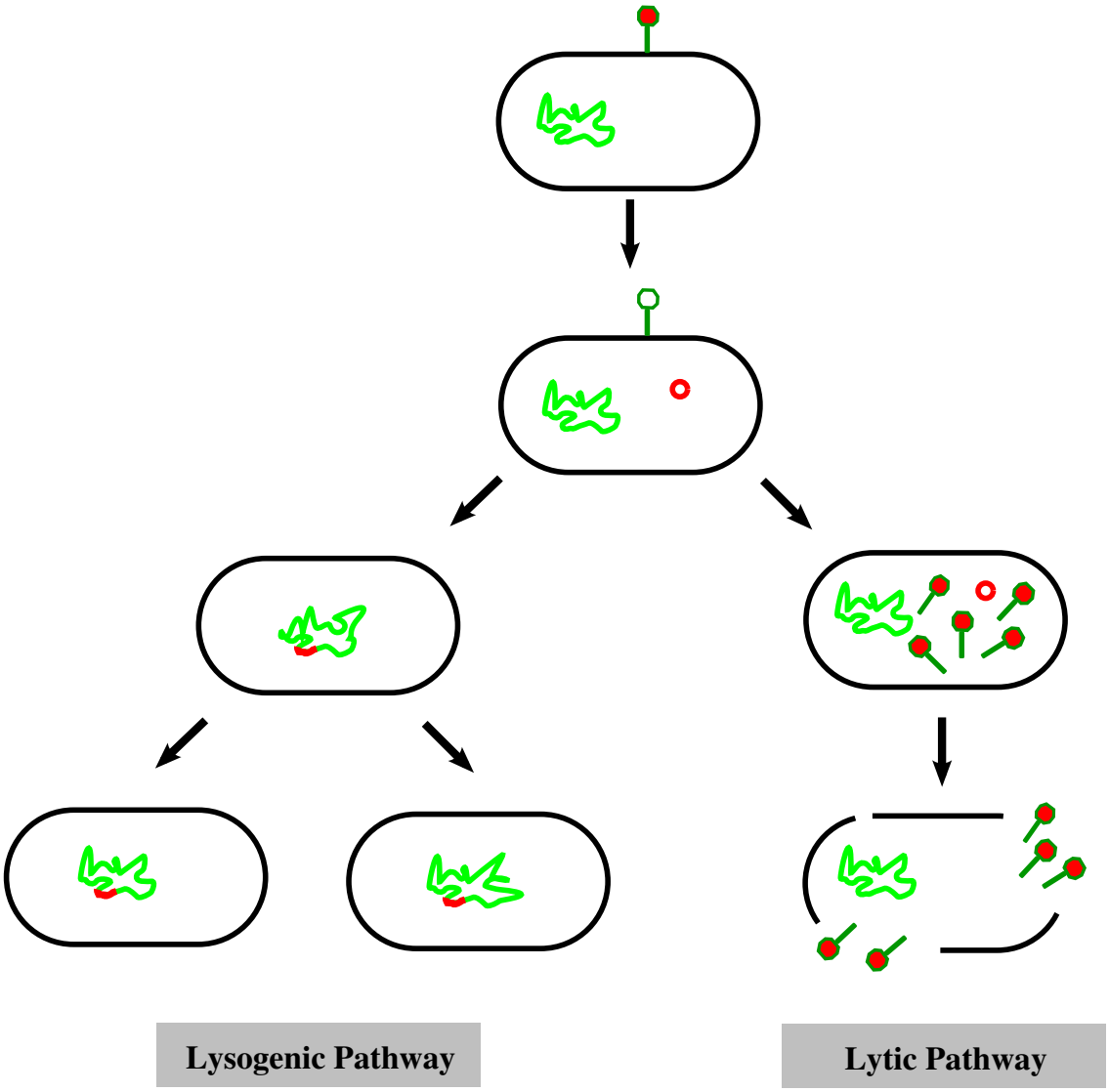


Figure 7.5. Phage λ lysis/lysogeny developmental pathway. Phage λ has two pathways to multiply itself called the *lytic* pathway and the *lysogenic* pathway. In the lytic pathway, the phage first creates proteins needed for formation of new viruses. It then replicates its DNA to create new viruses inside the cell. These viruses burst the cell to escape to infect other cells. In the lysogenic pathway, the phage integrates its DNA into the host chromosome. It then replicates its DNA passively via cell division.

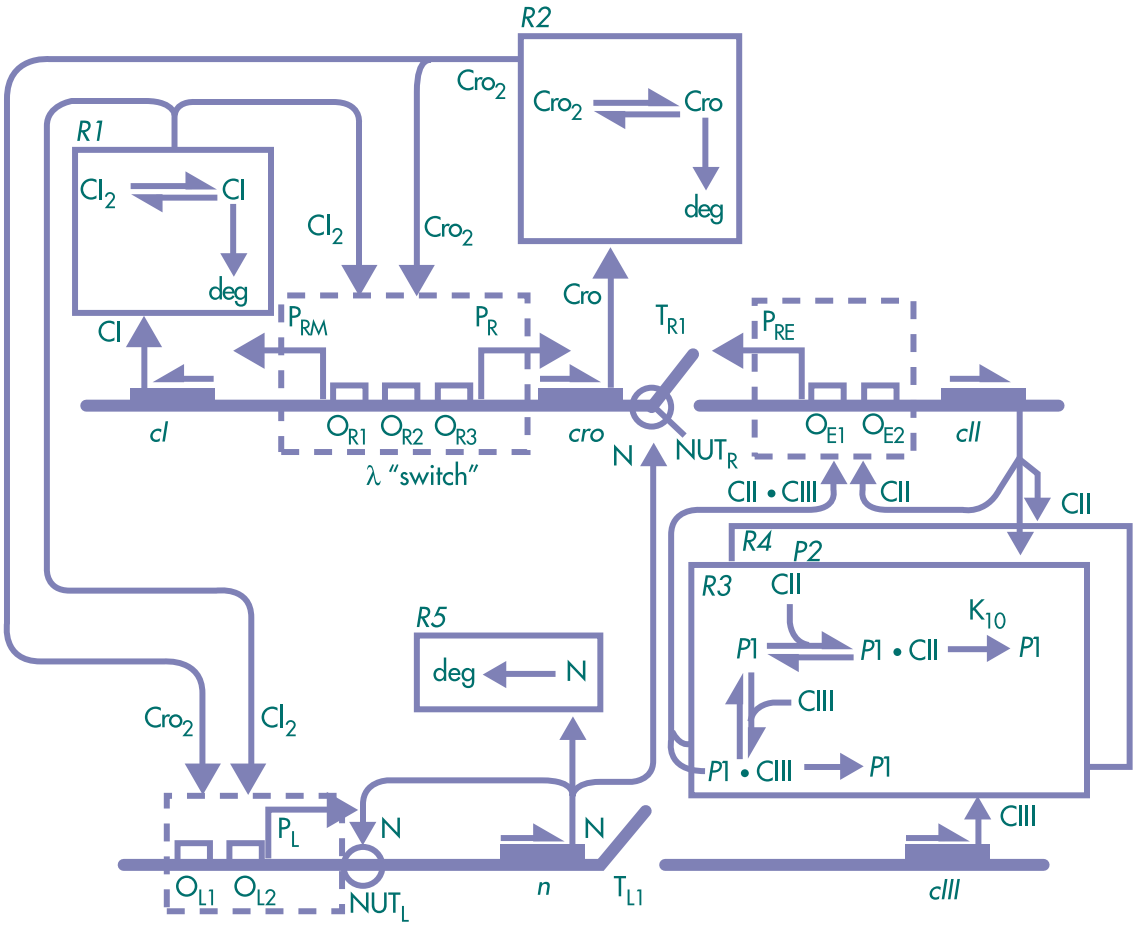


Figure 7.6. Phage λ decision circuit. The key proteins involved in the phage λ lysis/lysogeny developmental decision are *CI*, *Cro*, *N*, *CII*, and *CIII*. The lysis/lysogeny decision is a race conditon between the states of *CI* and *Cro*. A high concentration of *CI* leads to the lysogenic pathway, while a high concentration of *Cro* leads to the lytic pathway. In order for the phage to take the lysogenic pathway, antiterminator, *N*, needs to be synthesized at the early stage of the cell cycle. This antiterminator can help *RNAP* go through the termination sites, T_{L1} and T_{R1}, facilitating transcriptions of genes *cII* and *cIII*. *CIII* can prevent *CII* from degrading by binding to proteases *P1* and *P2* [9], and *CII* activates the transcription of *cI* from the P_{RE} promoter. Further description of the genetic circuit can be found in [9] (courtesy of [3]).

of *cro* is higher. Thus, the favored outcome of the lysis/lysogeny decision is lysis at the early stage of the decision. In order for the phage to take the lysogenic pathway, the enhanced transcription of *cI* from the P_{RE} promoter which is activated by the presence of *CII* is required. For this to happen, the antiterminator, N, needs to be synthesized at the early stage of the decision so that it can help *RNAP* go through the termination sites, T_{L1} and T_{R1} to facilitate synthesis of *CII* and *CIII* at the early stage of the decision. Since *CIII* can prevent *CII* from degrading, a high concentration of *CIII* can lead to a high concentration of *CII*.

We have constructed a REB model for the phage λ decision circuit system which is described in Appendix B. Our initial REB model includes 55 species and 69 reactions, and the set of interesting species, $\mathbf{S_i}$, includes *CI* and *Cro*. This model is then automatically abstracted using REB2SAC. The abstraction engine in REB2SAC is configured for the phage λ decision circuit model so that it collectively applies REB abstraction methods as shown in Figure 7.7.

The seven abstraction methods, irrelevant node elimination (line 3), modifier constant propagation (line 4), rapid equilibrium approximation (line 5), standard quasi-steady-state approximation (line 6), operator site reduction (line 7), similar reaction combination (line 8), and dimerization reduction (line 9), are applied iteratively until there is no change in the model. Irrelevant node elimination and

Algorithm 7.2.1 *Model LambdaAbstractionEngine(Model M)*

```
1:  repeat
2:      M' ← M
3:      M ← IrrelevantNodeElim(M)
4:      M ← ModifierConstantProp(M)
5:      M ← RapidEqApprox(M)
6:      M ← StandardQSSA(M)
7:      M ← OpSiteReduction(M)
8:      M ← SimilarReactionComb(M)
9:      M ← DimerReduction(M)
10: until M' = M
11: return M
```

Figure 7.7. Top level abstraction algorithm of the phage λ decision circuit model.

modifier constant propagation are applied first to reduce the complexity of the model without compromising accuracy. The rapid equilibrium approximation is applied before the standard quasi-steady-state approximation so that, whenever the model contains patterns that match the conditions for both methods, the former has precedence in order to reduce the complexity of the reaction rate laws. The similar reaction combination is applied right after the operator site reduction to immediately combine the structurally similar reactions that are often generated by operator site reduction. The dimerization reduction is placed after operator site reduction since an operator site with a dimer molecule as a transcription factor cannot be reduced otherwise. After collectively applying the REB abstraction methods, the REB model is reduced to only 5 species and 11 irreversible reactions as shown graphically in Figure 7.8. This figure shows the biological gene-regulatory network of the phage λ lysis/lysogeny decision circuit, and it is quite similar to the high-level hand-generated diagram in Figure 7.6. The structure of this graph, however, is automatically generated using abstractions from the low level model.

The goal of our analysis using this computational model is to determine the probability that the lysogenic pathway is chosen under various conditions. For example, it has been shown experimentally that the probability of lysogeny in-

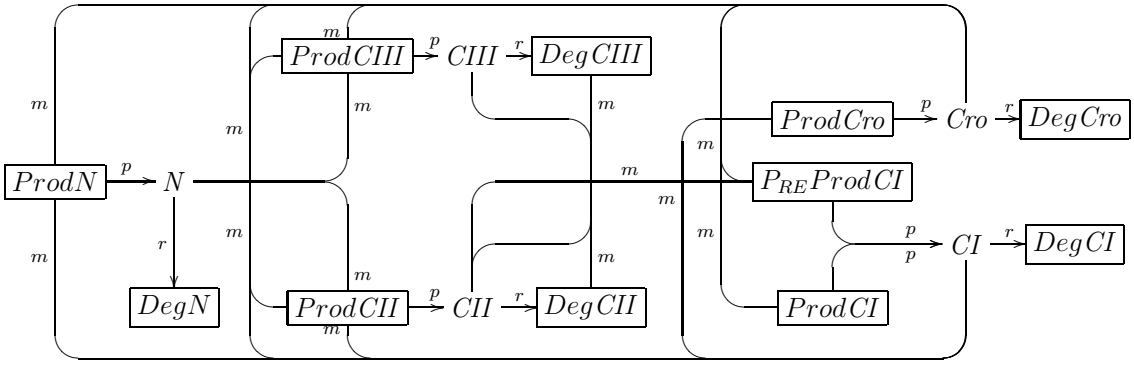
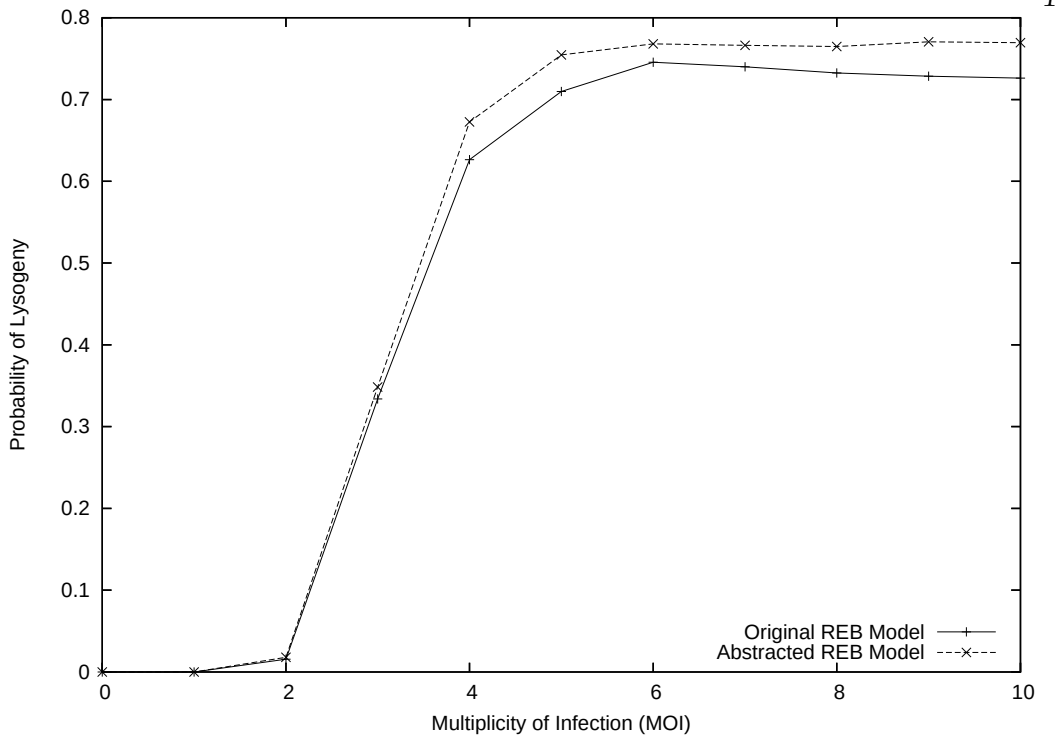


Figure 7.8. Structure of the abstracted model of the phage λ developmental decision gene-regulatory pathway.

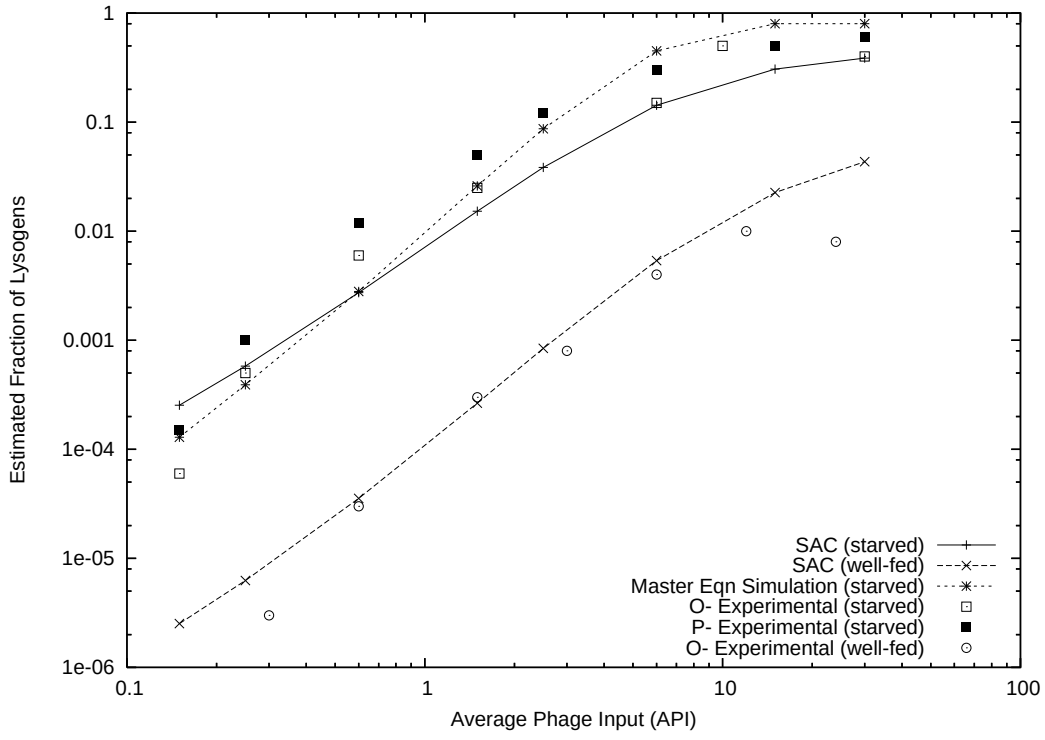
creases as the *multiplicity of infection* (MOI)—the number of phages simultaneously infecting the same cell—increases [70]. Thus, our analysis first predicts the effects of MOI on the probability of lysogeny. For this analysis, both the original model and the abstracted one are simulated for 10,000 runs using the same simulator, an optimized implementation of SSA within REB2SAC, on a 3GHz Pentium4 with 1GB of memory to have a reasonable statistical confidence as well as to measure the speedup gained via abstractions. Each simulation is run for up to one cell cycle while tracking the number of molecules of *CI* and *Cro*. If the number of *CI* molecules exceeds 328 (i.e., 145 *CI* dimers) before the number of *Cro* molecules exceeds 133 (i.e., 55 *Cro* dimers), then the simulation run is said to result in lysogeny [9]. The simulations are run for MOIs ranging from 1 to 50. While the simulation of the original REB model takes 56.5 hours, the abstracted model takes only 9.8 hours, which is a speedup of more than 5.7 times. Figure 7.9(a) shows the probability of lysogeny for MOIs from 0 to 10 for both the original REB model and the abstracted one. The results are nearly the same, yet with a substantial acceleration in runtime.

The n-ary transformation is able to automatically convert our reduced REB model for the phage λ decision circuit into a SAC model. However, since the species *CI* and *Cro* influence many reactions, our automated analysis finds that 10 critical levels are needed for species *CI* and 10 are needed for species *Cro*. This is too many critical levels for the Markov chain analyzer within ATACS. Fortunately, many of these critical levels are very close together and can be combined with little loss in accuracy. Therefore, while we decided to use nine levels for species *CI* and four levels for *CII*, we used only two levels for each of the species *Cro*, *N*, and *CIII*.

We analyzed the SAC model using Markov chain analysis. The probability of lysogeny is calculated by summing the probability of states that reach the highest level of *CI*. We compare our results with both experimental data and previous simulations performed by Arkin et al. on a complete master equation model. The experimental results are from Kourilsky [70]. Since it was not practical to measure the number of phages that infect any given cell, Kourilsky measured the fraction of cells that commit to lysogeny versus *average phage input* (API) (i.e., the proportion



(a)



(b)

Figure 7.9. Results from the phage λ decision circuit model. (a) Comparison of simulation results for the probability of lysogeny over MOI from the original model and its abstracted model where each data point has a margin of error of less than 0.01 with a 95 percent confidence. (b) Comparison of SAC results to experimental data.

of phages to *E. coli* within the population). Kourilsky performed experiments for both "starved" *E. coli* and those in a "well-fed" environment. He found that the fraction that commits to lysogeny increases with increasing API, and that this fraction increases by more than an order of magnitude in a starved environment over a well-fed environment.

To map simulated MOI data onto API data, Arkin et al. used a Poisson distribution of the phage infections over the populations:

$$P(M, A) = \frac{A^M}{M!} e^{-A} \tag{7.1}$$

$$F_{\text{lysogens}}(A) = \sum_M P(M, A) \cdot F(M) \tag{7.2}$$

where M is the MOI, A is the API, and $F(M)$ is the probability of lysogeny determined by Markov analysis. We also used this method to map our MOI data. The results are shown in Figure 7.9(b). The individual points represent experimental measurements while the lines represent simulation results. Both the Arkin et al. simulation and our SAC model results track the starved data points reasonably well. Our SAC model results, however, are found in less than 7 minutes of computation time on a 3GHz Pentium4 with 1GB of memory. While modern computer technology and algorithmic improvements would greatly improve the simulation time of the Arkin et al. model, these results would still take several hours to generate on a similar computer to ours. Another notable benefit of our SAC method is that it can also produce simulation results for the well-fed case in about 7 minutes. These results could likely not be generated even today using the Arkin et al. master equation simulation method, since the number of simulation runs necessary is inversely proportional to the probability of lysogeny (i.e., about two orders of magnitude greater in the well-fed case than in the starved one).

CHAPTER 8

CONCLUSIONS

Systems biology—albeit still its infancy—has the potential to revolutionize how biological research is conducted. As more and more critical biological data are becoming available at a rapid pace and as the biological questions being addressed are becoming more complex and challenging, integration of computational methods with the process of biological research becomes more imminent. Consequently, development of efficient and effective computational analysis is crucial to the success of systems biology research to gain further understanding of systems-level biological properties and to apply such knowledge to better control the functions of biological systems. This dissertation has introduced one such methodology and illustrates its usefulness by applying it to a number of systems. This chapter concludes the dissertation by first summarizing the dissertation in Section 8.1, and then presenting possible future work in Section 8.2.

8.1 Summary

This dissertation presents a general methodology for systematically and automatically abstracting the complexities of large-scale biochemical reaction-based networks. The REB model abstractions significantly facilitate efficient temporal behavior analysis of such systems by substantially reducing the problem dimensionality in both species and reactions, thus potentially allowing for both simulation time acceleration and computability gains while facilitating a high-level view of the network. To improve the numerical analysis time, a REB model can be further abstracted to a FSS model to allow for state space exploration based temporal behavior analysis approach on the underlying Markov chain. The system state space can be more aggressively reduced by transforming a REB model to a SAC model,

enabling further improvement in the computational analysis time. Furthermore, since our approach allows for multiple levels of abstraction, it is broadly applicable to a wide range of biological systems and their representations—from CCK models to SCK models—including the genetic regulatory networks upon which we have chosen to focus in this dissertation.

The abstraction methods presented in this dissertation coupled with a number of temporal behavior analysis methods are implemented in our modeling and analysis tool REB2SAC [73, 74]. By performing these transformations systematically and automatically and allowing an easily-configurable reduction control using REB2SAC, accuracy and efficiency of modeling biochemical systems at various levels of resolution can be significantly improved. Furthermore, to achieve better user experience of the tool, REB2SAC is integrated into a graphical-user-interface-based analysis tool called BioSim [2].

As a case study, we have illustrated applications of individual REB abstraction methods using small systems. This reveals that each abstraction method—even applied alone—has the potential to substantially accelerate the simulation time while maintaining a reasonable accuracy. Furthermore, we have collectively applied several abstraction methods within REB2SAC to systems-level analysis of a couple of genetic regulatory networks.

These preliminary results are promising. In the analysis of temperature control of expression of type 1 pili in *E. coli*, we have quantitatively estimated and analyzed the temperature effects on the wild-type and *FimB*-promoted ON-to-OFF switching probabilities in minimal medium, which are shown to be consistent with those expected empirically from [45]. Furthermore, using REB2SAC, we have demonstrated how the models with various levels of abstraction can be generated and utilized to obtain results that agree with those from the original *fim* switch inversion model at two orders of magnitude improvement in computational performance. In the analysis of the phage λ developmental pathway, we have demonstrated that the probabilities of lysogeny for various MOI points obtained from the original phage λ circuit model can be approximated well by the results from the abstracted model

with substantial computational gain. Furthermore, using the SAC model of the phage λ decision circuit, we are able to estimate the experimental results of the fraction of lysogens over API under various conditions [70]. The SAC model results are generated in a matter of minutes while the simulation results of REB models with a reasonable statistical confidence would have taken many hours to generate. Therefore, from case studies, among other things, we are able to: (1) ascertain the internal self-consistency of our approach by successfully cross-validating each abstraction level output against the results of the full underlying SCK model simulations; and (2) accurately estimate the biologically relevant properties, which typically require substantial numbers of hours of computation time via the original REB representation, yet could be computed in only minutes using our abstraction approach.

8.2 Future Work

Although our automatic and systematic modeling and analysis methodology for biochemical systems has demonstrated its effectiveness and usefulness, it can be improved in many ways. This section summarizes several such research investigations that we believe deserve some attention.

8.2.1 New Modeling Language

While standardized biochemical modeling languages such as SBML provide a framework to, for example, ease exchange of models and development of modeling and analysis tools, they come with some disadvantages. One such limitation of SBML when it comes to automatic and systematic model abstraction is that it cannot—without using proprietary annotations—specify types of reactions due to its low level representation of species' interactions to attempt to accommodate many biochemical systems. Thus, using SBML as an input language to REB2SAC, a heuristic approach such as identification of an enzyme species must be added to the algorithms of various enzymatic reaction model abstractions. If, however, a modeling language is capable of expressing which interaction is an enzymatic

reaction and which species is used for an enzyme of a given enzymatic reaction, then implementation of automated model abstraction can become much easier. Such a high-level specification is particularly useful for representing transcriptional genetic regulatory networks whereby *cis*-regulatory element configurations can be clearly and compactly specified without introducing complex species. Thus, it provides the user with a better interpretability of models. Therefore, developing such a higher-level representation language that addresses the issues concerning automatic generation of various abstracted models, coupled with a compiler that transforms such a language to SBML, can be very useful.

8.2.2 More Abstraction Methods

REB2SAC can be improved by developing more abstraction methods. These methods can be system-specific or generalized. While the advantage of generalized abstraction methods is apparent, system-specific abstraction methods can be tremendously useful since each system has different types of assumptions as well as species' interactions. For example, a metabolic pathway has many sequential reactions to transfer energies, which may be very different from reactions often seen in genetic regulatory networks. Thus, tailored model abstraction methods for each system may be able to substantially reduce the complexity of a given system compared with generalized abstraction methods.

8.2.3 Intelligent Abstraction Selection

In the current REB2SAC, although there is a default setting, the user chooses which abstractions to apply and in what order. While this provides a great flexibility in the selection of abstraction methods, this requires the user to be familiar with the low-level details such as what each abstraction method does and the unique identifier of each abstraction method. This may not be practical as the number of abstraction methods increases. Thus, if the tool itself is able to decide which abstractions to apply based on accuracy and efficiency criteria that the user specifies, then the usability of the tool increases tremendously. This requires an

intelligent abstraction selection which statically determines that set of abstraction methods produces a model whose results closely satisfy the user's demands. This means, for example, that, given an enzymatic reaction model, the tool systematically determines from the model structure, the initial condition, and the parameter values whether or not its PPTA model produces more accurate results than its QSSA model.

8.2.4 Spatiotemporal Model Abstraction

Throughout this dissertation, regardless of a CCK model or a SCK model, our abstraction methodology makes the well-stirred assumption. While this assumption significantly reduces the complexity of a model, it may not be able to capture characteristics of an *in vivo* biochemical system where localizations play a crucial role in its system behavior. In such cases, in order to more accurately predict the system behavior, spatial properties must be included in a computational model, resulting in a substantial increase in complexity. Thus, systematic and automatic abstraction of such spatiotemporal models is believed to be very promising.

8.2.5 More Case Studies

In this dissertation, we have primarily focused on genetic regulatory networks because of, among other things, data availability. The two networks used in Chapter 7 are among the most well-studied systems, which is appropriate for the proof of concept of our abstraction methodology. Thus, in order to further analyze the usefulness of our methodology, it can be applied to other biochemical systems such as other genetic regulatory networks, signal transduction networks, and metabolic networks. For example, we are in the process of applying our methodology to three more biological systems to facilitate efficient computational analysis. The first system is for a design of a genetic Muller C-element using transcriptional regulatory elements [87]. The second system is to analyze how to optimize a nonviral polymeric nucleic acid delivery to the nucleus for drug effects [118]. The third system is the MAP kinase cascade subsystem, in which the scaffold

called Ste5 binds to various proteins whereby combinatorial explosion via different complex formations provides significant computational challenges [4]. Such case studies provide not only more *in silico* predictions on biologically relevant properties but also better insights into which directions this research should take.

APPENDIX A

FIM SWITCH INVERSION MODEL

The detailed *fim* inversion model uses a low-level reaction-based representation, which describes reaction-scale molecular process abstraction that generally satisfies the Markovian requirement of the SSA. The switch inversion system described here consists of two major subnetworks: the production-degradation model for *FimB* and *FimE*; and the model describing the configuration of the *fim* switch itself (Figure 7.1). Both are subject to external control by the global regulator proteins: *H-NS*, *IHF*, and *Lrp*. As a further simplification, *E. coli* is assumed to be a cylinder 2μm long and 1μm in cross-sectional diameter in minimal medium conditions. Thus, the concentration of one molecule of species is set as 1nM throughout, and the effects of leucine on *Lrp* binding is ignored. This appendix describes how the detailed model is constructed via each of the constitutive subnetworks. It also describes how the values of the parameters and initial concentrations are determined for each temperature setting.

A.1 FimB and FimE Regulation Model

A small protein *H-NS* represses the expression of both *fimB* and *fimE* by occupying the DNA regions containing the *fimB* and *fimE* promoters and by preventing *RNAP* from binding [89]. The reaction subnetwork of this process is given in Figure A.1.

Importantly, *H-NS* activity is controlled, in part, by the ambient temperature, and consequently, so is the production of *FimB* and *FimE*. It has been further reported that the *hns* gene is auto-regulated and that the concentration of *H-NS* generally remains constant, except during cold shock [11]. Thus, it must be changes in promoter site binding/unbinding rates, rather than the variations in the concen-

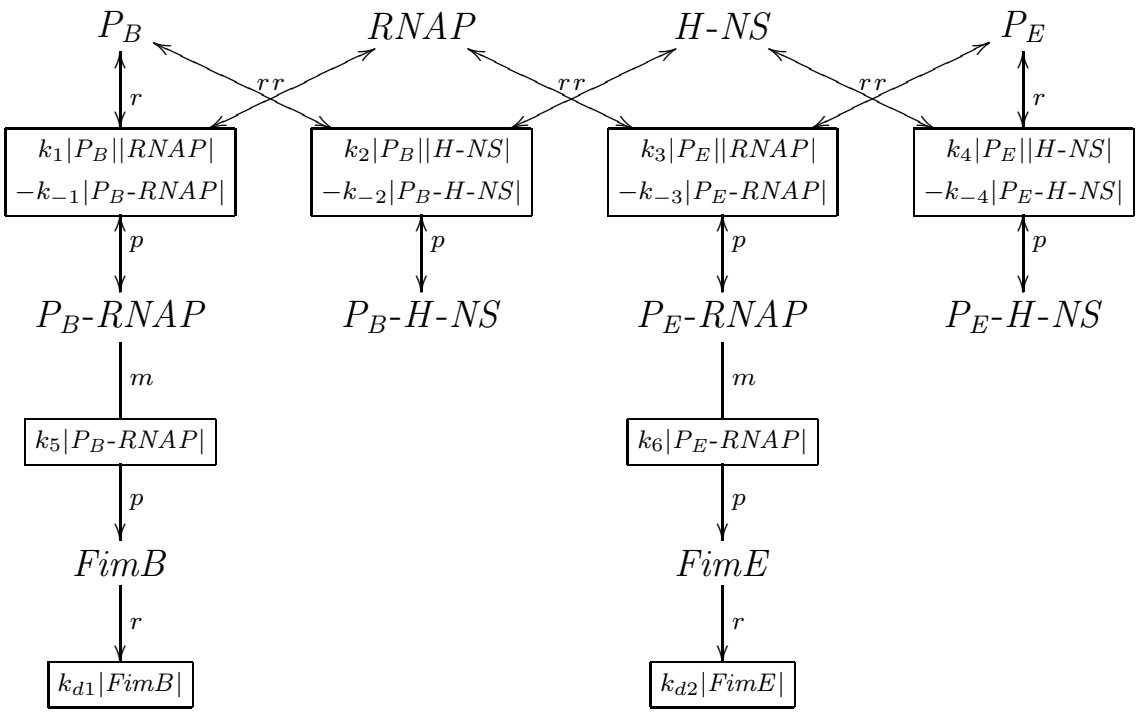


Figure A.1. Detailed model subnetwork of *FimB* and *FimE* regulation. Repression of *fimB* and *fimE* transcription by *H-NS* is represented by binding of *H-NS* to their promoter sites, which prevents binding of *RNAP*.

tration of *H-NS* that are responsible for the thermo-regulation of *fimB* and *fimE* expression. Furthermore, although the transcription of both *fimB* and *fimE* is repressed by *H-NS*, this effect varies with temperature. From 30 °C to 37 °C, the expression of *fimB* increases about two-fold (119 vs. 195 Miller units), while the expression of *fimE* decreases about four-fold (226 vs. 61 Miller units) [89]. This reveals that the up-regulation of *FimB* by *H-NS* is induced by higher temperatures, as is the case for a number of proteins due to a decrease in the concentration of the oligomeric *H-NS*, which has a higher affinity for binding to DNA [90].

To estimate the *H-NS* rate constants for binding and unbinding to P_B (i.e., k_2 and k_{-2}) at 28 °C, 37 °C, and 42 °C, our model uses K_D values for the interaction of *H-NS* with DNA at 25 °C, 37 °C, and 42 °C from *S. typhmurium* [90]. The *H-NS* rate constants involved in P_E (i.e., k_4 and k_{-4}) are inferred analogously, though unlike the *fimB* case the negative modulation by *H-NS* is reduced at higher temperatures. At 28 °C, *fimE* is estimated to produce 200 proteins in one cell generation, while at 37 °C, it produces 61 proteins. These numbers are chosen to be comparable with the ratio of the *fimE* expression data at 30 °C and 37 °C from [89]. To reduce the *FimE* production even further at 42 °C, *fimE* is assumed to produce 25 proteins. The K_D values of *H-NS* binding to P_B and P_E used in our model are shown in Table A.1. The binding rate constants are derived from these K_D values by assuming a rapid unbinding rate and by setting the unbinding rate constant to 10s^{-1}.

The initial concentration of *RNAP* is chosen to be 30nM, which is the same as the one in the phage λ developmental decision pathway model in *E. coli* [9]. The initial concentration of *H-NS* as well as the *RNAP* binding and unbinding

Table A.1. K_D for *H-NS* binding to *cis* elements.

Temperature (°C)	**K_D at P_B (μM)**	**K_D at P_E (μM)**
28	260	442
37	442	80
42	735	30

rate constants for both promoter sites (i.e., k_1, k_{-1}, k_3, and k_{-3}) are derived by assuming 50 percent occupancy of *H-NS* and 25 percent occupancy of *RNAP* at P_B at 37 °C. This configuration is found to be effective to model the thermo-regulation of *fimB* and *fimE* expression by *H-NS*. While the obtained *H-NS* amount is significantly higher than the number reported in [11] (i.e., 14,000), this may be explained by, for example, our insufficient knowledge of the underlying *H-NS* dimerization and tetramerization and using the total *H-NS* concentration to model the thermo-regulation.

The value of k_5 is derived by assuming that *FimB* is produced around 200 times in one cell generation at 37 °C. Using this value, *fimB* produces approximately 400 proteins in one cell generation in *hns*-negative type at 37 °C. Thus, the ratio of the *FimB* productions in the *hns*-negative type and the wild type at 37 °C is closely comparable with the one in the gene expression data from [89]. The production rate constant of *FimE* (i.e., k_6) is chosen to be the same as that of *FimE*.

Our model also includes degradation reactions for *FimB* and *FimE* as shown in Figure A.1. The value of the degradation rate constant of *FimB* (i.e., k_{d1}) is chosen so that its production reaction and the degradation reaction equilibrate when the concentration of *FimB* is 100nM at 37 °C. This number is chosen as the best fit from the range of 1–100nM thought to be a reasonable value for $|FimB|$ and $|FimE|$ [116]. The degradation rate constant of *FimE* (i.e., k_{d2}) is then given the same value as that of *FimB*.

The initial concentrations of the two recombinases are determined by first running an ODE simulation of the *FimB* and *FimE* regulation model for a two cell generation time span, given each recombinase concentration is initially set to 0nM. The concentrations of the two recombinases are then retrieved at the end of the simulation run for each temperature setting. The concentrations of *FimB* and *FimE* obtained from this scheme, shown in Table A.2, are designated as initial concentrations.

Table A.2. Initial concentrations of *FimB* and *FimE* for each temperature.

Temperature (°C)	$\lvert FimB\rvert_{\mathbf{0}}$ **(nM)**	$\lvert FimE\rvert_{\mathbf{0}}$ **(nM)**
28	74	100
37	100	31
42	124	13

A.2 The *fim* Switch Configuration Model

The second major subnetwork includes binding/unbinding reactions for the *fim* DNA element, which can lead to an ON-to-OFF switch inversion. These reactions are reverse-engineered from the thermodynamic states of various *fim* DNA element configurations, as discussed below, based on Gibbs free energy, ΔG, values given in [116]. Table A.3 lists the 18 states of the *fim* switch configuration, given that the switch is in the ON position. In this table, *IRX* represents both IRL and IRR sites shown in Figure 7.1, where the two recombinases can bind so as to invert the *fim* switch, *IHF-X* corresponds to the two *IHF* binding sites, IHF I and IHF II, and *Lrp-X* represents the three *Lrp* sites: Lrp-I, Lrp-II, and Lrp-III. The symbols, i, j, k, and m, represent the numbers of molecules of *IHF*, *FimB*, *FimE*, and *Lrp* bound to the switch DNA region, while k_p represents the switching reaction rate constant. Since only states 3-8, where *IHF* is bound to *IHF-X* and either recombinase species is bound to *IRX*, are configured to invert the *fim* switch from ON to OFF, the values of k_p are set to 0 for states 1-2, and 9-18, while the values of k_p for states 3-8 are derived using our qualitative knowledge on the switching regulation, and chosen so that results from our detailed model fit the empirical results. For example, since the switching rates are faster when *Lrp* occupies *Lrp-I* and/or *Lrp-II*, but not *Lrp-III*, the values of $k_p(5)$ are chosen to be much greater than those of $k_p(3)$ and $k_p(6)$.

The detailed *fim* switch configuration model is constructed by first reverse-engineering the underlying reactions from the equilibrium statistical thermodynamics model, where the probability of each switch DNA configuration is defined by the parameters shown in Table A.3. This is accomplished by using the hypothesis

Table A.3. State table for the ON state DNA binding of the *fim* switch based on [116].

State	*IHF-X*	*IRX*	*Lrp-X*	ΔG (kcal)	k_p (s^{-1})	i	j	k	m
1	-	-	-	0	0	0	0	0	0
2	*IHF*	-	-	−13	0	1	0	0	0
3	*IHF*	*FimE*	-	−23	6.53e-8	1	0	1	0
4	*IHF*	*FimB*	-	−23	6.5e-7	1	1	0	0
5	*IHF*	*FimE*	*Lrp*	−47	3.0e-4	1	0	1	2
6	*IHF*	*FimE*	*Lrp*	−59.3	8.0e-5	1	0	1	3
7	*IHF*	*FimB*	*Lrp*	−47	3.7e-6	1	1	0	2
8	*IHF*	*FimB*	*Lrp*	−59.3	7.5e-7	1	1	0	3
9	-	*FimE*	-	−10	0	0	0	1	0
10	-	*FimB*	-	−10	0	0	1	0	0
11	-	*FimE*	*Lrp*	−34	0	0	0	1	2
12	-	*FimE*	*Lrp*	−46.3	0	0	0	1	3
13	-	*FimB*	*Lrp*	−34	0	0	1	0	2
14	-	*FimB*	*Lrp*	−46.3	0	0	1	0	3
15	-	-	*Lrp*	−24	0	0	0	0	2
16	-	-	*Lrp*	−36.3	0	0	0	0	3
17	*IHF*	-	*Lrp*	−37	0	1	0	0	2
18	*IHF*	-	*Lrp*	−49.3	0	1	0	0	3

that the binding and unbinding reactions are much more rapid as compared to the associated switching or gene expression rates [7]. Then, the corresponding binding and unbinding reactions are estimated from the standard free energy relationship, $\Delta G = -RT \ln (k_f/k_r)$ using a rapid unbinding rate constant of 1.0s^{-1}. For example, the reaction-based model for switch state 6 is reverse-engineered as shown in Figure A.2. In this reaction scheme, the new species P_{fim} represents the open binding sites, while *S6* corresponds to the bound configuration where one molecule of *IHF* binds to *IHF-X*, one molecule of *FimE* binds to *IRX*, and three molecules of *Lrp* bind to *Lrp-X*. In other words, *S6* could be thought of as the switching dynamics analog of the closed-complex configuration in transcription modeling. The association and dissociation rate constants for binding are deduced from $k_{s6}/k_{-s6} = \exp(-\Delta G_{31b}/RT)$.

The temperature effect in the modeling of the *fim* switch configuration is expressed by directly changing the initial concentration of *Lrp*. Furthermore, in our

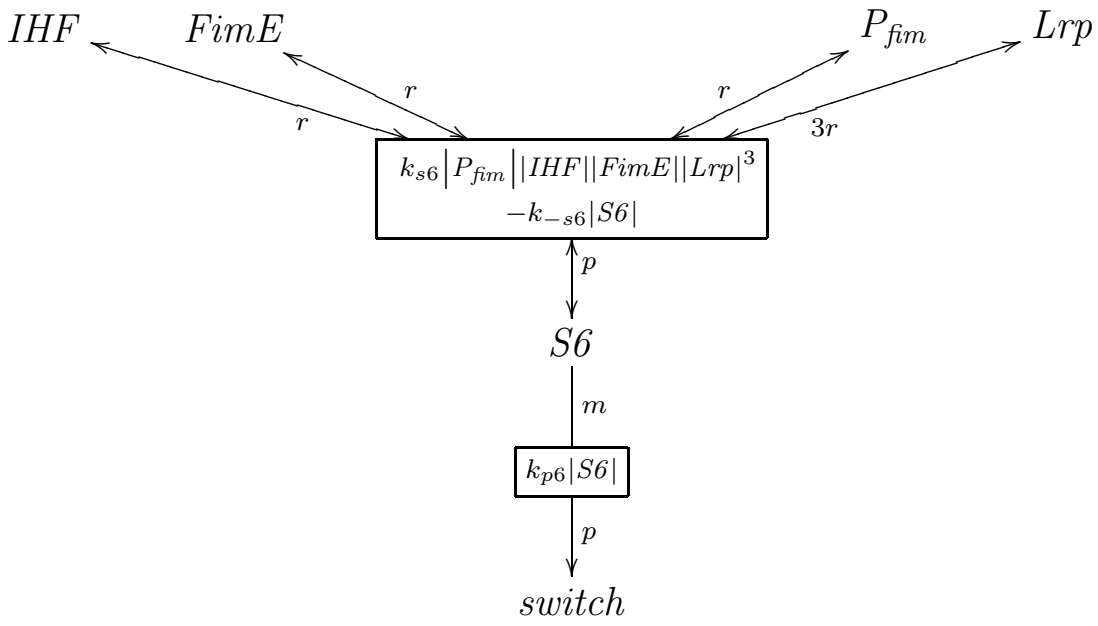


Figure A.2. Detailed model reaction of the *fim* switch inversion through state 6.

model, the concentration of *Lrp* is quantified for each temperature setting based on the temperature tuning mechanism of *Lrp* as illustrated in Figure A.3. The concentration of *Lrp* at 37 °C is chosen to be 5nM as this value is determined to be the physiologic concentration of free *Lrp* in the cell at 37 °C in [116]. The concentration of *Lrp* at the other two temperature settings is set so that it qualitatively agrees with the observation on the temperature tuning mechanism in [116]. At 28 °C, $|Lrp|_0$ is set to 2nM so that *Lrp* molecules are unlikely to occupy Lrp-I and Lrp-II, and moreover to prevent *Lrp* molecules from binding to Lrp-III, while, at 42 °C, $|Lrp|_0$ is set to 20nM so that *Lrp* molecules are likely to occupy all three Lrp binding sites.

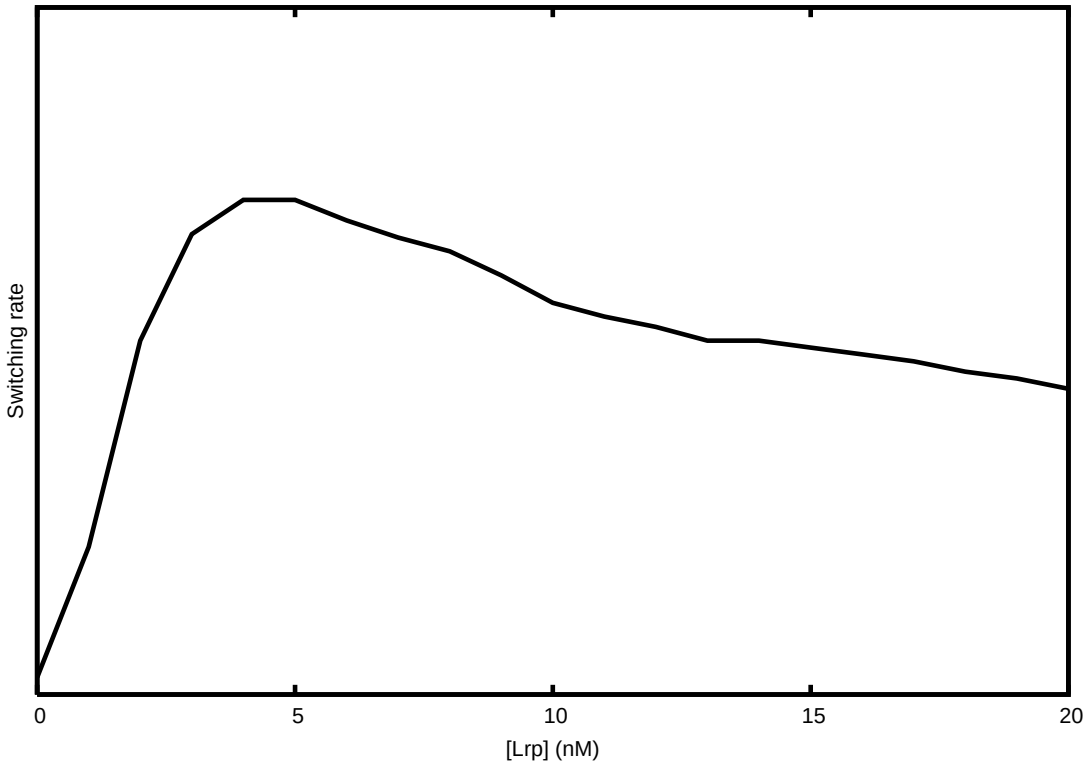


Figure A.3. Temperature tuning mechanism of *Lrp*. At very low concentration, *Lrp* is unlikely to occupy the *Lrp* binding sites, and the *fim* switching rate is low. When the concentration is around 5nM, *Lrp* tends to occupy *Lrp-1* and/or *Lrp-2* but not *Lrp-3*, and this configuration activates the switching. As the concentration of *Lrp* increases even further, *Lrp* is likely to occupy *Lrp-3* as well as *Lrp-1* and *Lrp-2*, and this inhibits the switching.

APPENDIX B

PHAGE λ DECISION CIRCUIT MODEL

Our phage λ lysis/lysogeny decision circuit model is based on [108, 9] and includes the five genes: *cI*; *cro*; *cII*; *cIII*; and *N*, and four promoters: P_{RM}; P_R; P_{RE}; and P_L which are depicted in Figure 7.6. This appendix presents the chemical reaction level representation of our phage λ decision circuit model. The reactions shown in Figure B.1 model the behavior of the P_{RE} promoter where CII protein activates the production of CI protein. The dimerazation and degradation reactions for CI and Cro are shown in Figure B.2. The production and degradation model of the protein N is shown in Figure B.3. The production of the CIII protein from promoter P_L is modeled with the reactions shown in Figure B.4. The CII production model is shown in Figure B.5. Our degradation mechanism of CII and CIII is shown in Figure B.6. Figure B.7 presents reactions for the configuration of the λ switch (i.e., O_R), and Figure B.8 shows the rate constants of the λ switch model.

$$
\begin{aligned}
P_{RE} + \text{RNAP} &\overset{K_{ORE2}}{\longleftrightarrow} P_{RE} \cdot \text{RNAP} \\
P_{RE} + \text{CII} &\overset{K_{ORE3}}{\longleftrightarrow} P_{RE} \cdot \text{CII} \\
P_{RE} + \text{CII} + \text{RNAP} &\overset{K_{ORE4}}{\longleftrightarrow} P_{RE} \cdot \text{CII} \cdot \text{RNAP} \\
P_{RE} \cdot \text{RNAP} &\overset{k_{PREb}}{\longrightarrow} P_{RE} \cdot \text{RNAP} + 10\text{CI} \\
P_{RE} \cdot \text{CII} \cdot \text{RNAP} &\overset{k_{PRE}}{\longrightarrow} P_{RE} \cdot \text{CII} \cdot \text{RNAP} + 10\text{CI}
\end{aligned}
$$

Constant	*Value*	*Constant*	*Value*
K_{ORE2}	0.01	k_{PREb}	0.00004 s^{-1}
K_{ORE3}	0.00726	k_{PRE}	0.015 s^{-1}
K_{ORE4}	0.00161		

Figure B.1. Chemical reaction network model of the promoter P_{RE}.

$$
\begin{aligned}
2\text{CI} &\overset{K_2}{\longleftrightarrow} \text{CI}_2 \\
2\text{Cro} &\overset{K_5}{\longleftrightarrow} \text{Cro}_2 \\
\text{CI} &\overset{k_1}{\longrightarrow} () \\
\text{Cro} &\overset{k_4}{\longrightarrow} ()
\end{aligned}
$$

Constant	*Value*
k_1	0.0007 s^{-1}
K_2	0.1
k_4	0.0025 s^{-1}
K_5	0.1

Figure B.2. Model for CI and Cro dimerazation and degradation.

$$
\begin{aligned}
O_L + \mathrm{Cro}_2 &\overset{K_{OL2}}{\longleftrightarrow} O_L \cdot \mathrm{Cro}_2 \\
O_L + \mathrm{CI}_2 &\overset{K_{OL4}}{\longleftrightarrow} O_L \cdot \mathrm{CI}_2 \\
O_L + 2\mathrm{Cro}_2 &\overset{K_{OL7}}{\longleftrightarrow} O_L \cdot 2\mathrm{Cro}_2 \\
O_L + \mathrm{CI}_2 + \mathrm{Cro}_2 &\overset{K_{OL8}}{\longleftrightarrow} O_L \cdot \mathrm{CI}_2 \cdot \mathrm{Cro}_2 \\
O_L + 2\mathrm{CI}_2 &\overset{K_{OL10}}{\longleftrightarrow} O_L \cdot 2\mathrm{CI}_2 \\
O_L + \mathrm{RNAP} &\overset{K_{OL6}}{\longleftrightarrow} O_L \cdot \mathrm{RNAP} \\
O_L \cdot \mathrm{RNAP} &\overset{k_{PL}}{\longrightarrow} O_L \cdot \mathrm{RNAP} + 10\mathrm{N} \\
\mathrm{N} &\overset{k_7}{\longrightarrow} ()
\end{aligned}
$$

Constant	*Value*	*Constant*	*Value*
K_{OL2}	0.4132	K_{OL8}	0.014
K_{OL4}	0.2025	K_{OL10}	0.058
K_{OL6}	0.6942	k_{PL}	0.011 s^{-1}
K_{OL7}	0.0158	k_7	0.00231 s^{-1}

Figure B.3. Model for N production from promoter P_L and N degradation.

$$
\begin{aligned}
\mathrm{NUT}_L + \mathrm{N} &\overset{K_{\mathrm{NUT}}}{\longleftrightarrow} \mathrm{NUT}_L \cdot \mathrm{N} \\
O_L \cdot \mathrm{RNAP} + \mathrm{NUT}_L &\overset{0.2*k_{PL}}{\longrightarrow} O_L \cdot \mathrm{RNAP} + \mathrm{NUT}_L + 10\mathrm{CIII} \\
O_L \cdot \mathrm{RNAP} + \mathrm{NUT}_L \cdot \mathrm{N} &\overset{k_{PL}}{\longrightarrow} O_L \cdot \mathrm{RNAP} + \mathrm{NUT}_L \cdot \mathrm{N} + 10\mathrm{CIII}
\end{aligned}
$$

Figure B.4. Model for CIII production from promoter P_L. Note that K_{NUT} is 0.2.

$$
\begin{array}{rcl}
\mathrm{NUT}_R1 + \mathrm{N} & \overset{K_{\mathrm{NUT}}}{\longleftrightarrow} & \mathrm{NUT}_R1 \cdot \mathrm{N} \\
\mathrm{NUT}_R2 + \mathrm{N} & \overset{K_{\mathrm{NUT}}}{\longleftrightarrow} & \mathrm{NUT}_R2 \cdot \mathrm{N} \\
\mathrm{NUT}_R3 + \mathrm{N} & \overset{K_{\mathrm{NUT}}}{\longleftrightarrow} & \mathrm{NUT}_R3 \cdot \mathrm{N} \\
\mathrm{NUT}_R4 + \mathrm{N} & \overset{K_{\mathrm{NUT}}}{\longleftrightarrow} & \mathrm{NUT}_R4 \cdot \mathrm{N} \\
\hline
O_R12 \cdot \mathrm{RNAP} + \mathrm{NUT}_R1 & \overset{0.5*k_{PR}}{\longrightarrow} & O_R12 \cdot \mathrm{RNAP} + \mathrm{NUT}_R1 + 10\mathrm{CII} \\
O_R12 \cdot \mathrm{RNAP} + \mathrm{NUT}_R1 \cdot \mathrm{N} & \overset{k_{PR}}{\longrightarrow} & O_R12 \cdot \mathrm{RNAP} + \mathrm{NUT}_R1 \cdot \mathrm{N} + 10\mathrm{CII} \\
O_R \cdot 2\mathrm{RNAP} + \mathrm{NUT}_R2 & \overset{0.5*k_{PR}}{\longrightarrow} & O_R \cdot 2\mathrm{RNAP} + \mathrm{NUT}_R2 + 10\mathrm{CII} \\
O_R \cdot 2\mathrm{RNAP} + \mathrm{NUT}_R2 \cdot \mathrm{N} & \overset{k_{PR}}{\longrightarrow} & O_R \cdot 2\mathrm{RNAP} + \mathrm{NUT}_R2 \cdot \mathrm{N} + 10\mathrm{CII} \\
O_R \cdot \mathrm{CI}_2 \cdot \mathrm{RNAP} + \mathrm{NUT}_R3 & \overset{0.5*k_{PR}}{\longrightarrow} & O_R \cdot \mathrm{CI}_2 \cdot \mathrm{RNAP} + \mathrm{NUT}_R3 + 10\mathrm{CII} \\
O_R \cdot \mathrm{CI}_2 \cdot \mathrm{RNAP} + \mathrm{NUT}_R3 \cdot \mathrm{N} & \overset{k_{PR}}{\longrightarrow} & O_R \cdot \mathrm{CI}_2 \cdot \mathrm{RNAP} + \mathrm{NUT}_R3 \cdot \mathrm{N} + 10\mathrm{CII} \\
O_R \cdot \mathrm{Cro}_2 \cdot \mathrm{RNAP} + \mathrm{NUT}_R4 & \overset{0.5*k_{PR}}{\longrightarrow} & O_R \cdot \mathrm{Cro}_2 \cdot \mathrm{RNAP} + \mathrm{NUT}_R4 + 10\mathrm{CII} \\
O_R \cdot \mathrm{Cro}_2 \cdot \mathrm{RNAP} + \mathrm{NUT}_R4 \cdot \mathrm{N} & \overset{k_{PR}}{\longrightarrow} & O_R \cdot \mathrm{Cro}_2 \cdot \mathrm{RNAP} + \mathrm{NUT}_R4 \cdot \mathrm{N} + 10\mathrm{CII}
\end{array}
$$

Figure B.5. Model for CII production from O_R operator ($K_{\mathrm{NUT}} = 0.2$).

$$
\begin{array}{rcl}
\mathrm{CII} + \mathrm{CIII} & \overset{\frac{K_8 k_{10} |\mathrm{CII}|}{1 + K_{11} |\mathrm{CIII}|}}{\longleftrightarrow} & \mathrm{CIII} \\
\mathrm{CIII} & \overset{k_13}{\longrightarrow} & ()
\end{array}
$$

Constant	*Value*
K_8	0.001
k_{10}	$8.0\mathrm{s}^{-1}$
K_{11}	0.001
k_{13}	$0.002\mathrm{s}^{-1}$

Figure B.6. Model for CII and CIII degradation.

$O_R + \mathrm{CI}_2$	$\overset{K_{OR2}}{\longleftrightarrow}$	$O_R \cdot \mathrm{CI}_2$
$O_R + \mathrm{Cro}_2$	$\overset{K_{OR5}}{\longleftrightarrow}$	$O_R \cdot \mathrm{Cro}_2$
$O_R + \mathrm{RNAP}$	$\overset{K_{OR8}}{\longleftrightarrow}$	$O_R3 \cdot \mathrm{RNAP}$
$O_R + \mathrm{RNAP}$	$\overset{K_{OR9}}{\longleftrightarrow}$	$O_R12 \cdot \mathrm{RNAP}$
$O_R3 \cdot \mathrm{RNAP}$	$\overset{k_{PRMb}}{\longrightarrow}$	$O_R3 \cdot \mathrm{RNAP} + 10\mathrm{CI}$
$O_R12 \cdot \mathrm{RNAP}$	$\overset{k_{PR}}{\longrightarrow}$	$O_R12 \cdot \mathrm{RNAP} + 10\mathrm{Cro}$
$O_R + 2\mathrm{CI}_2$	$\overset{K_{OR10}}{\longleftrightarrow}$	$O_R \cdot 2\mathrm{CI}_2$
$O_R + 2\mathrm{Cro}_2$	$\overset{K_{OR13}}{\longleftrightarrow}$	$O_R \cdot 2\mathrm{Cro}_2$
$O_R + \mathrm{CI}_2 + \mathrm{Cro}_2$	$\overset{K_{OR17}}{\longleftrightarrow}$	$O_R \cdot \mathrm{CI}_2 \cdot \mathrm{Cro}_2$
$O_R + 2\mathrm{RNAP}$	$\overset{K_{OR16}}{\longleftrightarrow}$	$O_R \cdot 2\mathrm{RNAP}$
$O_R + \mathrm{CI}_2 + \mathrm{RNAP}$	$\overset{K_{OR23}}{\longleftrightarrow}$	$O_R \cdot \mathrm{CI}_2 \cdot \mathrm{RNAP}$
$O_R + \mathrm{Cro}_2 + \mathrm{RNAP}$	$\overset{K_{OR26}}{\longleftrightarrow}$	$O_R \cdot \mathrm{Cro}_2 \cdot \mathrm{RNAP}$
$O_R + \mathrm{RNAP} + \mathrm{CI}_2$	$\overset{K_{OR24}}{\longleftrightarrow}$	$O_R \cdot \mathrm{RNAP} \cdot \mathrm{CI}_2$
$O_R13 + \mathrm{RNAP} + \mathrm{CI}_2$	$\overset{K_{OR25}}{\longleftrightarrow}$	$O_R13 \cdot \mathrm{RNAP} \cdot \mathrm{CI}_2$
$O_R + \mathrm{RNAP} + \mathrm{Cro}_2$	$\overset{K_{OR27}}{\longleftrightarrow}$	$O_R \cdot \mathrm{RNAP} \cdot \mathrm{Cro}_2$
$O_R \cdot 2\mathrm{RNAP}$	$\overset{k_{PRMb}}{\longrightarrow}$	$O_R \cdot 2\mathrm{RNAP} + 10\mathrm{CI}$
$O_R \cdot 2\mathrm{RNAP}$	$\overset{k_{PR}}{\longrightarrow}$	$O_R \cdot 2\mathrm{RNAP} + 10\mathrm{Cro}$
$O_R \cdot \mathrm{CI}_2 \cdot \mathrm{RNAP}$	$\overset{k_{PR}}{\longrightarrow}$	$O_R \cdot \mathrm{CI}_2 \cdot \mathrm{RNAP} + 10\mathrm{Cro}$
$O_R \cdot \mathrm{Cro}_2 \cdot \mathrm{RNAP}$	$\overset{k_{PR}}{\longrightarrow}$	$O_R \cdot \mathrm{Cro}_2 \cdot \mathrm{RNAP} + 10\mathrm{Cro}$
$O_R \cdot \mathrm{RNAP} \cdot \mathrm{CI}_2$	$\overset{k_{PRM}}{\longrightarrow}$	$O_R \cdot \mathrm{RNAP} \cdot \mathrm{CI}_2 + 10\mathrm{CI}$
$O_R13 \cdot \mathrm{RNAP} \cdot \mathrm{CI}_2$	$\overset{k_{PRMb}}{\longrightarrow}$	$O_R13 \cdot \mathrm{RNAP} \cdot \mathrm{CI}_2 + 10\mathrm{CI}$
$O_R \cdot \mathrm{RNAP} \cdot \mathrm{Cro}_2$	$\overset{k_{PRMb}}{\longrightarrow}$	$O_R \cdot \mathrm{RNAP} \cdot \mathrm{Cro}_2 + 10\mathrm{CI}$
$O_R + 3\mathrm{CI}_2$	$\overset{K_{OR29}}{\longleftrightarrow}$	$O_R \cdot 3\mathrm{CI}_2$
$O_R + 3\mathrm{Cro}_2$	$\overset{K_{OR30}}{\longleftrightarrow}$	$O_R \cdot 3\mathrm{Cro}_2$
$O_R + 2\mathrm{CI}_2 + \mathrm{Cro}_2$	$\overset{K_{OR31}}{\longleftrightarrow}$	$O_R \cdot 2\mathrm{CI}_2 \cdot \mathrm{Cro}_2$
$O_R + \mathrm{CI}_2 + 2\mathrm{Cro}_2$	$\overset{K_{OR34}}{\longleftrightarrow}$	$O_R \cdot \mathrm{CI}_2 \cdot 2\mathrm{Cro}_2$
$O_R + \mathrm{RNAP} + 2\mathrm{CI}_2$	$\overset{K_{OR37}}{\longleftrightarrow}$	$O_R \cdot \mathrm{RNAP} \cdot 2\mathrm{CI}_2$
$O_R + \mathrm{RNAP} + 2\mathrm{Cro}_2$	$\overset{K_{OR38}}{\longleftrightarrow}$	$O_R \cdot \mathrm{RNAP} \cdot 2\mathrm{Cro}_2$
$O_R + \mathrm{RNAP} + \mathrm{Cro}_2 + \mathrm{CI}_2$	$\overset{K_{OR39}}{\longleftrightarrow}$	$O_R \cdot \mathrm{RNAP} \cdot \mathrm{Cro}_2 \cdot \mathrm{CI}_2$
$O_R + \mathrm{RNAP} + \mathrm{CI}_2 + \mathrm{Cro}_2$	$\overset{K_{OR40}}{\longleftrightarrow}$	$O_R \cdot \mathrm{RNAP} \cdot \mathrm{CI}_2 \cdot \mathrm{Cro}_2$
$O_R \cdot \mathrm{RNAP} \cdot 2\mathrm{CI}_2$	$\overset{k_{PRM}}{\longrightarrow}$	$O_R \cdot \mathrm{RNAP} \cdot 2\mathrm{CI}_2 + 10\mathrm{CI}_2$
$O_R \cdot \mathrm{RNAP} \cdot 2\mathrm{Cro}_2$	$\overset{k_{PRMb}}{\longrightarrow}$	$O_R \cdot \mathrm{RNAP} \cdot 2\mathrm{Cro}_2 + 10\mathrm{CI}$
$O_R \cdot \mathrm{RNAP} \cdot \mathrm{Cro}_2 \cdot \mathrm{CI}_2$	$\overset{k_{PRMb}}{\longrightarrow}$	$O_R \cdot \mathrm{RNAP} \cdot \mathrm{Cro}_2 \cdot \mathrm{CI}_2 + 10\mathrm{CI}$
$O_R \cdot \mathrm{RNAP} \cdot \mathrm{CI}_2 \cdot \mathrm{Cro}_2$	$\overset{k_{PRM}}{\longrightarrow}$	$O_R \cdot \mathrm{RNAP} \cdot \mathrm{CI}_2 \cdot \mathrm{Cro}_2 + 10\mathrm{CI}$

Figure B.7. Model for the λ switch.

$Constant$	$Value$	$Constant$	$Value$
K_{OR2}	0.2165	K_{OR27}	0.01186
K_{OR5}	0.449	K_{OR29}	0.00081
K_{OR8}	0.1362	K_{OR30}	0.00069
K_{OR9}	0.69422	K_{OR31}	0.02133
K_{OR10}	0.06568	K_{OR34}	0.00322
K_{OR13}	0.03342	K_{OR37}	0.0079
K_{OR16}	0.09455	K_{OR38}	0.00026
K_{OR17}	0.1779	K_{OR39}	0.00112
K_{OR23}	0.00967	K_{OR40}	0.00008
K_{OR24}	0.0019	k_{PRMb}	0.001s^{-1}
K_{OR25}	0.02569	k_{PRM}	0.011s^{-1}
K_{OR26}	0.25123	k_{PR}	0.014s^{-1}

Figure B.8. Constants for the λ switch model.

REFERENCES

[1] BioATACS. http://www.async.ece.utah.edu/bio-bin/biover/.

[2] BioSim. http://www.async.ece.utah.edu/tools.

[3] U.S. Department of Energy Genomics: GTL program. http://genomicsgtl.energy.gov.

[4] The YeastPheromoneModel.org wiki. http://yeastpheromonemodel.org.

[5] Abraham, J. M., Freitag, C. S., Clements, J. R., and Eisenstein, B. I. An invertible element of DNA controls phase variation of type 1 fimbriae of *Escherichia coli*. *Proc Natl Acad Sci U S A. 82* (1985), 5724–5727.

[6] Achimescu, S., and Lipan, O. Signal propagation in nonlinear stochastic gene regulatory networks. *IEE Proc. Sys. Bio. 153* (2006), 120–134.

[7] Ackers, G. K., Johnson, A. D., and Shea, M. A. Quantitative model for gene regulation by λ phage repressor. *Proc. Natl. Acad. Sci. USA 79* (1982), 1129–1133.

[8] Arkin, A., and Fletcher, D. Fast, cheap and somewhat in control. *Genome Biology 7*, 8 (2006), 114.

[9] Arkin, A., Ross, J., and McAdams, H. Stochastic kinetic analysis of developmental pathway bifurcation in phage lambda-infected escherichia coli cells. *Genetics 149* (1998), 1633–1648.

[10] Ascher, U. M., and Petzold, L. R. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. SIAM, 1998.

[11] Atlung, T., and Ingmer, H. H-NS: a modulator of environmentally regulated gene expression. *Molecular Microbiology 24* (1997), 7–17.

[12] Baldi, P., and Hatfield, G. W. *DNA Microarrays and Gene Expression*. Cambridge University Press, 2002.

[13] Baltimore, D. Our genome unveiled. *Nature 409*, 6822 (Feb. 2001), 814–816.

[14] Barker, N., Myers, C., and Kuwahara, H. Learning genetic regulatory network connectivity from time series data. In *The 19th International Conference on Industrial, Engineering, and Other Applications of Applied Intelligent Systems* (2006).

[15] Berry, R. S., Rice, S. A., and Ross, J. *Physical Chemistry (2nd Edition)*. Oxford University Press, New York, 2000.

[16] BioSPICE. `http://www.biospice.org/`.

[17] Blomfield, I. C., Calie, P. J., Eberhardt, K. J., McClain, M. S., and Eisenstein, B. I. Lrp stimulates phase variation of type 1 fimbriation in *Escherichia coli* k-12. *J Bacteriol. 175* (1993), 27–36.

[18] Blomfield, I. C., Kulasekara, D. H., and Eisenstein, B. I. Integration host factor stimulates both FimB- and FimE- mediated site-specific DNA inversion that controls phase variation of type 1 fimbriae expression in escherichia coli. *Mol Microbiol 23* (1997), 705–717.

[19] Blomfield, I. C., McClain, M. S., Princ, J. A., Calie, P. J., and Eisenstein, B. I. Type 1 fimbriation and *fimE* mutants of *Escherichia coli* k-12. *J. Bacteriol. 173* (1991), 5298–5307.

[20] Borisuk, M. T., and Tyson, J. J. Bifurcation analysis of a model of mitotic control in frog eggs. *Journal of Theoretical Biology 195*, 1 (Nov. 1998), 69–85.

[21] Brent, R. A partnership between biology and engineering. *Nature Biotechnology 22* (2004), 1211 – 1214.

[22] Briggs, G. E., and Haldane, J. B. S. A note on the kinetics of enzyme action. *Biochem. J. 19* (1925), 339–339.

[23] Bunker, D. L., Garrett, B., Kleindienst, T., and Long III, G. S. Discrete simulation methods in combustion kinetics. *Combustion and Flame 23* (1974), 373–379.

[24] Cai, L., Friedman, N., and Xie, X. S. Stochastic protein expression in individual cells at the single molecule level. *Nature 440*, 7082 (Mar. 2006), 358–362.

[25] Cao, Y., Gillespie, D., and Petzold, L. Accelerated stochastic simulation of the stiff enzyme-substrate reaction. *J. Chem. Phys. 123* (2005).

[26] Cao, Y., Gillespie, D., and Petzold, L. Avoiding negative populations in explicit tau leaping. *Journal of Chemical Physics 123* (2005).

[27] Cao, Y., Gillespie, D., and Petzold, L. The slow-scale stochastic simulation algorithm. *Journal of Chemical Physics 122* (2005).

[28] Cao, Y., Gillespie, D., and Petzold, L. Efficient stepsize selection for the tau-leaping method. *J. Chem. Phys.* (2006).

[29] Cao, Y., Li, H., and Petzold, L. Efficient formulation of the stochastic simulation algorithm for chemically reacting system. *J. Chem. Phys. 121* (2004), 4059–4067.

[30] Cao, Y., and Petzold, L. Trapezoidal tau-leaping formula for the stochastic simulation of biochemical systems. In *Foundations of Systems Biology in Engineering* (2005), pp. 149–152.

[31] Causton, H. C., Ren, B., Koh, S. S., Harbison, C. T., Kanin, E., Jennings, E. G., Lee, T. I., True, H. L., Lander, E. S., and Young, R. A. Remodeling of yeast genome expression in response to environmental changes. *Mol. Biol. Cell 12*, 2 (Feb. 2001), 323–337.

[32] Chatterjee, A., Vlachos, D. G., and Katsoulakis, M. A. Binomial distribution based tau-leap accelerated stochastic simulation. *J. Chem. Phys. 122* (2005), 024112.

[33] Collins, F. S., Green, E. D., Guttmacher, A. E., and Guyer, M. S. A vision for the future of genomics research. *Nature 422*, 6934 (Apr. 2003), 835–847.

[34] Crick, F. H. Central dogma of molecular biology. *Nature 227* (1970), 561–563.

[35] Dacol, D., and Rabitz, H. Sensitivity analysis of stochastic kinetic models. *J. Math. Phys. 25* (1984).

[36] Davidson, E. H., Rast, J. P., Oliveri, P., Ransick, A., Calestani, C., Yuh, C.-H., Minokawa, T., Amore, G., Hinman, V., Arenas-Mena, C., Otim, O., Brown, C. T., Livi, C. B., Lee, P. Y., Revilla, R., Rust, A. G., Pan, Z. j., Schilstra, M. J., Clarke, P. J. C., Arnone, M. I., Rowen, L., Cameron, R. A., McClay, D. R., Hood, L., and Bolouri, H. A genomic regulatory network for development. *Science 295*, 5560 (Mar. 2002), 1669–1678.

[37] DeRisi, J. L., Iyer, V. R., and Brown, P. O. Exploring the metabolic and genetic control of gene expression on a genomic scale. *Science 278*, 5338 (Oct. 1997), 680–686.

[38] Edwards, J. S., Ibarra, R. U., and Palsson, B. O. In silico predictions of *Escherichia coli* metabolic capabilities are consistent with experimental data. *Nature Biotechnology 19* (2001), 125 – 130.

[39] Eisenstein, B. I. Phase variation of type 1 fimbriae in escherichia coli is under transcriptional control. *Science 214* (1981), 337–339.

[40] Elowitz, M. B., Levine, A. J., Siggia, E. D., and Swain, P. S. Stochastic gene expression in a single cell. *Science 297* (2002), 1183–1186.

[41] Fall, C., Marland, E., Wagner, J., and Tyson, J., Eds. *Computational Cell Biology*. Springer, 2002.

[42] Finney, A., and Hucka, M. Systems Biology Markup Language (SBML) level 2: Structures and facilities for model definitions, 2003.

[43] Friedman, N., Linial, M., Nachman, I., and Pe'er, D. Using bayesian networks to analyze expression data. *Journal of Computational Biology 7*, 3–4 (2000), 601–620.

[44] Funahashi, A., Tanimura, N., Morohashi, M., and Kitano, H. Celldesigner: A process diagram editor for gene-regulatory and biochemical networks. *BIOSILICO 1* (2003), 159–162.

[45] Gally, D. L., Bogan, J. A., Eisenstein, B. I., and Blomfield, I. C. Environmental regulation of the fim switch controlling type 1 fimbrial phase variation in escherichia coli k-12: Effects of temperature and media. *J Bacteriol. 175* (1993), 6186–6193.

[46] Gally, D. L., Rucker, T. J., and Blomfield, I. C. The leucine-responsive regulatory protein binds to the fim switch to control phase variation of type 1 fimbrial expression in *Escherichia coli* k-12. *J Bacteriol. 176* (1994), 5665–5672.

[47] Gardiner, C. W. *Handbook of Stochastic Methods: For Physics, Chemistry and the Natural Sciences*, 3rd ed. Springer, 2004.

[48] Gibson, M., and Bruck, J. An efficient algorithm for generating trajectories of stochastic gene regulation reactions. Tech. rep., California Institute of Technology, 1998.

[49] Gibson, M., and Bruck, J. Efficient exact stochastic simulation of chemical systems with many species and many channels. *J. Phys. Chem. A 104* (2000), 1876–1889.

[50] Gillespie, D. The chemical langevin equation. *Journal of Chemical Physics 113*, 1 (2000).

[51] Gillespie, D., and Petzold, L. Improved leap-size selection for accelerated stochastic simulation. *Journal of Chemical Physics 119* (2003).

[52] Gillespie, D. T. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics 22* (1976), 403–434.

[53] Gillespie, D. T. Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem. 81*, 25 (1977), 2340–2361.

[54] Gillespie, D. T. *Markov Processes An Introduction for Physical Scientists.* Academic Press, Inc., 1992.

[55] Gillespie, D. T. A rigorous derivation of the chemical master equation. *Physica A 188* (1992), 404–425.

[56] Gillespie, D. T. Approximate accelerated stochastic simulation of chemically reacting systems. *Journal of Chemical Physics 115*, 4 (2001), 1716–1733.

[57] Gillespie, D. T. *Handbook of Materials Modeling*. Springer, 2005, ch. 5.11, pp. 1735–1752.

[58] Golding, I., Paulsson, J., Zawilski, S. M., and Cox, E. C. Real-time kinetics of gene activity in individual bacteria. *Cell 123* (2005), 1025–1036.

[59] Gunawan, R., Cao, Y., Petzold, L., and Doyle, F. J. Sensitivity analysis of discrete stochastic systems. *Biophysical Journal 88* (2005), 2530–2540.

[60] Guptasarma, P. Does replication-induced transcription regulate synthesis of the myriad low copy number proteins of *Escherichia coli*? *BioEssays 17* (1995), 987–997.

[61] Henderson, I., Owen, P., and Nataro, J. Molecular switches - the ON and OFF of bacterial phase variation. *Molecular Microbiology 33* (1999), 919–932.

[62] Henri, V. Lois générales de laction des diastases. Hermann, Paris.

[63] Hermsen, R., Tans, S., and ten Wolde, P. R. Transcriptional regulation by competing transcription factor modules. *PLoS Computational Biology 2*, 12 (Dec. 2006), 164–.

[64] Jong, H. D. Modeling and simulation of genetic regulatory systems: A literature review. *J. Comp. Biol. 9*, 1 (2002), 67–103.

[65] Keener, J., and Sneyd, J. *Mathematical Physiology*. Springer, 1998.

[66] Kincaid, D., and Cheney, W. *Numerical Analysis*, 2nd ed. Brooks/Cole Publishing Company, 1996.

[67] Kitano, H. Computational systems biology. *Nature 420*, 6912 (Nov. 2002), 206–210.

[68] Kitano, H. Systems biology: A brief overview. *Science 295* (2002), 1662 – 1664.

[69] Klemm, P. Two regulatory fim genes, fimB and fimE, control the phase variation of type 1 fimbriae in escherichia coli. *EMBO 5* (1986), 1389–1393.

[70] Kourilsky, P. Lysogenization by bacteriophage lambda: I. multiple infection and the lysogenic response. *Mol. Gen. Genet. 122* (1973), 183–195.

[71] Kulasekara, H., and Blomfield, I. The molecular basis for the specificity of fimE in the phase variation of type 1 fimbriae of escherichia coli k-12. *Molecular Microbiology 31* (1999), 1171–1181.

[72] Kuwahara, H., and Myers, C. Production-passage-time approximation: A new approximation method to accelerate the simulation process of enzymatic reactions. In *The 11th Annual International Conference on Research in Computational Molecular Biology* (2007).

[73] Kuwahara, H., Myers, C., Barker, N., Samoilov, M., and Arkin, A. Asynchronous abstraction methodology for genetic regulatory networks. In *The Third International Workshop on Computational Methods in Systems Biology* (2005).

[74] Kuwahara, H., Myers, C., Barker, N., Samoilov, M., and Arkin, A. Automated abstraction methodology for genetic regulatory networks. *Trans. on Comput. Syst. Biol. VI* (2006), 150–175.

[75] Kuwahara, H., Myers, C., and Samoilov, M. Abstracted stochastic analysis of type 1 pili expression in E. coli. In *The 2006 International Conference on Bioinformatics and Computational Biology* (2006).

[76] Levine, M., and Tjian, R. Transcription regulation and animal diversity. *Nature 424*, 6945 (July 2003), 147–151.

[77] Lodish, H., Berk, A., Zipursky, L. S., Matsudaira, P., Baltimore, D., and Darnell, J. *Molecular Cell Biology.* W. H. Freeman and Company, 1999.

[78] Lotka, A. J. Undamped oscillations derived from the law of mass action. *J. Am. Chem. Soc. 42* (1920), 1595.

[79] Maxam, A. M., and Gilbert, W. A new method for sequencing dna. *Proc. Natl. Acad. Sci. 74* (1977), 560564.

[80] McAdams, H. H., and Arkin, A. Stochastic mechanisms in gene expression. *Proceedings of the National Academy of Sciences USA 94*, 3 (1997), 814–819.

[81] McAdams, H. H., and Arkin, A. Genetic regulation at the nanomolar scale: It's a noisy business. *Trends Genet. 15*, 2 (1999), 65–69.

[82] McClain, M. S., Blomfield, I. C., Eberhardt, K. J., and Eisenstein, B. I. Inversion-independent phase variation of type 1 fimbriae in *Escherichia coli. J Bacteriol. 175* (1993), 43354344.

[83] McQuarrie, D. A. Stochastic approach to chemical kinetics. *J. Applied Prob. 4* (1967), 413–478.

[84] Michaelis, L., and Menten, M. Die kinetik der invertinwirkung. *Biochem. Z. 49* (1913), 333–369.

[85] Munsky, B., and Khammash, M. The finite state projection algorithm for the solution of the chemical master equation. *J. Chem. Phys. 124* (2006).

[86] Newman, J. R. S., Ghaemmaghami, S., Ihmels, J., Breslow, D. K., Noble, M., DeRisi, J. L., and Weissman, J. S. Single-cell proteomic analysis of s. cerevisiae reveals the architecture of biological noise. *Nature 441*, 7095 (June 2006), 840–846.

[87] Nguyen, N., Kuwahara, H., Myers, C., and Keener, J. The design of a genetic muller C-element. In *The 13th IEEE International Symposium on Asynchronous Circuits and Systems* (2007).

[88] Olsen, P. B., and Klemm, P. Localization of promoters in the fim gene cluster and the effect of H-NS on the transcription of fimB and fimE. *FEMS Microbiology Letters 116* (1994), 95–100.

[89] Olsen, P. B., Schembri, M. A., Gally, D. L., and Klemm, P. Differential temperature modulation by H-NS of the fimb and fime recombinase genes which control the orientation of the type 1 fimbrial phase switch. *FEMS Microbiology Letters 162* (1998), 17–23.

[90] Ono, S., Goldberg, M. D., Olsson, T., Esposito, D., Hinton, J. C. D., and Ladbury, J. E. H-NS is a part of a thermally controlled mechanism for bacterial gene regulation. *Biochem J. 391* (2005), 203–213.

[91] Oshima, T., Ito, K., Kabayama, H., and Nakamura, Y. Regulation of *lrp* gene expression by H-NS and Lrp proteins in *Escherichia coli*: Dominant negative mutations in *lrp*. *Molecular and General Genetics MGG 247* (1995), 521–528.

[92] Pedraza, J. M., and van Oudenaarden, A. Noise Propagation in Gene Networks. *Science 307*, 5717 (2005), 1965–1969.

[93] Peles, S., Munsky, B., and Khammash, M. Reduction and solution of the chemical master equation using time scale separation and finite state projection. *J. Chem. Phys. 125* (2006).

[94] Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T. *Numerical Recipes in C: The art of Scientific Computing*, 2nd ed. Cambridge University Press, 1992.

[95] Ptashne, M. *A Genetic Switch*. Cell Press & Blackwell Scientific Publishing, 1992.

[96] Rao, C. V., and Arkin, A. P. Stochastic chemical kinetics and the quasi-steady-state assumption: Application to the gillespie algorithm. *J. Phys. Chem. 118*, 11 (2003).

[97] Rao, C. V., Wolf, D. M., and Arkin, A. P. Control, exploitation and tolerance of intracellular noise. *Nature 420* (2002), 231–238.

[98] Rathinam, M., Cao, Y., Petzold, L., and Gillespie, D. Stiffness in stochastic chemically reacting systems: The implicit tau-leaping method. *Journal of Chemical Physics 119* (2003), p12784–94.

[99] Roesch, P. L., and Blomfield, I. C. Leucine alters the interaction of the leucine-responsive regulatory protein (Lrp) with the fim switch to stimulate site-specific recombination in Escherichia coli. *Molecular Microbiology 27* (1998), 751–761.

[100] Ruvkun, G., and Hobert, O. The taxonomy of developmental control in caenorhabditis elegans. *Science 282*, 5396 (Dec. 1998), 2033–2041.

[101] Samoilov, M. Stochastic effects in enzymatic biomolecular systems: Framework, fast & slow species and quasi-steady state approximations. In *Workshop on Dynamical Stochastic Modeling in Biology* (2003).

[102] Samoilov, M., Plyasunov, S., and Arkin, A. P. Stochastic amplification and signaling in enzymatic futile cycles through noise-induced bistability with oscillations. *Proceedings of the National Academy of Sciences US 102*, 7 (2005), 2310–5.

[103] Samoilov, M. S., and Arkin, A. P. Deviant effects in molecular reaction pathways. *Nature Biotechnology 24* (2006), 1235–1240.

[104] Schena, M., Shalon, D., Davis, R. W., and Brown, P. O. Quantitative monitoring of gene expression patterns with a complementary dna microarray. *Science 270*, 5235 (Oct. 1995), 467–470.

[105] Schlögl, F. On thermodynamics near a steady state. *Zeitschrift für Physik A Hadrons and Nuclei V248*, 5 (Oct. 1971), 446–458.

[106] Schnell, S., and Mendoza, C. Enzyme kinetics of multiple alternative substrates. *Journal of Mathematical Chemistry 27* (2000), 155–170.

[107] Segel, Lee A., and Slemrod, Marshall. The quasi-steady-state assumption: A case study in perturbation. *SIAM Review 31*, 3 (sep 1989), 446–477.

[108] Shea, M. A., and Ackers, G. K. The or control system of bacteriophage lambda: a physical-chemical model for gene regulation. *J. Mol. Biol. 181* (1985), 211–230.

[109] Singer, K. Application of the theory of stochastic processes to the study of irreproducible chemical reactions and nucleation processes. *J. Royal Statistical Society 15* (1953), 92–106.

[110] Stewart, W. J. *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press, 1994.

[111] The International Human Genome Mapping Consortium. Initial sequencing and analysis of the human genome. *Nature 409*, 6822 (Feb. 2001), 860–921.

[112] Tian, T., and Burrage, K. Binomial leap methods for simulating stochastic chemical kinetics. *J. Chem. Phys 121* (2004), 10356–10364.

[113] Van der Weyden, L., Adams, D. J., and Bradley, A. Tools for targeted manipulation of the mouse genome. *Physiol. Genomics 11* (2002), 133–164.

[114] van Kampen, N. G. *Stochastic Processes in Physics and Chemistry.* Elsevier, 1992.

[115] Volterra, V. Leccons sur la théorie mathématique de lal lutte pour la vie. *Gauthiers-Villars, Paris* (1931).

[116] Wolf, D. M., and Arkin, A. P. Fifteen minutes of *fim*: Control of type 1 pili expression in *E. coli*. *OMICS: A Journal of Integrative Biology 6*, 1 (2002), 91–114.

[117] Yu, J., Smith, V. A., Wang, P. P., Hartemink, A. J., and Jarvis, E. D. Advances to bayesian network inference for generating causal networks from observational biological data. *Bioinformatics 20* (December 2004), 3594–3603.

[118] Zhou, J., Yockman, J. W., Kim, S. W., and Kern, S. E. Intracellular kinetics of non-viral gene delivery using polyethylenimine carriers. *Pharmaceutical Research 24* (2007), 1079–1087.