Hybrid Modeling of Dynamic Networks: Towards Standardized Representation and Automated Model Design

by

Khaled Sayed Ahmed Sayed

B. S. in Systems and Biomedical Engineering, Cairo University, Egypt, 2012

M. Sc. in Systems and Biomedical Engineering, Cairo University, Egypt, 2015

Submitted to the Graduate Faculty of the

Swanson School of Engineering in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2019

UNIVERSITY OF PITTSBURGH

SWANSON SCHOOL OF ENGINEERING

This dissertation was presented

by

Khaled Sayed Ahmed Sayed

It was defended on

November 13, 2019

and approved by

Zhi-Hong Mao, Ph.D, Professor, Department of Electrical and Computer Engineering

Amro El-Jaroudi, Ph.D, Associate Professor, Department of Electrical and Computer Engineering

Samuel J. Dickerson, Ph.D, Assistant Professor, Department of Electrical and Computer Engineering

James R. Faeder, Ph.D, Associate Professor, Department of Computational and Systems Biology

Cheryl A. Telmer, Ph.D, Research Biologist, Department of Biological Sciences, Carnegie Mellon University

Dissertation Director: Natasa Miskov-Zivanov, Ph.D, Assistant Professor, Department of Electrical and Computer Engineering

Hybrid Modeling of Dynamic Networks: Towards Standardized Representation and Automated Model Design

Khaled Sayed Ahmed Sayed, Ph.D.

University of Pittsburgh, 2019

Computational modeling has become an efficient tool for studying complex systems such as biological, environmental, and economic systems. For instance, computational models of intra- and inter-cellular networks can provide important insights into biological systems and guide scientists toward the missing information, and thus, speed up biological discoveries and reduce the cost and time of conducting unnecessary or impractical biochemical experiments. Creating accurate models requires the knowledge not only about the key components of the system and their causal or mechanistic relationships, but also about relative timing of events, as well as the various timescales of relevant processes. In biology, different timescales of processes can be captured with reaction rate constants and modeled with differential equations, however, these quantitative constants are often unknown or difficult to measure, especially for large models. Therefore, abstraction methods such as logical and discrete modeling have been developed to overcome the limitations of uncertain information. Although several methods to incorporate timing into discrete models have been proposed, a methodology for modeling common biological motifs at varying timescales and at different regulatory conditions is necessary. In this work, we introduce a methodology to standardize modeling of common motifs occurring in many domains, with a special focus on accurately incorporating timing information into dynamic network models. Additionally, we propose several deterministic and stochastic simulation approaches that can be used to analyze these models. Overall, the goal of this work is to facilitate automated design of

hybrid models of dynamic networks. Finally, we demonstrate our efforts using existing models of intra- and inter-cellular signaling occurring in T cell differentiation and budding yeast cell cycle. We also apply the developed methods on socio-economic systems such as food insecurity in South Sudan in order to study the factors leading to humanitarian crises.

Table of Contents

Acknowledgement ..... **xvii**

1.0 Introduction **1**

1.1 Motivation **1**

1.2 Scope **2**

1.3 Contribution **6**

1.4 Dissertation Statement **7**

1.5 Dissertation Organization **7**

2.0 Background **8**

2.1 Logical Modeling **8**

2.2 Timing in Logical Models **9**

2.2.1 Buffer Insertion **10**

2.2.2 "Dummy" Nodes **12**

2.2.3 Boolean Delay Equations (BDEs) **13**

2.2.4 Piecewise Linear Differential Equations (PWLDEs) **15**

2.2.5 Overview of the Current Methods **17**

3.0 DiSH Simulator **19**

3.1 Simulation Schemes **19**

3.1.1 Simultaneous (SMLN) Scheme **21**

3.1.2 Random-order Sequential (RSQ) Scheme **23**

3.1.3 Round-Based Random-order Sequential (RB-RSQ) Scheme **23**

3.1.4 Step-Based Random-order Sequential (SB-RSQ) Scheme **24**

3.1.5 Ranked-order Sequential (RKSQ) Scheme 26

3.2 Additional Functionalities 26

3.2.1 Grouped rules 26

3.2.2 Toggle Implementation 27

3.3 Discrete DiSH 28

4.0 **Model Representation 33**

4.1 Tabular Representation Format 33

4.2 Representation of Common Biological Motifs 37

4.2.1 Simple Interactions 38

4.2.2 Binding Interactions 40

4.2.3 Nested Interactions 41

4.2.3.1 Positive Regulation of Activation 41

4.2.3.2 Positive Regulation of Inhibition 42

4.2.3.3 Negative Regulation of Activation. 43

4.2.3.4 Negative Regulation of Inhibition. 43

4.2.4 Gene expression 44

4.2.5 Receptor activation 45

4.2.6 Translocation Motif 46

5.0 Time in Discrete Models 48

5.1 Time Modeling 49

5.1.1 General State Transitions 51

5.1.2 **Conditioned State Transitions 53**

5.1.2.1 Reset Delays 54

5.1.2.2 No-Reset Delays 55

5.1.3 Spontaneous and Balancing delays 55

5.2 Timing in Motifs 56

5.2.1 Receptor Activation 56

5.2.2 Translocation 57

5.2.3 Gene Expression 57

5.2.4 Phosphorylation, Dephosphorylation, and Ubiquitination 58

5.2.5 Complex Formation 58

5.3 Timing under Different Simulation Schemes 58

5.3.1 SMLN Scheme 59

5.3.2 RSQ Scheme 60

5.3.3 Random-delay SMLN update scheme 61

5.3.4 Overview of the Proposed Delay Methods 62

5.4 Inferring Timing Parameters from Data 64

5.4.1 Data preparation 65

5.4.2 Optimization-based Simulations 67

6.0 Applications 70

6.1 Naïve T cell differentiation 70

6.1.1 **Model Description 71**

6.1.2 Effect of simulation schemes 73

6.1.3 Effect of scoring functions 78

6.1.4 Effect of time modeling 90

6.2 Buddying yeast cell cycle 98

6.2.1 Model Description 99

6.2.2 Model Simulations..... 100

6.3 Food security in South Sudan..... 105

6.3.1 Model Description 105

6.3.2 Model Simulations..... 107

6.3.3 Effect of timing 111

7.0 Conclusion and Future Work 118

Bibliography 121

List of Tables

Table 2-1 A comparison between the available time abstraction methods..... 18

Table 4-1 Element type and ID database..... 35

Table 4-2 The list of cellular locations and their IDs from the Gene Ontology (GO) database. 35

Table 4-3 A tabular representation for the toy example in Figure 4-1(a). 39

Table 4-4 A tabular representation for the toy example in Figure 4-1(b). 39

Table 4-5 A tabular representation for the nested interactions in Figure 4-3. 42

Table 4-6 A tabular representation for the gene expression motif in Figure 4-4..... 45

Table 4-7 A tabular representation for the gene expression motif in Figure 4-5..... 46

Table 4-8 A tabular representation for the gene expression motif in Figure 4-6..... 47

Table 5-1 A truth table with delays for $x3$ in the toy example of Figure 5-2. 54

Table 5-2 Highlights of the proposed methods in terms of the comparison in Table 2-1. ... 63

Table 6-1 Values of delay variables in Figure 6-7..... 91

Table 6-2 Values of delay variables in Figure 6-7 before and after applying the optimization algorithm in section 5.4..... 97

Table 6-3 Conditions for activation of model elements with corresponding activation and inhibition delays. 101

Table 6-4 Manual and optimized delay steps assigned to model elements. S represents spontaneous delays, $\theta k \rightarrow l$ represents the regulation delay for the transition from discrete level k to l, and B represents the balancing behavior delays (balancing behabior is set to decrease)..... 113

Table 6-5 Manual and optimized delay steps for the reset of model elements. 114

List of Figures

Figure 2-1 A toy example of three interacting elements and the corresponding Boolean update functions. 9

Figure 2-2 a) A toy example of four interacting components, b) the equivalent logic circuit. 10

Figure 2-3 The representation of the toy example in Figure 2-2 using the Buffer Insertion method. (a) The toy example with the added delay nodes, (b) the equivalent logic circuit, and (c) the timing diagram of $x4$ using the simultaneous update scheme. 11

Figure 2-4 The representation of the toy example in Figure 2-2 using the dummy nodes method. (a) The toy example with the added delay nodes, (b) the equivalent logic circuit, and (c) the timing diagram of $x4$ using the simultanoues update scheme. 13

Figure 2-5 The representation of the toy example in Figure 2-2 using the BDEs. (a) The toy example with delays, (b) the equivalent logic circuit without a clock signal, and (c) the timing diagram of $x4$ over a continuous time period. 15

Figure 2-7 The timing diagram of $x4$ in Figure 2-6(a) under random changes in the regulators' values. 17

Figure 2-6 The representation of the toy example in Figure 2-2 using the Piecewise linear differential equations. (a) The modified toy example, (b) the equivalent logic circuit without a clock signal, and (c) the changes of $X4$ and $x4$ when $f4$ is updated. 17

Figure 3-1 a) A toy example: three nodes (A, B and C), and their update rules specified. b) Simulation schemes. 20

Figure 3-2 STG for the toy example in Figure 3-1(a), for (a) the SMLN scheme and (b) the RSQ scheme. Labels on graph edges indicate which elements are selected for update. .. 22

Figure 3-3 STG for the toy example in Figure 3-1(a) using the RB-RSQ scheme. 24

Figure 3-4 Logic rules and STG in the (a) USB-RSQ and (b) NUSB-RSQ scheme. 25

Figure 3-5 An example of ranked rules..... 26

Figure 3-6 (a) Grouped and Ranked rules, (b) Grouped rules with different rates, (c) two small examples of STG when grouped rules are used. 27

Figure 4-1 A toy example for a small network that can be simulated by calculating (a) logical scores or (b) arithmetic scores 39

Figure 4-2 Schematic representation of a situation common to many biological signaling pathways where the regulation of complex formation, $x2$ binding to $x3$, is regulated by a third protein, $x1$, so that the $x2/x3$ complex can activate $x4$ and $x5$. 40

Figure 4-3 Examples of nested interactions. a) Positive regulation of Activation interaction, b) Positive regulation of Inhibition interaction, c) Negative regulation of Activation interaction, d) Negative regulation of Inhibition interaction..... 42

Figure 4-4 Gene expression motif. 44

Figure 4-5 Receptor activation motif. 45

Figure 4-6 Translocation motif. 47

Figure 5-1 The representation of the toy example in Figure 2-2 using the proposed methods. (a) The modified toy example showing the propagation and the regulations delays, (b) The element update schematic using the logical scoring, and (c) the element update schematic using the arithmetic scoring. 51

Figure 5-2 A toy example of three interacting components. 54

Figure 5-3 Timing diagram for the toy example in Figure 5-1(a) using the SMLN simulation scheme. 60

Figure 5-4 Timing diagram for the toy example in Figure 5-1(a) using the RSQ simulation scheme 61

Figure 6-1 Trajectories for Foxp3 using different simulation schemes: a) SMLN, b) RB-RSQ, c) USB-RSQ, d) NUSB-RSQ, e) RKSQ, f) Toggling feature with USB-RSQ, and g) A list of simulation scenarios. 77

Figure 6-2 Steady state values of Foxp3 and IL-2 using a) logical scoring, b) arithmetic scoring, and c) the difference between the values using the logical scoring minus the arithmetic scoring. 80

Figure 6-3 Steady state values of Foxp3 using a) the logical scoring, b) the arithmetic scoring with PTEN initialized at 100% and with different number of levels, N, and TCR values. The green (blue) highlighted range represents the percentage of Foxp3+ cells with low (high) antigen dose in the experimental results (Miskov-Zivanov, Turner, Kane, Morel, & Faeder, 2013a). 81

Figure 6-4 Trajectories of Foxp3, IL-2, mTOR, and PTEN using a) the logical scoring, b) the arithmetic scoring with N = 6, different initial levels of TCR, and PTEN is initialized at 100%..... 83

Figure 6-5 a) A network diagram of the upstream elements of Foxp3 as well as the simulation trajectories of b) PTEN, c) mTOR, d) Foxp3, e) SMAD3, f) NFAT, and g) STAT5 from the original model, using logical and arithmetic scoring with N = 3, PTEN initialized at different levels, and TCR is set to 50% (low dose). 85

Figure 6-6 Percentage of attractors with antigen dose removal at time steps T1:T5 as a) in the original model, b) using the logical scoring, c) using the arithmetic scoring, d) attractors frequencies when the antigen dose is removed at T3. 87

Figure 6-7 A subnetwork of the extended naive T cell differentiation model before (left) and after (right) assigning propagation and regulation delays. 90

Figure 6-8 Foxp3 trajectories under the simulation scenarios shown in Figure 6-1(g) usnig the Random-Delay SMLN update scheme for the extended model. 92

Figure 6-9 A comparison between Foxp3 trajectories obtained by different models with the experimental results under low Ag dose. 93

Figure 6-10 A comparison between the simulation time of running the extended model using the Buffer Insertion (BI) method and the Propagation Delay method (PD) when a different amount of delay steps (x-axis) are added to different percentages of model elements and interactions. 95

Figure 6-11 Change in naïve T cell model size in terms of a) number of variables b) number of interactions at different amounts of delay steps that are added to different percentages of model elements and interactions. 95

Figure 6-12 Foxp3 trajectories when the extended model is simulated with the manually defined delays and the delays found by the optimization algorithm compared to the experimental data..... 97

Figure 6-13 A network diagram of the buddying yeast cell cycle model in (Irons, 2009). ... 99

Figure 6-14 Cyclic attractors of the budding yeast cell cycel model. a) Attractors obtained with the No-Reset delay method, b) attractors obtained with the Reset delays method. Each

row represents an attractor state where white/black boxes represent an active/inactive state of each model element. 102

Figure 6-15 A comparison between the simulation time of the budding yeast cell cycle model using the Dummy Nodes (DN) method and the Regulation Delay method (RD) when a different amount of delay steps (x-axis) are added to different percentages of model elements. 103

Figure 6-16 Change in the budding yeast model size in terms of a) number of variables b) number of interactions at different amounts of delay steps that are added to different percentages of model elements and interactions. 104

Figure 6-17 Network diagram of the food security in South Sudan model. Nodes with many interactions are highlighted with different colors for better visualization. 107

Figure 6-18 Simulation trajectories of the four scenarios (left column) and historical data (right column) for a) crop yield, b) conflict, c) inflation, and d) diseases. 109

Figure 6-19 Simulation trajectories for a) food security and b) refugees population. 111

Figure 6-20 Average simulation trajectories of a) crop yield, b) conflict, c) inflation, and d) diseases with manual and optimized delays. Harvest season is highlighted with blue boxes in the crop yield plot. 115

Figure 6-21 average simulation trajectories of a) crop yield, b) conflict, c) food security, d) inflation, e) diseases, and f) inflation using the Random-Delay SMLN scheme with the optimized delays under the four simulation scenarios described in section 6.3.2. 117

Acknowledgement

I would like to express my sincere thanks and gratefulness to my advisor, Prof. Natasa Miskov-Zivanov for her endless support and continuous guidance. I wouldn't have been the person who I'm today without her invaluable advice and help. I really enjoyed being her student and had a great time working with her.

I'd also like to extend my gratitude to Dr. Cheryl Telmer who has been helping me since my first day at Pitt. I wouldn't have reached this far without her support and guidance. I want also to thank all my lab members for their valuable feedback on my work throughout my PhD which allowed me to write a better dissertation.

I would like to express my deepest appreciation and sincere thanks to my committee members, Prof. James Faeder, Prof. Zhi-Hong Mao, and Prof. Samuel Dickerson, for their great feedback and valuable comments. Also, I'm really thankful and grateful to Prof. Amro El-Jaroudi and Prof. Mahmoud El-Nokali for their support and encouragement and for the valuable discussions throughout my journey at Pitt.

I would like also to express my gratitude to all my family and friends who have always been there for me. I'd like to thank Aya Fawzy for having faith in me and for supporting me during my difficult times. I'd also like to thank Khalid Gomaa, Hossam El-Dien, Ali Abdullah, Yassin Khalifa, Amr Mahmoud, Mohamed Bayoumy, Rashad Eletreby, Yasmine Ahmed, Mona Ramadan, Adam Butchy, Busra Susam, and Faezeh Movahedi for their support and encouragement.

To my beloved mother, brothers, sisters and to the memory of my father

1.0 Introduction

1.1 Motivation

Studying complex systems is enhanced by building computational models that improve our understanding of the system and guide researchers to conduct more focused experiments or to collect more relative data. Building accurate models of complex systems requires not only knowledge about the key elements in the system and their regulatory sets, but also the relative timing of events as well as the different timescales of the different processes. For example, timescales of intra- and inter-cellular events in biological systems range from fractions of seconds to hours or even days. For instance, protein-protein interactions, signaling events such as phosphorylation reactions, and the physical movements of signaling molecules can take milliseconds while DNA replication and mRNA degradation can last for hours.

Recent efforts to automate the extraction of information about complex systems events, followed by fast assembly of reliable and useful models, have shown that the available information is often incomplete, or specific interaction mechanisms are unknown. While timing information can be captured within reaction rate constants, and used in differential equations when modeling biochemical networks, often these constants are unknown and difficult to measure. Therefore, to overcome the problems of missing or uncertain information, researchers have applied more abstract or coarse-grained approaches, such as logical and discrete modeling. These modeling approaches include time representations varying from discrete to continuous, and utilize analysis methods such as formal methods, and deterministic or stochastic simulations. To consider different timescales of events, varying nature of component interactions (also called motifs), and the

stochasticity in event occurrence time, a comprehensive representation and analysis framework is necessary. Such a framework will allow for capturing key, or available, system attributes in a standardized manner and allow for rapid automated model design and exploration.

1.2 Scope

Creating hypotheses through studying and analyzing available biological data has become the focus of the computational and systems biology realm, where mathematical and engineering principles are utilized to model and analyze biological interactions. Modeling biological systems can help biologists predict future states of the system under different perturbations, thus leading to insightful hypotheses, as well as guiding experimental designs (Epstein, 2008; Singh & Dhar, 2015). In the last few decades, scientists have introduced several modeling methodologies to reveal the underlying rules that control a biological system at different scales according to the available data. For instance, biochemical reactions can be modeled by a set of ordinary differential equations (ODEs) that can be used to describe the rate of production or consumption of metabolites when the kinetic parameters are available (Materi & Wishart, 2007; Schmidt & Jirstrand, 2005). Additionally, considering the cell as a system of mixed molecules allows for modeling the interactions between these different molecules using reaction rule-based modeling which is based on Gillespie's algorithm (Gillespie et al., 1998). In reaction rule-based modeling, a set of human-readable rules is written to describe reactive motifs in a particular model, such as phosphorylation and binding events, instead of listing all possible molecules and their corresponding reactions (Harris et al., 2016). Existing tools can quickly translate these rules into ODEs, which can be solved and simulated to create dynamic trajectories showing model element changes over time.

Although reaction rule-based modeling introduces a simple way for creating biological models, it still requires quantitative data, such as reaction rate constants and other kinetic parameters in order to correctly simulate the model (Blinov, Faeder, Goldstein, & Hlavacek, 2004). For large systems, it is usually difficult to find all kinetic parameters for all reactive motifs, and hence, qualitative modeling approaches such as logical modeling are introduced in order to model biological events using a more abstract yet powerful approach (Kauffman, 1969; Thomas, 1973). In logical models, each model element is represented by a Boolean variable which can take one of two values: 1 or 0, representing on/off states or high/low concentrations or activity levels, respectively. Additionally, a logic update rule which is formed usually using primitive logic operators such as AND, OR, and NOT is assigned to each variable to specify the relationship between the model element and its regulators. The variables and their update rules constitute an *executable model* that can be simulated (Wang, Saadatpour, & Albert, 2012).

Despite its apparent simplicity, logical modeling can encompass positive and negative feed-forward and feedback loops, which makes it an efficient tool for capturing complex dynamics of intra- and inter-cellular networks such as oscillatory behaviors and multi-stability (R. Albert & Robeva, 2015). Additionally, in the absence of quantitative parameters, the information to construct logical models can be automatically extracted from qualitative textual descriptions utilizing natural language processing (NLP) methods (Sayed, Bocan, & Miskov-Zivanov, 2018; Sayed, Telmer, Butchy, & Miskov-Zivanov, 2017). However, restricting model elements to have only two values (i.e., 0 or 1) limits modelers from studying cases where multiple drug doses or multiple protein activity levels are needed to better represent the biological system (Miskov-Zivanov, Turner, Kane, Morel, & Faeder, 2013a). Therefore, techniques have been developed to model multiple activity levels, while preserving the advantages of abstraction in qualitative models

(Aldridge, Saez-Rodriguez, Muhlich, Sorger, & Lauffenburger, 2009; Gan & Albert, 2018; Miskov-Zivanov, Marculescu, & Faeder, 2013; Morris, Saez-Rodriguez, Clarke, Sorger, & Lauffenburger, 2011; Schaub, Henzinger, & Fisher, 2007). In (Aldridge et al., 2009) and (Morris et al., 2011), authors introduced fuzzy logic and constrained fuzzy logic as methods for mapping quantitative biological data into multi-valued variables that can be used to build a logical network. However, the lack of quantitative data in some cases limits the application of their methods to model large systems. Miskov-Zivanov et al. (Miskov-Zivanov, Marculescu, et al., 2013) and Gen et. al. (Gan & Albert, 2018) modeled multi-valued elements by creating multiple Boolean variables corresponding to the discrete value that each element can take. Following this approach, Sayed et. al. (Khaled Sayed, Yu-Hsin Kuo, Anuva Kulkarni, & Natasa Miskov-Zivanov, 2017a) introduced DiSH-simulator (Discrete, Stochastic, Heterogeneous model simulator), which implements discrete multi-valued elements as multiple Boolean variables.

Although the aforementioned methods allow representation of multiple qualitative levels for each variable, they increase the size and complexity of models due to the creation of multiple Boolean variables to represent each element. Representing discrete values using multiple Boolean variables requires defining logic rules in terms of the individual Boolean variables, which increases the size and complexity of the executable model, and increases the difficulty of constructing and interpreting the logic rules. For example, if an element has n regulators and each one of them is modeled by m discrete values, then one must create a truth table of size m^{n+1} in order to represent the output for each Boolean variable (Miskov-Zivanov, Marculescu, et al., 2013). Although logic minimization techniques such as Quine-McCluskey algorithm (McCluskey, 1956; Quine, 1955) can be used to reduce update rules, the complexity of creating an executable model still increases with more discrete levels and the number of element regulators. To reduce the model complexity,

Schaub et. al. (Schaub et al., 2007) suggested using discrete variables and applying mathematical operations instead of constructing logic rules as functions of Boolean variables. In the methodology introduced by (Schaub et al., 2007), the next value of the regulated element is determined by comparing the score of positive regulators to the score of negative regulators. The positive and negative scores are calculated by adding up the activity level of each regulator after multiplying it by a weight of 1 (in case of activation) or -1 (in case of inhibition). Consequently, the value of the regulated element goes up by one discrete level if the positive score is greater than the negative score, and vice versa. The regulator's value stays at its current state if the positive and negative scores are equal. Although this method reduces the model complexity by avoiding the step of creating several Boolean variables, it does not provide a way for modeling logical operations such as AND and OR which might be needed to model different biological motifs such as receptor activation (Sayed, Telmer, & Miskov-Zivanov, 2016) and complex formation in addition to using only two values for weights (i.e. 1 or -1).

The weights used by Schaub et. al. (Schaub et al., 2007) were introduced to differentiate between the activators and inhibitors of the regulated element. However, intra- and inter-cellular networks and biological systems in general are complex and some regulators might have a different impact on the regulated element (Miskov-Zivanov, Turner, Kane, Morel, & Faeder, 2013a). Therefore, assigning different weights to different interactions might be useful in capturing accurate dynamics. Therefore, we expand the capabilities of DiSH-simulator to include three main functionalities: (1) defining logic rules in terms of discrete variables, (2) enabling the assignment of weights on different biological interactions in the discrete logic rules, and (3) modeling different timescales of complex interactions. The added functionalities to DiSH-simulator provide a way for creating hybrid models utilizing different update functions and allow for automating model

design and analysis especially for large and complex systems that require abstract representations in order to avoid the problem of missing and uncertain information. We also apply the developed tools in this dissertation into other domains such as socio-economic systems to evaluate their effectiveness in building accurate models of complex systems.

1.3 Contribution

The goal of this research is to develop methods for building and simulating accurate models of dynamic networks through developing standardized representations for common network motifs, as well as incorporating timing information into discrete models of complex systems. The following points summarize the main contributions of this work:

- Development and implementation of a simulation framework that supports models at several different abstraction levels, incorporates timing information where available, and relies on a standardized notation for representing network motifs.
- Development and implementation of formal methods for incorporating timing information in discrete models, to capture varying timescales of different events.
- Development of an optimization heuristic to match the *in-silico* simulation timescales with the timescales of the available data.
- Application and evaluation of the developed methods using several existing computational models and experimental results.

1.4 Dissertation Statement

Standardized representation is essential for automating modeling process, while incorporating timing information within this representation allows for designing accurate and reliable models; the work on developing methods and tools that rely on such representation, and additionally, account for missing or uncertain information in the modeled system, will significantly speed up the modeling of complex and complicated systems and advance our understanding and explanation of these systems.

1.5 Dissertation Organization

The dissertation includes 7 chapters. Chapter 2 provides background on logical modeling and a review of the current literature on different time representations in discrete models. Chapter 3 introduces our DiSH simulator with a description of the different simulation schemes as well as our methods for simulating models with discrete variables. Chapter 4 presents our model representation format which shows how different network motifs are represented and simulated. Chapter 5 introduces the developed methods for incorporating timing information into discrete models of intra- and inter-cellular networks and discusses how the proposed methods can be applied on various biological motifs and how the simulation trajectories are affected by the utilized simulation scheme. In chapter 6, we show and discuss the results obtained when the introduced methods are applied to three complex systems models which are naïve T cell differentiation, buddying yeast cell cycle, and food security in South Sudan. Chapter 7 concludes the dissertation and lists the future work.

2.0 Background

2.1 Logical Modeling

Complex networks can be represented by a directed graph of interacting components $G(V, E)$ where $v_i \in V$ represents a node, while $e_{ij} \in E$ represents an edge from node v_i to node v_j (Wang, Saadatpour, & Albert, 2012). In biology, nodes in graph G represent elements in the biological network such as proteins, genes, mRNA, and/or cellular processes while edges represent regulatory interactions (e.g., activation or inhibition) among elements. A logical model $B(X, F)$ is defined by a set of Boolean variables X and their corresponding update functions F. Each variable $x_i \in X$ represents a model element and can take one of two values, 0 for inactive and 1 for active (i.e. $x_i \in \{0,1\}$). Additionally, an update function $f_i \in F$ is assigned to each model variable to determine its next-state value based on the current-state values of the element's regulators $x_{j=1,\dots,k_i}$, where k_i is the total number of regulators of an element v_i that is modeled by x_i. The initial values of the set of variables X and their update functions F form an executable model that can be simulated in order to study the evolution of the model elements over time. The update functions are created using primitive logic operators such as AND, OR, and NOT, which specify the regulation logic, and hence, f_i can be evaluated into either 0 or 1 (i.e. f_i: $\{0,1\}^{k_i} \rightarrow \{0,1\}$). Figure 2-1 shows a toy example of three interacting elements and their regulatory sets where variables x_2 and x_3 have to be active in order to switch the value of variable x_1 from 0 to 1. In contrast, either x_1 or x_3 can switch the value of x_3 from 0 to 1 when it is active.

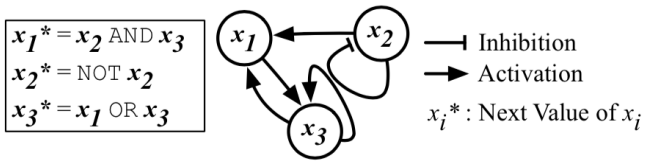


Figure 2-1 A toy example of three interacting elements and the corresponding Boolean update functions.

Although complex systems contain many components, usually only a few of them are used as markers of particular behaviors of the system under different scenarios. These elements are referred to as elements of interest (EOI), and their values can determine the state of the system at each simulation step t. The total number of states of a complex system is calculated as 2^p where p is the number of the EOI. Therefore, constructing and analyzing a Boolean model for a complex system network require 6 steps; 1) listing all system components and their influence sets (i.e. activators and inhibitors), 2) assigning a Boolean variable x_i for each model element, 3) defining an update function f_i for each variable to determine its value at simulation step t+1, 4) specifying different simulation scenarios and assigning initial values for all variables accordingly, 5) identifying the EOI that will determine the state of the system at each simulation step, and 6) running simulations to create trajectories for each model element showing how its value changes over time.

2.2 Timing in Logical Models

Logical and discrete models are categorized into two main groups according to the way the update functions are evaluated over time. The first category can be realized as *clocked logic circuits* where the update functions are evaluated at discrete time steps either simultaneously

(synchronous) or sequentially (asynchronous) (Kauffman, 1993; Thomas & d'Ari, 1990) while the second category can be described as *event-triggered models* where an update function is evaluated only if a change happens in the value of one of the element's regulators and hence the updates do not explicitly depend on time (*unclocked logic circuits*). The first category is referred to as discrete-time logical models while the second category is considered as continuous-time logical models (Dee & Ghil, 1984; Ghil & Mullhaupt, 1985).

In the following subsections, we use the small network in Figure 2.2(a) as an example of a few network interactions and show how the current methods model the different timescales by introducing delays as well as utilizing different simulation schemes. Also, Figure 2.2(b) shows an equivalent logic circuit for the example in Figure 2.2(a) where x_1 and x_2 are activators of x_4 while x_3 is an inhibitor.

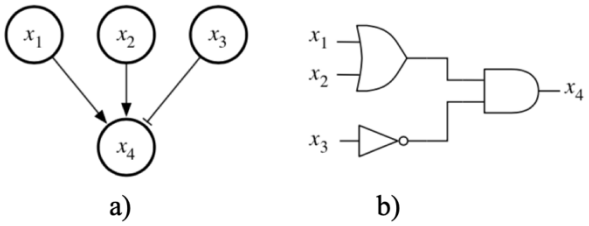


Figure 2-2 a) A toy example of four interacting components, b) the equivalent logic circuit.

2.2.1 Buffer Insertion

Buffer insertion is an intuitive method for adding delays to an interaction. In the buffer insertion method, modelers insert extra nodes between the regulator and the regulated element as shown in Figure 2-3(a) (Miskov-Zivanov, Turner, Kane, Morel, & Faeder, 2013a). The equivalent

logic circuit is shown in Figure 2-3(b) where a buffer is added between each input signal and the next logic gate in addition to a clock signal that triggers the updates. Therefore, any change in the regulators' values will be delayed by the specified amount of delay steps. The update rule for the regulated element x_4 has to be written in terms of the delay variables such as

$$x_4 = (d_1\ OR\ d_2)\ AND\ (NOT\ d_3) \tag{2-1}$$

Where $d_1 = x_1$, $d_2 = x_2$, and $d_3 = x_3$. Figure 2-3(c) shows a timing diagram for the example in Figure 2-3(a) under arbitrary changes in the values of the regulators (i.e. x_1, x_2, x_3) assuming that $d_1 = 1$, $d_2 = 1$, and $d_3 = 2$ steps. The timing diagram in Figure 2-3(c) shows how x_4 changes over time under the simultaneous update scheme where all update rules are evaluated at the same time. If the random sequential update scheme (only one update rule is evaluated at each simulation

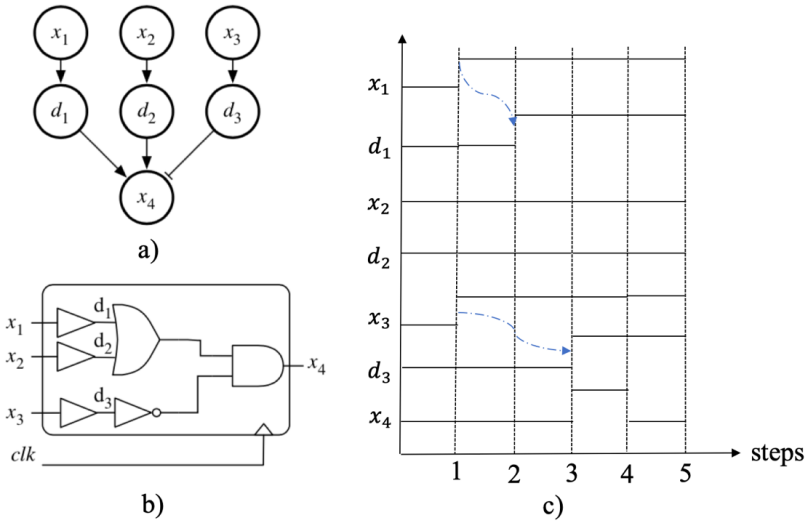


Figure 2-3 The representation of the toy example in Figure 2-2 using the Buffer Insertion method. (a) The toy example with the added delay nodes, (b) the equivalent logic circuit, and (c) the timing diagram of x_4 using the simultaneous update scheme.

step) is utilized, the actual number of delay steps may vary because the update rules of the delay nodes might not be selected for evaluation at some simulation steps.

2.2.2 "Dummy" Nodes

The method of creating "dummy" nodes is first introduced by Iron (Irons, 2009), in order to add delays to the updates of specific elements. In (Irons, 2009), the delays are added after evaluating the update function instead of inserting delay nodes between each regulator and the regulated element as in the buffer insertion method. Figure 2-4(a) shows how the dummy nodes are inserted into the toy example of Figure 2-2 where f_A is the activation, and f_d is the degradation functions. The degradation process is defined as the removal of the activation signal which is different from the inhibition signal that is determined by x_3. Figure 2-4(b) shows the equivalent logic circuit assuming that the activation delay is 2 steps, the degradation delay is 3 steps, and with no inhibition delays (i.e. the inhibition signal from x_3). The timing diagram in Figure 2-4(c) shows the time course of x_4 using the simultaneous update scheme according to the logic update rules given by Equations (2-2) through (2-7) that describe the relationships between the nodes in Figure 2-4(a).

$$f_A = x_1 \; \boldsymbol{OR} \; x_2 \tag{2-2}$$

$$d_1 = f_A \tag{2-3}$$

$$d_2 = d_1 \tag{2-4}$$

$$f_{d1} = f_A \; \boldsymbol{AND} \; d_1 \tag{2-5}$$

$$f_{d2} = \; f_{d1} \; \boldsymbol{AND} \; (f_A \boldsymbol{OR} \; d_1 \boldsymbol{OR} \; d_2) \tag{2-6}$$

$$x_4 = \; (f_{d1} \; \boldsymbol{OR} \; f_{d2} \;) \; \boldsymbol{AND} \; (\boldsymbol{NOT} \;\; x_3) \tag{2-7}$$

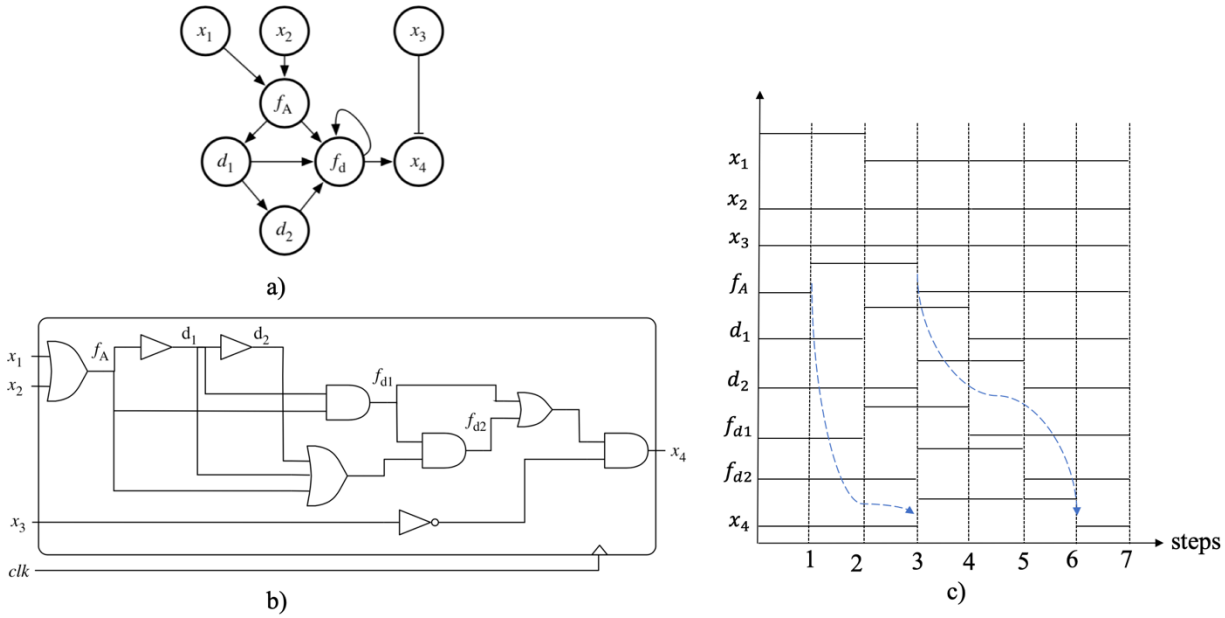


Figure 2-4 The representation of the toy example in Figure 2-2 using the dummy nodes method. (a) The toy example with the added delay nodes, (b) the equivalent logic circuit, and (c) the timing diagram of x_4 using the simultanoues update scheme.

2.2.3 Boolean Delay Equations (BDEs)

Boolean delay equations are used to build *event-triggered models* where the update functions are only evaluated when a regulation signal is received by the regulated element and not evaluated at specific simulation steps (Dee & Ghil, 1984; Ghil & Mullhaupt, 1985). Event-

triggered models are continuous-time models where a delay $\tau_{ij} \in \mathbb{R}$ is assigned to the interaction between elements v_i and v_j defining the amount of time needed for a regulation signal from a regulator, v_i, to reach a regulated element, v_j (Sevim, Gong, & Socolar, 2010). The change in the value of an upstream element (e.g. v_i) triggers the re-evaluation of the update functions of all downstream elements (e.g. v_j) after the specified time delay (i.e. τ_{ij}) has elapsed. Formally, the value of a model variable x_j at time t is determined by equation (2-8) as:

$$x_j(t) = f_j\left(x_1\left(t-\tau_{1j}\right), \ldots, x_k\left(t-\tau_{kj}\right)\right), \quad j = 1, 2, \ldots, n \tag{2-8}$$

where *n* is the total number of elements in the model, *k* is the total number of regulators of element v_j, and f_j is the update function which is defined by modelers using logical operators.

Figure 2-5(a) shows the representation of the toy example in Figure 2-2 after assigning delays to each interaction. It is also shown in Figure 2-5(b) that there is no need for a clock signal to trigger evaluating an update function. Additionally, the timing diagram in Figure 2-5(c) shows the evolution of the x_4 according to the logic update rule that is defined as:

$$x_4(t) = [x_1(t-\tau_{14}) \ \boldsymbol{OR} \ x_2(t-\tau_{24})] \ \boldsymbol{AND} \ [\boldsymbol{NOT} \ x_3(t-\tau_{34})] \tag{2-9}$$

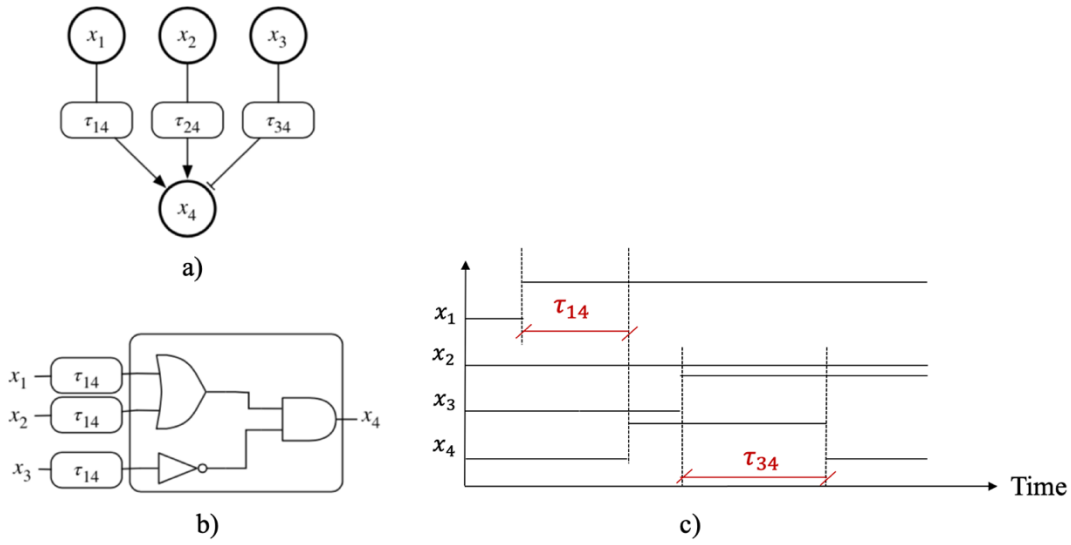


Figure 2-5 The representation of the toy example in Figure 2-2 using the BDEs. (a) The toy example with delays, (b) the equivalent logic circuit without a clock signal, and (c) the timing diagram of x_4 over a continuous time period.

2.2.4 Piecewise Linear Differential Equations (PWLDEs)

Piecewise linear differential equations are used to build continuous-time continuous-state models where each element is represented by two variables; a continuous variable, X_i, and a Boolean variable, x_i. The value of the continuous variable X_i is determined by the differential equation shown by Equation (2-10) as:

$$\frac{dX_i}{dt} = \lambda_i \, f_i(x_{i1}, x_{i2}, \ \ldots, x_{ik}) - \gamma_i X_i \tag{2-10}$$

Where f_i: $\{0,1\}^k \rightarrow \{0,1\}$ is a Boolean function of the regulators of x_i and it is determined by the modeler, k is the total number of regulators, $i = 1,2, \ \ldots, \ n$ where n is the total number of elements in the model, λ_i is production constant, and γ_i is the degradation rate.

The analytical solution of Equation (2-10) is given as:

$$X_i(t) = X_i(t_0)\, e^{-\gamma_i(t-t_0)} + \frac{\lambda_i}{\gamma_i} f_i(x_{i1}, x_{i2}, \ \ldots, x_{ik})(1 - e^{-\gamma_i(t-t_0)}) \tag{2-11}$$

Where t_0 is the time when a change happens in the value of f_i. The Boolean states of each continuous variable are calculated as:

$$x_i = \begin{cases} 1 & if \ \ X_i \geq 0.5 \\ 0 & if \ \ X_i < 0.5 \end{cases} \tag{2-12}$$

with a production delay, $T_{i,\,1/2} = \ln(\frac{2\lambda_i}{2\lambda_i - \gamma_i})$ when $x_i(t_0) = 0$ and $f_i = 1$ and a degradation delay, $T_{i,\,1/2} = \frac{1}{\gamma_i}\ln(2)$ when $x_i(t_0) = 1$ and $f_i = 0$. The models built using the piecewise linear differential equations are event-triggered models and hence, any change in the regulators' values triggers the re-evaluation of the update function f_i. Figure 2-6(a) shows the representation of the toy example in Figure 2-2 using piecewise linear differential equations while Figure 2-6(b) shows the analogy of the toy example in terms of a logic circuit. Figure 2-6(c) shows the relationships between the update function f_4 as well as the continuous and discrete variables X_4 and x_4. Additionally, Figure 2-7 shows the timing diagram of x_4 with random changes in the regulators' values according to the update function, f_4 that is defined as:

$$f_4(x_1, x_2, x_3) = (x_1 \ \boldsymbol{OR} \ x_2) \ \boldsymbol{AND} \ (\boldsymbol{NOT} \ x_3) \tag{2-13}$$

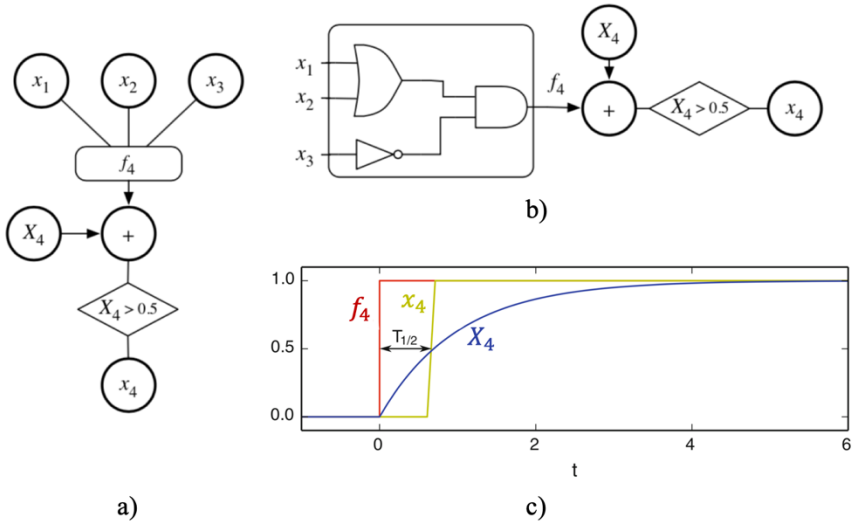


Figure 2-6 The representation of the toy example in Figure 2-2 using the Piecewise linear differential equations. (a) The modified toy example, (b) the equivalent logic circuit without a clock signal, and (c) the changes of X_4 and x_4 when f_4 is updated.

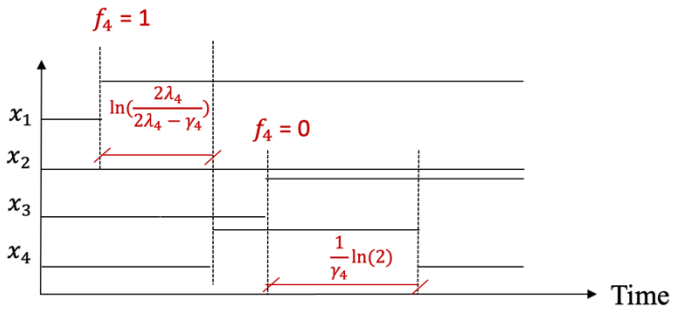


Figure 2-7 The timing diagram of x_4 in Figure 2-6(a) under random changes in the regulators' values.

2.2.5 Overview of the Current Methods

Table 2-1 shows a comparison between the available methods for incorporating timing information into logical models. The table shows that each method can add delays to either delay the signal or to delay the regulation process but not both. Also, the added delays in all methods do not depend on the values of the regulators or the regulated element and hence it is not possible to assign different delays under various regulation conditions (i.e. different combinations of the current values of the regulators and the regulated element)

Table 2-1 A comparison between the available time abstraction methods.

	Buffer Insertion	Dummy Nodes	BDEs	PWLDEs
States, x	Discrete	Discrete	Discrete	Discrete/Continuous
Time, t	Discrete	Discrete	Continuous	Continuous
Update orders	Simultaneous/ Sequential	Simultaneous/ Sequential	Event-triggered	Event-triggered
Type of Delays	Signal delays	Regulation delays	Signal delays	Regulation delays
Conditions for Delays	Delays do not depend on the value of the regulator or the regulated element.			

3.0 DiSH Simulator[1]

In this chapter, we present our simulator, DiSH (Discrete, Stochastic, Heterogeneous model simulation), a stochastic simulator for discrete models with heterogeneous elements and interactions. DiSH implements multiple simulation schemes, and allows for fast simulation of discrete models. Compared to other existing simulators (I. Albert, Thakar, Li, Zhang, & Albert, 2008; Bock, Scharp, Talnikar, & Klipp, 2014; Danos, Feret, Fontana, Harmer, & Krivine, 2008; Faeder, Blinov, & Hlavacek, 2009; Gonzalez, Naldi, Sanchez, Thieffry, & Chaouiya, 2006; Helikar & Rogers, 2009; Müssel, Hopfensitz, & Kestler, 2010; Yu, Tung, Park, Lim, & Yoo, 2012; Zheng et al., 2010), our contribution with DiSH is two-fold: (1) We developed a representation that allows for multi-valued hierarchical models to be translated into logic circuit-like models and used by our simulator; (2) We incorporated rates of biological processes by associating probabilities with element update rules.

3.1 Simulation Schemes

In this section, we describe model execution schemes that our simulator supports (also shown in Figure 3-1(b)). Two main categories of simulation schemes are supported by DiSH:

[1] Based on Khaled Sayed, Yu-Hsin Kuo, Anuva Kulkarni, and Natasa Miskov-Zivanov. "DiSH simulator: Capturing dynamics of cellular signaling with heterogeneous knowledge." In Proceedings of the 2017 Winter Simulation Conference, p. 64. © [2017] IEEE Press.

Simultaneous (SMLN), also sometimes referred to as synchronous, and Sequential (SQ), also referred to as asynchronous scheme. As shown in Figure 3-1(b), in the SQ scheme, elements can be updated using either Ranked-order (RKSQ) or Random-order (RSQ) scheme. In addition, the selection of an element to be updated next in the RSQ scheme can be done in two ways: Round-

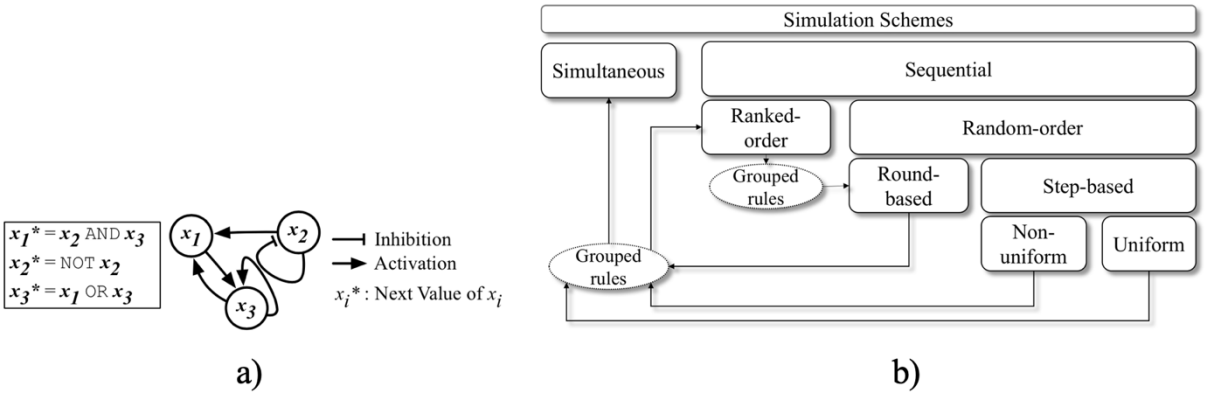


Figure 3-1 a) A toy example: three nodes (A, B and C), and their update rules specified. b) Simulation schemes.

based (RB-RSQ) selection and Step-based (SB-RSQ) selection, as described in the subsequent sections. The probability of selecting a rule to be updated in the SB-RSQ scheme can be Uniform (USB-RSQ), that is, the same probability value is assigned to each logic rule, or Non-Uniform (NUSB-RSQ), where the assigned probability is different for each rule.

Formally, the following holds for all the simulation schemes that we use. Let £ be the set of all possible states of the system, then we can compute the size of £ as: $|£| = N_1 \cdot N_2 \cdot \ldots \cdot N_n$ where N_i is the number of discrete levels assigned to element v_i and n is the total number of elements in the model. In general, the probability $p(\boldsymbol{S^{t+1}} = \boldsymbol{S}_m)$ that, at time step $t+1$, the state $\boldsymbol{S}$ of the overall model is equal $\boldsymbol{S}_m$, where m is an integer between 1 and |£|, can be computed as:

$$p(\mathbf{S}^{t+1} = \mathbf{S}_m) = \sum_{l=1}^{|\mathcal{E}|} p(\mathbf{S}^{t+1} = \mathbf{S}_k | \mathbf{S}^t = \mathbf{S}_l) \cdot p(\mathbf{S}^t = \boldsymbol{S}_l) \tag{3-1}$$

Furthermore, the probability that element v_i will take value V_j where $j = 0, .., N_i - 1$ in time step $t + 1$ during simulation, $p(\mathbf{E}_i^{t+1} = \mathrm{V}_j)$, is then given by the following equation:

$$p(\mathbf{E}_i^{t+1} = \mathrm{V}_j) = P_{v_i} \cdot p(f_{v_i}^t = \mathrm{V}_j) + (1 - P_{v_i}) \cdot p(\mathbf{E}_i^t = \mathrm{V}_j) \tag{3-2}$$

where P_{v_i} represents the probability that element v_i is selected to be updated next, and $f_{v_i}^t$ is the update function for element v_i at time step t during simulation. Then, $p(f_{v_i}^t = \mathrm{V}_j)$ can be computed as:

$$p(f_{v_i}^t = \mathrm{V}_j) = \sum_{l=1}^{|\mathcal{E}|} p(f_{v_i}^t = \mathrm{V}_j | \mathbf{S}^t = \mathbf{S}_l) \cdot p(\mathbf{S}^t = \mathbf{S}_l) \tag{3-3}$$

Therefore, one can then compute the probability of state $\boldsymbol{S^{t+1}}$ being equal to state $\boldsymbol{S}_m = (\mathrm{V}_{m,1}, \mathrm{V}_{m,2}, \dots, \mathrm{V}_{m,n})$, where $\mathrm{V}_{m,i}$ is the value of element v_i in state $\boldsymbol{S}_m$ as:

$$p(\mathbf{S}^{t+1} = \mathbf{S}_m) = \prod_{i=1}^{n} p(\mathbf{E}_i^{t+1} = \mathrm{V}_{m,i}) \tag{3-4}$$

3.1.1 Simultaneous (SMLN) Scheme

In the SMLN scheme, all elements are updated simultaneously, that is, current state values of all variables are used to simultaneously compute next state values. SMLN scheme is therefore

deterministic: for each state, there is only one possible next state. In SMLN, if an initial state of the system is given, one can determine the steady state or steady cycle that the system will reach. In other words, the probability P_{v_i} in Equation (3-2) is equal 1 for all elements v_i, and in each given time step t, the probabilities $p(\mathbf{S}^t = \mathbf{S}_l)$ will be equal 0 for all but one value l=L, for which it will be $p(\mathbf{S}^t = \mathbf{S}_\mathrm{L}) = 1$. A state transition graph (STG) resulting from the simulation of our toy example from Figure 3-1(a), using the SMLN scheme, is depicted in Figure 3-2. Given elements of the toy model, x_1, x_2, x_3, and their corresponding values, V_1, V_2, V_3, if the simulation starts at time $t = 0$ we denote this as state $\boldsymbol{S^0} = (V_1^0, V_2^0, V_3^0)$. The probability of the following states

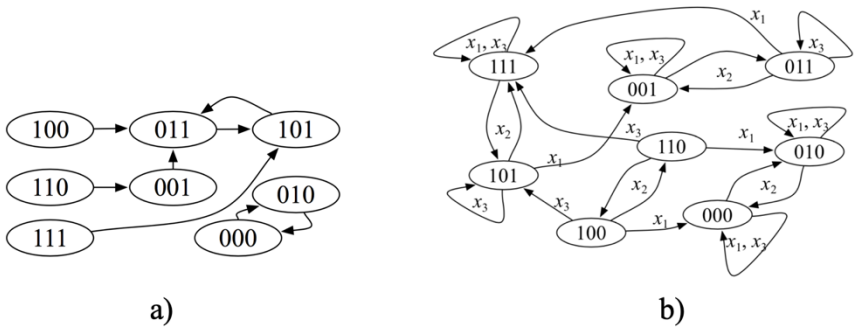


Figure 3-2 STG for the toy example in Figure 3-1(a), for (a) the SMLN scheme and (b) the RSQ scheme. Labels on graph edges indicate which elements are selected for update.

$\mathbf{S}^{t+1} = \mathbf{S}_m$ for time steps $t = 0, \ldots, T$ where T is the total number of simulation steps, will in each time step be equal 1 for one specific $m = M_t$, and will be 0 for all other $m \neq M_t$. For example, in our toy system, as shown in Figure 3-2(a), if the simulation starts from any state except states 000 and 010, we will always reach state 101, and the model will then oscillate between states 011 and 101. In addition, if the simulation starts at state 000 or 010, it keeps oscillating between these two states and never moves to any other state.

3.1.2 Random-order Sequential (RSQ) Scheme

In the RSQ scheme, model variables are not updated simultaneously, instead they are updated sequentially and in random order. In other words, once element v_i is updated in time step t by computing its new value according to its update function f_{v_i}, this new value is used to determine model element values in the following time steps until the same element v_i is selected for update again. This scheme allows for modeling cellular signaling and processes in a more realistic way than in the SMLN scheme, as it accounts for the randomness that exists in the timing of biological events. The STG of our toy example system, when simulated using the RSQ scheme, is shown in Figure 3-2(b). As shown in the STG, a state can have multiple next states, and thus, a given initial state can be followed by multiple different paths through STG and result in different steady-states. This means that the probability P_{v_i} in Equation 3-2 does not have to be equal 1 for a given element v_i at a given time step t, as it was the case in the SMLN scheme. Instead, this probability will depend on the method used for selecting elements for update, as described in the following subsection. Furthermore, in each given time step t, the probabilities $p(\mathbf{S}^t = \mathbf{S}_l)$ can vary between 0 and 1 for different values of *l*.

3.1.3 Round-Based Random-order Sequential (RB-RSQ) Scheme

The RB-RSQ simulation scheme has been previously described in (I. Albert et al., 2008; S. Li, Assmann, & Albert, 2006) and used in (Miskov-Zivanov, Turner, Kane, Morel, & Faeder, 2013a). It is important to distinguish here between simulation *step* and simulation *round*. While the simulation step accounts for updating value of a single element, and can also correspond to time step in our earlier discussion, the simulation round represents a cycle within which **all**

elements are updated **exactly once** according to their update functions. Formally, if a model has n elements, v_i, $i = 1,..,n$, and their update functions are f_{v_i}, then each round consists of n steps, $ROUND = (STEP_1,\ldots,STEP_n)$, and in each round a new random order is determined, in which the values of these function are computed. Thus, in a given round R, the element update order is a random permutation P_r of the vector $(STEP_1,\ldots,STEP_n)$: $update\_order_R = (STEP_{R_1},\ldots,STEP_{R_n}) = P_r(STEP_1,\ldots,STEP_n)$ such that the step at which element v_i, is updated is $STEP_{R_i}$. Given that every element gets updated exactly once within a round, the probability P_{v_i} that an element v_i is selected for update in a given time step t depends on the $STEP_{R_i}$ within round when this update occurs: $P_{\mathbf{v}_i} = 1/\left(n - \left(STEP_{R_i} - 1\right)\right)$. The two simulation rounds when the RB-RSQ method is applied to our toy example starting at state 100 are shown in Figure 3-3.

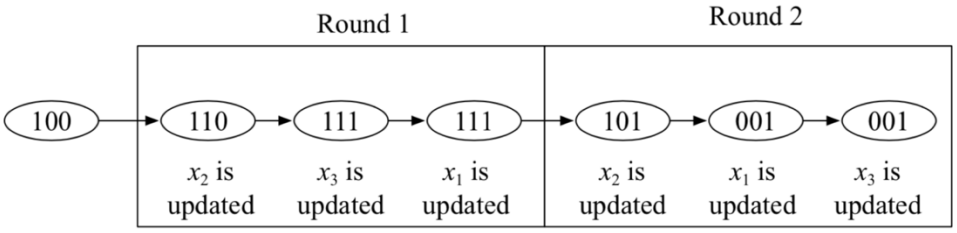


Figure 3-3 STG for the toy example in Figure 3-1(a) using the RB-RSQ scheme.

3.1.4 Step-Based Random-order Sequential (SB-RSQ) Scheme

In the SB-RSQ simulation scheme, one model element is chosen for update in each time step. There are no round-based restrictions for element selection, and therefore, the same element can be updated in consecutive steps. In addition, elements can be chosen for update all at the same rate or at different rates. Thus, we define the following two sub-schemes.

*Uniform (*USB-RSQ*):* In this simulation approach, all elements have the same update rate, and for a model with n elements, in each step the probability for an element v_i to get selected for update is: $P_{v_i} = 1/n$

This approach is used when the time scales of changes in system elements are not well known, and thus the default approach is to assume that the rates at which elements are updated are equal.

*Non-uniform (*NUSB-RandSeq*):* In this simulation approach, each model element v_i has an assigned update rate, r_{v_i}. We assign update rates to elements based on prior knowledge, such that the system evolves over time following the rate of change observed in experiments. For example, since gene transcription and translation occur at a different time scale compared to protein modifications, we assign higher rates to protein interactions and lower rates to gene activation and protein synthesis. In this simulation approach, the probability of element being selected next for update is a function of these update rates: $P_{v_i} = \frac{r_{v_i}}{\sum_{i=1}^{n} r_{v_i}}$ where the probability of selecting an element v_i to update its rule is proportional to the sum of all update rates in the model. We illustrate both types of SB scheme, USB and NUSB, in Figure 3-4(a) and (b).

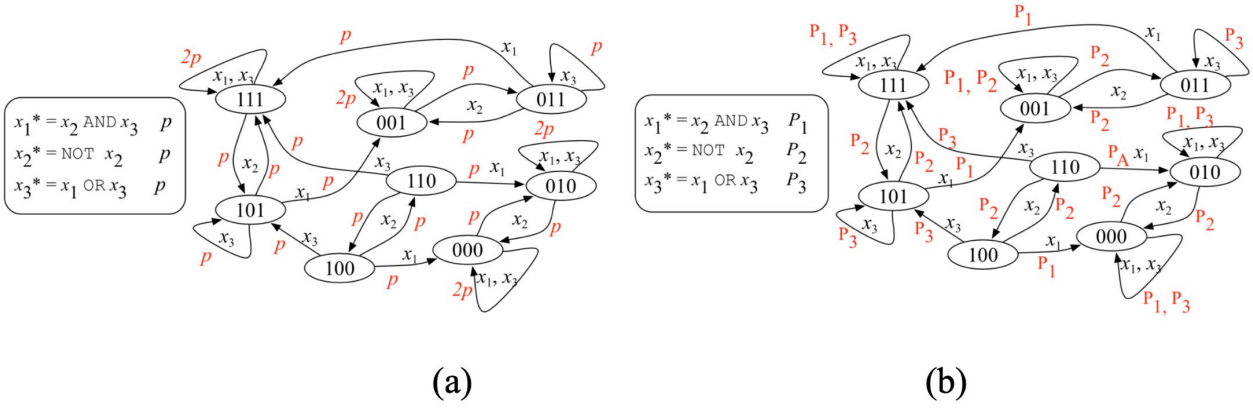


Figure 3-4 Logic rules and STG in the (a) USB-RSQ and (b) NUSB-RSQ scheme.

3.1.5 Ranked-order Sequential (RKSQ) Scheme

Element update rules can also be assigned *rank numbers*. This feature is adopted from the BooleanNet tool developed by (I. Albert et al., 2008). Those rules that have same rank are executed using RB-RSQ scheme. Similarly, groups with different rank are executed according to their rank: all rules in the group with rank 1 are executed first, then all the rules with rank 2 are executed, and so on. As shown in Figure 3-5, x_2 and x_3 should be updated first in random order before we update x_1 which has rank 2.

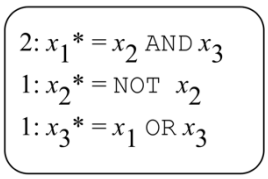


Figure 3-5 An example of ranked rules.

3.2 Additional Functionalities

3.2.1 Grouped rules

Often, we are interested in grouping some variables together such that they are updated either simultaneously, or in the order in which they are listed in the model file. Examples of such situations are (i) all Boolean variables representing the *same model element* v_i are grouped and updated simultaneously; (ii) if there are *different model elements* that need to be updated at the same time, all their corresponding variables will be grouped and updated simultaneously; (iii) if it

is required for a group of different model elements or for a group of model variables corresponding to the same element to be updated in a specific order, but in random order with the other elements or variables in the model, they are grouped and executed sequentially, in the order in which their update rules are listed in the model file. The update rules of all variables that need to be updated together are specified within curled braces '{}' in the model file. In the SMLN scheme, this does not change the execution of rules, since the grouped rules are executed at the same time with the other rules. However, in the SQ scheme, the grouped rules are executed together when the group is selected to be updated at a specific time step. Figures 3-6(a) and (b) show examples where nodes x_1 and x_3 are grouped and ranked (Figure 3-6(a)) or given different update rates (Figure 3-6(b)). The difference in resulting simulated model behavior is illustrated with two state diagrams when the first (initial) state is the same, as shown in Figure 3-5(c).

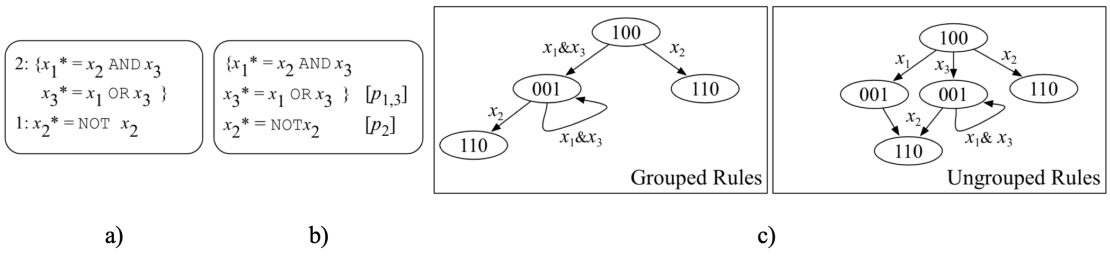


Figure 3-6 (a) Grouped and Ranked rules, (b) Grouped rules with different rates, (c) two small examples of STG when grouped rules are used.

3.2.2 Toggle Implementation

It is possible to toggle the value of a variable (i.e., switch from 1 to 0 and from 0 to 1) at a specific round or step by specifying this in the model file next to the variable initialization. This is often a useful feature of simulations that allows us to closely mimic wet-lab experiments. For

example, toggling the value of the T-cell receptor (TCR) signal in the T-cell model in (Miskov-Zivanov, Turner, Kane, Morel, & Faeder, 2013a) allows for studying the impact of the duration of a high signal on the system's behavior. This functionality should be used with the RSQ simulation schemes.

3.3 Discrete DiSH

In this section, we expand the capabilities of DiSH-simulator to include two main functionalities: (1) defining rules in terms of multi-valued variables, and (2) enabling the assignment of weights to different biological interactions.

Similar to the steps of creating and analyzing Boolean models, listed in Section 2.1, the design of discrete models starts with listing all model elements, where each model element is assigned (a) a discrete variable, (b) an update rule which is a function of the element's activators and inhibitors, (c) a number of allowable discrete levels, *N*. Next, discrete model simulation requires (a) initial values for all model elements for a given simulation scenario, and (b) specifying a simulation scheme. In this work, we extend the DiSH-simulator (Khaled Sayed, Yu-Hsin Kuo, Anuva Kulkarni, & Natasa Miskov-Zivanov, 2017a) to include two approaches for evaluating the update rules of multi-valued variables.

To update the values of model elements, we evaluate the differential strength of their positive and negative regulators (i.e. activators and inhibitors). The strength of the activators and inhibitors can be either calculated by simply using max, min, and *N*'s complement in place of AND, OR, and NOT in logical models (Aldridge et al., 2009; Morris et al., 2011) , or by assigning weights to activating and inhibiting interactions and using addition, subtraction, and multiplication

operators. More formally, let $\mathbb{F}_N = \{0,\ 1,\ \ldots,\ N-1\}$, where N is the maximum number of discrete levels that are assigned to each model element, and $X = \{x_1, x_2,\ \ldots, x_n\}$ be a finite set of the variables that are assigned to model elements. The value of a variable, x_i, represents the discrete level of the element's concentration or activity at each simulation step t such that $x_i(t) \in \mathbb{F}_N$. The next value of a variable, $x_i(t+1)$, is determined by the current value, $x_i(t)$, and the differential strength of activators and inhibitors, $S_i(t)$, or:

$$x_i(t+1) = f(x_i(t), S_i(t)) \tag{3-5}$$

where $S_i(t)$ is calculated by subtracting the regulation score of the set of inhibitors, X_{I_i}, from the regulation score of the set of activators, X_{A_i}, of element x_i at simulation time step t as:

$$S_i(t) = S_{A_i}(t) - S_{I_i}(t) \tag{3-6}$$

$S_{A_i}(t)$ is called *activators score* while $S_{I_i}(t)$ is called *inhibitors score*. The sets of activators and inhibitors of model elements are assigned by the modeler such that:

$$X_{A_i} = \{x_{A_{i,1}}, x_{A_{i,2}},\ \ldots, x_{A_{i,m}}\} \subset X, \text{ and}$$

$$X_{I_i} = \{x_{I_{i,1}}, x_{I_{i,2}},\ \ldots, x_{I_{i,l}}\} \subset X$$

where m and l are the number of activators and number of inhibitors for element x_i respectively.

In order to calculate both $S_{A_i}(t)$ and $S_{I_i}(t)$, we apply a function $g \in \{g_1, g_2\}$ such that $S_{A_i}(t) = g(x_{A_{i,1}}(t), x_{A_{i,2}}(t),\ \ldots, x_{A_{i,m}}(t))$ and $S_{I_i}(t) = g(x_{I_{i,1}}(t), x_{I_{i,2}}(t),\ \ldots, x_{I_{i,l}}(t))$. Function g_1 applies a set of operators $\diamond_1 \in \{min,\ max,\ N's\ complement\}$ while function g_2 applies the set

of operators $\diamond_2 \in \{+, -, *\}$. The score calculated by $g_1: \mathbb{F}_N^k \to \mathbb{F}_N$, for $k \in \{m, l\}$, is called *logical* score because $\diamond_1$ contains operators that are equivalent to the logical operators AND, OR, and NOT while the score calculated by $g_2: \mathbb{F}_N^k \to \mathbb{R}$ is called *arithmetic* score because $\diamond_2$ contains only arithmetic operators. The order of applying an operator $\diamond \in \{\diamond_1, \diamond_2\}$ when calculating $S_{A_i}(t)$ and $S_{I_i}(t)$ is determined by modelers when they list all the activators and inhibitors of an element x_i. Additionally, a weight, w_{ij}, can be assigned to the interaction between x_i and x_j, and included in calculations of arithmetic score when applying function g_2 such that:

$$S_{A_i}(t) = g_2(w_{A_{i,1}} * x_{A_{i,1}}(t), w_{A_{i,2}} * x_{A_{i,2}}(t), \ \dots, w_{A_{i,m}} * x_{A_{i,m}}(t)) \tag{3-7}$$

$$S_{I_i}(t) = g_2(w_{I_{i,1}} * x_{I_{i,1}}(t), w_{I_{i,2}} * x_{I_{i,2}}(t), \ \dots, w_{I_{i,l}} * x_{I_{i,l}}(t)) \tag{3-8}$$

Since cellular components do not change their expression or activity level suddenly, and instead they degrade or form steadily, we apply a *staircase* function f, as shown in Equation (3-9), to increment/decrement an element's value at simulation step t+1 depending on the calculated regulation score $S_i(t)$.

$$x_i(t+1) = f(x_i(t), S_i(t)) = \begin{cases} min(N-1, \ x_i(t) + \delta) & S_i(t) > 0 \\ balancing/spontaneous & S_i(t) = 0 \\ max(0, \ x_i(t) - \delta) & S_i(t) < 0 \end{cases} \tag{3-9}$$

where δ is a discrete increment/decrement value that is determined by the modeler. In this work, we assume $\delta = 1$, that is, we increase/decrease the value of the regulated element by at most one level. In general, a modeler can choose to use $\delta \geq 1$. For example, a threshold value can be given,

S_T, such that, when $S_i(t) > S_T$, and $x_i(t) < N - 2$, the next value, $x_i(t+1)$, can increase directly to $N - 1$.

When an element has both activators and inhibitors, its regulation score can be either positive, zero, or negative. As shown in Equation (3-9), positive or negative scores indicate that the element value will increase or decrease, respectively, by a value δ. Additionally, a zero score is also possible, and it can occur when there is a balance between positive and negative regulation. In this work, we allow modelers to specify whether the value of the regulated element should increase, decrease, or hold when the score is zero; and we call this a *balancing behavior*.

For elements with only activators (or only inhibitors), the regulation score can only be positive (negative) or zero and consequently, the value of elements with only activators (inhibitors) will only ever increase (decrease) or hold. In order to make the value of the elements with only activators (or only inhibitors) decrease (increase) when the regulation score is zero, we allow modelers to choose whether or not they apply a *spontaneous behavior* (i.e. spontaneous decrease or spontaneous increase) for those elements.

When spontaneous decrease is applied for an element with only activators, the update function *f* will take the form:

$$x_i(t+1) = f(x_i(t), S_i(t)) = \begin{cases} min(N-1,\ x_i(t) + \delta) & S_{A,i}(t) > 0 \\ max(0,\ x_i(t) - \delta) & S_{A,i}(t) = 0 \end{cases} \quad (3\text{-}10)$$

Similarly, in case of applying spontaneous increase for elements with only inhibitors, the function *f* will be defined as:

$$x_i(t+1) = f(x_i(t), S_i(t)) = \begin{cases} ma\, x(0,\ x(t) - \delta) & S_{I,i}(t) < 0 \\ min\ (N-1,\ x_i(t) + \delta) & S_{I,i}(t) = 0 \end{cases} \quad (3\text{-}11)$$

The application of spontaneous behavior is necessary for the value of the regulated element to follow the calculated value of its regulatory set (in this case, either activators or inhibitors). However, we still include the option to disable this behavior for cases where the modeler may wish to represent a cumulative regulation effect. It is worth mentioning that the Discrete-DiSH simulator can be used to simulate Boolean models by applying the spontaneous behavior to all elements that have only positive or only negative regulators in addition to choosing the balancing behavior to be 'decrease' for the elements that have both activators and inhibitors and setting N = 2.

4.0 Model Representation[2]

In this chapter, we introduce a representation format that can be used to construct and represent discrete models of complex systems with focus on intra- and inter-cellular biological networks. The introduced format enables biologists to build models more easily and to use all the functionalities that included in DiSH-simulator. It is important to note that the tabular representation does not include final update rules, that is, the tabular version of the model is further translated into an executable model that can be simulated.

4.1 Tabular Representation Format

The model is created and stored in a spreadsheet table where each row in the model table corresponds to one specific model element (i.e., modeled system component), while the columns are organized in several groups: (i) information about the modeled system element, (ii) information about the element's regulators, and (iii) information about knowledge sources. This format enables straightforward model extension to represent both additional system elements as new rows in the table, and additional element-related features by including new columns in the table.

---

[2] Based on Khaled Sayed, Cheryl A. Telmer, Adam A. Butchy, and Natasa Miskov-Zivanov. "Recipes for translating big data machine reading to executable cellular signaling models." In International Workshop on Machine Learning, Optimization, and Big Data, pp. 1-15. © [2017] Springer Nature.

The first group of fields in our representation format includes **element-related information**. This information is either used by the executable model, or kept as background information to provide specific details about the element when creating a hypothesis or explaining outcomes of wet-lab experiments. The first group of fields includes the following:

A- `Name` – full name of element, e.g., "Epidermal growth factor receptor".

B- `Nomenclature ID` – name commonly used in the field for cellular components, e.g., "EGFR" is used for "Epidermal growth factor receptor".

C- `Type` –types of biological components that are stored in different databases as in Table 4-1.

D- `Unique ID` – we use identifiers corresponding to elements that are listed in databases, according to Table 4-1.

E- `Location` - we include subcellular locations and the extracellular space, as listed in Table 4-2.

F- `Location ID` – we use location identifiers as listed in Table 4-2.

G- `Cell line` – The cell line of the wet-lab experiment.

H- `Cell type` – The type of cells that are used in the experiment.

I- `Tissue type` – The name of the tissue that contains the cells.

J- `Organism` – The name of the organism where the cells are extracted.

K- `Variable name` – variable names, currently include above described fields B, C, E, and H.

Table 4-1 Element type and ID database.

Element Type	Database Name
Protein	UniProt (UniProt)
Protein Family	Pfam(Pfam), InterPro (InterPro)
Protein Complex	Bioentities (Bioentities)
Chemical	PubChem (PubChem)
Gene	HGNC (HGNC)
Biological process	GO (GO), MeSH (MeSH)

Table 4-2 The list of cellular locations and their IDs from the Gene Ontology (GO) database.

Location Name	Location ID
Cytoplasm	GO:0005737
Cytosol	GO:0005829
Plasma Membrane	GO:0005886
Nucleus	GO:0005634
Mitochondria	GO:0005739
Extracellular	GO:0005576
Endoplasmic Reticulum	GO:0005783

The second group of fields in the representation format includes element **regulators-related information** that is mainly used by executable models, with a few fields used for bookkeeping, similar to the first group of fields.

L- `Positive regulators` – list of positive regulators of the element.

M- `Negative regulators` – list of negative regulators of the element.

N- `Interaction type` – for each listed regulator, in case it is known whether interaction is direct or indirect.

O- `Interaction mechanism` – for each known direct interaction, if the mechanism of interaction is known.

P- `Spontaneous behavior` – This column can include one of the three options {NA, Yes, No} where "NA" means not applicable and it is used when the regulated element has both "Activators" and "Inhibitors" while "Yes" or "No" are used to indicate whether the modeler wants to apply the spontaneous regulation or not in case of having only "Activators" or only "Inhibitors".

Q- `Balancing behavior` – the "balancing behavior" column can include one of four options: {NA, Decrease, Increase, Hold}. If an element has only "Activators" or only "inhibitors", the corresponding "balancing behavior" value will be "NA"; otherwise, the modeler should indicate whether the regulated element should "Decrease", "Increase" or "Hold" its value when the regulation score, $S_i(t)$ is zero.

R- `Maximum Discrete Level (N)` –Allows modelers to assign a different number of discrete levels for each element.

S- `Initial value` – Contains the initial discrete value of the element. If the modeler wants to randomly initialize and element, the word "random" can be placed instead of the discrete value.

The third group of fields in our representation includes **interaction-related provenance information.**

T- `Reference paper IDs` – for each interaction, we list IDs of published papers that mention the interaction.

U- `Sentences` – for each interaction, we list sentences describing the interaction.

It is worth mentioning that this representation format can be converted into the SBML format to be used by different software tools and shared between different working groups. Additionally, the tabular format provides an interface that can be easily created or read by biologists, and generated or parsed by a machine.

4.2 Representation of Common Biological Motifs[3]

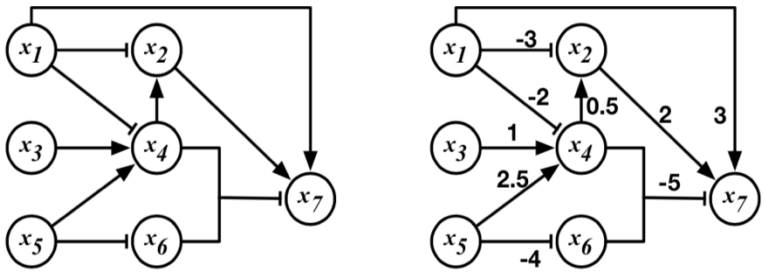
Biological interactions vary in nature and can take multiple forms. In this section, we show how to represent some of the different biological interactions into our representation format in order to build a model in a standardized way. Here, we assume that biological processes such as phosphorylation, acetylation, methylation, and transcription represent positive regulations, while

[3] Based on Sayed, Khaled, Cheryl A. Telmer, and Natasa Miskov-Zivanov. "Motif modeling for cell signaling networks." In 2016 8th Cairo International Biomedical Engineering Conference (CIBEC), pp. 114-117 ©[2016] IEEE.

dephosphorylation, ubiquitination, demethylation, and gene degradation represent negative regulations.

4.2.1 Simple Interactions

Simple interactions such as activation and inhibition are translated into the tabular format by listing all the activators in the `Positive regulators` column and all the inhibitors in the `Negative regulators` column. If an element has multiple activators or multiple inhibitors, the modeler has to list them in specific notations in order to indicate which logic or arithmetic operator should be used to evaluate the regulation score. To apply logical scoring, the activators and inhibitors have to be separated by a comma or a comma inside parentheses to indicate max (logical OR) or min (logical AND) operations respectively. Alternatively, to apply the arithmetic scoring, the activators and inhibitors have to be listed using plus, minus, and multiplication signs. Figure 4-1 shows a toy example of a small network, with and without weights, that can be simulated by either calculating logical scores (Figure 4-1(a)) or arithmetic scores (Figure 4-1(b)). The representation of the network in Figure 4-1(a) is shown in Table 4-3 while the representation of the network in Figure 4-1(b) is described in Table 4-4.



a) b)

Figure 4-1 A toy example for a small network that can be simulated by calculating (a) logical scores or (b) arithmetic scores

Table 4-3 A tabular representation for the toy example in Figure 4-1(a).

Variable name	Positive regulators	Negative regulators
x_2	x_4	x_1
x_4	x_3, x_5	x_1
x_6		x_5
x_7	x_1, x_2	(x_4, x_6)

Table 4-4 A tabular representation for the toy example in Figure 4-1(b).

Variable name	Positive regulators	Negative regulators
x_2	$0.5 * x_4$	$3 * x_1$
x_4	$1 * x_3 + 2.5 * x_5$	$2 * x_1$
x_6		$4 * x_5$
x_7	$3 * x_1 + 2 * x_2$	$5 * (x_4 * x_6)$

4.2.2 Binding Interactions

Binding interactions represent formation of protein complexes in most cases. However, in order to include both individual proteins and complexes in which they participate within a single model, we defined rules for representing complexes into our model representation format as described by the example shown in Figure 4-2.

If an element is a complex that is formed by the binding of two other elements, we create a separate table row for each component of the protein complex, and change the regulation set as follows: If the formation of complex x_2/x_3 is regulated by x_1, then we create two rows; one for element x_2, which is also positively regulated by x_7, and one for element x_3. The positive regulation rule for element x_2 becomes (x_1 AND x_3) OR x_7, while the positive regulation rule for element x_3 becomes (x_1 AND x_2). Additionally, if an element is regulated by a complex, we list all components of that complex as positive regulators for the element. In the example in Figure 4-2, the positive regulation rule for element x_4 is (x_2 AND x_3) because x_4 is regulated by the complex x_2/x_3.

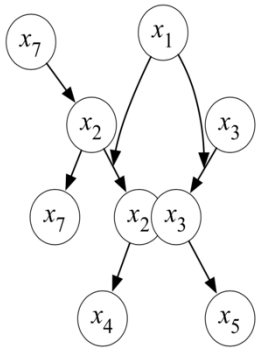


Figure 4-2 Schematic representation of a situation common to many biological signaling pathways where the regulation of complex formation, x_2 binding to x_3, is regulated by a third protein, x_1, so that the x_2/x_3 complex can activate x_4 and x_5.

4.2.3 Nested Interactions

Some interactions are nested where an element can regulate another activation or inhibition interaction. The following sub-sections show several examples of these interactions and how they are translated into the model tabular format.

4.2.3.1 Positive Regulation of Activation

As shown in Figure 4-3(a), element x_1 is activating element x_2, while element x_3 is positively regulating the interaction between x_1 and x_2. We also include in this and the following examples an element x_4 and assume that x_4 is a negative regulator of x_2 to show how the nested interactions can be combined with the simple interactions in our model format.

The example in Figure 4-3(a) shows that x_3 will activate x_2 only when x_1 is active. If x_1 is inactive, only x_4 will inhibit x_2, while x_3 will not have any effect on x_2. This can be represented as shown in the first row of Table 4-5 where the notation "$\{x_1\}[x_3]$" is used to indicate that x_3 is a necessary activator of x_2 but not sufficient since x_1 has to be active in order to see the effect of x_3. We call this notation as "necessary pair". The activators score of element x_2, $S_{A_{x_2}}(t)$, is calculated for the necessary pair "$\{x_1\}[x_3]$" as:

$$S_{A_{x_2}}(t) = max(x_1, x_3)\ only\ if\ x_1 > 0 \qquad (4\text{-}1)$$

If we assume that x_1, x_2, and x_3 are multi-valued variables with three discrete levels (i.e. 0, 1, and 2), and in case of x_1 = 1 and x_3 = 2, then $max(x_1, x_3)$ will be 2 which means that x_3 enhances the positive regulation of x_2 by x_1.

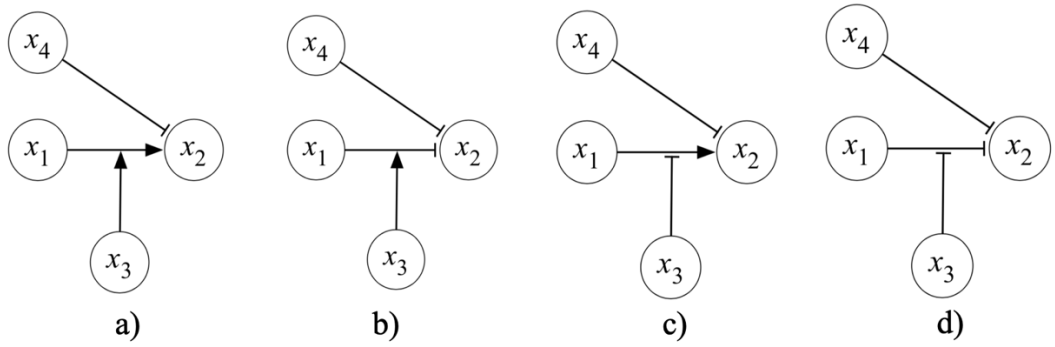


Figure 4-3 Examples of nested interactions. a) Positive regulation of Activation interaction, b) Positive regulation of Inhibition interaction, c) Negative regulation of Activation interaction, d) Negative regulation of Inhibition interaction

Table 4-5 A tabular representation for the nested interactions in Figure 4-3.

Variable name	Positive regulators	Negative regulators
x_2	$\{x_1\}[x_3]$	x_4
x_2		$\{x_1\}[x_3], x_4$
x_2	$(x_1,!\, x_3)$	x_4
x_2		$(x_1,!\, x_3), x_4$

4.2.3.2 Positive Regulation of Inhibition

Figure 4-3(b) shows an example of a nested interaction where $x_1$inhibits x_2, and x_3 positively regulates this inhibition, which means that x_3 will increase the inhibition of x_2 by x_1, when x_1 is active. This motif is translated into our tabular format as shown in the second row of Table 4-5. The inhibition score for element x_2 is calculated as:

$$S_{I_{x_2}}(t) = max(x_4, max(x_1, x_3)\ only\ if\ x_1 > 0) \quad (4\text{-}2)$$

In case of x_1 = 1 and x_3 = 2, $S_{I_{x_2}}(t)$ will equal $max(2, x_4)$ which means that x_3 enhances the inhibition of x_2 by x_1. However, if x_1 is zero, $S_{I_{x_2}}(t)$ will equal x_4 (the other inhibition signal).

4.2.3.3 Negative Regulation of Activation.

The example in Figure 4-3(c) shows that x_3 negatively regulates the activation of x_2 by x_1. So, if x_1 is inactive/low, x_3 will not have any effect on x_2. The translation of this event is show in the third row of Table 4-5 and the activation score is calculated as follows:

$$S_{A_{x_2}}(t) = min(x_1, NOT\ x_3) \quad (4\text{-}3)$$

Where "!" is used to indicate logical NOT (or N's complement).

4.2.3.4 Negative Regulation of Inhibition.

Figure 4-3(d) shows that x_3 negatively regulates the inhibition of x_2 by x_1 which is translated into the tabular format as shown in the fourth row of Table 4-5. The inhibition score is calculated for this interaction as:

$$S_{I_{x_2}}(t) = max(x_4, min(x_1, NOT\ x_3)\ only\ if\ x_1 > 0) \quad (4\text{-}4)$$

Which means that the inhibition of x_2 by x_1 can happen only when x_3 is inactive. It is worth mentioning that Equations (4-1) through (4-4) utilize logical scoring. However, the arithmetic scoring can also be applied by replacing *min* and *max* by multiplication and addition signs.

4.2.4 Gene expression

Gene expression is a process of synthesizing proteins to reflect the information encoded by their corresponding genes. To model gene transcription and translation, we define three model elements and their corresponding variables: gene, mRNA, and protein. As shown in Figure 4-4, the activation of gene X is controlled by regulators such as transcription factors (TFs) which work as positive regulators and/or inhibitors (Inh). An mRNA, Xrna, is then activated and only regulated by gene X. The activation of Xrna leads to synthesis of protein, Xprotein. It is worth mentioning that the mRNA is used to synthesize proteins that are then translocated to various cellular compartments (e.g. cytoplasm or plasma membrane).

The translation of the gene expression motif into the tabular format is shown in Table 4-6 where we create three rows for the three variables and list their regulators on the positive and negative regulators columns.

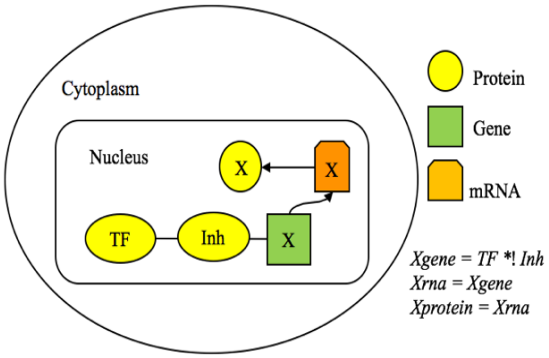


Figure 4-4 Gene expression motif.

Table 4-6 A tabular representation for the gene expression motif in Figure 4-4.

Variable name	Positive regulators	Negative regulators
Xgene	TF	Inh
Xrna	Xgene	
Xprotein	Xrna	

4.2.5 Receptor activation

To implement receptor activation in our models, we define three elements: receptor on the cell membrane, ligand in extracellular space, and activated receptor on membrane. As shown in Figure 4-5, receptor Y exists on the cell membrane but it is not active until it binds with the extracellular ligand X. Only the activated version of Y can relay the signal to the cytoplasmic molecules and activate protein Z. In addition, receptor Y and the extracellular ligand X may result from expression of X and Y gene elements in the model. The translation of this motif is shown in Table 4-7 where Y has to be initialized at high to indicate its existence.

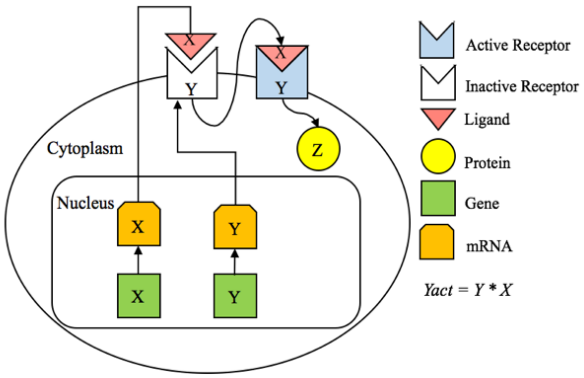


Figure 4-5 Receptor activation motif.

Table 4-7 A tabular representation for the gene expression motif in Figure 4-5.

Variable name	Positive regulators	Negative regulators
Y		
Yact	(Y,X)	

4.2.6 Translocation Motif

Some cellular elements have to be translocated from their original cellular compartment to other compartments to regulate elements there. To model this process, we define two elements in our model: elements in its original compartment, and element in the new compartment. While the element in original compartment can be regulated by other elements, the translocated element is regulated by the original element and possibly by the translocation regulators. For example, element Z may be regulated first by the active form of receptor Y before it translocate into the nucleus as depicted in Figure 4-6. Also, element Z inside the nucleus (Znuc) is only regulated by the cytoplasmic form of Z (Zcyto). Only Znuc and not Zcyto can regulate element X, which can be translocated into the extracellular matrix (Xex) after activation. There might be some other examples of translocations such as the translocation from the nucleus to the cytoplasm (element A in Figure 4-6) and the direct diffusion of some molecules (e.g. steroid hormones) from the extracellular matrix into the cytoplasm (element B). The representation of the translocation motif in the tabular format is shown in Table 4-8 assuming that there are no inhibitors.

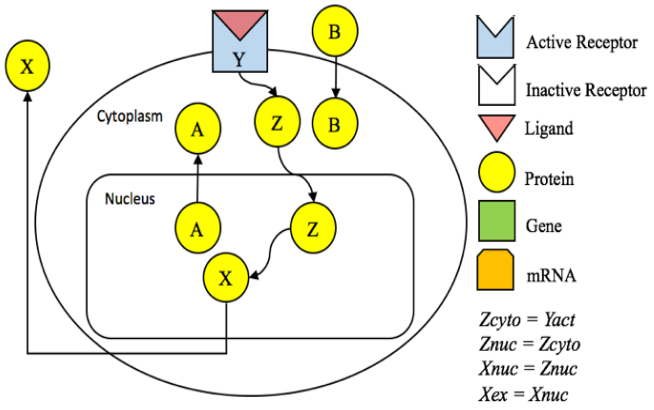


Figure 4-6 Translocation motif.

Table 4-8 A tabular representation for the gene expression motif in Figure 4-6.

Variable name	Positive regulators	Negative regulators
Zcyto	Yact	
Znuc	Zcyto	
Xnuc	Znuc	
Xex	Xnuc	

5.0 Time in Discrete Models

The time abstraction methods that are presented in Chapter 2 show that a more accurate representation of timing of biological events can take one of two forms: (1) delaying the regulation signals by either inserting extra nodes between the element and its regulators (e.g. the buffer insertion and the "dummy" nodes methods) or by updating the regulated element's value according to delayed versions of its regulators (e.g. the Boolean delay equations method), (2) delaying the transition of the element's value from one Boolean level to another after the update rule is evaluated (e.g. Piecewise linear differential equations).

Although the time abstraction methods that are briefly described in Chapter 2 provide a way for incorporating timing information into logical models, each method has its drawbacks. For instance, modelers cannot assign different amounts of delays for the activation verses the inhibition using the buffer insertion method because the inserted nodes delay the signal from specific regulators to the regulated element and do not delay the regulation process (i.e. activation or inhibition). Also, when the random sequential update scheme is utilized, the signal can take a longer time to propagate from the regulator to the regulated element than previously specified because updating some of the delay nodes might be skipped due to the random updates. On the other hand, the "dummy" nodes method which allows for assigning delays to the regulation process, does not allow for delaying the regulation signal which might be needed to represent some biological motifs such as translocation. Additionally, the model with dummy nodes becomes more complicated with adding a few numbers of delays as shown in Figure 2-4(b) where adding 2 delay steps for activation and 3 delay steps for degradation ended up with a large logic circuit with many update rules even without assigning inhibition delays. Models with BDEs are event-triggered

which only allow for delaying the regulation signal and not the regulation process. On the other hand, models with piecewise linear differential equations allow for delaying the regulation process and not the individual regulation signals. Also, the delay values are determined by the reaction rate constants of the biological event which are quantitative values that are difficult to measure especially for large models.

Therefore, developing methodologies for representing time in discrete models considering the different timescales of systems events, the different natures of biological motifs and the stochasticity of the complex systems events is necessary. In this chapter, we describe our methodology for incorporating timing information into discrete models of complex networks that can be built using automatic literature reading. The proposed methods allow for simulating systems events with different timescales using discrete-time, discrete-state simulators such as DiSH (Khaled Sayed, Yu-Hsin Kuo, Anuva Kulkarni, & Natasa Miskov-Zivanov, 2017a). Unlike the existing methods, we allow modelers to model the different timing of both signal propagation and element regulation which provides a more realistic time representation in discrete models of complex networks as well as giving flexibility to modelers to model different natures of events.

5.1 Time Modeling

In order to incorporate timing information into discrete models of complex networks, we define two timing parameters:

(1) Propagation delay, τ_{ij}, which is defined as the length of time needed for a change in a regulator, x_i, to be seen by a regulated element, x_j.

(2) Response time, θ_j, which is defined as the length of time needed for a regulated element x_j to change its value from one discrete level to another after its regulation score, S_j, changes.

The propagation delay, τ_{ij} can be defined as a set of multiple delays representing each transition in the value of a regulator, x_i, (i.e. the transition from one activity level to another) such that

$$\tau_{ij} \in \{\tau_{ij,0\to1}, \tau_{ij,1\to2}, \dots, \tau_{ij,N-2\to N-1}, \tau_{ij,N-1\to N-2}, \dots, \tau_{ij,2\to1}, \tau_{ij,1\to0}\}$$

where *N* is the maximum number of discrete levels assigned to element v_i. However, we assume that the propagation time is the same for all transitions because the signal travels through the same medium and we can use only one τ_{ij} value.

Similarly, the response time, θ_j, can be defined by multiple regulation delay variables representing each transition in the value of the regulated element, x_j, such that

$$\theta_j \in \{\theta_{j,0\to1}, \theta_{j,1\to2}, \dots, \theta_{j,N-2\to N-1}, \theta_{j,N-1\to N-2}, \dots, \theta_{j,2\to1}, \theta_{j,1\to0}\}$$

Figure 6-1 shows the representation of the toy example in Figure 2-2 using the proposed methods as well as the equivalent schematics that show how the scoring functions are applied. Two different methods are introduced in this work for updating the element's value when propagation delays and response times are used; (1) general state transitions and (2) conditioned state transitions. In the general state transitions method, the response time, θ_j, is related to the transition from one discrete level to another, regardless of the regulators' values (i.e. the regulated element's value will go up/down as long as the activators score is higher/lower than the inhibitors

score) while in the conditioned state transitions, the response time, θ_j, can vary with varying the regulators' values as explained in section 5.1.2.

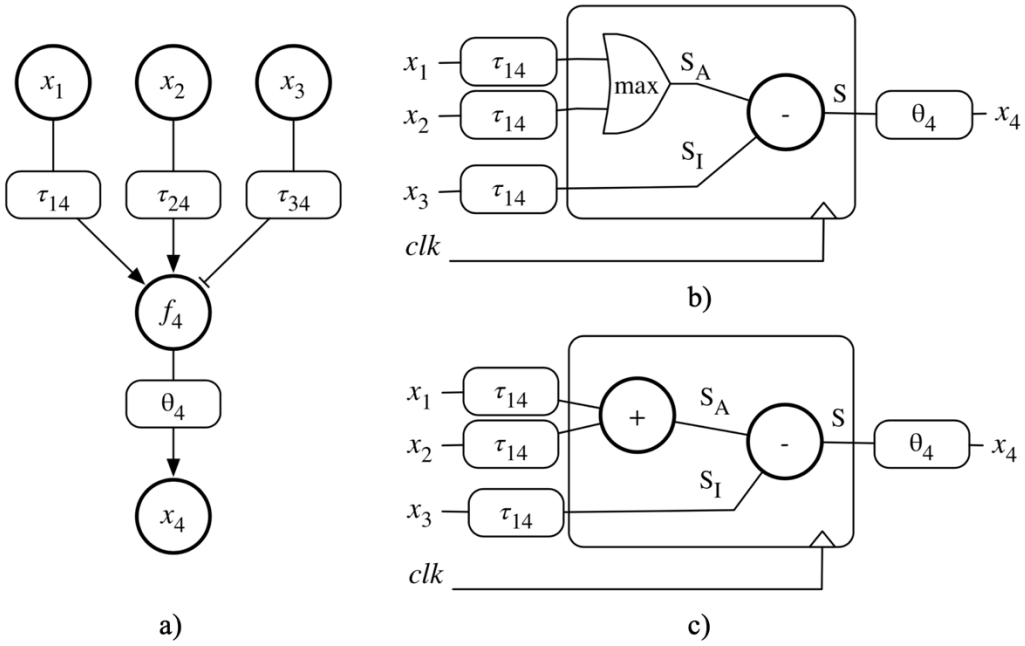


Figure 5-1 The representation of the toy example in Figure 2-2 using the proposed methods. (a) The modified toy example showing the propagation and the regulations delays, (b) The element update schematic using the logical scoring, and (c) the element update schematic using the arithmetic scoring.

5.1.1 General State Transitions

To update the regulated element's value under the general state transitions method, the regulation score, $S_j(t)$, that is defined by Equation (3-6) has to be changed to include the propagation delay, τ_{ij}, as follows:

$$S_i(t) = S_{A_i}(t) - S_{I_i}(t) \tag{5-1}$$

$$S_{A_j}(t) = g(x_{A_{j,1}}(t - \tau_{1j}), x_{A_{j,2}}(t - \tau_{2j}), \dots, x_{A_{j,m}}(t - \tau_{mj})) \tag{5-2}$$

$$S_{I_j}(t) = g(x_{I_{j,1}}(t - \tau_{1j}), x_{I_{j,2}}(t - \tau_{2j}), \dots, x_{I_{j,l}}(t - \tau_{lj})) \tag{5-3}$$

where $x_{A_{j,i}}$ ($x_{I_{j,i}}$) is the i^{th} activator (inhibitor) of the regulated element, v_j. Accordingly, the next value of a regulated element, $x_j(t+1)$, will be defined as:

$$x_j(t+1) = f(x_j(t), S_j(\theta_j)) = \begin{cases} min(N-1,\ x_j(t) + \delta) & S_j(\theta_j) > 0 \\ balancing/spontaneous & S_j(\theta_j) = 0 \\ max(0,\ x_i(t) + \delta) & S_j(\theta_j) < 0 \end{cases} \tag{5-4}$$

where $S_j(\theta_j)$ represents the regulation score during the whole response time period θ_j.

In some cases, the regulation score $S_j(\theta_j)$ changes before the response time, θ_j, has elapsed. Therefore, we apply a "noise rejection" technique in order to avoid any unnecessary and quick changes. In this work, we define "noise" as follows: if the regulation score, $S_j(\theta_j)$, changes from one state to another and stays at the new state for a small number of time steps (i.e. less than or equal to $\tau_{Noise} < \theta_j$), the new state of $S_j(\theta_j)$ is rejected and the state of $S_j(\theta_j)$ that was stable for a number of steps $> (\theta_i - \tau_{Noise})$ is used to update the value of $x_j(t+1)$. The value of τ_{Noise} can be set by the modeler based on background knowledge, otherwise, it is assumed as $\tau_{Noise} = 0.1 * \theta_j$.

5.1.2 Conditioned State Transitions

We allow modelers to assign different response times under different regulation conditions. For example, if an element has *k* activators and *k* inhibitors, one can assign a long response time if the inhibitors have the same values as the activators and a short response time if the differential strength of activators and inhibitors is large. In this case, instead of applying the scoring and update functions given by Equations (5-1) and (5-4), we allow modelers to create a truth table with a different response times, θ_j, under different conditions (i.e. all possible values of activators and inhibitors.) as shown in Table 5-1 which represents an example truth table for the toy example in Figure 5-2 assuming that x_1, x_2, and x_3 are discrete variables with three levels (i.e. 0, 1 , and 2). The shaded values on the columns show all possible states of x_2 and x_3 at simulation step *t* while the shaded values on the rows represent all possible states of x_1 at step *t.* The values inside the cells represent the value of the regulated element x_3 at simulation step t+1. The number of assigned delays under each regulation condition are indicated by the number placed after the point in each cell. For example, 1.4d means that the value $x_3(t+1)$ will change to 1 after 4 delay steps if the values of x_1, x_2, and x_3 stay at 1, 1, and 0 for four consecutive simulation steps. The concept of noise reduction is also applied here which means that if any of the regulators' values switched from one discrete level to another and returns back to its previous value within τ_{Noise}, this change will be considered as noise and will be rejected. Otherwise, the new values of the regulators will be considered in the next update of the regulated element.

In this work, we introduce two techniques for considering delays when updating the regulated element's value using the conditioned state transitions method: (1) Reset Delays, and (2) No-Reset Delays as explained below.

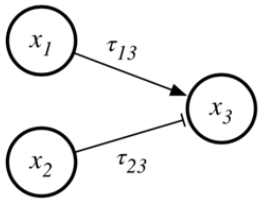


Figure 5-2 A toy example of three interacting components.

Table 5-1 A truth table with delays for x_3 in the toy example of Figure 5-2.

		$x_2(t-\tau_{23}), x_3(t)$								
		0,0	0,1	0,2	1,0	1,1	1,2	0,2	1,2	2,2
$x_1(t-\tau_{13})$	0	0	1	2	0	0	1	0	1	0
	1	1	2	2	1.4d	2.5d	2	2	1	1.4d
	2	1	2	2	1.2d	2.3d	2	2	2	1.2d

5.1.2.1 Reset Delays

In the reset delays method, the regulators values as well as the current value of the regulated element (i.e. a specific regulation condition) have to hold for the whole period of the assigned response time, θ_j, in order for the regulated element to change its state. For example, x_3 should change its current state from 0 to 1 after 4 delay steps when $x_1 = 1$ and $x_2 = 1$ according to Table 5-1. However, if the value of either x_1 or x_2 changes during the 4 delay steps, the value of x_3 doesn't change and we restart counting the 4 delay steps whenever the same condition is repeated (i.e. the condition when $x_1 = 1,\ x_2 = 1$, and $x_3 = 0$).

5.1.2.2 No-Reset Delays

In the No-Reset delays method, the response time, θ_j, is related primarily to the next state of the regulated element instead of depending on both the regulators' values and the current value of the regulated element. For example, the next value of x_3 is equal to 1 under 4 conditions according to Table 5-1 where x_3 switches to 1 after 4 delay steps when $x_1 = 1,$ and $x_2 = 1\ or\ 2$ and switches to 1 after 2 delay steps when $x_1 = 2,$ and $x_2 = 1\ or\ 2$. If any of these conditions occur, we start counting up until the assigned response time has elapsed and then, we switch the value of x_3. However, if the regulation condition changes to another condition that is supposed to lead to the same outcome (i.e. the same next value) before the response time has elapsed, we keep counting up and do not reset the counting. For example, if $x_1 = 2$ and $x_2 = 2$, the delay variable starts from 0 and increases to 2 (i.e. the assigned delays under that condition) and then the value of x_3 switches from 2 to 1. However, if the condition changes after only 1 delay step to another condition that should lead to the same next value (i.e. $x_3 = 1$) such as $x_1 = 1$ and $x_2 = 1,$ we start counting from 1 instead of 0 and keep increasing the counting up to 4 and then switch the value of x_3 from 0 to 1.

5.1.3 Spontaneous and Balancing delays

When the spontaneous and balancing behaviors are applied, a delay can be assigned to each transition in the element's value to mimic the amount of time needed for an element to go back to its previous state when the regulation signal is off. For example, if an element has only activators and the activation signal is turned off, a modeler can slow down the decrease in the element's value over time by adding more delays to the spontaneous decrease option. Similarly, if an element has both positive and negative regulators and the score is balanced, the modeler can specify the amount

of time needed for the regulated element's value to increase or decrease by adding a suitable number of delay steps.

5.2 Timing in Motifs

In this section, we propose some rules for assigning different delays to different biological motifs. These rules do not show how to choose the best propagation and/or regulation delays for each element, but instead they show whether we should assign delays for a biological event or not in order to standardize the way of creating discrete models for inter- and intra-cellular networks especially when an automated reading technique is involved in building large models.

5.2.1 Receptor Activation

Since some receptors take a considerable time in order to get activated once the ligands bind to them, we specify a regulation delay, θ, to each receptor in the model in order to represent the time needed for homo or heterodimerization and recruitment of adaptor proteins that are involved in the activation process. Here, we assume that the ligands are already in the extracellular space and hence we do not assign a propagation delay to the ligand. However, if a ligand has to move from a cellular compartment to the extracellular space, one can assign a propagation delay to represent the translocation time.

5.2.2 Translocation

If a regulator is not in the same cellular compartment as the regulated element, the regulator has to translocate to the same compartment first. Therefore, we create two variables representing the translocation process as described in section 4.2.6. Then, we specify a propagation delay, τ, to represent the time needed for an element to move from one compartment to another. For example, if an element v_i translocate from the nucleus to the cytoplasm, we create two variables, x_{i_cyto} and x_{i_nuc}, and assign a propagation delay to x_{i_nuc} which is the only activator of x_{i_cyto}.

5.2.3 Gene Expression

Gene expression is one of the biological events that require a large amount of time than protein-protein interactions since adaptor proteins and RNA polymerases has to be recruited before starting the process of transcription which requires more time. Therefore, we assign a regulation delay, θ, to all the gene variables in the model to represent the transcription time while another regulation delay is assigned to all the mRNA variables to represent the translation time. We also assume that the gene transcription process does not stop once started. Therefore, if the update rule of a gene variable is selected to be evaluated at a specific simulation step, we wait until the response time that is assigned to the selected gene has elapsed and then we update the value of the corresponding variable even if the regulation signal has changed during the response time.

5.2.4 Phosphorylation, Dephosphorylation, and Ubiquitination

We deal with most of the phosphorylation processes as activation events while the dephosphorylation and ubiquitination processes are assumed to be inhibition events. Therefore, we assign both propagation and regulation delays to these events.

5.2.5 Complex Formation

Complexes are usually formed from the binding of multiple elements and therefore, each element can have a propagation time. Additionally, if the complex takes time to be formed, we also assign a regulation delay.

5.3 Timing under Different Simulation Schemes

The effect of the added delays varies according to the simulation scheme that is used to simulate the model. In the following subsections, we show how the two major simulation schemes (i.e. Simultaneous and Random Sequential) affect the simulation trajectories when the propagation delay and the response time are applied. The toy example in Figure 5-1(a) will be used in the following subsections to illustrate the key differences between the simulation trajectories under different simulation schemes. Figure 5-1(a) shows how the small network in Figure 2-2 is realized in our modeling framework where a propagation delay, τ_{ij}, is assigned to each signal from the regulators (i.e. $x_1, x_2,\ and\ x_3$) to the regulated element, x_4 in addition to assigning a regulation delay, θ_4, to x_4 in order to represent the response time. Assuming that the logical relationship

between the activators of x_4 is OR, and assuming that we calculate a logical scoring, then the scoring function can be evaluated as:

$$S_{A_4}(t) = g\big(x_{A_{4,1}}(t-\tau_{14}), x_{A_{4,2}}(t-\tau_{24})\big) = max\big(x_2(t-\tau_{14}), x_2(t-\tau_{24})\big) \quad (5\text{-}5)$$

$$S_{I_4}(t) = g(x_{A_{4,3}}(t-\tau_{34})) = x_3(t-\tau_{34}) \quad (5\text{-}6)$$

$$S_3(t) = max\big(x_2(t-\tau_{14}), x_2(t-\tau_{24})\big) - x_3(t-\tau_{34}) \quad (5\text{-}7)$$

Then we apply Equation (5-4) to calculate the value of $x_4(t+1)$ if the update rule of x_4 is selected to be evaluated at any simulation step. For simplicity, we assume that N = 3, τ_{14} = 3steps, τ_{24} = 2 step, τ_{34} = 4 steps, θ_4 = 3 steps for any transition, and $\tau_{Noise} = 1$ step.

5.3.1 SMLN Scheme

In the SMLN update scheme, all update functions are evaluated at every time step. However, the value of the elements that require response time will not be updated until the response time elapses. Additionally, the value of the regulators that will be taken into consideration when the regulation score is calculated will be delayed by the amount of the assigned propagation delay as shown in the timing diagram in Figure 5-3 which shows how the value of x_4 changes over the simulation steps assuming arbitrary changes in the values of x_1, x_2, and x_3. At each simulation step, the value of the regulation score, S_4 is evaluated according to the values of x_i at simulation step $t - \tau_{i4}$, then the value of x_4 is updated according to the value of the regulation score, S_4, at time step $t - \theta_4$ considering the timing of the noise.

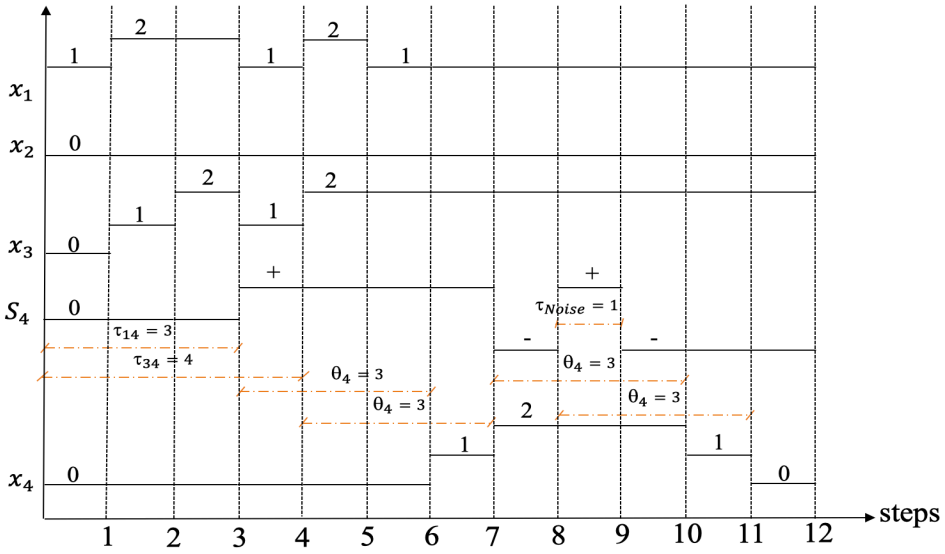


Figure 5-3 Timing diagram for the toy example in Figure 5-1(a) using the SMLN simulation scheme.

5.3.2 RSQ Scheme

Under the RSQ simulation scheme, only one update function is evaluated at each simulation step. Here, we assume that there are two signals: (1) a clock signal that generates simulation steps and (2) an *enable* signal that determines which update function is being evaluated at the current simulation step. The timing diagram in Figure 5-4 shows the changes in the value of x_4 using the RSQ simulation scheme where the red bars indicate the update functions that are chosen to be evaluated at each simulation step. It's shown that the value of S_4 has to stay at one state for a time period of $(\theta_4 - \tau_{Noise})$ in order to have a change in the value of x_4. Also, the state of S_4 changes according to the old values of the regulators (i.e. $x_1, x_2, and\ x_3$).

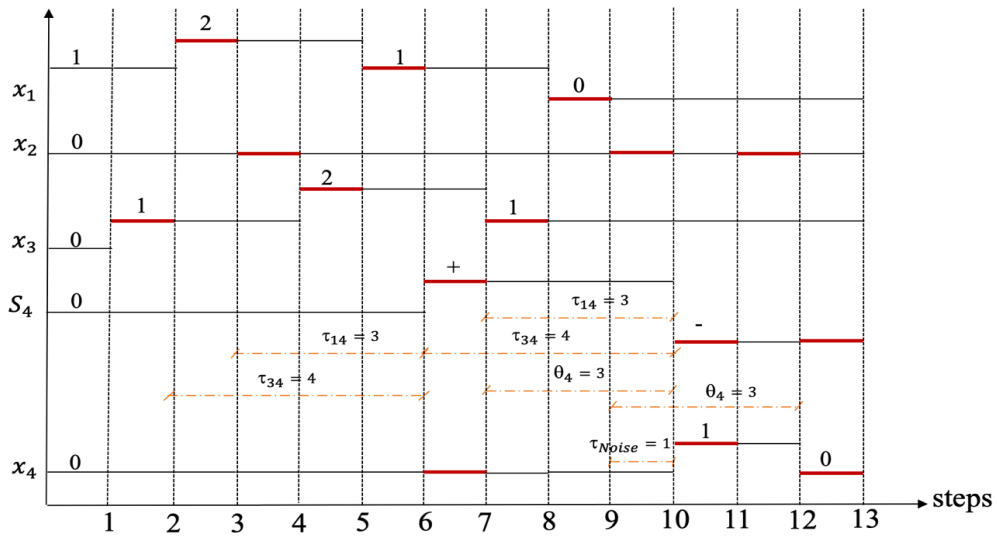


Figure 5-4 Timing diagram for the toy example in Figure 5-1(a) using the RSQ simulation scheme

5.3.3 Random-delay SMLN update scheme

Each of the two major update schemes (i.e. SMLN and RSQ) has its pros and cons. For instance, the SMLN scheme runs fast and is useful for studying the steady states of the system but it doesn't provide accurate transient responses. On the other hand, the RSQ scheme provides more accurate transient and steady state responses but it takes more time to run. With delays, we can run fast simulations using the SMLN scheme and still get accurate trajectories and dynamic responses with randomized delay values. In the Random-Delay SMLN scheme, the number of assigned delays is changed randomly in each simulation run leading to stochastic simulations. Here, we consider the assigned delay value as the mean for a statistical distribution (such as uniform distribution) and randomly choose a different delay value in each simulation run. The minimum and maximum values of the uniform distribution are defined as $x - \delta$ and $x + \delta$, respectively, where x is the assigned delay value and δ is a small number determined by the modeler.

5.3.4 Overview of the Proposed Delay Methods

The highlights of the proposed methods are shown in Table 5-2. Comparing the proposed methods to the methods summarized in Table 2-1, we can see that the proposed methods allow for modeling propagation delays as well as regulation times while the current methods in literature allow for representing either propagation delays or regulation times. Also, we allow for assigning fixed delays as well as varying delays where the values of the assigned delays can be chosen from an interval if the modeler wants to model the stochasticity in the system. We also incorporate our timing methods into discrete-time, discrete-state systems and run multiple simulation schemes such as Simultaneous, Random Sequential, and Random-Delay Simultaneous.

Table 5-2 Highlights of the proposed methods in terms of the comparison in Table 2-1.

	General State Transitions	**Conditioned State Transitions**
States, x	Discrete	Discrete
Time, t	Discrete	Discrete
Update orders	SMLN/Random-delay SMLN /RSQ	SMLN/RSQ
Type of Delays	Propagation and regulation delays	Propagation and regulation delays
Delay values	Fixed (assigned by a modeler) or varying (randomly chosen from an interval)	
Conditions for Delays	Different delays can be assigned depending on either the combined values of the regulators and the regulated elements or the type of transition that the regulated element is supposed to make.	
No-Reset Delays updates	Not Applied	The overall influence doesn't change during the delay interval, while individual regulator values may change, the expected regulated element transition stays the same.
Reset Delays updates	Not Applied	The overall influence doesn't change during the delay interval, and individual regulator values are not allowed to change; if they change, while the overall influence value doesn't change, the delay interval starts over.

5.4 Inferring Timing Parameters from Data

One of the goals of this work is to develop methods and tools that allow modelers to automatically incorporate timing information into discrete models. Therefore, providing an automated method for estimating the suitable number of delay steps needed for different events is helpful and can be used as a guide in building good models. In this section, we describe our methods for estimating the timing parameters for a few interactions based on the available data. For example, in Miskov-Zivanov et. al. (Miskov-Zivanov, Turner, Kane, Morel, & Faeder, 2013a), the flow cytometry data was used to count the percentage of cells that have high expression of Foxp3 under high and low antigen (Ag) doses after 18 hours, 3, and 7 days from the onset of the stimulation. Although the highest percentage of $Foxp3^+$ cells was ~33% in the experimental data under low Ag dose, the simulation results of the logical model in (Miskov-Zivanov, Turner, Kane, Morel, & Faeder, 2013a) showed that the percentage of cells with $Foxp3^+$ is a 100% which means that all naïve T cells should be differentiated into Treg cells under low Ag dose. This could lead to false conclusions especially when the obtained results under low Ag dose are compared to the simulation results of high TGFβ stimulation (in case of cancer) which show that almost all the naïve T cells are differentiated into Treg cells even with high Ag dose.

In order to automate the process of assigning delays to specific model elements and interactions, we utilized an optimization-based simulations technique which is the Nelder-Mead algorithm (Nelder & Mead, 1965) that runs many simulations and keep changing the timing parameters until it finds a minimum error between the simulation results and the experimental or historical data. The first step in applying the Nelder-Mead algorithm is to define a cost function which is basically the difference between the simulation trajectories and the experimental results.

In the following subsections, we describe our steps in defining and running the optimization-based simulation algorithm.

5.4.1 Data preparation

Before we run simulations, we need to define the cost function. The cost function in this work is defined as the mean square error between the simulation trajectories and the available time series data. More formally, if x represents the simulation trajectory of a model element, X, which has experimental or historical time series data, y, then the cost function f will be defined as:

$$f = \frac{1}{m}\sum_{i=1}^{m}(x_i - y_i)^2 \tag{5-8}$$

where m is the total number of data points. One important issue here is that x and y should have the same length and the same units. In discrete models, the trajectory of a discrete variable x is usually calculated as the average of the x values over hundreds or thousands of simulations runs. The average trajectory is then normalized by dividing its values by the maximum discrete level assigned to x which leads to trajectory values between 0 and 1. In order for x and y to have the same units, the historical or experimental data, y, should also be normalized. Equation 5-9 shows how the normalization is performed.

$$y_{normalized} = \frac{y - \min(y)}{\max(y) - \min(y)} \tag{5-9}$$

The next step is to make sure that both x and y have the same length. Although the length of the simulation trajectory is variable and defined by the modeler, the length of the historical data or experimental results is usually fixed and limited by the length of the collected data. In order to make both x and y have the same length, modelers have to assume how much time each simulation step represents. Fox example, in the naïve T cell differentiation model (Miskov-Zivanov, Turner, Kane, Morel, & Faeder, 2013a), the percentage of cells with high Foxp3 expression was collected at four time points which are the beginning of the Ag dose stimulation, after 18 hours, 3 days, and 7 days. Therefore, in order to map the simulation steps into actual timing, the modeler has to assume that each simulation step corresponds to u time units (e.g. sec, min, hour, … etc). Then, the total number of simulation steps is calculated as

$$total\ number\ of\ simulation\ steps = \frac{total\ \ time\ of\ data\ collection}{u} \qquad (5\text{-}10)$$

After determining the total number of simulation steps, a linear interpolation is performed in order to expand the length of the historical data and provide data points at each time unit. For example, if we assume that each simulation step corresponds to 5 min (i.e. $u = 5\ min/step$) and we know that the historical or experimental data in the naïve T cell differentiation model were collected over 7 days, then the total number of simulation steps should be $\frac{7(days)*24(hours)*60(\mathrm{min})}{5\ min/step} = 2016$ steps. The linear interpolation is then utilized to provide continuous data over the 2016 steps instead of the four data points that were collected over 7 days. Now, the length and the units of the experimental or historical data, y, are the same as the values of the normalized average trajectory, x, which allows for evaluating the cost function.

5.4.2 Optimization-based Simulations

The goal of the optimization-based simulation algorithm is to search for the optimal values of the timing parameters that minimize the cost function defined by equation 5-8. In this work, we apply the Nelder-Mead algorithm (Nelder & Mead, 1965) which utilizes a simplex or polytope of $n+1$ vertices in n dimensions to find a local optimum where n is the total number of timing parameters. For a model with l nodes, k interactions, and p discrete levels, the maximum number of timing parameters (i.e. propagation delay variables, τ, and regulation delay variables, θ) is calculated as:

$$n = k + l * [2 * (p - 1)] \tag{5-11}$$

where the value $[2 * (p - 1)]$ represents the number of transitions of the node value. For example, if a node is modeled by 3 discrete levels, the value of that node will have $2 * (3 - 1) = 4$ transitions which are the transitions from 0 to 1, 1 to 2, in case of activation, and from 2 to 1, 1 to 0 in case of inhibition.

Equation 5-10 shows that the number of parameters that the Nelder-Mead algorithm should find values for is increasing linearly with the number of nodes, interactions, and discrete levels which makes the task of finding a global optimum is very difficult and hence the algorithm will always be stuck in a local minimum that might not be the best value. Therefore, limiting the search space by specifying the most influential timing parameters will have a huge impact on the performance of the algorithm. Specifying the most important timing parameters requires a closer look at the initial simulation results in order to spot the regulators that have a large influence on

the nodes of interest. The following steps summarize the process of finding the optimal timing parameters semi-automatically:

1- Determine the nodes in the model that have historical or experimental data and call them elements of interest (EOI).
2- Map each simulation step to a specific time interval and calculate the total number of simulation steps using equation 5-10.
3- Perform data normalization as indicated by equation (5-11)
4- Perform linear interpolation on the available data in order to get values at each time point.
5- Run initial simulations and look at the average trajectories of the EOI and compare them with the interpolated data.
6- If the simulation trajectories have the same trend as the historical or experimental data, check how fast the simulation trajectories reach the steady states.
    a. If the simulation trajectories reach the steady state faster than the historical data, then (I) look at the trajectories of the regulators of the EOI and find the regulators that largely affect its behavior. (II) Add propagation or regulation delays to the regulator. (III) The regulator that largely influences the behavior of the EOI is now considered the new element of interest. (V) Repeat step number 6 until the simulation trajectories of the original EOI becomes as close as possible to the trend of the historical data.
    b. If the simulation trajectories reach the steady state slower than the historical data, then (I) look at the trajectories of the regulators of the EOI and find the regulators that largely affect its behavior. (II) Add delays to the other parts of

the network so that the signal that controls the behavior of the EOI propagates faster. (III) Repeat step number 6 until the simulation trajectories of the original element of interest becomes as close as possible to the trend of the historical data.

7- If the simulation trajectories do not have the same trend as the historical or experimental data, change the structure of the model or the update functions until they match.

6.0 Applications

In this chapter, we show results for applying the methods introduced in this dissertation on three discrete models of complex systems, namely, naïve T cell differentiation, buddying yeast cell cycle, and food security in South Sudan.

6.1 Naïve T cell differentiation

T cells are part of the immune system which can detect foreign bodies (*antigens*) and initiate immune responses in order to attack invaders. The naïve T cells can be differentiated into either helper cells (Th), which activate an immune response, or regulatory cells (Treg), which suppress the Th mediated immunity and prevent autoimmune diseases that can be caused by the prolonged activation of Th cells (Shevach, 2000). However, the uncontrolled suppression of the immune response by Treg cells can lead to serious diseases such as cancers (Curiel et al., 2004). Therefore, understanding the key factors that control the differentiation outcomes of naïve T cells into Th or Treg cells can help in developing therapies for activating Th cells in case of cancer, while activating Treg cells in case of autoimmune diseases (Facciabene, Motz, & Coukos, 2012). It has been shown that both strength and duration of the antigen (Ag) dose stimulation steer the differentiation process, where low Ag dose leads to the differentiation into Treg cells, while high Ag dose leads to the production of Th cells (Turner, Kane, & Morel, 2009). Additionally, Miskov-Zivanov et al. (Miskov-Zivanov, Turner, Kane, Morel, & Faeder, 2013a) showed that the duration of the Ag dose stimulation highly affects the differentiation outcome, where a prolonged high Ag

dose will lead to more Th cells, while a short-time high dose stimulation will produce more Treg cells.

6.1.1 Model Description

The logical model in (Miskov-Zivanov, Turner, Kane, Morel, & Faeder, 2013a) includes 39 elements representing extracellular molecules, cell receptors, and intracellular molecules that propagate the signal from the cell receptors to the nucleus of the naïve T cell to start gene expression. The two signals that are necessary to activate the naïve T cells include the binding of the major histocompatibility complex (MHC) molecules and the co-stimulatory ligand CD86 which are located on the surface of the antigen-presenting cells (APC) to the T cell receptor (TCR) and the co-stimulatory receptor CD28 respectively. Once the ligands on the APC bind to the receptors on the membrane of the naïve T cell, the signal is relayed to the intracellular space through activating three transcription factors which are Activator Protein 1 (AP-1), Nuclear Factor of Activated T-cells (NF-AT), and Nuclear Factor-κ-light-chain-enhancer of B-cells (NF-κB). These activated transcription factors translocate to the nucleus to start the expression of gene Interleukin 2 (*IL-2*) as well as the α subunit of IL-2 receptor (i.e. IL-2R, also known as CD25) (Kim & Leonard, 2002; Macian, 2005). The IL-2R receptor has three subunits; α, β, and δ. Subunits β and δ are always present on the cell membrane and by expressing the α subunit, IL-2R becomes capable of receiving IL-2 from the extracellular space and activate transcription factor Signal Transducer and Activator of Transcription 5 (STAT5) which activates Foxp3; a transcription factor that regulates the production of Treg cells through inhibiting the expression of *IL-2* (Turner et al., 2009; Yao et al., 2007; Ziegler, 2006).

The expression of *FOXP3* gene starts after the binding of either STAT5 alone, three transcription factors which are NF-AT, AP-1, and STAT5, or two transcription factors; NF-AT and mother against decapentaplegic homolog 3 (Smad3) to its promoter (Ohkura, Kitagawa, & Sakaguchi, 2013). Another feedback loop that controls the relative expression of IL-2 and Foxp3 is controlled by the expression of Foxp3 where the activity of Foxp3 upregulates PTEN -a phosphatase present in naïve cells- which is inhibited by the TCR activation (Buckler, Liu, & Turka, 2008). PTEN also inhibits the signaling through the PI3K/Akt/mTOR pathway and hence allow more Foxp3 expression which results in generating more Treg cells. By studying the expression levels of IL-2 and Foxp3 under different stimulation conditions, we can determine the type of the differentiated cells: Treg cells are characterized by the high expression of Foxp3 and low expression of IL-2, while Th cells are characterized by the opposite expression levels (i.e. high IL-2 and low Foxp3). The experimental results in (Miskov-Zivanov, Turner, Kane, Morel, & Faeder, 2013a) showed that almost all naïve T cells (>90%) are differentiated into Th cells under high Ag dose while ~33% of the naïve cells are differentiated into Treg cells under the low Ag dose, forming a *mixed population* of Th and Treg cells.

Recent data from (Kerdiles et al., 2010) has shown that the increase in PTEN levels, in case of low Ag dose stimulation, is due to the transcription factor FoxO1 while the phosphorylation of FoxO1 by Akt inhibits the differentiation into Treg by limiting the increase in PTEN levels. Hawse et. al. (Hawse et al., 2015) extended the model developed by Miskov-Zivanov et. al. (Miskov-Zivanov, Turner, Kane, Morel, & Faeder, 2013a) by incorporating the new information into the model and evaluating the expression levels of Foxp3 under low Ag dose. Their simulation results showed better performance where the percentage of $Foxp3^{+}$ cells was found to be ~70% which is

more closer to the experimental data than the simulation results of the original model that has been developed by (Miskov-Zivanov, Turner, Kane, Morel, & Faeder, 2013a).

6.1.2 Effect of simulation schemes

In the following discussion we refer to *run* as a single simulation run from the starting point, when we assign initial values to all variables, through a pre-determined number of rounds or number of steps (depending on the simulation scheme used). A *trajectory* of values is obtained for each element in the run, and the trajectories may vary across consecutive runs. The values for variables in each round or step are averaged over all the runs and this *average trajectory* can be plotted for analysis. In this section, we investigate the effect of the different simulation schemes introduced in sections 3.1 and 3.2 by simulating the original model using Boolean variables in order to keep similar simulation settings as in Miskov-Zivanov et. al. (Miskov-Zivanov, Turner, Kane, Morel, & Faeder, 2013a) where elements TCR, PIP3, and PI3K are implemented in the executable model using two Boolean variables, such as TCR_{HIGH} and TCR_{LOW} to represent discrete values 0, 1, and 2.

Here, we simulate two main scenarios; Scenario I, which includes 5 sub-scenarios, and Scenario II, which includes 3 sub-scenarios, all of them listed in Figure 6-1(g). The five sub-scenarios under Scenario I consider low and high levels of antigen dose (TCR_{HIGH}=0 and TCR_{LOW}=1, TCR_{HIGH}=1 and TCR_{LOW}=0, respectively), as well as different initial values of proteins TGF-β and AKT, also responsible for regulating the T-cell differentiation. On the other hand, the three sub-scenarios of Scenario II consider the antigen dose removal at certain time steps to reflect the impact of the time at which the high antigen dose is removed during the wet-lab experiments on the differentiation outcomes of naïve T cells. The removal of the high antigen dose

is simulated by toggling the value of the TCR_{HIGH} variable from 1 to 0 at specific simulation steps, which changes the value of the TCR variable from HIGH to LOW.

In this section, we show the simulation results obtained by running the simulator using the simulation schemes that were described in sections 3.1 and 3.2, for Foxp3 which is the main indicator of the differentiation outcomes. However, we discuss the response of Foxp3 in terms of other key elements such as IL-2, mTORC1, CD25, and STAT5, for the scenarios shown in Figure 6-1(g) where mTORC1 is a key player in the inhibition of Foxp3 at high antigen dose, while the early activation of Foxp3 by CD25/STAT5 pathway is an essential requirement for the differentiation of the naïve T cells into Treg cells at low antigen dose (Miskov-Zivanov, Turner, Kane, Morel, & Faeder, 2013a).

The SMLN scheme is deterministic as each state has only one possible next state. All variable values at time $t+1$ are computed using the values of their regulators at time t. The SMLN scheme computes steady-states that the system can reach, but it cannot account for the randomness of occurrence of events that is common for biological systems. In Figure 5-1(a), we show the simulation results for Scenarios I using the SMLN scheme. The T cell model is simulated 1000 times (i.e., 1000 runs), each run consisting of 50 steps, assuming random initial values for all elements, except for TCR, TGF-β and AKT which were selected to satisfy the requirements of each sub-scenario in Figure 6-1(g). It can be seen that the steady-state value of Foxp3 in sub-scenario I1 (high antigen dose) is lower than the steady-state value in sub-scenario I2 (low antigen dose), which was expected, according to (Miskov-Zivanov, Turner, et al., 2013a). However, the steady-state value of Foxp3 is not very low in sub-scenario I1 because the expression level of STAT5 and CD25 is high (activators of Foxp3), even with high expression level of mTORC1 (inhibitor of Foxp3). Also, we can see some oscillations in the transient response of Foxp3 in

Figure 6-1(a), which arise from the random initializations of the model elements enabling each simulation run to start from a different state, i.e., a different point in the state space. Additionally, these plots show that the SMLN scheme is useful for quickly identifying different steady-states that the system can reach. However, the SMLN scheme may not provide accurate trajectories for studying the transient response of the system, as it does not account for the stochasticity in the biological systems.

A more detailed analysis can be performed using RSQ scheme, which computes transient behavior of elements, and provides a better resolution of small changes occurring on element trajectories. As described in section 3.1.2, the RSQ scheme has two sub-types, round-based (RB-RSQ) and step-based (SB-RSQ) schemes. Here, we show simulation results using the RB-RSQ scheme for all sub-scenarios of Scenario I, where the value of each element is updated once per round according to the element's update rule. The results of our DiSH simulator are in agreement with those presented in (Miskov-Zivanov, Turner, et al., 2013a), that were obtained using the simulator described in (I. Albert et al., 2008). As shown in Figure 6-1(b), the steady-state value of Foxp3 is low (high) in scenario I1 (I2) which represents the high (low) antigen dose. In addition, Foxp3 exhibits a transient increase at high antigen dose (Scenario I1), due to the activation of CD25 and STAT5. This increase of the Foxp3 activation is quickly turned off because of the activation of mTORC1, which is a Foxp3 inhibitor. We can also see that initializing TGF-β at high level, with low antigen dose stimulation (Scenario I4) can provide a rapid increase in the Foxp3 expression, which inhibits any transient response of IL-2. The other sub-type of the RSQ simulation scheme is the SB-RSQ scheme which has also two sub-types, the uniform (USB) and non-uniform (NUSB) probability simulation schemes. The simulation results for Scenario I with the USB-RSQ scheme are shown in Figure 6-1(c). The simulation results in Figure 6-1(c) are

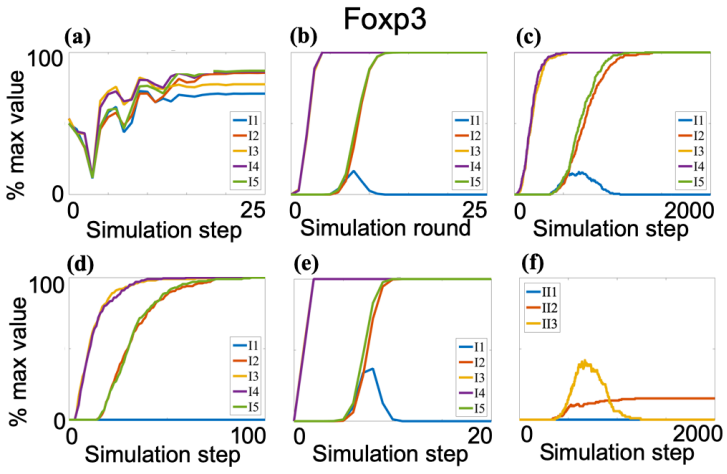
similar to the ones in Figure 6-1(b), where the RB-RSQ scheme was used, except that the transient responses shown in Figure 6-1(c) are delayed. This is happening due to the nature of the updating scheme. So, while each element gets updated once per round in the RB-RSQ scheme, only one element is updated in a step in the SB-RSQ scheme. Due to such updating schemes, the RB simulations will show faster rates of change than the SB ones. In the RB simulation, the number of time steps in each round is equal to the number of variables. In the T cell model, there are 61 variables, and in each of the 50 rounds, each variable is updated 50 times. On the other hand, in the step-based simulation, it may take more than 50 steps to update all elements because some elements may be updated several times within those 50 steps, while some other elements will not get updated. Figure 6-1(c) also emphasizes an interesting biological finding which was confirmed by the experimental results in (Miskov-Zivanov, Turner, et al., 2013a), suggesting that initializing TGF-β at high even with high antigen dose stimulation (Scenarios I3 and I4) will produce more Treg cells.

As described previously, in the NUSB-RSQ scheme, in each time step one variable is chosen for update according to the assigned update probabilities. When studying the T cell model using this scheme, we divide all the variables of the T-cell model into two blocks. Block A contains CD25, Foxp3 and IL2 variables, and has lower probability value, which is 0.1, and Block B, which contains the rest of the variables, is assigned probability 0.9. The blocks have been constructed using prior knowledge about the biological system – it is known that protein-protein interactions (Block B) occur at a faster speed than transcription reactions in Block A (such as transcription of the FOXP3 gene in the nucleus). Figure 6-1(d) shows that we get fast transient responses for Foxp3 because of the fast transient response of the elements in Block B (i.e. STAT5 and mTORC1),

which regulate Foxp3. However, the overall biological behavior of the system is almost the same as in the USB-RSQ scheme, with faster response since Block B elements are updated more often.

In the RKSQ simulation scheme, we can order the rules based on a priori knowledge about the sequence of the biological events. Here, we assign rank 1 for the rules that represent the cell membrane elements (e.g., TCR) and rank 2 for the elements that are regulated by the cell membrane elements, and we continue with the same procedure until we reach the last elements in the signaling pathway (e.g., gene transcriptions usually have the last rank). Figure 6-1(e) shows the simulation results for the sub-scenarios of Scenario I using the RKSQ scheme. The results for RKSQ are almost the same as the results shown in Figure 6-1(b), which were obtained using the RB-RSQ scheme, suggesting that the RKSQ scheme can be used if the information about the order of the biological events are known. Also, this shows that the RSQ scheme is able to capture the biological events even if the prior knowledge about the signaling events is not available.

Finally, we ran the simulator using USB-RSQ scheme for the three sub-scenarios II1, II2, and II3, when the TCR signal is turned off at simulation steps 100, 300, and 1500, respectively.



(g)

Scenario name	Description	Specific Conditions
I1	High Antigen dose	TCR_{HIGH} = True
I2	Low Antigen Dose	TCR_{LOW} = True
I3	High antigen dose and TGF-β is high	TCR_{HIGH}= True, TGF_{BETA} = True
I4	Low antigen dose and TGF-β is high	TCR_{LOW} = True, TGF_{BETA} = True
I5	High Antigen Dose and AKT is off	TCR_{HIGH} = True, AKT_{OFF} = True
II1	High antigen dose, toggle TCR_{HIGH}	Toggle at Step 100
II2	High antigen dose, toggle TCR_{HIGH}	Toggle at Step 300
II3	High antigen dose, toggle TCR_{HIGH}	Toggle at Step 1500

Figure 6-1 Trajectories for Foxp3 using different simulation schemes: a) SMLN, b) RB-RSQ, c) USB-RSQ, d) NUSB-RSQ, e) RKSQ, f) Toggling feature with USB-RSQ, and g) A list of simulation scenarios.

The simulation results in Figure 6-1(f) show that the time at which the TCR signal is turned off is critical for the T cell differentiation as confirmed by (Miskov-Zivanov, Turner, et al., 2013a). It can be seen that turning off the TCR signal at very early simulation step (e.g., Scenario II1) will lead to undifferentiated cells that are characterized by the low expression of both Foxp3 and IL-2. On the other hand, turning off the TCR signal at an intermediate step (e.g., Scenario II2) will cause the naïve T cells to be differentiated into Treg cells that are characterized by high expression of Foxp3. Additionally, turning off the TCR signal at later steps (e.g., Scenario II3) will produce more Th cells which are characterized by low expression of Foxp3. This behavior can be explained by looking at the trajectories of mTORC1 and CD25/STAT5 where the inhibition signal for Foxp3 through mTORC1 lasts longer when we remove the antigen dose at later simulation steps.

The SMLN and RSQ simulation schemes in DiSH enable analysis of both dynamic system behavior and its attractors. Deterministic discrete model simulations that assume simultaneous element update enable quick attractor analysis. However, when we have prior knowledge about faster and slower events, simultaneous simulations are not a good choice. In the discrete modeling approach, we can incorporate a priori knowledge about observed difference in event rates using delays and probabilistic simulation. Therefore, DiSH simulator allows scientists to look at biological systems from various perspectives, and learn more about the system by conducting multiple simulations under different conditions.

6.1.3 Effect of scoring functions

In this section, we study the effect of the scoring functions introduced in section 3.3 on the behavior of the original model in (Miskov-Zivanov, Turner, Kane, Morel, & Faeder, 2013a) by focusing on results for Foxp3 and IL-2 under different scenarios and conditions such as: multiple

discrete levels N, different initial values of PTEN, and different TCR signal strengths. Here, we assume that each simulation run represents one cell so that we can identify the heterogenous population by studying the values of Foxp3 and IL-2 at the end of each simulation run (i.e., the steady state value).

Figure 6-2 shows the steady state values of Foxp3 and IL-2 using the logical scoring (Figure 6-2(a)), arithmetic scoring (Figure 6-2(b)), and the difference between the steady state values of Foxp3 and IL-2 using the logical scoring minus the arithmetic scoring (Figure 6-2(c)) under different parameterizations (i.e. different N, initial PTEN values, and initial TCR signal strength) . It's shown in Figure 6-2(a) that the steady state value of Foxp3 is influenced by all parametrizations used in this work. Figure 6-2(a) shows that the activity level of Foxp3 is around 25% (indicating a heterogenous population) with low TCR levels. Specifically, Foxp3 is around 25% when the initial value of PTEN is equivalent to the initial value of TCR and both are initialized at low and medium levels (i.e. TCR and PTEN = 33%, 50%, and 67%) when N is 3 or 4. However, when N is greater than 4, Foxp3 stays at 0% with low TCR levels indicating wrong differentiation outcome. This happens because increasing N leads to a slow increase in the value of Foxp3 which prevents PTEN from increasing after reaching 0% due to the inhibition by TCR since S_A = *max*(PTEN, Foxp3) and S_I = TCR. If PTEN stays low, Foxp3 stays low as well (see Figure 5) and hence we see low levels of Foxp3 even with low TCR levels. When the initial value of TCR is 100% or it is larger than the initial value of PTEN (specifically when N = 4 and 5), the steady state value of Foxp3 reaches 0% which is similar to the case of high Ag dose in the original model (Miskov-Zivanov, Turner, Kane, Morel, & Faeder, 2013a). The values of IL-2 in Figure 6-2(a) are opposite to the values of Foxp3 which is expected due to the reciprocal relationship between Foxp3 and IL-2 (Miskov-Zivanov, Turner, Kane, Morel, & Faeder, 2013a).

Applying the arithmetic scoring with delays and weights as described in section 4.3 provides more cases where Foxp3 is not at either 0 or 100% which indicates a heterogenous population of Treg and Th cells. As seen in Figure 6-2(b), the steady state value of Foxp3 is decreasing with increasing TCR when PTEN is initialized at 100%. Foxp3 is at 0% when PTEN is less than 100% and N = 3 or 4. However, Foxp3 can reach a 100% when N is larger than 4 and TCR is very low (i.e. TCR = 20% and 25%) even when PTEN is less than 100%. Figure 6-2(c) shows that, in most cases, both logical and arithmetic scoring can obtain similar results. However, we can see some differences due to the added weights and delays with the arithmetic scoring.

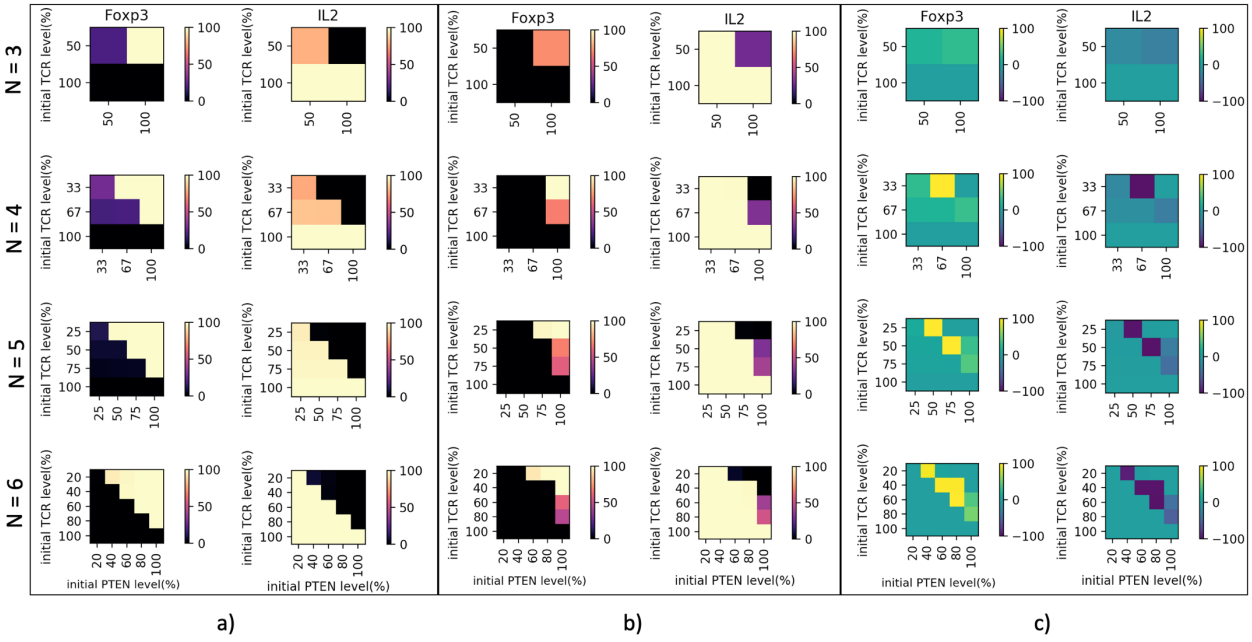


Figure 6-2 Steady state values of Foxp3 and IL-2 using a) logical scoring, b) arithmetic scoring, and c) the difference between the values using the logical scoring minus the arithmetic scoring.

In order to compare the simulation results of applying the logical and the arithmetic scoring with the experimental results, we show in Figure 6-3 the steady state values of Foxp3 at different discrete levels including N = 2 (i.e. the results from the original model) with PTEN initialized at 100% as in (Miskov-Zivanov, Turner, Kane, Morel, & Faeder, 2013a). The Foxp3 values obtained using the logical scoring (Figure 6-3(a)) are all at either 0% (when TCR = 100%) or a 100% (when TCR < 100%) which is similar to the simulation results of the original model. On the other hand, Figure 6-3(b) shows that the steady state value of Foxp3 decreases with increasing the TCR value at all values of N. The intermediate values of Foxp3 indicate that we get a heterogenous population at some cases when TCR is low which is more similar to the experimental results that are highlighted in Figure 6-3 by green and blue bars that represent the percentages of the Foxp3^{+} cells

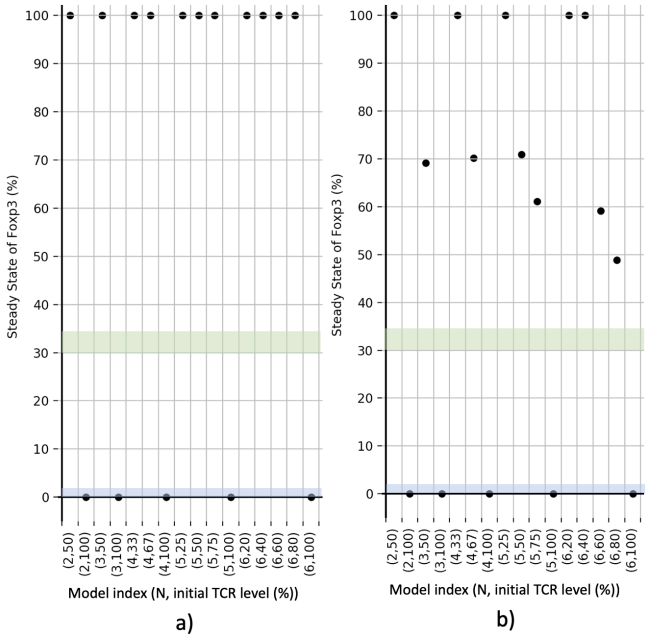


Figure 6-3 Steady state values of Foxp3 using a) the logical scoring, b) the arithmetic scoring with PTEN initialized at 100% and with different number of levels, N, and TCR values. The green (blue) highlighted range represents the percentage of Foxp3+ cells with low (high) antigen dose in the experimental results (Miskov-Zivanov, Turner, Kane, Morel, & Faeder, 2013a).

in the wet-lab experiment when applying low and high Ag doses, respectively (Miskov-Zivanov, Turner, Kane, Morel, & Faeder, 2013a).

Figure 6-4 shows simulation trajectories for Foxp3, IL-2, mTOR, and PTEN using the logical (Figure 6-4(a)) and the arithmetic scoring (Figure 6-4(b)) with N = 6, PTEN is initialized at 100%, and TCR is initialized at different levels. It is depicted in Figure 6-4(a) that PTEN stays high with all TCR levels except when the initial level of TCR is 100%. High PTEN levels prevent mTOR from going up which, in return, allows Foxp3 to increase and inhibit IL-2 indicating the differentiation into Treg cells. In contrast, when TCR is initialized at 100%, PTEN decreases to zero which allows mTOR to increase and inhibits Foxp3 and hence, IL-2 increases indicating the differentiation into Th cells. Applying the arithmetic scoring with delays and weights as described in section 4.3 shows that we can get heterogenous populations with TCR at intermediate levels such as 60 and 80%. Figure 6-4(b) shows how PTEN is decreased with increasing the TCR level which results in trajectories of mTOR at intermediate levels. Having mTOR at levels that are not either 0 or 100% allows Foxp3 to increase to medium levels indicating the differentiation into both Th and Treg cells which is closer to the experimental results that show a heterogenous population of Treg and Th cells under low Ag dose stimulation.

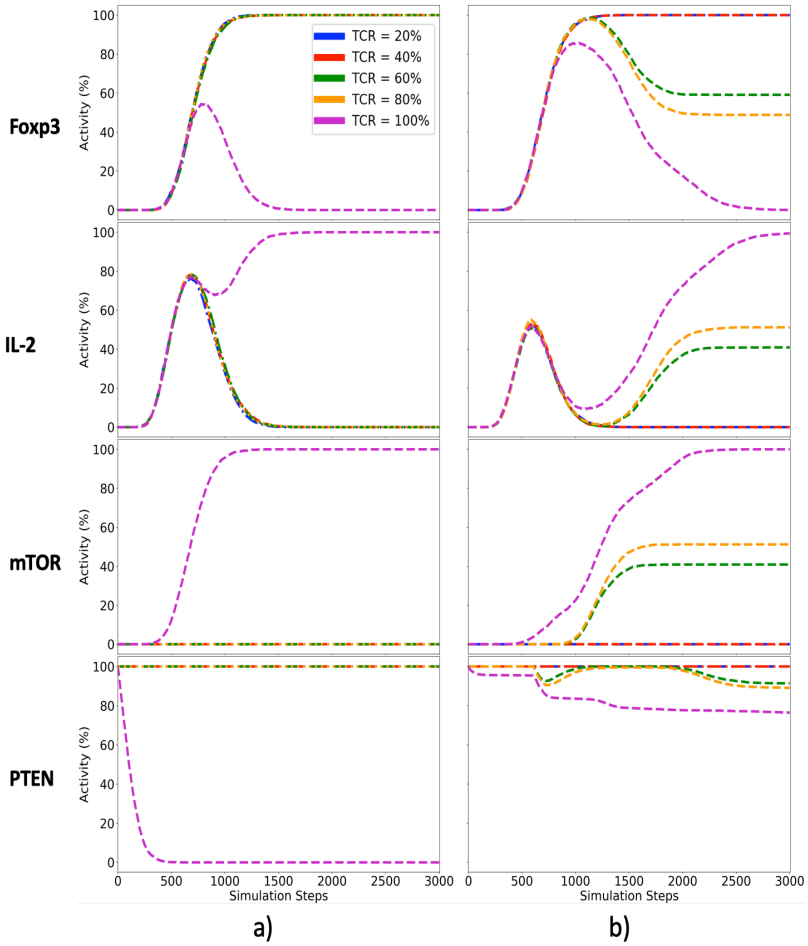


Figure 6-4 Trajectories of Foxp3, IL-2, mTOR, and PTEN using a) the logical scoring, b) the arithmetic scoring with N = 6, different initial levels of TCR, and PTEN is initialized at 100%.

Figure 6-4 shows how PTEN plays an important role in determining the differentiation outcome of the naïve cells by controlling mTOR which inhibits Foxp3. Therefore, in Figure 6-5, we show the complete pathway from PTEN to Foxp3 and its regulators (i.e. SMAD3, NFAT, and STAT5) as well as the simulation trajectories of PTEN, mTOR, Foxp3, SMAD3, NFAT, and STAT5 with N = 3, TCR = 50%, and initial PTEN levels = 50 and 100% to study the effect of changing the initial value of PTEN on response of Foxp3 and IL-2. The network diagram in Figure 6-5(a) shows the pathway from PTEN to Foxp3 as presented in (Miskov-Zivanov, Turner, Kane, Morel, & Faeder, 2013a) with the nodes that were initialized at levels greater than 0% highlighted

in red. The simulation trajectories of PTEN and Foxp3 show that we can get intermediate levels of Foxp3 at steady state in two cases; when PTEN is initialized at 50% with logical scoring, and when PTEN is initialized at 100% with arithmetic scoring. Initializing PTEN at 50% with TCR at 50% makes PTEN decreases when the logical scoring is applied because the inhibitors score (S_I = TCR) is equivalent to the activators score (S_A = PTEN + Foxp3, Foxp3 is initialized at 0) and the balancing behavior is set to "decrease". The decrease in PTEN is followed by an increase due to the increase in Foxp3 as a response to the signal from TCR and CD28 that propagate through the PI3K/PIP3/mTOR pathway. When PTEN is initialized at 100% with the logical scoring, it does not decrease since the inhibitors score is always less than the activators score.

On the other hand, initializing PTEN at 50% and applying the arithmetic scoring keeps PTEN at an intermediate level that can't inhibit mTOR and hence Foxp3 reaches 0% at steady state even with a low TCR level. Low Foxp3 expression with low TCR levels is not correct in terms of the experimental results that show high Foxp3 levels under low Ag dose stimulation (Miskov-Zivanov, Turner, Kane, Morel, & Faeder, 2013a). However, initializing PTEN at 100% and applying the arithmetic scoring shows more accurate results (i.e. heterogenous population) since Foxp3 reaches a medium level with low TCR value (i.e. TCR = 50%). The reason for having Foxp3 at a medium concentration is that the added weight to the inhibition signal from TCR to PTEN makes the inhibitors score ($S_I = w_{TCR-|PTEN}$*TCR) equivalent to the activators score (S_A = PTEN + Foxp3, Foxp3 is initialized at 0) at the beginning of the simulations which allows PTEN to decrease. It is also shown in Figures 6-5(e), (f), and (g) that the three activators of Foxp3 have the same response under the different initializations of PTEN and the two update methods (i.e. logical or arithmetic scoring), which shows that only mTOR can change the response of Foxp3.

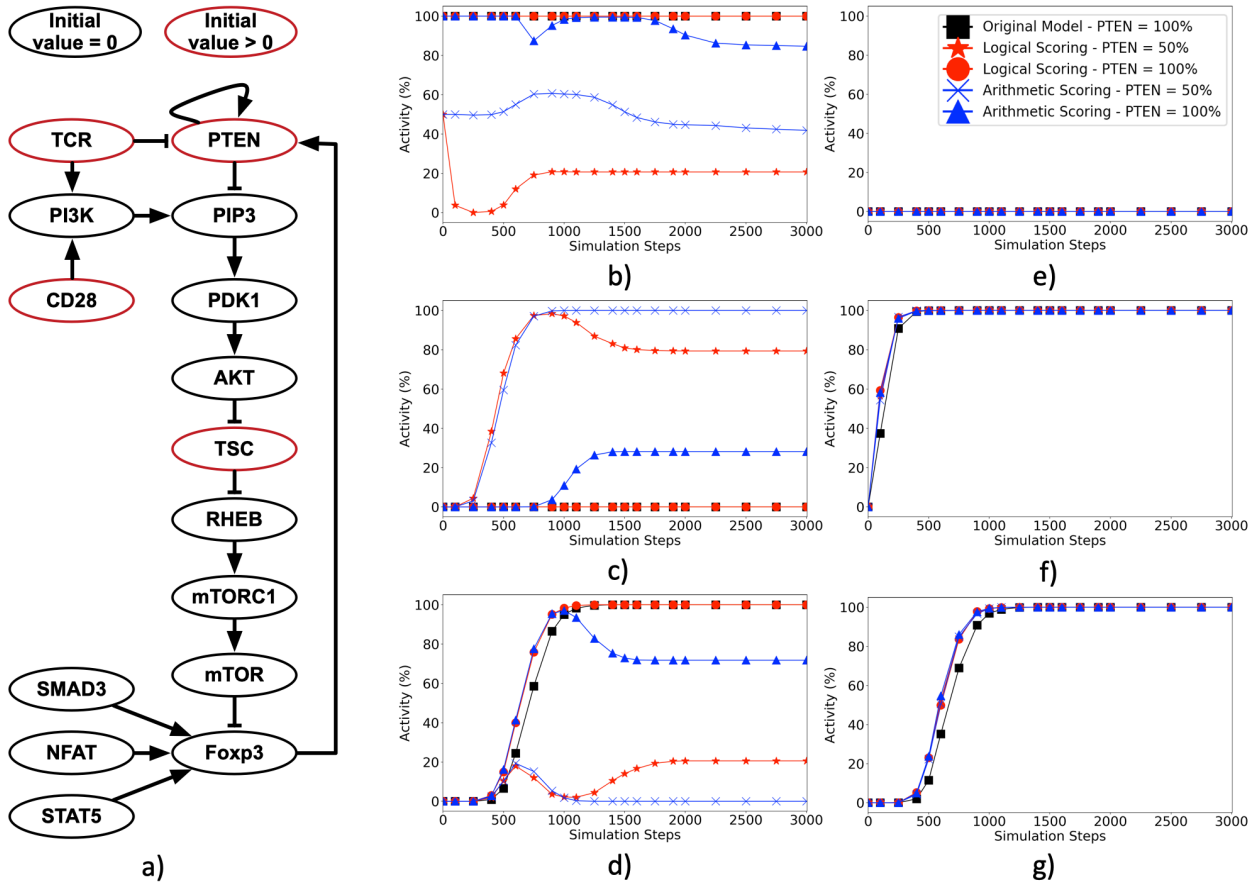
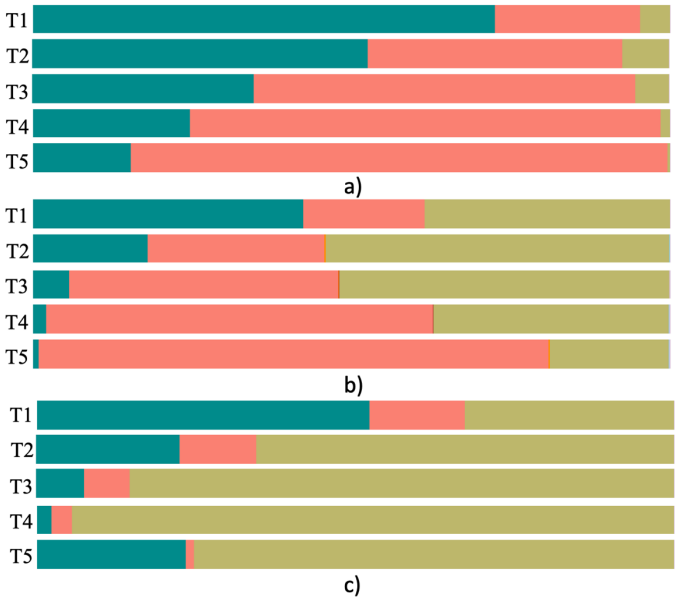


Figure 6-5 a) A network diagram of the upstream elements of Foxp3 as well as the simulation trajectories of b) PTEN, c) mTOR, d) Foxp3, e) SMAD3, f) NFAT, and g) STAT5 from the original model, using logical and arithmetic scoring with N = 3, PTEN initialized at different levels, and TCR is set to 50% (low dose).

Figures 6-3, 6-4, and 6-5 show that initializing PTEN at 100% and applying logical scoring with low levels of TCR failed to reproduce the experimental results that showed a heterogenous population of Treg, Th, and undifferentiated cells in the case of low Ag dose stimulation (Miskov-Zivanov, Turner, Kane, Morel, & Faeder, 2013a). Therefore, Miskov-Zivanov et. al. (Miskov-Zivanov, Turner, Kane, Morel, & Faeder, 2013a) studied the effect of varying, not only the strength of the Ag dose, but also the duration of a high Ag dose stimulation on the differentiation outcomes. The *in-silico* experiment in (Miskov-Zivanov, Turner, Kane, Morel, & Faeder, 2013a) was conducted by initializing TCR at 100% and toggling its value to 0% at different time points and

counting the number of Th, Treg, and undifferentiated cells at steady state. In this work, we conducted a similar experiment with N = 3, initial value of PTEN = 100%, and initial TCR value = 100% to study the effect of using the two update methods (i.e. logical and arithmetic scoring) on the steady state responses of key model elements when varying the duration of applying a specific stimulus such as the Ag dose. Figure 6-6 shows the frequency of the fixed-point attractors that we got when the level of TCR was toggled from 100% to 0% at time points, designated T1,…, T5: 300, 400, 500, 600, and 700 from (a) the original model, (b) using logical, and (c) arithmetic scoring methods. Since N = 3 and we are focusing on two model elements (i.e. Foxp3 and IL-2), we can get up to 9 different attractors as shown in Figure 6-6(d). The most frequent attractors were A0, A2, and A6 which correspond to undifferentiated cells, Th cells, and Treg cells, respectively.

It is also shown in Figure 6-6(a), which shows results of toggling TCR in the original model, that toggling the TCR value at later steps favors attractor A2 which corresponds to the differentiation into Th cells (i.e. IL-2 = 100% and Foxp3 = 0% at steady state) and shows that we can get a heterogenous population of cells if a strong TCR signal is applied but doesn't last for a sufficient amount of time. A similar trend is seen Figure 6-6(b) when the logical scoring is applied but with more frequencies of attractor A6 and less frequencies of attractor A0. This happens because increasing the number of discrete levels, N, to 3 means that PTEN takes more time to decrease from 100% to 0% when TCR is still high. The delayed decrease of PTEN allows less Th cells to develop since Foxp3 will have more time to increase which enhances the chance of PTEN to recover after toggling TCR to zero (TCR is a PTEN inhibitor). Similarly, Figure 6-6(c) shows that applying the arithmetic scoring with delayed inhibition of PTEN gives even less time for Th cells to develop and increases the chance of PTEN recovery. Therefore, when more time is given to TCR to stay high such as toggling the TCR value at T5 in Figure 6-6(c), we can see more



Attractor	Foxp3	IL-2	Frequency of Attractors at T3 (%)		
			Original Model	Logical Scoring	Arithmetic Scoring
A0			35	5.9	7.7
A1			0	0	0
A2			59.8	42.1	7.2
A3			0	0.1	0
A4			0	0	0
A5			0	0.1	0
A6			5.2	51.7	85.1
A7			0	0	0
A8			0	0.1	0

Steady State Level (%)	0	50	100
Color Code			

d)

Figure 6-6 Percentage of attractors with antigen dose removal at time steps T1:T5 as a) in the original model, b) using the logical scoring, c) using the arithmetic scoring, d) attractors frequencies when the antigen dose is removed at T3.

undifferentiated cells (i.e. A0) that can be differentiated into Th cells if TCR stays high for a longer time.

The first version of DiSH-simulator has been introduced as a tool for simulating logical models of biological systems allowing modelers to run in-silico simulations using many update schemes as well as integrating information from automated literature reading into models using model extension methods such as the genetic algorithm (Sayed et al., 2018; Sayed, Telmer, et al., 2017). In this section, we introduced and new version of DiSH simulator that expand its capabilities that are described in sections 3.1 and 3.2 by allowing modelers to assign as many discrete levels as needed to model elements and run simulations using two update methods; logical and arithmetic scoring. The results of simulating the naïve T cell differentiation model by using both scoring methods showed that we can get, not only results similar to those obtained by

simulating the original model in (Miskov-Zivanov, Turner, Kane, Morel, & Faeder, 2013a), but showed also that we can recapitulate, more accurately, the experimental results which included a heterogenous population of Treg, Th, and undifferentiated cells under low Ag doses stimulation.

Although both scoring methods along with the other introduced methods (i.e. staircase update function, spontaneous and balancing behavior, normalization, and delays) were capable of recapitulating the simulation results in (Miskov-Zivanov, Turner, Kane, Morel, & Faeder, 2013a) in addition to capturing more dynamics such as the heterogenous population, there are key differences between the logical and arithmetic scoring methods. In logical scoring, the *min* and *max* functions can have a huge influence on the regulated element if one of its regulators is at 0 or N-1. For example, if an element, x, has three ANDed activators, the activators score, S_A, will always be 0 if any one of the activators is 0 no matter what the values of the other two activators are. Similarly, if the three activators are ORed and only one of them is at N-1, the value of S_A will always be N-1 even if the other two activators are at 0. In these situations, only one regulator can influence the overall update rule which, in some cases, not preferable and limits the ability of modelers to combine the effect of multiple regulators. On the other hand, the arithmetic scoring considers all the regulators' values and combines their effect according to the assigned weights which gives modelers more flexibility and control over their models. However, choosing the correct weights requires more knowledge about the system in addition to going through multiple iterations of model simulations and evaluations trying to find the best set of parameters.

Choosing the number of discrete levels, N, can also affect the simulation results as shown in Figure 6-2(a) where Foxp3 reaches 0% at steady state when N is greater than 4 even with low TCR levels which is the opposite to the results of the original model in (Miskov-Zivanov, Turner, Kane, Morel, & Faeder, 2013a). Increasing N means that the element will take more time (i.e.

simulation steps) to change from 0% to 100% or vice versa because it has to go to the intermediate levels first which increases the chance of receiving an interruption signal that makes the element stay at an intermediate level or to switch back to its original level. For example, if an element *x* has 6 discrete levels and it is initialized at level 0, then the update rule has to be selected 5 times before *x* reaches level N-1 assuming that the activators score is larger than the inhibitors score during the whole time. If we run the USB-RSQ simulation scheme, then the selection of the same update function 5 times requires, on average, *5*k* simulation steps where *k* is the total number of elements in the model. During these simulation steps, the inhibitors score of *x* can increase to be more than or equal to the activators score forcing *x* to stop increasing or it could make it decrease according to the specified rules of the balancing and spontaneous behaviors. The effect of increasing N is shown in Figure 6-6 where more Treg cells are developed when the logical and arithmetic scoring are used with N = 3 compared to the results of the original model with N = 2. Increasing N for PTEN from 2 to 3 makes PTEN take more time to decrease which is reflected on Foxp3 as a delayed inhibition that changes the differentiation outcome.

The results of this work indicate also that the structure of the network and the logical relationships between the system components are not the only factors that determine the transient and the steady state responses of the elements of interest, but also the model parameterizations and the update method. The results show that we can still get a heterogenous population of cells by changing the number of discrete levels and the initial values of some of the key elements such as PTEN even without changing the structure of the network. Having many parameterizations to change and multiple update methods to choose from provides more degrees of freedom for modelers to build accurate models even when they are restricted to change the model structure due to a priori knowledge that prevents them from changing the relationship between model elements.

These degrees of freedom include also deciding whether to apply the spontaneous behavior or not and what balancing behavior is suitable for each model element. For example, if the value of a model element is not supposed to increase after being degraded due to an inhibition signal, a modeler can shut its spontaneous behavior off. Similarly, having the ability to decide whether the value of the element increases, decreases, or stay the same when its activators and inhibitors scores are balanced is useful when there is no information that favors one type of regulation over the other and gives modelers more control over their models which allows them to test multiple options in order to get the most accurate results.

6.1.4 Effect of time modeling

The extended naïve T cell differentiation model in (Hawse et al., 2015; Miskov-Zivanov, Turner, et al., 2013b) is used to study the effect of incorporating timing information into discrete models on the performance of the model in terms of the percentage of cells with high Foxp3

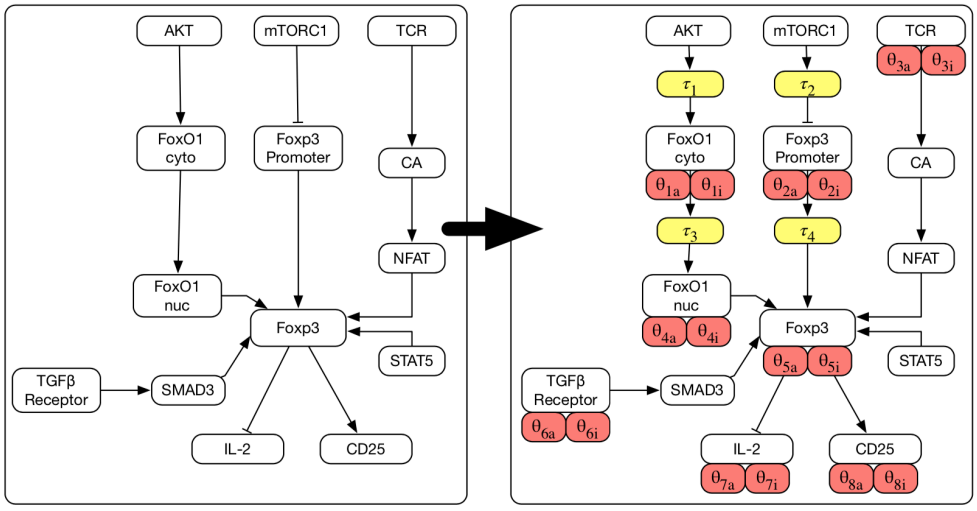


Figure 6-7 A subnetwork of the extended naive T cell differentiation model before (left) and after (right) assigning propagation and regulation delays.

expression. Here, we rebuilt the model in (Hawse et al., 2015) with representing the common motifs as described in sections 4.2.4, 4.2.5, and 4.2.6 in order to follow a unified way for building models that contain gene expression, receptor activation, and translocation motifs. Figure 6-7 shows the main pathways in the extended model that needed delays in order to get simulation trajectories similar to the experimental results. The propagation and regulation delays shown in Figure 6-7 were determined manually with multiple iterations of simulations and comparisons between the simulation trajectories and the experimental data. Since all model variables are

Table 6-1 Values of delay variables in Figure 6-7.

	Delay variable	Number of delay steps
Propagation delays	τ_1	5
	τ_2	8
	τ_3	5
	τ_4	6
Regulation delays	$\{\theta_{1a}, \theta_{1i}\}$	$\{2,10\}$
	$\{\theta_{2a}, \theta_{2i}\}$	$\{2,3\}$
	$\{\theta_{3a}, \theta_{3i}\}$	$\{2,2\}$
	$\{\theta_{4a}, \theta_{4i}\}$	$\{0,10\}$
	$\{\theta_{5a}, \theta_{5i}\}$	$\{3,2\}$
	$\{\theta_{6a}, \theta_{6i}\}$	$\{2,1\}$
	$\{\theta_{7a}, \theta_{7i}\}$	$\{2,1\}$
	$\{\theta_{8a}, \theta_{8i}\}$	$\{3,2\}$

Boolean, the regulation delays are represented by two variables, θ_a and θ_i which represent the activation and inhibition delays respectively. Table 6-1 shows the values of the delay variables that are added manually to the model.

Following the steps-to-time mapping procedure in section 5.4.1, we assumed that each simulation step corresponds to 3.36 hours and hence, the total number of simulation steps needed to provide results for 7 days is calculated by Equation 5-10 to be 50 steps. Leveraging the process of adding delays to the extended model, we utilized the Random-Delay SMLN update scheme to simulate the model because it provides accurate transient and steady state responses with less simulation steps and less simulation time as depicted in Figure 6-8 which shows the Foxp3 response when the extended model is simulated under the simulation scenarios presented in Figure 6-1(g). Figure 6-8 shows that the steady state value of Foxp3 under low Ag dose (Scenario I1) is more closer to the steady state value of Foxp3 in the experimental results (i.e. ~33%). It also shows that the percentage of cells with high Foxp3 reaches 0% at steady state with high Ag dose (Scenario I2) which is equivalent to the experimental results. The simulation results in Figure 6-8 show also that a 100% of the cells will be differentiated into Treg cells with high Foxp3 if TGFβ, which is secreted by cancer cells, is present under either high or low Ag dose (Scenarios I3 and I4)

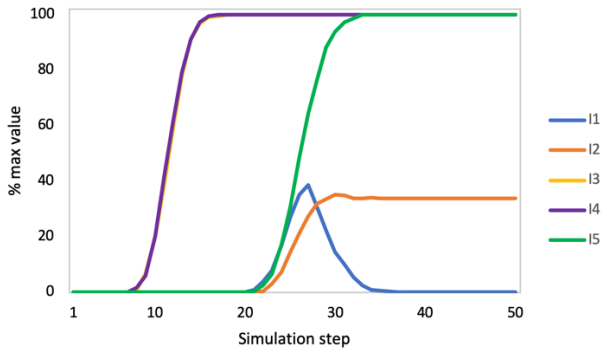


Figure 6-8 Foxp3 trajectories under the simulation scenarios shown in Figure 6-1(g) usnig the Random-Delay SMLN update scheme for the extended model.

suggesting that the cancer cells will shut down the immune response even with high Ag stimulus. The results of Scenario I5 also confirms the findings in (Miskov-Zivanov, Turner, Kane, Morel, & Faeder, 2013a) where inhibiting Akt will lead to the differentiation into Treg cells even with high Ag dose stimulation which means that the Random-Delay SMLN scheme is efficient in providing accurate simulation trajectories with less simulation steps and less simulation time. The trajectories in Figure 6-8 are obtained by taking the average over a 1000 simulation runs. The simulation time of simulating one scenario for 1000 runs is about 2.5 min when the SMLN or the Random-Delay SMLN update schemes are utilized while the simulation time of one scenario can take up to 10 min when the RSQ update scheme is utilized.

In order to evaluate the effect of the added delays on the performance of the model, we compare the trajectories of Foxp3 obtained by the original model in (Miskov-Zivanov, Turner, Kane, Morel, & Faeder, 2013a), the extended model in (Hawse et al., 2015), the extended model with motifs, and the experimental results with trajectories obtained by the extended model with delays. Figure 6-9 shows that both the extended model and the extended model with motifs provide better results than the original model. The model with motifs provides a slightly better trajectory than the extended model without motifs because motifs modeling adds delays to the genes,

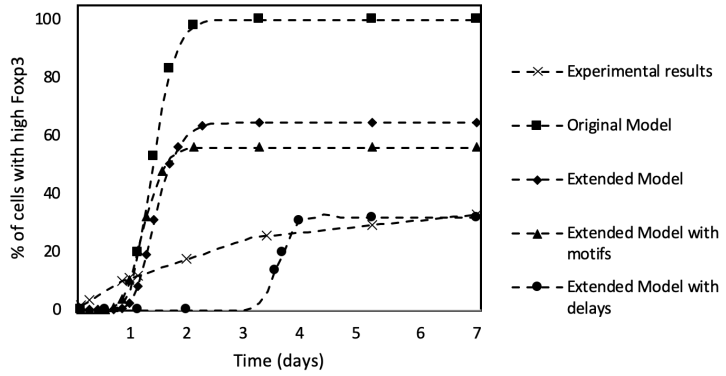


Figure 6-9 A comparison between Foxp3 trajectories obtained by different models with the experimental results under low Ag dose.

receptors and translocated elements as discussed in (Sayed et al., 2016). The extended model with delays provides the most accurate Foxp3 trajectory with steady state value equivalent to the percentage of $Foxp3^{+}$ cells in the experimental results after 7 days of the low Ag dose stimulation as shown in Figure 6-9. These results suggest that the propagation and regulation delays are efficient tools for incorporating timing information into discrete models of complex systems when time series data is available to calibrate the model.

Comparing the performance of the propagation and regulation delays method to the buffer insertion method that was used in (Miskov-Zivanov, Turner, Kane, Morel, & Faeder, 2013a) shows that the proposed delay method has a fixed simulation time when different delay steps are added to many percentages of nodes and interactions in the model. Figure 6-10 shows how the simulation time changes with respect to the percentage of nodes and interactions that have delays as well as the number of delay steps added. We chose 10%, 20%, and 30% of the nodes and interactions in the extended model at random and assigned 2, 4, 6, 8, and 10 delay steps to compare the effect of the buffer insertion method and the proposed delay method on the simulation time. The simulation times shown in Figure 6-10 are obtained by running the Random-Delay SMLN scheme with 50 simulation steps and 1000 simulation runs. Figure 6-10 shows that the simulation time increases with both the number of added delay steps and the number of nodes and interactions that have delays if the buffer insertion method was used while the simulation time is almost fixed when the proposed delay method is applied (i.e. the propagation and regulation delays). The reason for the increase in simulation time when the buffer insertion method is utilized is that we have to create extra nodes in order to delay the signal from one node to another while the proposed method offers delaying the signals without creating any extra nodes which makes the model size smaller as shown in Figure 6-11. Figure 6-11 shows how the numbers of variables and interactions increase with

increasing the number of assigned delay steps when the buffer insertion method is applied and shows that the model size is fixed when the proposed method is used.

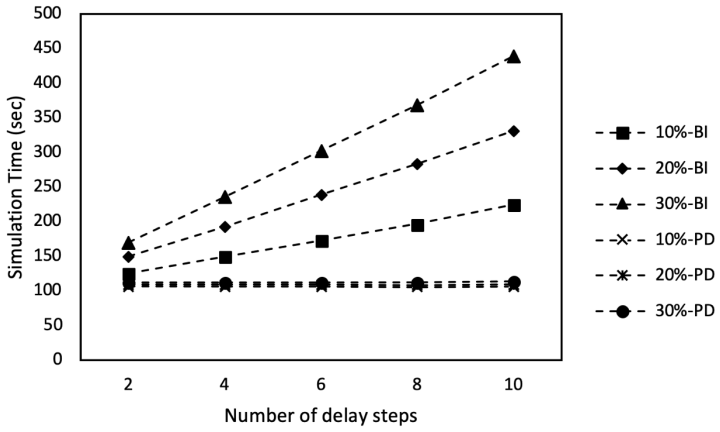


Figure 6-10 A comparison between the simulation time of running the extended model using the Buffer Insertion (BI) method and the Propagation Delay method (PD) when a different amount of delay steps (x-axis) are added to different percentages of model elements and interactions.

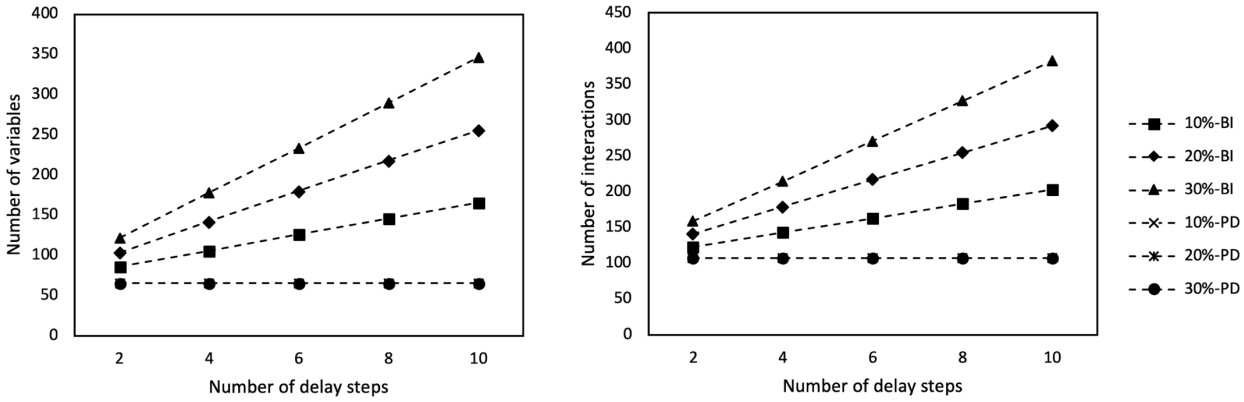


Figure 6-11 Change in naïve T cell model size in terms of a) number of variables b) number of interactions at different amounts of delay steps that are added to different percentages of model elements and interactions.

As mentioned in sections 5.4, the timing parameters (i.e. the propagation and regulation delays) could be inferred using the Nelder-Mead optimization algorithm when time series data are available. The extended naïve T cell model contains 66 Boolean variables and 107 interactions which means that the optimization algorithm has to find optimal values for 346 timing parameters if we don't specify a smaller set to focus on. Here, we focus on the set of parameters shown in Figure 6-7 which were obtained manually and run the Nelder-Mead algorithm in order to see if there is a better set of values that can decrease the error between the experimental results and the simulation trajectories of Foxp3. The cost function was defined by Equation 5-8 as the difference between the interpolated Foxp3 experimental data and the simulation trajectories under low and high Ag doses. Figure 6-12 shows the trajectories of Foxp3 under low Ag dose stimulation using the manually added delays and the optimized delay values as well as the experimental results. As seen in Figure 6-12, the steady state values of both manual and optimized delays are similar while the transient response is different. The transient response of the model with optimized delays is less delayed and closer to the transient response of the experimental results. This suggests that some of the manual delays were not necessary and we could have used a less amount of delays. Table 6-2 shows the actual delay steps that were added manually and the updated value using the optimization algorithm.

Although the optimization algorithm didn't change the values of the timing parameters drastically as shown in Table 6-2, it provided better transient response for Foxp3 which enhances the performance of the model by decreasing the error between the experimental results and the simulation trajectories. It also confirms that the manually added delays were good enough to start with which increases the modeler's confidence in the values chosen for the timing parameters.

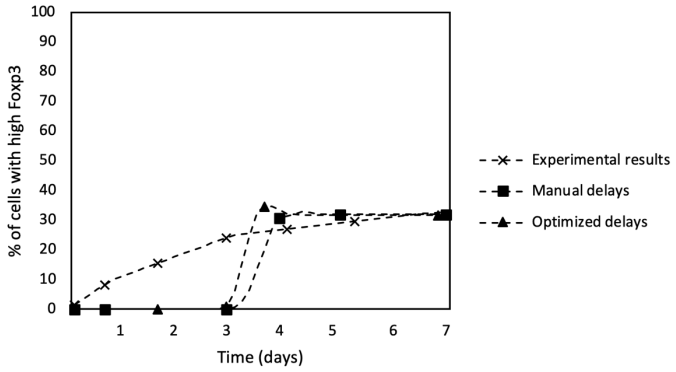


Figure 6-12 Foxp3 trajectories when the extended model is simulated with the manually defined delays and the delays found by the optimization algorithm compared to the experimental data.

Table 6-2 Values of delay variables in Figure 6-7 before and after applying the optimization algorithm.

	Delay variable	Manual delay steps	Optimized delay steps
Propagation delays	τ_1	5	**4**
	τ_2	8	8
	τ_3	5	5
	τ_4	6	6
Regulation delays	$\{\theta_{1a}, \theta_{1i}\}$	{2,10}	{2,10}
	$\{\theta_{2a}, \theta_{2i}\}$	{2,3}	{2,3}
	$\{\theta_{3a}, \theta_{3i}\}$	{2,2}	**{1,1}**
	$\{\theta_{4a}, \theta_{4i}\}$	{0,10}	{0,10}
	$\{\theta_{5a}, \theta_{5i}\}$	{3,2}	**{1,0}**
	$\{\theta_{6a}, \theta_{6i}\}$	{2,1}	{2,1}
	$\{\theta_{7a}, \theta_{7i}\}$	{2,1}	**{1,1}**
	$\{\theta_{8a}, \theta_{8i}\}$	{3,2}	**{3,1}**

6.2 Buddying yeast cell cycle

The budding yeast *Saccharomyces cerevisiae* is an organism that is commonly used as a biological model for studying the eukaryotic cell cycle. The first phase in the eukaryotic cell cycle is S phase where each chromosome in the DNA is replicated into two identical chromatids in a process called DNA synthesis. After DNA synthesis, the cell enters M phase or mitosis where replicated chromosomes are separated into two nuclei before the cell divides into two new cells. The eukaryotic cell cycle can be considered as two phases process; S phase and M phase which are separated by two gap phases G1 and G2. Features such as cell shape, size, and DNA damage are considered as checkpoints that are built into the regulatory process to ensure successful cell cycle process (Bähler, 2005). The *S. cerevisiae* cell grows slightly different from other eukaryotes during cell cycle where a bud forms from the side of the cell to hold one pole of the mitotic spindle. After cell division (CD), the bud becomes the daughter cell with one copy of the DNA.

Mathematical models have been developed in the past to study the *S. cerevisiae* cell cycle using different approaches such as ordinary differential equations (ODEs) and logical modeling (Chen et al., 2004; Chen et al., 2000; Irons, 2009; F. Li, Long, Lu, Ouyang, & Tang, 2004). Irons (Irons, 2009) built a logical model to summarize the existing knowledge about the buddying yeast cell cycle and to introduce a method for incorporating timing information into logical models with Boolean variables. The time modeling method described in (Irons, 2009) is called “dummy nodes” and it has been described in section 2.2.2. Here, we rebuild Irons’ model using our modeling framework and use DiSH to simulate it. We also compare our time modeling methods to the dummy nodes method in terms of model size and simulation time. Since the delays in the model in (Irons, 2009) are assigned under different regulation conditions, we apply the Conditioned State Transition technique described in section 5.1.2.

6.2.1 Model Description

The *S. cerevisiae* cell cycle model in (Irons, 2009) is a logical model with 18 Boolean variables. The four cell cycle events “DNA synthesis”, “Bud growth”, “entry into Mitosis”, and “cell division” are refered to as “S”, “B”, “M”, and “CD” in the model respectively. The transition from the gap phase (G1) to S phase begins with cyclin Cln3 activating Cln2, Clb5 through SMBF and inhibiting cyclin-dependent kinase inhibitor (CKI) through the SMBF/Clb5 pathway. The transition to S phase promotes the DNA synthesis and Bud formation processes. Following Bud formation, Clb2 becomes active through B and SFF leading to the transition into M phase which ends with the activation of Cdc14. The activation of Cdc14 triggers a number of interactions such as activating Cdh1, Swi5, and CKI which results in cell division (CD) and preparing the cell for another cycle of G1 phase. Figure 6-13 shows a newtork diagram of the model. A full description of the main interactions in the model are presented in (Irons, 2009).

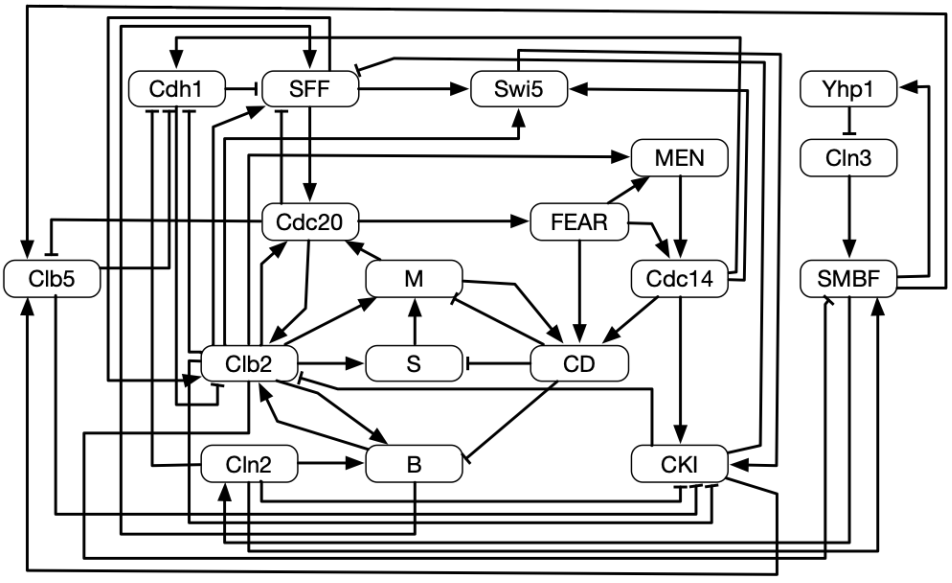


Figure 6-13 A network diagram of the buddying yeast cell cycle model in (Irons, 2009).

Since the model contains different kinds of biological interactions and processes, the author in (Irons, 2009) assigned different delay steps to each interaction and event. For example, all protein-protein interactions, phosphorylation, and dephosphorylation interactions were assumend to have 0 delay steps while interactions corresponding to transcrption, degrdation, and cell cycle events are assumed to have varying delay values. In Irons' model, all transcriptional activations are assumed to have 1 delay step except for a few elements. For example, the activation of Clb5 takes 3 delay steps when CKI is present because Clb5 and CKI mutually inhibit each other and it takes more time for Clb5 to build up and become active. The activation delay of the cell cycle events S, M, and CD are assumed to have 1 delay step while the Bud formation event is assumed to have 5 delay steps. The value of the assigned delays are different under different regulation conditions as shown in Table 6-3 which includes all elements that have delays and the corresponding regulation conditions. The rest of the update rules are fully described in (Irons, 2009).

6.2.2 Model Simulations

For the purpose of conducting a fair comparison, the simultaneous (SMLN) update scheme is used to simulate the buddying yeast cell cycle model because it is the scheme used in (Irons, 2009). We also compare the attractors of the model when the Reset and the No-Reset Delays are utilized as explained in section 5.1.2. Figure 6-14 shows the cyclic attractors obtained by the No-Reset (part a) and Reset (part b) delay methods where white/black represent active/inactive state. Each row in Figure 6-14 represnts an attractor state and each column shows the value of the corresponding element in a specific attractor. The attractors in Figure 6-14 are cyclic which means

Table 6-3 Conditions for activation of model elements with corresponding activation and inhibition delays.

Variable name	Activation Condition	Activation delay steps	Inhibition delay steps
Cln3	Yhp1 = 0	1	0
Cln2	SMBF = 1	1	0
Clb5	Cdc20 = 0 **AND** CK1 = 1 **AND** SMBF = 1	3	0
	Cdc20 = 0 **AND** CK1 = 0 **AND** SMBF = 1	1	2
Yhp1	SMBF = 1	1	5
Cdc20	M = 1 **AND** Clb2 = 1 **AND** SFF = 1	1	0
Swi5	(Clb2 = 0 **AND** SFF = 1)	1	2
	(Cdc14 = 0 **AND** SFF = 1)	1	2
CKI	(Cdc14 = 1 **AND** Swi = 1)	1	0
	(Cln2 = 0 **AND** Clb5 = 0 **AND** Clb2 = 0 **AND** Swi = 1)	1	0
S	CD = 0 **AND** (Clb5 = 1 **OR** Clb2 = 1)	1	0
B	CD = 0 **AND** (Cln2 = 1 **OR** Clb5 = 1)	5	0
M	CD = 0 **AND** S = 1 **AND** Clb2 = 1	1	0
CD	M = 1 **AND** FEAR = 1 **AND** Cdc14 = 1	1	0

that the attractor states will be repeated if the model is initialized at attractor state A0 (i.e. simulation step #0) and reaches attractor state A18 (with No-Reset delays) or A21 (with Reset delays). These cyclic attractors represent the budding yeast cell cycle which starts with S phase (DNA synthesis) and go through B (budding), M (mitosis), and CD (cell division) phases before the new cell goes through the same cycle. Figure 6-14 shows similar attractor to those reported in (Irons, 2009) which means that the developed methodologies were capable of recapitulating the results of the original model. Figure 6-14(a) shows that the No-Reset delay method is equivalent to the dummy nodes method in terms of the similarities between the attaractors (both generate 19 attractors (A0→ A18) and same changes happen at the same simulation steps). The differences between the No-Reset and Reset methods are shown in Figure 6-14 where the Reset delays method shows more delayed attractors than those obtained with the No-Reset delays (21 attractors vs. 18 attractors). These differences are due to the delayed activation of Clb5 which takes one more step to be activated with the Reset delay method. The assigned activation delay to Clb5 is 3 steps when

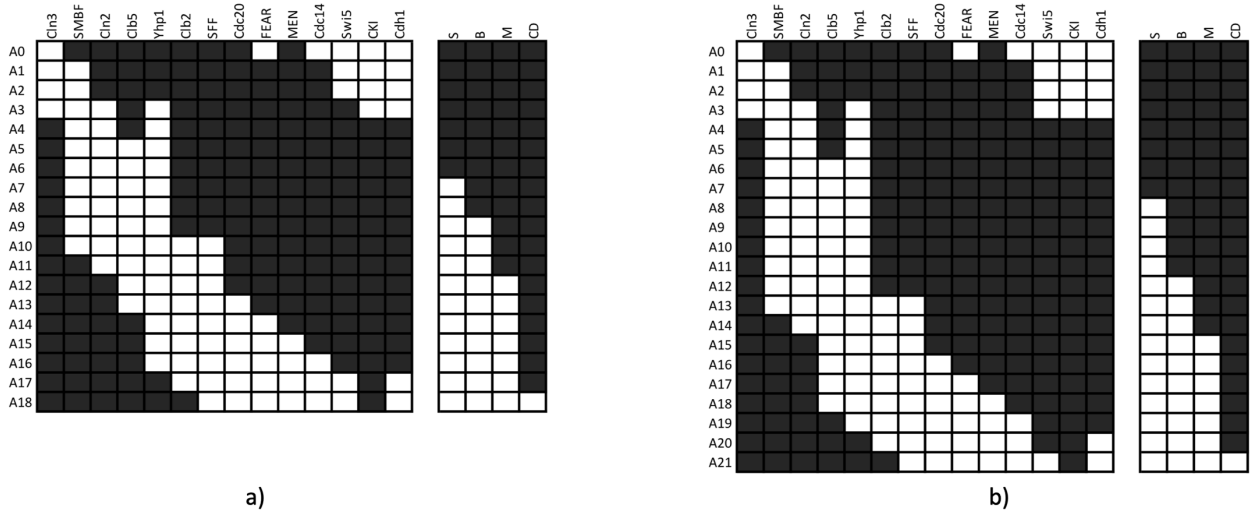


Figure 6-14 Cyclic attractors of the budding yeast cell cycel model. a) Attractors obtained with the No-Reset delay method, b) attractors obtained with the Reset delays method. Each row represents an attractor state where white/black boxes represent an active/inactive state of each model element.

CKI = 1, SMBF = 1, and Cdc20 = 0 and it is only one step when CKI = 0, SMBF = 1, and Cdc20 = 0 as shown in Table 6-3. Therefore, with the No-Reset delay method, Clb5 changes to 1 after three steps following the change of SMBF from 0 to 1. However, Clb5 needs one more step with the Reset delay method because CKI changes to 0 at step # 4 (A4) which resets the delay variable.

Although the Reset and No-Reset delay methods were able to reacpitualte the results of the dummy node method, there is a significant decrease in the simulation time and the model size when the Reset and No-Reset methods are used. Figure 6-15 shows how the simulation time increases when the dummy nodes method is used to add different delay steps (e.g. 2, 4, …, 10) to different percentages of model elements (e.g. 10%, 20%, and 30%) while the simulation time is almost the same when the proposed delay methods are used. Figure 6-16 shows that the number of variables and number of interactions do not change with the proposed delay methods while these numbers increase significantly with the dummy nodes method.

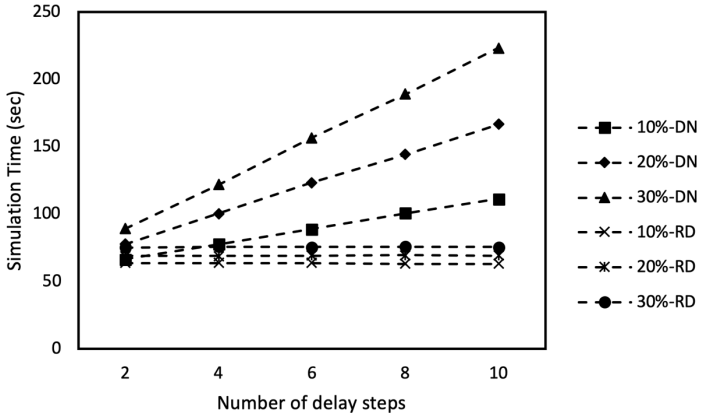


Figure 6-15 A comparison between the simulation time of the budding yeast cell cycle model using the Dummy Nodes (DN) method and the Regulation Delay method (RD) when a different amount of delay steps (x-axis) are added to different percentages of model elements.

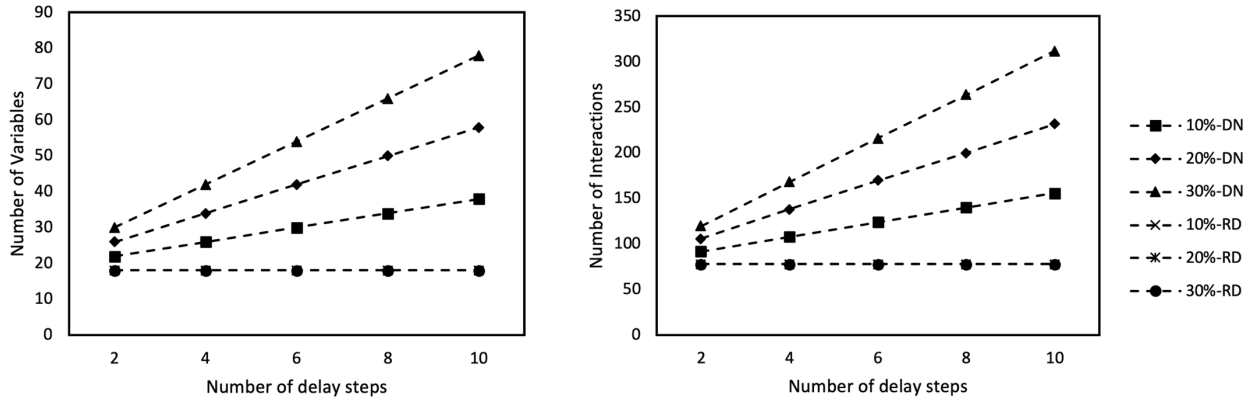


Figure 6-16 Change in the budding yeast model size in terms of a) number of variables b) number of interactions at different amounts of delay steps that are added to different percentages of model elements and interactions.

In addition to the decrease in the simulation time and the model size, the introduced methods in this work are easier to implement and can be easily integrated into an automated framework for model construction than the dummy nodes method which requires a human intervention to create a complex network for the added delays, as shown in section 2.2.2. The need for a human to build a complex network of nodes and developing complex update functions for each node is error-prone and time consuming.

6.3 Food security in South Sudan

South Sudan is an African country that received its independence from Sudan on July 9, 2011. Following its independence, a civil conflict erupted in December 2013 as a result of power struggle. Civil conflict affected mostly rural areas and led to sever agriculture damage, disruption in food production, plundering of crops and livestock, loss of life, health, assets, and income which caused acute food insecurity and malnutrition (FAO, 2016). In this section, we introduce a discrete model that was built to study the factors leading to food insecurity in South Sudan using automated literature reading. An automated reading engine such as Eidos (Sharp et al., 2019) is utilized to extract events from published literature that describe the situation in South Sudan and to build an initial version of the model. The automatically created model is then refined by removing the wrong interactions, changing the update rules, and adding interactions manually in order to make the simulation results match historical data. The historical data that are used to calibrate the model are indicators for conflict, inflation, crop yield, rainfall, and refugees population that were collected from international organizations websites such as Food and Agriculture Organization of the United Nations (FAO), World Bank, International Monetary Fund (IMF), The Armed Conflict Location & Event Data Project (ACLED, 2019), and World Health Organization (WHO).

6.3.1 Model Description

The discrete model of the food security in South Sudan contains 38 nodes and 91 interactions. Four nodes in the model are inputs (i.e. nodes without regulators) which are rainfall, drought, fertile soil, and assistance. The input nodes can be used to test different scenarios such as predicting what will happen if international organizations provide assistance to the people of South

Sudan and/or how refugees population will change if conflict is decreased. We also used the historical data of 4 nodes which are crop yield, conflict, inflation, and diseases (i.e. infectious diseases such as HIV and malaria) to calibrate the model. The simulations were assumed to represent the situation in Jonglei region during the period from January 2014 to December 2018 and hence, all the historical data were collected during that time period with different resolutions (the data were collected daily, monthly or yearly). The data were discretized and normalized as described in section 5.4.1 and the discrete values corresponding to January 2014 were used as initial values for all the nodes that we have data for except rainfall. The discretized rainfall data (collected from (CHIRPS, 2019)) is used as input during the whole time period and not just the initial value.

Figure 6-17 includes a network diagram of the model and shows that the direct positive regulators for food security are market supply, food availability, assistance and road accessibility while the direct negative regulators are physical insecurity, trade disruption, and food price. Trade disruption is caused by conflict and road accessibility is limited by conflict as well which means that food security will decrease if conflict erupts in the region. Conflict also indirectly increases physical insecurity and food price which leads to a decreased food security. Therefore, food security can be restored if conflict is contained and foreign assistance is introduced. In the following sections, we study the effect of introducing assistance and decreasing conflict at the beginning of 2018 on food security and the number of refugees in South Sudan as well as studying the effect of timing on the simulation trajectories of these two nodes.

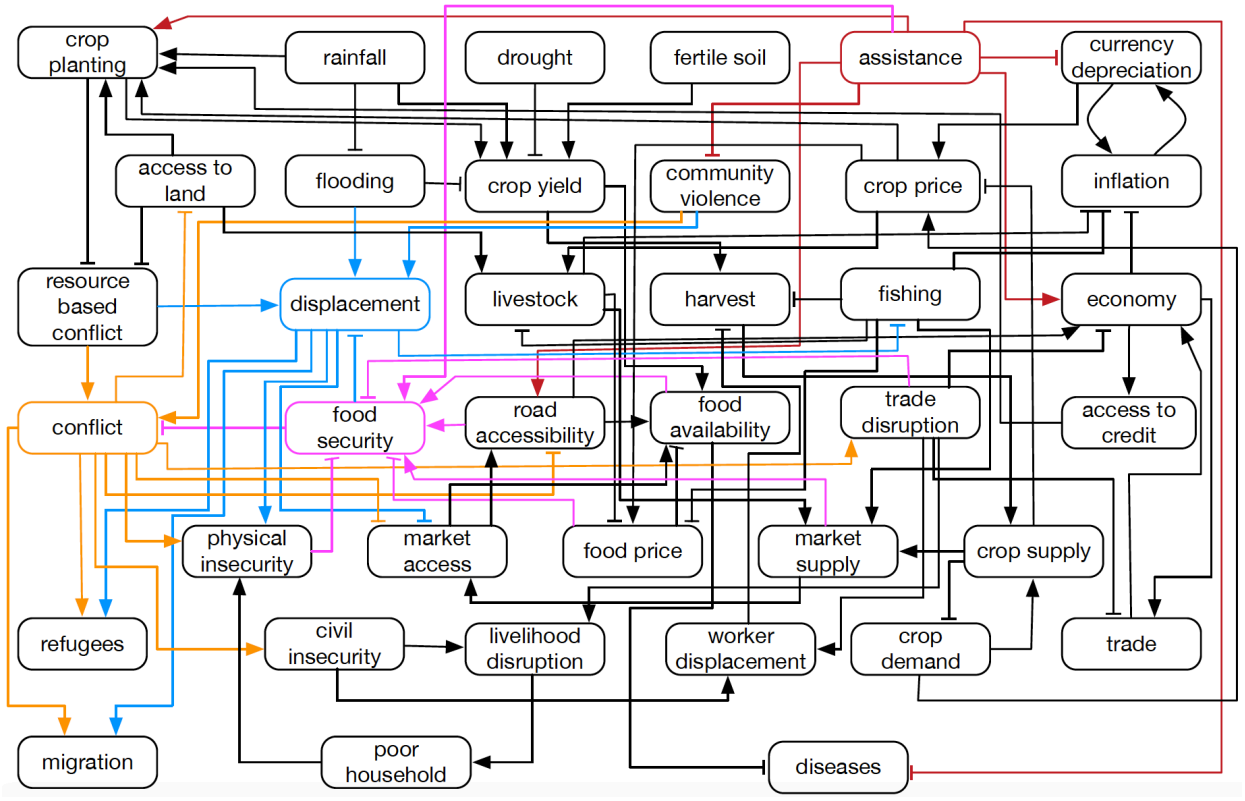


Figure 6-17 Network diagram of the food security in South Sudan model. Nodes with many interactions are highlighted with different colors for better visualization.

6.3.2 Model Simulations

In order to calibrate the model and to predict how food security and the number of refugees will change when assistance is delivered and conflict is reduced, we simulated the model under four scenarios. Scenario 0 represents the situation without introducing assistance and without decreasing conflict, Scenario 1 represents the situation when assistance is delivered but without decreasing conflict, Scenario 2 represents the situation when conflict is decreased to zero but without introducing assistance, and Scenario 3 represents the situation with assistance and reduced conflict. Scenario 0 is designed to calibrate the model and to assure that the average simulation

trajectories match the trends of the historical data for the specified nodes. The data from January 2014 to December 2017 were used to calibrate the model before introducing interventions in January 2018.

Each model variable is assigned three discrete levels (i.e. 0, 1, and 2) except rainfall which is modeled by 10 discrete levels because of the high resolution of the rainfall data. We also used the Random-Sequential update scheme (RSQ) to simulate the model assuming that each simulation step corresponds to one week. We used the toggle functionality described in section 3.2.2 to toggle the value of assistance from 0 to 2 (in scenarios 1 and 3) and conflict from 2 to 0 (in scenarios 2 and 3) at simulation step 209 which corresponds to January 2018. Figure 6-18 shows the historical data and the average simulation trajectories of crop yield, conflict, inflation, and diseases under the four simulation scenarios. It's shown in Figure 6-18 that the historical data is heterogenous with different scales and different time spans. For example, the historical data of crop yield, inflation, and diseases are yearly while the conflict data is monthly. We can also see that the average simulation trajectories of scenario 0 match the general trend of the historical data. The average simulation trajectory of crop yield represents weekly changes and the general trend is increasing as indicated by the historical yearly data. Similarly, the average trajectory of conflict shows a decrease in the middle of the simulation followed by a small increase which resembles the general trend of the conflict monthly data. The simulation trajectory of inflation also follows the trend of the yearly inflation data showing an increase over the whole simulation time before it stabilizes at a medium level. The diseases node combines data for the percentage of HIV prevalence in adult population and the number of malaria incidents per 1,000 population at risk and the simulation trajectory of scenario 0 shows a steady decrease which is also shown in the historical data trends.

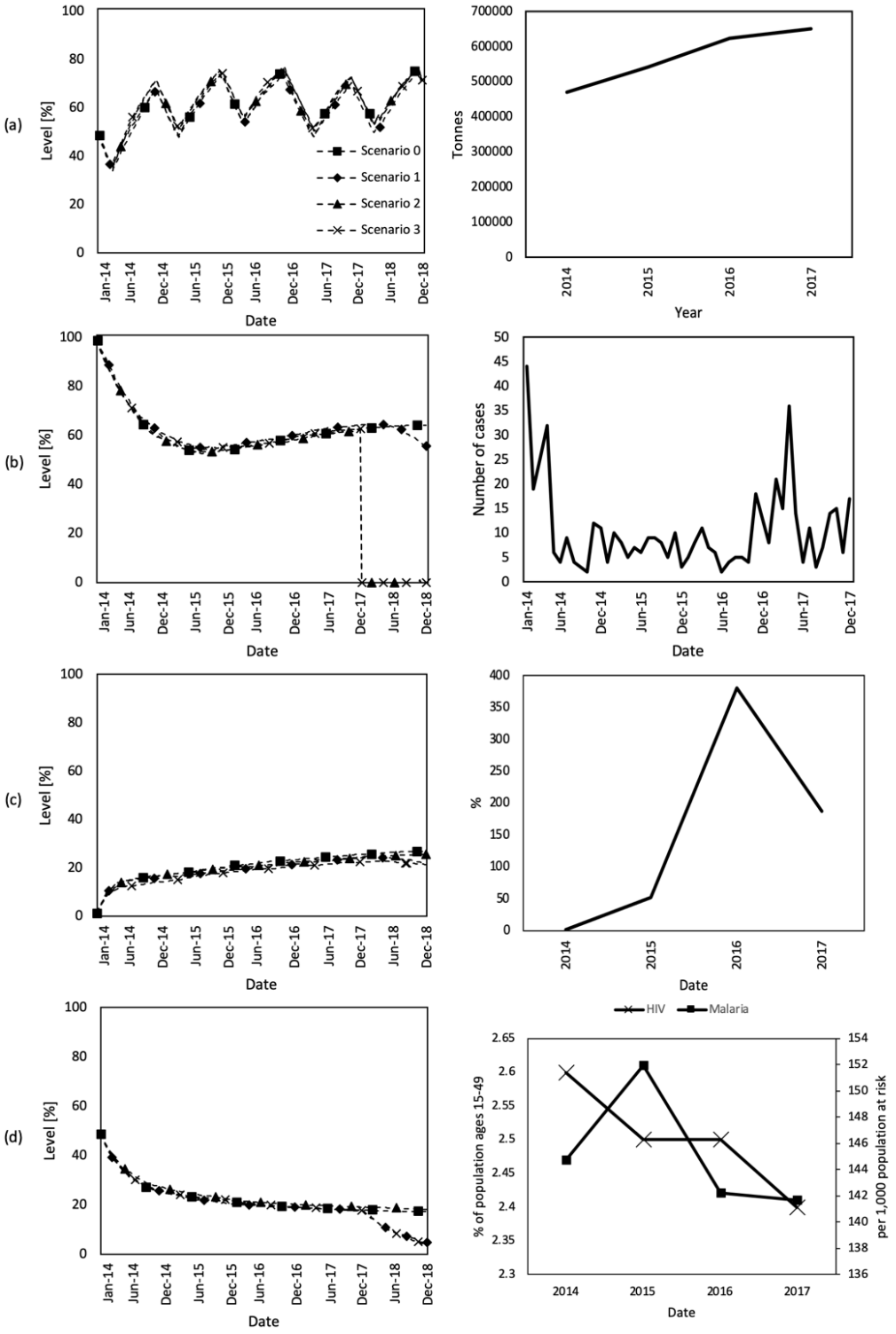


Figure 6-18 Simulation trajectories of the four scenarios (left column) and historical data (right column) for a) crop yield, b) conflict, c) inflation, and d) diseases.

We don't see a large difference in crop yield with the different simulation scenarios because the regulators of crop yield do not change much when assistance is introduced or conflict is decreased. The main driver for crop yield is rainfall which increases and decreases depending on the season and hence, the simulation trajectory of crop yield oscillates. Although there is a direct connection between assistance and crop planting which is a positive regulator for crop yield, the effect of increasing assistance (i.e. scenario 1) on crop yield is not significant because assistance is ORed with the other positive regulators of crop planting which makes the effect on crop yield negligible if any other regulator is active (e.g. rainfall). We can also see in Figure 6-18(b) that conflict starts to decrease at the end of the simulation (i.e. before December 2018) when assistance is introduced at the beginning of 2018 (scenario 1). In scenarios 2 and 3, conflict is toggled to 0 in order to see the effect of decreasing conflict on food security and the number of refugees as shown in Figure 6-19. Figure 6-18(c) shows that inflation is slightly decreased at the end of the simulation time when assistance is introduced (scenario 1) and it decreases a bit more with decreasing conflict (scenarios 2 and 3). Additionally, the average trajectory of diseases shows that introducing assistance, either with or without decreasing conflict (scenario 2), can help in decreasing the prevalence of infectious diseases.

Figure 6-19 shows the average simulation trajectories of food security and refugees' population under the four scenarios. As shown in Figure 6-19(a), the model suggests that food security will continue to decrease because of the increase in trade disruption and physical insecurity that is caused by conflict. When conflict is decreased at the beginning of 2018 (scenario 2), food security is not recovered quickly unless a humanitarian assistance is introduced (scenario 3). The food security trajectory of scenario 1 suggests that introducing assistance can help even with the existence of conflict. However, when conflict is decreased and assistance is introduced,

food security can be improved even more. Figure 6-19(b) shows that the number of refugees will keep increasing, even if assistance is introduced (scenario 1), as long as conflict exists. In scenarios 2 and 3, the number of refugees starts to decrease when conflict is toggled to 0 at the beginning of 2018.

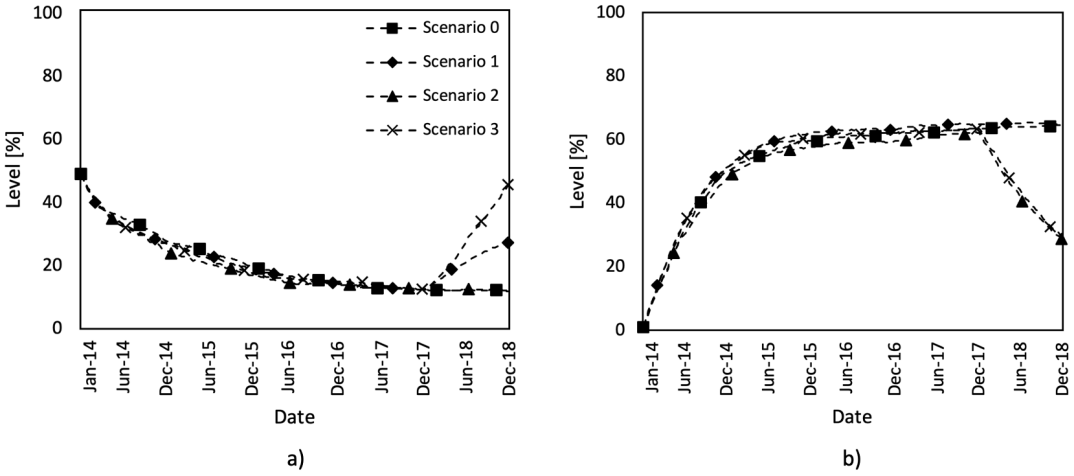


Figure 6-19 Simulation trajectories for a) food security and b) refugees population.

6.3.3 Effect of timing

In this section, we study the effect of adding delays to model elements and interactions in order to better recapitulate the trends of the historical data. Although the simulation trajectories in section 6.3.2 which were obtained using the RSQ scheme show that the model can provide accurate trends, the timing of events might be shifted or unmatched. For example, Figure 6-18(a) shows that crop yield is oscillating with an overall increasing trajectory that matches the trend of the yearly historical data but the increase and decrease in the crop yield trajectory do not match the time of the harvest and the rainy seasons in South Sudan. The harvest season in South Sudan starts in September following a rainy season which runs from June to September. Therefore, we added

delays to the model and utilized the Random-Delay SMLN update scheme described in section 5.3.3, assuming that each simulation step corresponds to a month, to get better matching between the simulation trajectories and the historical data.

The procedure described in section 5.4 is used to find the best set of delays that can be added to make the trajectories of the four elements in Figure 6-18 match the timing of their historical data. We started with crop yield and manually added regulation delays to the crop yield node and its upstream regulators until the average simulation trajectory showed an increase during the time period of the harvest season and a decrease during the rainy season which is determined by the monthly rainfall data that was obtained from (CHIRPS, 2019). The same manual procedure is used to add delays to the conflict, diseases, and inflation nodes and their upstream regulators until the average simulation trajectories were close enough to the trends of the historical data. Once the added regulation and propagation delays were selected manually, the Nelder-Mead optimization technique was utilized to find the optimal delay values using the optimization-based simulation methodology described in section 5.4.2. The historical data of conflict, inflation, and diseases as well as synthesized data for crop yield that show an increase during the harvest season were used to create the cost function for the optimization algorithm as described by Equation 5-8.

Tables 6-4 and 6-5 show model variables and their corresponding manual and optimized delays. Since each model element is assigned three discrete values, there are 4 state transitions (i.e. from 0 to 1, 1 to 2, 2 to 1, and 1 to 0) and hence, we created four regulation delay variables, θ. We also created two more delay variables to represent the spontaneous and the balancing behavior delays. The changed values between the manual and optimized delays are highlighted in red.

Table 6-4 Manual and optimized delay steps assigned to model elements. S represents spontaneous delays, $\theta_{k\to l}$ represents the regulation delay for the transition from discrete level *k* to *l*, and B represents the balancing behavior delays (balancing behabior is set to decrease)

Variable name	Manual delays S, $[\theta_{0\to1}, \theta_{1\to2}, \theta_{2\to1}, \theta_{1\to0}]$, B	Optimized delays S, $[\theta_{0\to1}, \theta_{1\to2}, \theta_{2\to1}, \theta_{1\to0}]$, B
Access_to_land	6, [0,0,5,7], 0	6, [0,0,**4**,7], 0
Civil_insecurity	6, [6,6,0,0], 0	**5**, [6,6,0,0], 0
Community_violence	0, [6,6,2,3], 7	0, [6,6,2,3], **6**
Conflict	0, [6,4,5,5], 5	0, [6,4,5,4], 4
Crop_demand	6, [6,6,0,0], 0	6, [**5**,6,0,0], 0
Crop_planting	8, [6,8,0,0], 0	**7**, [**5**,8,0,0], 0
Crop_price	0, [5,5,6,4], 6	0, [5,**4**,6,4], 6
Crop_supply	6, [3,2,0,0], 0	6, [3,2,0,0], 0
Crop_yield	0, [5,5,0,0], 5	0, [**4**,5,0,0], 5
Currency_depreciation	0, [5,5,0,0], 0	0, [5,5,0,0], 0
Economy	0, [4,3,5,6], 5	0, [4,3,**4**,6], 5
Fishing	8, [0,0,4,2], 0	8, [0,0,4,2], 0
Food_price	8, [5,7,0,0], 0	**7**, [5,7,0,0], 0
Food_security	0, [4,3,5,4], 6	0, [4,**2**,5,4], 6
Inflation	4, [3,4,0,0], 0	**3**, [3,**3**,0,0], 0
Livelihood_disruption	5, [4,5,4,2], 0	5, [4,5,4,2], 0
Livestock	0, [5,6,2,3], 6	0, [5,6,2,3], **5**
Local_market_access	0, [5,6,7,6], 6	0, [**4**,**5**,7,6], 6

Table 6-5 Manual and optimized delay steps for the reset of model elements.

Variable name	Manual delays S, $[\theta_{0\to1}, \theta_{1\to2}, \theta_{2\to1}, \theta_{1\to0}]$, B	Optimized delays S, $[\theta_{0\to1}, \theta_{1\to2}, \theta_{2\to1}, \theta_{1\to0}]$, B
Market_supply	6, [6,5,0,0], 0	6, [6,**4**,0,0], 0
Poor_household	5, [6,5,0,0], 0	5, [6,5,0,0], 0
Refugees	6, [6,6,0,0], 0	6, [6,6,0,0], 0
Resource_based_conflict	5, [0,0,5,6], 0	5, [0,0,5,6], 0
Trade_disruption	6, [6,6,0,0], 0	6, [6,6,0,0], 0
Worker_displacement	7, [6,6,0,0], 0	7, [**5**,6,0,0], 0
Diseases	Hold, [5,6,4,6], 0	Hold, [**4**,6,4,6], 0
Food_availability	0, [5,6,4,5], 6	0, [5,**5**,4,5], 6

The effect of adding manual and optimized delays is shown in Figure 6-20 which includes the average trajectories of crop yield, conflict, inflation, and diseases. It can be seen in Figure 6-20(a) that the increase in crop yield (highlighted with blue boxes) is synchronized with the harvest season which follows the rainy season (where crop yield is decreased). However, there is no difference between the average trajectory obtained with the manual or the optimized delays. Figure 6-20(b) shows that conflict is more decreased compared to the simulation results in Figure 6-18(b) which includes the average trajectory of conflict without delays. It's also shown in Figure 6-20(b) that the optimized delays make conflict decreases more which is closer to the trend of the historical data in Figure 6-18(b) indicating that the optimization algorithm was capable of finding a better set of delay values that make conflict matches better the historical data. In Figures 6-20(c) and (d), we see a small decrease in the trajectories of inflation and diseases with the optimized delays than

the manual delays which is not significant but it shows that the optimization algorithm doesn't damage the model if it can't find a better solution.

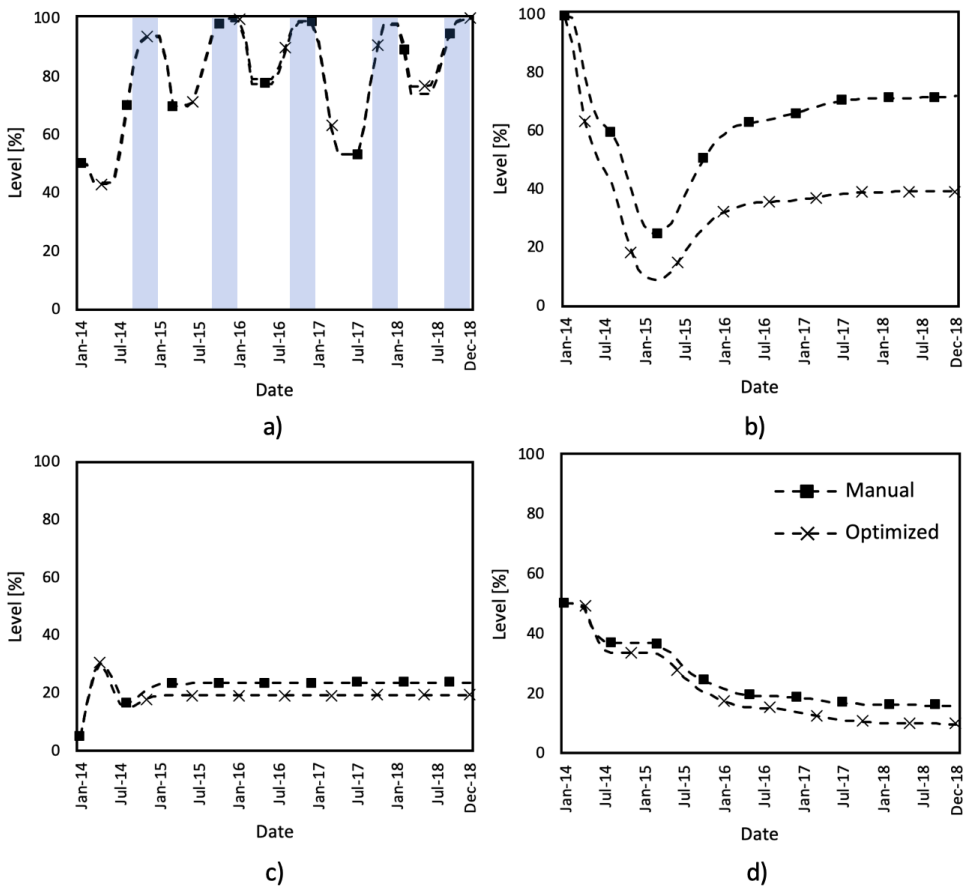


Figure 6-20 Average simulation trajectories of a) crop yield, b) conflict, c) inflation, and d) diseases with manual and optimized delays. Harvest season is highlighted with blue boxes in the crop yield plot.

The effect of adding the optimized delays on food security and the refugees' population is tested under the different simulation scenarios described in section 6.3.2 as shown in Figure 6-21. Similar to the results in Figures 6-18 and 6-19, we see that food security is improved in scenario 1 when assistance is introduced in January 2018 and in scenario 3 when both assistance is introduced and conflict is decreased at the beginning of 2018. There is a small increase in food security under scenario 2 when only conflict is decreased and without introducing assistance which suggests that assistance is crucial for the developing country even after the conflict is stopped. Figures 6-21(d) and (e) show also that inflation and diseases can be decreased if assistance is introduced regardless of the conflict status. However, Figure 6-21(f) shows that the number of refugees can decrease dramatically if conflict is decreased even without assistance.

The simulation results in Figures 6-18, 6-19, and 6-21 show that both the RSQ and Random-Delay SMLN schemes can provide average trajectories similar to the trends of the historical data and can predict what will happen for food security and refugees' population when interventions such as providing humanitarian assistance and stopping conflict are made. However, each method represents the time of events differently. In the RSQ scheme, time is implicit and evaluating each update function can be separated by a number of simulation steps (i.e. time) equivalent to the number of model variables since the probability of evaluating a function at a specific simulation is $1/n$ where n is the total number of model variables. On the other hand, time is explicitly represented as delay steps in the Random-Delay SMLN scheme where all update functions can be evaluated at each simulation step after the delay steps have elapsed. Although the Random-Delay SMLN scheme with the introduced time modeling methods are efficient in recapitulating the trends of the historical data, knowledge about the relative timing of events has to be available. If such knowledge is not available, the RSQ scheme can provide accurate

simulation trajectories with less effort because of the randomness that the simulation scheme introduces when selecting an update function for updating. Additionally, using an optimization-based simulation method for finding the optimal set of delay variables can help in increasing the modeler's confidence in the manually added delays and can make the simulation trajectories match the historical or experimental trends of data which improves the model accuracy.

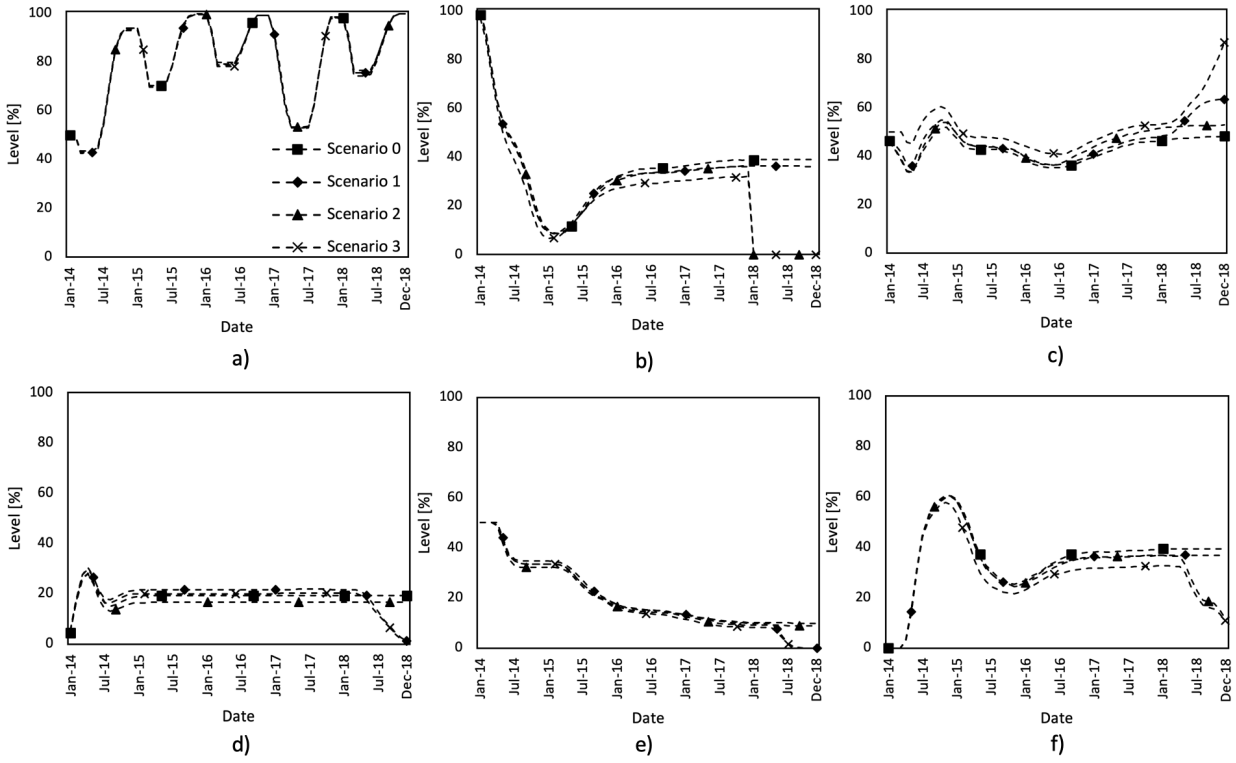


Figure 6-21 average simulation trajectories of a) crop yield, b) conflict, c) food security, d) inflation, e) diseases, and f) inflation using the Random-Delay SMLN scheme with the optimized delays under the four simulation scenarios described in section 6.3.2.

7.0 Conclusion and Future Work

In this work, we introduced methods for facilitating the process of creating and analyzing discrete models of complex systems with special focus on explicit representation of time in order to provide means for modeling the relative timing of systems' events. We developed a representation format that is easy for modelers to interact with and easy for machines to process when automated model creation through literature reading is needed. The representation format works as a scaffold for holding the information extracted from automated or manual literature reading as well as being a good interface for modelers to change the parameterizations of the model such as adding initial values, changing the structure of the update functions, adding delays, controlling the spontaneous and balancing behaviors, and storing the metadata of the model. The representation format is also readable by the DiSH simulator which is developed to simulate logical and discrete models under many simulation schemes as described in Chapter 3. We also introduced techniques for evaluating the update functions when multiple discrete levels are assigned to model elements. The developed techniques utilize min, max and/or weighted sum functions to evaluate the update functions at each simulation step. The modeler can also control the spontaneous behavior of elements that have only positive or only negative regulators as well as choosing what should happen if the strength of the positive and negative regulators of an element is equal (balancing behavior). Choosing between the min, max, or weight sum functions and controlling the spontaneous and balancing behavior is all integrated with the representation format with special notations that makes it easy for modelers to make many changes. Additionally, we developed methods for standardizing the representation of common network motifs in order to

facilitate the process of converting the unstructured interactions that can be extracted using automated literature reading into well-structured interactions with specific update functions.

We developed also techniques for incorporating timing information into discrete and logical models of complex systems in order to consider the relative timing of systems events. The developed methods include assigning delays to control the time needed for a regulation signal to travel from the regulator to the regulated element and to control the time needed by an element to change its state from one discrete level to another. We also allow modelers to assign different delays under different regulation conditions which gives them flexibility to try multiple options when building the model. The developed delay methods allowed us to introduce a new simulation scheme (i.e. Random-Delay SMLN) which runs fast as the SMLN scheme and provides stochastic and accurate trajectories as the RSQ but with explicit time representation. Additionally, we utilized the Nelder-Mead algorithm to optimally find the best set of delay parameters that reduce the error between the average simulation trajectories and the trend of the historical or experimental data. The developed optimization-based simulation technique allows for direct mapping between the simulation steps and the actual timing of systems events and provides a means for matching the historical and experimental data in an automated way.

The developed methods in this dissertation were applied on three models of complex systems such as the naïve T cell differentiation, the budding yeast cell cycle, and food security in South Sudan. The introduced methods showed better performance compared to the methods in literature and indicated that we can get better simulation results in less time and with less complex models. The time modeling methods showed that, when timing information is available, we can get more accurate trajectories that have transient and steady state responses similar to the trends of the historical and experimental data. If timing information are not available, as in the food

security model, the RSQ without delays can provide accurate trajectories in less time and with less effort because we don't need to try many delay values. The results of applying the optimization algorithm suggest that we can improve the trend of the simulation trajectories if we start from a good initial point and therefore modelers have to manually add delays to selected model elements based on background knowledge until the behavior of the elements of interest becomes closer to the trend of the experimental or historical data which is time consuming and labor intensive.

As a future work, more optimization-based simulation techniques such as the genetic algorithm could be used to find the optimal delay values. The genetic algorithm is a global optimization algorithm that traverses the solution space with less probability of getting stuck in a local minimum. Also, developing methods for automatically extracting timing information from literature and linking them to the optimization algorithm could be useful in speeding up the modeling process because modelers might not need to add manual delays to start the optimization algorithm with. Computer vision techniques could be used to read charts and figures from published literature and the extracted information can be organized into a time series format that could be used by modelers to build the cost function that is needed by the optimization algorithms. Additionally, code parallelization is essential for reducing the simulation time and speeding up the run times of the optimization-based simulation techniques which will allow modelers to try many parameterizations in a less amount of time. The introduced methods could also be applied on cancer progression and organs development models where time is crucial and can largely affect the simulation results.

Bibliography

ACLED. (2019). The Armed Conflict Location & Event Data Project. Retrieved from https://www.acleddata.com/

Albert, I., Thakar, J., Li, S., Zhang, R., & Albert, R. (2008). Boolean network simulations for life scientists. *Source code for biology and medicine, 3*(1), 16.

Albert, R., & Robeva, R. (2015). Signaling networks: Asynchronous boolean models. In *Algebraic and discrete mathematical methods for modern biology* (pp. 65-91): Elsevier.

Aldridge, B. B., Saez-Rodriguez, J., Muhlich, J. L., Sorger, P. K., & Lauffenburger, D. A. (2009). Fuzzy logic analysis of kinase pathway crosstalk in TNF/EGF/insulin-induced signaling. *PLoS computational biology, 5*(4), e1000340.

Bähler, J. (2005). Cell-cycle control of gene expression in budding and fission yeast. *Annu. Rev. Genet., 39*, 69-94.

Bioentities. Bioentities Database. Retrieved from https://github.com/sorgerlab/bioentities

Blinov, M. L., Faeder, J. R., Goldstein, B., & Hlavacek, W. S. (2004). BioNetGen: software for rule-based modeling of signal transduction based on the interactions of molecular domains. *Bioinformatics, 20*(17), 3289-3291.

Bock, M., Scharp, T., Talnikar, C., & Klipp, E. (2014). BooleSim: an interactive Boolean network simulator. *Bioinformatics, 30*(1), 131-132.

Buckler, J. L., Liu, X., & Turka, L. A. (2008). Regulation of T-cell responses by PTEN. *Immunological reviews, 224*(1), 239-248.

Chen, K. C., Calzone, L., Csikasz-Nagy, A., Cross, F. R., Novak, B., & Tyson, J. J. (2004). Integrative analysis of cell cycle control in budding yeast. *Molecular biology of the cell, 15*(8), 3841-3862.

Chen, K. C., Csikasz-Nagy, A., Gyorffy, B., Val, J., Novak, B., & Tyson, J. J. (2000). Kinetic analysis of a molecular model of the budding yeast cell cycle. *Molecular biology of the cell, 11*(1), 369-391.

CHIRPS. (2019). Climate Hazards Group InfraRed Precipitation with Station Data. Retrieved from https://developers.google.com/earth-engine/datasets/catalog/UCSB-CHG_CHIRPS_DAILY

Curiel, T. J., Coukos, G., Zou, L., Alvarez, X., Cheng, P., Mottram, P., . . . Burow, M. (2004). Specific recruitment of regulatory T cells in ovarian carcinoma fosters immune privilege and predicts reduced survival. *Nature medicine, 10*(9), 942.

Danos, V., Feret, J., Fontana, W., Harmer, R., & Krivine, J. (2008). Rule-based modelling, symmetries, refinements. In *Formal methods in systems biology* (pp. 103-122): Springer.

Dee, D., & Ghil, M. J. S. J. o. A. M. (1984). Boolean difference equations, I: Formulation and dynamic behavior. *44*(1), 111-126.

Epstein, J. M. (2008). Why model? *Journal of Artificial Societies and Social Simulation, 11*(4), 12.

Facciabene, A., Motz, G. T., & Coukos, G. (2012). T-regulatory cells: key players in tumor immune escape and angiogenesis. *Cancer research, 72*(9), 2162-2171.

Faeder, J. R., Blinov, M. L., & Hlavacek, W. S. (2009). Rule-based modeling of biochemical systems with BioNetGen. *Systems biology*, 113-167.

FAO. (2016). *PEACE AND FOOD SECURITY Investing in resilience to sustain rural livelihoods amid conflict*. Retrieved from http://www.fao.org/3/a-i5591e.pdf

Gan, X., & Albert, R. (2018). General method to find the attractors of discrete dynamic models of biological systems. *Physical Review E, 97*(4), 042308.

Ghil, M., & Mullhaupt, A. J. J. o. s. p. (1985). Boolean delay equations. II. Periodic and aperiodic solutions. *41*(1-2), 125-173.

Gillespie, A., Rokugawa, S., Matsunaga, T., Cothern, J. S., Hook, S., & Kahle, A. B. (1998). A temperature and emissivity separation algorithm for Advanced Spaceborne Thermal

Emission and Reflection Radiometer (ASTER) images. *IEEE transactions on geoscience and remote sensing, 36*(4), 1113-1126.

GO. Gene Ontology Database. Retrieved from http://geneontology.org

Gonzalez, A. G., Naldi, A., Sanchez, L., Thieffry, D., & Chaouiya, C. (2006). GINsim: a software suite for the qualitative modelling, simulation and analysis of regulatory networks. *Biosystems, 84*(2), 91-100.

Harris, L. A., Hogg, J. S., Tapia, J.-J., Sekar, J. A., Gupta, S., Korsunsky, I., . . . Faeder, J. R. (2016). BioNetGen 2.2: advances in rule-based modeling. *Bioinformatics, 32*(21), 3366-3368.

Hawse, W. F., Sheehan, R. P., Miskov-Zivanov, N., Menk, A. V., Kane, L. P., Faeder, J. R., & Morel, P. A. (2015). Cutting edge: differential regulation of PTEN by TCR, Akt, and FoxO1 controls CD4+ T cell fate decisions. *The Journal of Immunology, 194*(10), 4615-4619.

Helikar, T., & Rogers, J. A. (2009). ChemChains: a platform for simulation and analysis of biochemical networks aimed to laboratory scientists. *BMC systems biology, 3*(1), 58.

HGNC. Database of Human Gene Names. Retrieved from http://www.genenames.org

InterPro. InterPro Database. Retrieved from https://www.ebi.ac.uk/interpro/

Irons, D. (2009). Logical analysis of the budding yeast cell cycle. *Journal of theoretical Biology, 257*(4), 543-559.

Kauffman, S. A. (1969). Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of theoretical Biology, 22*(3), 437-467.

Kauffman, S. A. (1993). *The origins of order: Self-organization and selection in evolution*: Oxford University Press, USA.

Kerdiles, Y. M., Stone, E. L., Beisner, D. L., McGargill, M. A., Ch'en, I. L., Stockmann, C., . . . Hedrick, S. M. (2010). Foxo transcription factors control regulatory T cell development and function. *Immunity, 33*(6), 890-904.

Kim, H. P., & Leonard, W. J. (2002). The basis for TCR-mediated regulation of the IL-2 receptor α chain gene: role of widely separated regulatory elements. *The EMBO journal, 21*(12), 3051-3059.

Li, F., Long, T., Lu, Y., Ouyang, Q., & Tang, C. (2004). The yeast cell-cycle network is robustly designed. *Proceedings of the National Academy of Sciences, 101*(14), 4781-4786.

Li, S., Assmann, S. M., & Albert, R. (2006). Predicting essential components of signal transduction networks: a dynamic model of guard cell abscisic acid signaling. *PLoS Biol, 4*(10), e312.

Macian, F. (2005). NFAT proteins: key regulators of T-cell development and function. *Nature Reviews Immunology, 5*(6), 472.

Materi, W., & Wishart, D. S. (2007). Computational systems biology in drug discovery and development: methods and applications. *Drug discovery today, 12*(7-8), 295-303.

McCluskey, E. J. (1956). Minimization of Boolean functions. *Bell Labs Technical Journal, 35*(6), 1417-1444.

MeSH. MeSH Database. Retrieved from https://www.ncbi.nlm.nih.gov/mesh

Miskov-Zivanov, N., Marculescu, D., & Faeder, J. R. (2013). *Dynamic behavior of cell signaling networks: model design and analysis automation.* Paper presented at the Proceedings of the 50th Annual Design Automation Conference.

Miskov-Zivanov, N., Turner, M. S., Kane, L. P., Morel, P. A., & Faeder, J. R. (2013a). Duration of T cell stimulation as a critical determinant of cell fate and plasticity. *Science signaling, 6*(300), ra97.

Morris, M. K., Saez-Rodriguez, J., Clarke, D. C., Sorger, P. K., & Lauffenburger, D. A. (2011). Training signaling pathway maps to biochemical data with constrained fuzzy logic: quantitative analysis of liver cell responses to inflammatory stimuli. *PLoS computational biology, 7*(3), e1001099.

Müssel, C., Hopfensitz, M., & Kestler, H. A. (2010). BoolNet—an R package for generation, reconstruction and analysis of Boolean networks. *Bioinformatics, 26*(10), 1378-1380.

Nelder, J. A., & Mead, R. (1965). A simplex method for function minimization. *The computer journal, 7*(4), 308-313.

Ohkura, N., Kitagawa, Y., & Sakaguchi, S. (2013). Development and maintenance of regulatory T cells. *Immunity, 38*(3), 414-423.

Pfam. Pfam Database. Retrieved from http://pfam.xfam.org/

PubChem. PubChem Database. Retrieved from https://pubchem.ncbi.nlm.nih.gov

Quine, W. V. (1955). A way to simplify truth functions. *The American Mathematical Monthly, 62*(9), 627-631.

Sayed, K., Bocan, K. N., & Miskov-Zivanov, N. (2018). *Automated Extension of Cell Signaling Models with Genetic Algorithm.* Paper presented at the Engineering in Medicine and Biology Society (EMBC), 2018 40th Annual International Conference of the IEEE.

Sayed, K., Kuo, Y.-H., Kulkarni, A., & Miskov-Zivanov, N. (2017a). *DiSH simulator: Capturing dynamics of cellular signaling with heterogeneous knowledge*. Paper presented at the 2017 Winter Simulation Conference (WSC), Las Vegas, NV.

Sayed, K., Telmer, C. A., Butchy, A. A., & Miskov-Zivanov, N. (2017). *Recipes for Translating Big Data Machine Reading to Executable Cellular Signaling Models.* Paper presented at the International Workshop on Machine Learning, Optimization, and Big Data.

Sayed, K., Telmer, C. A., & Miskov-Zivanov, N. (2016). *Motif modeling for cell signaling networks.* Paper presented at the Biomedical Engineering Conference (CIBEC), 2016 8th Cairo International.

Schaub, M. A., Henzinger, T. A., & Fisher, J. (2007). Qualitative networks: a symbolic approach to analyze biological signaling networks. *BMC systems biology, 1*(1), 4.

Schmidt, H., & Jirstrand, M. (2005). Systems Biology Toolbox for MATLAB: a computational platform for research in systems biology. *Bioinformatics, 22*(4), 514-515.

Sevim, V., Gong, X., & Socolar, J. E. J. P. c. b. (2010). Reliability of transcriptional cycles and the yeast cell-cycle oscillator. *6*(7), e1000842.

Sharp, R., Pyarelal, A., Gyori, B., Alcock, K., Laparra, E., Valenzuela-Escárcega, M. A., . . . Tang, Z. (2019). *Eidos, INDRA, & Delphi: From free text to executable causal models.* Paper presented at the Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations).

Shevach, E. M. (2000). Regulatory T cells in autoimmmunity. *Annual review of immunology, 18*(1), 423-449.

Singh, V., & Dhar, P. K. (2015). *Systems and synthetic biology*: Springer.

Thomas, R. (1973). Boolean formalization of genetic control circuits. *Journal of theoretical Biology, 42*(3), 563-585.

Thomas, R., & d'Ari, R. (1990). *Biological feedback*: CRC press.

Turner, M. S., Kane, L. P., & Morel, P. A. (2009). Dominant role of antigen dose in CD4+ Foxp3+ regulatory T cell induction and expansion. *The Journal of Immunology, 183*(8), 4895-4903.

UniProt. UniProt Database. Retrieved from http://www.uniprot.org

Wang, R.-S., Saadatpour, A., & Albert, R. (2012). Boolean modeling in systems biology: an overview of methodology and applications. *Physical biology, 9*(5), 055001.

Yao, Z., Kanno, Y., Kerenyi, M., Stephens, G., Durant, L., Watford, W. T., . . . Moriggl, R. (2007). Nonredundant roles for Stat5a/b in directly regulating Foxp3. *Blood, 109*(10), 4368-4375.

Yu, S. J., Tung, T. Q., Park, J., Lim, J., & Yoo, J. (2012). ezBioNet: A modeling and simulation system for analyzing biological reaction networks. *Journal of the Korean Physical Society, 61*(8), 1267-1273.

Zheng, J., Zhang, D., Przytycki, P. F., Zielinski, R., Capala, J., & Przytycka, T. M. (2010). SimBoolNet—a Cytoscape plugin for dynamic simulation of signaling networks. *Bioinformatics, 26*(1), 141-142.

Ziegler, S. F. (2006). FOXP3: of mice and men. *Annu. Rev. Immunol., 24*, 209-226.