The Algebra of Systems Biology

Alan A. Veliz-Cuba

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

Doctor of Philosophy
in
Mathematics

Reinhard Laubenbacher, Chair
John Burns
Abdul Salam Jarrah
Henning Mortveit
Brett Tyler

July 5, 2010
Blacksburg, Virginia

Keywords: Mathematical Biology, Systems Biology, Finite Dynamical Systems, Reverse Engineering, Model Reduction, Finite Fields, Polynomial Algebra, Discrete Models.

The Algebra of Systems Biology

Alan A. Veliz-Cuba

ABSTRACT

In order to understand biochemical networks we need to know not only how their parts work but also how they interact with each other. The goal of systems biology is to look at biological systems as a whole to understand how interactions of the parts can give rise to complex dynamics. In order to do this efficiently, new techniques have to be developed. This work shows how tools from mathematics are suitable to study problems in systems biology such as modeling, dynamics prediction, reverse engineering and many others. The advantage of using mathematical tools is that there is a large number of theory, algorithms and software available. This work focuses on how algebra can contribute to answer questions arising from systems biology.

This work received support from:
NSF Grant DMS-0511441

Dedication

Dedico este trabajo a mis queridos padres por su esfuerzo en darme la mejor educación posible, a mi amada esposa porque su apoyo ha sido fundamental durante estos años y a mis adorados hijos por ser mi mayor fuente de inspiración y motivación.

Acknowledgments

I am very grateful to my advisor, Reinhard Laubenbacher. His mentoring, help and patience have been fundamental during my years in grad school and my research.

I thank the members of my committee, John Burns, Henning Mortveit, Brett Tyler and especially Abdul Salam Jarrah; Abdul has been a great help during these years. I also thank the professors at the Math Department, especially Peter Haskell and Ezra Brown.

I thank Eileen Shugart for her advice and patience during the time I taught classes and Abigail Kohler for all her time and advice that made me enjoy teaching. I also thank the staff at the Math Department and VBI, especially Hannah Swiger, Nicole Sutphin and Betsy Williams.

I thank my undergraduate advisor, Edgar Vera Saravia, who got me interested in mathematics.

I thank my colleagues at the Math Department and VBI, especially David Murrugarra, José María Menéndez and Brandilyn Stigler.

I thank my parents, Rayda Cuba de la Cruz and Ricardo Veliz Revolledo, who gave me a loving home and worked very hard to give me the best education. *Gracias mamá y papá por haberme criado en un hogar lleno de amor y gracias por su esfuerzo para darme la mejor educación posible.* I also thank my brother and sister, Ervin and Carol, they always made me give the best of me as the oldest brother.

I thank my wife, Sandra Paredes; her love, patience and support have been fundamental all these years. I also thank my children, Adrian and Allison, they are my source of inspiration and motivation.

Contents

1 Introduction **1**

1.1 Mathematical View of Systems Biology 1

1.1.1 Systems Biology 1

1.1.2 Mathematical Modeling 2

1.1.3 Example 2

1.2 Polynomial Dynamical Systems 5

1.2.1 Definition of Polynomial Dynamical Systems 5

1.2.2 Simulation of Polynomial Dynamical Systems 6

1.3 Organization of the Thesis 7

2 Algebraic Models **11**

2.1 The Lac Operon 11

2.2 Model of the Lac Operon 13

2.2.1 Modeling Background 14

2.2.2 Boolean Network 15

2.2.3 Core Subnetworks 19

2.2.4 Alternative Models 22

2.3 Discussion 23

3 Prediction of Dynamics from Network Structure **25**

3.1 Effect of Negative Feedback Loops on the Dynamics of BNs 25

3.1.1 Distance to positive-feedback 27

3.1.2 Example . . . 30
3.1.3 Method . . . 31
3.1.4 Results . . . 34
3.1.5 Discussion . . . 37
3.2 Dynamics of Conjunctive Networks . . . 38
3.2.1 Background . . . 40
3.2.2 Networks with Strongly Connected Dependency Graph . . . 48
3.2.3 Networks with general dependency graph . . . 51
3.2.4 Bounds on the cycle structure . . . 53
3.2.5 A Lower Bound . . . 54
3.2.6 An Upper Bound . . . 55
3.2.7 Discussion . . . 60

4 Model Reduction and Inference **61**
4.1 Reduction of Boolean Networks . . . 61
4.1.1 Reduction Method . . . 61
4.1.2 Properties of the Reduction Method . . . 66
4.1.3 Application . . . 70
4.1.4 Discussion . . . 72
4.2 Reverse Engineering of Regulatory Networks . . . 72
4.2.1 Preliminaries . . . 74
4.2.2 Algorithm . . . 76
4.2.3 Application to Continuous Data . . . 77
4.2.4 Reverse Engineering Systems . . . 78
4.2.5 Example . . . 78
4.2.6 Discussion . . . 80

5 Polynomial Form of Other Discrete Frameworks **81**
5.1 Algorithm for Logical Models . . . 82

5.1.1 Definition of Logical Models 83
5.1.2 Polynomial Form of Logical Models 84
5.1.3 Functionality of Edges of Boolean Models 84
5.1.4 Functionality of Circuits of Boolean Models 85
5.1.5 Functionality of Circuits of Logical Models 86
5.2 Examples 87
5.2.1 Logical Model of the Core Lambda Switch 87
5.2.2 Logical Model of the *Drosophila* Cell Cycle 89
5.2.3 Logical Model for Th-Lymphocyte Differentiation 90
5.3 Algorithm for K-Bounded Petri Nets 94
5.3.1 Definition of Petri Nets 94
5.3.2 Polynomial Form of K-Bounded Petri nets 96
5.4 Examples 97
5.4.1 Petri net for the *Drosophila* Cell Cycle 97
5.4.2 Petri net for the Core Lambda Switch 99
5.5 Algebra and Parameters 101
5.6 Performance 103
5.7 Compatibility of the Algebraic Framework with Other Frameworks 104
5.7.1 Example 105
5.8 Discussion 107

List of Figures

1.1 Nodes of the system f with the corresponding activators/repressors. A normal arrow indicates activation and a circle indicates inhibition. 3

1.2 Systems that have similar "local" features as f. 3

1.3 Wiring diagram of the systems in Figure 1.2. 4

1.4 Wiring diagram of f. 6

1.5 State space of f. 7

2.1 Wiring diagram of F. 17

2.2 Bifurcation diagram of our Boolean network and a typical bifurcation diagram showing bistability of an ODE (two saddle-node bifurcation). 19

2.3 Discrete version of the experiments performed in [84] for our Boolean network. 20

2.4 Wiring diagram of f. 21

2.5 Wiring diagram of the model without inducer inclusion and without catabolite repression. 23

2.6 Wiring diagram of the reduced model without inducer inclusion and without catabolite repression. 23

3.1 An unsigned directed graph. 30

3.2 Phase space of f. 31

3.3 Phase space of g. 32

3.4 The best fit-lines of the averages of the numbers (lengths) of limit cycles from Table 3.1. 33

3.5 5-node networks. Histogram of slopes of best-fit lines to the average length of limit cycles (horizontal axis) vs. percentage of experiments with a given slope (vertical axis). The panels include networks with increasing PF-distance, with 25%, 50%, 75%, and 100% of the maximal distance, respectively. 36

3.6 5-node networks. Histogram of slopes of best-fit lines to the average number of limit cycles (horizontal axis) vs. percentage of experiments with a given slope (vertical axis). The panels include networks with increasing PF-distance, with 25%, 50%, 75%, and 100% of the maximal distance, respectively. 37

3.7 The dependency graph(left) and the phase space(right) of f from Example 3.2.1. 41

3.8 The dependency graph of f from Example 3.2.2. 41

3.9 The strongly connected components of f (left) and their poset (right). 44

4.1 Wiring diagram of f before (left) and after (right) using Algorithm S. Arrows indicate positive paths and circles indicate negative paths. 63

4.2 Wiring diagram of f before (left) and after (right) using algorithm R to eliminate x_3. 64

4.3 Wiring diagram of the full (left) and reduced (right) network for Example 4.1.13. 69

4.4 Wiring diagram of the full (left) and reduced (right) network for Example 4.1.14. 69

4.5 Wiring diagram for the Th-Lymphocyte differentiation network. 70

4.6 Wiring diagram for the reduced network of the Th-Lymphocyte differentiation network. 71

4.7 Wiring diagram for h obtained using our method (left) and current methods (right). Note: the diagram on the right is unsigned. 74

4.8 Wiring diagram of the ODE system. 79

4.9 Reverse engineered wiring diagram using our method (left) and previous methods (right). Note: the edges on the right are unsigned. 80

5.1 The network for the core Lambda switch. An arrow (circle) indicates activation (repression). 88

5.2 A network for the *Drosophila* cell cycle [14]. 89

5.3 The regulatory network for Th-Lymphocyte differentiation. 91

List of Tables

1.1 Number of steady states for the systems in Figure 1.2. 4

3.1 The average of the numbers (lengths) of limit cycles of the networks from Example 3.1.3. 31

3.2 The percentage of experiments that conform the hypotheses 35

3.3 The number of experiments that did not conform to our hypothesis for 5-node networks. We considered PF-distance 25%, 50%,75%, and 100% of the maximal distance . For each d, we considered 1000 experiments. 35

4.1 Data set for function $h(x)$. 73

5.1 Parameters for the Core Lambda Switch. 87

5.2 Nonzero parameters for the *Drosophila* cell cycle model. 89

5.3 The nonzero parameters for the Th-Lymphocyte differentiation network. . . 92

5.4 Nonzero and unknown parameters for the *Drosophila* cell cycle. 101

5.5 Timing for the Gröbner basis computations: **N**=number of nodes, **Tr**=timings (in seconds) for $f_1^r(x) - x_1 = \cdots = f_n^r(x) - x_n$=0, where $r = 1, \ldots, 5$. The networks used were: Fission yeast [22], Budding yeast [68], Th cell differentiation [89], Th cell differentiation [74], Th cell differentiation [76], T-cell receptor [65], respectively. All networks were Boolean except the fourth one which had some nodes with 3 states. 103

Chapter 1

Introduction

1.1 Mathematical View of Systems Biology

1.1.1 Systems Biology

In order to understand biochemical networks we need to know not only how their parts work but also how they interact with each other. The reductionist approach has been successful in revealing the parts of such systems and it is the goal of systems biology to look at the system as a whole to understand how the way the parts work with each other (usually in a nonlinear manner) can give rise to complex dynamics. Therefore, an understanding of the design principles of biochemical networks, such as gene regulatory, metabolic, or intracellular signaling networks is a central concern of systems biology. In particular, the intricate interplay between network topology and resulting dynamics is crucial to our understanding of such networks, as is their presumed modular structure. Features that relate network topology to dynamics may be considered "robust" in the sense that their influence does not depend on detailed quantitative features such as exact flux rates. Topological features of interest in this context are feed-forward, positive and negative feedback loops. There is broad consensus that feedback loops have a decisive effect on dynamics, which has been studied extensively through the analysis of mathematical network models, both continuous and discrete. Indeed, it has long been appreciated by biologists that positive and negative feedback loops play a central role in controlling the dynamics of a wide range of biological systems. Thomas et al. [111] conjectured that positive feedback loops are necessary for multistationarity whereas negative feedback loops are necessary for the existence of periodic behavior. Proofs for different partial cases of these conjectures have been given, see, e.g., [31; 86; 103]. Feed-forward loops, on the other hand, can cause delay or acceleration in activation or inhibition [4; 72].

1.1.2 Mathematical Modeling

Many mathematical models of biological systems are given as systems of differential equations which are appropriate for understanding mechanisms. However, for many networks, the available data quantity and quality is not sufficient to build detailed quantitative models, which require many parameters that are frequently unknown. On the other hand, in Boolean models of biological networks, each variable can only attain two values (0/1 or "on/off"); these values represent whether a gene is being expressed, or the concentration of a protein is above a certain threshold. When detailed information on kinetic rates of protein-DNA or protein-protein interactions is lacking, and especially if regulatory relationships are strongly sigmoidal, such models are useful in theoretical analysis, because they serve to focus attention on the basic dynamic characteristics while ignoring specifics of reaction mechanisms, see [3; 16; 61; 63]. Furthermore, there is increasing evidence that the architecture of gene regulatory networks has a strong effect on their dynamics. In [3] it is shown with the help of a Boolean network model that the dynamics of the segment polarity network in *Drosophila melanogaster* is determined by the topology of the wiring diagram and the transcriptional logic of the genes involved rather than the specific kinetics of the system. Also, discrete models can capture key dynamic features of biological networks and can be used successfully for hypothesis generation [69; 76; 92]. Another advantage of discrete models is that qualitative networks eliminate the need to estimate kinetic parameters, thereby reducing model complexity. They also tend to be more intuitive and easily accessible to life scientists. Therefore, discrete modeling frameworks (Boolean and multistate) are receiving more attention in systems biology.

1.1.3 Example

Systems biology is about how the parts of a biological system work together, whereas the reductionist approach is about how each part works. From a mathematical point of view, the way they complement each other can be illustrated by the following example. Consider a system, f, with four nodes (or genes), x_1, x_2, x_3, x_4. Now, suppose that using the reductionist approach we have found out that the update rules (using a Boolean model) for these genes are: $f_1 = f_1(x_2, x_3) = x_2 \wedge x_3$, $f_2 = f_2(x_1, x_4) = x_1 \wedge \neg x_4$, $f_3 = f_3(x_2) = x_2$ and $f_4 = f_4(x_3) = \neg x_3$, where $\wedge, \neg$ are the logical operators AND, NOT, respectively. The AND operator indicates that all activators have to be present and all repressors (given by the NOT operator) have to be absent to activate a gene (remember that $0 \wedge 0 = 1 \wedge 0 = 0 \wedge 1 = 0$, $1 \wedge 1 = 1$,$\neg 0 = 1$ and $\neg 1 = 0$). This system has two steady states (states where $f_i(x_1, \ldots, x_n) = x_i$ for all i), namely $(0, 0, 0, 1)$ and $(1, 1, 1, 0)$: if $(x_1, x_2, x_3, x_4) = (0, 0, 0, 1)$ then $f_1 = 0 \wedge 0 = 0$, $f_2 = 0 \wedge \neg 1 = 0$, $f_3 = 0$ and $f_4 = \neg 0 = 1$, hence $(0, 0, 0, 1)$ is in fact a steady state. Similarly it follows that $(1, 1, 1, 0)$ is a steady state. Therefore, this system is bistable.

From a reductionist point of view the information we have can be summarized in Figure 1.1. Furthermore, Figure 1.2 shows systems that at first sight should behave similar to f. Notice

that for all these systems, x_1 has two activators, x_2 has one activator and one repressor, x_3 has one activator and x_4 has one repressor.

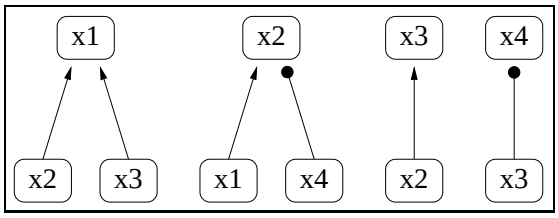


Figure 1.1: Nodes of the system f with the corresponding activators/repressors. A normal arrow indicates activation and a circle indicates inhibition.

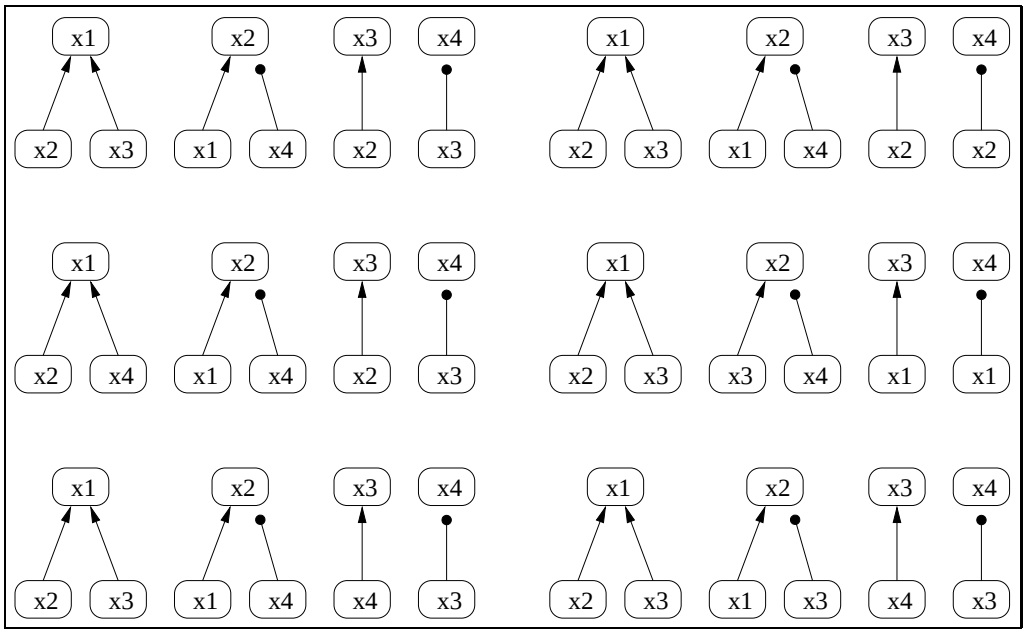


Figure 1.2: Systems that have similar "local" features as f.

However, Table 1.1 shows that the number of steady states is not the same for all systems. This means that knowing the "parts" of the system is not enough to understand its behavior. It is the role of systems biology to understand the way the parts interact with other influences the dynamics of the system.

Now, if we look at the systems from a systems biology point of view we can see that their wiring diagrams are different. That is, it is not the rules that are different (all systems have the same type of rules), but how they are connected. Figure 1.3 shows the wiring diagram (also known as connectivity or dependency graph) of the systems given in Figure 1.2.

We can see that the bistable systems share a common property, they all have positive feedback loops only (closed circuits where the product of the signs is positive) and the other systems have a negative feedback loop. In fact, this is not a coincidence as there is a theorem which

Table 1.1: Number of steady states for the systems in Figure 1.2.

2	2
1	2
0	0

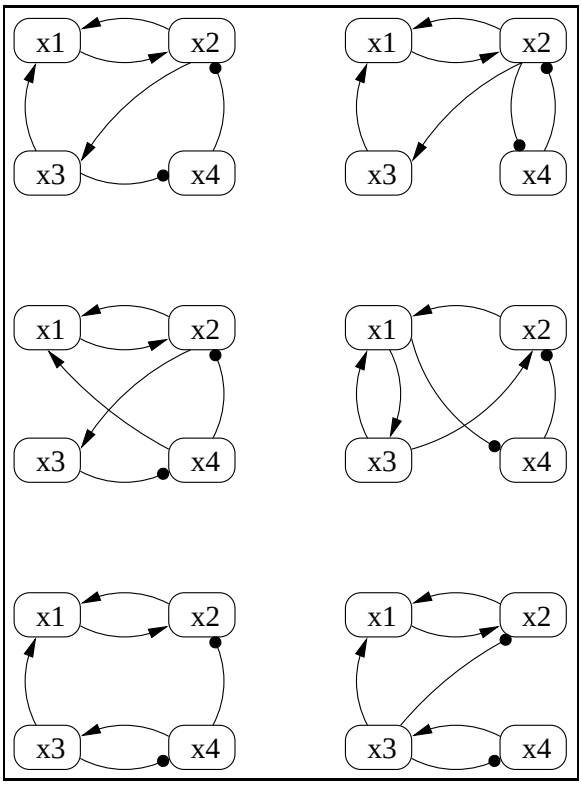


Figure 1.3: Wiring diagram of the systems in Figure 1.2.

guarantees that if a system has positive feedback loops only, then it must have at least two steady states. Also, another theorem states that if a system has two steady states, then it must have at least one positive feedback loop.

This is an example of how mathematics can contribute to systems biology, by providing theorems, algorithms and software that allow the elucidation of the role of network topology on the dynamics of biological systems. Also, a more complicated example can be obtained by replacing the nodes x_1, x_2, x_3, x_4 by systems and showing that knowing the parts is not enough to understand the whole system.

1.2 Polynomial Dynamical Systems

1.2.1 Definition of Polynomial Dynamical Systems

A Boolean network or model is a function f defined over $\{0,1\}^n$, that is, $f : \{0,1\}^n \to \{0,1\}^n$. The coordinate functions of f, $f_1, f_2, \ldots, f_n$ are functions $f_i : \{0,1\}^n \to \{0,1\}$ that indicate what the future value of a node or gene i is in terms of its regulators. For example, in Section 1.1 we have $f : \{0,1\}^4 \to \{0,1\}^4$ given by $f(x_1, x_2, x_3, x_4) = (x_2 \wedge x_3, x_1 \wedge \neg x_4, x_2, \neg x_3)$.

More generally, we consider models of biological systems, such as molecular networks, with n interacting molecular species whose states can be described by an n-tuple with entries from a finite set $\mathbb{F}$. The model consists of a set of rules that allow the system to evolve from one state to the next, so that it can be represented as a finite dynamical system $f : \mathbb{F}^n \longrightarrow \mathbb{F}^n$.

If we consider f only as a set function, then there are few mathematical tools that can help analyze f. One way to introduce mathematical structure, and thereby mathematical tools to study f, is by imposing the algebraic structure of a number system on $\mathbb{F}$, akin to the introduction of a coordinate system in affine space, which gives access to analytical methods in geometry. It is worth pointing out that this algebraic structure is used heavily in the case of Boolean networks. The evaluation of Boolean functions uses the fact that $\mathbb{F} = \{0,1\}$ is equipped with an addition, where 0 is the additive identity, and with a multiplication, where 1 is the multiplicative identity. The two are connected through the rule $1 + 1 = 0$. For example, the algebraic representation of the Boolean function in Section 1.1 becomes $f(x_1, x_2, x_3, x_4) = (x_2x_3, x_1 + x_1x_4, x_2, 1 + x_3)$ (we can obtain the algebraic representation by using the identities $a \wedge b = ab$, $a \vee b = a + b + ab$ and $\neg a = 1 + a$).

Once $\mathbb{F}$ carries such a structure, we can make use of a fundamental property of finite fields: *Let $h : \mathbb{F}^n \to \mathbb{F}$ be a function, where $\mathbb{F}$ is a field with p elements. Then, h can be written as a polynomial.* One can find the polynomial form of h using the formula $h(x) = \sum_{c \in \mathbb{F}^n} h(c) \prod_{j=1}^{n} (1 - (x_j - c_j)^{p-1})$, where the right-hand side is computed modulo p.

Thus, we can represent f via its coordinate functions: $f = (f_1, \ldots, f_n) : \mathbb{F}^n \longrightarrow \mathbb{F}^n$, where each f_i is a polynomial function, so that evaluation and analysis of f can be done using polynomial arithmetic over $\mathbb{F}$. We shall call f a *polynomial dynamical system (PDS)* over $\mathbb{F}$ of dimension n. That is, a PDS is a function $f : \mathbb{F}^n \to \mathbb{F}^n$, where $\mathbb{F}$ is a finite field.

Example 1.2.1 *Consider the PDS $f : \mathbb{F}^2 \to \mathbb{F}^2$ where $\mathbb{F} = \mathbb{F}_3 = \{0,1,2\}$ (the field with 3 elements) and $f(x_1, x_2) = (x_1 + x_2^2 + x_2, x_1x_2 + 1)$. Its wiring diagram is shown in Figure 1.4.*

The advantage of looking at a discrete model as a PDS is that any problem arising from biology has an algebraic counterpart. For example, computing the steady states of a discrete

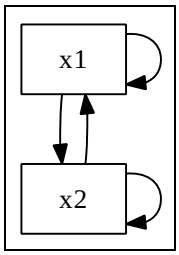


Figure 1.4: Wiring diagram of f.

model becomes the algebraic problem of solving a system of polynomial equations :

$$f_1 - x_1 = 0 \ , \ f_2 - x_2 = 0 \ , \ \dots \ , \ f_n - x_n = 0.$$

For example, the steady states of the PDS in Example 1.2.1 are given by:

$$x_1 + x_2^2 + x_2 - x_1 = 0 \ , x_1x_2 + 1 - x_2 = 0.$$

or

$$x_2^2 + x_2 = 0 \ , x_1x_2 + 1 - x_2 = 0.$$

which has the unique solution $(2, 2)$.

1.2.2 Simulation of Polynomial Dynamical Systems

The evolution of a PDS (or any finite dynamical system), f, is given by the equation

$$\begin{aligned} x(t+1) &= f(x(t)) \\ x(0) &= x_0 \end{aligned}$$

A trajectory of a PDS is a "solution" to the equation above. That is, a trajectory is the sequence $\{x_0, f(x_0), f^2(x_0), \dots, f^i(x_0), \dots\}$, also denoted $x_0 \to f(x_0) \to f^2(x_0) \to \dots \to f^i(x_0) \to \dots$

Example 1.2.1 (cont.) Consider

$$\begin{aligned} f(x_1, x_2) &= (x_1 + x_2^2 + x_2, x_1x_2 + 1) \\ x(0) &= (2, 0) \end{aligned}$$

Then, the trajectory is $(2,0) \to (2,1) \to (1,0) \to (1,1) \to (0,2) \to (0,1) \to (2,1) \to \dots$

The state space of a PDS is made up of all trajectories and it can be represented as a graph, given by Figure 1.5 (This graph was generated using DVD [55]).

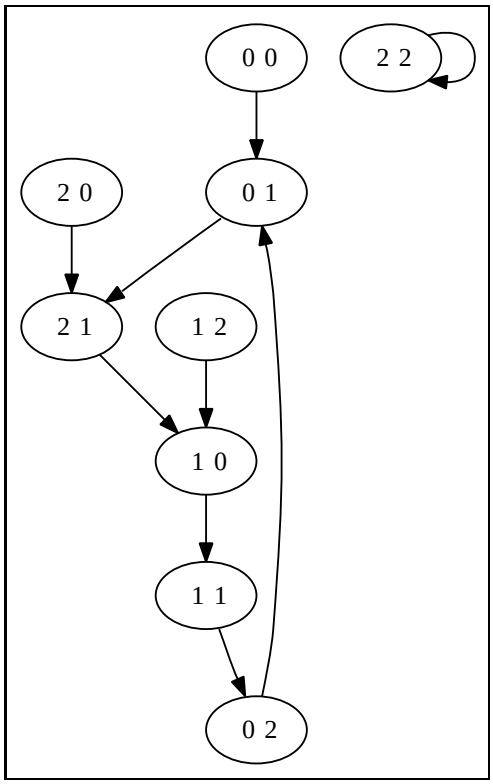


Figure 1.5: State space of f.

An attractor or limit cycle of length t is the trajectory with t elements $x_0 \to f(x_0) \to f^2(x_0) \ldots \to f^{t-1}(x_0)$ such that $f^t(x_0) = x_0$, where t is minimal. An attractor of length 1 ($f(x_0) = x_0$) is called a steady state or a fixed point. Since the state space is finite, it follows that every trajectory must end in an attractor. For our example we have one limit cycle of length 1 and one of length 5. We can denote this symbolically by $CS(f) = \mathcal{C}^1 + \mathcal{C}^5$.

1.3 Organization of the Thesis

Chapter 2: Algebraic Models

A Boolean model can be constructed by writing biological information in terms of Boolean operators (e.g. AND, OR, NOT). For example, when two proteins have to bind for activation to occur or when promoter regions are not independent, then the AND operator would be used; on the other hand, when a gene has multiple promoter regions and they are independent, then the OR operator can capture this property. Also, negative regulation can be represented by the NOT operator.

In this chapter we include an example of a biological system, the lac operon in E. coli, and

show how a Boolean model, and therefore a PDS, is constructed. Also, we show how that although the Boolean model is a coarse representation of the biological system, it can explain bistability as well as an ODE model. This means that bistability in the lac operon is caused not by the kinetics of the system, but by the intrinsic topological structure. Furthermore, we show that even a smaller model is capable of explaining bistability. This further supports that bistability is produced by the structure of the wiring diagram. This chapter is based on a paper written in collaboration with Brandilyn Stigler. My contributions to the paper were the construction of the model and reduced model as well as their analysis.

Chapter 3: Prediction of Dynamics from Network Structure

After one has a model, the next step is to analyze it. In particular, steady states and limit cycles are of great interest. One way to analyze a PDS is through simulation; however, since the size of the state space grows exponentially with respect to the number of nodes, simulation quickly becomes unfeasible. Another way to analyze the model is to identify key features of the wiring diagram that constraint the dynamics. For example, it is known that positive feedback loops are necessary in order to have multistability. On the other hand, it has been conjecture that negative feedback loops are responsible for periodic behavior.

Section 1 of this chapter is about the role of negative feedback loops on the dynamics of Boolean networks. We define a measure (PF-distance) that determines how far from monotone a network is. Since it is believed that networks far from monotone are chaotic, we made a computational study to determine the effect of increasing the distance to monotone on the number and length of attractors. The results indicate that networks close to monotone have more attractors which have short length whereas networks that are far from monotone have few attractors which have large length. Since the larger an attractor is, the less robust it becomes to fluctuations, our results show that networks far from monotone tend to be more chaotic. This section is based on a published paper written in collaboration with Eduardo Sontag, Reinhard Laubenbacher and Abdul Salam Jarrah: *The effect of negative feedback loops on the dynamics of Boolean networks. Biophysical Journal, 95:518-526, 2008.* My contributions to the paper were the design of the experiments, algorithms, coding and the analysis of the results.

The next step in our goal to understand the role of network topology on the dynamics is to find formulas for the number of limit cycles. Since the general problem is unfeasible (even determining the existence of fixed points is NP-complete, [121]), we focused on the family of conjunctive Boolean networks. Conjunctive networks correspond to systems where genes are activated by the synergistic action of its regulators. It is important to mention that the work done for linear systems ([27; 46]) and our work for conjunctive networks (which are nonlinear networks) cover the two extremes of type of networks.

Section 2 of this chapter is about the role of the wiring diagram on the dynamics of Boolean networks made up of the AND operator, called conjunctive Boolean networks. We define a

measure of the connectivity of modules of the wiring diagram, namely, the loop number of the strongly connected components and show that for networks with strongly connected wiring diagram the loop number has all the information needed to determine the number of attractors of any given length. Furthermore, we provide formulas for the number of attractors. In the case where the wiring diagram has more than one strongly connected component, we provide lower and upper bounds for the number of attractors. These bounds are given in algebraic terms of the connectivity of the strongly connected components (the poset structure) and the attractors of strongly connected components. This section is based on a published paper written in collaboration with Abdul Salam Jarrah and Reinhard Laubenbacher: *The dynamics of conjunctive and disjunctive Boolean network models. Bull. Math. Bio., 2010.* My contributions to the paper where the statements and proofs of the main results.

Finally, our goal is to generalize the results about conjunctive networks to networks that allow negative regulations and to any finite field. Our recent work shows that the formulas for the number of limit cycles of conjunctive networks is also valid for "conjunctive" networks over any finite set. Also, we have shown that finding the number of steady states of a "signed" conjunctive network can be transformed to the problem of counting maximal independent sets in a graph.

Chapter 4: Model Reduction and Inference

Another way to analyze a model is to reduce it and use the reduce model to make predictions about the original model. For example, the quasi-steady-state assumption has been successfully used to reduce and analyze continuous systems.

Section 1 of this chapter is about the reduction of Boolean networks. We give a reduction algorithm that not only reduces the size of a network but also elucidates the role of the wiring diagram on the dynamics. In particular, we prove that topological features are preserved and that the number of steady states remains unchanged after reduction. Our algorithm is based on a discrete version of the quasi-steady-state assumption for continuous systems. Furthermore, we use the reduction method to study a model of Th differentiation and show that the reduced network can explain the steady states and other properties of the original model.

When there is not enough information about a regulatory network to construct a model, one has to consider the problem of reverse engineering a model or features of the model (such as the wiring diagram) from data. Using tools from algebra we can give this problem a mathematical formulation.

Section 2 of this chapter is about a reverse engineering method for regulatory networks. Our method is a generalization of the algorithm proposed in [53]. Given some data set from a network, we define in algebraic terms the set of all possible signed wiring diagrams that are compatible with such data. Then, we use tools from algebraic geometry, namely the primary

decomposition of an ideal, to find the wiring diagrams that are "minimal". We apply our results to an ODE system and show that it performs better that the algorithm in [53].

Chapter 5: Polynomial Form of Other Discrete Frameworks

Since there are many different discrete frameworks capable of modeling regulatory networks, we studied how they compare to our algebraic framework and whether or not they were compatible. We considered logical models and Petri nets, which cover the vast majority of discrete models.

In this chapter we show that logical models and Petri nets can be written as PDSs and that tools from polynomial algebra can be used to analyze their dynamical properties. This means that our results can be applied to a large number of discrete models. This section is based on apublished paper written in collaboration with Abdul Salam Jarrah and Reinhard Laubenbacher: *Polynomial algebra of discrete models in systems biology, Bioinformatics, 26:1637-1643, 2010*. My contributions to the paper were the algorithms and coding to transform logical models and Petri nets to PDSs.

Chapter 2

Algebraic Models

The *lac* operon in *Escherichia coli* has been studied extensively and is one of the earliest gene systems found to undergo both positive and negative control. The *lac* operon is known to exhibit bistability, in the sense that the operon is either induced or uninduced. Many dynamical models have been proposed to capture this phenomenon. While most are based on complex mathematical formulations, it has been suggested that for other gene systems network topology is sufficient to produce the desired dynamical behavior.

We present a Boolean network as a discrete model for the *lac* operon. Our model includes the two main glucose control mechanisms of catabolite repression and inducer exclusion. We show that this Boolean model is capable of predicting the ON and OFF steady states and bistability. Further we present a reduced model which shows that *lac* mRNA and lactose form the core of the *lac* operon, and that this reduced model exhibits the same dynamics. This work suggests that the key to model qualitative dynamics of gene systems is the topology of the network and Boolean models are well suited for this purpose.

This chapter is based on a paper written in collaboration with Brandilyn Stigler. My contributions to the paper were the construction of the model and reduced model as well as their analysis.

2.1 The Lac Operon

The *lac* operon in the bacterium *Escherichia coli* has been used as a model system of gene regulation since the landmark work by Jacob and Monod in 1961 [52]. This system of genes is responsible for the metabolism of lactose in the absence of glucose. Its study has led to numerous insights into sugar metabolism, including how the presence of a substrate could trigger induction of its catabolizing enzyme, yet in the presence of a preferred energy source, namely glucose, the substrate is rendered ineffective. Originally termed the "glucose

effect", catabolite repression became known as one of the mechanisms by which glucose regulates the induction of sugar-metabolizing operons. A second glucose-dependent control mechanism was also identified through analysis of the *lac* operon. This mechanism, called inducer exclusion, refers to the suppression of the *lac* permease protein in the presence of extracellular glucose, thereby preventing the transport of lactose into the cell. Early work on the *lac* operon led to the discovery that transcription of an operon's genes is subject to positive (activator proteins initiate transcription) or negative (repressor proteins prevent transcription) control via induction (inducer molecules bind to regulatory proteins to initiate transcription) or repression (corepressors are used to prevent transcription). The *lac* operon is one of the earliest examples of a inducible system of genes being under both positive and negative control.

Now we describe the components and features of the *lac* operon which we include in the model. This description is summarized largely from the material provided in the online book [39] (additional citations are given as necessary).

The *lac* operon contains three structural genes, *lacZ, lacY,* and *lacA*, and is a negative inducible system: the repressor protein LacI prevents transcription of the *lac* genes, and the operon is induced by allolactose, an isomer of lactose. Extracellular lactose is thought to be readily available, but can diffuse into the cell at low concentrations. Once in the cell, lactose can induce the operon, though with lower probability than allolactose. Transcription of the *lac* genes produces a single mRNA, whose translation gives rise to the following proteins: β-galactoside permease (LacY), a membrane-bound protein which transports lactose into the cell; β-galactosidase (LacZ), an intracellular protein which cleaves lactose into glucose and its stereoisomer galactose, and in a separate reaction converts lactose into allolactose; and β-galactoside transacetylase (LacA) which transfers an acetyl group from acetyl-CoA to β-galactosides.

Glucose is thought to regulate the *lac* operon through two key mechanisms: catabolite repression and inducer exclusion. In the absence of glucose, the catabolite activator protein CAP (also known as CRP for cAMP receptor protein) forms a complex with cAMP which binds to a site upstream of the *lac* promoter region. Binding of the cAMP-CAP complex makes a conformational change in the DNA, thereby allowing RNA polymerase to bind to the DNA and enhancing transcription of the *lac* genes. Transcription continues until extracellular glucose is available. However, when glucose is abundant, cAMP synthesis is inhibited [117] and the repressor protein LacI can bind to the operator region of the operon, preventing transcription of the *lac* genes. Furthermore, the transport of lactose into the cell by permease is inhibited by external glucose, a mechanism referred to as *inducer exclusion.* Finally, the presence of (sufficient amounts of) intracellular glucose shuts off the operon, a phenomenon referred to as *catabolite repression.*

2.2 Model of the Lac Operon

There are many formulations modeling the behavior and interaction of the *lac* genes. The first model was proposed by Goodwin two years after the discovery of the *lac* operon [37]. Since then there has been a steady flow of models following the advances in biological insight of the system, with the majority describing operon induction using artificial nonmetabolizable compounds such as IPTG and TMG [84; 96; 113; 114]. The first model to consider catabolite repression and inducer exclusion with cells grown in both glucose and lactose and lactose used as the inducer was presented by Wong *et al.* [117]. Their model consisted of up to 13 ordinary differential equations involving 65 parameters. Further Santillán and coauthors have presented mathematical models and analysis purporting bistability in that the operon is either induced or uninduced [94–96; 119], which has been observed experimentally [83; 84]. These findings have given rise to the analogy of the *lac* operon acting as a biological switch [43; 84].

Most mathematical models of the *lac* operon, as well as other genetic systems, are given as systems of differential equations which are appropriate for understanding mechanisms. However, discrete modeling frameworks are receiving more attention in systems biology for their use in analyzing entire state spaces and facilitating global-level predictions. In fact it has been argued that network topology and interaction type (activation/inhibition) are sufficient for capturing dynamics of gene networks (see [3; 76] for proofs of concept in other model organisms). A consequence of this observation is that the use of qualitative networks eliminates the need to estimate parameters, thereby reducing model complexity.

Finite dynamical systems (FDSs, which can be seen as PDSs), provide a computational advantage in that entire state spaces can be computed and explored analytically using methods such as generalized logical analysis [76; 111], in contrast to continuous modeling frameworks. The advantage of having the state space given explicitly is that features such as steady states and limit cycles can be essentially read from the state space. Such predictions, as well as others provided by FDSs, are often easier to verify experimentally than their continuous counterparts, due to their qualitative description; that is, precise experimental conditions are not required to verify qualitative predictions. Boolean networks (BNs), a special class of FDSs, permit an intuitive, yet formal mathematical description of network dynamics by encoding network topology and interaction type in Boolean expressions (see [23] for examples).

Currently few discrete models of the *lac* genes exist. Gianchandani and coauthors introduced a Boolean model of the *lac* operon, excluding the regulatory effects of glucose and lactose and demonstrated that an alternate method for identifying all steady states using a matrix representation of the Boolean model [32]. Setty *et al.* defined a logical function, a specialization of FDSs, for the transcription of the *lac* genes in terms of the proteins regulating the operon, namely CRP and LacI [98]. Although the authors initially aimed to construct a simple Boolean function to mimic the switching behavior of the operon, they discovered that

AND-like and OR-like expressions could not reproduce the complexity that the *lac* genes exhibited. Instead they found that a logical function on 4 states (as opposed to 2 states - 0 and 1) was more biologically relevant. Mayo *et al.* tested and showed that this logical function was robust with respect to point mutations, that is, given the formulation in [98], the operon is still functional after point mutations [73]. One limitation in both models is that they do not predict bistability.

We propose a logical model for the *lac* operon which predicts bistability and includes the two main control mechanisms of glucose, namely catabolite repression and inducer exclusion. In order to facilitate interpretation, we have added variables so as to present the model as a BN. We are able to show that the model has only two steady states, corresponding to the operon being either ON or OFF (induced or uninduced) and that the model exhibits bistability.

An important question is to identify the key players in a network, in this case for the purpose of determining the drivers of the dynamics. Aguda and Goryachev provided a systematic approach for reducing a network derived from literature to a subnetwork which can be thought of as the core of the essential qualitative behavior [1]. Albert and Othmer [3] showed that the topology and interaction type are the determining factors in producing the steady-state behavior. We corroborate these findings by considering a reduced Boolean model involving only the *lac* genes and lactose. We show that the dynamics of the reduced model matches that of the full model. Our results further support the hypothesis that the topology is the key to dynamical properties.

2.2.1 Modeling Background

A Boolean network (BN) $f = (f_1, \ldots, f_n)$ on n variables is a tuple of functions defined over the set $\{0,1\}$ such that for each $i = 1, \ldots, n$, the function f_i determines the next state of variable i and is written in terms of the Boolean operators $\vee, \wedge, \neg$ (logical OR, AND, and NOT, respectively). The values 0 and 1 are the *states* of the variables.

"Network topology" refers to the connectivity structure of a network and is typically represented as a directed graph. For a BN, a *wiring diagram* is a directed graph on the variables of the system (in this case, mRNA, proteins, and sugars) with edges defined in the following way: there is a directed edge from variable x_i to x_j if the function f_{x_j} for x_j depends on x_i. An edge from x_i to x_j has a circle at its head x_j if NOT x_i, denoted by $\neg x_i$, appears in f_{x_j} (we consider this edge to correspond to an inhibitory interaction); otherwise, edges have arrows at their heads. We call directed cycles *feedback loops*. The *parity* of a feedback loop (or a path) can be either +1 or -1 and is calculated as follows: assign -1 to an edge if it is inhibitory and +1 otherwise. The parity of a feedback loop is the product of the signs on the edges of the loop. If the parity of a feedback loop is +1, we call the loop *positive*; otherwise, it is *negative*.

"Dynamics" refers to the state transitions of the network as a whole. To generate the

dynamics of a BN f on n variables, we evaluate its functions on all possible combinations of 0/1 n-tuples. The dynamics can be viewed as a directed graph, called the *state space* of f. In this graph each node is an n-tuple, called a *state of the system* f; there is a directed edge from a to b if f evaluated at the current state a gives state b; that is, if $f(a) = b$. Hence, b represents that next state of the system. Directed cycles are called *limit cycles*. If length of the cycle is 1, then it is called a *fixed point*. In the context of scientific applications, fixed points are also referred to as *steady states*, which we use in this discourse. We draw the state space using the visualization software DVD [55].

2.2.2 Boolean Network

In this subsection we present the Boolean network (BN) for the *lac* operon that models gene regulation such as the two main control mechanisms of glucose, namely catabolite repression and inducer exclusion. The BN consists of variables and functions, each representing mRNAs, proteins and sugars. We assume that each biomolecule can be either 0 or 1 (absent/inactive or present/active). In order to perform a comparison with an ODE model we considered that extracellular lactose can be in three states (low/0, medium/1, high/2) which model using the variables L_e and L_{em}; they indicate high concentration and medium concentration of extra cellular lactose, respectively. $(L_e, L_{em}) = (0,0), (0,1), (1,1)$ means that the concentration of extracellular lactose is low, medium and high, respectively. The same was done for lactose, allolactose and the repressor protein.

The Boolean variables are labeled as follows:

- M = *lac* mRNA
- P, B = *lac* permease and β-galactosidase, respectively
- C = catabolite activator protein CAP
- R = repressor protein LacI
- R_m = (at least) medium concentration of repressor protein LacI
- L, A= lactose and allolactose (inducer), respectively
- L_m, A_m = (at least) medium concentration of lactose and allolactose, respectively
- L_e, G_e= extracellular lactose, glucose, respectively
- L_{em}= (at least) medium concentration of extracellular lactose.

Deriving Boolean Networks

Here we illustrate how to derive a Boolean function for mRNA based on the information in Section 2.1. The operator $\wedge$ indicates that both variables need to be present, that is, activation occurs by the synergistic action of its regulators; on the other hand, the operator $\vee$ indicates that either variable is enough for activation to occur, that is, activation occurs independently of the other variable. The operator $\neg$ indicates that the variable is a repressor and that it has to be absent for activation to occur. For any variable a, the function F_a determines the value of a after one time unit.

Boolean function for M: To have mRNA being transcribed we need the activator protein (CAP), but also the repressor to be absent. That is, the future Boolean value of M is given by the presence of C AND the ABSENCE of R AND the ABSENCE of R_m. Hence, the Boolean function for M is $F_M = C \wedge \neg R \wedge \neg R_m$.

Boolean functions for P, B: To have the lac proteins being produced we need mRNA. Hence, the Boolean function for P, B are $F_P = F_B = M$.

Boolean function for C: To have CAP in high concentrations we need glucose to be absent. That is, the future Boolean value of C is given by the ABSENCE of G_e. Hence, the Boolean function for C is $F_C = \neg G_e$.

Boolean function for R: The repressor will be active as long as there is no allolactose. That is, the future Boolean value of R is given by the ABSENCE of A AND the ABSENCE of A_m. Hence, the Boolean function for R is $F_R = \neg A \wedge \neg A_m$.

Boolean function for R_m: The repressor will be active at (at least) a medium level as long as there is no allolactose or if the repressor was active to begin with. That is, the future Boolean value of R_m is given by {the ABSENCE of A AND the ABSENCE of A_m} OR the presence of R. Hence, the Boolean function for R is $F_{R_m} = (\neg A \wedge \neg A_m) \vee R$.

Boolean function for A: To have allolactose in high concentrations we need both, β-galactosidase and lactose to be present. That is, the future Boolean value of A is given by the presence of B AND L. Hence, the Boolean function for A is $F_A = B \wedge L$.

Boolean function for A_m: To have allolactose in medium concentrations we need lactose to be present in at least a medium concentration (it will be transformed to allolactose by a basal number of β-galactosidase molecules which we are not including in the model). That is, the future Boolean value of A_m is given by the presence of L OR L_m. Hence, the Boolean function for A_m is $F_{A_m} = L \wedge L_m$.

Boolean function for L: To have lactose in high concentrations we need both, permease and extracellular lactose to be present, but also glucose to be absent. That is, the future Boolean value of L is given by the presence of P AND L_e and the ABSENCE of G_e. Hence, the Boolean function for L is $F_L = P \wedge L_e \wedge \neg G_e$.

Boolean function for L_m: To have lactose in medium concentrations we need extracellular

lactose to be present in at least a medium concentration together with permease, or extracellular lactose to be present in concentrations high enough so that it will enter the cell (by diffusion or a basal number of permease molecules which we are not including in the model); we also need glucose to be absent. That is, the future Boolean value of L_m is given by the {{presence of L_{em} AND P} OR the presence of L_e} AND the ABSENCE of G_e. Hence, the Boolean function for L_m is $F_{L_m} = ((L_{em} \wedge P) \vee L_e) \wedge \neg G_e$.

Therefore, the BN is given by:

$$\begin{array}{llll} F_M = C \wedge \neg R \neg R_m & , & F_P = M \\ F_B = M & , & F_C = \neg G_e \\ F_R = \neg A \wedge \neg A_m & , & F_{R_m} = (\neg A \wedge \neg A_m) \vee R \\ F_A = L \wedge B & , & F_{A_m} = L \vee L_m \\ F_L = P \wedge L_e \wedge \neg G_e & , & F_{L_m} = ((L_{em} \wedge P) \vee L_e) \wedge \neg G_e \end{array}$$

where L_e and G_e (lactose and glucose) are considered as fixed parameters in the model. We use F to refer to the model consisting of this BN.

Network Topology

The wiring diagram for the model F is shown in Figure 2.1 (L_e, L_{em} are shown as a single node, as well as R, R_m, A, A_m and L, L_m). We can identify topological features such as the feedback loops in F. We see that there are positive feedback loops involving M, e.g. $M \to P \to L \to A \to R \to M$. Also, we can see positive paths from L_e to M and negative paths from G_e to M. Note that there are no negative feedback loops.

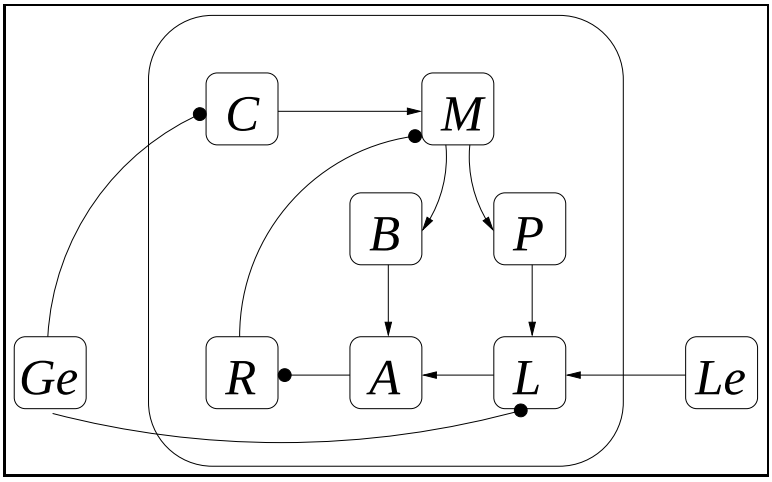


Figure 2.1: Wiring diagram of F.

Dynamics and Bistability

The dynamics of F can be computed by evaluating the functions on all possible combinations of vectors $(M, P, B, C, R, R_m, A, A_m, L, L_m)$ with 0-1 entries (see Section 2.2.2 for more details). We say that the operon is OFF when the value of the triple $(\mathbf{M}, \mathbf{P}, \mathbf{B})$ is $(\mathbf{0}, \mathbf{0}, \mathbf{0})$ and ON when $(\mathbf{M}, \mathbf{P}, \mathbf{B}) = (\mathbf{1}, \mathbf{1}, \mathbf{1})$. For simplicity we will focus on the steady states.

When *glucose* is high ($G_e = 1$) we have the unique steady state $(\mathbf{0}, \mathbf{0}, \mathbf{0}, 0, 1, 1, 0, 0, 0, 0)$ corresponding to the lac operon being OFF (all initializations eventually reach this steady state). On the other hand if glucose is low ($G_e = 0$) we have the following cases:

1. For low/0 extracellular lactose (that is, $L_e = L_{em} = 0$), there is only one steady state, (**0**,**0**,**0**,1,1,1,0,0,0,0) that corresponds to the operon being OFF (all of the 2^{10} initializations eventually reach this steady state).
2. For medium/1 extracellular lactose (that is, $L_e = 0, L_{em} = 1$), there are two steady states, (**0**,**0**,**0**,1,1,1,0,0,0,0) that corresponds to the operon being OFF and (**1**,**1**,**1**,1,0,0,0,1,0,1), that corresponds to the operon being ON (all of the 2^{10} initializations eventually reach one of the steady states). That is, the model is bistable.
3. For high/2 extracellular lactose (that is, $L_e = L_{em} = 1$), there is only one steady state, (**1**,**1**,**1**,1,0,0,1,1,1,1) that corresponds to the operon being ON (all of the 2^{10} initializations eventually reach this steady state).

In summary, the model predicts that the *lac* operon is OFF when extracellular glucose is available. When glucose is not available, the model predicts that the operon is OFF, bistable or ON when extracellular lactose is low, medium or high, respectively. This is consistent with the reports of bistability, see for example [84; 94]. The comparison of our Boolean model with ODE models can be illustrated with the bifurcation diagram shown in Figure 2.2. This figure shows that both models have the same qualitative behavior.

Experiments

We also performed experiments similar to those in [84]. In order to do this we considered a stochastic version of our model and we considered glucose to be absent ($G_e = 0$):

$$\begin{array}{llll} \mathcal{F}_M & = C \wedge \neg R \neg R_m & \mathcal{F}_P & = M \\ \mathcal{F}_B & = M & \mathcal{F}_C & = 1 \\ \mathcal{F}_R & = \neg A \wedge \neg A_m & \mathcal{F}_{R_m} & = (\neg A \wedge \neg A_m) \vee R \\ \mathcal{F}_A & = L \wedge B & \mathcal{F}_{A_m} & = L \vee L_m \\ \mathcal{F}_L & = P \wedge L_e & \mathcal{F}_{L_m} & = (L_{em} \wedge P) \vee L_e \end{array}$$

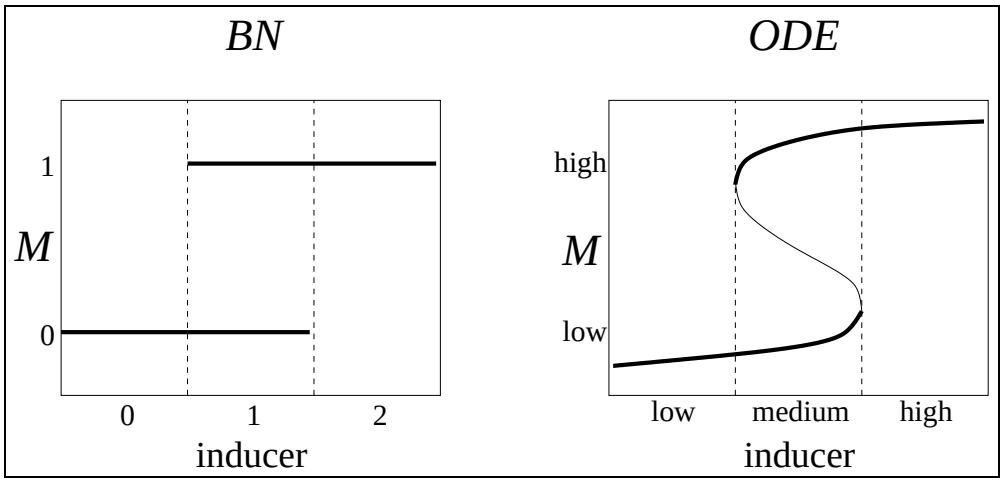


Figure 2.2: Bifurcation diagram of our Boolean network and a typical bifurcation diagram showing bistability of an ODE (two saddle-node bifurcation).

where extracellular lactose is given as the discretization of a random variable, $\mathcal{L}_e \sim N(\mu, \sigma)$ taken from a normal distribution with mean μ and variance σ^2. Then, extracellular lactose (L_e) is given as a function of $\mathcal{L}_e$ defined by

$$\text{Extracellular lactose} = \begin{cases} \text{low}/0 & (\text{i.e. } L_e = L_{em} = 0), & \text{if } \mathcal{L}_e < 1 \\ \text{medium}/1 & (\text{i.e. } L_e = 0, L_{em} = 1), & \text{if } 1 \leq \mathcal{L}_e < 2 \\ \text{high}/2 & (\text{i.e. } L_e = L_{em} = 1), & \text{if } 2 \leq \mathcal{L}_e \end{cases}$$

The results of our experiments are in Figure 2.3. We can see the same qualitative behavior of the experiments performed in [84, Figure 2.b]. We can see that as we decrease the concentration of the inducer (top panel) the population of cells start turning their operon OFF; also, as we increase the concentration of the inducer (bottom panel) the population of cells start turning their operon ON.

2.2.3 Core Subnetworks

An important question is whether the dynamical properties of the model is a direct consequence of its topology and the type of interactions in the model. If network topology and interaction type are sufficient for maintaining certain dynamical properties, we expect a reduced model to have dynamics equivalent to the original model. This would suggest that the qualitative dynamical properties of the *lac* operon are dependent on intrinsic characteristics such as topological features and interaction type and not on extrinsic characteristics such as the specific proteins, sugars involved or the size of the network. This in turn implies that in order to obtain an accurate qualitative description of the *lac* operon we need to focus

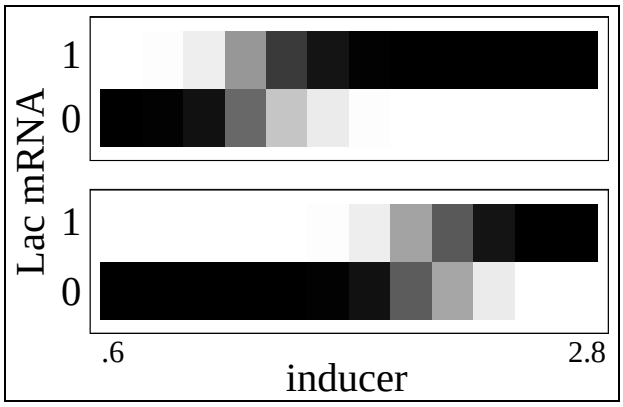


Figure 2.3: Discrete version of the experiments performed in [84] for our Boolean network.

on using the correct network topology and interaction type. The importance of network topology and interaction type has been proposed by Albert and Othmer [3], among others, and studied by Aguda [1]. Alternatively, if the dynamics of the model is highly dependent on the size, the model would not be useful to study the dynamics of the biological system: changing the size of the network could significantly alter the predictions.

We provide a reduced network that retains topological features, such as feedback loops, present in the original model. Since many existing models with bistable behavior include glucose and lactose, we will construct a reduced model using M, L, L_e, G_e only. We denote the reduced model by f.

Boolean function for M: To have mRNA being transcribed we need the activator protein (CAP) which will be present if glucose is absent; we also need the repressor to be absent which will happen if there is lactose in the cell. That is, the future Boolean value of M is given by the ABSENCE of G_e AND {the presence of L_m OR L}. Hence, the Boolean function for M is $f_M = \neg G_e \wedge (L \vee L_m)$.

Boolean function for L: To have lactose in high concentrations we need both, permease (which means we need mRNA) and extracellular lactose to be present, but also glucose to be absent. That is, the future Boolean value of L is given by the presence of P AND L_e and the ABSENCE of G_e. Hence, the Boolean function for L is $f_L = M \wedge L_e \wedge \neg G_e$.

Boolean function for L_m: To have lactose in medium concentrations we need extracellular lactose to be present in at least a medium concentration together with permease (which means we need mRNA), or extracellular lactose to be present in concentrations high enough so that it will enter the cell; we also need glucose to be absent. That is, the future Boolean value of L_{em} is given by the {{presence of L_{em} AND M} OR the presence of L_e} AND the ABSENCE of G_e. Hence, the Boolean function for L_m is $f_{L_m} = ((L_{em} \wedge M) \vee L_e) \wedge \neg G_e$.

Therefore, the reduced BN is given by:

$$\begin{aligned} f_M &= \neg G_e \wedge (L \vee L_m) \\ f_L &= M \wedge L_e \wedge \neg G_e \\ f_{L_m} &= ((L_{em} \wedge M) \vee L_e) \wedge \neg G_e \end{aligned}$$

Network Topology

The wiring diagram for the model f is shown in Figure 2.4 (L_e, L_{em} are shown as a single node, as well as L, L_m). From the reduced model, we can clearly see the roles of the parameters on the core subnetwork whose size is 30% of that of F. We can see that the paths from G_e and L_e to M are still present in the model; also, the signs of these paths have not changed. G_e inhibits M and L (catabolite repression and inducer exclusion, respectively) whereas L_e activates L. Furthermore, we still have a positive feedback loop involving M.

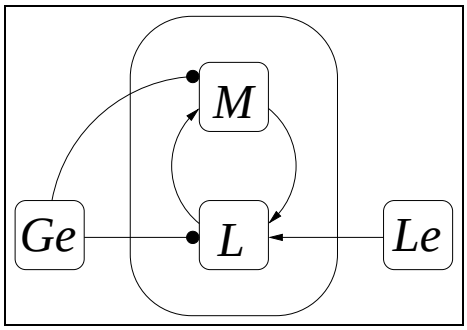


Figure 2.4: Wiring diagram of f.

Dynamics

The dynamics of f can be computed by evaluating the functions on all possible combinations of vectors (M, L, L_m) with 0-1 entries. We say that the operon is OFF when the value of $\mathbf{M}$ is $\mathbf{0}$ and ON when $\mathbf{M} = \mathbf{1}$.

When *glucose* is high ($G_e = 1$) we have the unique steady state $(\mathbf{0}, 0, 0)$ corresponding to the lac operon being OFF (all initializations eventually reach this steady state). On the other hand if glucose is low ($G_e = 0$) we have the following cases:

1. For low/0 extracellular lactose, there is a single steady state, ($\mathbf{0}$,0,0), that corresponds to the operon being OFF (all initializations eventually reach this steady state).
2. For medium/1 extracellular lactose, there are two steady states, ($\mathbf{0}$,0,0), that corresponds to the operon being OFF and ($\mathbf{1}$,0,1), that corresponds to the operon being ON (all initializations eventually reach one of the steady states). That is, the model is bistable.

3. For high/2 extracellular lactose, there is a single steady state, (**1**,1,1), that corresponds to the operon being ON (all initializations eventually reach this steady state).

In summary, the model predicts that the *lac* operon is OFF when extracellular glucose is available. When glucose is not available, the model predicts that the operon is OFF, bistable or ON when extracellular lactose is low, medium or high, respectively. Hence, the reduced network has the same dynamics as the full network. We observe that the reduced model has only one positive feedback loop, whereas the model F has several more. Since the reduced model still exhibits bistability, this suggests that it does not depend on the number of positive feedback loops but simply on the existence of such a loop.

It is important to notice that the state space of the reduced model has 2^3 states which is about 0.8% of the size of the state space of F, that is, 2^{10} states. We also considered models with different numbers of variables, ranging from 4 to 9 vertices (plus 2 parameters), and obtained the same results. Hence, we conclude that the dynamical properties of the model are independent of its size.

2.2.4 Alternative Models

Since it has been argued that inducer exclusion is not necessary for bistability [113], we considered alternative models without inducer exclusion and without catabolite repression (Figure 2.5) and bistability was also observed.

Model without inducer inclusion

$$\begin{array}{llll} F_M &= C \wedge \neg R \neg R_m & F_P &= M \\ F_B &= M & F_C &= \neg G_e \\ F_R &= \neg A \wedge \neg A_m & F_{R_m} &= (\neg A \wedge \neg A_m) \vee R \\ F_A &= L \wedge B & F_{A_m} &= L \vee L_m \\ F_L &= P \wedge L_e & F_{L_m} &= (L_{em} \wedge P) \vee L_e \end{array}$$

Model without catabolite repression

$$\begin{array}{llll} F_M &= C \wedge \neg R \neg R_m & F_P &= M \\ F_B &= M & F_C &= 1 \\ F_R &= \neg A \wedge \neg A_m & F_{R_m} &= (\neg A \wedge \neg A_m) \vee R \\ F_A &= L \wedge B & F_{A_m} &= L \vee L_m \\ F_L &= P \wedge L_e \wedge \neg G_e & F_{L_m} &= ((L_{em} \wedge P) \vee L_e) \wedge \neg G_e \end{array}$$

Furthermore, their reduced versions (Figure 2.6) also showed bistability.

Reduced model without inducer inclusion

$$\begin{array}{ll} f_M &= \neg G_e \wedge (L \vee L_m) \\ f_L &= M \wedge L_e \\ f_{L_m} &= (L_{em} \wedge M) \vee L_e \end{array}$$

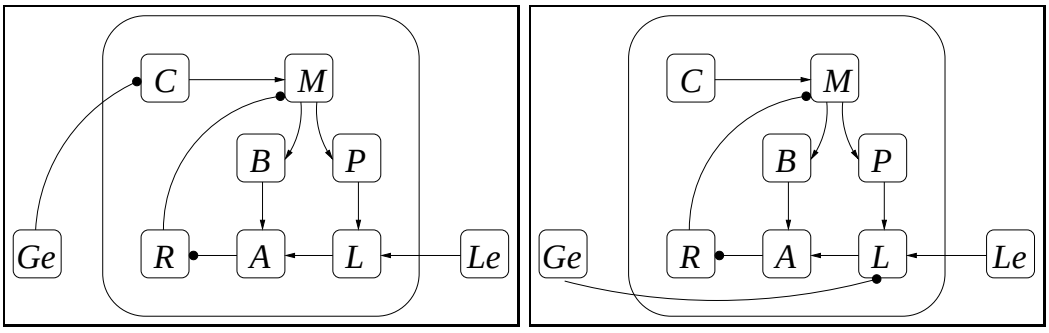


Figure 2.5: Wiring diagram of the model without inducer inclusion and without catabolite repression.

Reduced model without catabolite repression

$$\begin{aligned} f_M &= L \vee L_m \\ f_L &= M \wedge L_e \wedge \neg G_e \\ f_{L_m} &= ((L_{em} \wedge M) \vee L_e) \wedge \neg G_e \end{aligned}$$

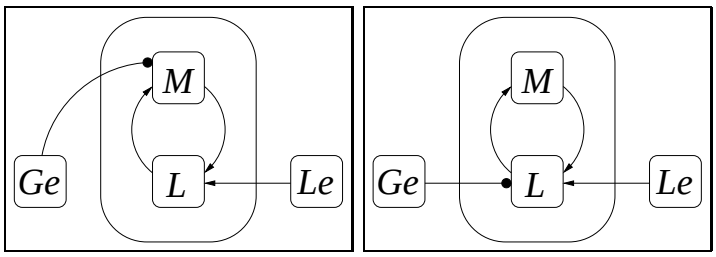


Figure 2.6: Wiring diagram of the reduced model without inducer inclusion and without catabolite repression.

2.3 Discussion

Many authors have studied the problem of inferring dynamical properties of a system from the network structure [56; 100]. Furthermore, for special classes of BNs and ODEs it has been proven that the network structure contains all the information needed to determine certain dynamical properties [43; 56]. On the other hand, it has been claimed that network topology and sign of interactions are more important than quantitative functionality of the components of a system [3]. To test this hypothesis, we applied the ideas in [3] to lactose metabolism.

The contributions in this work are twofold. First, we proposed a BN as a discrete model for the *lac* operon and included the glucose control mechanisms of catabolite repression and inducer exclusion. We showed that our model exhibits the ON/OFF switching dynamics and

that bistability is also observed, in accordance with the work of Santillán and coauthors [84; 94–96; 119]. Since it has been argued that inducer exclusion is not necessary for bistability [113], we considered alternative models without inducer exclusion and bistability was also observed.

Second, we presented a reduced model which shows that *lac* mRNA and lactose form the core of the *lac* operon, and that this reduced model also exhibits the same dynamics. This result suggests that the presence of positive feedback loops is a better indicator of dynamics than the individual constituents that comprise the feedback loops. Taken together, this work supports the idea that topology is the driver of dynamics [3; 76].

A future work may be to extend this model to a multi-state framework, which has the potential to provide a more refined qualitative description of the *lac* operon. Such a framework may allow for inclusion of other features of the operon, such as multiple promoter and operator regions.

Chapter 3

Prediction of Dynamics from Network Structure

3.1 Effect of Negative Feedback Loops on the Dynamics of BNs

Feedback loops play an important role in determining the dynamics of biological networks. Moreover, it is widely believed [101] that an abundance of negative loops should result in the existence of "chaotic" behavior in the network. This section provides strong evidence in support of this latter conjecture. In order to study the role of negative feedback loops, we introduce the notion of "distance to positive feedback (PF-distance)" which in essence captures the number of "independent" negative feedback loops in the network, a property inherent in the network topology. We focus here on Boolean network (BN) models, a popular model type for biochemical networks, initially introduced by S. Kauffman [58]. In particular, we study BN models in which each directed edge can be characterized as either an inhibition or an activation. Through a computational study it is shown that PF-distance has a strong influence on network dynamics and correlates very well with the number and length of limit cycles in the phase space of the network. To be precise, it is shown that, as the number of independent negative feedback loops increases, the number (length) of limit cycles tends to decrease (increase). These conclusions are consistent with the fact that certain natural biological networks exhibit generally regular behavior and have fewer negative feedback loops than randomized networks with the same numbers of nodes and connectivity.

Boolean networks constructed from monotone Boolean functions (i.e. each node or "gate" computes a function which is increasing on all arguments) are of particular interest, and have been studied extensively, in the electronic circuit design and pattern recognition literature [33; 78], as well as in the computer science literature; see e.g. [8; 9; 34] for recent references. For Boolean and all other finite iterated systems, all trajectories must either settle into

equilibria or to periodic orbits, whether the system is made up of monotone functions or not, but monotone networks have always somewhat shorter cycles. This is because periodic orbits must be anti-chains, i.e. no two different states can be compared; see [33; 99]. An upper bound may be obtained by appealing to Sperner's Theorem ([5]): Boolean systems on n variables can have orbits of period up to 2^n, but monotone systems cannot have orbits of size larger than $\binom{n}{\lfloor n/2 \rfloor} \approx 2^n \sqrt{\frac{2}{n\pi}}$; these are all classical facts in Boolean circuit design [33]. It is also known that the upper bound is tight [33], in the sense that it is possible to construct Boolean systems on n variables, made up of monotone functions, for which orbits of the maximal size $\binom{n}{\lfloor n/2 \rfloor}$ given by Sperner's Theorem exist. This number is still exponential in n. However, anecdotal experience suggests that monotone systems constructed according to reasonable interconnection topologies and/or using restricted classes of gate functions, tend to exhibit shorter orbits [38; 112]. One may ask if the *architecture* of the network, that is, the structure of its wiring diagram, helps ensure shorter orbits. In this direction, the paper [8] showed that on certain graphs, called there "caterpillars", monotone networks can only have cycles of length at most two in their phase space.

We ask the even more general question of whether networks that are not necessarily made up from monotone functions, but which are "close to monotone" (in a sense to be made precise, roughly meaning that there are few independent negative loops) have shorter cycles than networks which are relatively farther to monotone.

In [101], it was conjectured that "smaller distance to monotone" should correlate with more ordered (less "chaotic") behavior, for random Boolean networks. A partial confirmation of this conjecture was provided in [66], where the relationship between the dynamics of random Boolean networks and the ratio of negative to positive feedback loops was investigated, albeit only for the special case of small Kauffman-type NK and NE networks, and with the additional restriction that all nodes have the same function chosen from AND, OR, or UNBIAS. Based on computer simulations, the authors of [66] found a positive (negative) correlation between the ratio of fixed points (other limit cycles) and the ratio of positive feedback loops. Observe that this differs from our conjecture in two fundamental ways: (1) our measure of disorder is related to the number of "independent" negative loops, rather than their absolute number, and (2) we do not consider that the number of positive loops should be part of this measure: a large number of negative loops will tend to produce large periodic orbits, even if the negative to positive ratio is small due to a larger number of positive loops.

Thus, in the spirit of the conjecture in [101], our goal is to study of the effect of independent negative feedback loops on network dynamics, based on an appropriately defined measure of *distance to positive-feedback*. We study the effect of this distance on features of the network dynamics, namely the number and length of limit cycles. Rather than focusing on the number of negative feedback loops in the network, as the characteristic feature of a network, we focus on the number of switches of the activation/inhibition character of edges that need to be made in order to obtain a network that has only positive feedback loops. We relate

this measure to the cycle structure of the phase space of the network.

There are three different motivations for posing the question that we ask in this section. The first is that most biological networks appear to have highly regular dynamical behavior, settling upon simple periodic orbits or steady states. The second motivation is that it appears that real biological networks such as gene regulatory networks and protein signaling networks are indeed close to monotone [21; 71; 102]. Thus, one may ask if being close to monotone correlates in some way with shorter cycles. Unfortunately, as mentioned above, one can build networks that are monotone yet exhibit exponentially long orbits. This suggests that one way to formulate the problem is through a statistical exploration of graph topologies, and that is what we do here. A third motivation arises from the study of systems with continuous variables, which arguably provide more accurate models of biochemical networks. There is a rich theory of continuous-variable monotone (to be more precise, "cooperative") systems. These are systems defined by the property that an inequality $\mathbf{a}(0) < \mathbf{b}(0)$ in initial conditions propagates in time so that the inequality $\mathbf{a}(t) < \mathbf{b}(t)$ remains true for all future times $t > 0$. Note that this is entirely analogous to the Boolean case, when one makes the obvious definition that two Boolean vectors satisfy the inequality $\mathbf{a} = (a_1, \ldots, a_n) \leq \mathbf{b} = (b_1, \ldots, b_n)$ if $a_i \leq b_i$ for each $i = 1, \ldots, n$ (setting $0 < 1$). Monotone continuous systems have convergent behavior. For example, in continuous-time (ordinary differential models), they cannot admit any possible stable oscillations [42; 49; 50], and, when there is only one steady state, every bounded solution converges to this unique steady state (monostability), see [20]. When, instead, there are multiple steady-states, the Hirsch Generic Convergence Theorem [47–49; 99] is the fundamental result; it states, under an additional technical assumption ("strong" monotonicity) that generic bounded solutions must converge to the set of steady states. For discrete-time strongly monotone systems, generically also stable oscillations are allowed besides convergence to equilibria, but no more complicated behavior. In neither case, discrete-time or continuous-time continuous monotone systems, one observes "chaotic" behavior. It is an open question whether continuous systems that are in some sense close to being monotone have more regular behavior, in a statistical sense, than systems that are far from being monotone, just as for the Boolean analog considered in this section. The Boolean case is more amenable to computational exploration than continuous-variable systems, however. Since long orbits in discrete systems may be viewed as an analog of chaotic behavior, we focus on lengths of orbits.

This section is based on a published paper written in collaboration with Eduardo Sontag, Reinhard Laubenbacher and Abdul Salam Jarrah [100]. My contributions to the paper were the design of the experiments, algorithms, coding and the analysis of the results.

3.1.1 Distance to positive-feedback

One can proceed in several ways to define precisely the meaning of distance to positive feedback. One associates to a network made of unate (definition below) Boolean functions

a signed graph whose edges have signs (positive or negative) that indicate how each variable affects each other variable (activation or inhibition). The first definition, explored in [21; 51; 101; 102] starts from the observation that in a network with all monotone node functions there are no negative undirected cycles. Conversely, if the dependency graph has no undirected negative parity cycles (a "sign-consistent" graph), then a change of coordinates (globally replacing a subset of the variables by their complements) renders the overall system monotone. Thus, asking what is the smallest number of sign-flips needed to render a graph sign-consistent is one way to define distance to monotone. This approach makes contact with areas of statistical physics (the number in question amounts to the ground energy of an associated Ising spin-glass model), as well as with the general theory of graph-balancing for signed graphs [120] that originated with Harary [44]. It is also consistent with the generally accepted meaning of "monotone with respect to some orthant order" in the ODE literature as a system that is cooperative under some inversion of variables.

A second, and different, definition, starts from the fact that a network with all monotone node functions has, in particular, no negative-sign *directed* loops. For a strongly connected graph, the property that no directed negative cycles exist is equivalent to the property that no undirected negative cycles exist. However, for non-strongly connected graphs, the properties are not the same. Thus, this second property is weaker. The second property is closer to what biologists and engineers mean by "not having negative feedbacks" in a system, and hence is perhaps more natural for applications. In addition, it is intuitively clear that negative feedbacks should be correlated to possible oscillatory behavior. (This is basically Thomas' conjecture. See [102] for precise statements for continuous-time systems; interestingly, published proofs of Thomas' conjecture use the first definition, because they appeal to results from monotone dynamical systems.) Thus, one could also define distance to monotone as the smallest number of sign-flips needed to render a graph free of negative directed loops. To avoid confusion, we will call this notion, which is the one studied in this section, *distance to positive-feedback*, or just "PF-distance".

We give here the basic definitions of the concepts relevant to the study.

Definition 3.1.1 *Let $k = \{0, 1\}$ be the field with two elements. We order the two elements as $0 < 1$. This ordering can be extended to a partial ordering on k^n by comparing vectors coordinate-wise in the lexicographic ordering.*

1. *A Boolean function $h : k^n \longrightarrow k$ is monotone if, whenever $\mathbf{a} \leq \mathbf{b}$ coordinate-wise, for $\mathbf{a}, \mathbf{b} \in k^n$, then $h(\mathbf{a}) \leq h(\mathbf{b})$. Equivalently, a Boolean function is monotone if it can written using only the operators $\wedge$ and $\vee$.*

2. *A Boolean function h is unate if, whenever x_i appears in h, the following holds: Either*

   (a) *For all $a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n \in k$,*
   $h(a_1, \ldots, a_{i-1}, 0, a_{i+1}, \ldots, a_n) \leq h(a_1, \ldots, a_{i-1}, 1, a_{i+1}, \ldots, a_n)$ (that is, x_i increases the value of h or is an activator), or

(b) For all $a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n \in k$,
$h(a_1, \ldots, a_{i-1}, 0, a_{i+1}, \ldots, a_n) \geq h(a_1, \ldots, a_{i-1}, 1, a_{i+1}, \ldots, a_n)$ *(that is,* x_i *decreases the value of* h *or is an inhibitor).*

The definition of unate function is equivalent to requiring that whenever a_i *appears in* h*, then it appears either everywhere as* a_i *or everywhere as* $\neg a_i := 1 + a_i$*. Also, every unate function,* h*, is of the form* $h(x_1, \ldots, x_n) = f(s_1 x_1, \ldots, s_n x_n)$ *where* f *is a monotone function and* $s_i x_i = x_i$ *(and we say that* x_i *is a positive input or an activator) or* $s_i x_i = \neg x_i$ *(and we say that* x_i *is a negative input or an inhibitor).*

Let f be a Boolean network with variables $x_1, \ldots, x_n$, and coordinate functions $f_1, \ldots, f_n$. That is, $f = (f_1, \ldots, f_n) : k^n \longrightarrow k^n$. We can associate to f its *dependency graph* $\mathcal{D}(f)$: The vertices are $v_1, \ldots, v_n$, corresponding to the variables $x_1, \ldots, x_n$, and there is an edge $v_i \to v_j$ if and only if x_i appears in f_j. If all coordinate functions f_i of f are unate, then the dependency graph of f is a signed graph. Namely, we associate to an edge $v_i \to v_j$ a "+" if f_j preserves the ordering as in 2(a) of Definition 3.1.1 (that is, if it is an activator) and a "-" if it reverses the ordering as in 2(b) of Definition 3.1.1 (that is, if it is an inhibitor). For later use we observe that this graph (as any directed graph) can be decomposed into a collection of strongly connected components, with edges between strongly connected components going one way but not the other. (Recall that a strongly connected directed graph is one in which any two vertices are connected by a directed path.) That is, the graph can be represented by a partially ordered set in which the strongly connected components make up the elements and the edge direction between components determines the order in the partially ordered set.

Definition 3.1.2 *Let* f *be a Boolean network with unate Boolean functions and* $\mathcal{D}(f)$ *be its signed dependency graph. Then*

1. f *is a* positive-feedback network (PF) *if* $\mathcal{D}(f)$ *does not contain any odd parity directed cycles. (The parity of a directed cycle is the product of the signs of all the edges in the cycle.)*
2. *The* PF-distance *of* f *is the smallest number of signs that need to be changed in the dependency graph to obtain a PF network. We denote this number by* $|\mathcal{D}(f)|$ *or simply* $|f|$.

Notice that for a given directed graph G, different assignments of sign to the edges produce graphs with varying PF-distance. In particular, there is a maximal PF-distance that a given graph topology can support.

The dynamics of f are presented in a directed graph, called the *phase space* of f, which has the 2^n elements of k^n as a vertex set, and there is an edge $\mathbf{a} \to \mathbf{b}$ if $f(\mathbf{a}) = \mathbf{b}$. It

is straightforward to see that each component of the phase space has the structure of a directed cycle, a *limit cycle*, with a directed tree feeding into each node of the limit cycle. The elements of these trees are called *transient states*.

In this section we relate the dynamics of a Boolean network to its PF-distance. The following is a motivational example that explains the main results.

3.1.2 Example

Example 3.1.3 *Let G be the directed graph depicted in Figure 3.1. It is easy to check that the maximal PF-distance of G is 3.*

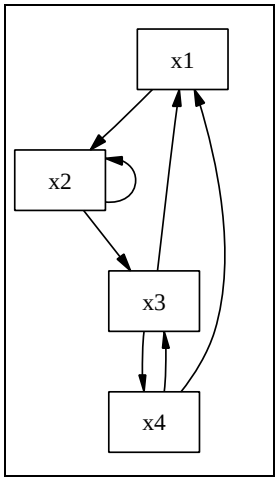


Figure 3.1: An unsigned directed graph.

Let $f = (x_3 \vee \neg x_4, x_1 \wedge x_2, x_2 \wedge \neg x_4, \neg x_3) : \{0,1\}^4 \longrightarrow \{0,1\}^4$ and $g = (\neg x_3 \vee x_4, x_1 \wedge \neg x_2, x_2 \wedge x_4, \neg x_3) : \{0,1\}^4 \longrightarrow \{0,1\}^4$. It is clear that f and g are sign-modifications of the same PF network $(x_3 \vee x_4, x_1 \wedge x_2, x_2 \wedge x_4, x_3)$, in particular, they have the same (unsigned) dependency graph. However, the PF-distance of f is 0 while it is 3 for g. The phase space of f is depicted in Figure 3.2 and that of g on Figure 3.3. Notice that f has two limit cycles of lengths 1 and 2, respectively, while g has only one limit cycle of length 4.

For each distance $0 \leq d \leq 3$, we analyze the dynamics of 10 random unate networks and their sign modifications of distance d on the directed graph in Figure 3.1. The average of the numbers (lengths) of limit cycles is computed as in Table 3.1. The best fit-line of the averages of the number (length) of limit cycles is computed and its slope is reported as in Fig. 3.4. The details of this analysis are provided in the Supporting Information *of [100].*

We repeated the experiment in Example 3.1.3 above for many different graphs and observed that the slope of the best fit-line of the length (respectively, number) of limit cycles is positive

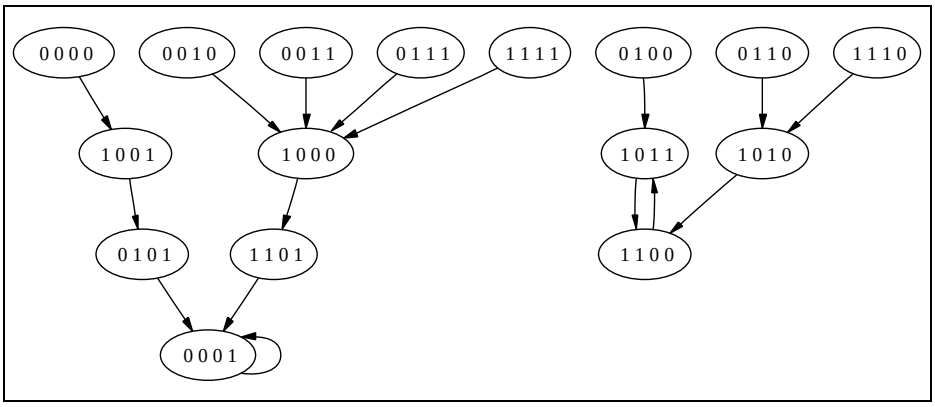


Figure 3.2: Phase space of f.

Table 3.1: The average of the numbers (lengths) of limit cycles of the networks from Example 3.1.3.

d	Av. Num.	Av. Len.
0	3.5	1.23
1	2.80	1.25
2	2.50	1.52
3	1.20	3.50

(respectively, negative) most of the time. In the methods section we present the details of the experiment and the algorithms used in the computations. The results of these experiments are described next.

3.1.3 Method

The main results of this section relate the PF-distance of Boolean networks with the number and length of their limit cycles. Specifically, our hypothesis is that, for Boolean networks consisting of unate functions, *as the PF-distance increases, the total number of limit cycles decreases on average and their average length increases.* This is equivalent to saying that for most or all experiments this slope is negative for the number of limit cycles and is positive for their length.

To test this hypothesis we analyzed the dynamics of more than six million Boolean networks arranged in about 130,000 experiments on random graphs with the number of nodes 5, 7, 10, 15, 20, or 100 and maximum in-degree 5 for each node.

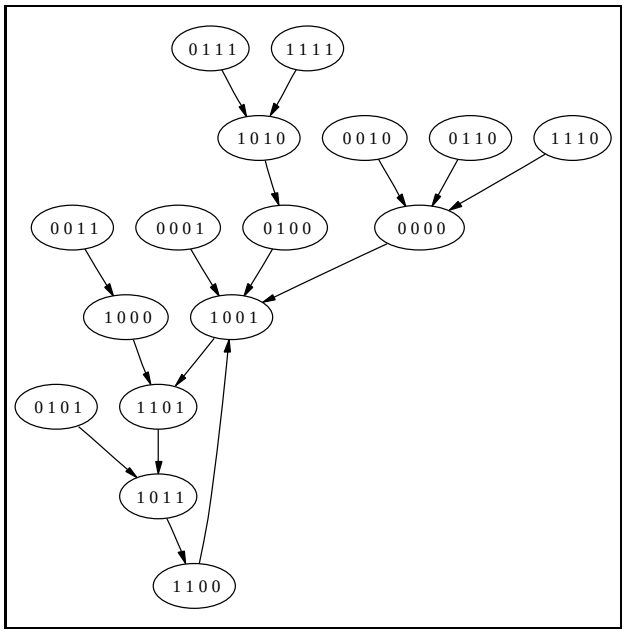


Figure 3.3: Phase space of g.

Random generation of unate functions

We generated a total of more than 130,000 random directed graphs, where each graph has 5,7,10,15, 20, or 100 nodes, with maximum in-degree 5 for each node. The graphs were generated as random adjacency matrices, with the restriction that each row has at least one 1 and at most five 1's. For each row, a number $1 \leq k \leq 5$ is randomly chosen and then, k numbers are randomly selected from $\{1, \ldots, n\}$. For each graph directed G, we generated 10 Boolean networks with unate functions and dependency graph G, by using the following fact.

Lemma 3.1.4 *A Boolean function f of n variables is unate if and only if it is of the form $f(\mathbf{x}) = g(\mathbf{x}+\mathbf{s})$, where g is a monotone Boolean function of n variables and $\mathbf{s} \in k^n$ and "+" denotes addition modulo 2.*

Proof. If f is unate, then each variable x_i appears in f always as x_i or always as $\neg x_i$. Suppose that all x_i appear without negations. Then, f is constructed using $\wedge$ and $\vee$. Hence f is monotone. Otherwise, let $\mathbf{s} \in k^n$ be the vector whose i-th entry is 1 if and only if x_i appears as $\neg x_i$ in f. Then, $g(\mathbf{x}) = f(\mathbf{x}+\mathbf{s})$ is a monotone function and $f(\mathbf{x}) = g(\mathbf{x}+\mathbf{s})$. The converse is clear. $\square$

So in order to generate unate functions it is sufficient to generate monotone functions. We generated the set M_i of monotone functions in i variables by exhaustive search for $i = 1, \ldots, 5$. (For example, M_5 has 6894 elements.) Unate functions for a given signed

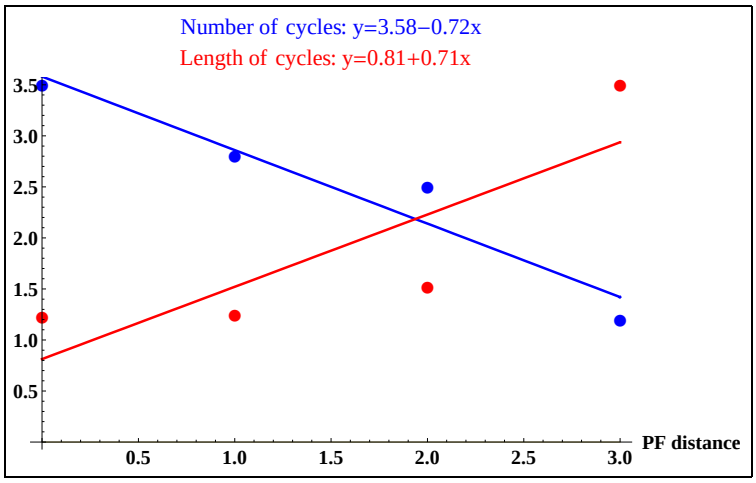


Figure 3.4: The best fit-lines of the averages of the numbers (lengths) of limit cycles from Table 3.1.

dependency graph can then be generated by choosing random functions from M_i and random vectors $\mathbf{s} \in k^n$. The nonzero entries in $\mathbf{s}$ for a given node correspond to the incoming edges with negative sign in the dependency graph. Using this process we generated Boolean networks with unate Boolean functions.

We then carry out the following steps.

The Experiment

Let G be a random unsigned directed graph on n nodes with a maximal PF-distance t, and let $D \leq t$. Consider 10 unate Boolean networks chosen at random with G as their dependency graph.

1. For $1 \leq d \leq D$, let G_d be a signed graph of G of distance d.
   (a) For each network f of the ten networks,
      i. Let g be a modified network of f such that $\mathcal{D}(g) = G_d$; the signed dependency graph of g is G_d.
      ii. Compute the number and length of all limit cycles in the phase space of g.
   (b) Compute the average number N (respectively, average length L) of limit cycles in the phase spaces of the $g's$.
2. Compute the slope s_N (respectively, s_L) of the best fit-line of the $N's$ (respectively, $L's$).

Computation of PF-distance

Let f be a Boolean network with unate Boolean functions and let $|f|$ be its PF-distance. The proofs of the following facts are straightforward.

1. Suppose the dependency graph of f has a negative feedback loop at a vertex. Let f' be the Boolean network obtained by changing a single sign to make the loop positive. Then, $|f| = |f'| + 1$.
2. Let $H_1, \dots H_s$ be the strongly connected components of the dependency graph $D(f)$. Then, $|\mathcal{D}(f)| = \sum_{i=1}^{s} |H_i|$.

The algorithm for computing $|\mathcal{D}(f)|$ now follows.

Algorithm: Distance to PF

Input: A signed, directed graph G.
Output: $|G|$; the PF-distance of G.

Let $d = 0$.

1. Let $G_1, \dots, G_r$ be the collection of all signed graphs obtained by making exactly d sign changes in G.
2. For $i = 1, \dots, r$
   If G_i is PF, then RETURN $|G| = d$.
3. Otherwise, $d := d + 1$, Go to Step 1.

In Step 2 above, to check whether a strongly connected graph is PF, it is equivalent to check whether it has any (undirected) negative cycles, which can easily be done in many different ways, see, e.g., [102]. This algorithm must terminate, since G has finitely many edges and hence the PF-distance of G is finite.

If G has m directed edges, then there are 2^m possible sign assignments. However, to compute the maximal PF-distance, one does not need to find the PF-distance of such possible assignments, see *Supporting Information* of [100] for the algorithm we used to compute the maximal PF-distance.

3.1.4 Results

In Table 3.2 we present the percentage of experiments that do conform to our hypothesis for the average of number of limit cycles as well as for the average length of limit cycles.

It should be mentioned here that a computationally expensive part of an experiment is the computation of the maximal PF-distance of a given directed graph and becomes prohibitive for even modest-size graphs, with, e.g., 10 nodes. So unlike in Example 3.1.3, for networks on more than 5 nodes, we only considered PF-distances that are less than or equal to the number of nodes in the network. (See the Methods Section for a detailed description of the experiment.) In fact for graphs with 20 (respectively, 100) nodes, all considered networks have PF-distance less than or equal to 5 (respectively, 10). We argue below that this is the reason for the drop in the percentage of experiment that conform to our hypothesis as the number of nodes increases.

Table 3.2: The percentage of experiments that conform the hypotheses

n	Num. of Exp.	Av. Num.	Av. Len.
5	117000	99.75	99.83
7	5000	97.82	99.92
10	6000	95.70	99.58
15	2921	95.72	98.25
20	331	90.03	94.86
100	659	77.39	93.93

For networks on 5 nodes, we analyzed the dynamics of 4000 experiments by varying the PF-distance considered in the computations. Table 3.3 shows the number of experiments that do not conform to our hypothesis as we vary the considered PF-distance.

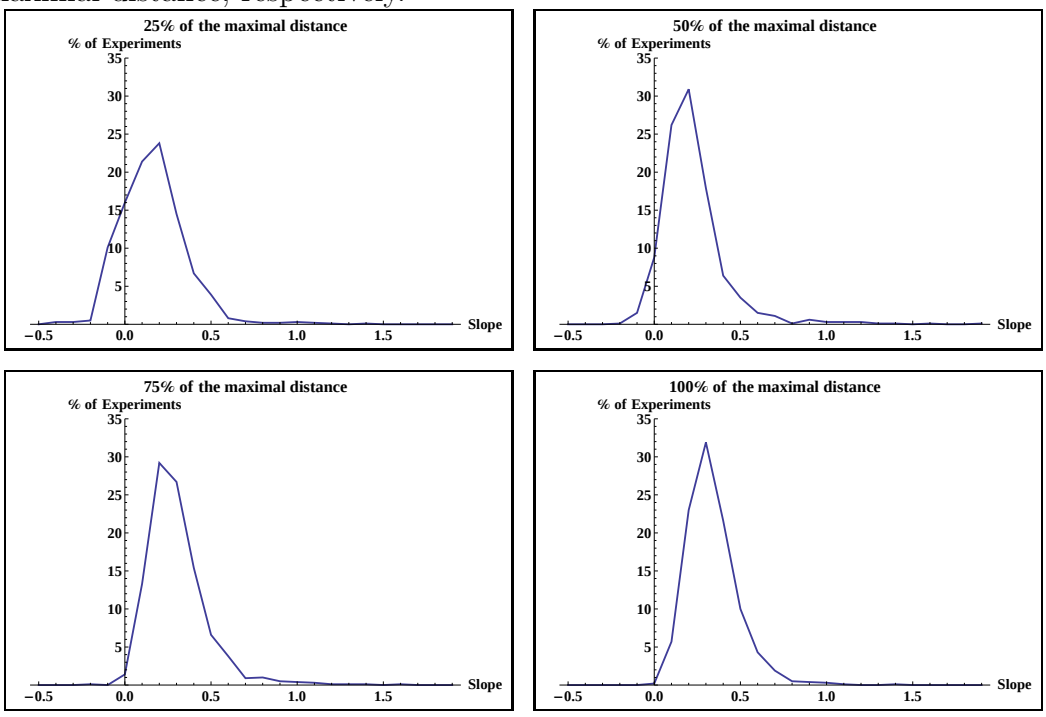
Table 3.3: The number of experiments that did not conform to our hypothesis for 5-node networks. We considered PF-distance 25%, 50%,75%, and 100% of the maximal distance . For each d, we considered 1000 experiments.

D	Av. Num.	Av. Len.	Med. Num.	Med. Len.
25%	26	114	29	542
50%	4	16	18	59
75%	0	1	6	3
100%	1	0	2	0

We also present the results in the form of histograms, where the horizontal axis represents the slope of the lines of best fit and the vertical axis represents the percentage of experiments that confirm our hypothesis. Figure 3.5 shows the results of the 4000 experiments on 5-node networks. The histograms, from left to right, show the results when PF-distance of the network is 25%, 50%, 75% and 100% of the maximal PF-distance. It can be seen in the right-most figure that almost all experiments show positive slope of the best-fit line, thereby conforming to the conjecture. Similar results for the average number of limit cycles are shown in Figure 3.6, demonstrating that if the PF-distance of networks is allowed the whole

possible range, almost all the experiments conform to our hypothesis as we already noticed in Table 3.3. However, these histograms show the distribution of slopes over different distances.

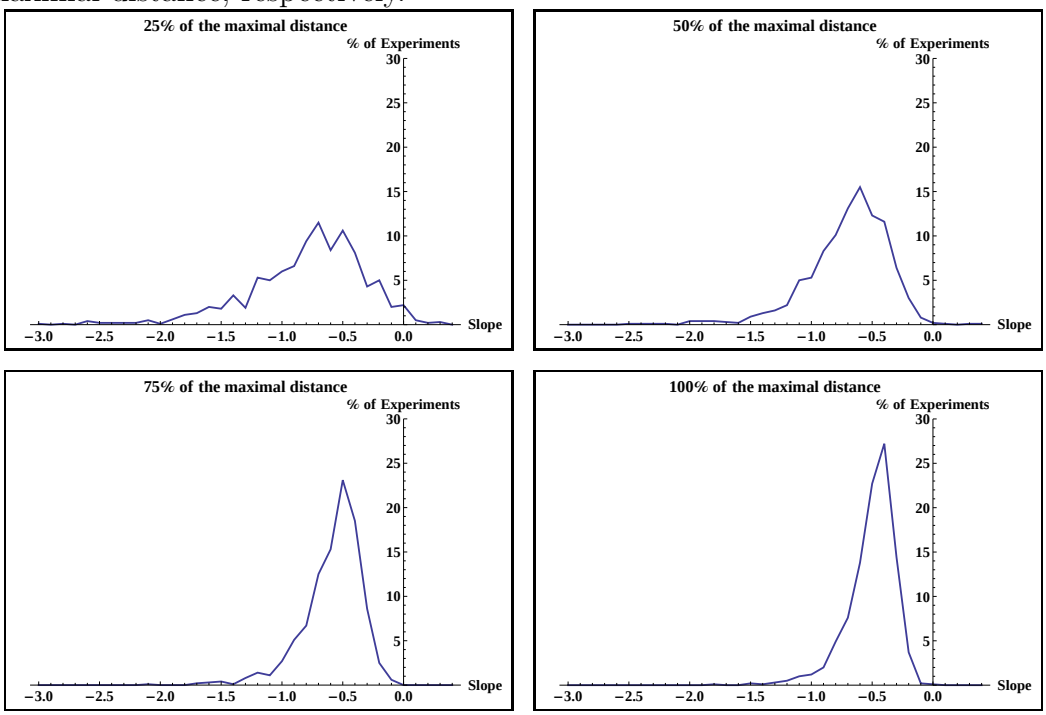
Figure 3.5: 5-node networks. Histogram of slopes of best-fit lines to the average length of limit cycles (horizontal axis) vs. percentage of experiments with a given slope (vertical axis). The panels include networks with increasing PF-distance, with 25%, 50%, 75%, and 100% of the maximal distance, respectively.



We have carried out similar computations for networks with 7 (5000 experiments)and 10 (6000 experiments) nodes; see the Supporting information of [115] for details. The results there are not quite as clear as for 5 node networks. For instance, for networks with 10 nodes (and up to 4 incoming edges per node) 31 out of 1000 experiments did not conform to our hypothesis for networks with PF-distance up to 5. However, as we explained before, this is likely due to the fact that we "truncated" the experiments before reaching the maximal PF-distance due to the computational complexity.

In summary, the extensive computations support our hypothesis that, as the PF-distance increases, the total number of limit cycles decreases on average and their average length increases. Furthermore, the slopes of the best-fit lines increasingly conform to our hypothesis the closer the PF-distance of the networks comes to the maximum PF-distance of the network topology.

Figure 3.6: 5-node networks. Histogram of slopes of best-fit lines to the average number of limit cycles (horizontal axis) vs. percentage of experiments with a given slope (vertical axis). The panels include networks with increasing PF-distance, with 25%, 50%, 75%, and 100% of the maximal distance, respectively.



3.1.5 Discussion

Negative feedback loops in biological networks play a crucial role in controlling network dynamics. The new measure of "distance to positive feedback (PF-distance)" introduced in this section is designed to capture the notion of "independent" feedback loops. We have shown that PF-distance correlates very well with the average number and length of limit cycles in networks, key measures of network dynamics. By analyzing the dynamics of more than six millions Boolean networks, we have provided evidence that networks with a larger number of independent negative feedback loops tend to have longer limit cycles and thus may exhibit more "random" or "chaotic" behavior. Furthermore, the number of limit cycles tends to decrease as the number of independent negative feedback loops increases.

In general, the problem of computing the PF-distance of a network is NP-complete, as MAX-CUT can be mapped into it as a special case; see [21; 102] for a discussion for the analogous problem of distance to monotone. The question of computing *distance to monotone* has been the subject of a few recent papers [21; 51]. The first two of these proposed a randomized algorithm based on a semi-definite programming relaxation, while the last one suggested an efficient deterministic algorithm for graphs with small distance to monotone. Since a

strongly connected component of a graph is monotone if and only if it has the PF property, methods for computing PF distance for large graphs may be developed by similar techniques. Work along these lines is in progress.

3.2 Dynamics of Conjunctive Networks

There is increasing evidence that the architecture of gene regulatory networks has a strong effect on their dynamics. In [3] it is shown with the help of a Boolean network model that the dynamics of the segment polarity network in *Drosophila melanogaster* is determined by the topology of the wiring diagram and the transcriptional logic of the genes involved rather than the specific kinetics of the system. Similarly, the key feature of bistability is established for the *lac* operon in *E coli* through a Boolean network model in [107]. It is also shown there through a network reduction process that the driver of dynamics is the interplay between the positive and negative feedback loops in this system. Thomas et al. [111] conjectured that negative feedback loops are necessary for periodic dynamics whereas positive feedback loops are necessary for multistationarity. These conjectures have been proven in several cases [31; 86; 103]. In [100] evidence was provided that biochemical networks are "close to monotone" through computational experiments using Boolean networks. The distance to monotone was measured by the number of "independent" negative feedback loops. Thus, Boolean network models lends themselves very well to modeling of biological systems.

For these and other reasons, Boolean networks and more general multi-state discrete models, such as logical models [110], have been used quite extensively to model a variety of biochemical networks, see, e.g., [22; 27; 58; 61; 66]. They are intuitive and are amenable to extensive computational analysis. There have been several studies, e.g., [87], investigating the classes of Boolean functions that are particularly well-suited to expressing the logic of gene regulation. One such class is that of so-called nested canalyzing functions (NCF), introduced in [60]. This class of functions has good dynamic properties, which make them suitable as models for molecular networks [59; 60]. It has also been shown [45; 60; 82] that in published Boolean network models of molecular networks, the majority of functions that appear are nested canalyzing. Here we consider the subclass consisting of those nested canalyzing functions that are constructed only with the AND operator, the *conjunctive* functions. Another class for which the results of this section hold is that of disjunctive Boolean networks, constructed using the OR operator. From the point of view of gene regulation, AND functions correspond to synergistic regulation of a gene by several transcription factors. There is increasing evidence that this type of mechanism is common in regulatory networks, e.g., [41; 77; 81].

When focusing on the role of feedback loops a first step is to consider networks whose wiring diagram is a directed graph which is strongly connected, that is, in which there is a directed path between any two vertices. A general wiring diagram can be decomposed into a collection of strongly connected components that form a partially ordered set, with a strongly connected

component A less than another one, B, if there are directed edges from A to B (but not the other way). Any feedback loop has to be contained within a strongly connected component of the wiring diagram. In the language of biochemical reaction networks, this condition is referred to as "weakly reversible." To a strongly connected graph one can associate its *loop number* [17], defined as follows. Choose a vertex and consider all directed loops at this vertex. The loop number is the greatest common divisor of the lengths (number of edges) of all such loops.

One of the main results in this section (Theorem 3.2.24) is that the loop number completely controls the long-term behavior of a conjunctive (respectively, disjunctive) Boolean network with a strongly connected wiring diagram. Precisely, the only lengths of limit cycles that can appear are divisors of the loop number. If m divides the loop number and $m = \prod_{i=1}^{r} p_i^{k_i}$ is the prime factorization of m, then the number $|A(m)|$ of periodic states of the network of period m is given by the formula

$$|A(m)| = \sum_{i_1=0}^{1} \cdots \sum_{i_r=0}^{1} (-1)^{i_1+i_2+\cdots+i_r} 2^{p_1^{k_1-i_1} p_2^{k_2-i_2} \ldots p_r^{k_r-i_r}}. \tag{3.2.1}$$

Dividing by m, we obtain the exact number of attractors of length m. Thus, in the case of conjunctive Boolean networks with a strongly connected wiring diagram, we can obtain complete information about their long-term dynamic behavior from the feedback loops in the wiring diagram.

The general case is considerably more complicated and depends on subtle topological features of the connections between the strongly connected components. The effect of a limit cycle in one strongly connected component can be "killed" by the input from another strongly connected component. We have established an upper bound on the number of limit cycles of a given length, which in general is not sharp. However, we are able to give a sharp lower bound as a polynomial function of the sizes of the different antichains in the partially ordered set of strongly connected components. These bounds agree for fixed points in general conjunctive networks, so that we obtain a precise formula for the number of fixed points in the general case. The way in which the bounds are derived shows that if we are interested in fixed points, then we may assume that each strongly connected component consists of a unique directed cycle, and strongly connected components are connected by a single edge. These results show that in order to understand the dynamics of the entire network it is very important to consider the effects of one strongly connected component on another one. An interesting aspect of these bounds is that they are realized by a class of networks that resemble Boolean networks with so-called frozen regions, introduced by S. Kauffman [62].

This section is based on a published paper written in collaboration with Abdul Salam Jarrah and Reinhard Laubenbacher [56]. My contributions to the paper where the statements and proofs of the main results; furthermore, recent work generalizes the results to networks defined over any finite field.

3.2.1 Background

Let $\mathbb{F}_2 := \{0, 1\}$ be the Galois field with two elements. We view a Boolean network on n variables as a dynamical system

$$f = (f_1, \dots, f_n) : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n.$$

Here, each of the coordinate functions $f_i : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2$ is a Boolean function which can be written uniquely as a polynomial where the exponent of each variable in each term is at most one [70]. In particular $AND(a, b) = ab$ and $OR(a, b) = a + b + ab$. We use polynomial forms of Boolean functions throughout this section.

Two directed graphs are usually assigned to each such system. The *dependency graph* (or wiring diagram) $\mathfrak{D}(f)$ of f has n vertices corresponding to the Boolean variables $x_1, \dots, x_n$ of f. There is a directed edge $i \to j$ if x_i appears in the function f_j. That is, $\mathfrak{D}(f)$ encodes the variable dependencies in f.

The dynamics of f is encoded by its *phase space*, denoted by $\mathcal{S}(f)$. It is the directed graph with vertex set $\mathbb{F}_2^n$ and a directed edge from $\mathbf{u}$ to $\mathbf{v}$ if $f(\mathbf{u}) = \mathbf{v}$. For each $\mathbf{u} \in \mathbb{F}_2^n$, the sequence $\{\mathbf{u}, f(\mathbf{u}), f^2(\mathbf{u}), \dots\}$ is called the *orbit* of $\mathbf{u}$. If $\mathbf{u} = f^t(\mathbf{u})$ and t is the smallest such number, the sequence $\{\mathbf{u}, f(\mathbf{u}), f^2(\mathbf{u}), \dots, f^{t-1}(\mathbf{u})\}$ is called a *limit cycle* of length t, and $\mathbf{u}$ is called a *periodic* point of *period* t. The *height* of f, denoted by $\mathrm{h}(f)$, to be the least positive integer s such that $f^s(\mathbf{u})$ is a periodic point for all $\mathbf{u} \in \mathbb{F}_2^n$. The point $\mathbf{u}$ is called a *fixed point* if $f(\mathbf{u}) = \mathbf{u}$. Since $\mathbb{F}_2^n$ is finite, every orbit must include a limit cycle. A *component* of the phase space $\mathcal{S}(f)$ consists of a limit cycle and all orbits of f that contain it. Hence, the phase space is a disjoint union of components. We define the *period* of f, denoted by $\mathrm{p}(f)$, to be the least common multiple of the lengths of all limit cycles in the phase space of f. If every limit cycle is of length 1, the system f is called a *fixed-point* system. We denote the *cycle structure* of f in the form of the generating function

$$\mathcal{C}(f) = \sum_{i=1}^{\infty} C(f)_i \mathcal{C}^i, \tag{3.2.2}$$

where $C(f)_i$ denotes the number of cycles of length i in the phase space of f. Since the phase space of f is finite, $C(f)_i = 0$ for almost all i.

Example 3.2.1 *The dependency graph of* $f = (x_2x_3, x_1x_3, x_2) : \mathbb{F}_2^3 \longrightarrow \mathbb{F}_2^3$ *and its phase space are in Figure 3.7. The graphs were generated using DVD [55].*

In Example 3.2.1, the dependency graph is strongly connected and the greatest common divisor of the length of all of its loops, the loop number, is 1. from the phase space it is clear that f has two cycles of length 1 (fixed points) and no limit cycles of any other length, hence the cycle structure of f is $\mathcal{C}(f) = 2\mathcal{C}^1$ and its height is $\mathrm{h}(f) = 4$.

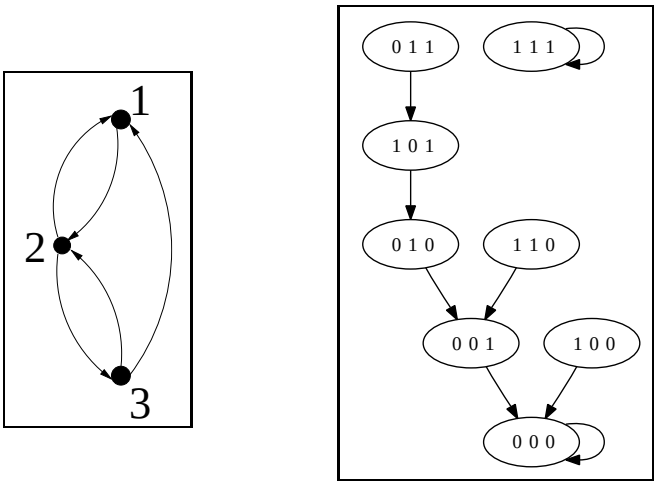


Figure 3.7: The dependency graph(left) and the phase space(right) of f from Example 3.2.1.

Example 3.2.2 *Consider the system* $f = (x_2x_3, x_1, x_2, x_3x_4, x_1x_6, x_3x_4x_5) : \mathbb{F}_2^6 \longrightarrow \mathbb{F}_2^6$. *Its dependency graph is in Figure 3.8. After generating the phase space of* f *using DVD [55], we can easily see that is has 4 fixed points and one limit cycle of length 2, that is,* $\mathcal{C}(f) = 4\mathcal{C}^1 + 1\mathcal{C}^2$, *and the height of* f *is 6.*

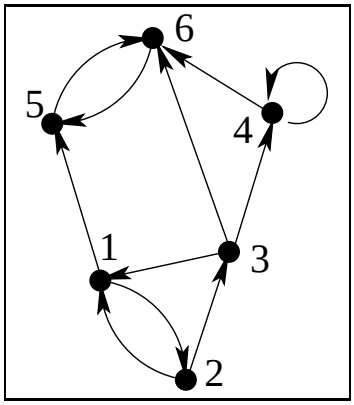


Figure 3.8: The dependency graph of f from Example 3.2.2.

The main results of this section constrain the structure of the phase space of a network by its dependency graph. In particular, using formula (3.2.1), Theorem 3.2.24 determines the exact cycle structure for networks with strongly connected dependency graph.

Next we list some of the main results in the literature about the relationship between the structure of Boolean networks and their phase space structure.

- When all coordinate Boolean functions are the XOR function (that is, the functions are linear polynomials), the structure of the phase space is determined completely by

invariants of a matrix representation. In fact, this is the case for linear systems over arbitrary finite fields [19; 27; 46]. We have developed and implemented an algorithm based on [46] to compute the precise cycle structure; see [54].

- For Boolean networks where all coordinate functions are symmetric threshold functions, it has been shown that all cycles in the phase space are either fixed points or are of length 2 [35].
- For Boolean cellular automata with the majority rule, the number of fixed points was determined in [2].
- In [7], the authors studied AND-OR networks (Boolean networks where each local function is either an OR or AND function) with directed dependency graphs. Formulae for the maximum number of fixed points are obtained.
- The main result in [6] is an upper bound for the number of fixed points in Boolean regulatory networks.

In this section we focus on the class of conjunctive, respectively disjunctive, Boolean networks, that is, Boolean networks whose coordinate functions use only the AND operator, respectively the OR operator. The following represent the main previous attempts to mathematically analyze this class of Boolean networks.

- A family of this class of networks has been analyzed in [34]. The authors studied conjunctive Boolean networks (which they called AND-nets) and disjunctive Boolean networks (OR-nets) on an undirected dependency graph, that is, on graphs in which each edge is bidirectional and hence the dependency graph consists of cycles of length two. In particular, the result that OR-nets have only fixed points and possibly a limit cycle of length two [34, Lemma 1] follows directly from our results as we explain in Remark 3.2.12.
- In [10], the authors study a smaller family where each edge is undirected and each node in the network has a self loop. In particular, they showed that disjunctive Boolean networks (which they called OR-PDS) have only fixed points as limit cycles [10, Theorem 3.3]; this follows from our results as we show in Remark 3.2.12.

The following result shows that we can focus only on conjunctive Boolean networks, since for any disjunctive Boolean network there exists a conjunctive Boolean network that has exactly the same dynamics after switching 0 and 1.

Theorem 3.2.3 *Let $f = (f_1, \dots, f_n) : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$, with $f_i = x_{i_1} \vee \cdots \vee x_{i_{j_i}}$, be a disjunctive Boolean network. Consider the conjunctive Boolean network $g = (g_1, \dots, g_n) : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$, where $g_i = x_{i_1} \wedge \cdots \wedge x_{i_{j_i}}$. Then, the two phase spaces $\mathcal{S}(f)$ and $\mathcal{S}(g)$ are isomorphic as directed graphs.*

Proof. Let $\neg : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$ be defined by $\neg(x_1, \ldots, x_n) = (1 + x_1, \ldots, 1 + x_n)$. Then, it is easy to see that $f(x_1, \ldots, x_n) = (\neg \circ g \circ \neg)(x_1, \ldots, x_n)$. Thus $\mathcal{S}(f)$ and $\mathcal{S}(g)$ are isomorphic directed graphs. $\square$

Remark 3.2.4 *Let G be a graph on n vertices such that the in-degree for each vertex is non-zero (G has no* source vertex*). Then, there is one and only one conjunctive network f on n nodes such that $\mathfrak{D}(f) = G$. Thus, there is a one-to-one correspondence between the set of all conjunctive Boolean networks on n nodes where none of the local functions are constant and the set of directed graphs on n vertices where none of the vertices are a source.*

This correspondence was used in [17] to decide when a given Boolean monomial system (conjunctive Boolean network) is a fixed-point system, as we will recall in the next section. Next we recall some results from graph theory as well as results about powers of positive matrices that we will use to obtain upper bounds for the lengths of transients.

Let $f : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$ be a conjunctive Boolean network, $G = \mathfrak{D}(f)$ and A the adjacency matrix of G. We will assume here and in the remainder of the section that none of the coordinate functions of f are constant, that is, all vertices of G have positive in-degree.

The Adjacency Matrix

Define the following relation on the vertices of G: $a \sim b$ if and only if there is a directed path from a to b and a directed path from b to a. It is easy to check that $\sim$ is an equivalence relation. Suppose there are t equivalence classes $V_1, \ldots, V_t$. For each equivalence class V_i, the subgraph G_i with the vertex set V_i is called a *strongly connected component* of G. The graph G is called *strongly connected* if G consists of a unique strongly connected component. A trivial strongly connected component is a graph on one vertex and no self loop. Since such components do not influence the cycle structure, throughout this article, we assume all strongly connected components are non-trivial.

There exists a permutation matrix P that permutes the rows and columns of A such that

$$PAP^{-1} = \begin{bmatrix} A_1 & A_{12} & \cdots & A_{1t} \\ 0 & A_2 & \cdots & A_{2t} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A_t \end{bmatrix} \tag{3.2.3}$$

where A_i is the adjacency matrix of the component G_i, and A_{ij} represents the edges from the component G_i to G_j, see [13, Theorem 3.2.4]. The form in (3.2.3) is called the *Frobenius Normal Form* of A.

Remark 3.2.5 *The effect of the matrix P on the dependency graph is the relabeling of the vertices of G such that the diagonal blocks correspond to strongly connected components of G. In Example 3.2.2 above, the adjacency matrix of the dependency graph is in normal form.*

Example 3.2.2 (Cont.). The dependency graph G in Figure 3.8 has 3 strongly connected components and their vertex sets are: $V_1 = \{1, 2, 3\}, V_2 = \{4\}$, and $V_3 = \{5, 6\}$, see Figure 3.9 (left).

Let G_i be a strongly connected and let h_i be the conjunctive Boolean network with dependency graph $\mathfrak{D}(h_i) = G_i$. Let $h : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$ be the conjunctive Boolean network defined by $h = (h_1, \ldots, h_t)$. That is, the dependency graph of h is the disjoint union of the strongly connected graphs $G_1, \ldots, G_t$. That is, h is obtained from g by deleting all edges between strongly connected components.

Example 3.2.2 (Cont.). The conjunctive Boolean network h corresponding to the disjoint union is $h : \mathbb{F}_2^6 \longrightarrow \mathbb{F}_2^6$, given by

$$h(x_1, \ldots, x_6) = (h_1(x_1, x_2, x_3), h_2(x_4), h_3(x_5, x_6)),$$

where $h_1(x_1, x_2, x_3) = (x_2x_3, x_1, x_2)$, $h_2(x_4) = x_4$, and $h_3(x_5, x_6) = (x_6, x_5)$.

Now define the following order relation on the strongly connected components $G_1, \ldots, G_t$ of the dependency graph $\mathfrak{D}(f)$ of the network f.

$$G_i \preceq G_j \text{ if there is at least one edge from a vertex in } G_i \text{ to a vertex in } G_j. \tag{3.2.4}$$

In this way we obtain a partially ordered set $\mathcal{P}$. In this section, we relate the dynamics of f to the dynamics of its strongly connected components and their poset $\mathcal{P}$.

Example 3.2.2 (Cont.). The poset of the strongly connected components of f is in Figure 3.9 (right).

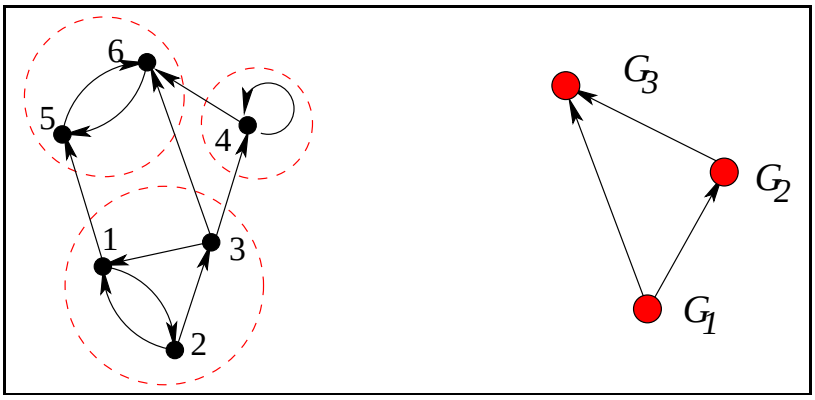


Figure 3.9: The strongly connected components of f (left) and their poset (right).

For any non-negative matrix A, the sequence $\{A, A^2, \ldots\}$ has been studied extensively, see, for example, [13; 91]. Next we use the Boolean operators AND and OR to examine the sequence of powers of A and infer results about the conjunctive Boolean network that corresponds to the adjacency matrix A.

Powers of Boolean Matrices

Let A, B be $n \times n$ Boolean matrices (all entries are either 0 or 1). Define $A \otimes B$ such that $(A \otimes B)_{ij} = \bigvee_{k=1}^{n} (A_{ik} \wedge B_{kj})$, where $\vee$ (respectively, $\wedge$) is the Boolean OR (respectively, AND) operator.

Proposition 3.2.6 *Let A, B be as above and let $f, g : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$ be the two conjunctive Boolean networks that correspond to the adjacency matrices A and B, respectively. That is, $f_i = x_1^{a_{i1}} x_2^{a_{i2}} \cdots x_n^{a_{in}}$ and $g_i = x_1^{b_{i1}} x_2^{b_{i2}} \cdots x_n^{b_{in}}$ for all i. Then, the adjacency matrix corresponding to $f \circ g$ is $A \otimes B$.*

Proof. It is easy to see that, for all i,

$$\begin{aligned} f_i(g_1, \ldots, g_n) &= g_1^{a_{i1}} \cdots g_n^{a_{in}} \\ &= (x_1^{b_{11}} x_2^{b_{12}} \cdots x_n^{b_{1n}})^{a_{i1}} \cdots (x_1^{b_{n1}} x_2^{b_{n2}} \cdots x_n^{b_{nn}})^{a_{in}} \\ &= x_1^{a_{i1}b_{11}+\cdots+a_{in}b_{n1}} \cdots x_n^{a_{i1}b_{1n}+\cdots+a_{in}b_{nn}} \\ &= x_1^{\sum_{j=1}^n a_{ij}b_{j1}} \cdots x_n^{\sum_{j=1}^n a_{ij}b_{jn}}. \end{aligned}$$

Since we are working over $\mathbb{F}_2$, for all $1 \leq k \leq n$, we have $x_k^q = x_k$ for all positive integers q. Thus

$$\begin{aligned} x_k \text{ divides } f_i(g_1, \ldots, g_n) &\iff \sum_{j=1}^{n} a_{ij} b_{jk} \geq 1 \\ &\iff a_{ij_0} b_{j_0 k} = 1 \text{ for some } j_0 \\ &\iff a_{ij_0} \wedge b_{j_0 k} = 1 \text{ for some } j_0 \\ &\iff \bigvee_{j=1}^{n} a_{ij} \wedge b_{jk} = 1 \\ &\iff (A \otimes B)_{ik} = 1. \end{aligned}$$

Thus the matrix $(A \otimes B)$ is the adjacency matrix of $f \circ g$. $\square$

Throughout this section, we use A^s to denote $\underbrace{A \otimes \cdots \otimes A}_{s \text{ times}}$.

Corollary 3.2.7 *Let f be a conjunctive Boolean network and let A be the adjacency matrix of its dependency graph $\mathfrak{D}(f)$. Then, A^s is the adjacency matrix of f^s.*

The Loop Number

An invariant of a strongly connected graph, called the loop number, was defined in [17]. We generalize this definition to any directed graph.

Definition 3.2.8 *The* loop number *of a strongly connected graph is the greatest common divisor of the lengths of its simple (no repeated vertices) directed cycles. We define the loop number of a trivial strongly connected graph to be 0. The loop number of any directed graph G is the least common multiple of the loop numbers of its non-trivial strongly connected components.*

Remark 3.2.9 *Let G be a directed graph and A its adjacency matrix.*

1. *The loop number of G is the same as the* index of cyclicity *of G as in [97] and the* index of imprimitivity *of A as in [12; 13].*
2. *The loop number can be computed in polynomial time; see [17] for an algorithm.*
3. *If the loop number of G is 1, the adjacency matrix A of G is called* primitive*, see [13].*

Example 3.2.2 (Cont.). The loop numbers of V_1, V_2, V_3 are 1,1,2, respectively. See Figure 3.9 (left). In particular, the loop number of G is 2.

Definition 3.2.10 *The exponent of an irreducible matrix A with loop number c is the least positive integer k such that $A^{k+c} = A^k$.*

The following lemma follows from Proposition 3.2.6 and the definition above.

Lemma 3.2.11 *Let f be a conjunctive Boolean network and let A be the adjacency matrix of its dependency graph $\mathfrak{D}(f)$. Suppose the loop number of A is c and its exponent is k. Then, $\mathrm{h}(f) = k$ and $\mathrm{p}(f)$ divides c. In particular, $C(f)_i = 0$ for every $i \nmid c$ and hence Equation (3.2.2) becomes*

$$\mathcal{C}(f) = \sum_{i|c} C(f)_i \mathcal{C}_i, \tag{3.2.5}$$

Example 3.2.2 (Cont.). The phase space $\mathcal{S}(f)$ has 2^6 vertices, its cycle structure is $\mathcal{C}(f) = 4\mathcal{C}_1 + 1\mathcal{C}_2$. In particular, the period of f is 2 which is the same as the loop number of its dependency graph G.

Remark 3.2.12 *It is clear that if x_i appears in f_i for all i, then the loop number of $\mathfrak{D}(f)$ is 1 and hence $\mathcal{S}(f)$ has only fixed points as limit cycles, this was shown in [10, Theorem 3.3]. Also if each edge in the dependency graph is undirected, then the dependency graph is made up of simple cycles of length 2 and hence the loop number of $\mathfrak{D}(f)$ is either two or one. Thus $\mathcal{S}(f)$ has only fixed points and possibly cycles of length two, which was shown in [34, Lemma 1].*

For the case when $\mathfrak{D}(f)$ is strongly connected $\mathrm{p}(f) = c$ as was shown in [17, Theorem 4.13]; in particular, $\mathcal{S}(f)$ has a simple cycle of length l if and only if l divides c. In general, however, this is not the case.

Example 3.2.13 *Consider the Boolean network* $f = (x_2, x_1, x_2x_5, x_3, x_4) : \mathbb{F}_2^5 \longrightarrow \mathbb{F}_2^5$. *The graph* $\mathfrak{D}(f)$ *has two strongly connected components with loop numbers 2 and 3 respectively, and hence the loop number of* $\mathfrak{D}(f)$ *is 6. However, it is easy to see, using DVD [55], that* $\mathcal{C}(f) = 3\mathcal{C}_1 + 1\mathcal{C}_2 + 2\mathcal{C}_3$, *and hence* f *has no cycle of length 6.*

The exponent has been studied extensively and upper bounds are known, [97, Theorem 3.11] presents an upper bound for the exponent of any irreducible matrix. Using the lemma above, we rewrite this upper bound for the height of any Boolean monomial system with a strongly connected dependency graph.

Theorem 3.2.14 *Let* f *be a conjunctive Boolean network with strongly connected dependency graph* $\mathfrak{D}(f)$, *and suppose the loop number of* $\mathfrak{D}(f)$ *is* c. *Then*

$$\mathrm{h}(f) \leq \begin{cases} (n-1)^2 + 1, & \text{if } c = 1; \\ \max\{n-1, \dfrac{n^2-1}{2} + \dfrac{n^2}{c} - 3n + 2c\}, & \text{if } c > 1. \end{cases}$$

Proof. The proof follows from Proposition 3.2.6 above and [97, Theorem 3.11]. □

Furthermore, [97, Theorem 3.20] also implies an upper bound for the height of any conjunctive Boolean network.

Theorem 3.2.15 *Let* f *be any conjunctive Boolean network on* n *nodes. Then,* $\mathrm{h}(f) \leq 2n^2 - 3n + 2$.

Proof. The proof follows from Proposition 3.2.6 above and [97, Theorem 3.20]. □

We close this section with a classical theorem about positive powers of Boolean matrices. This theorem has the remarkable corollary that almost all conjunctive Boolean networks have a strongly connected dependency graph and, furthermore, have only fixed points as limit cycles.

Theorem 3.2.16 *[13, Theorem 3.5.11] Let* $N(n)$ *be the number of conjunctive Boolean networks on* n *nodes with strongly connected dependency graph and loop number 1. Then*

$$\lim_{n\to\infty} \frac{N(n)}{2^{n^2}} = 1.$$

In particular, since there are 2^{n^2} *different conjunctive Boolean networks on* n *nodes, as* $n \to \infty$, *almost all conjunctive Boolean networks on* n *nodes have a strongly connected dependency graph and have only fixed points as limit cycles.*

Although there is a wealth of information about powers of non-negative matrices such as the transient length or possible cycle length, very little seems to be known about the number of cycles of a given length in the phase space and that is the main goal of this section. In the next section we give a complete answer to this problem for conjunctive Boolean networks with strongly connected dependency graph. For this class of systems, we find the number of cycles of any possible length in the phase space.

3.2.2 Networks with Strongly Connected Dependency Graph

In this section we give an exact formula for the cycle structure of conjunctive Boolean networks with strongly connected dependency graphs. Since "almost all" conjunctive Boolean networks have this property by Theorem 3.2.16, one may consider this result as giving a complete answer for conjunctive Boolean networks in the limit. However, in the next section we will also consider networks with general dependency graphs and give upper and lower bounds for the cycle structure.

Before deriving the desired state space results we prove some needed facts about general finite dynamical systems.

Lemma 3.2.17 *Let $f : X \longrightarrow X$ be a finite dynamical system, with X a finite set, and let $\mathbf{u} \in X$ be a periodic point of period t.*

1. *If $f^s(\mathbf{u}) = \mathbf{u}$, then t divides s.*
2. *The period of $f^j(\mathbf{u})$ is t, for any $j \geq 1$.*
3. *If $f^s(\mathbf{u}) = f^j(\mathbf{u})$, then t divides $s - j$.*

Proof. Since t is the period of $\mathbf{u}$ and $f^s(\mathbf{u}) = \mathbf{u}$, by definition $s \geq t$. Thus $s = qt + r$, where $0 \leq r < t$. Now $\mathbf{u} = f^s(\mathbf{u}) = f^r(f^{qt}(\mathbf{u})) = f^r(\mathbf{u})$. Since $r < t$ and t is the period of $\mathbf{u}$, $r = 0$ and hence t divides s. That proves (1). The proof of (2) is straightforward, since $f^t(f^j(\mathbf{u})) = f^j(f^t(\mathbf{u})) = f^j(\mathbf{u})$. Now we prove (3). Suppose $s \geq j$. Since $f^s(\mathbf{u}) = f^j(\mathbf{u})$, we have $f^{s-j}(f^j(\mathbf{u})) = f^s(\mathbf{u}) = f^j(\mathbf{u})$. Thus, by (1), the period of $f^j(\mathbf{u})$ divides $s - j$. But the period of $f^j(\mathbf{u})$ is t, by (2). □

Lemma 3.2.18 *Let $f : X \longrightarrow X$ be a finite dynamical system. Then, $\mathrm{p}(f) = t$ and $\mathrm{h}(f) = k$ if and only if t and k are the least positive integers such that $f^{m+t}(\mathbf{u}) = f^m(\mathbf{u})$ for all $m \geq k$ and $\mathbf{u} \in X$.*

Proof. For all $\mathbf{u} \in X$ and $m \geq k$, $f^m(\mathbf{u})$ is a periodic point and hence its period divides t. Thus $f^{m+t}(\mathbf{u}) = f^t(f^m(\mathbf{u})) = f^m(\mathbf{u})$.

Now suppose t and k are the least positive integers such that $f^{m+t}(\mathbf{u}) = f^m(\mathbf{u})$ for all $m \geq k$ and $\mathbf{u} \in X$. We want to show that $\mathrm{p}(f) = t$ and $\mathrm{h}(f) = k$. It is clear that $f^d(\mathbf{u})$ is periodic for all $\mathbf{u} \in X$ and k is the least such positive number. Thus the height of f is k. Also, the period f is t, since t is the least positive integer such that $f^t(\mathbf{u}) = \mathbf{u}$ for any periodic point $\mathbf{u} \in X$. $\square$

Theorem 3.2.19 *Let $f : X \longrightarrow X$ and $g : Y \longrightarrow Y$ be two finite dynamical systems. Define the system $h : X \times Y \longrightarrow X \times Y$ by $h(\mathbf{u}, \mathbf{u}') = (f(\mathbf{u}), g(\mathbf{u}'))$. Then, $\mathcal{S}(h) = \mathcal{S}(f) \times \mathcal{S}(g)$.*

Proof. This follows from the fact that $h(\mathbf{u}, \mathbf{u}') = (\mathbf{v}, \mathbf{v}')$ if and only if $f(\mathbf{u}) = \mathbf{v}$ and $g(\mathbf{u}') = \mathbf{v}'$ for all $\mathbf{u} \in X$ and $\mathbf{u}' \in Y$. $\square$

Corollary 3.2.20 *Let f, g and h be as in Theorem 3.2.19. Then, the period of h is $\mathrm{p}(h) = \mathrm{lcm}\{\mathrm{p}(f), \mathrm{p}(g)\}$ and its height is $\mathrm{h}(h) = \max\{\mathrm{h}(f), \mathrm{h}(g)\}$.*

Proof. It is easy to see that a set $A \subset X \times Y$ is a cycle in $\mathcal{S}(h)$ if and only if A_X (respectively, A_Y) is a cycle in $\mathcal{S}(f)$ (respectively, $\mathcal{S}(g)$), where $A_X := \{\mathbf{u} \in X : (\mathbf{u}, \mathbf{v}) \in A \text{ for some } \mathbf{v} \in Y\}$. Furthermore, it is clear that the length of the cycle A is the least common multiple of the lengths of A_X and A_Y. Thus $\mathrm{p}(h) = \mathrm{lcm}\{\mathrm{p}(f), \mathrm{p}(g)\}$. The proof of $\mathrm{h}(h) = \max\{\mathrm{h}(f), \mathrm{h}(g)\}$ follows from the definition of height. $\square$

Recall Equation (3.2.2), in particular, that $C(f)_m$ is the number of cycles of length m in the phase space of f, the following corollary follows directly from Theorem 3.2.19.

Corollary 3.2.21 *Let f, g and h be as above. Then, the cycle structure of h is $\mathcal{C}(h) = \mathcal{C}(f)\mathcal{C}(g) := \sum_{m|\mathrm{p}(h)} C(h)_m \mathcal{C}^m$, where*

$$C(h)_m = \sum_{\substack{s|p(f) \\ t|p(g) \\ \mathrm{lcm}\{s,t\}=m}} \gcd\{s,t\} C(f)_s C(g)_t. \tag{3.2.6}$$

That is, the generating function for the cycle structure of h is the product of the generating functions of f and g, where $\mathcal{C}^s \cdot \mathcal{C}^t = \gcd\{s,t\}\mathcal{C}^{\mathrm{lcm}\{s,t\}}$.

Let $f : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$ be a conjunctive Boolean network. Assume that the dependency graph $\mathfrak{D}(f)$ of f is strongly connected with loop number c. For any divisor k of c, it is well-known that the set of vertices of $\mathfrak{D}(f)$ can be partitioned into k non-empty sets $W_1, \ldots, W_k$ such that each edge of $\mathfrak{D}(f)$ is an edge from a vertex in W_i to a vertex in W_{i+1} for some i with $1 \leq i \leq k$ and $W_{k+1} = W_1$. For a proof of this fact see [13, Lemma 3.4.1(iii)] or [17, Lemma 4.7].

By definition the length of any cycle in the phase space $\mathcal{S}(f)$ of f divides c, the period of f. For any positive integers p, k that divide c, let $A(p)$ be the set of periodic points of period p and let $D(k) := \bigcup_{p|k} A(p)$.

Lemma 3.2.22 *The cardinality of the set $D(k)$ is $|D(k)| = 2^k$.*

Proof. Let $\Phi : \mathbb{F}_2^k \longrightarrow D(k)$ be defined by

$$\Phi(x_1, \ldots, x_k) = (\underbrace{x_1, \ldots, x_1}_{s_1 \text{ times}}, \underbrace{x_2, \ldots, x_2}_{s_2 \text{ times}}, \ldots, \underbrace{x_k, \ldots, x_k}_{s_k \text{ times}}),$$

where, without loss of generality, $W_i = \{v_{i,1}, \ldots, v_{i,s_i}\}$ for all $1 \leq i \leq k$. We will show that Φ is a bijection. First we show that Φ is well-defined. Let $z = \Phi(x_1, \ldots, x_k)$, by [17, Theorem 4.10], there exists a positive integer m such that

$$f^{mk+j}(z) = (\underbrace{x_{j+1}, \ldots, x_{j+1}}_{s_1 \text{ times}}, \underbrace{x_{j+2}, \ldots, x_{j+2}}_{s_2 \text{ times}}, \ldots, \underbrace{x_j, \ldots, x_j}_{s_k \text{ times}}).$$

In particular, $f^k(z) = f^{mk+k}(z) = z$. Thus $z = \Phi(x_1, \ldots, x_k) \in D(k)$. Since it is clear that Φ is one-to one, it is left to show that Φ is onto.

Let $z \in D(k)$ which means that $f^k(z) = z$. It is enough to show that $z_{i,h} = z_{i,g}$ for all $1 \leq h, g \leq s_{i_k}$ and $1 \leq i \leq k$. Suppose not. Without loss of generality, let $z_{i,h} = 1$ and $z_{i,g} = 0$. Since k divides c, $f^c(z) = z$. But there is a path from $v_{i,h}$ and $v_{i,g}$ of length divides c, contradiction. Hence Φ is a bijection and $|D(k)| = 2^k$. $\square$

Corollary 3.2.23 *If p is a prime number and p^k divides c for some $k \geq 1$, then*

$$|A(p^k)| = 2^{p^k} - 2^{p^{k-1}}.$$

Proof. It is clear that $|D(1)| = 2$. Namely, $(0, \ldots, 0)$ and $(1, \ldots, 1)$ are the only two fixed points. Now if p is prime and $k \geq 1$, then the proof follows from the fact that $D(p^k) = D(p^{k-1}) \biguplus A(p^k)$, where $\biguplus$ is the disjoint union. $\square$

Next we prove Theorem 3.2.24 which gives the exact number of periodic points of any possible length.

Theorem 3.2.24 *Let f be a conjunctive Boolean network whose dependency graph is strongly connected and has loop number c. If $c = 1$, then f has the two fixed points $(0, 0, \ldots, 0)$ and $(1, 1, \ldots, 1)$ and no other limit cycles. If $c > 1$ and m is a divisor of c, then the number of periodic states of period m is $|A(m)|$ as in Equation (3.2.1).*

Proof. The statement for $c = 1$ is part of the previous corollary. Now suppose that $c > 1$. For $1 \leq j \leq r$, let $m_j = p_j^{k_j - 1} \prod_{i=1, i \neq j}^{r} p_i^{k_i}$. Then, $D(m) = A(m) \biguplus (\bigcup_{j=1}^{r} D(m_j))$, where $\biguplus$ is a disjoint union, in particular,

$$A(m) = D(m) \setminus \bigcup_{j=1}^{r} D(m_j).$$

The formula 3.2.1 follows from the inclusion-exclusion principle and the disjoint union above. □

Corollary 3.2.25 *If m divides c, then the number of cycles of length m in the phase space of f is $C(f)_m = \dfrac{|A(m)|}{m}$. Hence the cycle structure of f is*

$$\mathcal{C}(f) = \sum_{m|c} \frac{|A(m)|}{m} \mathcal{C}^m.$$

Remark 3.2.26 *Notice that the cycle structure of f depends on its loop number only. In particular, a conjunctive Boolean network with loop number 1 on a strongly connected dependency graph only has as limit cycles the two fixed points $(0, 0, \ldots, 0)$ and $(1, 1, \ldots, 1)$, regardless of how many vertices its dependency graph has.*

3.2.3 Networks with general dependency graph

Let $f : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$ be a conjunctive Boolean network with dependency graph $\mathfrak{D}(f)$. Without loss of generality, the adjacency matrix of $\mathfrak{D}(f)$ is in the form (3.2.3). Let $G_1, \ldots, G_t$ be the strongly connected components of $\mathfrak{D}(f)$ corresponding to the matrices $A_1, \ldots, A_t$, respectively. Furthermore, suppose that none of the G_i is trivial (i.e., A_i is the zero matrix). For $1 \leq i \leq t$, let h_i be the conjunctive Boolean network that has G_i as its dependency graph and suppose that the loop number of h_i is c_i. In particular, the loop number of f is $c := \operatorname{lcm}\{c_1, \ldots, c_t\}$.

First we study the effect of deleting an edge in the dependency graph between two strongly connected components. Let G_1 and G_2 be two strongly connected components in $\mathfrak{D}(f)$ and suppose $G_1 \preceq G_2$. Let $x \longrightarrow y$ be a directed edge in $\mathfrak{D}(f)$ between a vertex x in G_1 and a vertex y in G_2. Let $\mathfrak{D}'$ be the graph $\mathfrak{D}(f)$ after deleting this edge, and let g be the conjunctive Boolean network such that $\mathfrak{D}(g) = \mathfrak{D}'$.

Theorem 3.2.27 *Any cycle in the phase space of f is a cycle in the phase space of g. In particular $\mathcal{C}(f) \leq \mathcal{C}(g)$ componentwise.*

Proof. Let $\mathcal{C} := \{\mathbf{u}, f(\mathbf{u}), \dots, f^{m-1}(\mathbf{u})\}$ be a cycle of length m in $\mathcal{S}(f)$. To show that $\mathcal{C}$ is a cycle in $\mathcal{S}(g)$, it is enough to show that, whenever the x-value in $\mathbf{u}$ is 0, then the y'-value in $\mathbf{u}$ is 0 for all $y' \in G_2$ such that there is an edge from y' to y. Thus, the value of y is determined already by the value of y' and the edge (x, y) does not contribute anything new and hence $\mathcal{C}$ is a cycle in $\mathcal{S}(g)$.

Suppose the loop number of G_1 (respectively, G_2) is a (respectively, b). Now, any path from x (respectively, y) to itself is of length pa (respectively, qb) where $q, p \geq T$ and T is large enough, see [17, Corollary 4.4]. Thus there is a path from x to y of length $qa + 1$ for any $q \geq T$. Also, there is a directed path from y to y' of length $qb - 1$ for any $q \geq T$. This implies the existence of a path from x to y' of length $q(a + b)$ for all $q \geq T$.

Now $\mathbf{u} = f^m(\mathbf{u}) = f^{mk}(\mathbf{u})$, for all $k \geq 1$. Choose q, k large enough such that $q(a+b) = km \geq T$. Then, the value of y' in $f^{q(a+b)}(\mathbf{u})$ is 0, since there is a path from x to y' of length $q(a+b)$ and the value of x is 0. Therefore, the value of y' in $\mathbf{u}$ is zero, since $\mathbf{u} = f^{mk}(\mathbf{u}) = f^{q(a+b)}(\mathbf{u})$. □

Corollary 3.2.28 *Let f and $\mathfrak{D}(f)$ be as above. For any two strongly connected components in $\mathfrak{D}(f)$ that are connected, drop all but one of the edges. Let $\mathfrak{D}'$ be the new graph and let g be the conjunctive Boolean network such that $\mathfrak{D}(g) = \mathfrak{D}'$. Then, any cycle in $\mathcal{S}(f)$ is a cycle in $\mathcal{S}(g)$. In particular $\mathcal{C}(f) \leq \mathcal{C}(g)$ componentwise.*

Notice that there are many different possible g one can get and each one of them has the same poset as f and provides an upper bound for the cycle structure of f. However, we do not have a polynomial formula for the cycle structure of g. When we delete all edges between any two strongly connected components we get an easy formula for the cycle structure for the corresponding system as we describe below. In Section 3.2.6 we will present an improved upper bound for the cycle structure of f.

Let $h : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$ be the conjunctive Boolean network with the disjoint union of $G_1, \dots, G_t$ as its dependency graph. That is, $h = (h_1, \dots, h_t)$. For h, there are no edges between any two strongly connected components of the dependency graph of the network. Its cycle structure can be completely determined from the cycles structures of the h_i alone.

Theorem 3.2.29 *Let $\mathcal{C}(h_i) = \sum_{j|l_i} a_{i,j}\mathcal{C}^j$ be the cycle structure of h_i. Then, the cycle structure of h is $\mathcal{C}(h) = \prod_{i=1}^t \mathcal{C}(h_i)$ and the number of cycles of length m (where $m|l$) in the phase space of h is*

$$C(h)_m = \sum_{\substack{j_i|l_i \\ \operatorname{lcm}\{j_1,\dots,j_t\}=m}} \frac{j_1 \cdots j_t}{m} \prod_{i=1}^t a_{i,j_i}. \tag{3.2.7}$$

Proof. This follows directly from Corollary 3.2.21. □

Corollary 3.2.30 *Let f and h be as above. The number of cycles of any length in the phase space of f is less than or equal to the number of cycles of that length in the phase space of h. That is $\mathcal{C}(f) \leq \mathcal{C}(h)$ componentwise.*

Example 3.2.13 (Cont.). Here $h = (h_1, h_2)$, where $h_1(x_1, x_2) = (x_2, x_1)$ and $h_2(x_3, x_4, x_5) = (x_5, x_3, x_4)$. It is easy to check that $\mathcal{C}(h_1) = 2\mathcal{C}^1 + 1\mathcal{C}^2$, and $\mathcal{C}(h_2) = 2\mathcal{C}^1 + 2\mathcal{C}^3$. By Theorem 3.2.29, $\mathcal{C}(h) = 4\mathcal{C}^1 + 2\mathcal{C}^2 + 4\mathcal{C}^3 + 2\mathcal{C}^6$. In particular, $\mathcal{C}(f) \leq \mathcal{C}(h)$.

3.2.4 Bounds on the cycle structure

Let f, h, $G_1, \ldots, G_t$ be as above and let c_i be the loop number of G_i. Furthermore, let $\mathcal{P}$ be the poset of the strongly connected components. Let Ω be the set of all maximal antichains in $\mathcal{P}$. For $J \subseteq [t] = \{1, \ldots, t\}$, let $x_J := \prod_{j \in J} x_j$ and let $\overline{J} := [t] \setminus J$.

Definition 3.2.31 *Let $\mathcal{A}$ be a subset of the set of limit cycles of h and let s_i be the number of limit cycles of length i in $\mathcal{A}$. Define $\|\mathcal{A}\|$ by*

$$\|\mathcal{A}\| := \sum_i s_i \mathcal{C}^i.$$

Remark 3.2.32 *Using the inclusion-exclusion principle, if $\mathcal{A}, \mathcal{A}_1, ..., \mathcal{A}_s$ are subsets of the set of limit cycles of h and $\mathcal{A} = \bigcup_i^s \mathcal{A}_i$, then,*

$$\|\mathcal{A}\| = \sum_{J \subseteq [s]} (-1)^{|J|+1} \| \bigcap_{j \in J} \mathcal{A}_j \|.$$

Definition 3.2.33 *For any subset $J \subseteq [t]$, let*

$$J^{\preceq} := \{k : G_j \preceq G_k \text{ for some } j \in J\},\ J^{\succeq} := \{k : G_j \succeq G_k \text{ for some } j \in J\},$$

$$J^{\prec} := \{k : G_j \prec G_k \text{ for some } j \in J\}, \text{ and } J^{\succ} := \{k : G_j \succ G_k \text{ for some } j \in J\}.$$

A limit cycle $\mathcal{C}$ in the phase space of f is J_0 (respectively, J_1) if the G_j component of $\mathcal{C}$ is 0 (respectively, 1) for all $j \in J$.

Notice that the sets defined above are closely related to upper and lower order ideals in posets, see [104, p. 100].

Example 3.2.2 (Cont.). Let $J = \{2\}$. Then, $J^{\preceq} = \{2, 3\}$, $J^{\succeq} = \{1, 2\}$, $J^{\prec} = \{3\}$, and $J^{\succ} = \{1\}$.

Definition 3.2.34 *A limit cycle $\mathcal{C}$ in $\mathcal{S}(h)$ is* $J-$regular *if $\mathcal{C}$ is both $(J^{\prec})_0$ and $(J^{\succ})_1$, for some maximal antichain J in $\mathcal{P}$. A limit cycle is called* regular *if it is $J-$regular for some maximal antichain J. That is, an element of a $J-$regular limit cycle has 0 in all entries corresponding to strongly connected components lying above components in J and 1 in all entries corresponding to components lying below components in J.*

Lemma 3.2.35 *Let $\mathcal{J} \subseteq \Omega$ be a set of maximal antichains. For $J \in \mathcal{J}$, let A_J be the set of all $J-$regular limit cycles in the phase space of f. Then,*

$$\| \bigcap_{J\in\mathcal{J}} A_J\| = \prod_{j\in\bigcap_{J\in\mathcal{J}} J} \mathcal{C}(h_j).$$

Proof. It is easy to see that a regular limit cycle $\mathcal{C} \in \bigcap_{J\in\mathcal{J}} A_J$ if and only if the only non trivial components of $\mathcal{C}$ are in $j \in \bigcap_{J\in\mathcal{J}} J$. □

3.2.5 A Lower Bound

It is easy to see that a regular cycle is actually a limit cycle in the phase space of f, and hence the numbers of regular cycles of different length provide lower bounds for the number of limit cycles of different lengths in the phase space of f. Next we express these numbers in terms of the cycle structures $\mathcal{C}(h_i)$ of the strongly connected components.

Lemma 3.2.36 *The cycle structure of the regular limit cycles in $\mathcal{S}(h)$ is*

$$\sum_{\mathcal{J}\subseteq\Omega} (-1)^{|\mathcal{J}|+1} \prod_{j\in\bigcap_{J\in\mathcal{J}} J} \mathcal{C}(h_j). \tag{3.2.8}$$

Proof. By definition, the set of regular limit cycles is $\bigcup_{J\in\Omega} A_J$. Now, using the inclusion-exclusion principle and Lemma 3.2.35, it follows that

$$\begin{aligned}\bigcup_{J\in\Omega} A_J &= \sum_{\mathcal{J}\subseteq\Omega} (-1)^{|\mathcal{J}|+1} \| \bigcap_{J\in\mathcal{J}} A_J\| \\ &= \sum_{\mathcal{J}\subseteq\Omega} (-1)^{|\mathcal{J}|+1} \prod_{j\in\bigcap_{J\in\mathcal{J}} J} \mathcal{C}(h_j).\end{aligned}$$

□

Next we derive a sharp lower bound of the number of cycles of all possible lengths.

Theorem 3.2.37 *Consider the function*

$$\mathcal{L}(z_1, \ldots, z_t) := \sum_{\mathcal{J}\subseteq\Omega} (-1)^{|\mathcal{J}|+1} \prod_{j\in\bigcap_{J\in\mathcal{J}} J} z_i.$$

Then, for any conjunctive Boolean network f with subnetworks $h_1, \dots, h_t$ and Ω its set of maximal antichains in the poset of f, we have

$$\mathcal{L}(\mathcal{C}(h_1), \dots, \mathcal{C}(h_t)) \leq \mathcal{C}(f). \tag{3.2.9}$$

Here, the function $\mathcal{L}$ is evaluated using the "multiplication" described in Corollary 3.2.21. This inequality provides a sharp lower bound on the number of limit cycles of f of a given length.

Proof. The inequality follows from the previous lemma. Now to show that this lower bound is sharp, it is enough to present a Boolean monomial dynamical system f such that $\mathcal{C}(f) = \mathcal{L}(\mathcal{C}(h_1), \dots, \mathcal{C}(h_t))$.

Let f be such that whenever there is a directed edge from G_i to G_j in the poset $\mathcal{P}$, there is a directed edge from every vertex in G_i to every vertex in G_j in the dependency graph of f. It is enough to check that any cycle in the phase space of f is regular. Let $\mathcal{C}$ be a cycle in the phase space of f. Let $\mathcal{C}(j)$ be the projection of $\mathcal{C}$ on the strongly connected component G_j. It is clear that $\mathcal{C}(j)$ is cycle in the $\mathcal{C}(h_j)$. Suppose that $\mathcal{C}(j)$ is non trivial, that is, it is not one of the fixed points $(0, \dots, 0)$ or $(1, \dots, 1)$. Now if $G_i \preceq G_j$, then all entries corresponding to G_i must be one. On the other hand, if $G_i \succeq G_j$, then all entries corresponding to G_i must be zero. Therefore, $\mathcal{C}$ is regular. □

Note that the left side of the inequality (3.2.9) is a polynomial function in the variables $\mathcal{C}(h_i)$, with coefficients that are functions of the cycle numbers of various lengths of the h_i and the anti-chains of the poset of strongly connected components. That is, the sharp lower bound is a polynomial function depending exclusively on measures of the network topology.

3.2.6 An Upper Bound

Next we present a polynomial whose coefficients provide upper bounds for the number of possible limit cycles in the phase space of f. Similar to the lower bound polynomial, this upper bound polynomial is in terms of the strongly connected components of f and their poset. However, this upper bound is not sharp. In fact, we will show that it is not possible to give a sharp upper bound that is in a polynomial form with constant coefficients. First we prove the following set-theoretic equality which we need in this section.

Lemma 3.2.38 *Let $\{A_i : i \in \Delta\}$ be a finite collection of finite sets of limit cycles in $\mathcal{S}(h)$, $A = \bigcap_i A_i$, and $B_i \subseteq A_i$. Then, for $K \subseteq \Delta$,*

$$\| \bigcap_{k \in K} (A_k \setminus B_k) \| = \sum_{L \subseteq K} (-1)^{|L|} \| A \cap \bigcap_{k \in L} B_k \|.$$

Proof. To begin, observe that

$$\begin{aligned}\bigcap_{k\in K}(A_k \setminus B_k) &= \bigcap_{k\in K}(A \setminus B_k)\\ &= A \setminus \bigcup_{k\in K}(A\cap B_k).\end{aligned}$$

Then, using the inclusion-exclusion principle we obtain:

$$\begin{aligned}\|\bigcap_{k\in K}(A_k \setminus B_k)\| &= \|A \setminus \bigcup_{k\in K}(A\cap B_k)\|\\ &= \sum_{L\subseteq K}(-1)^{|L|}\|\bigcap_{k\in L}(A\cap B_k)\|\\ &= \sum_{L\subseteq K}(-1)^{|L|}\|A\cap\bigcap_{k\in L}B_k\|.\end{aligned}$$

□

Definition 3.2.39 *A limit cycle $\mathcal{C}$ in $\mathcal{S}(h)$ is called* admissible *if, for any $j\in[t]$,*

1. *If the coordinates of states in $\mathcal{C}$ corresponding to component G_j are 0, then $\mathcal{C}$ is $(j^{\preceq})_0$; and*
2. *If the coordinates of states in $\mathcal{C}$ corresponding to component G_j are 1, then $\mathcal{C}$ is $(j^{\succeq})_1$.*

Notice that any limit cycle in the phase space of f is admissible. In particular, we have the following lemma.

Lemma 3.2.40 *The number of admissible limit cycles in $\mathcal{S}(h)$ is an upper bound for the number of limit cycles in $\mathcal{S}(f)$.*

For $J\subseteq[t]$, let Z_J be the set of all J_0 limit cycles and let O_J be the set of all J_1 limit cycles. Let $K,L\subseteq[t]$, then it is easy to see that$Z_K\cap Z_L = Z_{K\cup L}$, $O_K\cap O_L = O_{K\cup L}$, and the cycle structure of $Z_K\cap O_L$ is

$$\|Z_K\cap O_L\| = \langle K,L\rangle \prod_{j\in\overline{K\cup L}}\mathcal{C}(h_j),$$

where

$$\langle K,L\rangle = \begin{cases}0, & \text{if } K\cap L\neq\emptyset,\\ 1, & \text{if } K\cap L=\emptyset.\end{cases}$$

Theorem 3.2.41 *The cycle structure of the admissible limit cycles in $\mathcal{S}(h)$ is equal to*

$$\sum_{\substack{I\subseteq N\subseteq[t]\\ J\subseteq M\subseteq[t]}}(-1)^{|N|+|M|+|I|+|J|}\langle I_N,J^M\rangle\prod_{k\in\overline{I_N\cup J^M}}\mathcal{C}(h_k),$$

where $I_N=I^{\succeq}\cup N$, $J^M=J^{\preceq}\cup M$.

Proof. By definition, it follows that the set of admissible limit cycles is

$$Adm:=\{\text{limit cycles in }\mathcal{S}(h)\}\setminus\bigcup_{i,j\in[t]}(Z_{\{i\}}\setminus Z_{\{i\}^{\succeq}})\cup(O_{\{j\}}\setminus O_{\{j\}^{\preceq}}).$$

Using the inclusion-exclusion principle, it follows that

$$\|Adm\|=\prod_{k\in[t]}\mathcal{C}(h_k)+\sum_{\substack{N\subseteq[t]\\ M\subseteq[t]\\ N\cup M\neq\emptyset}}(-1)^{|N|+|M|}\|\bigcap_{i\in N}(Z_{\{i\}}\setminus Z_{\{i\}^{\succeq}})\cap\bigcap_{j\in M}(O_{\{j\}}\setminus O_{\{j\}^{\preceq}})\|. \quad (3.2.10)$$

On the other hand, since $\bigcap_{i\in N}Z_{\{i\}}=Z_N$, $\bigcap_{j\in M}O_{\{j\}}=O_M$, $Z_i\subseteq Z_{\{i\}^{\succeq}}$ and $O_j\subseteq O_{\{j\}^{\preceq}}$ it follows from Lemma 3.2.38 that

$$\begin{aligned}
\|\bigcap_{i\in N}(Z_{\{i\}}\setminus Z_{\{i\}^{\succeq}})\ \cap\ &\bigcap_{j\in M}O_{\{j\}}\setminus O_{\{j\}^{\preceq}})\|\\
&=\sum_{\substack{I\subseteq N\\ J\subseteq M}}(-1)^{|I|+|J|}\|(Z_N\cap O_M)\cap(\bigcap_{i\in N}Z_{\{i\}^{\succeq}}\cap\bigcap_{j\in M}O_{\{j\}^{\preceq}})\|\\
&=\sum_{\substack{I\subseteq N\\ J\subseteq M}}(-1)^{|I|+|J|}\|(Z_N\cap O_M)\cap(Z_{\bigcup_{i\in N}\{i\}^{\succeq}}\cap O_{\bigcup_{j\in M}\{j\}^{\preceq}})\|\\
&=\sum_{\substack{I\subseteq N\\ J\subseteq M}}(-1)^{|I|+|J|}\|Z_N\cap O_M\cap Z_{I^{\succeq}}\cap O_{J^{\preceq}}\|\\
&=\sum_{\substack{I\subseteq N\\ J\subseteq M}}(-1)^{|I|+|J|}\|Z_{N\cup I^{\succeq}}\cap O_{M\cup J^{\preceq}}\|\\
&=\sum_{\substack{I\subseteq N\\ J\subseteq M}}(-1)^{|I|+|J|}\|Z_{I_N}\cap O_{J^M}\|\\
&=\sum_{\substack{I\subseteq N\\ J\subseteq M}}(-1)^{|I|+|J|}\langle I_N,J^M\rangle\prod_{k\in\overline{I_N\cup J^M}}\mathcal{C}(h_k).
\end{aligned}$$

Equation (3.2.10) then becomes

$$\begin{aligned}\|Adm\| &= \prod_{k\in[t]} \mathcal{C}(h_k) + \sum_{\substack{N\subseteq[t]\\ M\subseteq[t]\\ N\cup M\neq\emptyset}} (-1)^{|N|+|M|} \sum_{\substack{I\subseteq N\\ J\subseteq M}} (-1)^{|I|+|J|}\langle I_N, J^M\rangle \prod_{k\in\overline{I_N\cup J^M}} \mathcal{C}(h_k)\\ &= \prod_{k\in[t]} \mathcal{C}(h_k) + \sum_{\substack{I\subseteq N\subseteq[t]\\ J\subseteq M\subseteq[t]\\ N\cup M\neq\emptyset}} (-1)^{|N|+|M|+|I|+|J|}\langle I_N, J^M\rangle \prod_{k\in\overline{I_N\cup J^M}} \mathcal{C}(h_k)\\ &= \sum_{\substack{I\subseteq N\subseteq[t]\\ J\subseteq M\subseteq[t]}} (-1)^{|N|+|M|+|I|+|J|}\langle I_N, J^M\rangle \prod_{k\in\overline{I_N\cup J^M}} \mathcal{C}(h_k).\end{aligned}$$

□

By Lemma 3.2.40 we now have an upper bound for the number of cycles of any given length. This upper bound is clearly not sharp, since it is easy to find admissible limit cycles in the phase space of h that are not cycles in the phase space of f. The advantage of this upper bound, however, is that it is a polynomial in terms of the strongly connected components and their poset. Furthermore, we will show next that this bound gives the exact number of fixed points.

Theorem 3.2.42 *Let f be a conjunctive Boolean network. Then, any admissible fixed point is regular and hence the number of fixed points $C(f)_1$ in the phase space of f is*

$$C(f)_1 = \sum_{\mathcal{J}\subseteq\Omega} (-1)^{|\mathcal{J}|+1}\, 2^{|\bigcap_{J\in\mathcal{J}} J|}. \tag{3.2.11}$$

Proof. It follows from the definitions that any admissible fixed point is regular, and hence $\mathcal{U}(2\mathcal{C}^1,\dots,2\mathcal{C}^1) = \mathcal{L}(2\mathcal{C}^1,\dots,2\mathcal{C}^1)$. In particular, the number of fixed points $C(f)_1$ is the coefficient of $\mathcal{C}^1$ in $\mathcal{L}(2\mathcal{C}^1,\dots,2\mathcal{C}^1)$. By Equation (3.2.8), we get

$$\begin{aligned}\mathcal{L}(2\mathcal{C}^1,\dots,2\mathcal{C}^1) &= \sum_{\mathcal{J}\subseteq\Omega} (-1)^{|\mathcal{J}|+1} \prod_{j\in\bigcap_{J\in\mathcal{J}} J} 2\mathcal{C}^1\\ &= \sum_{\mathcal{J}\subseteq\Omega} (-1)^{|\mathcal{J}|+1}\, 2^{|\bigcap_{J\in\mathcal{J}} J|}\mathcal{C}^1.\end{aligned}$$

□

An example

Each system g from Corollary 3.2.28 has the same poset as f and the cycle structure of each g is an upper bound for the cycle structure of f. However, using the poset alone, or using

the cycle structure of the strongly connected components alone, one can not expect to find a polynomial form with constant coefficients that would provide a sharp upper bound, as we now show.

Let $\mathcal{P}$ be the poset on two nodes G_1 and G_2 where $G_1 \preceq G_2$. Let f be a conjunctive Boolean network with a dependency graph that has only two strongly connected components which are connected by one edge. Let F be the set of all such systems. We will show that there is no polynomial $W(x_1, x_2) = \sum_{i,j} a_{ij} x_1^i x_2^j$ such that, for all $f \in F$, $W(\mathcal{C}(h_1), \mathcal{C}(h_2)) = \mathcal{C}(f)$, where h_1 and h_2 are the conjunctive Boolean networks corresponding to the two strongly connected components of f.

Suppose the loop number of h_1 is 1 and that of h_2 is q where q is prime. Then, $\mathcal{C}(h_1) = 2\mathcal{C}^1$ and $\mathcal{C}(h_2) = 2\mathcal{C}^1 + \frac{2^q-2}{q}\mathcal{C}^q$. It is easy to check that $\mathcal{C}(f) = 3\mathcal{C}^1 + \frac{2^q-2}{q}\mathcal{C}^q$. Since $\mathcal{C}(f) = W(\mathcal{C}(h_1), \mathcal{C}(h_2))$, we have

$$\begin{aligned}
3\mathcal{C}^1 + \frac{2^q-2}{q}\mathcal{C}^q &= W(2\mathcal{C}^1, 2\mathcal{C}^1 + \frac{2^q-2}{q}\mathcal{C}^q) \\
&= \sum_{i,j} a_{ij}(2\mathcal{C}^1)^i(2\mathcal{C}^1 + \frac{2^q-2}{q}\mathcal{C}^q)^j \\
&= \sum_{i,j} 2^i a_{ij}(2\mathcal{C}^1 + \frac{2^q-2}{q}\mathcal{C}^q)^j \\
&= \sum_{i,j} 2^i a_{ij}[2^j\mathcal{C}^1 + [(2 + \frac{2^q-2}{q})^j - 2^j]\mathcal{C}^q] \\
&= \sum_{i,j} 2^{i+j} a_{ij}\mathcal{C}^1 + \sum_{i,j} 2^i a_{ij}[(2 + \frac{2^q-2}{q})^j - 2^j]\mathcal{C}^q.
\end{aligned}$$

Equating coefficients, for any q, we have $\frac{2^q-2}{q} = \sum_{i,j} 2^i a_{ij}[(2+\frac{2^q-2}{q})^j - 2^j]$. Therefore, $a_{ij} = 0$ for all $i, j \geq 1$ and hence $W(x_1, x_2)$ must be of the form $W(x_1, x_2) = a_{11}x_1x_2 + a_{10}x_1 + a_{01}x_2 - a_{00}$.

Now suppose the loop number of h_1 is p and that of h_2 is q. Then, any cycle in the phase space of f is regular and hence it is easy to check that $\mathcal{C}(f) = \mathcal{C}(h_1) + \mathcal{C}(h_2) - \mathcal{C}_1$. In particular, $a_{11} = 0$ and hence $W(x_1, x_2) = a_{10}x_1 + a_{01}x_2 - a_{00}$.

However, for the case when the loop number of h_1 is 4 and that of h_2 is 6, we have $\mathcal{C}(f) = 3\mathcal{C}^1 + 3\mathcal{C}^2 + 2\mathcal{C}^3 + 4\mathcal{C}^4 + 11\mathcal{C}^6 + 2\mathcal{C}^{12}$, $\mathcal{C}(h_1) = 2\mathcal{C}^1 + \mathcal{C}^2 + 3\mathcal{C}^4$ and $\mathcal{C}(h_2) = 2\mathcal{C}^1 + \mathcal{C}^2 + 2\mathcal{C}^3 + 9\mathcal{C}^6$. In particular, $\mathcal{C}(f) \neq W(\mathcal{C}(h_1), \mathcal{C}(h_2))$.

3.2.7 Discussion

Discrete models have been used effectively to separate the roles of kinetics and network topology in biochemical reaction networks, in particular gene regulation networks. An important feature of the network topology is the collection of interacting feedback loops. In this section we studied the question of how the feedback loops in the wiring diagram of the network affect the dynamics for the class of conjunctive Boolean network models that have been identified as useful models for gene regulatory networks. They are constructed from certain types of nested canalyzing functions for which it is possible to precisely answer this question.

For conjunctive Boolean networks it is the partially ordered set of strongly connected components of the wiring diagram and their loops numbers which control the dynamics of the network in precise ways. This insight allows far-reaching conclusions about network dynamics from its topology, an important step toward an understanding of the design principles of biochemical networks. The next step toward this goal is to generalize the work in this section to networks with positive and negative edges. For this, a numerical measure like the loop number is required which takes into account the network topology and the signs of the feedback loops.

Chapter 4

Model Reduction and Inference

4.1 Reduction of Boolean Networks

Boolean networks have been successfully used in modeling gene regulatory networks such as the *Drosophila* segment polarity network [3], the yeast cell-cycle network [68] and the Th regulatory network [76]. Boolean networks provide a nice theoretical framework that allows for simulation, control theory and reverse engineering.

However, their analysis is not a trivial task. For example, the problem of finding steady states has been shown to be NP-complete [121]. One way to overcome these kind of problems is to develop mathematical and computational tools [36; 53; 55; 88]. While these tools allow to answer some questions, such as what the steady states are, it is often not intuitive why such answers were obtained. Another way is to reduce the network while keeping the main features; a reduced network is easier to analyze and could not only help answering questions, but can also give insight of why such answers were obtained. This in turn, provides a better understanding of the problem that is being studied.

In this section a reduction method for Boolean networks is proposed. The reduction method can make the analysis easier and can also elucidate the role of network topology on the dynamics. We will focus on the existence, number and type of steady states.

4.1.1 Reduction Method

Reduction Steps

We now provide the reduction steps to reduce a Boolean network and its corresponding wiring diagram. The idea behind the reduction method is simple: the wiring diagram and Boolean functions should reflect direct regulation and hence nonfunctional edges and variables should

be removed; also, vertices can be deleted, without losing important information, by allowing its functionality to be "inherited" to other variables.

1. We simplify the Boolean functions and wiring diagram:
   (a) Reduce Boolean expressions using Boolean algebra. This will delete variables that are not functional.
   (b) Delete edges that do not correspond to Boolean expressions. That is, we delete edges that are non functional.
2. We delete vertices with no self loop, that is, vertices whose Boolean function does not depend on it. Let x_i be a vertex such that f_{x_i} does not depend on x_i.
   (a) For all vertices $x_i \to y$, that is, for all vertices whose Boolean function depends on x_i, replace the Boolean function for y, $f_y(x_1, \ldots, x_i, \ldots, x_k)$, by $f_y(x_1, \ldots, f_{x_i}, \ldots, x_k)$.
   (b) Replace edges $y \to x_i \to z$ by $y \to z$ and delete x_i (and edges from/to x_i)

We will see that these steps give rise to a reduced network that keeps features of the wiring diagram and dynamical properties of the original network.

Reduction Algorithm

We present an algorithm to simplify Boolean functions and their wiring diagram (S); and an algorithm to eliminate a vertex x (R).

Algorithm S

Input: $f = (f_1, \ldots, f_n)$ and the adjacency matrix, A.
For $i = 1, \ldots, n$:

1. Simplify f_i using Boolean algebra
2. Construct A corresponding to variables appearing in f_i

Output: (Simplified) $f = (f_1, \ldots, f_n)$ and A.

Example 4.1.1 *Consider the Boolean network given by* $f = (f_1, f_2, f_3) = ((x_2 \wedge x_3) \vee x_2, (x_1 \wedge x_3) \vee \neg x_2, \neg x_1)$ *with wiring diagram given in Figure 4.1. Algorithm S gives as an output* $f = (f_1, f_2, f_3) = (x_2, (x_1 \wedge x_3) \vee \neg x_2, \neg x_1)$ *with wiring diagram given in Figure 4.1. We can clearly see that Algorithm S detected that the although the variable* x_3 *appears in*

f_1, it is not functional; hence x_3 is removed from f_1 (using Boolean algebra) and the edge $x_3 \to x_1$ is deleted as well. That is, after using S, we obtain an accurate representation of f and its wiring diagram.

Figure 4.1: Wiring diagram of f before (left) and after (right) using Algorithm S. Arrows indicate positive paths and circles indicate negative paths.

If we say that a variable z depends on x, we mean that f_z depends on x. Also, if we write $f_i = f_i(x_{i_1}, \ldots, x_{i_k})$, we are saying that we are considering the function f_i in terms of $x_{i_1}, \ldots, x_{i_k}$, even if f_i does not depend on some of the x_{i_j}'s; however, it does mean that f_i does not depend on the other variables. This convention will make some of the definitions and proofs simpler.

Algorithm R

Input: $f = (f_1, \ldots, f_n)$, the adjacency matrix (A), and a vertex x that does not depend on itself.

1. Find the variables that depend on x: $z_1, \ldots, z_r$
2. For $i = 1, 2, \ldots, \ldots, r$
   Replace $f_{z_i} = f_{z_i}(x, \ldots)$ by $f_{z_i} = f_{z_i}(f_x, \ldots)$
3. Let $f^{[x]} = (f_1, \ldots, \check{f}_x, \ldots, f_n)$ (where $\check{f}_x$ means that f_x is omitted) and simplify it using S.
4. Let $A^{[x]}$=adjacency matrix of the wiring diagram of $f^{[x]}$

Output: $f^{[x]}$ and $A^{[x]}$.

Example 4.1.2 *Consider the Boolean network given by $f = (f_1, f_2, f_3) = (x_2, (x_1 \wedge x_3) \vee \neg x_2, \neg x_1)$, with wiring diagram given in Figure 4.2. After using algorithm R we obtain the network $f^{[x_3]} = h = (h_1, h_2) = (x_2, (x_1 \wedge \neg x_1) \vee \neg x_2) = (x_2, \neg x_2)$ with wiring diagram in Figure 4.2. We can see that the functionality of x_3, that is, "being $\neg x_1$", is inherited to x_2 and simplified using S.*

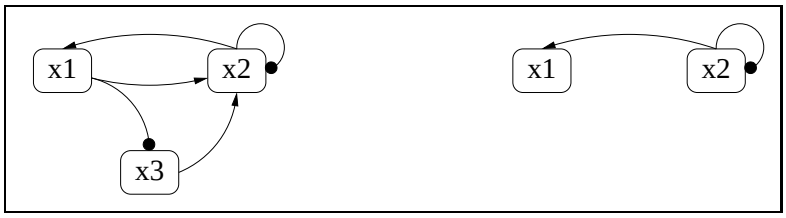


Figure 4.2: Wiring diagram of f before (left) and after (right) using algorithm R to eliminate x_3.

The Boolean network obtained by eliminating vertices $x_{i_1}, \dots, x_{i_k}$ is denoted by $f^{[x_{i_1},\dots,x_{i_k}]}$ with wiring diagram $A^{[x_{i_1},\dots,x_{i_k}]}$. The following proposition states that the order in which variables are eliminated is not important.

Proposition 4.1.3 *Using the notation of Algorithm R we have the following:* $f^{[x_i,x_j]} = f^{[x_j,x_i]}$ *and* $A^{[x_i,x_j]} = A^{[x_j,x_i]}$

Proof. Without loss of generality we can assume $x_i = x_1$ and $x_j = x_n$.

Let $g' = (g'_1, \dots, g'_{n-1}) = f^{[x_n]}$, $g = (g_2, \dots, g_{n-1}) = g'^{[x_1]} = f^{[x_n,x_1]}$, $h' = (h'_2, \dots, h'_n) = f^{[x_1]}$, $h = (h_2, \dots, h_{n-1}) = h'^{[x_n]} = f^{[x_1,x_n]}$. We consider 4 cases.

Case 1. x_n and x_1 do not depend on each other.

Then, since f_1 does not depend on x_n (nor itself), we have $f_1 = f_1(x_2, \dots, x_{n-1})$ and since f_n does not depend on x_1 (nor itself) we have $f_n = f_n(x_2, \dots, x_{n-1})$.

Then, $g'_1 = f_1(x_1, x_2, \dots, x_{n-1})$ and $g'_i = f_i(x_1, x_2, \dots, x_{n-1}, f_n(x_2, \dots, x_{n-1}))$ for $i = 2, \dots, n-1$. Then, for $i = 2, \dots, n-1$ we have

$$\begin{aligned} g_i &= g'_i(g'_1, x_2, \dots, x_{n-1}) \\ &= f_i(g'_1, x_2, \dots, x_{n-1}, f_n(x_2, \dots, x_{n-1})) \\ &= f_i(f_1(x_2, \dots, x_{n-1}), x_2, \dots, x_{n-1}, f_n(x_2, \dots, x_{n-1})) \end{aligned}$$

Similarly, it follows that $h_i = f_i(f_1(x_2, \dots, x_{n-1}), x_2, \dots, x_{n-1}, f_n(x_2, \dots, x_{n-1}))$ for $i = 2, \dots, n-1$. Hence, $g_i = h_i$ for $i = 2, \dots, n-1$ and $g = h$.

Case 2. x_n depends on x_1 but x_1 does not depend on x_n.

Then, since f_n depends on x_1 (and not on x_n), we have $f_n = f_n(x_1, \dots, x_{n-1})$ and since f_1 does not depend on x_n (nor x_1), we have $f_1 = f_1(x_2, \dots, x_{n-1})$.

Then, $g'_1 = f_1(x_2, \dots, x_{n-1})$ and $g'_i = f_i(x_1, \dots, x_{n-1}, f_n(x_1, \dots, x_{n-1}))$ for $i = 2, \dots, n-1$.

Then, for $i = 2, \dots, n-1$ we have

$$\begin{aligned} g_i &= g_i'(g_1', x_2, \dots, x_{n-1}) \\ &= f_i(g_1', x_2, \dots, x_{n-1}, f_n(g_1', x_2, \dots, x_{n-1})) \\ &= f_i(f_1(x_2, \dots, x_{n-1}), x_2, \dots, x_{n-1}, f_n(f_1(x_2, \dots, x_{n-1}), x_2, \dots, x_{n-1})) \end{aligned}$$

On the other hand, $h_i' = f_i(f_1(x_2, \dots, x_{n-1}), x_2, \dots, x_n)$ for $i = 2, \dots, n-1$ and $h_n' = f_n(f_1(x_2, \dots, x_{n-1}), x_2, \dots, x_{n-1})$. Then, for $i = 2, \dots, n-1$ we have

$$\begin{aligned} h_i &= h_i'(x_2, \dots, x_{n-1}, h_n') \\ &= f_i(f_1(x_2, \dots, x_{n-1}), x_2, \dots, x_{n-1}, h_n') \\ &= f_i(f_1(x_2, \dots, x_{n-1}), x_2, \dots, x_{n-1}, f_n(f_1(x_2, \dots, x_{n-1}), x_2, \dots, x_{n-1})) \end{aligned}$$

Hence, $g_i = h_i$ for $i = 2, \dots, n-1$ and $g = h$.

Case 3. x_1 depends on x_n but x_n does not depend on x_1. It is analogous to Case 2.

Case 4. x_1 and x_n depend on each other.

Then, since f_n depends on x_1 (and not on x_n), we have $f_n = f_n(x_1, \dots, x_{n-1})$ and since f_1 depends on x_n (and not on x_1), we have $f_1 = f_1(x_2, \dots, x_n)$.

Then, $g_1' = f_1(x_2, \dots, x_{n-1}, f_n(x_1, \dots, x_{n-1}))$ and $g_i' = f_i(x_1, \dots, x_{n-1}, f_n(x_1, \dots, x_{n-1}))$ for $i = 2, \dots, n-1$. Then, for $i = 2, \dots, n-1$ we have

$$\begin{aligned} g_i &= g_i'(g_1', \dots, x_{n-1}) \\ &= f_i(g_1', x_2, \dots, x_{n-1}, f_n(g_1', \dots, x_{n-1})) \\ &= f_i(f_1(x_2, \dots, x_{n-1}, f_n(x_1, \dots, x_{n-1})), x_2, \dots, x_{n-1}, f_n(f_1(x_2, \dots, x_{n-1}, f_n(x_1, \dots, x_{n-1})), x_2, \dots, x_{n-1} \end{aligned}$$

On the other hand, $h_n' = f_n(f_1(x_2, \dots, x_n), x_2, \dots, x_{n-1})$ and $h_i' = f_i(f_1(x_2, \dots, x_n), x_2, \dots, x_n)$ for $i = 2, \dots, n-1$. Then, for $i = 2, \dots, n-1$ we have

$$\begin{aligned} h_i &= h_i'(x_2, \dots, x_{n-1}, h_n') \\ &= f_i(f_1(x_2, \dots, h_n'), x_2, \dots, x_{n-1}, h_n') \\ &= f_i(f_1(x_2, \dots, x_{n-1}, f_n(f_1(x_2, \dots, x_n), x_2, \dots, x_{n-1})), x_2, \dots, x_{n-1}, f_n(f_1(x_2, \dots, x_n), x_2, \dots, x_{n-1})) \end{aligned}$$

Notice that $f_1(x_2, \dots, x_{n-1}, f_n(x_1, \dots, x_{n-1}))$ does not depend on x_1 and $f_n(f_1(x_2, \dots, x_n), x_2, \dots, x_{n-1})$ does not depend on x_n ($g = g'^{[x_1]}$ and $h = h'^{[x_n]}$ would be undefined otherwise). Then, for $i = 2, \dots, n-1$

$$\begin{aligned} g_i &= f_i(f_1(x_2, \dots, x_{n-1}, f_n(x_1, \dots, x_{n-1})), x_2, \dots, x_{n-1}, f_n(f_1(x_2, \dots, x_{n-1}, f_n(x_1, \dots, x_{n-1})), x_2, \dots, x_{n-1} \\ &= f_i(f_1(x_2, \dots, x_{n-1}, f_n(x_1, \dots, x_{n-1})), x_2, \dots, x_{n-1}, f_n(f_1(x_2, \dots, x_{n-1}, x_n), x_2, \dots, x_{n-1})) \\ &= f_i(f_1(x_2, \dots, x_{n-1}, f_n(f_1(x_2, \dots, x_n), \dots, x_{n-1})), x_2, \dots, x_{n-1}, f_n(f_1(x_2, \dots, x_{n-1}, x_n), x_2, \dots, x_{n-1})) \\ &= h_i \end{aligned}$$

Hence, $g = h$. $\square$

The Boolean network obtained by eliminating all variables that can be eliminated (variables that do not have a self loop) is denoted by f^R with wiring diagram A^R. From Proposition 4.1.3, it follows that f^R and A^R are independent of the order chosen to eliminate vertices (but they do depend on the choice of variables to be eliminated). Also, it may be the case that f^R and A^R are empty; in the case they are not empty, each vertex has a self loop.

Example 4.1.4 *Consider the Boolean network f defined by:*

$$\begin{array}{ll} f_1 = x_9 \vee x_{11} & f_7 = x_4 \\ f_2 = \neg x_7 \wedge x_{12} & f_8 = x_5 \\ f_3 = 0 & f_9 = x_6 \wedge \neg x_{12} \\ f_4 = x_1 \wedge \neg x_{10} & f_{10} = x_7 \vee x_{11} \\ f_5 = x_2 \wedge \neg x_{10} & f_{11} = (x_7 \vee x_{11}) \wedge \neg x_{12} \\ f_6 = x_3 \wedge \neg x_8 & f_{12} = x_8 \wedge \neg x_{11} \end{array}$$

This Boolean network corresponds to the logical model for Th-lymphocyte differentiation presented in [89]. It turns out that we can eliminate the variables $x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}$. Then $f^R = h = (h_{11}, h_{12}) = (x_{11} \wedge \neg x_{12}, \neg x_{11} \wedge x_{12})$. Notice that each vertex in the wiring diagram of the reduced network has a self loop.

4.1.2 Properties of the Reduction Method

Reduction Method and Dynamical Properties

We now show that the original and reduced network share important dynamical properties. The next theorem states that the reduction method does not create nor destroy steady states.

Theorem 4.1.5 *Let f be a Boolean network and $g = f^{[x_{i_1}, x_{i_2}, \ldots, x_{i_k}]}$ with $k < n$. Consider the projection $\pi : \{0,1\}^n \to \{0,1\}^{n-k}$ defined by*
$\pi(x_1, \ldots, x_n) = (x_1, \ldots, x_{i_1-1}, \check{x}_{i_1}, x_{i_1+1}, \ldots, x_{i_k-1}, \check{x}_{i_k}, x_{i_k+1} \ldots, x_n)$. Then, π defines a one to one correspondence between the set of steady states of f and the set of steady states of g.

Proof. We only need to prove the theorem for $k = 1$; the general case follows by induction. Without loss of generality we can assume that $g = (g_1, \ldots, g_{n-1}) = f^{[x_n]}$.

If $z = (z_1, \ldots, z_n)$ is a steady state of f, that is, $f(z) = z$, or $f_i(z) = z_i$ for $i = 1, \ldots, n$, we want to show that $\pi(z) = (z_1, \ldots, z_{n-1})$ is a steady state of g, that is, $g(\pi(z)) = \pi(z)$ or $g_i(\pi(z)) = z_i$ for $i = 1, \ldots, n-1$. On the other hand, if $\widetilde{z} = (z_1, \ldots, z_{n-1})$ is a steady state of g, that is, $g_i(\widetilde{z}) = z_i$ for $i = 1, \ldots, n-1$, we want to show that there is a unique steady state of f, z, such that $\pi(z) = \widetilde{z}$.

Without loss of generality, suppose x_n depends on x_i for $i = 1, \dots, x_r$; and x_i for $i = s, \dots, n-1$ are the variables that depend on x_n. Notice that it may be the case that $r = 0$ or $s = n$; in that case x_n would depend on no variables or no variable would depend on x_n. Then, $f_n = f_n(x_1, \dots, x_r)$; also, for $i = s, \dots, n-1$ we have $f_i = f_i(x_{i1}, x_{i2}, \dots, x_n)$. Then, for $i = 1, \dots, s-1$ we have $g_i(\pi(x)) = f_i(x)$ and for $i = s, \dots, n-1$ we have $g_i = f_i(x_{i1}, x_{i2}, \dots, f_n(x_1, \dots, x_r))$.

Let z be a steady state of f. Then, for $i = 1, \dots, s-1$ we have $g_i(\pi(z)) = f_i(z) = z_i$; for $i = s, \dots, n-1$, $g_i(\pi(z)) = f_i(z_{i1}, z_{i2}, \dots, f_n(z_1, \dots, z_r)) = f_i(z_{i1}, z_{i2}, \dots, z_n) = z_i$. Then, $g(\pi(z)) = \pi(z)$.

Let $\widetilde{z}$ be a steady state of g. Define $z = (\widetilde{z}, f_n(z_1, \dots, z_r))$, that is, $z_n = f_n(z_1, \dots, z_r)$; we claim that z is a steady state of f. Notice that $\pi(z) = \widetilde{z}$. For $i = 1, \dots, s-1$ we have that $f_i(z) = g_i(\pi(z)) = g_i(\widetilde{z}) = z_i$; also, for $i = s, \dots, n-1$ we have that $f_i(z) = f_i(z_{i1}, z_{i2}, \dots, z_n) = f_i(z_{i1}, z_{i2}, \dots, f_n(z_1, \dots, z_r)) = g_i(\pi(z)) = z_i$; also, $f_n(z) = f_n(z_1, \dots, z_r) = z_n$. This shows that $z = (\widetilde{z}, f_n(z_1, \dots, z_r))$ is a steady state of f. We now show that z is unique. Let $z' = (z'_1, \dots, z'_n)$ be another steady state of f such that $\pi(z') = \widetilde{z}$; it follows that $z' = (\widetilde{z}, z'_n)$. Since $z'_n = f_n(z') = f_n(z'_1, \dots, z'_r) = f_n(z_1, \dots, z_r) = z_n$, then, $z = (\widetilde{z}, z_n) = z'$. Hence, the steady state is unique. □

Corollary 4.1.6 *Let f be a Boolean network and $g = f^R$ not empty. Then, there is a one to one correspondence between the set of steady states of f and the set of steady states of g.*

Corollary 4.1.7 *Let f be a Boolean network and $g = f^R$. If g is empty, then f has a unique steady state.*

Proof. Suppose g is empty. Without loss of generality, suppose x_1 is the last variable to be eliminated. That is, $h = f^{[x_2, \dots, x_n]}$ and $g = h^{[x_1]}$. Since the wiring diagram of $h : \{0,1\} \to \{0,1\}$ has only the vertex x_1 and it cannot have a self loop, it follows that h cannot be the Boolean function $h(x_1) = x_1$ or $h(x_1) = \neg x_1$. Then, h has to be the Boolean function $h(x_1) = 0$ or $h(x_1) = 1$ and hence it has a single steady state ($x_1 = 0$ or $x_1 = 1$, respectively). □

Corollary 4.1.8 *Let f be a Boolean network and $g = f^{[x_{i_1}, x_{i_2}, \dots, x_{i_k}]}$ with $k < n$. Then, f is a oscillatory system (the only attractors it has are periodic orbits) if and only if g is a oscillatory system.*

Proof. f is an oscillatory system if and only if f does not have any steady state if and only if g does not have any steady state if and only if g is an oscillatory system. □

Reduction Method and Topological Properties

We now show that the original and reduced network share topological properties. The next theorem states that the reduction method does not create new paths nor it changes their signs.

Theorem 4.1.9 *Let f be a Boolean network and $g = f^{[x_{i_1},x_{i_2},\dots,x_{i_k}]}$ with $k < n$. Then, if there is path from y to z in the wiring diagram of g, there is also a path from y to z in the wiring diagram of f (or equivalently, if there is no path from y to z in the wiring diagram of f, there is no path from y to z in the wiring diagram of g). Furthermore, if all paths involved in the reduction from y to z in the wiring diagram of f are positive (negative), the corresponding paths from y to z in the wiring diagram of g are positive (negative).*

Proof. We only need to prove the theorem for $k = 1$; the general case follows by induction. Without loss of generality we can assume that $g = (g_1, \dots, g_{n-1}) = f^{[x_n]}$. Suppose there is a path from y to z in the wiring diagram of g. Without loss of generality we can suppose the path is $y = x_1 \to x_2 \to \dots \to x_{r-1} \to x_r = z$. It follows that g_i depends on x_{i-1} for $i = 2, \dots, r$.

We claim that there is a path from x_1 to x_2 in the wiring diagram of f. If x_2 depends on x_n, $g_2 = f_2(x_{j_1}, \dots, x_{j_t}, f_n)$. Then, since g_2 depends on x_1, it follows that x_1 is one of the x_{j_l}'s or f_n depends on x_1; then, it follows that there is a path from x_1 to x_2 in the wiring diagram of f. If, on the other hand, x_2 does not depend on x_n, $g_2 = f_2(x_{j_1}, \dots, x_{j_t})$. Then, since g_2 depends on x_1, it follows that x_1 is one of the x_{j_l}'s; then, there is a path from x_1 to x_2 in the wiring diagram of f. Similarly, it follows that there is a path from x_{i-1} to x_i for $i = 2, \dots, r$ in the wiring diagram of f and hence, there is path from $y = x_1$ to $z = x_r$ in the wiring diagram of f.

Now, to conclude the proof of the theorem it is enough to show that if the paths $x_{n-2} \to x_{n-1}$ and $x_{n-2} \to x_n \to x_{n-1}$ in the wiring diagram of $f = (f_1, \dots, f_n)$ are positive, so is the path $x_1 \to x_3$ in the wiring diagram of $g = (g_1, \dots, g_{n-1})$. Since $f_n(x) = f_n(\dots, x_{n-2})$ and $f_{n-1}(x) = f_{n-1}(\dots, x_{n-2}, x_n)$, then $g_{n-1}(\dots, x_{n-2}) = f_{n-1}(\dots, x_{n-2}, f_n(\dots, x_{n-2}))$. To show that the path $x_{n-2} \to x_{n-1}$ in the wiring diagram of g is positive, we need to show that $g_{n-1}(\dots, 0) \leq g_{n-1}(\dots, 1)$. Since the paths $x_{n-2} \to x_{n-1}$ and $x_{n-2} \to x_n \to x_{n-1}$ in the wiring diagram of f are positive, then $f_{n-1}(\dots, 0, f_n(\dots, 0)) \leq f_{n-1}(\dots, 0, f_n(\dots, 1)) \leq f_{n-1}(\dots, 1, f_n(\dots, 1))$. Hence, $g_{n-1}(\dots, 0) \leq g_{n-1}(\dots, 1)$. Then, the last part of the theorem follows by induction. □

Corollary 4.1.10 *Let f be a Boolean network and $g = f^R$. Then, if there is a path from y to z in the wiring diagram of g, there is also a path from y to z in the wiring diagram of f.*

Corollary 4.1.11 *Let f be a Boolean network and $g = f^{[x_{i_1},x_{i_2},\dots,x_{i_k}]}$ with $k < n$. Then, if there is a feedback loop at y in the wiring diagram of g, there is also a feedback loop at y in the wiring diagram of f.*

Corollary 4.1.12 *Let f be a Boolean network and $g = f^R$. Then, if there is a self loop at y in the wiring diagram of g, there is also a feedback loop at y in the wiring diagram of f.*

Example 4.1.13 *The next example shows how the reduction method can allow to detect nonfunctional paths. Consider the Boolean network:*
$f = (f_1, f_2, f_3, f_4) = (\neg x_1, x_3 \vee \neg x_4, x_1, x_1)$ with wiring diagram shown in Figure 4.3. If we reduce the network by deleting vertices x_3, x_4 we obtain $f^{[x_3,x_4]} = g = (g_1, g_2) = (\neg x_1, 1)$. We can clearly see that the reduced network does not have any path from x_1 to x_2. Also, the reduced network does not have any steady state; hence, by Theorem 4.1.5, f does not have steady states.

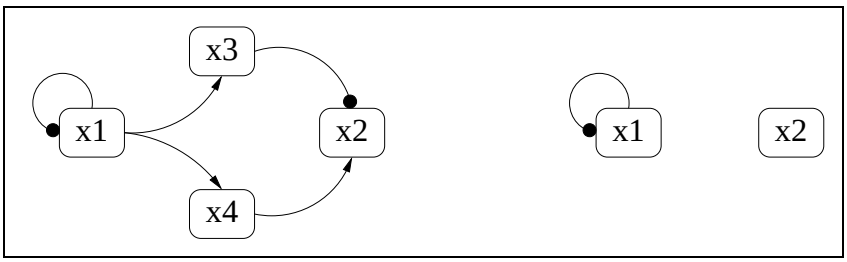


Figure 4.3: Wiring diagram of the full (left) and reduced (right) network for Example 4.1.13.

Example 4.1.14 *The next example shows how the reduction method can allow to detect nonfunctional feedback loops. Consider the Boolean network:*
$f = (f_1, f_2, f_3, f_4) = (x_2, x_3 \vee x_4, x_1, \neg x_1)$ with wiring diagram shown in Figure 4.4. We can see that there are two feedback loops (one positive and one negative). If we reduce the network by deleting vertices x_3, x_4 we obtain $f^{[x_3,x_4]} = g = (g_1, g_2) = (x_2, 1)$. We can clearly see that the reduced network does not have any feedback loop. Also, it is easy to see that g has a unique steady state, (1,1); hence, by Theorem 4.1.5 f has a unique steady state.

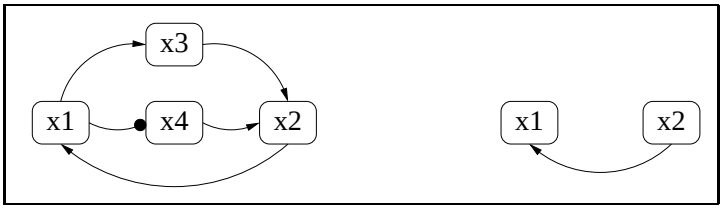


Figure 4.4: Wiring diagram of the full (left) and reduced (right) network for Example 4.1.14.

4.1.3 Application

We now consider the Boolean model presented in [89]. It is a model for Th-lymphocyte differentiation. Its wiring diagram is given in Figure 4.5. The variables and Boolean functions of the model are given as follows:

variable	Boolean function
x_1=INF-γ	$f_1 = x_9 \vee x_{11}$
x_2=IL-4	$f_2 = \neg x_7 \wedge x_{12}$
x_3=IL-12	$f_3 = 0$
x_4=IFN-γ R	$f_4 = x_1 \wedge \neg x_{10}$
x_5=IL-4R	$f_5 = x_2 \wedge \neg x_{10}$
x_6=IL-12R	$f_6 = x_3 \wedge \neg x_8$
x_7=STAT1	$f_7 = x_4$
x_8=STAT6	$f_8 = x_5$
x_9=STAT4	$f_9 = x_6 \wedge \neg x_{12}$
x_{10}=SOCS1	$f_{10} = x_7 \vee x_{11}$
x_{11}=T-bet	$f_{11} = (x_7 \vee x_{11}) \wedge \neg x_{12}$
x_{12}=GATA-3	$f_{12} = x_8 \wedge \neg x_{11}$

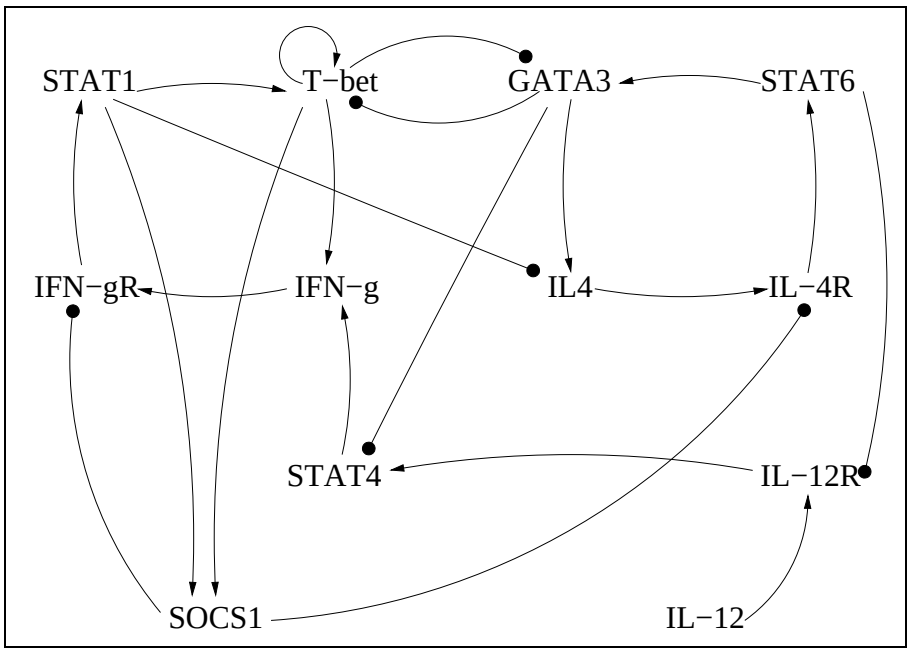


Figure 4.5: Wiring diagram for the Th-Lymphocyte differentiation network.

Notice that this Boolean network has $2^{12} = 4096$ states.

In Example 4.1.4 we observed that the reduction method gives the following reduced model with wiring diagram given in Figure 4.6:

variable	Boolean function
x_{11}=T-bet	$h_{11} = x_{11} \wedge \neg x_{12}$
x_{12}=GATA-3	$h_{12} = \neg x_{11} \wedge x_{12}$

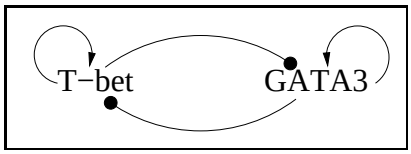


Figure 4.6: Wiring diagram for the reduced network of the Th-Lymphocyte differentiation network.

Notice that the reduced network has only $2^2 = 4$ states which is about 0.1% of the number of states of the full network. It is important to mention that our reduction method collapsed the positive feedback loop GATA-3→ IL4→ IL4-R→ STAT6→ GATA-3 to a single self loop at a vertex. It is easy to see that there are 3 steady states: $(\mathbf{0}, \mathbf{0}), (\mathbf{0}, \mathbf{1}), (\mathbf{1}, \mathbf{0})$. Furthermore, we can explain the nature of the steady states: if x_{12} =GATA-3=0, then x_{11} =T-bet can be 0 or 1 for (x_{11}, x_{12}) to be a steady state; on the other hand, if x_{12} =GATA-3=1, then x_{11} =T-bet must be 0. Since the reduced network has 3 steady states, by Theorem 4.1.5, the larger network has also 3 steady states: $s_1 = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, \mathbf{0}, \mathbf{0}), s_2 = (0, 1, 0, 0, 1, 0, 0, 1, 0, 0, \mathbf{0}, \mathbf{1}), s_3 = (1, 0, 0, 0, 0, 0, 0, 0, 0, 1, \mathbf{1}, \mathbf{0})$. In the reduced network, the existence of the self loops at GATA3, T-bet and the positive feedback loop T-bet$\rightleftarrows$ GATA3 suggests that they are the key to the dynamical properties; this in turn suggests that the corresponding feedback loops in the larger network are determining factors on the dynamics.

Now, let us see how the reduced network can help us understand the larger network. For the reduced network, it is not difficult to see that deleting one or both of the loops at GATA3 or T-bet results in the loss of the steady state (0,0). On the other hand, deleting one of the edges T-bet→ GATA-3, GATA-3→ T-bet does not change the steady states; however, if we delete both edges, a fourth steady state (1,1) is created. It is important to notice that this information can be easily obtained. We can expect that the larger network has similar properties; to check this we study the effect of deleting edges: deleting the loop at T-bet and other edges so that we do not have a feedback loop at T-bet, or any edge in the feedback loop [IL-4,IL-4R, STAT6,GATA-3] results in the loss of the steady state s_1 that corresponds to (T-bet,GATA3)=(0,0). On the other hand, deleting one of the edges T-bet→ GATA-3, GATA-3→ T-bet does not change the steady states; however, deleting both edges and other edges so that we do not have a path from T-bet to GATA3 and GATA3 to T-bet, results in the creation of a fourth steady state corresponding to (T-bet,GATA3)=(1,1). All these properties of the larger network are consistent with those of the reduced network that only

has 0.1% of the number of states. In summary, the reduction method generated a small network that allowed to easily study the existence and type of steady states and the role of the feedback loops on the dynamics.

4.1.4 Discussion

Boolean networks have been successfully used in modeling; they provide a theoretical framework that allows for simulation, control theory and reverse engineering. Since their analysis is not a trivial task, many mathematical and computational tools have been developed.

In this section we have proposed a reduction method that although simple, it can make the analysis of Boolean networks easier. In particular, we applied the reduction method to analyze the steady states of a Boolean model for Th-lymphocyte differentiation. The reduction method was not only able to make the analysis of steady states easier but was also able to explain the role of feedback loops.

Future work is to study how the reduction method can help in the analysis of limit cycles of a Boolean network. Also, another future project is the generalization of the reduction method to general finite dynamical systems.

4.2 Reverse Engineering of Regulatory Networks

Due to the limited information and data for some regulatory networks, one often encounters the problem of wanting to choose one (or "few") model from a large family of models. An example of this is the process of estimating parameters of an ODE; where one is basically reverse-engineering the value of the parameters from some data. In this type of problems one assumes that the system has certain form which is reflected in the terms that show up in the ODE.

However, in cases where there is not enough information about the structure of a regulatory network, one has to consider the problem of reverse engineering a model or features of the model (such as the wiring diagram) entirely from data. This problem has been studied from a continuous and discrete point of view by several authors [53; 67; 118]. In this section we propose a method for reverse engineering finite dynamical systems and show that it can be applied to continuous systems as well.

Our method is a generalization of the method proposed in [53]. Their method is based on defining all possible wiring diagrams that are compatible with some data set (using monomials ideals) and using mathematical tools from algebraic geometry to find the minimal wiring diagrams (given by the primary decomposition of the monomial ideals). Our method is based in the same mathematical tools but differs in the fact that it can be used to recover "signed" dependencies. Given that most biological interactions are of an activatory or inhibitory type,

this additional property of our method is important.

We can illustrate the difference between the methods in the following example.

Example 4.2.1 *Consider a function (e.g. the rules under which a gene depends on other genes or the environment), h, which depends potentially on 3 variables and suppose we have the following (Boolean) data set (corresponding to, say, experiments in four different environments):*

Table 4.1: Data set for function $h(x)$.

inputs			output
x_1	x_2	x_3	$h(x)$
0	0	1	0
1	0	1	1
0	1	1	1
1	1	0	0

Notice that there are inputs for which we have no information: $(x_1, x_2, x_3) = (0, 0, 0)$,$(1, 0, 0)$, $(0, 1, 0)$, $(1, 1, 1)$.

The method proposed in [53] consists in comparing the inputs that gave different outputs and identifying the possible variables that caused the difference. For example, the input $(0, 0, 1)$ *produces an output of* 0*, whereas the input* $(1, 0, 1)$ *produces an output of* 1*. Since the only difference in the input is* x_1*, we conclude that the change in the output was due to* x_1*, which means that the function does depend on variable* 1*. By making all possible comparisons we have that h depends on variables:* 1 *and* 2 *and* $\{2$ *or* $3\}$ *and* $\{1$ *or* $3\}$*. Then, since the primary decomposition of the monomial ideal is* $< x_1, x_2, x_2x_3, x_1x_3 >=< x_1, x_2 >$*, we obtain that the (in this case unique) minimal set of variables that can explain the data is* $\{x_1, x_2\}$ *and furthermore, a polynomial that fits the data is* $Q(x) = x_1 + x_2$*. It is easy to check that this polynomial in fact reproduces the table; however, from a biological point of view it has the following drawback: Since* $Q(0, 0) = 0$ *and* $Q(1, 0) = Q(0, 1) = 1$ *we should conclude that* x_1 *and* x_2 *are activators for this function and by having both activators equal to 1 we should also have* $Q(1, 1) = 1$*; however,* $Q(1, 1) = 1 + 1 = 0$ *(arithmetic modulo 2). This implies that* x_1,x_2 *are not activators for this function; in fact, it is impossible to assign signs to the variables (Figure 4.7, right).*

On the other hand, our method not only keeps track of possible variables that cause differences in outputs, but also whether the variables went up or down, that is, their sign. For example, the input $(0, 0, 1)$ *produces an output of* 0*, whereas the input* $(1, 0, 1)$ *produces an output of* 1*. Since the only difference in the input is* x_1 *and it went from* 0 *to* 1*, we conclude that the change in the output was due to a positive change in* x_1*, which means that the function*

does depend on variable 1 *positively (or* 1^+*). By making all possible comparisons we have that* h *depends on variables:* 1^+ *and* 2^+ *and* $\{2^-$ *or* $3^+\}$ *and* $\{1^-$ *or* $3^+\}$*. Then, according to our method the (in this case unique) minimal set of variables that can explain the data is* $\{1^+, 2^+, 3^+\}$ *and furthermore, a polynomial that fits the data is* $Q(x) = (x_1 + x_2 + x_1x_2)x_3$ *(we will see the details later). It is easy to check that this polynomial in fact reproduces the table; furthermore, we can assign signs to the variables (already given by our method), in this case all variables are activators (Figure 4.7, left).*

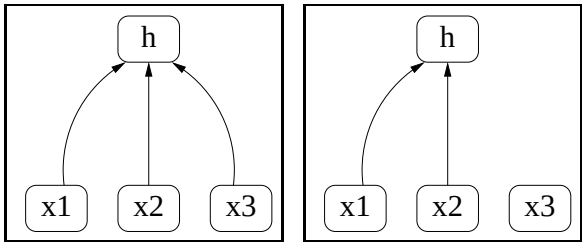


Figure 4.7: Wiring diagram for h obtained using our method (left) and current methods (right). Note: the diagram on the right is unsigned.

We can see that although both methods produce minimal sets, our method can produce signed minimal sets.

4.2.1 Preliminaries

Let h be a function on n variables (not necessarily Boolean), $x_1, \ldots, x_n$.

We say that h depends on x_i or that x_i is a non trivial variable of h if there exist a, b such that $h(x_1, \ldots, x_{i-1}, a, x_{i+1}) \neq h(x_1, \ldots, x_{i-1}, b, x_{i+1})$ for some values $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$. That is, there is an instance where changing the value of x_i changes the output of h.

We say that h depends positively on x_i or that x_i is an activator of h, if h depends on x_i and $h(x_1, \ldots, x_{i-1}, a, x_{i+1}) \leq h(x_1, \ldots, x_{i-1}, b, x_{i+1})$ for all values $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$, $a \leq b$. That is, increasing the value of x_i increases the output of h.

We say that h depends negatively on x_i or that x_i is an inhibitor of h, if h depends on x_i and $h(x_1, \ldots, x_{i-1}, a, x_{i+1}) \geq h(x_1, \ldots, x_{i-1}, b, x_{i+1})$ for all values $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$, $a \leq b$. That is, increasing the value of x_i decreases the output of h.

A function where each non trivial variable is an activator is called a monotone function. A function where each non trivial variable is an activator or an inhibitor is called a unate function (this is a generalization of unate Boolean functions in 3.1). Equivalently, h is unate if there is a partition of $\{1, \ldots, n\}$, P, N such that $x_P \leq y_P$ and $x_N \geq y_N$ imply that $h(x) \leq h(y)$. Denote $UN = \{h : h \text{ is unate}\}$ and $UN(P, N) = \{h : h \text{ is unate such that } x_P \leq y_P \text{ and } x_N \geq y_N \text{ imply that } h(x) \leq h(y)\}$.

Suppose we have partial information about a function; for example, that we know that the function satisfies $h(s_i) = v_i$ for all $i = 1, \dots, m$. That is, our data set is $D = \{(s_i, v_i) : i = 1, \dots, m\}$. Denote $UN(P, N; D) = \{h : h \in UN(P, N), h(s) = v \text{ for all } (v, s) \in D\}$. What we want is a unate function h such that $h(s) = v$ for all $(s, v) \in D$; furthermore, we want the function to depend on the least number of variables, P, N. That is, we want P, N minimal such that $UN(P, N; D)$ is not empty. There may be many such P, N and our algorithm finds all of them. We will refer to the sets $M = P \cup N$ as minimal sets (P, N will be called a minimal pair).

The following theorem states that when the data set has "unate properties" we can guarantee that a unate function that fits the data exists.

Theorem 4.2.2 *Suppose we have a data set, D. Let P, N be a partition of $\{1, \dots, n\}$. If $s_P \leq t_P$ and $s_N \geq t_N$ imply that $v \leq w$ for all $(s, v), (t, w) \in D$ then $UN(P, N; D)$ is not empty. Furthermore, if $h(x) \in UN(P, N; D)$ then $L(x) \leq h(x) \leq U(x)$ for all x, where $L(x) = max\{v : s_P \leq x_P, s_N \geq x_N, (s, v) \in D\}$ and $U(x) = min\{v : x_P \leq s_P, x_N \geq s_N, (s, v) \in D\}$.*

Proof. It follows from the definition of $L(x), U(x)$ that that $L(s) = U(s) = v$ for all $(s, v) \in D$ and that $L(x) \leq U(x)$ for all x. Now we show that $L(x), U(x) \in UN(P, N; D)$: If $x_P \leq y_P$ and $x_N \geq y_N$ then it is not difficult to show that $\{v : s_P \leq x_P, s_N \geq x_N, (s, v) \in D\} \subseteq \{v : s_P \leq y_P, s_N \geq y_N, (s, v) \in D\}$ and $\{v : s_P \leq x_P, s_N \geq x_N, (s, v) \in D\} \subseteq \{v : s_P \leq y_P, s_N \geq y_N, (s, v) \in D\}$, therefore $L(x) \leq L(y)$ and $U(x) \leq U(y)$. That is, the activators, inhibitors of $L(x)$ (and $U(x)$) are in $\{x_k : k \in P\}$, $\{x_k : k \in N\}$, respectively. Hence, $L(x), U(x) \in UN(P, N; D)$.

Now, suppose $h(x) \in UN(P, N; D)$. For any x and $(s, v) \in D$ we have that if $s_P \leq x_P$ and $s_N \geq x_N$ then $v = h(s) \leq h(x)$; it follows that $L(x) \leq h(x)$. Also, for any x and $(s, v) \in D$ we have that if $x_P \leq s_P$ and $x_N \geq s_N$ then $h(x) \leq h(s) = v$; it follows that $h(x) \leq U(x)$. □

The following theorem states that when the data set has unate properties with respect to some variables then we can guarantee that a unate function that fits the data exists.

Corollary 4.2.3 *Suppose we have a data set, D. Let P, N be disjoint subsets of $\{1, \dots, n\}$ ($P \cup N$ is not necessarily equal to $\{1, \dots, n\}$). If $s_P \leq t_P$ and $s_N \geq t_N$ imply that $v \leq w$ for all $(s, v), (t, w) \in D$ then $UN(P, N; D)$ is not empty. Furthermore, if $h(x) \in UN(P, N; D)$ then $L(x) \leq h(x) \leq U(x)$ for all x, where $L(x) = max\{v : s_P \leq x_P, s_N \geq x_N, (s, v) \in D\}$ and $U(x) = min\{v : x_P \leq s_P, x_N \geq s_N, (s, v) \in D\}$.*

Proof. The proof follows from restricting the data set to the variables in $P \cup N$ and applying the theorem above. □

4.2.2 Algorithm

In this section we present the algorithm to find minimal wiring diagrams of finite dynamical systems (not necessarily Boolean systems).

Suppose we have a data set, D with m elements. Let $R = \mathbb{F}_2[x_1, \ldots, x_n, y_1, \ldots, y_n]$ be a ring with $2n$ variables (x_i will represent i^+ and y_i will represent i^-).

For $(s, v), (t, w) \in D$ such that $v < w$ define

$$P(s,t) = \prod_{s_i<t_i} x_i \prod_{s_j>t_j} y_j$$

For example, if $s = (0, 2, 0, 4, 1, 0)$, $t = (3, 0, 0, 4, 0, 1)$ then $P(s_i, s_j) = x_1x_6y_2y_5$. This means that according to this comparison the unate function depends on the variables 1^+ or 2^- or 5^- or 6^+.

Consider the set $W = \{P(s,t) : (s, v), (t, w) \in D, v < w\}$ and let I be the square-free monomial ideal generated by W.

Theorem 4.2.4 *Let P, N be disjoint subsets of $\subseteq \{1, \ldots, n\}$. $UN(P, N; D) \neq \emptyset$ if and only if $I \subseteq < x_i, y_j : i \in P, j \in N >$.*

Proof. Suppose $h \in UN(P, N; D)$. Let $P(s, t)$ be one of the generators of I, we have that $(s, v), (t, w) \in D$ and $v < w$; then $h(s) = v < w = h(t)$. Since $h(s) < h(t)$ it follows that $s_i < t_i$ for some $i \in P$ or $s_j > t_j$ for some $k \in N$. Then, $P(s, t) \in < x_i, y_j : i \in P, j \in N >$ (one factor of $P(s, t)$ is in the ideal). This implies that $I \subseteq < x_i, y_j : i \in P, j \in N >$.

Now suppose that $I \subseteq < x_i, y_j : i \in P, j \in N >$, then $P(s, t) \in < x_i, y_j : i \in P, j \in N >$ for all $(s, v), (t, w) \in D$ such that $v < w$; that is, $P(s, t)$ has as a factor x_i for some $i \in P$ or y_j for some $j \in N$. Then, for $(s, v), (t, w) \in D$ such that $v < w$ it follows that $s_i < t_i$ for some $i \in P$ or $s_j > t_j$ for some $j \in N$. Hence, it follows that for all $(s, v), (t, w) \in D$, $s_P \leq t_P$ and $s_N \geq t_N$ imply that $v \leq w$. By Corollary 4.2.3 we have that $UN(P, N; D) \neq \emptyset$. □

This means that the ideal I encodes all possible sets that are compatible with the data. Let $I_1, \ldots, I_r$ be the ideals in the primary decomposition of I, $I = I_1 \cap I_2 \cap \ldots \cap I_r$. It turns out that I_j are generated by set of the form $\{x_{k_1}, \ldots, x_{k_l}\}$; that is, the generators are single variables. Denote by M_k the set of generators of I_k

The following corollary states that the sets M_k give the minimal sets such that there is a function for which it is possible to assign a sign to the inputs (each variable is either an activator or an inhibitor). The proof follows from the theorem above.

Corollary 4.2.5 *The minimal P, N such that $UN(P, N; D) \neq \emptyset$ are given by M_k where $P = \{i : x_i \in M_k\}$ and $N = \{j : y_j \in M_k\}$ are disjoint.*

Example 4.2.1 (cont.) From Table 4.1 we obtain that $W = \{x_1, x_2, y_2x_3, y_1x_3\}$. The primary decomposition of the ideal is $I =< x_1, x_2, x_3 > \cap < x_1, x_2, y_1, y_2 >$. Then, the generators are $\{x_1, x_2, x_3\}$ for the first and $\{x_1, x_2, y_1, y_2\}$ for the second. Therefore, we have the (unique) minimal set $\{1^+, 2^+, 3^+\}$.

Example 4.2.6 *Consider the data* $S = \{(0,1,2,1,0), (0,1,2,1,1), (0,1,2,1,4), (3,0,0,0,0), (1,1,1,1,3)\}$, $V = \{0,0,1,3,4\}$. *Then* $W = \{x_5, x_1y_2y_3y_4, x_1y_3x_5,\ x_5,\ x_1y_2y_3y_4y_5,\ x_1y_3x_5,\ x_1y_2y_3y_4y_5,\ x_1y_3y_5,\ y_1x_2x_3x_4x_5\}$.

The primary decomposition of the monomial ideal generated by W *is* $I =< x_1, x_5 > \cap < x_5, y_3 > \cap < x_5, y_2, y_5 > \cap < x_5, y_4, y_5 >$. *Therefore, the minimal sets are* $\{1^+, 5^+\}$ *and* $\{3^-5^+\}$. *Notice that in [53] the minsets found were* $\{1,5\}$, $\{2,5\}$, $\{3,5\}$,$\{4,5\}$*; however, we have shown that only two of those admit a sign assignment.*

4.2.3 Application to Continuous Data

One way to apply our algorithm to continuous data is to discretize it (see [25] for an example of a discretization method) and apply it to the discretized data set. However, this is prone to errors or overfitting due to discretization thresholds. To overcome this we developed a method to apply our algorithm to continuous data while maintaining the use of discretization/thresholds to a minimum.

Suppose we have a data set, D with m elements. Now the elements of D are continuous; that is, $s \in \mathbb{R}^n$,$v \in \mathbb{R}$ for all $(s, v) \in D$. Let $R = \mathbb{F}_2[x_1, \ldots, x_n, y_1, \ldots, y_n]$ be a ring with $2n$ variables, as before.

Let $\Theta = (\theta_1, \theta_2, \ldots, \theta_n) \in \mathbb{R}^n, \theta \in \mathbb{R}$ be thresholds.

For $(s, v), (t, w) \in D$ such that $\theta < w - v$ define

$$P(s,t) = \prod_{\theta_i < t_i - s_i} x_i \prod_{-\theta_j > t_j - s_j} y_j$$

Consider the set $W = \{P(s,t) : (s, v), (t, w) \in D, \theta < w - v\}$ and let I be the square-free monomial ideal generated by W. The other steps of the algorithm remain unchanged.

Notice that our method is in essence discretizing the differences $t_i - s_i$ (and $w - v$), whereas a typical discretization method would compare the discretization of t_i with the discretization of s_i. That is, our method performs about half the number of discretizations and, therefore, it is less prone to errors. Furthermore, the thresholds can be chosen based on how how much noise or stochasticity the system has.

4.2.4 Reverse Engineering Systems

Given a dynamical system (continuous or discrete) in n variables, a typical data set consists of one or more time series $T = \{s_1 \to s_2 \to \dots s_k = (s_{k1}, \dots, s_{kn}) \dots \to s_m\}$ (each s_k has n entries), which is data obtained by taking measurements at different times. From the time series, we can obtain the data set for function k by considering: $S = \{s_1, s_2, \dots, s_{m-1}\}$, $V = \{s_{2k}, s_{3k}, \dots, s_{mk}\}$; that is, we consider all input vectors and the k-th entry of the corresponding output. Then, we can apply our method to $D = \{(s_i, s_{i+1k}) : i = 1, \dots, m-1\}$.

Remark 4.2.7 *In applications it may be of interest to preprocess the time series before applying our method. For example, if we want to reverse engineer the rate of change (i.e. from an ODE model) $\frac{dx_k}{dt} = h(x)$, we may want to modify the set V to represent not the current value of the system but the rate of change of the system. That is, $V = \{(s_{2k} - s_{1k})/\ \triangle_1 , (s_{3k} - s_{2k})/\ \triangle_2, \dots, (s_{mk} - s_{m-1k})/\ \triangle_{m-1}\}$, this would provide a better representation of the problem in question (other approximations of the rate of change could be used as well).*

Remark 4.2.8 *In applications it may be of interest to leave some variables "free", in the sense that we should not require them to be neither activators or inhibitors. For example, due to decay or degradation our method might interpret that a variable is its own repressor. This is better seen in a ODE, $\frac{dx_k}{dt} = h(x) - rx_k$; our method might interpret the data caused by the decay term $-rx_k$ as a negative regulation. We can do this by slightly modifying the algorithm to*

$$P(s,t) = \prod_{\theta_i < t_i - s_i, i \neq k} x_i \prod_{-\theta_j > t_j - s_j, i \neq k} y_i \prod_{\theta_k < |t_k - s_k|} x_k$$

That is, we disregard the sign of x_k.

4.2.5 Example

We applied our algorithm to reverse engineer the ODE system consisting of 5 nodes presented in [24, Chapter 6]:

$$\begin{aligned}
\tfrac{dG_1}{dt} &= f(G_1(t))h(G_3(t)) - G_1(t)\\
\tfrac{dG_2}{dt} &= f(G_1(t))h(G_3(t)) - G_2(t)\\
\tfrac{dG_3}{dt} &= h(G_1(t))h(G_3(t))h(G_5(t)) - G_3(t)\\
\tfrac{dG_4}{dt} &= h(G_3(t)) - G_4(t)\\
\tfrac{dG_5}{dt} &= f(G_1(t))f(G_3(t)) - G_5(t)
\end{aligned}$$

where $f(G) = 1 + \frac{G}{0.01+G}$ corresponds to activation, $h(G) = \frac{0.01}{0.01+G}$ corresponds to inhibition and $-G_i(t)$ corresponds to decay. Notice that f and h are increasing and decreasing, respectively. The wiring diagram of the ODE is shown in Figure 4.8.

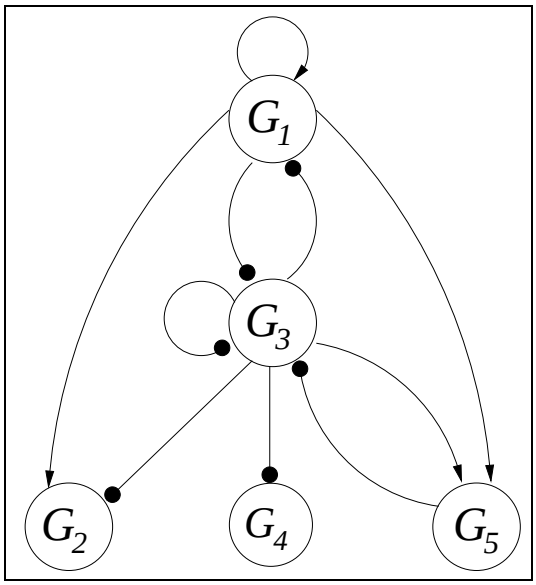


Figure 4.8: Wiring diagram of the ODE system.

We considered two initializations, $G(0) = (G_1(0), G_2(0), G_3(0), G_4(0), G_5(0)) = (0, 0, 0, 0, 0)$ and $G(0) = (1, 1, 1, 1, 1)$. We considered wild type (the original ODE) and five knockout time courses (ODE obtained by making $\frac{dG_j}{dt} = G_j(0) = 0$). Each time course consists of measurements of the system at $t = 0, 2, 4, \ldots, 20$; that is, we have a total of 12 time courses of length 10 each.

The result of our algorithm is shown on Figure 4.9 (left). We chose a threshold of 5% of the range of the corresponding variable (e.g. if variable j varies between 0 and 3.5 then the threshold is $\theta_j = 3.5 \times 0.05$). For comparison purposes we also show the wiring diagram obtained without requiring the unate properties (as in [53; 67] and [24, Chapter 6]), Figure 4.9 (right).

Our method performed better in two aspects (because of Remark 4.2.8 we will disregard self regulation for the comparison): First, we were able to detect all interactions, whereas the other wiring diagram missed one ($G_5 \to G_3$); furthermore, we also recovered the correct signs, whereas the other wiring diagram does not give such information. In particular, there were no false negatives. Second, even the false positives can be explained. For example, the edge $G_1 \to G_4$ is consistent with the fact that G_1 is an inhibitor for G_3 and G_3 is an inhibitor for G_4, then we have that G_1 regulates G_4 as an indirect activator. It is not difficult to show that all false positives have this property.

The reason for indirect regulations to show up as direct is simple: the time elapsed from one measurement to the next allows indirect regulations to appear in a single time step; for example, an increase in G_1 decreases G_3 which increases G_4; so unless the time intervals are small enough, we will see that an increase in G_1 caused an increase in G_4. This implies that after using our method (or any reverse-engineering method) it is possible to improve

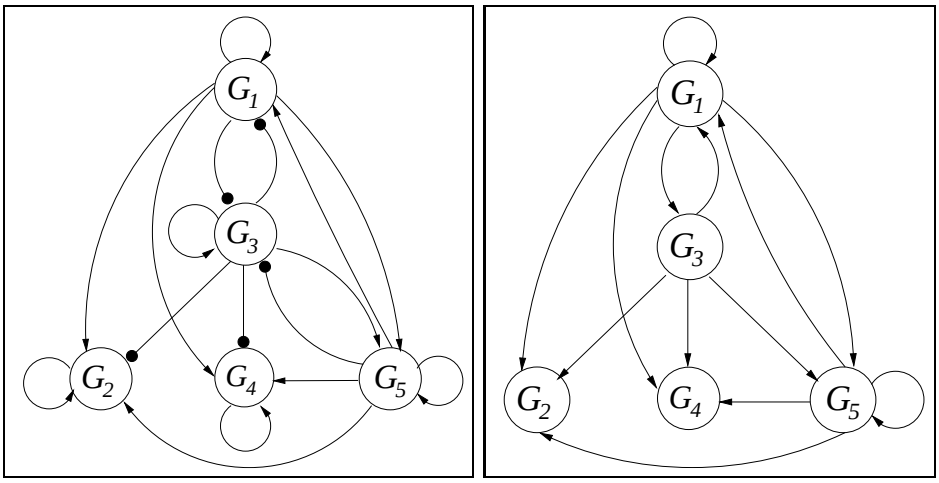


Figure 4.9: Reverse engineered wiring diagram using our method (left) and previous methods (right). Note: the edges on the right are unsigned.

the wiring diagram by taking into consideration these topological features; for example, if we have an edge $i \to j$ and a path $i \to k \to j$, it is possible that the $i \to j$ is simply an artifact of the algorithms and therefore could be deleted.

4.2.6 Discussion

Due to the limited information about regulatory networks many reverse engineering methods have been proposed for different mathematical frameworks [53; 67; 118]. For these type of problems, discrete models are a good choice because they focus on qualitatively features and can capture important features of the system (see Chapter 2., also [3]). Previous methods [53; 67] have the draw back that they only look for unsigned interactions.

Our method has the advantage that it can detect signed interactions and it can be applied to continuous data, not relying on any discretization method. We have seen in 4.2.5 that our method performed better than current methods and that even the existence of false positives could be explained and corrected.

Future work is to study how our method performs with real systems and with noisy/stochastic systems. Also, the fact that false positives may be detected can provide new ideas on how to reduce a wiring diagram to produce a minimal network. On the other hand, since Corollary 4.2.3 characterizes all functions corresponding to any minimal set, our method also has the potential to produce minimal models.

Chapter 5

Polynomial Form of Other Discrete Frameworks

Finite dynamical systems, that is, discrete dynamical systems with a finite state space, have been used extensively in systems biology to model a variety of biochemical networks, such as metabolic networks, gene regulatory networks, and signal transduction networks. Boolean networks and their generalization, the so-called multistate logical models [109; 110], are the main types of finite dynamical systems that have been used successfully in modeling biological networks (see, e.g., [3; 11; 28; 29; 68; 75; 92; 93]). Petri nets have also been shown to be a good modeling paradigm for this field (see, e.g., [15; 30; 40; 64]). Together, these model types represent a large class of discrete models in systems biology that are capable of simulating deterministic as well as stochastic processes.

Several tools and techniques have been developed to simulate and analyze discrete models. For logical models the open-source software GINsim (http://gin.univ-mrs.fr) is available, and for Petri nets, the user has access to a wide variety of software. The laboratory of M. Heiner provides software with a particular focus on applications to systems biology (http://www-dssz.informatik.tu-cottbus.de/). Analysis tools for logical models, including Boolean networks, are described in [80]. In addition to a variety of simulation and visualization tools, other graph theoretic analysis tools are available for the identification of steady states and strongly connected components of the regulatory graph. There are algorithms to compute dead states (steady states), as well as T-invariants and P-invariants, which can be computed via linear algebra methods. A survey of the use of Petri nets in systems biology is [85].

In this chapter we describe how PDS can be used to study these types of models and make accessible a broad range of mathematical tools for model analysis. The fundamental observation underlying this framework is that logical models and k-bounded Petri nets are particular instantiations of what we shall call "algebraic models," that is, time-discrete dynamical systems

$$f = (f_1, \ldots, f_n) : \mathbb{F}^n \longrightarrow \mathbb{F}^n$$

where each coordinate function f_i is a function of the n variables $x_1, \ldots, x_n$, each of which takes on values in a finite set $\mathbb{F}$ with algebraic structure, and each f_i is a polynomial. Aside from the mathematical simplicity of their definition, an important feature of polynomial dynamical systems is that one can employ a number of mathematical tools for their analysis. For our purposes, the principal tool is the capability to symbolically solve systems of nonlinear polynomial equations quite efficiently. This can be used, for instance, to compute the steady states and other features of an algebraic model.

We consider models of biological systems, such as biochemical networks with n interacting molecular species whose states can be described by an n-tuple with entries from a finite set $\mathbb{F}$. The model consists of a set of rules that allow the system to evolve from one state to the next, so that it can be represented as a time-discrete dynamical system $f : \mathbb{F}^n \longrightarrow \mathbb{F}^n$. Both logical models and k-bounded Petri nets are of this form. In particular, they can be seen as PDSs. For example, in a model of a gene regulatory network, the set $\mathbb{F}$ is $\{0, 1\}$, representing the state of a gene as either ON (1) or OFF (0).

In the next sections, we will show that logical models and bounded Petri nets can be represented as polynomial dynamical systems. One can then use powerful symbolic computation software, from open source specialty packages such as *CoCoA* (http://cocoa.dima.unige.it/), *Singular* (http://www.singular.uni-kl.de), and *Macaulay2* (http://www.math.uiuc.edu/Macaulay2) to general commercial packages such as *Mathematica* and *Maple*. In this section, we will use *Macaulay2* for all computations.

This section is based on a published paper written in collaboration with Abdul Salam Jarrah and Reinhard Laubenbacher [115]. My contributions to the paper were the algorithms and coding to transform logical models and Petri nets to PDSs.

5.1 Algorithm for Logical Models

Logical models have been used for modeling biological phenomena and have been useful for obtaining valuable insight and qualitative information [110]. Such analysis is based on the topology of the network and the type of regulation of the different interactions in the network, which are believed to be the key features that determine dynamical properties [3; 110]. Logical models have the advantage of being intuitive and relatively simple to construct, even with no information about reaction rates or functionality, and still keeping qualitative information [3]. However, their analysis is not a trivial task. For example, the problem of finding the steady states of a logical model is NP-complete (even to determine the existence of steady states is NP-complete) [121].

5.1.1 Definition of Logical Models

There are several ways to define logical models [15; 18; 80], and here we use the definition given in [80].

Definition 5.1.1 *A logical model is a triple $(\mathcal{V}, \mathcal{E}, \mathcal{K})$, where:*

1. *$\mathcal{V} = \{v_1, \dots, v_n\}$ is the set of vertices or nodes. Each v_i has a maximum expression level, m_i. The set $\mathcal{S} = [0, m_1] \times \dots \times [0, m_n]$ (Cartesian product) is called the* state space *of the logical model and its elements are called states.*
2. *$\mathcal{E}$ is the set of arcs. The elements of $\mathcal{E}$ have the form (v_i, v_j, θ), where $1 \leq \theta \leq m_i$ is called a threshold for (v_i, v_j). Let $\mathcal{I}(j) = \{v : (v, v_j, \theta) \in \mathcal{E}\}$, called input of v_j, be the set of vertices that have an edge ending in v_j. Notice that an arc (v_i, v_j) is allowed to have multiplicity corresponding to different thresholds; the number of thresholds is denoted by $m_{i,j}$. The thresholds are indexed in increasing order, $1 \leq \theta_{i,j,1} < \dots < \theta_{i,j,m_{i,j}} \leq m_i$; $\theta_{i,j,k}$ is the k-th threshold for the arc (v_i, v_j). By convention we define $\theta_{i,j,m_{i,j}+1} = m_i + 1$ (actually, $\theta_{i,j,m_{i,j}+1}$ can be defined as any number greater than m_i) and $\theta_{i,j,0} = 0$. Let $x = (x_1, \dots, x_n)$ be a state; we say that the k-th interaction for input v_i of v_j is active if $\theta_{i,j,k} \leq x_i < \theta_{i,j,k+1}$; we denote this by $\Theta^{i,j}(x_i) = k$.*
3. *$\mathcal{K} = \{K_i : \prod_{v_j \in \mathcal{I}(i)} [0, m_{j,i}] \to [0, m_i], i = 1, \dots n\}$ is the set of parameters.*

Consider a state $x = (x_1 \dots, x_n)$. The future value for node v_i, with $\mathcal{I}(i) = \{v_{i_1}, \dots, v_{i_r}\}$, is determined as follows: Compute $\Theta^i(x_{i_1}, \dots, x_{i_r}) = (\Theta^{i_1,i}(x_{i_1}), \dots, \Theta^{i_r,i}(x_{i_r})) = (k_1, \dots, k_r)$, indicating that the k_j-th interaction for input v_{i_j} of v_i is active; then compute $f_i(x) = K_i(k_1, \dots, k_r)$.

This last value determines whether the value of v_i tends to increase, decrease or remain the same. To be precise, the future value of v_i is given by:
$g_i(x) = \rho(x_i, f_i(x))$, where

$$\rho(t, u) = \begin{cases} t + 1, & \text{if } u > t \\ t, & \text{if } u = t \\ t - 1, & \text{if } u < t \end{cases}$$

With this notation, $g = (g_1, \dots, g_n) : \mathcal{S} \to \mathcal{S}$ is a finite dynamical system, and so is $f = (f_1, \dots, f_n) : \mathcal{S} \to \mathcal{S}$. Notice that g ensures that each variable changes at most one unit (continuity), whereas f does not.

The dynamics of a logical model is given by the *phase space*, defined as the graph $\mathcal{S}$ with an edge from x to y if $\{i : x_i \neq y_j\} = \{j\}$ for some j and $y_j = g_j(x)$ (asynchronous dynamics) or as the graph $\mathcal{S}$ with an edge from x to y if $y = g(x)$ (synchronous dynamics). Notice that g may be replaced by f if we do not require continuity.

An edge $i \to j$ is called *positive* (*negative*) if increasing the i-th entry of x causes $g_i(x)$ to increase (decrease) (using the natural order on the integer sets $[0, m_i]$. If changing the i-th entry of x does not cause any change in $g_i(x)$, we say that the edge is nonfunctional.

5.1.2 Polynomial Form of Logical Models

Let $(\mathcal{V}, \mathcal{E}, \mathcal{K})$ be a logical model as above. Choose a prime number p such that $p \geq m_i + 1$ for all $1 \leq i \leq n$ ($[0, m_i]$ has $m_i + 1$ elements) and let $\mathbb{F} = \mathbb{F}_p = \{0, 1, \ldots, p-1\}$ be the field with p elements. Note that we may consider $\mathcal{S} \subseteq \mathbb{F}^n$.

Consider a vertex v_i and let g_i (or f_i if we do not require continuity) be its coordinate function. Our goal is to represent g_i as a polynomial in terms of its inputs, say $x_{i_1}, \ldots, x_{i_r}$. That is, we need a polynomial function defined on $\mathbb{F}^r$ with values in $\mathbb{F}$. Denote $a \wedge b = min\{a, b\}$, using the natural order on the set $\mathbb{F}$, viewed as integers. To extend the domain of g_i from $\prod_{v_j \in \mathcal{I}(i)} [0, m_{j,i}]$ to $\mathbb{F}^r$ we define $g_i(x_{i_1}, \ldots, x_{i_r}) = g_i(x_{i_1} \wedge m_{i_1}, \ldots, x_{i_r} \wedge m_{i_r})$ for $(x_{i_1}, \ldots, x_{i_r}) \in \mathbb{F}^r$. The polynomial form of $g_i : \mathbb{F}^r \to \mathbb{F}$ is then

$$g_i(x) = \sum_{(c_{i_1}, \ldots, c_{i_r}) \in \mathbb{F}^r} g_i(c_{i_1}, \ldots, c_{i_r}) \prod_{v_j \in \mathcal{I}(i)} (1 - (x_j - c_j)^{p-1}),$$

where the right-hand side is computed modulo p.

Remark 5.1.2 *Many published logical models are Boolean, that is, $m_i = 1$ for all i. In this case we have $g_i = f_i$ and can write them explicitly in terms of the parameters:*

$$g_i(x) = \sum_{J \subseteq \mathcal{I}(i)} K_i(J) \prod_{v_j \in J} x_j \prod_{v_j \in \mathcal{I}(i) \setminus J} (1 - x_j).$$

Example 5.1.3 *Consider a Boolean logical model where x_2, x_3, x_4 regulate x_1, that is, $\mathcal{I}(1) = \{2, 3, 4\}$. Suppose that $K_1(\{\}) = K_1(\{3\}) = K_1(\{4\}) = K_1(\{3, 4\}) = 1$ and $K_1(J) = 0$ in all other cases. Using Remark 5.1.2, the polynomial form of g_1 is*
$g_1(x) = (1 - x_2)(1 - x_3)(1 - x_4) + (1 - x_2)x_3(1 - x_4) + (1 - x_2)(1 - x_3)x_4 + x_3(1 - x_2)x_4$, *which simplifies to $g_1 = 1 + x_2$.*

5.1.3 Functionality of Edges of Boolean Models

For $x \in \mathbb{F}_2^n$ and $a \in \mathbb{F}_2$ we denote $x^{j=a} = (x_1, \ldots, x_{j-1}, a, x_{j+1}, \ldots, x_n)$. An edge $j \to i$ is called *functional* if $g_i(x^{j=a}) \neq g_i(x^{j=b})$ for some $a \neq b \in \mathbb{F}_2$ and $x \in \mathbb{F}_2^n$. This means that an edge is functional if there is an instance where changing x_j changes the outcome of g_i.

The definition above implies that g_i depends on x_j and hence any expression of g_i must involve x_j. In particular, the following is true.

Proposition 5.1.4 *For a Boolean model with polynomial function $g = (g_1, \dots, g_n)$, an edge $j \to i$ is functional if and only if $g_i \notin \mathbb{F}_2[x_1, \dots, x_{j-1}, x_{j+1}, \dots, x_n]$ ($\mathbb{F}_2[S]$ denotes the ring of polynomials with coefficients in $\mathbb{F}_2$ and variables in S).*

A functional edge $j \to i$ is positive (or negative) if $g_i(x^{j=0}) \leq g_i(x^{j=1})$ for all $x \in \mathbb{F}_2^n$ (or $g_i(x^{j=0}) \geq g_i(x^{j=1})$ for all $x \in \mathbb{F}_2^n$).

5.1.4 Functionality of Circuits of Boolean Models

Feedback loops (circuits) are known to be responsible for many dynamical features (e.g., steady states and periodic behavior); therefore, the analysis of their functionality is of great interest. Here we use the polynomial dynamical systems framework to formulate and study the functionality context of circuits (defined in [79]). An edge $j \to i$ is called *functional at* x if $g_i(x^{j=0}) \neq g_i(x^{j=1})$.

Remark 5.1.5 *The polynomial $g_i(x)$ can be easily written as $g_i(x) = x_j h(x) + k(x)$; where h, k are polynomials that do not depend on x_j. The polynomial $h(x)$ is commonly called (in polynomial algebra) the partial derivative of g_i with respect to x_j at x, denoted by $\frac{\partial}{\partial x_j} g_i(x)$ (the square-free form of g_i is necessary for this definition to be consistent). With this remark, the definition above takes on the following algebraic form.*

Theorem 5.1.6 *An edge $j \to i$ is functional at x if and only if $\frac{\partial}{\partial x_j} g_i(x) = 1$.*

Definition 5.1.7 *The* functionality context *of an edge $j \to i$ is defined by*

$$S(j, i) = \{x : j \to i \text{ is functional at } x\}.$$

Using Theorem 5.1.6, we have the following equality.

Corollary 5.1.8 *The functionality context of an edge $j \to i$ is:*

$$S(j, i) = \{x : \frac{\partial}{\partial x_j} g_i(x) = 1\}.$$

Notice that $\{x : \frac{\partial}{\partial x_j} g_i(x) = 1\}$ can be computed algebraically by solving the equation $\frac{\partial}{\partial x_j} g_i(x) = 1$.

Definition 5.1.9 *Let $\mathcal{C} = \{(i_1, i_2), (i_2, i_3) \dots (i_k, i_1)\} \subseteq \mathcal{E}$ be a circuit. The functionality context of $\mathcal{C}$ is defined by*

$$S_{\mathcal{C}} = \bigcap_{(i,j) \in \mathcal{C}} S_{(i,j)}.$$

Corollary 5.1.10 *From Corollary 5.1.8 we have the following equality:*

$$S_{\mathcal{C}} = \{x : \frac{\partial}{\partial x_i} g_j(x) = 1, \text{ for all } (i,j) \in \mathcal{C}\}.$$

That is, the functionality context of a circuit can be computed algebraically by solving a system of polynomial equations.

5.1.5 Functionality of Circuits of Logical Models

Due to the way logical models are usually updated (continuity requirement), the functionality of circuits and even the sign of an edge (1,-1, or 0) is not as straightforward as for Boolean models. However, it can still be written in algebraic terms. Let $(\mathcal{V}, \mathcal{E}, \mathcal{K})$ be a logical model, and assume $g = (K_1 \circ \Theta^1, \ldots, K_n \circ \Theta^n)$ is already in polynomial form (see Section 3.1 and 3.2 in the main paper for a description of K_i and Θ^i); that is, $g : \mathbb{F}^n \to \mathbb{F}^n$ and $\mathcal{S} \subseteq \mathbb{F}^n$. We will use the definition in [90]. Notice that since the derivative notation in 5.1.4 comes from algebra and the following notation comes from logical models, they do not necessarily agree.

We define $\frac{\partial f_i}{\partial x_j^s}(x)$ for $s \in \{+,-\}$ (or $s \in \{+1,-1\}$) as follows: Let $x \in \mathbb{F}^n$ and $e_j = (0,\ldots,0,1,0,\ldots,0)$ (the j-th entry is 1).

If $x_j \leq p-2$ we define

$$\frac{\partial f_i}{\partial x_j^+}(x) = \begin{cases} 1, & \text{if } f_i(x) \leq x_i \text{ and } x_i + 1 \leq f_i(x+e_j), \\ -1, & \text{if } f_i(x+e_j) \leq x_i \text{ and } x_i + 1 \leq f_i(x), \\ 0, & \text{otherwise.} \end{cases}$$

If $x_j \geq 1$ we define

$$\frac{\partial f_i}{\partial x_j^-}(x) = \begin{cases} 1, & \text{if } f_i(x-e_j) \leq x_i - 1 \text{ and } x_i \leq f_i(x), \\ -1, & \text{if } f_i(x) \leq x_i - 1 \text{ and } x_i \leq f_i(x-e_j), \\ 0, & \text{otherwise.} \end{cases}$$

$\frac{\partial f_i}{\partial x_j^+}(x), \frac{\partial f_i}{\partial x_j^-}(x)$ are undefined otherwise.

Now consider the polynomial $Q : \mathbb{F}^2 \to \mathbb{F}$ given by
$Q(y,z) = \sum_{a \leq b}(1-(y-a)^{p-1})(1-(z-b)^{p-1})$. Notice that $Q(y,z) = 1$ if $y \leq z$ and $Q(y,z) = 0$ otherwise. We now use Q to write the definitions of $\frac{\partial f_i}{\partial x_j^+}(x)$ and $\frac{\partial f_i}{\partial x_j^-}(x)$ in an algebraic way.

$$\frac{\partial f_i}{\partial x_j^+}(x) = \begin{cases} 1, & \text{if } Q(f_i(x), x_i)Q(x_i+1, f_i(x+e_j)) = 1, \\ -1, & \text{if } Q(f_i(x+e_j), x_i)Q(x_i+1, f_i(x)) = 1, \\ 0, & \text{otherwise.} \end{cases}$$

Hence, $\frac{\partial f_i}{\partial x_j^+}(x) = Q(f_i(x), x_i)Q(x_i+1, f_i(x+e_j)) - Q(f_i(x+e_j), x_i)Q(x_i+1, f_i(x))$.

$$\frac{\partial f_i}{\partial x_j^-}(x) = \begin{cases} 1, & \text{if } Q(f_i(x-e_j), x_i-1)Q(x_i, f_i(x)) = 1, \\ -1, & \text{if } Q(f_i(x), x_i-1)Q(x_i, f_i(x-e_j)) = 1, \\ 0, & \text{otherwise.} \end{cases}$$

Hence, $\frac{\partial f_i}{\partial x_j^-}(x) = Q(f_i(x), x_i)Q(x_i+1, f_i(x+e_j)) - Q(f_i(x+e_j), x_i)Q(x_i+1, f_i(x))$.

Finally, for $\varepsilon \in \{-1,+1\}^n$ and $x \in \mathbb{F}^n$ we have $\partial f(x,\varepsilon) = (\frac{\partial f_i}{\partial x_j^{\varepsilon_j}}(x))$. Using the polynomial form we can write the concepts of functionality of circuits in an algebraic way as in Section 5.1.4. Notice that since $f_i(x)$ depends on the thresholds of the edges to i, so does Q; and therefore $\frac{\partial f_i}{\partial x_j^{\pm}}(x)$ depends on the threshold of $j \to i$. This implies that the functionality context of a circuit depends on the thresholds of its interactions.

It is important to mention that, depending on the definition of functionality used, one may have to modify the way the algebraic framework is used; for example, for Boolean models vs. logical models or for circuits with vs. without shortcuts.

5.2 Examples

5.2.1 Logical Model of the Core Lambda Switch

This example is based on the simplified logical model presented in [108] for the core Lambda switch. The logical model is defined by the network shown in Figure 5.1 and Table 5.1. The maximum levels of expressions are $m_1 = 1$ and $m_2 = 2$.

Table 5.1: Parameters for the Core Lambda Switch.

target	parameter
x_1=CI	$K_1(\emptyset) = 1, K_1(1,1) = 1, K_1(2,1) = 0, K_1((1,1),(2,1)) = 0$
x_2=Cro	$K_2(\emptyset) = 2, K_2(2,2) = 0, K_2(1,1) = 0, K_2((1,1),(2,2)) = 0$

If we choose the prime number $p = 3$, the functions defined over $\mathbb{F}_3^2$ are:

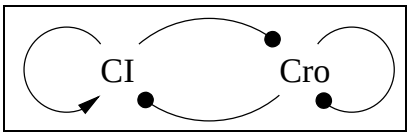


Figure 5.1: The network for the core Lambda switch. An arrow (circle) indicates activation (repression).

x_1 =CI, x_2 =Cro	$g_1(x)$	$g_2(x)$
00	1	1
01	0	2
02	0	1
10	1	0
11	0	0
12	0	1
20	1	0
21	0	0
22	0	1

Now we can find the polynomial form of g_1:
$g_1(x) = (1-x_1^2)(1-x_2^2) + (1-(x_1-1)^2)(1-x_2^2) + (1-(x_1-2)^2)(1-x_2^2)$, simplifying we have $g_1(x) = 1 + 2x_2^2$. Similarly, we obtain $g_2(x) = 1 + 2x_1^2 + 2x_2 + 2x_1^2x_2 + 2x_2^2$.

To find the steady states we solve the system of algebraic equations:

$$\begin{aligned} 1 + 2x_2^2 - x_1 &= 0, \\ 2 + x_1^2 + x_2 + 2x_1^2x_2 + 2x_2^2 + x_1^2x_2^2 - x_2 &= 0. \end{aligned}$$

We use *Macaulay2* to find the *Gröbner basis* of the *ideal* $I =< f_i(x) - x_i : i = 1, \ldots, n >$. First we define the ring of polynomials (`i1`), then we define the ideal generated by the polynomials that we want to be zero (`i2`), and finally we find the generators of the Gröbner basis of this ideal (`i3`). Since we chose the *lexicographic order*, the generators define an equivalent system that allows for backward substitution.
Macaulay2 code:

```
i1 : R=ZZ/3[x1,x2,MonomialOrder => Lex]/ideal (x1^3-x1, x2^3-x2);

i2 : I=ideal (1+2*x2^2-x1,2+x1^2+x2+2*x1^2*x2+2*x2^2+x1^2*x2^2-x2);

o2 : Ideal of R

i3 : gens gb I

o3 = | x2 x1-1 |
```

That is, the equivalent system that allows for backward substitution is:

$$\begin{aligned} x_1 - 1 &= 0, \\ x_2 &= 0. \end{aligned}$$

We can easily see that the only solution to this system is (1,0) as reported in [15].

5.2.2 Logical Model of the *Drosophila* Cell Cycle

This example is based on a simplified model for the *Drosophila* cell cycle first presented in [14]. The Boolean logical model is given in Figure 5.2 and Table 5.2.

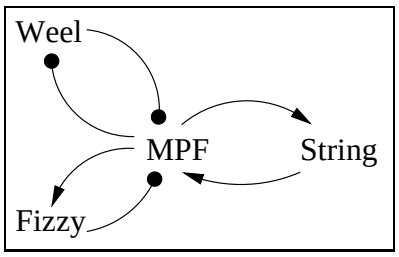


Figure 5.2: A network for the *Drosophila* cell cycle [14].

Table 5.2: Nonzero parameters for the *Drosophila* cell cycle model.

Target	**Parameter**
MPF	$K_M, K_{M,WS}, K_{M,S}, K_{M,W}$
Fizzy	$K_{F,M}$
Wee1	K_W
String	$K_{S,M}$

Let $x_1 = MPF$, $x_2 = Fizzy$, $x_3 = Wee1$, $x_4 = String$. By Remark 5.1.2, we can easily write the Boolean functions for x_1, x_2, x_3, x_4 in polynomial form (we only include non zero parameters in the sum):

$g_1(x) = (1-x_2)(1-x_3)(1-x_4) + x_3x_4(1-x_2) + x_4(1-x_2)(1-x_3) + x_3(1-x_2)(1-x_4)$, which simplifies to $g_1(x) = 1 + x_2$. Similar calculations for the other nodes yield:

$$\begin{aligned} g_1(x) &= 1 + x_2 \\ g_2(x) &= x_1 \\ g_3(x) &= 1 + x_1 \\ g_4(x) &= x_1 \end{aligned}$$

It follows that $Wee1 \to MPF$ and $String \to MPF$ are nonfunctional edges; that is, changes in $Wee1$ or $String$ do not affect MPF. Hence the network has only one functional (negative) circuit (see the Supporting Information of [115] for a formal definition of functional circuit), rather than the three assumed in [14]. In the transition from the complete model to the simplified model some functionality may be lost and the algebraic framework can easily detect this. This is important because in order to deduce dynamical properties from the network structure it is necessary to have the correct wiring diagram.

In Property 10 in [14] it is shown that the logical model has no steady states (this was done by proving that an equivalent Petri net is deadlock free). Algebraically, however, this corresponds to solving the system of equations:

$$\begin{aligned} 1 + x_2 - x_1 &= 0, \\ x_1 - x_2 &= 0, \\ 1 + x_1 - x_3 &= 0, \\ x_1 - x_4 &= 0. \end{aligned}$$

We now use *Macaulay2*:

```
i1 : R=ZZ/2[x1,x2,x3,x4,MonomialOrder => Lex]/ideal (x1^2-x1,
x2^2-x2,x3^2-x3,x4^2-x4);

i2 : I=ideal (1+x2-x1,x1-x2,1+x1-x3,x1-x4);

o2 : Ideal of R

i3 : gens gb I

o3 = | 1 |
```

That is, the equivalent system is:

$$1 = 0.$$

Therefore the system has no solution (we can also see this by observing that the first two equations imply that $x_2 = x_2 + 1$).

5.2.3 Logical Model for Th-Lymphocyte Differentiation

We now use the algebraic framework to analyze the logical model for the regulation of Th-lymphocyte differentiation presented in [89], where the authors used a Petri net framework to

study functionality of edges and circuits. We will make the same analysis using the algebraic framework.

The logical model is defined by the network shown in Figure 5.3 and Table 5.3; the table shows the parameters that are not zero. The polynomial functions are:

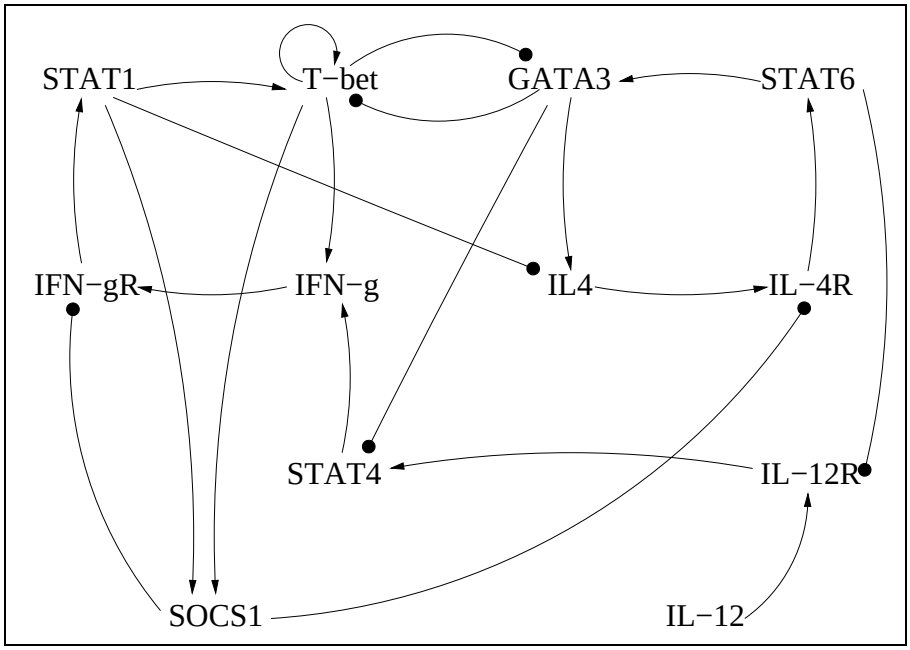


Figure 5.3: The regulatory network for Th-Lymphocyte differentiation.

$$
\begin{aligned}
g_1(x_9, x_{11}) &= x_9 + x_{11} + x_9x_{11},\\
g_2(x_7, x_{12}) &= (1 - x_7)x_{12},\\
g_3(\ ) &= 0,\\
g_4(x_1, x_{10}) &= x_1(1 - x_{10}),\\
g_5(x_2, x_{10}) &= x_2(1 - x_{10}),\\
g_6(x_3, x_8) &= x_3(1 - x_8),\\
g_7(x_4) &= x_4,\\
g_8(x_5) &= x_5,\\
g_9(x_6, x_{12}) &= x_6(1 - x_{12}),\\
g_{10}(x_7, x_{11}) &= x_7 + x_{11} + x_7x_{11},\\
g_{11}(x_7, x_{11}, x_{12}) &= (x_7 + x_{11} + x_7x_{11})(1 - x_{12}),\\
g_{12}(x_8, x_{11}) &= x_8(1 - x_{11}).
\end{aligned}
$$

Table 5.3: The nonzero parameters for the Th-Lymphocyte differentiation network.

target	parameter
x_1=INF-γ	$K_1(9)$,$K_1(11)$,$K_1(9,11)$
x_2=IL-4	$K_2(12)$
x_3=IL-12	
x_4=IFN-γ R	$K_4(1)$
x_5=IL-4R	$K_5(2)$
x_6=IL-12R	$K_6(3)$
x_7=STAT1	$K_7(4)$
x_8=STAT6	$K_8(5)$
x_9=STAT4	$K_9(6)$
x_{10}=SOCS1	$K_{10}(7)$,$K_{10}(11)$,$K_{10}(7,11)$
x_{11}=T-bet	$K_{11}(7)$,$K_{11}(11)$,$K_{11}(7,11)$
x_{12}=GATA-3	$K_{12}(8)$

Regulatory Circuits

Let us analyze the functionality of $\mathcal{C}$=[IL-4,IL-4R, STAT6, GATA-3]=[2,5,8,12].

- First we find $\frac{\partial}{\partial x_2}g_5 = 1 - x_{10}$; $\frac{\partial}{\partial x_5}g_8 = 1$; $\frac{\partial}{\partial x_8}g_{12} = 1 - x_{11}$; $\frac{\partial}{\partial x_{12}}g_2 = 1 - x_7$.
- Then, we solve the following system of equations

$$\begin{cases} 1 - x_{10} = 1, \\ 1 = 1, \\ 1 - x_{11} = 1, \\ 1 - x_7 = 1, \end{cases}$$

  which has the solution $x_7 = x_{10} = x_{11} = 0$.
- Finally, $\mathcal{S}_{\mathcal{C}} = \{x : x_7 = x_{10} = x_{11} = 0\}$.

Hence, the functionality context of the circuit is the set of states for which STAT1, SOCS1 and T-bet are inactive as reported in [89].

Steady States

In order to find the steady states, we have to solve the following system of equations:

$$\begin{aligned}
x_9 + x_{11} + x_9x_{11} &= x_1,\\
(1-x_7)x_{12} &= x_2,\\
0 &= x_3,\\
x_1(1-x_{10}) &= x_4,\\
x_2(1-x_{10}) &= x_5,\\
x_3(1-x_8) &= x_6,\\
x_4 &= x_7,\\
x_5 &= x_8,\\
x_6(1-x_{12}) &= x_9,\\
x_7 + x_{11} + x_7x_{11} &= x_{10},\\
(x_7 + x_{11} + x_7x_{11})(1-x_{12}) &= x_{11},\\
x_8(1-x_{11}) &= x_{12}.
\end{aligned}$$

We now use *Macaulay2*:

```
i1 : R=ZZ/2[x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,MonomialOrder =>
Lex]/ideal (x1^2-x1, x2^2-x2, x3^2-x3, x4^2-x4, x5^2-x5, x6^2-x6,
x7^2-x7, x8^2-x8, x9^2-x9, x10^2-x10, x11^2-x11, x12^2-x12);

i2 : I=ideal (x9+x11+x9*x11-x1, (1-x7)*x12-x2, 0-x3, x1*(1-x10)-x4,
x2*(1-x10)-x5, x3*(1-x8)-x6, x4-x7, x5-x8, x6*(1-x12)-x9,
x7+x11+x7*x11-x10, (x7+x11+x7*x11)*(1-x12)-x11, x8*(1-x11)-x12);

o2 : Ideal of R

i3 : gens gb I

o3 = | x11x12 x10+x11 x9 x8+x12 x7 x6 x5+x12 x4 x3 x2+x12 x1+x11 |
```

That is, the equivalent system that allows for backward substitution is:

$$\begin{aligned} x_1 + x_{11} &= 0, \\ x_2 + x_{12} &= 0, \\ x_3 &= 0, \\ x_4 &= 0, \\ x_5 + x_{12} &= 0, \\ x_6 &= 0, \\ x_7 &= 0, \\ x_8 + x_{12} &= 0, \\ x_9 &= 0, \\ x_{10} + x_{11} &= 0, \\ x_{11}x_{12} &= 0. \end{aligned}$$

We can easily see that the solutions of this system (as reported in [89]) are:

$$\begin{aligned} S_1 &= (000000000000), \\ S_2 &= (010010010001), \\ S_3 &= (100000000110). \end{aligned}$$

5.3 Algorithm for K-Bounded Petri Nets

5.3.1 Definition of Petri Nets

A *k-bounded Petri net* is a 4-tuple (S, T, F, W), where:

1. $S = \{s_1, \ldots, s_n\}$ is the set of places.
2. $T = \{t_1, \ldots, t_m\}$ is the set of transitions.
3. $F \subseteq (S \times T) \cup (T \times S)$ is the set of "regular" arcs. $F^- \subseteq (S \times T)$ is the set of inhibitory arcs.
4. $W : F \to \mathbb{N}$ gives the arc weights of the regular arcs. $W^- : F^- \to \mathbb{N}$ gives the arc weights of the inhibitory arcs.

For a transition t, define:

$\bullet t = \{s \in S : (s,t) \in F\}$, $t\bullet = \{s \in S : (t,s) \in F\}$, $\overline{t} = \{s \in S : (s,t) \in F^-\}$.

The incidence matrix $A = (a_{ts})_{t\in T, s\in S}$ (or $A = (a_{ij})_{i,j=1}^{m,n}$) is an $m \times n$ matrix given by $a_{st} = W(t,s) - W(s,t)$ (we consider $W(e) = 0$ when $e \notin F$). We denote by A_t the column of A^T (or the row of A) corresponding to a transition $t \in T$. Notice that $A_t = A^T U_t$, where the entry of U_t corresponding to t is 1 and all others are 0.

An assignment $x = (x_s)_{s\in S} \in \mathbb{N}^n$ (or $x = (x_1, \dots, x_n)$) for all places is called a *marking*. Let x be a marking; a transition t is said to be *enabled* if $W(s,t) \leq x_s$ for all $s \in {}^\bullet t$ and $x_s \leq W^-(s,t) - 1$ for all $s \in \overline{t}$. Consider the function $C_t(x)$, defined by

$$C_t(x) = \begin{cases} 1, \text{if } W(s,t) \leq x_s \text{ for } s \in {}^\bullet t \text{ and } x_s < W^-(s,t) \text{ for } s \in \overline{t}, \\ 0, \text{otherwise.} \end{cases}$$

Thus, a transition t is enabled for x if $C_t(x) = 1$.

The evolution of the Petri net is given by *firing* transitions; by firing an enabled transition t, we update the value of the places with regular arcs from/to t. The phase space of the Petri net is obtained by firing transitions in an asynchronous manner. Let x be a marking: If a transition t is enabled, then firing t results in the marking $x + A_t$. This implies that for any marking x and any transition t, the Petri net can always evolve from x to $f_t(x) = x + C_t(x)A_t = x + C_t A^T U_t$ (If $C_t(x) = 0$, then $f_t(x) = x$; but if $C_t(x) = 1$, then $f_t(x) = x + A_t = x + A^T U_t$). In the case where $x \neq y = f_t(x)$ we write $x \xrightarrow{t} y$. Denote by $f_{\mathcal{T}}$ the function obtained by composing $f_{t_1} \circ \cdots \circ f_{t_l}$, where $t_1, \dots, t_l$ are the elements of $\mathcal{T}$. A trajectory is a path $x \xrightarrow{t_1} y \xrightarrow{t_2} \cdots$ and denoted by $x \xrightarrow{\mathcal{T}}$. For any trajectory $x \xrightarrow{\mathcal{T}} y$ we have $y = x + A^T \sum_{t\in\mathcal{T}} U_t$. The reachability graph for a marking x is the graph made up of all trajectories starting at x. The reachability graph of a set of markings S is the graph made up of all reachability graphs of all the elements of S.

We say that $(X_1, \dots, X_m) > 0$ if $X_i \geq 0$ and at least for one i we have $X_i > 0$. A P-invariant is an integer solution, $X > 0$, of $AX = 0$. A T-invariant is an integer solution, $Y > 0$, of $A^T Y = 0$. A marking x is dead if no transition is enabled, that is, if $C_t(x) = 0$ for all $t \in T$; this is equivalent to $f_t(x) = x$ for all $t \in T$.

A marking is *k-bounded* if the value of each place is at most k. The Petri net is *k-bounded* if any reachable marking (obtained by firing some sequence of transitions) from any k-bounded marking is also a k-bounded marking, that is, if $\{x : x_i \leq k\}$ contains the vertices of its reachability graph, which is equivalent to $f_t([0,k]^n) \subseteq [0,k]^n$ for all transitions t. Examples of 1-bounded Petri nets are Boolean regulatory Petri nets, presented in [14; 89; 105; 106]. When each place has its own "k", we have a more general definition: for $K = (k_1, \dots, k_n)$, a Petri net is K-bounded if $f_t([0,k_1] \times \cdots \times [0,k_n]) \subseteq [0,k_1] \times \cdots \times [0,k_n]$ for all transitions t. Examples of such Petri nets are multi-level regulatory Petri nets, presented in [15; 18]. Our framework is applicable to K-bounded Petri nets (including k-bounded Petri nets).

For a given Petri net, the analysis of its dynamics, checking for dead markings and the type of liveness are some of the typical questions that can be viewed as algebraically as we will see next. Also, when modeling biological systems, there is not a unique marking that is

of interest, but a whole family of markings corresponding to different initial states of the biological system. By looking at a Petri net as a polynomial dynamical system we can study all those markings at the same time.

5.3.2 Polynomial Form of K-Bounded Petri nets

Since the algebraic framework relies on finite fields, we consider K-bounded Petri nets and focus on the analysis of markings in $\mathcal{S} = [0, k_1] \times \cdots \times [0, k_n]$. Let p be a prime number such that $k_i + 1 \leq p$ for all i and let $\mathbb{F} = \mathbb{F}_p$. Then, for all $t \in T$, $f_t : \mathcal{S} \subseteq \mathbb{F}^n \to \mathbb{F}^n$.

We need to extend the functions f_t to all of $\mathbb{F}^n$ and determine their polynomial form. It suffices to give algebraic structure to the function $C_t : \mathcal{S} \to [0, 1]$ (since $f_t(x) = x + C_t(x)A_t$, the polynomial form of C_t will automatically give the polynomial form of f_t). In order to do this, consider the function $c_{[a,b]}(z) = 1$ if $a \leq z \leq b$ and $c_{[a,b]}(z) = 0$ otherwise (that is, $c_{[a,b]}$ is the characteristic function of $[a, b]$). Since $C_t(x) = \prod_{s \in {}^\bullet t} c_{[W(s,t),k_s]}(x_s) \prod_{s \in \overline{t}} c_{[0,W(s,t)-1]}(x_s)$, we only need to give algebraic structure to $c_{[a,b]}$ (the polynomial form of C_t will be given by the product of the polynomial forms of its factors). It is not difficult to see that the polynomial function $\sum_{a \leq r \leq b}(1 - (z - r)^{p-1})$ is equal to 1 if $a \leq z \leq b$ and 0 otherwise. Hence, the polynomial form of $c_{[a,b]}$ is $c_{[a,b]}(z) = \sum_{a \leq r \leq b}(1 - (z - r)^{p-1})$. This gives the polynomial form of $C_t(x)$, which, in turn, gives the polynomial form of $f_t(x) = x + C_t(x)A_t$, where $f_t : \mathbb{F}^n \to \mathbb{F}^n$.

Example 5.3.1 *Consider a transition t of a 1-bounded Petri net such that ${}^\bullet t = \{x_1, x_2\}$ and $\overline{t} = \{x_3, x_4\}$ (with weights equal to 1). Then $C_t(x) = c_{[1,1]}(x_1)c_{[1,1]}(x_2)c_{[0,0]}(x_3)c_{[0,0]}(x_4)$. Since $c_{[0,0]}(z) = 1 - z$ and $c_{[1,1]}(z) = z$, it follows that $C_t(x) = x_1x_2(1 - x_3)(1 - x_4)$. Hence the polynomial form of f_t is given by $f_t(x) = x + x_1x_2(1 - x_3)(1 - x_4)A_t$.*

Now we can state and solve algebraically some Petri net problems. For example, the problem of finding dead markings becomes the problem of solving polynomial equations, which can be easily addressed within the algebraic framework, as was the case for logical models. More precisely, we have the following remark.

Remark 5.3.2 *A marking x of a K-bounded Petri net is dead if and only if $x \in \mathcal{S}$ and $f_t(x) = x$ for all $t \in T$. Let $\varphi(x) = \prod_{i=1}^n c_{[0,k_i]}(x_i)$. Note that $\varphi(x) = 1$ if $x \in \mathcal{S}$ and $\varphi(x) = 0$ otherwise. Then, the set of dead markings is given by the solutions of the system of polynomial equations:*

$$f_{t_1}(x) = x \ , \quad \ldots \ , \quad f_{t_m}(x) = x \ , \quad \varphi(x) = 1.$$

Remark 5.3.3 *If the polynomial form of f_t for all transitions t is known, then we can easily recover A using the equation $C_t(x)A_t = f_t(x) - x$ and hence find the P- and T-invariants.*

5.4 Examples

5.4.1 Petri net for the *Drosophila* Cell Cycle

Consider the Petri net given in [14] for the *Drosophila* cell cycle (corresponding to the logical model in Section 5.2.2): Places: $\{s_1 = M, s_2 = F, s_3 = W, s_4 = S\}$.
Transitions: $\{t_1 = t_M, t_2 = t_{M,FWS}, t_3 = t_{M,F}, t_4 = t_{M,WS}, t_5 = t_{M,FW}, t_6 = t_{M,S}, t_7 = t_{M,FS}, t_8 = t_{M,W}, t_9 = t_F, t_{10} = t_{F,M}, t_{11} = t_W, t_{12} = t_{W,M}, t_{13} = t_S, t_{14} = t_{S,W}\}$.
$F = \{ (S, t_S), (F, t_F), (t_M, M), (t_{F,M}, F), (M, t_{F,M}), (t_{F,M}, M), (M, t_{S,M}), (t_{S,M}, M), (t_{S,M}, S), (W, t_{W,M}), (M, t_{W,M}), (t_{W,M}, M), (t_W, W), (F, t_{M,F}), (t_{M,F}, F), (M, t_{M,F}), (W, t_{M,W}), (t_{M,W}, W), (t_{M,W}, M), (t_{M,S}, M), (S, t_{M,S}), (t_{M,S}, S), (W, t_{M,WS}), (t_{M,WS}, W), (t_{M,WS}, M), (S, t_{M,WS}), (t_{M,WS}, S), (F, t_{M,FS}), (t_{M,FS}, F), (M, t_{M,FS}), (S, t_{M,FS}), (t_{M,FS}, S), (W, t_{M,FW}), (t_{M,FW}, W), (F, t_{M,FW}), (t_{M,FW}, F), (M, t_{M,FW}), (W, t_{M,FWS}), (t_{M,FWS}, W), (F, t_{M,FWS}), (t_{M,FWS}, F), (M, t_{M,FWS}), (S, t_{M,FWS}), (t_{M,FWS}, S)\}$ with $W(e) = 1$ for all $e \in F$.
The inhibitory arcs are $F^- = \{(W, t_{M,FS}), (W, t_W), (W, t_M), (W, t_{M,F}), (W, t_{M,S}), (F, t_M), (F, t_{M,WS}), (F, t_{M,W}), (F, t_{M,S}), (F, t_{F,M}), (S, t_{M,W}), (S, t_M), (S, t_{M,F}), (S, t_{M,FW}), (S, t_{S,M}), (M, t_M), (M, t_S), (M, t_F), (M, t_W), (M, t_{M,S}), (M, t_{M,W}), (M, t_{M,WS})\}$ with $W^-(e) = 1$ for all $e \in F^-$. It follows that A^T is given by

$$\begin{pmatrix} & t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & t_8 & t_9 & t_{10} & t_{11} & t_{12} & t_{13} & t_{14} \\ M & 1 & -1 & -1 & 1 & -1 & 1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ F & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ W & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ S & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}.$$

This Petri net is 1-bounded (shown in [14]) and therefore we can use the algebraic framework. Let us compute the polynomial form of f_{t_M}. First, notice that $c_{[0,0]}(z) = 1 - z$, $c_{[1,1]}(z) = z$. Then, $C_{t_M}(x) = c_{[0,0]}(x_1)c_{[0,0]}(x_2)c_{[0,0]}(x_3)c_{[0,0]}(x_4)$ and, therefore, $C_{t_M}(x) = (1 - x_1)(1 - x_2)(1 - x_3)(1 - x_4)$. Since $A_{t_M} = (1, 0, 0, 0)$, it follows that $f_{t_M}(x) = x + (1 - x_1)(1 - x_2)(1 - x_3)(1 - x_4)(1, 0, 0, 0)$; then $f_{t_M} = (x_1 + (1 - x_1)(1 - x_2)(1 - x_3)(1 - x_4), x_2, x_3, x_4)$. Similarly:

$$
\begin{aligned}
f_{t_M} &= (x_1 + (1-x_1)(1-x_2)(1-x_3)(1-x_4), x_2, x_3, x_4),\\
f_{t_{M,FWS}} &= (x_1 + x_1x_2x_3x_4, x_2, x_3, x_4),\\
f_{t_{M,F}} &= (x_1 + x_1x_2(1-x_3)(1-x_4), x_2, x_3, x_4),\\
f_{t_{M,WS}} &= (x_1 + (1-x_1)(1-x_2)x_3x_4, x_2, x_3, x_4),\\
f_{t_{M,FW}} &= (x_1 + x_1x_2x_3(1-x_4), x_2, x_3, x_4),\\
f_{t_{M,S}} &= (x_1 + (1-x_1)(1-x_2)(1-x_3)x_4, x_2, x_3, x_4),\\
f_{t_{M,FS}} &= (x_1 + x_1x_2(1-x_3)x_4, x_2, x_3, x_4),\\
f_{t_{M,W}} &= (x_1 + (1-x_1)(1-x_2)x_3(1-x_4), x_2, x_3, x_4),\\
f_{t_F} &= (x_1, x_2 + (1-x_1)x_2, x_3, x_4),\\
f_{t_{F,M}} &= (x_1, x_2 + x_1(1-x_2), x_3, x_4),\\
f_{t_W} &= (x_1, x_2, x_3 + (1-x_1)(1-x_3), x_4),\\
f_{t_{W,M}} &= (x_1, x_2, x_3 + x_1x_3, x_4),\\
f_{t_S} &= (x_1, x_2, x_3, x_4 + (1-x_1)x_4),\\
f_{t_{S,W}} &= (x_1, x_2, x_3, x_4 + x_1(1-x_4)).
\end{aligned}
$$

Now we can state some Petri net problems algebraically. For example, in order to find the dead markings, we have to solve the following system of equations ($\varphi(x) = 1$ for all x):

$$
\begin{aligned}
((1-x_1)(1-x_2)(1-x_3)(1-x_4), 0, 0, 0) &= (0,0,0,0),\\
(x_1x_2(1-x_3)(1-x_4), 0, 0, 0) &= (0,0,0,0),\\
((1-x_1)(1-x_2)x_3x_4, 0, 0, 0) &= (0,0,0,0),\\
(x_1x_2x_3(1-x_4), 0, 0, 0) &= (0,0,0,0),\\
((1-x_1)(1-x_2)(1-x_3)x_4, 0, 0, 0) &= (0,0,0,0),\\
(x_1x_2(1-x_3)x_4, 0, 0, 0) &= (0,0,0,0),\\
((1-x_1)(1-x_2)x_3(1-x_4), 0, 0, 0) &= (0,0,0,0),\\
(0, (1-x_1)x_2, 0, 0) &= (0,0,0,0),\\
(0, x_1(1-x_2), 0, 0) &= (0,0,0,0),\\
(0, 0, (1-x_1)(1-x_3), 0) &= (0,0,0,0),\\
(0, 0, x_1x_3, 0) &= (0,0,0,0),\\
(0, 0, 0, (1-x_1)x_4) &= (0,0,0,0),\\
(0, 0, 0, x_1(1-x_4)) &= (0,0,0,0);
\end{aligned}
$$

or the system:

$$\begin{aligned}
(1-x_1)(1-x_2)(1-x_3)(1-x_4) &= 0,\\
(x_1x_2(1-x_3)(1-x_4) &= 0,\\
(1-x_1)(1-x_2)x_3x_4 &= 0,\\
x_1x_2x_3(1-x_4) &= 0,\\
((1-x_1)(1-x_2)(1-x_3)x_4 &= 0,\\
x_1x_2(1-x_3)x_4 &= 0,\\
(1-x_1)(1-x_2)x_3(1-x_4) &= 0,\\
(1-x_1)x_2 &= 0,\\
x_1(1-x_2) &= 0,\\
(1-x_1)(1-x_3) &= 0,\\
x_1x_3 &= 0,\\
(1-x_1)x_4 &= 0,\\
x_1(1-x_4) &= 0.
\end{aligned}$$

We now use *Macaulay2*:

```
i1 : R=ZZ/2[x1,x2,x3,x4,MonomialOrder => Lex]/ideal (x1^2-x1,
x2^2-x2,x3^2-x3,x4^2-x4);

i2 : I=ideal ((1-x1)*(1-x2)*(1-x3)*(1-x4),x1*x2*(1-x3)*(1-x4),
(1-x1)*(1-x2)*x3*x4,x1*x2*x3*(1-x4),(1-x1)*(1-x2)*(1-x3)*x4,
x1*x2*(1-x3)*x4,(1-x1)*(1-x2)*x3*(1-x4),(1-x1)*x2,x1*(1-x2),
(1-x1)*(1-x3),x1*x3,(1-x1)*x4,x1*(1-x4));

o2 : Ideal of R

i3 : gens gb I

o3 = | 1 |
```

Since the Gröbner basis of the ideal is $\{1\}$, the system of equations has no solutions; hence this Petri net does not have any dead markings as reported in [14].

5.4.2 Petri net for the Core Lambda Switch

Consider the Petri net model in [15] for the core lambda switch (corresponding to the logical model in Section 5.2.1):

$S = \{CI, Cro\}$.
$T = \{t_1 = t^+_{CI}, t_2 = t^-_{CI,\{(CI,1),(Cro,1)\}}, t_3 = t^+_{Cro}, t_4 = t^-_{Cro,\{(Cro,2)\}}, t_5 = t^-_{Cro,\{(CI,1)\}}, t_6 = t^-_{Cro,\{(CI,1),(Cro,2)\}}\}$.
$F = \{(t_1, CI), (Cro, t_1), (CI, t_2), (Cro, t_2), (t_2, Cro), (t_3, Cro), (Cro, t_4), (t_4, Cro), (CI, t_5), (Cro, t_5), (CI, t_6), (Cro, t_6), (t_6, Cro)\}$.
$F^- = \{(CI, t_1), (CI, t_3), (Cro, t_3), (CI, t_4), (Cro, t_5)\}$.

$W(e) = 2$ for $e \in \{(Cro, t_4), (Cro, t_6)\}$ and $W(e) = 1$ for the other arcs in F. Also, $W^-(e) = 2$ for $e \in \{(Cro, t_3), (Cro, t_5)\}$ and $W(e) = 1$ for the other arcs in F^-.

It follows that A^T is given by

$$\begin{pmatrix} & t_1 & t_2 & t_3 & t_4 & t_5 & t_6 \\ CI & 1 & -1 & 0 & 0 & 0 & 0 \\ Cro & 0 & 0 & 1 & -1 & -1 & -1 \end{pmatrix}.$$

This Petri net is (1,2)-bounded (see [15]) and therefore we can use the algebraic framework. Notice that $\varphi(x) = c_{[0,1]}(x_1)c_{[0,2]}(x_2) = (x_1 + 1)^2$. Let us compute the polynomial form of f_{t_5}. First, notice that $c_{[0,0]}(z) = (1 - z)(1 + z)$, $c_{[1,1]}(z) = z(2 - z)$, $c_{[2,2]}(z) = z(1 - z)$, $c_{[1,2]}(z) = z^2$ and $c_{[0,1]}(z) = (1 + z)^2$. Then, $C_{t_5}(x) = c_{[1,1]}(x_1)c_{[1,2]}(x_2)c_{[0,1]}(x_2)$ which reduces to $C_{t_5}(x) = x_1(2 - x_1)x_2(2 - x_2)$. Since $A_{t_5} = (0, -1)$, it follows that $f_{t_5}(x) = x + x_1(2 - x_1)x_2(2 - x_2)(0, -1) = (x_1, x_2 - x_1(2 - x_1)x_2(2 - x_2))$. The polynomial forms for all the transitions are:

$$\begin{aligned} f_1(x) &= (x_1 + (1 - x_1)(1 + x_1)(1 - x_2)(1 + x_2), x_2), \\ f_2(x) &= (x_1 - x_1(2 - x_1)x_2^2, x_2), \\ f_3(x) &= (x_1, x_2 + (1 - x_1)(1 + x_1)(1 + x_2)^2), \\ f_4(x) &= (x_1, x_2 - (1 - x_1)(1 + x_1)x_2(1 - x_2)), \\ f_5(x) &= (x_1, x_2 - x_1(2 - x_1)x_2(2 - x_2)), \\ f_6(x) &= (x_1, x_2 - x_1(2 - x_1)x_2(1 - x_2)). \end{aligned}$$

In order to find the dead markings, we have to solve the system:

$$\begin{aligned} (1 - x_1)(1 + x_1)(1 - x_2)(1 + x_2) &= 0, \\ x_1(2 - x_1)x_2^2 &= 0, \\ (1 - x_1)(1 + x_1)(1 + x_2)^2 &= 0, \\ (1 - x_1)(1 + x_1)x_2(1 - x_2) &= 0, \\ x_1(2 - x_1)x_2(2 - x_2) &= 0, \\ x_1(2 - x_1)x_2(1 - x_2) &= 0, \\ (x_1 + 1)^2 - 1 &= 0. \end{aligned}$$

The last equation corresponds to the requirement that $x \in \mathcal{S}$. Using *Macaulay2*, we obtain the marking $(1, 0)$ as the only dead marking as reported in [15].

5.5 Algebra and Parameters

An important feature of the algebraic framework is the ability to treat the parameters as variables and study all corresponding models at the same time, which we illustrate with a small logical model. (The same analysis can be done for Petri nets.)

Suppose, for example, that in the model for the *Drosophila* cell cycle (Section 5.2) some parameters are unknown:

Table 5.4: Nonzero and unknown parameters for the *Drosophila* cell cycle.

Target	**Parameter**
MPF	$K_M, K_{M,S}$
Fizzy	$K_{F,M}$
Wee1	K_W
String	$K_{S,M}$
unknown parameters	$K_1 = K_{M,W}, K_2 = K_{M,WS}$ $K_3 = K_{M,FS}$

We find the polynomial form of f_1. We only include unknown parameters and nonzero parameters:
$f_1 = (1-x_2)(1-x_3)(1-x_4) + (1-x_2)(1-x_3)x_4 + K_1(1-x_2)x_3(1-x_4) +$
$K_2(1-x_2)x_3x_4 + K_3x_2(1-x_3)x_4$. The other polynomials can be easily found. Then, the polynomial form of the logical model is:

$$\begin{aligned}
f_1 &= 1 + x_2 + x_3 + K_1x_3 + x_2x_3 + K_1x_2x_3 + K_3x_2x_4 + K_1x_3x_4 + K_2x_3x_4 + \\
&\quad K_1x_2x_3x_4 + K_2x_2x_3x_4 + K_3x_2x_3x_4, \\
f_2 &= x_1, \\
f_3 &= 1 + x_1, \\
f_4 &= x_1.
\end{aligned}$$

Now let us focus on finding steady states of this logical model. This model has 3 unknown parameters, and if we were to analyze each model, we would have to analyze $2^3 = 8$ logical models, as the complexity grows exponentially with respect to the number of unknown parameters. However, using the algebraic framework we can treat the parameters K_1, K_2, K_3 as variables and analyze the 8 models at the same time.

The system of equations that gives the steady states is

$$\begin{aligned} x_1 &= 1 + x_2 + x_3 + K_1x_3 + x_2x_3 + K_1x_2x_3 + K_3x_2x_4 + K_1x_3x_4 \\ &\quad + K_2x_3x_4 + K_1x_2x_3x_4 + K_2x_2x_3x_4 + K_3x_2x_3x_4, \\ x_2 &= x_1, \\ x_3 &= 1 + x_1, \\ x_4 &= x_1. \end{aligned}$$

We now use *Macaulay2*:

```
i1 : R=ZZ/2[x1,x2,x3,x4,K1,K2,K3,MonomialOrder => Lex]/ideal
(x1^2-x1, x2^2-x2,x3^2-x3,K1^2-K1,K2^2-K2,K3^2-K3);

i2 : I=ideal {1+x2+x3+K1*x3+x2*x3+K1*x2*x3+ K3*x2*x4+ K1*x3*x4+
K2*x3*x4+ K1*x2*x3*x4+ K2*x2*x3*x4+ K3*x2*x3*x4- x1, x1-x2, 1+x1-x3,
x1-x4};

o2 : Ideal of R

i3 : gens gb I

o3 = | K1K3+K1 x4K3+x4 x4K1+K1 x3+x4+1 x2+x4 x1+x4 |
```

Then, the equivalent system that allows backward substitution is

$$\begin{aligned} x_1 + x_4 &= 0, \\ x_2 + x_4 &= 0, \\ x_3 + x_4 - 1 &= 0, \\ K_1(x_4 - 1) &= 0, \\ x_4(K_3 - 1) &= 0, \\ K_1(K_3 - 1) &= 0. \end{aligned}$$

From this system we can obtain valuable information. For example, if x is a steady state, then $x_1 = x_2 = x_4 = x_3 - 1$; that is, the only possible steady states are $(0, 0, 1, 0)$ and $(1, 1, 0, 1)$. This is true no matter what the parameters K_1, K_2, and K_3 are.

We can also easily solve this system and see the steady states for any choice of parameters. For example, consider $K_1 = 1, K_3 = 0$. Since they do not satisfy the first equation, there are no steady states. Consider $K_1 = K_2 = K_3 = 0$; we have the system
$x_4 = 0, x_3 = 1 - x_4, x_2 = x_4, x_1 = x_4$,
which has the unique steady state $x = (0, 0, 1, 0)$.

We can also use the system above to determine for which choice of parameters a given state is a steady state of the system. For example, suppose we are interested in finding the parameters for which $(0,0,1,0)$ is a steady state, then we have the system $K_1(K_3-1)=0, 0(K_3-1)=0, K_1=0$; therefore $K_1=0$ and the other parameters are "free." Hence, $(0,0,1,0)$ is a steady state for $K=(0,K_2,K_3)$.

Suppose now we are interested in finding the parameters for which $(1,1,0,1)$ is a steady state. In this case, we have the system $K_1(K_3-1)=0, K_3-1=0, K_1=0$; that is, $K_1=0, K_3=1$ and K_2 is free. Hence, $(1,1,0,1)$ is a steady state for $K=(0,K_2,1)$ (note that in this case $(0,0,1,0)$ is a steady state as well).

5.6 Performance

We tested our algorithm to compute steady states of several published models, that is, we recorded the required time for computing a lexicographic Gröbner basis (which then allows backward substitution to solve the system of equations $f_1(x)-x_1=0, \ldots, f_n(x)-x_n=0$.) The results are shown in Table 5.5. Furthermore, to compute limit cycles (using a parallel update) of length r, one can compute the lexicographic Gröbner basis for the system $f_1^t(x)-x_1=0, \ldots, f_n^t(x)-x_n=0$, where $t=1,...,r$. It is clear that the solution set for $t=1$ is the set of steady states; the states in the solution set for $t=2$, which are not steady states, are periodic states with period two and so on.

Table 5.5: Timing for the Gröbner basis computations: **N**=number of nodes, **Tr**=timings (in seconds) for $f_1^r(x)-x_1=\cdots=f_n^r(x)-x_n$=0, where $r=1,\ldots,5$. The networks used were: Fission yeast [22], Budding yeast [68], Th cell differentiation [89], Th cell differentiation [74], Th cell differentiation [76], T-cell receptor [65], respectively. All networks were Boolean except the fourth one which had some nodes with 3 states.

N	**T1**	**T2**	**T3**	**T4**	**T5**
10	.002	.001	.001	.001	.001
12	.003	.041	.526	1.539	.342
12	.001	.003	.002	.004	.009
17	.004	.006	.007	.155	4.95
23	.002	.008	.008	.179	30.276
40	.011	.023	.029	.023	.047

Notice that in Table 5.5 the size of the network does not seem to be correlated to the computation time. This is due to the fact that polynomial algebra computations depend more strongly on the complexity of the polynomial equations and not as much on the size or complexity of the logical model. This means that the method can scale well to large networks and that it can complement other methods very well. (Typically, the more terms

the polynomial form of the model has the more complex the Gröbner basis calculation. In the Boolean case, for instance, a conjunction leads to a polynomial with one term, whereas a disjunction results in 3 terms.)

5.7 Compatibility of the Algebraic Framework with Other Frameworks

We have shown that logical models and Petri nets can be seen as polynomial functions. On the other hand, Petri nets have been used to represent logical models [14]; their method consists in associating a place in a Petri net to each node in the logical model and to associate transitions and arcs in the Petri net to interactions between nodes (to avoid inhibitory arcs, places can be duplicated so that to any inhibitory arc from x to t, there is a regular arc from $\overline{x}$ to t). Then, one can ask if there is any relation between the polynomial form of a logical model and the polynomial form of a Petri net that came from the same logical model. We now show that these polynomial forms are equivalent. This will show that not only does our mathematical framework have rich mathematical structure, but it is also compatible with existing relationships between model types.

Theorem 5.7.1 *Let $(\mathcal{V},\mathcal{E},\mathcal{K})$ be a logical model and $\mathcal{S}=[0,m_1]\times\cdots\times[0,m_n]$. Then, the state space of the polynomial model (restricted to $\mathcal{S}$) associated to the logical model is equal to the state space of the polynomial model (restricted to $\mathcal{S}$) associated to the Petri net that is associated to the logical model.*

Proof. We will restrict the polynomial systems to $\mathcal{S}$. First, note that, by construction, the state space of the logical model is equal to the state space of the associated polynomial model. On the other hand, again by construction, the Petri net and the associated polynomial model have the same state space. Now, since the Petri net and the logical model have the same state space, it follows that the associated polynomials have the same state space. □

Theorem 5.7.2 *Let $(\mathcal{V},\mathcal{E},\mathcal{K})$ be a logical model. Denote by g_i the polynomial function corresponding to node i associated to this logical model. Denote by h_t the polynomial function corresponding to transition t associated to the Petri net that is associated to the logical model. Then, when we restrict the functions to $\mathcal{S}$, we have the following functional equalities:*

$$g_i = x_i + \sum_{t\in\bullet i}((h_t)_i - x_i),$$

where $(h_t)_i$ denotes the polynomial component of h_t corresponding to x_i and $\bullet i=\{s:$ there is an arc from s to $i\}$; and

$$(h_t)_j = x_j$$

for $j\neq i$ and $t\in\bullet i$.

Proof. Since we restrict the functions to $\mathcal{S}$, we can use the fact that they correspond to updating logical models and Petri nets. Notice that node i in the logical model corresponds to place i in the Petri net (they have the same value); and using a parameter to update a state in the logical model corresponds to updating a marking using the corresponding transition in the Petri net. First, if we are at a steady state/dead marking, no parameter/transition would change the current state/marking and the first equality follows in this case (each term in the sum is zero). On the other hand, consider a state/marking that is not a steady state/dead marking. Since only one parameter can be used to update node i, it follows that only one transition in $\bullet i$ is enabled. Then, it follows that $\sum_{t\in\bullet i}((h_t)_i - x_i)$ is the change that occurs at node i by updating the transition that is enabled in $\bullet i$. But this is by definition the change that occurs by updating node i in the logical model. Since this last value is $g_i - x_i$, it follows that $\sum_{t\in\bullet i}((h_t)_i - x_i) = g_i - x_i$. Then, the first equality follows in this case.

The second part of the theorem follows from the fact that each transition t can only change the place i for which there is an arc from t to i. This means that other places, $j \neq i$, remain unaffected by this transition. □

5.7.1 Example

As an example we will again use the *Drosophila* cell cycle model presented in [14].

We have already seen that the polynomial model associated to this logical model is given by:

$$\begin{aligned} g_1 &= 1 + x_2, \\ g_2 &= x_1, \\ g_3 &= 1 + x_1, \\ g_4 &= x_1. \end{aligned}$$

On the other hand, the Petri net associated with this logical model was presented in Section 5.4.1. It is given by the polynomials:

$$
\begin{aligned}
h_{t_1}(x_1,x_2,x_3,x_4) &= (1+x_2+x_1x_2+x_3+x_1x_3+x_2x_3+x_1x_2x_3+x_4+x_1x_4+x_2x_4\\
&\quad +x_1x_2x_4+x_3x_4+x_1x_3x_4+x_2x_3x_4+x_1x_2x_3x_4,x_2,x_3,x_4),\\
h_{t_{1,3}}(x_1,x_2,x_3,x_4) &= (x_1+x_3+x_1x_3+x_2x_3+x_1x_2x_3+x_3x_4+x_1x_3x_4\\
&\quad +x_2x_3x_4+x_1x_2x_3x_4,x_2,x_3,x_4),\\
h_{t_{1,4}}(x_1,x_2,x_3,x_4) &= (x_1+x_4+x_1x_4+x_2x_4+x_1x_2x_4+x_3x_4+x_1x_3x_4\\
&\quad +x_2x_3x_4+x_1x_2x_3x_4,x_2,x_3,x_4),\\
h_{t_{1,2}}(x_1,x_2,x_3,x_4) &= (x_1+x_1x_2+x_1x_2x_3+x_1x_2x_4+x_1x_2x_3x_4,x_2,x_3,x_4),\\
h_{t_{1,23}}(x_1,x_2,x_3,x_4) &= (x_1+x_1x_2x_3+x_1x_2x_3x_4,x_2,x_3,x_4),\\
h_{t_{1,24}}(x_1,x_2,x_3,x_4) &= (x_1+x_1x_2x_4+x_1x_2x_3x_4,x_2,x_3,x_4),\\
h_{t_{1,34}}(x_1,x_2,x_3,x_4) &= (x_1+x_3x_4+x_1x_3x_4+x_2x_3x_4+x_1x_2x_3x_4,x_2,x_3,x_4),\\
h_{t_{1,234}}(x_1,x_2,x_3,x_4) &= (x_1+x_1x_2x_3x_4,x_2,x_3,x_4),\\
h_{t_2}(x_1,x_2) &= (x_1,x_1x_2),\\
h_{t_{2,1}}(x_1,x_2) &= (x_1,x_1+x_2+x_1x_2),\\
h_{t_3}(x_1,x_3) &= (x_1,1+x_1+x_1x_3),\\
h_{t_{3,1}}(x_1,x_3) &= (x_1,x_3+x_1x_3),\\
h_{t_4}(x_1,x_4) &= (x_1,x_1x_4),\\
h_{t_{4,1}}(x_1,x_4) &= (x_1,x_1+x_4+x_1x_4).
\end{aligned}
$$

Notice that

$$
\begin{aligned}
\bullet 1 &= \{t_1,t_{1,3},t_{1,4},t_{1,2},t_{1,23},t_{1,24},t_{1,34},t_{1,234}\},\\
\bullet 2 &= \{t_2,t_{2,1}\},\\
\bullet 3 &= \{t_3,t_{3,1}\},\\
\bullet 4 &= \{t_4,t_{4,1}\}.
\end{aligned}
$$

It is easy to see that the second part of Theorem 5.7.2 holds.

The first part of Theorem 5.7.2 can easily be checked. For example, for node 2, we know that $g_2=x_1$ and

$$
\begin{aligned}
x_2+\sum_{t\in\bullet 2}((h_t)_2-x_2) &= x_2+((h_{t_2})_2-x_2)+((h_{t_{2,1}})_2-x_2)\\
&= x_2+(x_1x_2-x_2)+(x_1+x_2+x_1x_2-x_2)\\
&= x_1.
\end{aligned}
$$

By observing the polynomial form of the Petri net model we notice that we increased the complexity of the model but it did not increase the information the model can provide. The polynomial form retains all the information about the logical model. This is important because it means that by using the algebraic framework we are not increasing the complexity of the model, but we are providing additional algebraic structure that allows for a systematic analysis of the model.

5.8 Discussion

The problem of giving mathematical structure to logical models has been studied by several authors, e.g., [26]. For the purpose of computation and analysis, the structure proposed here provides a class of simple and easily defined mathematical objects that can model both logical models and bounded Petri nets. It has the advantage that it makes accessible the theoretical concepts, algorithms, and software from polynomial algebra, such as Gröbner basis theory, which underlies many of the algorithms for solving systems of polynomial equations.

It is important to mention that in this section we focus on finding steady states and dead markings, not because that is the only application of our framework, but because it can be translated directly into the algebraic problem of solving polynomial equations without much difficulty; the framework we propose can be also used to answer other questions. For instance, theory, algorithms, and software were used in [67] and subsequent papers to give a solution to the problem of reverse-engineering of gene regulatory networks from experimental time course data. This allows us to find all logical models that satisfy certain properties as done by SMBioNet (http://smbionet.lami.univ-evry.fr) with the additional advantage of allowing a systematic study and classification of such models. One can also solve the reverse problem, that is, given a family of models, by using our framework, we find properties that they all satisfy (see Section 5.5). Furthermore, the theory inherent in the algebraic framework can give rise to theorems. For example, the algebraic structure was used to give an exact formula for the structure of the phase space of linear systems [27; 46] and lower and upper bounds for the number of limit cycles of conjunctive and disjunctive Boolean networks [56]. Also, the family of nested canalizing functions has been shown to have an algebraic structure (toric variety) that allows their characterization [57]. These functions appear frequently in Boolean models of regulatory networks and their dynamics have desirable properties. Functionality of circuits can also be studied within this framework (see the Supporting Information of [115]).

Polynomial algebra can therefore complement the existing analysis tools for logical models and Petri nets, and we have shown several examples of its use. It is also worth mentioning that Petri nets are a special case of bipartite models, consisting of two sets of nodes, representing "places" and "events," respectively, connected by directed edges. More general bipartite models are quite common in systems biology, e.g., [116], and they might be amenable to analysis with similar methods. It is also worth mentioning that, while the ex-

amples used here are all gene regulatory networks, there are examples of logical/Petri net models of other kinds of molecular networks, see, e.g. [89], so that the analysis framework described here is more widely applicable.

A potential disadvantage of the algebraic framework is that it is less intuitive than either logical models or Petri nets. Even in the Boolean case it is often difficult to give a biological interpretation to a Boolean function in polynomial form, even if the equivalent Boolean expression is quite meaningful. However, with appropriate software, the typical user wanting to analyze a logical model or Petri net does not need to explicitly manipulate polynomial functions or even be aware that they are used in the analysis. On the other hand, some questions or aspects of logical models and Petri nets may not currently have a direct counterpart in the algebraic framework; for example, the asynchronous update commonly used in logical models does not currently have a direct counterpart in the polynomial algebra framework. This deserves further investigation.

Bibliography

[1] B. Aguda and A. Goryachev. From pathways databases to network models of switching behavior. *PLoS Computational Biology*, 3(9):1674–1678, 2007.

[2] Z. Agur, A.S. Fraenkel, and S.T. Klein. The number of fixed points of the majority rule. *Discrete Math.*, 70(3):295–302, 1988.

[3] R. Albert and H. Othmer. The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in *Drosophila melanogaster*. *Journal of Theoretical Biology*, 223:1–18, 2003.

[4] U. Alon. Network motifs: theory and experimental approaches. *Nature Reviews Genetics*, 8:450–461, 2007.

[5] I. Anderson. *Combinatorics of Finite Sets*. Dover Publications, Mineola, N.Y., 2002.

[6] J. Aracena. Maximum number of fixed points in regulatory boolean networks. *Bulletin of Mathematical Biology*, 2008.

[7] J. Aracena, J. Demongeot, and E. Goles. Fixed points and maximal independent sets in AND-OR networks. *Discrete Appl. Math.*, 138(3):277–288, 2004.

[8] J. Aracena, J. Demongeot, and E. Goles. On limit cycles of monotone functions with symmetric connection graph. *Theor. Comput. Sci.*, 322(2):237–244, 2004.

[9] J. Aracena, J. Demongeot, and E. Goles. Positive and negative circuits in discrete neural networks. *IEEE Trans Neural Networks*, 15(1):77–83, 2004.

[10] C. Barrett, W. Chen, and M. Zheng. Discrete dynamical systems on graphs and Boolean functions. *Math. Comput. Simul.*, 66(6):487–497, 2004.

[11] C. Barrett, C. Herring, J. Reed, and B. Palsson. The global transcriptional regulatory network for metabolism in *Escherichia coli* exhibits few dominant functional states. *Proc. Natl. Acad. Sci. U.S.A.*, 102(52):19103–19108, 2005.

[12] A. Berman and R. Plemmons. *Nonnegative matrices in the mathematical sciences*, volume 9 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1994. Revised reprint of the 1979 original.

[13] R. Brualdi and H. Ryser. *Combinatorial matrix theory*, volume 39 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 1991.

[14] C. Chaouiya, E. Remy, P. Ruet, and D. Thieffry. Qualitative modelling of genetic networks: From logical regulatory graphs to standard petri nets. *ICATPN'04*, pages 137–156, 2004.

[15] C. Chaouiya, E. Remy, and D. Thieffry. Petri net modelling of biological regulatory networks. *Journal of Discrete Algorithms*, 6:165–177, 2008.

[16] M. Chaves, R. Albert, and E.D. Sontag. Robustness and fragility of Boolean models for genetic regulatory networks. *J. Theoret. Biol.*, 235(3):431–449, 2005.

[17] O. Colón-Reyes, R. Laubenbacher, and B. Pareigis. Boolean monomial dynamical systems. *Annals of Combinatorics*, 8:425–439, 2004.

[18] J.-P. Comet, H. Klaudel, and S. Liauzu. Modeling multi-valued genetic regulatory networks using high-level petri nets. In G. Ciardo and P. Darondeau, editors, *Proceedings of the International Conference on the Application and Theory of Petri Nets*, pages 208–227. Springer-Verlag, 2005.

[19] P. Cull. Linear analysis of switching nets. *Kybernetik*, 8:31–39, 1971.

[20] E.N. Dancer. Some remarks on a boundedness assumption for monotone dynamical systems. *Proc. of the AMS*, 126:801–807, 1998.

[21] B. DasGupta, G.A. Enciso, E.D. Sontag, and Y. Zhang. Algorithmic and complexity aspects of decompositions of biological networks into monotone subsystems. *BioSystems*, 90:161–178, 2007.

[22] M. Davidich and S. Bornholdt. Boolean network model predicts cell cycle sequence of fission yeast. *PLoS ONE*, 3(2):e1672, Feb 2008.

[23] M. Davidich and S. Bornholdt. The transition from differential equations to Boolean networks: A case study in simplifying a regulatory network model. *Journal of Theoretical Biology*, 25:269–277, 2008.

[24] E. Dimitrova. *Polynomial models for systems biology: Data discretization and term order effect on dynamics*. PhD thesis, Virginia Polytechnic Institute and State University, 2006.

[25] E. Dimitrova, R. Laubenbacher, and J. McGee. Discretization of time series data. Available at `http://polymath.vbi.vt.edu/discretization/`, 2005.

[26] A. Egri-Nagy and C. Nehaniv. Algebraic properties of automata associated to Petri nets and applications to computation in biological systems. *Biosystems*, 94:297–307, 2008.

[27] B. Elspas. The theory of autonomous linear sequential networks. *IRE Transaction on Circuit Theory*, pages 45–60, March 1959.

[28] C. Espinosa-Soto, P. Padilla-Longoria, and E. Alvarez-Buylla. A gene regulatory network model for cell-fate determination during *Arabidopsis thaliana* flower development that is robust and recovers experimental gene expression profiles. *Plant Cell*, 16(11):2923–2939, 2004.

[29] A. Faure, A. Naldi, C. Chaouiya, and D. Thieffry. Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle. *Bioinformatics*, 22(14):124–131, 2006.

[30] D. Formanowicz, A. Sackmann, P. Formanowicz, and J. Blazewicz. Petri net based model of the body iron homeostasis. *J. Biomed. Informatics*, 40:476–485, 2007.

[31] J. Gauzé. Positive and negative circuits in dynamical systems. *Journal of Biological Systems*, 6(1):11–15, 1998.

[32] E. Gianchandani, J. Papin, N. Price, A. Joyce, and B. Palsson. Matrix formalism to describe functional states of transcriptional regulatory systems. *PLoS Computational Biology*, 2(8):0902–0917, August 2006.

[33] E.N. Gilbert. Lattice theoretic properties of frontal switching functions. *Journal of Mathematics and Physics*, 33:57–67, 1954.

[34] E. Goles and G. Hernández. Dynamical behavior of Kauffman networks with and-or gates. *Journal of Biological Systems*, 8(2):151–175, 2000.

[35] E. Goles and J. Olivos. Periodic behaviour of generalized threshold functions. *Discrete mathematics*, 30:187–189, 1980.

[36] A.G. Gonzalez, A. Naldi, L. Snchez, D.Thieffry, and C. Chaouiyag. Ginsim : a software suite for the qualitative modelling, simulation and analysis of regulatory networks. *Biosystems*, 84(2):91100, 2006.

[37] B. Goodwin. *Temporal Organization in Cells*. Academic Press, New York, 1963.

[38] F. Greil and B. Drossel. Kauffman networks with threshold functions. *European Physical Journal B*, 57:109–113, 2007.

[39] A. Griffiths, J. Miller, D. Suzuki, R. Lewontin, and W. Gelbart. *Introduction to Genetic Analysis*. W. H. Freeman and Co., New York, 1999.

[40] S. Grunwald, A. Speer, J. Ackermann, and I. Koch. Petri net modelling of gene regulation of the duchenne muscular dystrophy. *Biosystems*, 92(2):189–205, 2008.

[41] B.M. Gummow, J.O. Sheys, V.R. Cancelli, and G.D. Hammer. Reciprocal regulation of a glucocorticoid receptor-steroidogenic factor-1 transcription complex on the dax-1 promoter by glucocorticoids and adrenocorticotropic hormone in the adrenal cortex. *Mol. Endocrinology*, 20(11):2711–2723, 2006.

[42] K. Hadeler and D. Glas. Quasimonotone systems and convergence to equilibrium in a population genetics model. *J. Math. Anal. Appl.*, 95:297–303, 1983.

[43] Á. Halász, V. Kumar, M. Imieliński, C. Belta, O. Sokolsky, S. Pathak, and H. Rubin. Analysis of lactose metabolism in *E.coli* using reachability analysis of hybrid systems. *IET Systems Biology*, 1(2):130–148, 2007.

[44] F. Harary. On the notion of balance of a signed graph. *Michigan Mathematical Journal*, 2:143–146, 1953.

[45] S. Harris, B. Sawhill, A. Wuensche, and S. Kauffman. A model of transcriptional regulatory networks based on biases in the observed regulation rules. *Complex.*, 7(4):23–40, 2002.

[46] A. Hernández-Toledo. Linear finite dynamical systems. *Communications in Algebra*, 33(9):2977–2989, 2005.

[47] M. Hirsch. Differential equations and convergence almost everywhere in strongly monotone flows. *Contemporary Mathematics*, 17:267–285, 1983.

[48] M. Hirsch. Systems of differential equations that are competitive or cooperative ii: Convergence almost everywhere. *SIAM J. Mathematical Analysis*, 16:423–439, 1985.

[49] M. Hirsch and H.L. Smith. Monotone dynamical systems. In *Handbook of Differential Equations, Ordinary Differential Equations (second volume)*. Elsevier, Amsterdam, 2005.

[50] M.W. Hirsch. The dynamical systems approach to differential equations. *Bull. A.M.S.*, 11:1–64, 1984.

[51] F. Hüffner, N. Betzler, and R. Niedermeier. Optimal edge deletions for signed graph balancing. In *Proceedings of the 6th Workshop on Experimental Algorithms (WEA07), June 6-8, 2007, Rome*. Springer-Verlag, 2007.

[52] F. Jacob and J. Monod. Genetic regulatory mechanisms in the synthesis of proteins. *Journal of Molecular Biology*, 3:318–356, 1961.

[53] A. Jarrah, R. Laubenbacher, B. Stigler, and M. Stillman. Reverse-engineering of polynomial dynamical systems. *Advances in Applied Mathematics*, 39(4):477–489, 2007.

[54] A. Jarrah, R. Laubenbacher, M. Stillman, and P. Vera-Licona. An efficient algorithm for the phase space structure of linear dynamical systems over finite fields. Submitted, 2008.

[55] A. Jarrah, R. Laubenbacher, and H. Vastani. DVD: Discrete visualizer of dynamics. Available at `http://dvd.vbi.vt.edu`.

[56] A. Jarrah, R. Laubenbacher, and A. Veliz-Cuba. The dynamics of conjunctive and disjunctive boolean network models. *Bull. Math. Bio.*, 2010.

[57] A. Jarrah, B. Raposa, and R. Laubenbacher. Nested canalyzing, unate cascade, and polynomial functions. *Physica D:Nonlinear Phenomena*, 233(2):167–174, 2007.

[58] S. Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *J. Theor. Biol.*, 22:437–467, 1969.

[59] S. Kauffman, C. Peterson, B. Samuelsson, and C. Troein. Random Boolean network models and the yeast transcriptional network. *PNAS*, 100(25):14796–14799, 2003.

[60] S. Kauffman, C. Peterson, B. Samuelsson, and C. Troein. Genetic networks with canalyzing boolean rules are always stable. *PNAS*, 101(49):17102–17107, 2004.

[61] S.A. Kauffman. Homeostasis and differentiation in random genetic control networks. *Nature*, 224:177–178, 1969.

[62] S.A. Kauffman. *Origins of Order: Self-Organization and Selection in Evolution*. Oxford Univ. Press, Oxford, 1993.

[63] S.A. Kauffman and K. Glass. The logical analysis of continuous, nonlinear biochemical control networks. *Journal of Theoretical Biology*, 39:103–129, 1973.

[64] J. Kielbassa, R. Bortfeldt, S. Schuster, and I. Koch. Modeling the u1 snrnp assembly pathway in alternative splicing in human cells using Petri nets. *Comput. Biol. Chem.*, 33(1):46–61, 2009.

[65] S. Klamt, J. Saez-Rodriguez, J. Lindquist, L. Simeoni, and E. Gilles. A methodology for the structural and functional analysis of signaling and regulatory networks. *BMC Bioinformatics*, 7(56), 2006.

[66] Y.-K. Kwon and K.-H. Cho. Boolean dynamics of biological networks with multiple coupled feedback loops. *Biophysical Journal*, 92:2975–2981, 2007.

[67] R. Laubenbacher and B. Stigler. A computational algebra approach to the reverse engineering of gene regulatory networks. *Journal of Theoretical Biology*, 229:523–537, 2004.

[68] F. Li, T. Long, Y. Lu, Q. Ouyang, and C. Tang. The yeast cell-cycle network is robustly designed. *Proc. Natl. Acad. Sci. U.S.A.*, 101(14):4781–4786, 2004.

[69] S. Li, S.M. Assmann, and R. Albert. Predicting essential components of signal transduction networks: A dynamic model of guard cell abscisic acid signaling. *PLoS. Biol.*, 4(10), 2006.

[70] R. Lidl and H. Niederreiter. *Finite fields*, volume 20 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, second edition, 1997.

[71] A. Maayan, R. Iyengar, and Eduardo D. Sontag. Intracellular regulatory networks are close to monotone systems. Technical Report npre.2007.25.1, Nature Precedings, January 2007.

[72] S. Mangan and Uri Alon. Structure and function of the feed-forward loop network motif. *PNAS*, 100:11980–11985, 2003.

[73] A. Mayo, Y. Setty, S. Shavit, A. Zaslaver, and U. Alon. Plasticity of the *cis*-regulatory input function of a gene. *PLoS Biology*, 4(4):0555–0561, 2006.

[74] L. Mendoza. A network model for the control of the differentiation process in Th cells. *Biosystems*, 84:101–114, 2006.

[75] L. Mendoza, D. Thieffry, and E. Alvarez-Buylla. Genetic control of flower morphogenesis in arabidopsis thaliana: a logical analysis. *Bioinformatics*, 15:593–606, 1999.

[76] L. Mendoza and I. Xenarios. A method for the generation of standardized qualitative dynamical systems of regulatory networks. *Theoretical Biology and Medical Modelling*, 3(13):1–18, 2006.

[77] M. Merika and S.H. Orkin. Functional synergy and physical interactions of the erythroid transcription factor gata-1 with the krüppel family proteins sp1 and eklf. *Mol. Cell. Biol.*, 15(5):2437–2447, 1995.

[78] M.L. Minsky. *Computation: finite and infinite machines*. Prentice-Hall, Englewood Cliffs, N.J., 1967.

[79] A. Naldi, D. Thieffry, and C. Chaouiya. Decision diagrams for the representation and analysis of logical models of genetic networks. *CMSB'07*, LNCS/LNBI 4695:233–247, 2007.

[80] A. Naldi, D. Thieffry, and Claudine Chaouiya. Decision diagrams for the representation and analysis of logical models of genetic networks. In M. Calder and S. Gilmore, editors, *CMSB*, pages 233–247. Springer-Verlag, 2007.

[81] D. H. Nguyen and P. D'haeseleer. Deciphering principles of transcription regulation in eucaryotic genomes. *Mol. Sys. Biol.*, doi:10.1038/msb4100054, 2006.

[82] S. Nikolajewaa, M. Friedela, and T. Wilhelm. Boolean networks with biologically relevant rules show ordered behaviorstar, open. *Biosystems*, 90(1):40–47, 2007.

[83] A. Novick and M. Weiner. Enzyme induction as an all-or-none phenomenon. *Proceedings of the National Academy of Sciences of the United States of America*, 43(7):553–566, 1957.

[84] E. Ozbudak, M. Thattai, H. Lim, B. Shraiman, and A. van Oudenaarden. Multistability in the lactose utilization network of *Escherichia coli*. *Nature*, 427:737–740, 2004.

[85] M. Peleg, D. Rubi, and R. Altman. Using Petri net tools to study properties and dynamics of biological systems. *J. Amer. Medical Informatics Assoc.*, 12(2):181–199, 2005.

[86] E. Plahte, T. Mestl, and S. Omholt. Feedback loops, stability and multistationarity in dynamical systems. *Journal of Biological Systems*, 3(2):409–413, 1995.

[87] L. Raeymaekers. Dynamics of boolean networks controlled by biologically meaningful functions. *J. Theor. Biol.*, 218(3):331–341, 2002.

[88] E. Remy and P. Ruet. From elementary signed circuits to the dynamics of boolean regulatory networks. *Bioinformatics*, 24:220–226, 2008.

[89] E. Remy, P. Ruet, L. Mendoza, D. Thieffry, and C. Chaouiya. From logical regulatory graphs to standard petri nets: Dynamical roles and functionality of feedback circuits. *In Transactions on Computation Systems Biology VII (TCSB)*, pages 55–72, 2006.

[90] A. Richard and J.-P. Comet. Necessary conditions for multistationarity in discrete dynamical systems. *Discrete Applied Mathematics*, 155:2403–2413, 2007.

[91] V. N. Sachkov and V. E. Tarakanov. *Combinatorics of nonnegative matrices*, volume 213 of *Translations of Mathematical Monographs*. American Mathematical Society, Providence, RI, 2002. Translated from the 2000 Russian original by Valentin F. Kolchin.

[92] J. Saez-Rodriguez, L. Alexopoulos, J. Epperlein, R. Samaga, D. Lauffenburger, S. Klamt, and P. Sorger. Discrete logic modelling as a means to link protein signalling networks with functional analysis of mammalian signal transduction. *Mol. Sys. Biol.*, 5:331, 2009.

[93] A. Samal and S. Jain. The regulatory network of *E. coli* metabolism as a Boolean dynamical system exhibits both homeostasis and flexibility of response. *BMC Syst. Biol.*, 2(21), 2008.

[94] M. Santillán. Bistable behavior in a model of the *lac* operon in *Escherichia coli* with variable growth rate. *Biophysical Journal*, 94(6):20652081, 2008.

[95] M. Santillán and M. Mackey. Influence of catabolite repression and inducer exclusion on the bistable behavior of the *lac* operon. *Biophysical Journal*, 86(3):1282–1292, 2004.

[96] M. Santillán, M. Mackey, and E. Zeron. Origin of bistability in the *lac* operon. *Biophysical Journal*, 92(11):3830–3842, 2007.

[97] B. De Schutter and B. De Moor. On the sequence of consecutive powers of a matrix in a Boolean algebra. *SIAM Journal on Matrix Analysis and Applications*, 21(1):328–354, 2000.

[98] Y. Setty, A. Mayo, M. Surette, and U. Alon. Detailed map of a cis-regulatory input function. *Proceedings of the National Academy of Sciences of the United States of America*, 100(13):77027707, 2003.

[99] H. Smith. *Monotone dynamical systems: An introduction to the theory of competitive and cooperative systems, Mathematical Surveys and Monographs, vol. 41.* AMS, Providence, RI, 1995.

[100] E. Sontag, A. Veliz-Cuba, R. Laubenbacher, and A. Jarrah. The effect of negative feedback loops on the dynamics of boolean networks. *Biophysical Journal*, 95:518–526, 2008.

[101] E.D. Sontag. Molecular systems biology and control. *Eur. J. Control*, 11(4-5):396–435, 2005.

[102] E.D. Sontag. Monotone and near-monotone biochemical networks. *Journal of Systems and Synthetic Biology*, 1(2):59–87, 2007.

[103] C. Soule. Graphic requirements for multistationarity. *ComPlexUs*, 1:123–133, 2003.

[104] R. P. Stanley. *Enumerative Combinatorics I.* Translations of Mathematical Monographs. Wadsworth Inc., Belmont, CA, 1986.

[105] L. J. Steggles, R. Banks, O. Shaw, and A. Wipat. Qualitatively modelling and analysing genetic regulatory networks: a petri net approach. *Bioinformatics*, 23:336–343, 2007.

[106] L.J. Steggles, R. Banks, and A. Wipat. Modelling and analysing genetic networks: From boolean networks to petri nets. In *Computational Methods in Systems Biology. International Conference*, pages 127–141. Springer-Verlag, 2006.

[107] B. Stigler and A. Veliz-Cuba. Network topology as a driver of bistability in the lac operon. *http://arxiv.org/abs/0807.3995*, 2008.

[108] D. Thieffry and R. Thomas. Dynamical behavior of biological regulatory networks–II. Immunity control in bacteriophage lambda . *Bull. Math. Biol.*, 57:277–297, 1995.

[109] R. Thomas. Boolean formalisation of genetic control circuits. *J. Theor. Biol.*, 42:565–583, 1973.

[110] R. Thomas. *Biological Feedback*. CRC, 1990.

[111] R. Thomas, D. Thieffry, and M. Kaufman. Dynamical behaviour of biological regulatory networks - i. biological role of feedback loops and practical use of the concept of the loop-characteristic state. *Bulletin of Mathematical Biology*, 57(2):247–276, 1995.

[112] Predrag T. Tosic and Gul Agha. Characterizing configuration spaces of simple threshold cellular automata. In *ACRI*, pages 861–870, 2004.

[113] M. van Hoek and P. Hogeweg. In silico evolved *lac* operons exhibit bistability for artificial inducers, but not for lactose. *Biophysical Journal*, 91(8):28332843, 2006.

[114] M. van Hoek and P. Hogeweg. The effect of stochasticity on the *Lac* operon: An evolutionary perspective. *PLoS Computational Biology*, 3(6):1071–1082, 2007.

[115] A. Veliz-Cuba, A. Jarrah, and R. Laubenbacher. Polynomial algebra of discrete models in systems biology. *Bioinformatics*, 26:1637–1643, 2010.

[116] P. Warren, S. Queiros, and J. Jones. Flux networks in metabolic graphs. *Phys. Biol.*, 6(4):46006, 2009.

[117] P. Wong, S. Gladney, and J. Keasling. Mathematical model of the *lac* operon: Inducer exclusion, catabolite repression, and diauxic growth on glucose and lactose. *Biotechnology Progress*, 13(2):132–143, 1997.

[118] M. Yeung, J. Tegnér, and J. Collins. Reverse engineering gene networks using singular value decomposition and robust regression. *Proceedings of the National Academy of Science of the United States of America*, 99(9):6163–6168, 2002.

[119] N. Yildirim, M. Santillán, D. Horike, and M. Mackey. Dynamics and bistability in a reduced model of the *lac* operon. *Chaos*, 14(2):279–292, 2004.

[120] T. Zaslavsky. Bibliography of signed and gain graphs. *Electronic Journal of Combinatorics*, DS8, 1998.

[121] Q. Zhao. A remark on scalar equations for synchronous boolean networks with biological applications by C. Farrow, J. Heidel, J. Maloney, and J. Rogers. *IEEE Transactions on Neural Networks*, 16(6):1715–1716, 2005.