Clemson University
TigerPrints

All Dissertations | Dissertations

8-2016

Algebraic Geometry Arising from Discrete Models of Gene Regulatory Networks

Qijun He
Clemson University

Follow this and additional works at: http://tigerprints.clemson.edu/all_dissertations

Recommended Citation
He, Qijun, "Algebraic Geometry Arising from Discrete Models of Gene Regulatory Networks" (2016). *All Dissertations.* Paper 1730.

This Dissertation is brought to you for free and open access by the Dissertations at TigerPrints. It has been accepted for inclusion in All Dissertations by an authorized administrator of TigerPrints. For more information, please contact awesole@clemson.edu.

ALGEBRAIC GEOMETRY ARISING FROM DISCRETE MODELS OF GENE REGULATORY NETWORKS

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Mathematical Sciences

---

by
Qijun He
August 2016

Accepted by:
Dr. Matthew Macauley, Committee Chair
Dr. Elena Dimitrova
Dr. Svetlana Poznanovikj
Dr. Neil Calkin

Abstract

Discrete models of gene regulatory networks have gained popularity in computational systems biology over the last dozen years. However, not all discrete network models reflect the behaviors of real biological systems. In this work, we focus on two model selection methods and algebraic geometry arising from these model selection methods.

The first model selection method involves biologically relevant functions. We begin by introducing k-*canalizing functions*, a generalization of nested canalizing functions. We extend results on nested canalizing functions and derived a unique extended monomial form of arbitrary Boolean functions. This gives us a stratification of the set of n-variable Boolean functions by canalizing depth. We obtain closed formulas for the number of n-variable Boolean functions with depth k, which simultaneously generalizes enumeration formulas for canalizing, and nested canalizing functions. We characterize the set of k-canalizing functions as an algebraic variety in $\mathbb{F}_2^{2^n}$. Next, e propose a method for the reverse engineering of networks of k-canalizing functions using techniques from computational algebra, based on our parametrization of k-canalizing functions. We also analyze binary decision diagrams of k-canalizing functions.

The second model selection method involves computing minimal polynomial models using Gröbner bases. We built up the connection between staircases and Gröbner bases. We provided a necessary and sufficient condition for the ideal $\mathbb{I}(V)$ to have a unique reduced Gröbner basis, using the concept of a basic staircase. We also provide a sufficient combinatorial characterization of $V \subset \mathbb{N}_p^n$ that yields a unique reduced Gröbner basis.

Dedication

To my parents, Hua Wang and Shiqiang He.

Acknowledgments

Foremost, I would like to express my deepest gratitude to my advisor, Matthew Macauley, whose immense knowledge, wisdom and encouragement has made my past five years an amazing life-changing experience. I am grateful for the guidance and mentorship he generously offered, not only on my career development, but also on how to be a good citizen in academia and what is truly important in life.

I would like to thank my committee members, Svetlana Poznanovikj and Neil Calkin, for their helpful comments during the research process and for all you have taught me in and out of the classroom. Sincere thanks to Elena Dimitrova for all of your insight and many contributions, especially with the unique Gröbner basis work.

During the course of my graduate studies, I have benefited significantly from the broad range of courses offered at Clemson. Besides, I have been fortunate to work with a few great scholars: Christine Heitsch, Elizabeth Drellich, Andrew Gainer-Dewar and Heather A. Harrington during the Mathematics Research Communities program, as well as Brandilyn Stigler, who I collaborated with remotely. Their expertise has enriched my research experience.

Finally, I am eternally grateful to my parents, Hua Wang and Shiqiang He, who brought me to this wonderful world and have always been there for me. I am also extremely blessed to have found my special one, Xiaoqing Gu, to accompany me throughout my Ph.D. journey, and the unknown but exciting adventures ahead of me.

Table of Contents

Chapter 1

Introduction

1.1 Gene Regulatory Networks in Molecular Biology

The word "gene" has become an increasing popular topic these days. Children learn early that who we, and the living world around us, is encoded in each and every cell of the organism. The concept of gene even shows up in common parlance, such as "it is in my genes". Indeed, genes hold the information to build and maintain an organism's cells and pass genetic traits to offspring. However, thinking of genes simply as a collection of building blocks of one's body paints an incomplete picture. It hides the fact that these "building blocks" know how to communicate and interact with each other. In other words, genes also contain instructions for the mechanisms through which genetic information is extracted and plays a role in cellular processes, such as controlling the response of a cell to environmental signals and replication of the DNA preceding the cell division. This process of genetic information extracting and utilizing is part of what is known as *gene regulation*. Gene regulation is an intricate process whose complexity makes it an extreme challenge for mathematical modeling, since it not only involves genes, but also involves DNA, RNA, proteins, and small molecules. This gives rise to the notion of a *gene regulatory network* (GRN). A GRN is a collection of regulators, which can be genes, proteins,

or enzymes, that interact with each other for a specific purpose. For example, a simple GRN consists of one or more input genes, metabolic and signaling pathways, regulatory proteins that integrate the input signals, several target genes, and the RNA and proteins produced from target genes.

The first discovery of a GRN is widely considered to be the identification in 1961 of the *lac* operon, discovered by Jacques Monod [23], in which proteins involved in lactose metabolism are expressed by *Escherichia coli* and some other enteric bacteria only in the presence of lactose and absence of glucose. Since its discovery, the *lac* operon has often been used as a model system of gene regulation. Over the past five decades, improvements in biotechnology have greatly accelerated the amount of experimental data available. While on the other hand, such explosion of data brings growing challenges for organizing the overwhelming amounts of disparate experimental data and for developing models that reflect the dependencies between the system's components. Different types of mathematical models have been developed in an attempt to capture gene regulatory mechanisms and dynamics [7, 18]. Most mathematical models of GRNs have been given as systems of differential equations, but discrete modeling frameworks are increasingly receiving attention for their use in offering global insights [34]. Discrete models tend to be simpler and can be more intuitive than continuous models. Also, discrete models do not depend on estimation of initial conditions or parameters which is quite an advantage over their continuous counterparts. Moreover, discrete models consider the effects of individual components within the network, not just measuring the network as a whole, so it is possible to observe how altering or perturbing a subset of the components can affect system dynamics. Finally, some discrete models have convenient algebraic representations, allowing us to employ tools and algorithms from algebraic geometry and computational algebra to construct appropriate network models.

One of the most studied discrete models is the Boolean Network, where each variable can only take 0 or 1 as its value (often interpreted as "absent" and "present", or "off" and "on").

Boolean models were first introduced to biology in 1969 to study the dynamic properties of gene regulatory networks [34]. They are useful in the case where one is interested in qualitative behavior. In particular, when network dynamics are determined by logical interactions rather than finely tuned kinetics, which may often be unknown, Boolean models would be one of the preferred candidate models.

1.2 Reverse Engineering of Gene Regulatory Networks

Technological advances in the life sciences have triggered an enormous accumulation of experimental data representing the activities of the living cell. Furthermore, few GRNs are as well understood as the *lac* operon, which prohibits model construction that is purely based on knowledge of the GRN. Therefore, instead of constructing a model and tune it to fit the data, people develop modeling methods that generate models directly from the data. These data-driven modeling methods are sometimes referred to as *reverse engineering*.

Generally speaking, the goal of reverse engineering in systems biology is to recover the *network topology* and *regulatory functions* of a network from observations. Network topology refers to the physical structure of the network, that is, how the components in the network are connected. It is often encoded as a directed graph, or a *wiring diagram*, where vertices represent the components of the network (genes, DNA, RNA, proteins, small molecules, etc.) and a directed edge is drawn between two vertices if one component is directly affected by the other (regulation, activation, prohibition etc.). Regulatory functions refer to the mechanism that how each component is affected by other components.

1.3 Boolean Networks

Given a gene regulatory network, one can create a *Boolean network* model by assigning Boolean variables to each node and representing the interactions as Boolean functions. Boolean networks are discrete-time, discrete-space dynamical systems first proposed by Stuart Kauffman in 1969 as models of GRNs [34]. A similar Boolean framework, called *logical models*, was proposed by René Thomas in 1973 [59].

If we take the vertex set of a Boolean network to be $V = \{1, 2, \ldots, n\}$, then the state of a node i is a Boolean variable $x_i \in \mathbb{F}_2 = \{0, 1\}$, and the vector

$$x(t) = (x_1(t), \ldots, x_n(t)) \in \mathbb{F}_2^n$$

is called the *system state*. Time is also discretized into steps $t = 0, 1, 2, \ldots$. Each node j has an *update function* $f_j \colon \mathbb{F}_2^n \to \mathbb{F}_2$ that determines the value of x_j for the next time step. Though the domain of f_j is $\mathbb{F}_2^n$, this function can only depend on the states of the nodes i such that (i, j) is an edge in the wiring diagram. Sometimes, these functions are written using Boolean variables, and other times they are written in polynomial form. For example, if we want to say that "*gene C is on if gene A is on and enzyme B is not present*", where by "on" we mean "being transcribed", we may write

$$f_C(t+1) = f_C(A(t), B(t)) = A(t) \wedge \overline{B(t)}\,. \tag{1.1}$$

Since it is understood that Boolean variables are functions of time, we will usually just write the above example as

$$f_C(A, B) = A \wedge \overline{B}\,.$$

At each time-step t, the states of each node are recomputed via the *global update function* $f \colon \mathbb{F}_2^n \to \mathbb{F}_2^n$ to get a new system state, $x(t+1)$. The most commonly used global update

function in Boolean models simply updates the nodes synchronously:

$$x(t+1) = f(x(t)) = (f_1(x(t)), \ldots, f_n(x(t)) \,. \tag{1.2}$$

Some models use an asynchronous update [44], but this raises the question of which of the possible $n!$ update orders to use. Thus, we will henceforth assume that a synchronous global update is used.

Definition 1. *A* Boolean network *is a pair* $(X, \mathcal{F})$ *consisting of a finite set of nodes* X *and a set* $\mathcal{F} = \{f_i\}_{i \in X}$ *of update functions, where each* $f_i \colon \mathbb{F}_2^n \to \mathbb{F}_2$.

Given a Boolean network $(X, \mathcal{F})$ as defined above, the wiring diagram is easy to construct – it is a directed graph with vertex set X and an edge (i, j) for each x_i that appears in the equation for f_j and is not fictitious. One can also construct its global update map $f : \mathbb{F}_2^{2^n} \to \mathbb{F}_2^{2^n}$, its wiring diagram, and a directed graph, called its *phase space*, that completely encodes the dynamics.

Definition 2. *The* phase space *of a Boolean network is the directed graph whose nodes are the* 2^n *system states and whose edge set is*

$$E = \{(x, f(x)) \mid x \in \mathbb{F}_2^n\}.$$

Each node in the phase space has exactly one out-going edge. Consequently, there are two types of nodes: those that lie on a directed cycle, called *periodic states*, and those that do not, called *transient states*. Every periodic state lies in a *cycle* of length $k \geq 1$. States on length-1 cycles are called *fixed points*. Transient states lie on chains that lead into periodic cycles. A transient state that has no predecessor is called a *garden-of-Eden state*.

As an example, consider a Boolean network on 3 nodes: $X = \{1, 2, 3\}$ with update functions $\mathcal{F} = \{f_i\}_{i=1}^3$ as shown in Figure 1.1.

Though Boolean networks are widely used as models of biological networks, one must

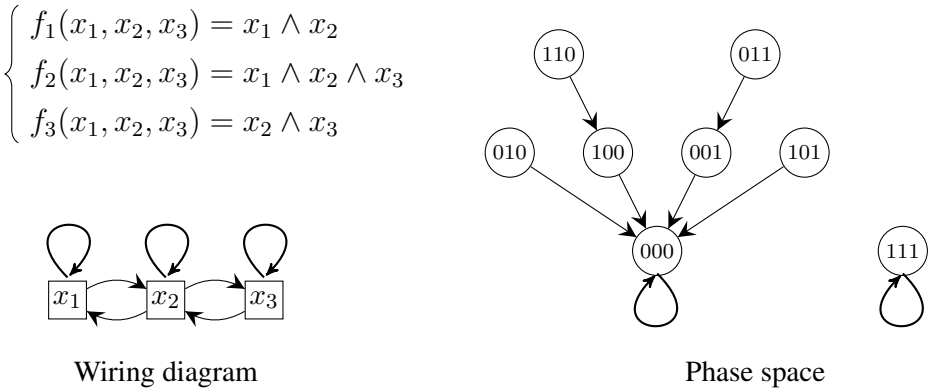

Figure 1.1: A simple Boolean network $(X, \mathcal{F})$ on 3 nodes.

remember the old adage that "*all models are wrong, but some models are useful*." Like any mathematical model, Boolean networks have several artifacts that have drawn criticism. One of these is the synchronous update: biological networks do not have a universal "central clock". In real gene regulatory networks, the state of each regulator is updated asynchronously. However, making the simplifying assumption of synchronous update will often still lead us to the correct overall network dynamics. A real gene regulatory network has to be robust enough so that it can withstand, so changing update order should not change the network dynamics drastically. Another issue is that the network is assumed to be static, whereas in reality, edges are continually added, removed, and changed. For example, consider a model of a disease network where edges represent social contacts. These social contacts are usually temporary. You might be in contact with person A in the morning, and then person B in the evening. In this case, you cannot infect person A with person B's disease, despite the fact that you're connected to both in a static social network. The interdisciplinary field of "evolving networks" is very popular due to these issues and more, but it is still in its infancy [3]. This is less of an issue for molecular networks, since molecular networks are often static.

We can rewrite a Boolean function in logical expression as a polynomial using the fol-

lowing arithmetic:

$$f(x,y) = x \wedge y \qquad f(x,y) = xy$$
$$g(x,y) = x \vee y \qquad g(x,y) = x + y + xy$$
$$h(x) = \overline{x} \qquad h(x) = 1 + x.$$

Hence a Boolean network is a special case of *polynomial dynamical systems* (PDS) [37] with $\mathbb{F} = \mathbb{F}_2$. The fact that the update function can be expressed as polynomial allows us to employ tools and algorithms from algebraic geometry and computational algebra to construct appropriate network models. In the remainder of this dissertation, when we mention a Boolean function, we would assume it is written in polynomial form.

1.4 Model Selection

One prominent problem in the application of Boolean models is that of selecting a model that is "biologically meaningful". Random Boolean networks were initially introduced by Kauffman [34, 35] as gene network models. In this setup, input variables are randomly selected for each node, (i.e., the wiring diagram is randomly wired), and each component is assigned a random update function according to a specified probability distribution. However, this model is often not a good mimic of real biological systems, since not all wiring diagrams and Boolean functions exhibit biological behavior. As a result, different model selection strategies have been developed, with the hope that restricting the model selection to networks with more appropriate dynamic behaviors. Some model selection strategies incorporate appropriate restrictions about the network topology so that the selected model will possess some desired property [21, 60]. Other model selection strategies restrict the update functions to be some specific biologically motivated classes of functions. For instance, the chain functions [19], the biologically meaningful functions [50], and the nested canalizing functions [32] have all been proposed due to their biologically relevant properties.

Chapter 2

Computational Algebra Basics

2.1 Ideals and Varieties

Ideals and varieties are essential structures in discrete modeling, especially for their utility in reverse engineering. Here, we include several key definitions and properties, as presented in [47].

Definition 3. *Let $\mathbb{F}$ be a field and $f_1, \ldots, f_s \in \mathbb{F}[x_1, \ldots, x_n]$. Then the set*

$$\mathbb{V}(f_1, \ldots, f_s) = \{(a_1, \ldots, a_n) \in \mathbb{F}^n : f_i(a_1, \ldots, a_n) = 0, 1 \leq i \leq n\}$$

is the affine variety *defined by $f_1, \ldots, f_s$.*

Thus, an affine variety $\mathbb{V}(f_1, \ldots, f_s) \subseteq \mathbb{F}^n$ is the set of all solutions of the system of equations $f_1(x_1, \ldots, x_n) = \cdots = f_s(x_1, \ldots, x_n) = 0$. Several important relationships exist between ideals and affine varieties. In particular, given a variety $V \in \mathbb{F}^n$, the set of *all* polynomials that vanish on V forms an ideal.

Definition 4. *Let $V \in \mathbb{F}^n$ be an affine variety, then we set*

$$\mathbb{I}(V) = \{f \in \mathbb{F}[x_1, \ldots, x_n] : f(a_1, \ldots, a_n) = 0, \textit{ for all } (a_1, \ldots, a_n) \in V\}.$$

The crucial observation is that $\mathbb{I}(V)$ is an ideal, called the ideal of V.

On the other hand, if $I \subseteq \mathbb{F}[x_1, \ldots, x_n]$ is an ideal, then

$$\mathbb{V}(I) = \{x \in \mathbb{F}^n : f(x) = 0, \forall f \in I\}$$

is an affine variety. In particular, if $I = \langle f_1, \ldots, f_s \rangle$, then $\mathbb{V}(f_1, \ldots, f_s) = \mathbb{V}(I)$. We can think of an ideal I as a collection of relations and $\mathbb{V}(I)$ is the set of points that satisfy these relations. As some relations can be derived from other relations, we just need to check the set of relations $f_1, \ldots, f_s$ that generates the ideal I instead of checking all relations in I.

An important relation between ideals and their associated varieties is the Ideal-Variety Correspondence, a result of Hilbert's well-known Nullstellensatz. For a field $\mathbb{F}$, this correspondence tells us that

1. For $V \subseteq \mathbb{F}^n$, $\mathbb{V}(\mathbb{I}(V)) = V$.

2. If $\mathbb{F}$ is algebraically closed and I is a radical ideal, then $\mathbb{I}(\mathbb{V}(I)) = I$.

In the case of Boolean networks, we will be working with polynomials over $\mathbb{F}_2$, which is not algebraically closed.

2.2 Gröbner Bases

As mentioned previously, instead of working on the whole ideal, it is often sufficient to work on a set of polynomials that generates the ideal, or a basis of the ideal. In practice,

people often prefer the basis to have some desired property, and consist of as few polynomials as possible. In particular, Gröbner bases play critical roles in PDS models of GRNs. Here, we present several definitions and properties associated with Gröbner bases which may be found in [47, 62]. A Gröbner basis is dependent upon its so-called monomial ordering. A polynomial in $\mathbb{F}[x_1, \ldots, x_n]$ is a linear combination of monomials of the form $x^\alpha := x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ over k, where α is the *n*-tuple exponent $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$.

Definition 5. *A* monomial order *is a total order on $\mathbb{N}^n$, satisfying:*

$$\alpha \prec \beta \Longrightarrow \alpha + \gamma \prec \beta + \gamma,$$

$$0 \prec \alpha,$$

for any α, β and γ in $\mathbb{N}^n$.

A monomial order is a total order on $\mathbb{N}^n$ that is compatible with polynomial multiplication and the degree of each variable. It is a natural extension of the elementary concept of degree for multivariate polynomials.

Example 6. *One common monomial ordering is the* lexicographic order*, which can be considered an alphabetical ordering. Here $\alpha \prec_{lex} \beta$ if the leftmost nonzero entry of $\beta - \alpha$ is positive. Notice that a particular order of the variables is assumed, and by changing this, we obtain $n!$ nonequivalent lexicographic orderings. For example, under the lexicographic order with $z \prec y \prec x$, we have $xz \prec xy$.*

Definition 7. *Let $f = \sum_{\alpha \in \mathbb{N}^n} a_\alpha x^\alpha \in \mathbb{F}[x_1, \ldots, x_n]$ be nonzero and $\prec$ be a monomial order. Then*

1. *The* multidegree *of f is $multideg(f) = \max_\prec \{\alpha \in \mathbb{N}^n : a_\alpha \neq 0\}$.*
2. *The* leading coefficient *of f is $LC(f) = a_{multideg(f)} \in \mathbb{F}$.*

3. *The* leading monomial *of f is $LM(f) = x^{multideg(f)}$.*

4. *The* leading term *of f is $LT(f) = LC(f) \cdot LM(f)$.*

Finally, for an ideal $I \subseteq \mathbb{F}[x_1, \ldots, x_n]$, $LT(I)$ is the set of leading terms of polynomials in I, and $\langle LT(I) \rangle$ is the ideal generated by $LT(I)$. $\langle LT(I) \rangle$ is called the *initial ideal* of I. We can now formally define a Gröbner basis.

Definition 8. *Let $\prec$ be a monomial order and $I \subseteq \mathbb{R}[x_1, \ldots, x_n]$ be nonzero. Then a subset $G = \{g_1, \ldots, g_t\}$ is a Gröbner basis for I if $\langle LT(g_1), \ldots, LT(g_t) \rangle = \langle LT(I) \rangle$. A Gröbner basis is* reduced *if the leading coefficient of each element of the basis is 1 and no monomial in any element of the basis is in the ideal generated by the leading terms of the other elements of the basis.*

The well-known Hilbert Basis Theorem tells us that this basis exists and is finitely generated. Given a fixed monomial ordering, an ideal I has a unique reduced Gröbner basis.

Example 9. *Let $I = \langle f_1, f_2 \rangle \in \mathbb{Z}[x, y]$, with $f_1 = x$ and $f_2 = x^2 + y$. Using lexicographic order with $y \prec x$, $y = x \cdot f_1 + 1 \cdot f_2$, so $y \in \langle LT(I) \rangle$. However, y is divisible by neither x nor x^2, so $y \notin \langle LT(f_1), LT(f_2) \rangle$. Therefore I is not a Gröbner basis for this ideal.*

Even with a fixed monomial ordering, polynomial division is not unique, as it depends on the order of the divisors.

Example 10. *Let $f_1 = y^2 + 1, f_2 = xy + 1 \in \mathbb{Z}[x, y]$ and $f = 2xy^2 + x - y$. Using lexicographic order with $y \prec x$ and the division algorithm for multivariate polynomials in [47], if we divide by f_1 and then f_2, we obtain a remainder of $-x - y$; however, if we reverse the order of the divisors, our remainder is $x - 3y$.*

If we are dividing by a Gröbner basis, however, our remainder is unique regardless of the order of the divisors. The *normal form* of a polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ with respect to an

ideal $I \subseteq \mathbb{F}[x_1, \ldots, x_n]$ is the remainder when dividing f by G, where G is the Gröbner basis for I. This normal form is unique up to monomial order, and f lies in I if and only if the normal form for f is zero. Moreover, the normal form consists of monomials that are not in $\langle LT(I) \rangle$. The monomials which do not lie in $\langle LT(I) \rangle$ are called *standard monomials*. Let $SM(I)$ denote the set of standard monomials of I. In particular, when $I = \mathbb{I}(V)$ is the ideal of a finite set V, we have $|SM(I)| = |V|$ [11].

Gröbner basis computation is still an active area of research. The first algorithm for doing so is known as *Buchberger's* Algorithm, first introduced in 1965. Buchberger's Algorithm can be seen as a multivariate, non-linear generalization of both Euclid's algorithm for computing polynomial greatest common divisors, and Gaussian elimination for linear systems. Worst-case computational complexity for computing Gröbner bases is believed to be exponential [42], although several speedups and special cases exist. For instance, there exist fairly efficient algorithms when the ideal is zero dimensional [1], or when it is toric [56]. Newer Gröbner basis algorithms and speedups have been developed, as in [16, 17, 20], some of which are used by current computer algebra systems.

In addition to their utility in discrete modeling, Gröbner bases have various applications in computational algebra. For instance, they are used to solve multivariate systems of polynomial equations, to determine whether a polynomial belongs to a given ideal [47], to determine whether or not two sets of polynomials give rise to the same ideal, and in elimination theory, which we will introduce in the next section.

2.3 Toric Ideals and Toric Varieties

In Chapter 3, we will encounter a class of functions that form a so-called *toric ideal*. Toric ideals and their corresponding varieties (toric varieties) are well-studied structures in algebraic geometry with computationally desirable properties [12, 49].

Definition 11. *A prime ideal generated by binomials is called a* toric ideal.

Here are some examples of toric ideals:

Example 12. $\langle x^3 - y^2 \rangle \subseteq \mathbb{C}[x, y]$.

Example 13. $\langle xz - yw \rangle \subseteq \mathbb{C}[x, y, z, w]$.

Example 14. $\langle x_i x_{j+1} - x_{i+1} x_j : 1 \leq i < j \leq d-1 \rangle \subseteq \mathbb{C}[x_1, \ldots, x_d]$.

Since prime ideals are radical, the Ideal-Variety Correspondence tells us that varieties corresponding to a toric ideals are also toric. The geometry of a toric variety is fully determined by the combinatorics of its associated fan, which often makes computations far more tractable. For example, one can compute a Gröbner basis of an ideal of a toric variety by only looking at the corresponding integer lattice [56, 57].

2.4 Elimination Theory

An important application of Gröbner bases is in elimination theory. In commutative algebra and algebraic geometry, elimination theory is the classical name for algorithmic approaches to eliminating some variables between polynomials of several variables. One can think of an ideal as a collection of relations that variables must satisfy. An ideal is often is described by giving its generating set (or a basis), which is a subset of relations that imply all relations in the ideal. In practice, people are often interested in the relations among a specific subset of variables in the ideal. For example, computing the projection of a variety onto a subspace. However, these relations might not be given directly in the generating set. Hence, one might need to think of a smart way to compute these relations. In other words, we would like to "eliminate" other variables in the ideal and only focus on the variables that we are interested in.

Definition 15. *Let $I \subseteq \mathbb{F}[x_1, \ldots, x_n]$ be an ideal. The ith elimination ideal I_i of I is the intersection $I_i = I \cap \mathbb{F}[x_{i+1}, \ldots, x_n]$.*

As one can see from the definition, I_i encodes all the relations among variables $x_{i+1}, \ldots, x_n$. In order to describe I_i, one needs to compute a basis of I_i and this can be done efficiently using a Gröbner basis.

Theorem 16. *Let $I \subseteq \mathbb{F}[x_1, \ldots, x_n]$ be an ideal. Let G be a Gröbner basis for I for the lexicographic order $\prec$ with $x_n \prec \ldots \prec x_1$. Then $G \cap \mathbb{F}[x_{i+1}, \ldots, x_n]$ is a Gröbner basis for I_i (for the induced lexicographic monomial order).*

Example 17. *Let $I = \langle x - t^2 - 1, y - t^3 - t \rangle \subseteq \mathbb{F}[t, x, y]$. Note that*

$$t^2 - x + 1, \quad tx - y, \quad ty - x^x + x, \quad x^3 - x^2 - y^2$$

is a Gröbner basis for I with respect to lexicographic order $y \prec x \prec t$. Theorem 16 implies that $x^3 - x^2 - y^2$ generates the first elimination ideal.

Chapter 3

Canalization in Boolean networks

3.1 Introduction

The *phenotype* of an organism consists of its observable traits, such as eye-color, height, or wing type. One can also speak of phenotypes of a population or species, such as having tails, opposible thumbs, or body hair. The nature vs. nurture paradigm summarizes the two primary factors that determine phenotype, both at the individual and population level: (i) environment and (ii) genetic makeup, the latter of which is called *genotype*. On one hand, the phenotype of an organism (or population) must be robust enough to withstand changes to its environment and genotype. On the other hand, at the population level, it must be flexible enough to evolve and better adapt to these changes. Canalization is a measure of the stability of a phenotype with respect to outside changes.

The term "canalization" was coined by geneticist Conrad Hal Waddington in 1942 as an attempt to quantify the reduced sensitivity of a phenotype to genetic and environmental perturbations [63]. Over 30 years later in [31], Kauffman introduced the notion of canalizing Boolean functions, in order to accurately reflect the behavior of biological systems in the setting of Boolean network models. Thirty years after that, Kauffman and collaborators further

expanded the canalization concept and introduced the class of nested canalizing functions [32], which can be thought of as functions that are fully "recursively canalizing."

In the last decade, canalizing functions have been extensively studied by researchers in the fields of mathematics, biology, physics, computer science, and electrical engineering. For example, Shmulevich and Kauffman showed that canalizing functions have lower activities and sensitivities than random Boolean functions, and this causes Boolean network models using these functions to be more stable; see [55] and [33]. More work on the dynamical stability of canalizing Boolean networks was done in [43] and in [30], where the authors explored the relationship between the proportion of canalizing functions in a network, and whether it lies in the ordered or chaotic dynamical regime. The evolution of canalizing Boolean networks was studied in [58]. Fourier analysis has shown that canalizing Boolean networks maximize mutual information [36]. On the more mathematical side, an exact formula was derived for the number of Boolean canalizing functions in [28]. Canalizing functions have been generalized from Boolean to over general finite fields in [45].

Nested canalizing functions (NCFs) have also gained significant attention. In [48] and [29], the authors study the phase diagram of Boolean networks with NCFs. A recursive formula for the number of nested canalizing functions was derived in [27], where they were shown to be what the electrical engineering community calls unate cascade functions [6]. NCFs have been studied algebraically through the lens of toric varieties [25], and in [40], where the authors obtained a unique algebraic form by writing an NCF in extended monomial layers. This allowed the authors to enumerate the number of NCFs. It also provided the tools for the development of an algorithm in [22] to reverse engineer a nested canalizing Boolean network from partial data. In [39], the authors generalized the notion of both canalizing and nested canalizing functions by introducing the class of partially nested canalizing functions. Loosely speaking, these are the functions that are "somewhat recursively canalizing." The dynamics of Boolean networks built with these functions has been studied in [39] and [24].

3.2 Canalizing and Nested Canalizing Functions

To make this chapter self-contained we will restate some well-known definitions; see, e.g., [32]. This is also needed because there are slight variations in certain definitions throughout the literature. Let $\mathbb{F}_2 = \{0, 1\}$ be the binary field, and let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be an n-variable Boolean function.

Definition 18. *A Boolean function $f(x_1, \ldots, x_n)$ is* essential *in the variable x_i if there exists a sequence $a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n \in \mathbb{F}_2$ such that*

$$f(a_1, \ldots, a_{i-1}, 0, a_{i+1}, \ldots, a_n) \neq f(a_1, \ldots, a_{i-1}, 1, a_{i+1}, \ldots, a_n).$$

In this case, we say that x_i is an essential variable *of f. Variables that are non-essential are* fictitious.

S. Kauffman defined canalizing Boolean functions in [31] to capture the general stability of gene regulatory networks. In that paper, a Boolean function f is canalizing in variable x_i, with canalizing input a and canalized output b, if, whenever x_i takes on the value a, the output of f is b, regardless of the inputs of other variables. As a consequence, constant functions are trivially canalizing. We will soon see why it is more mathematically natural to exclude these functions, among others. This is done by the following small adjustment to the original definition that does not change the overall idea.

Definition 19. *A Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is* canalizing *if there exists a variable x_i, a Boolean function $g(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$, and $a, b \in \mathbb{F}_2$ such that*

$$f(x_1, \ldots, x_n) = \begin{cases} b & x_i = a, \\ g \not\equiv b & x_i \neq a. \end{cases}$$

In this case, x_i is a canalizing variable*, the input a is the* canalizing input*, and the output value b when $x_i = a$ is the corresponding* canalized output.

The only difference of our definition is the added restriction that g can not be the constant function b. In other words, *we require a canalizing function to be essential in its canalizing variable*. The original definition was motivated by the stability of canalizing functions while our definition tries to capture the dominance of the canalizing variable. At first glance, our additional restriction might seem artificial or insignificant. However, it is unequivocally more natural when considering the algebraic structure of Boolean functions, which is at the heart of the stratification derived in this chapter.

In Definition 19, when the canalizing variable does not receive its canalizing input a, the function g obtained by plugging in $x_i = \overline{a}$ can be an arbitrary Boolean function. To better model a dynamically stable network, in [32] Kauffman proposed that in this case, there should be another variable x_j that is canalizing for a particular input, and so on. This leads to the following definition, where σ is a total ordering, or permutation, of $\{1, \ldots, n\}$. We write this as $\sigma = \sigma(1), \sigma(2), \ldots \sigma(n)$, and say that $\sigma \in \mathfrak{S}_n$, the symmetric group on n letters.

Definition 20. *A Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is* nested canalizing *with respect to the permutation $\sigma \in \mathfrak{S}_n$, inputs a_i and outputs b_i, for $i = 1, 2, \ldots, n$, if it can be represented in the form:*

$$f(x_1, \ldots, x_n) = \begin{cases} b_1 & x_{\sigma(1)} = a_1, \\ b_2 & x_{\sigma(1)} \neq a_1, x_{\sigma(2)} = a_2, \\ b_3 & x_{\sigma(1)} \neq a_1, x_{\sigma(2)} \neq a_2, x_{\sigma(3)} = a_3, \\ \vdots & \vdots \\ b_n & x_{\sigma(1)} \neq a_1, \ldots, x_{\sigma(n-1)} \neq a_{n-1}, x_{\sigma(n)} = a_n, \\ \overline{b_n} & x_{\sigma(1)} \neq a_1, \ldots, x_{\sigma(n-1)} \neq a_{n-1}, x_{\sigma(n)} \neq a_n. \end{cases} \tag{3.1}$$

The idea of nested canalizing is in that some sense, it is "recursively canalizing" for exactly n steps. As an analogy, one can consider a nested canalizing function as an onion. We can peel off variables one at a time by not taking the canalizing input of each variable (i.e., by plugging in $x_i = \overline{a_i}$). Before we peel off the 'inner' variables, we need to peel off the 'outer' variables first. In the end, we are left with the constant function $\overline{b_n}$. We will return to this onion analogy several times throughout this paper to highlight our main ideas.

Remark 21. *Since $b_n \neq \overline{b_n}$, a nested canalizing function is essential in all n variables.*

If a Boolean function is nested canalizing, then at least one (of all $n!$) ordering of the variables yields an equation in the form of Eq. (3.1). Note that such variable orderings are not unique, and the number of such orderings depends on the function f. For example, we can write the function $f_1(x, y, z) = xyz$ as in Eq. (3.1) using any of the 6 orderings of the variables $\{x, y, z\}$. In contrast, for $f_2(x, y, z) = x(yz + 1)$, only 2 orderings would work, namely (x, y, z) and (x, z, y).

3.3 k-canalizing Functions

Nested canalizing functions have a very restrictive structure and become increasingly sparse as the number of input variables increases [27]. In a real network model, it is often the case that not all variables exhibit nested canalizing behavior. Moreover, the first several canalizing variables play more central roles than the remaining variables. Thus, it is natural to consider functions that are canalizing, but not nested canalizing. For example, one function in the segment polarity gene in by Albert and Olhmer's seminal paper [4] is canalizing but not nested canalizing. For another example, one can look at the lactose (*lac*) operon, which regulates the transport and metabolism of lactose in *Escherichia coli*. In [52], a simple Boolean network

model of the *lac* operon was proposed, where the regulatory function for lactose was

$$f_L(t+1) = \overline{G_e} \wedge [(L \wedge \overline{E}) \vee (L_e \wedge E)] .$$

In a sentence, this means "internal lactose (L) will be present the following timestep if there is no external glucose (G_e), *and* at least one of the following holds:

- there already internal lactose present, but the enzyme β-galactosidase (E) that breaks it down is absent;
- there is external lactose (L_e) available and the *lac* permease transporter protein (also represented by E since they are co-transcribed) are present.

The variable $\overline{G_e}$ (though sometimes considered a parameter) is canalizing because it acts as a "shut-down" switch: if $G_e = 1$, then $f_L = 0$ regardless of the other variables. In other words, we can write this as

$$f_L(G_e, L_e, L, E) = \begin{cases} 0 & G_e = 1, \\ (L \wedge \overline{E}) \vee (L_e \wedge E) & G_e \neq 0. \end{cases}$$

The function $g = (L_e \wedge E) \vee (L \wedge \overline{E})$ is not canalizing, and so the 5-variable function f_L is canalizing but not nested canalizing. In the framework that we are about to define, this function has canalizing depth 1.

Due to both theoretical and practical reasons, a relaxation of the nested canalizing structure is often necessary. This was done in [39], where there authors defined partially nested canalizing functions, and then distinguished between the "active depth" and "full depth" of a function. Our definition of k-canalizing functions is similar to what it means in their paper to be "partially nested canalizing of active depth at least k." As before, the small differences are motivated by the desire to have a natural unique algebraic form. For notational purpose, we first

introduce the notion of a k-permutation.

Definition 22. *Let $[n]$ denote the set $\{1, 2, \ldots, n\}$. A k-*permutation *σ is an injective map from $[k]$ to $[n]$. We use $\mathfrak{S}_{n,k}$ to denote the set of all k-permutations.*

A k-permutation is a way of selecting k distinct objects from a list of n, such that the order of selection matters. k-permutations are also known as partial permutations or as sequences without repetition. $P(n, k)$, the number of k-permutations of n objects, is $\frac{n!}{(n-k)!}$. Moreover, $\mathfrak{S}_{n,n} = \mathfrak{S}_n$, since an n-permutation is just a regular permutation.

Definition 23. *A Boolean function $f(x_1, \ldots, x_n)$ is k-*canalizing*, where $0 \leq k \leq n$, with respect to the k-permutation $\sigma \in \mathfrak{S}_{n,k}$, inputs a_i, and outputs b_i, for $1 \leq i \leq k$, if*

$$f(x_1, \ldots, x_n) = \begin{cases} b_1 & x_{\sigma(1)} = a_1, \\ b_2 & x_{\sigma(1)} \neq a_1, x_{\sigma(2)} = a_2, \\ b_3 & x_{\sigma(1)} \neq a_1, x_{\sigma(2)} \neq a_2, x_{\sigma(3)} = a_3, \\ \vdots & \vdots \\ b_k & x_{\sigma(1)} \neq a_1, \ldots, x_{\sigma(k-1)} \neq a_{k-1}, x_{\sigma(k)} = a_k, \\ g \not\equiv b_k & x_{\sigma(1)} \neq a_1, \ldots, x_{\sigma(k-1)} \neq a_{k-1}, x_{\sigma(k)} \neq a_k. \end{cases} \tag{3.2}$$

where $g = g(\{x_i : i \in [n] \backslash \sigma([k])\})$ is a Boolean function on $n - k$ variables. When g is not a canalizing function, the integer k is the canalizing depth *of f. Furthermore, if g is not a constant function, then we call it a* core function *of f, denoted by f_C.*

As with canalizing and nested canalizing functions, the $g \not\equiv b_k$ condition ensures that f is essential in the final variable, $x_{\sigma(k)}$.

Remark 24. *Since $g \not\equiv b_k$, a function f that is k-canalizing with respect to $\sigma \in \mathfrak{S}_{n,k}$, inputs a_i and outputs b_i is essential in each $x_{\sigma(i)}$ for $i = 1, \ldots, k$.*

The representation of a k-canalizing function f in the form of Eq. (3.2), even when k is the canalizing depth, is generally not unique since it depends on the variable ordering. However, we will prove that several key properties, such as the canalizing depth and core function $f_C = g$ (if there is one), are independent of representation. It is worth noting that if g is constant, then g need not be unique, i.e., both $g \equiv 0$ and $g \equiv 1$ can arise. This is why we do not allow constant core functions. The following observation is elementary.

Remark 25. *If f is k-canalizing with respect to $\sigma \in \mathfrak{S}_{n,k}$, inputs a_i and outputs b_i, then any initial segment $x_{\sigma(1)}, \ldots, x_{\sigma(j)}$ with the same canalized output $b_1 = \cdots = b_j$ can be permuted to yield an equivalent form as in Eq.* (3.2)*.*

Definition 26. *If $f(x_1, \ldots, x_n)$ is k-canalizing with respect to $\sigma \in \mathfrak{S}_{n,k}$, inputs a_i and outputs b_i, then for each $j \leq k$, define the Boolean function $g_j^\sigma(\{x_i : i \in [n] \backslash \sigma([j])\})$ to be the result of plugging in $x_{\sigma(i)} = \overline{a_i}$, for $i = 1, \ldots, j$.*

In plain English, the function g_j^σ is the result of when the first j canalizing variables do *not* get their canalizing inputs. We can now show that the canalizing depth k and the core function f_C are independent of the order of the variables. Moreover, the ambiguity of variable orderings is well-controlled in that they are partitioned into blocks called *layers* via extended monomials, and variables can be permuted arbitrarily if and only if they lie in the same layer. This generalizes the observation in Remark 25.

Proposition 27. *Suppose an n-variable Boolean function f is k-canalizing with respect to the k-permutation σ, inputs a_i and outputs b_i, for $1 \leq i \leq k$, and k'-canalizing with respect to the permutation σ', inputs a'_j and outputs b'_j, for $1 \leq j \leq k'$, such that both g and g', obtained by substituting $\overline{a_i}$ for $x_{\sigma(i)}$ and $\overline{a'_j}$ for $x_{\sigma'(j)}$ respectively, are not canalizing. Then $k = k'$ and the resulting core functions, if they exist, are the same.*

Proof. Assume f is canalizing, because otherwise, $k = k' = 0$ and the result is trivial. Without losing generality we can assume $\sigma(1) \neq \sigma'(1)$, since if this were not the case, we could simply

input $\overline{a_1} = \overline{a_1'}$ for $x_{\sigma(1)} = x_{\sigma'(1)}$ and consider $g_1^{\sigma} = g_1^{\sigma'}$. (Note that if $\sigma(1) = \sigma'(1)$ and $a_1 \neq a_1'$, then $b_1 \neq b_1'$, which means that f is completely determined by the input to $x_{\sigma(1)} = x_{\sigma'(1)}$. In this case, f has only one essential variable, and so $k = 1$. Moreover, both g_1^{σ} and $g_1^{\sigma'}$ are constant functions. Thus f has no core function.)

Since g is non-canalizing, it is not essential in $x_{\sigma(1)}$, and thus $\sigma(1) = \sigma'(j^*)$ for some $1 < j^* \leq k'$. We claim that we may assume without loss of generality that $a_{j^*}' = a_1$ and $b_{j^*}' = b_1$. To see why, first suppose that $a_{j^*}' = \overline{a_1}$ and consider the two possible inputs to $x_{\sigma'(j^*)} = x_{\sigma(1)}$ in the function $g_{j^*-1}^{\sigma'}$. If this variable takes its canalizing input $\overline{a_1}$, then the output is b_{j^*}'. However, since f is canalizing in $x_{\sigma'(j^*)} = x_{\sigma(1)}$, then the other input a_1 would yield the output b_1. In other words, $g_{j^*-1}^{\sigma'}$ is completely determined by the input to $x_{\sigma'(j^*)}$, so all subsequent variables are fictitious. Therefore, $g_{j^*}^{\sigma'} = g'$ must be constant, hence $j^* = k'$. Moreover, this function must be $g' \equiv b_1$ because it only arises when $x_{\sigma'(j^*)} = x_{\sigma(1)}$ takes the canalizing input a_1. Since f is essential in $x_{\sigma'(j^*)} = x_{\sigma(1)}$, then Remark 24 implies that $b_{j^*}' = \overline{b_1}$, the opposite value of $g' \equiv b_1$. Thus, we have two equivalent ways to represent $g_{j^*-1}^{\sigma'} = g_{k'-1}^{\sigma'}$:

$$g_{k'-1}^{\sigma'} = \begin{cases} \overline{b_1} & x_{\sigma'(k')} = \overline{a_1}, \\ g' \equiv b_1 & x_{\sigma'(k')} = a_1. \end{cases} = \begin{cases} b_1 & x_{\sigma'(k')} = a_1, \\ g' \equiv \overline{b_1} & x_{\sigma'(k')} = \overline{a_1}. \end{cases} \tag{3.3}$$

In other words, switching the triple of values $(a_{k'}', b_{k'}', g')$ from $(\overline{a_1}, \overline{b_1}, b_1)$ to $(a_1, b_1, \overline{b_1})$ in the original representation of f with respect to $\sigma' \in \mathfrak{S}_n$ does not change the function, so we may assume that $a_{j^*}' = a_1$ and $b_{j^*}' = b_1$, as claimed. The proof for the case when $b_{j^*}' = \overline{b_1}$ is almost the same.

Since f is canalizing in $x_{\sigma'(j^*)} = x_{\sigma(1)}$ with input a_1 and output b_1, we must also have $b_j' = b_1$ for all $1 \leq j \leq j^*$. By Remark 25, we can create a new permutation σ'' by swapping the order of $x_{\sigma'(1)}$ and $x_{\sigma'(j^*)}$ in σ'. Clearly, f is k'-canalizing with respect to σ'' and $g_{k'}^{\sigma'} = g_{k'}^{\sigma''}$. Since $x_{\sigma(1)} = x_{\sigma''(1)}$, the result follows from induction on $g_1^{\sigma} = g_1^{\sigma''}$. We conclude that $k = k'$.

Finally, we need to show that when f has a core function f_C, it is unique. The non-canalizing functions g and g' are essential in the same set of variables. If they are both constant functions, then they actually need not be the same, due to the different ways to write g' as in Eq. (3.3). Otherwise, they are core functions for f, and are obtained by substituting the same set inputs for the same set of variables, thus we must have $f_C = g = g'$. □

It is worth noting that Definition 23 is similar to the definition of k-partially nested canalizing functions (k-PNCFs) in [39]. In fact, these two definitions hold the same motivation but are from different perspectives. In [39], the authors treat k-PNCFs as a subclass of Boolean functions. While we prefer to consider canalization as a property of Boolean functions and different functions have different extent of canalization. This provides us a well-defined way to classify all Boolean functions on n variables.

Returning to our onion analogy, now we can think of all Boolean functions as onions. For each Boolean function, we can try to peel off its variables as we did for nested canalizing functions. We will have to stop once we get to a non-canalizing function. In this sense, nested canalizing functions would be the 'best' onions since we can peel off all the variables and non-canalizing would be the 'worst'. The k-canalizing functions would be those for which one can be peeled off at least k variables. Though a unique core function $f_C = g$ only exists when g is non-constant, we will soon see how every Boolean function, whether or not it has a core function, has a unique *core polynomial* that extends the notion of a core function.

Example 28. *The Boolean function $f(x, y, z, w) = xy(z + w)$ has canalizing depth 2 and core function $f_C = z + w$.*

Remark 29. *In our framework, if we consider the set of all Boolean functions on n variables, then:*

- *The canalizing depth of a k-canalizing function is at least k.*

- *A non-canalizing function has canalizing depth* 0 *and its core function is itself.*
- *Every Boolean function is* 0*-canalizing.*
- *The* 1*-canalizing functions are precisely the canalizing functions.*
- *The* n*-canalizing functions are precisely the nested canalizing functions.*
- *If a function* f *has canalizing depth* k *and a constant core function, then* f *has* $n-k$ *fictitious variables, and is a nested canalizing function on its* k *essential variables.*

3.4 Characterizations of k-canalizing Functions

3.4.1 Polynomial Form of k-canalizing Functions

It is well-known [41] that any Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2^n$ can be uniquely expressed as a square-free polynomial, called its *algebraic normal form*. Equivalently, the set of Boolean functions on n variables is isomorphic to the quotient ring $R := \mathbb{F}_2[x_1, \ldots, x_n]/I$, where $I = \langle x_i^2 - x_i : 1 \leq i \leq n \rangle$. Henceforth in this section, when we speak of Boolean polynomials, we assume they are square-free. Additionally, we define $\hat{x}_i := (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$ for notational convenience.

Lemma 30. *A Boolean function* $f(x_1, \ldots, x_n)$ *is canalizing in variable* x_i*, for some* $1 \leq i \leq n$*, with input* a_i *and output* b_i*, if and only if*

$$f = (x_i + a_i)g(\hat{x}_i) + b_i\,,$$

for some polynomial $g \not\equiv 0$*.*

Proof. Suppose f is canalizing in x_i. Written in its algebraic normal form, f can be viewed as an element of the Euclidean domain $\mathbb{F}_2[x_1, \ldots, x_n]$. By the Euclidean algorithm, we can factor

it as

$$f = x_i\, q(\hat{x}_i) + r(\hat{x}_i)\,,$$

where q and r are the quotient and remainder of f when divided by x_i. Note that $b_i = a_i q(\hat{x}_i) + r(\hat{x}_i)$, and since $a_i + a_i = 0$ in $\mathbb{F}_2$,

$$f = (x_i + a_i)q(\hat{x}_i) + \big[r(\hat{x}_i) + a_i q(\hat{x}_i)\big] = (x_i + a_i)q(\hat{x}_i) + b_i\,.$$

The function $g(\hat{x}_i) := q(\hat{x}_i)$ is nonzero because f is essential in x_i. This establishes necessity, and sufficiency is obvious. □

By applying the above lemma recursively, we get the following theorem.

Theorem 31. *A Boolean function $f(x_1, \ldots, x_n)$ is k-canalizing, with respect to the k-permutation $\sigma \in \mathfrak{S}_{n,k}$, inputs a_i and outputs b_i, for $1 \leq i \leq k$, if and only if it has the polynomial form*

$$f(x_1, \ldots, x_n) = (x_{\sigma(1)} + a_1)g(\hat{x}_i) + b_1\,, \tag{3.4}$$

where

$$g(\hat{x}_i) = (x_{\sigma(2)} + a_2)\Big[\ldots\big[(x_{\sigma(k-1)} + a_{k-1})[(x_{\sigma(k)} + a_k)\bar{g} + \Delta b_{k-1}] + \Delta b_{k-2}\big]\ldots\Big] + \Delta b_1$$

for some polynomial $\bar{g} = \bar{g}(x_{\sigma(k+1)}, \ldots, x_{\sigma(n)}) \not\equiv 0$, where $\Delta b_i := b_{i+1} - b_i = b_{i+1} + b_i$, or equivalently,

$$f(x_1, \ldots, x_n) = \bar{g}\prod_{i=1}^{k}(x_{\sigma(i)} - a_i) + \sum_{j=1}^{k-1}\Delta b_{k-j}\prod_{i=1}^{k-j}(x_{\sigma(i)} - a_i) + b_1.$$

3.4.2 Dominance Layer and Extended Monomial Form of k-canalizing Functions

One weakness of Theorem 31 is that given a Boolean function f, the representation of f into the above form, even when k is exactly the canalizing depth, is not unique. For example, $f = x(y+1)(z+w)$ can be also written as $f = (y+1)x(z+w)$. In this example, x and y have bigger impact the z and w. In a k-canalizing function, some variables are "more dominant" than others. We will classify all variables of a Boolean function into different layers according to the extent of their dominance, extending work from [40] from NCFs to general Boolean functions. The "most dominant" variables will be precisely those that are canalizing. Recall that we are always working in the quotient ring $R = \mathbb{F}_2[x_1, \ldots, x_n]/I$, though at times it is helpful to consider the algebraic normal form of a polynomial as an element of $\mathbb{F}_2[x_1, \ldots, x_n]$.

Definition 32. *A Boolean function $M(x_1, \ldots, x_m)$ is an* extended monomial *in variables $x_1, \ldots, x_m$ if*

$$M(x_1, \ldots, x_m) = \prod_{i=1}^{m} (x_i + a_i),$$

where $a_i \in \mathbb{F}_2$ for each $i = 1, \ldots, m$.

An extended monomial in R is an extended monomial of a subset of $\{x_1, \ldots, x_n\}$. In other words, it is simply a product $\prod_{i=1}^{n} y_i$, where each y_i is either x_i, $\overline{x_i}$, or 1. Using extended monomials, we can refine Theorem 31 to obtain a unique *extended monomial form* of any Boolean function.

Proposition 33. *Given a Boolean function $f(x_1, \ldots, x_n)$, all variables are canalizing if and only if $f = M(x_1, \ldots, x_n) + b$, where M is an extended monomial in all variables.*

Proof. Suppose all n variables are canalizing in f, and so f is essential in every variable. Since x_1 is canalizing, Lemma 30 says that $f = (x_1 + a_1)g(\hat{x}_1) + b$ for some $a_1, b \in \mathbb{F}_2$, and $g \not\equiv 0$. In particular, this means that $(x_1 + a_1) \mid (f + b)$ in $\mathbb{F}_2[x_1, \ldots, x_n]$. Since x_2 is also canalizing,

$f(x_1, a_2, \ldots, x_n) \equiv b'$ for some a_2 and b'. Plugging in $x_1 = a_1$ yields $f(a_1, a_2, x_3, \ldots, x_n) \equiv b = b'$, and so

$$(x_2 + a_2) \mid (f + b) = (x_1 + a_1)g(x_2, \ldots, x_n)\,.$$

Since $x_1 + a_1$ and $x_2 + a_2$ are co-prime, we get $(x_2 + a_2) \mid g(x_2, \ldots, x_n)$. Note that $g(\hat{x}_1) \not\equiv 0$, hence, we have $g(\hat{x}_1) = (x_2 + a_2)g'(x_3, \ldots, x_n)$ where $g'(x_3, \ldots, x_n) \not\equiv 0$. Thus we have $f = (x_1 + a_1)(x_2 + a_2)g'(x_3, \ldots, x_n) + b$. Necessity of the proposition now follows from induction, and sufficiency is obvious. □

We are now ready to prove the main result of this section. This is a generalized version of Theorem 4.2 in [40]. We will obtain a new *extended monomial form* of a Boolean function f by induction. In this form, all variables will be classified into different layers according to their dominance. The canalizing variables are the *most dominant* variables. Thus, a Boolean function may have one, none, or many "most dominant" variables. As in [40], variables in the same layer will have the same level of dominance, with the variables in the outer layers being "more dominant" than those in the inner layers.

Theorem 34. *Every Boolean function $f(x_1, \ldots, x_n) \not\equiv 0$ can be uniquely written as*

$$f(x_1, \ldots, x_n) = M_1(M_2(\cdots(M_{r-1}(M_r p_C + 1) + 1)\cdots) + 1) + b, \tag{3.5}$$

where each $M_i = \prod_{j=1}^{k_i}(x_{i_j} + a_{i_j})$ is a nonconstant extended monomial, $p_C \not\equiv 0$ is the core polynomial *of f, and $k = \sum k_i$ is the canalizing depth. Each x_i appears in exactly one of $\{M_1, \ldots, M_r, p_C\}$, and the only restrictions on Eq.* (3.5) *are the following "exceptional cases":*

(i) If $p_C \equiv 1$ and $r \neq 1$, then $k_r \geq 2$;

(ii) If $p_C \equiv 1$ and $r = 1$ and $k_1 = 1$, then $b = 0$;

When f is a non-canalizing function, we simply have $p_C = f$.

Before we prove Theorem 34, we will define some terms and examine a few details, such as the subtle difference between the core function and core polynomial, and the "exceptional cases", by simple examples. This should help elucidate the more technical parts of the proof.

Definition 35. *A Boolean function f written in its unique form from Eq.* (3.5) *is said to be in* standard monomial form*, and r is its* layer number*. The i^{th}* dominance layer *of f, denoted L_i, is the set of essential variables of M_i. The set of essential variables of p_C is denoted L_∞, and these are called the* recessive variables *of f.*

As we will see, when f has a core function f_C, its core polynomial is either $p_C = f_C$ or $p_C = f_C + 1$. When the number of "+1"s that appear in Eq. (3.5), possibly including b, is even, we have $p_C = f_C$. Otherwise, we have $p_C = f_C + 1$. When a Boolean function f with canalizing depth $k > 0$ fails to have a core function, in other words, f is in fact a nested canalizing function on k variables, with $n - k$ fictitious variables then its core polynomial is simply $p_C = 1$.

Finally, we will examine the two "exceptional cases". Both of these are necessary to avoid double-counting certain functions and ensure uniqueness, as claimed in Theorem 34.

(i) *If $p_C \equiv 1$ and $r \neq 1$.* In this case, if $k_r = 1$, that is $M_r = x_i$ or $\overline{x_i}$, for some i. In either case, this innermost layer can be "absorbed" into the extended monomial M_{r-1}. For example, if $M_r = x_i$, then the inner two layers are

$$M_{r-1}(M_r + 1) + 1 = M_{r-1}(x_i + 1) + 1 = (x_i + 1) \prod_{j=1}^{k_{r-1}} (x_{i_j} + a_{i_j}) + 1, = \hat{M}_{r-1} + 1\,,$$

where $\hat{M}_{r-1} = \overline{x_i} M_{r-1}$ is an extended monomial. Thus, in this case we may assume that the innermost layer has at least two essential variables, hence $k_r \geq 2$.

(ii) If $p_C \equiv 1$ and $r = 1$ and $k_1 = 1$, then for some i, either $f = x_i + b$, or $f = \overline{x_i} + b$. Clearly, there are only two such functions, either $f = x_i$ or $f = \overline{x_i}$, and so allowing both $b = 0$ and $b = 1$ would double-count these. Thus, we may assume that $b = 0$.

Proof of Theorem 34. For any non-canalizing function $f \not\equiv 0$, $f = p_C$ and the uniqueness is obvious.

When f is canalizing, we induct on n. When $n = 1$, there are 2 canalizing functions, namely $x = (x)1$ and $x+1 = (x+1)1$, both satisfying Eq. (3.5). For the these 2 functions, since $p_C \equiv 1$, $r = 1$ and $k_1 = 1$, we must have $b = 0$, so the previous representation is also unique.

When $n = 2$, there are 12 canalizing functions, 4 of which are essential in 1 variable, and thus can be uniquely written as in Eq. (3.5). Now let us consider the 8 canalizing functions that are essential in 2 variables. It is easy to check for all these, both variables x_1 and x_2 are canalizing. Then by Proposition 33, all of them are of the form

$$(x_1 + a_1)(x_2 + a_2) + b = M_1 p_C + b\,,$$

where $M_1 = (x_1 + a_1)(x_2 + a_2)$ and $p_C \equiv 1$. In this case, we have $r = 1$ and $k_1 = 2$. Note that when $p_C \equiv 1$, the innermost layer must have at least two essential variables, so uniqueness holds. We have proved that Eq. (3.5) holds for $n = 1$ and $n = 2$.

Assume now that Eq. (3.5) is true for any canalizing function that is essential in at most $n - 1$ variables. Consider a canalizing function $f(x_1, \ldots, x_n)$. Suppose that x_{1_j} for each $j = 1, \ldots, k_1$ are all canalizing in f. With the same argument as in Proposition 33, we get $f = M_1 g + b$, where $M_1 = (x_{1_1} + a_{1_1}) \cdots (x_{1_{k_1}} + a_{1_{k_1}})$ and $g \not\equiv 0$. If g is non-canalizing, then Eq. (3.5) holds with $p_C = g$ and $r = 1$. If g is canalizing, then it is a canalizing function that is essential in at most $n - k_1 < n - 1$ variables. By our induction hypothesis, it can be uniquely written as

$$g = M_2(M_3(\cdots(M_{r-1}(M_r p_C + 1) + 1)\cdots) + 1) + b'\,.$$

Note that b' must be 1, otherwise all variables in M_2 will also be most dominant variables of f. This completes the proof. □

Remark 36. *For any Boolean function f:*

(i) Variables in two consecutive layers have different canalized outputs.

(ii) L_1 *consists of all the most dominant variables (canalizing variables) of* f*.*

Let us return to our onion analogy, where we previously were peeling off one variable at a time. Furthermore, imagine that each individual variable layer is white if the canalized output $a_i = 0$, and black if $a_i = 1$. Thus, we can think of an extended monomial layer L_i as a maximal block of variable layers of the same color. We can "peel off" an entire L_i at once by plugging in the non-canalizing input $x_{i_j} = \overline{a_{i_j}}$ for each variable in L_i. In other words, we can peel off all black layers, then all white layers, then all black layers, and so on. Moreover, we can read off the colors directly off of the function if it is written in the form of Eq. (3.2). However, recall that this form of a k-canalizing function, where g is non-canalizing, is not unique. By Theorem 34, the order of consecutive variables, $x_{\sigma(i)}$ and $s_{\sigma(i+1)}$, can be transposed if and only if they are in the same L_i. Based on this property, we can enumerate Boolean functions on n variables with canalizing depth k. Roughly speaking, we will do this by counting the number of different layer structures, and then counting the number of (non-canalizing) core functions. This last set is just the complement of the set of canalizing functions on those variables, which were enumerated in [28].

3.5 Enumeration of Boolean Functions by Canalizing Depth

Let $B(n, k)$ be the number of Boolean functions on n variables with canalizing depth exactly k. Exact formulas are known for $B(n, k)$ in a few special cases. The number of nested canalizing functions is $B(n, n)$. A recurrence for this was independently derived in the 1970s by engineers studying unate cascade functions [6, 54], and then a closed formula was found by mathematicians studying NCFs [40]. The quantity $B(4, k)$ was recently computed in [51]. In this section, we will present a general formula for $B(n, k)$.

Theorem 34 indicates that we can construct a Boolean functions with canalizing depth k by adding layers to a non-canalizing function on $n-k$ variables. Moreover, the complement of the set of non-canalizing functions are the canalizing functions. Hence, let us begin with a formula for C_n, the number of canalizing functions on n variables. This result was derived in [28] using a probabilistic method. We will include an alternative combinatorial proof using the *truth table* of a Boolean function f. This is the length-2^n vector $(f(x_i))_i$, given some fixed ordering $x_1, x_2, \ldots, x_{2^n}$ of the elements of $\mathbb{F}_2^n$.

Lemma 37. *The number C_n of canalizing Boolean functions on $n \geq 0$ variables, is given by:*

$$C_n = 2((-1)^n - n - 1) + \sum_{k=1}^{n} (-1)^{k+1} \binom{n}{k} 2^{k+1} 2^{2^{n-k}}.$$

Proof. We wish to count the number of Boolean functions that are canalizing in at least 1 variable. We can construct a truth table of a Boolean function that is canalizing in at least k variables by doing the following. First, pick k variables to be canalizing; there are $\binom{n}{k}$ ways to do this. Next, pick the canalizing input for each canalizing variable; there are 2^k ways to do that. Then, fill out the entries in the truth table of these canalizing inputs with the same canalized output; there are 2 ways to do that. The remaining table has 2^{n-k} entries, so there are $2^{2^{n-k}} - 1$ ways to fill it out such that the corresponding function is non-constant. By Inclusion-Exclusion, we have $\sum_{k=1}^{n} (-1)^{k+1} \binom{n}{k} 2^{k+1} (2^{2^{n-k}} - 1)$. Note that in this process, there are $2n$ functions of the form $x_i + a_i$, each being counted exactly twice, since we can pick either input as canalizing input. Therefore we have

$$C_n = \sum_{k=1}^{n} (-1)^{k+1} \binom{n}{k} 2^{k+1} (2^{2^{n-k}} - 1) - 2n = 2((-1)^n - n - 1) + \sum_{k=1}^{n} (-1)^{k+1} \binom{n}{k} 2^{k+1} 2^{2^{n-k}}.$$

□

As examples, one can check that $C_0 = 0$, $C_1 = 2$, $C_2 = 12$, $C_3 = 118$, $C_4 = 3512$, ...

This is consistent with the results in [28], though it should be noted that all numbers differ by 2 because we do not consider the constant functions to be canalizing.

Recall that there are 2^{2^n} Boolean functions on n variables. Since non-canalizing functions are complement of canalizing functions, the following is immediate.

Corollary 38. *The number $B^*(n,0)$ of non-constant core polynomials on n variables is*

$$B^*(n,0) = B(n,0) - 2 = (2^{2^n} - C_n) - 2 = 2^{2^n} - 2((-1)^n - n) + \sum_{k=1}^{n} (-1)^k \binom{n}{k} 2^{k+1} 2^{2^{n-k}}.$$

One can check that $B^*(n,0) = 0, 0, 2, 136$, for $n = 0, 1, 2, 3, \ldots$.

Before we derive the general formula for $B(n,k)$, let us first look at the special case when $k = n$. This was computed in [40], but we include a self-contained proof. Recall that a *composition of* n is a sequence $k_1, \ldots, k_r$ of non-empty integers such that $k_1 + \cdots + k_r = n$. By Theorem 34, the standard monomial form of a Boolean function with canalizing depth k involves a size-r composition of k with the additional property that $k_r \geq 2$.

Lemma 39. *For $n \geq 2$, the number $B(n,n)$ of nested canalizing functions on n variables is given by:*

$$B(n,n) = 2^{n+1} \sum_{r=1}^{n-1} \sum_{\substack{k_1+\ldots+k_r=n \\ k_i \geq 1,\, k_r \geq 2}} \binom{n}{k_1, \ldots, k_r},$$

where $\binom{n}{k_1,\ldots,k_r} = \frac{n!}{k_1! k_2! \ldots k_r!}$.

Proof. If a Boolean function is nested canalizing in n variables, then by Theorem 34, we know its core polynomial must be $B = 1$. Let us first fix the layer number r. Then for each choice of $k_1, \ldots, k_r$, with $k_1 + \ldots + k_r = n$, $k_i \geq 1$ and $k_r \geq 2$, there are $\binom{n}{k_1,\ldots,k_r}$ different ways to assign n variables to these r layers. For each variable x_j, we can pick either x_j or $x_j + 1$ to be in its corresponding extended monomial. Note that we also have 2 choices for b. So the number of

nested canalizing functions on n variables with exactly r layers is given by:

$$2^{n+1} \sum_{\substack{k_1+\ldots+k_r=n \\ k_i\geq 1,\, k_r\geq 2}} \binom{n}{k_1,\ldots,k_r}.$$

Then by summing over all possible layer numbers r, $1 \leq r \leq n-1$, we get the formula for $B(n,n)$. □

According to our definition, $B(1,1) = 2$. As example, one also can check $B(2,2) = 8$, $B(3,3) = 64$, $B(4,4) = 736$, ...

Now we are ready to derive the general formula for $B(n,k)$.

Theorem 40. *The number $B(n,k)$ of Boolean functions on n variables with canalizing depth k, for $1 \leq k \leq n$, is*

$$B(n,k) = \binom{n}{k}\left[B(k,k) + B^*(n-k,0)\cdot 2^{k+1}\sum \binom{k}{k_1,\ldots,k_r}\right],$$

where the sum is taken over all compositions of k, and the closed from of $B(k,k)$ is given by Lemma 39.

Proof. We can construct a Boolean function f on n variables with canalizing depth k by doing the following. First, pick k variables that are not in the core polynomial p_C. There are $\binom{n}{k}$ different ways to do that. Once we fixed the variables that are not in p_C, we need to consider the following two cases:

Case 1: $p_C \equiv 1$. Then f is actually a nested canalizing function on these k variables. There are $B(k,k)$ of them in total.

Case 2: $p_C \not\equiv 1$. Then p_C is a non-constant core polynomial on $n-k$ variables, so there

are $B^*(n-k,0)$ different choices for p_C. Using the same argument as in Lemma 39, there are

$$2^k \sum \binom{k}{k_1, \ldots, k_r}$$

different ways for those k variables to form the extended monomials in Eq. 3.5, where the sum is taken over all compositions of k. Note that we also have 2 ways to pick b. Therefore, in this case, there are

$$B^*(n-k,0) \cdot 2^{k+1} \sum \binom{k}{k_1, \ldots, k_r}$$

different Boolean functions.

By combining the above two cases, we get the formula for $B(n,k)$. □

Example 41. *As previously mentioned, the quantities $B(4,k)$ for $k = 0, \ldots, 4$ were computed in [51]. It is easy to check that these values are consistent with our general formula. There are $2^{2^4} = 65536$ Boolean functions on 4 variables. The number of functions with canalizing depth exactly k, for $k = 1, 2, 3, 4$ is*

$$\begin{aligned}
B(4,4) &= \binom{4}{4}(736+0) = 736, \\
B(4,3) &= \binom{4}{3}(64+0) = 256, \\
B(4,2) &= \binom{4}{2}(8+2\cdot 8\cdot 3) = 336, \\
B(4,1) &= \binom{4}{1}(2+136\cdot 4\cdot 1) = 2184.
\end{aligned}$$

Summing these yields the total number of canalizing functions on 4 variables,

$$C_4 = B(4,4) + B(4,3) + B(4,2) + B(4,1) = 736 + 256 + 336 + 2184 = 3512.$$

Thus, there are $B(4,0) = 65536 - 3512 = 62024$ non-canalizing functions on four variables,

including the two constant functions.

Note that k-canalizing functions are simply Boolean functions with depth at least k, therefore we immediately get the following equality.

Corollary 42. *The number of k_0-canalizing Boolean functions on n variables, $1 \leq k_0 \leq n$, is given by:*

$$\sum_{k=k_0}^{n} B(n,k) = \sum_{k=k_0}^{n} \binom{n}{k} \left[B(k,k) + B^*(n-k,0) \cdot 2^{k+1} \sum \binom{k}{k_1, \ldots, k_r} \right].$$

In particular, the canalizing functions are counted by the following identity:

$$C_n = \sum_{k=1}^{n} B(n,k) = \sum_{k=1}^{n} \binom{n}{k} \left[B(k,k) + B^*(n-k,0) \cdot 2^{k+1} \sum \binom{k}{k_1, \ldots, k_r} \right].$$

3.6 k-canalizing Functions in $\mathbb{F}_2^{2^n}$

Recall that the ring of Boolean functions is isomorphic to the quotient ring $R = \mathbb{F}_2[x_1, \ldots, x_n]/I$, where $I = \langle x_i^2 - x_i : 1 \leq i \leq n \rangle$. Indexing monomials $X_S := \prod_{i \in S} x_i$ by the subsets of $S \subseteq [n] := \{1, ..., n\}$ corresponding to the variables appearing in the monomial, we can write the elements of R as a square-free polynomial

$$R = \left\{ \sum_{S \subseteq [n]} c_S X_S : c_S \in \mathbb{F}_2 \right\}.$$

As a vector space over $\mathbb{F}_2$, R is isomorphic to $\mathbb{F}_2^{2^n}$ via the following correspondence

$$R \ni \sum_{S \subseteq [n]} c_S X_S \longleftrightarrow (c_\emptyset, \ldots, c_{[n]}) \in \mathbb{F}_2^{2^n}.$$

Based on this correspondence, we can associate any Boolean function with a point in $\mathbb{F}_2^{2^n}$. Moreover, given any subclass of Boolean functions, it is natural to ask for the corresponding subset in $\mathbb{F}_2^{2^n}$. In particular, we are going to discuss the subset $V^{k\text{-canalizing}} \subseteq \mathbb{F}_2^{2^n}$ corresponding to k-canalizing function in this section. Since $V^{k\text{-canalizing}}$ is in fact an algebraic variety, we will also come up with an algebraic parametrization of $V^{k\text{-canalizing}}$, namely, a collection of polynomial equations that encode the relations of the coefficients of k-canalizing functions. This parametrization describes the entire space of k-canalizing functions as a geometric object, whose properties can then be studied with the tools of algebraic geometry.

Definition 43. *Let $S \subseteq [n]$ be an non-empty subset whose largest element is r_S. The* completion *of S, denoted by $[r_S]$, is the set $[r_S] := \{1, \ldots, r_S\}$. For $S = \emptyset$, let $[r_\emptyset] = \emptyset$.*

Example 44.

$$[r_{\{1,2,5\}}] = [r_{\{5\}}] = [r_{\{3,5\}}] = \{1, 2, 3, 4, 5\}.$$

In [27], the authors derive the algebraic parametrization for the set of NCFs.

Theorem 45. *Let f be a Boolean polynomial in n variables, given by*

$$f = \sum_{S \subseteq [n]} c_S X_S.$$

The polynomial f is an NCF in the order $x_1, x_2, \ldots, x_n$ if and only if: $c_{[n]} = 1$, and for any subset $S \subseteq [n]$,

$$c_S = c_{[r_S]} \prod_{i \in [r_S] \setminus S} c_{[n]\setminus\{i\}}.$$

The basic idea behind Theorem 45 is actually pretty simple: if the monomial X_S appears in a nested canalizing polynomial f, then there must be "1" in the corresponding places if we write f in polynomial form Eq.(3.4).

Example 46.

$$f = x_1x_2x_3 + x_1x_2 + x_2x_3 + x_1 + x_2 + 1 = (x_1 + \underline{1})[x_2(x_3 + \underline{1}) + 1], \tag{3.6}$$

Let us look at $S = \{2\}$*, we have:*

$$c_{\{2\}} = c_{\{1,2\}} \cdot c_{\{2,3\}}.$$

The monomial $X_{\{2\}} = x_2$ *appears in* f *since* 1 *appears in all of the underlined entries in Eq.*(3.6) *if we write* f *in polynomial form of Eq.*(3.4)*. The* 1 *on the left corresponds to* $c_{\{2,3\}} = 1$ *and the* 1 *on the right corresponds to* $c_{\{1,2\}}$.

Theorem 47. *Let* f *be a Boolean polynomial in* n *variables, given by*

$$f = \sum_{S \subseteq [n]} c_S X_S.$$

The polynomial f *is a* k*-canalizing function in the order* $x_1, x_2, \ldots, x_k$*, if and only if there exists* $M \neq \emptyset \subseteq [n]\backslash[k]$*, such that* $c_{M\cup[k]} = 1$*, and for any subset* $S \subseteq [n]$*,*

$$c_S = \begin{cases} c_{[r_S]} \prod_{i \in [r_S]\backslash S} c_{M\cup[k]\backslash\{i\}} & S \subseteq [k] \\ c_{[k]\cup S} \prod_{i \in [k]\backslash S} c_{M\cup[k]\backslash\{i\}} & S \nsubseteq [k], \end{cases}$$

or, for all $M \neq \emptyset \subseteq [n]\backslash[k]$*,* $c_{M\cup[k]} = 0$*, and* $c_{[k]} = 1$*, then for any subset* $S \subseteq [n]$*,*

$$c_S = \begin{cases} \prod_{i \in [r_S]\backslash S} c_{[k]\backslash\{i\}} & S \subseteq [k] \\ 0 & S \nsubseteq [k] \end{cases}$$

In the latter case, the function is an NCF on k *variables, and fictitious on the other* $n - k$

variables.

Proof. First assume that the polynomial f is an n-variable Boolean k-canalizing function in the order $x_1, \ldots, x_k$, with canalizing input values a_i and corresponding canalized output values b_i, $1 \leq i \leq k$, and core polynomial $\bar{g} = \bar{g}(x_{k+1}, \ldots, x_n) \not\equiv 0$. Then by Theorem 31, f has the form:

$$f(x_1, \ldots, x_n) = \bar{g} \prod_{i=1}^{k} (x_i - a_i) + \sum_{j=1}^{k-1} \Delta b_{k-j} \prod_{i=1}^{k-j} (x_i - a_i) + b_1,$$

which can be expanded as

$$f(x_1, \ldots, x_n) = \bar{g} \sum_{T \subseteq [k]} X_T \prod_{\ell \in [k] \setminus T} a_\ell + \sum_{j=1}^{k-1} \Delta b_{k-j} \left(\sum_{T \subseteq [k-j]} X_T \prod_{\ell \in [k-j] \setminus T} a_\ell \right) + b_1. \tag{3.7}$$

Let us first consider the case when $\bar{g} \neq 1$. In this case, there exists $M \neq \emptyset \subseteq [n] \setminus [k]$, such that X_M appears in $\bar{g}$, and hence, $c_{M \cup [k]} = 1$. Next, consider subscripts of the form $S = M \cup [k] \setminus \{i\}$, $1 \leq i \leq k$. It is clear from Eq.(3.7) that X_M only appears in first part of the summand and hence, for $1 \leq i \leq k$,

$$c_{M \cup [k] \setminus \{i\}} = a_i = c_{M \cup [k]} c_{M \cup [k] \setminus \{i\}}.$$

Now, let us consider the subscripts of the form $S \not\subseteq [k]$. From Eq.(3.7) we can see that X_S could only show up in the first part of the summand. It would only appear if $X_{S \setminus [k]}$ appears in $\bar{g}$ and $\prod_{i \in [k] \setminus S} a_i = 1$. In other words, for all $S \not\subseteq [k]$, we must have

$$c_S = c_{[k] \cup S} \prod_{i \in [k] \setminus S} c_{M \cup [k] \setminus \{i\}}.$$

It is easy for us to check for any set $S \subseteq [k]$ such that $[r_S] = S$, we have

$$\begin{aligned} c_S = c_{[r_S]} &= c_{[k]} \prod_{i\in[k]\setminus S} a_i + \Delta b_{k-1} \prod_{i\in[k-1]\setminus S} a_i + \cdots + \Delta b_{r_S} \prod_{i\in[r_S]\setminus S} a_i \\ &= c_{[k]} \prod_{i\in[k]\setminus S} a_i + \Delta b_{k-1} \prod_{i\in[k-1]\setminus S} a_i + \cdots + \Delta b_{r_S}, \end{aligned}$$

since $\prod_{i\in[r_S]\setminus S} a_i = \prod_{i\in\emptyset} a_i = 1$. Now, let $S \subseteq [k]$ be any nonempty subset of $[k]$. By equating the coefficients of X_S, we have

$$\begin{aligned} c_S &= c_{[k]} \prod_{i\in[k]\setminus S} a_i + \Delta b_{k-1} \prod_{i\in[k-1]\setminus S} a_i + \cdots + \Delta b_{r_S} \prod_{i\in[r_S]\setminus S} a_i \\ &= \prod_{i\in[r_S]\setminus S} a_i \left(c_{[k]} \prod_{i\in[k]\setminus[r_S]} a_i + \Delta b_{k-1} \prod_{i\in[k-1]\setminus[r_S]} a_i + \cdots + \Delta b_{r_S} \right) \\ &= (\prod_{i\in[r_S]\setminus S} a_i) c_{[r_S]} = c_{[r_S]} \prod_{i\in[r_S]\setminus S} c_{M\cup[k]\setminus\{i\}}. \end{aligned}$$

The case when $\bar{g} = 1$ is similar.

Conversely, assume that there exists $M \neq \emptyset \subseteq [n]\setminus[k]$, such that $c_{M\cup[k]} = 1$, and the equations hold for a polynomial f. Clearly, f depends on $x_1, \ldots, x_k$. Hence in order to show f is k-canalizing, it is enough to show $f(\overline{a_1}, \ldots, \overline{a_{j-1}}, a_j, x_{j+1}, \ldots, x_n) = b_j$, for some $a_j, b_j \in \mathbb{F}_2$, and $1 \leq j \leq k$. Let $1 \leq j \leq k$, for all $S \subset [n]$ such that $j \notin S$, we either have $S \subseteq [j-1]$, or $S \not\subseteq [j-1]$. When $S \not\subseteq [j-1]$, we have $c_S \cdot c_{M\cup[k]\setminus\{j\}} = c_{S\cup\{j\}}$. By pairing c_S with $c_{S\cup\{j\}}$, such that $j \notin S$, we have

$$f(x_1, \ldots, x_n) = \sum_{S\subseteq[j]} c_S X_S + (c_{M\cup[k]\setminus\{j\}} + x_j) \sum_{S\not\subseteq[j-1], j\notin S} c_{S\cup\{j\}} X_S.$$

For $1 \leq j \leq k$, let $a_j = c_{M\cup[k]\setminus\{j\}}$. Then

$$f(\overline{a_1}, \ldots, \overline{a_{j-1}}, a_j, x_{j+1}, \ldots, x_n) = \sum_{S\subseteq[j-1]} (c_S + c_{S\cup\{j\}} c_{M\cup[k]\setminus\{j\}}) \prod_{i\in S}(1 + c_{M\cup[k]\setminus\{i\}})$$

is a constant we called b_j. Hence f is k-canalizing. The case when for all $M \neq \emptyset \subseteq [n]\setminus[k]$, $c_{M\cup[k]} = 0$ and $c_{[k]} = 1$ is similar.

□

For notational purposes, we generalize the idea of completion and define k-completion.

Definition 48. *Let $S \subseteq [n]$ be an non-empty subset. Let r_S be the largest element of S if $S \subseteq [k]$. The* k-completion *of S, denoted by $[r_S]_k$, is the set*

$$[r_S]_k = \begin{cases} \{1, \ldots, r_S\} & S \subseteq [k] \\ S \cup [k] & S \not\subseteq [k]. \end{cases}$$

For $S = \emptyset$, let $[r_\emptyset]_k = \emptyset$. In particular, $[r_S]_n = [r_S]$.

Using this new notation, we can restate Theorem 47, so the generalization from NCFs in Theorem 45 to k-canalizing becomes apparent.

Theorem 49. *Let f be a Boolean polynomial in n variables, given by*

$$f = \sum_{S\subseteq[n]} c_S X_S.$$

The polynomial f is a k-canalizing function in the order $x_1, x_2, \ldots, x_k$, if and only if there exists $M \neq \emptyset \subseteq [n]\setminus[k]$, such that $c_{M\cup[k]} = 1$, and for any subset $S \subseteq [n]$,

$$c_S = c_{[r_S]_k} \prod_{i\in[r_S]_k\setminus S} c_{M\cup[k]\setminus\{i\}}$$

or, for all $M \neq \emptyset \subseteq [n]\backslash[k]$, $c_{M\cup[k]} = 0$, *and* $c_{[k]} = 1$, *then for any subset* $S \subseteq [n]$,

$$c_S = \begin{cases} \prod_{i\in[r_S]_k\backslash S} c_{[k]\backslash\{i\}} & S \subseteq [k] \\ 0 & S \not\subseteq [k]. \end{cases}$$

In the latter case, the function is an NCF on k *variables, and fictitious on the other* $n - k$ *variables.*

We can now generalize the above theorem to any k-permutation σ of $[n]$.

Definition 50. *Let* σ *be a* k*-permutation of the elements of the set* $[n]$. *We define a new order relation* $<_\sigma$ *on the elements of* $\sigma([k])$ *as follows:* $\sigma(i) <_\sigma \sigma(j)$ *if and only if* $i < j$. *Let* S *be an non-empty subset of* $[n]$. *Let* r_S^σ *be the largest element of* S *with respect to the order relation* $<_\sigma$ *if* $S \subseteq \sigma([k])$. *The* k*-completion of* S *with respect to the* k*-permutation* σ, *denoted by* $[r_S^\sigma]_k$, *is the set*

$$[r_S^\sigma]_k = \begin{cases} \{\sigma(1), \ldots, \sigma(r_S)\} & S \subseteq \sigma([k]) \\ S \cup \sigma([k]) & S \not\subseteq \sigma([k]). \end{cases}$$

Theorem 51. *Let* f *be a Boolean polynomial in* n *variables, given by*

$$f = \sum_{S\subseteq[n]} c_S X_S.$$

Let σ *be a* k*-permutation of* $[n]$. *The polynomial* f *is a* k*-canalizing function in the order* $x_{\sigma(1)}, x_{\sigma(2)}, \ldots, x_{\sigma(k)}$, *if and only if there exists* $M \neq \emptyset \subseteq [n]\backslash\sigma([k])$, *such that* $c_{M\cup\sigma([k])} = 1$, *and for any subset* $S \subseteq [n]$,

$$c_S = c_{[r_S^\sigma]_k} \prod_{i\in[r_S^\sigma]_k\backslash S} c_{M\cup\sigma([k])\backslash\{\sigma(i)\}}$$

or, for all $M \neq \emptyset \subseteq [n]\backslash\sigma([k])$, $c_{M\cup\sigma([k]} = 0$, *and* $c_{\sigma([k])} = 1$, *then for any subset* $S \subseteq [n]$,

$$c_S = \begin{cases} \prod_{i\in[r_S^\sigma]_k\backslash S} c_{\sigma([k])\backslash\{\sigma(i)\}} & S \subseteq \sigma([k]) \\ 0 & S \nsubseteq \sigma([k]) \end{cases}$$

In the latter case, the function is an NCF on k *variables, and fictitious on the other* $n - k$ *variables.*

Corollary 52. *Let* σ *be a* k*-permutation of* $[n]$ *and let* $M \neq \emptyset \subseteq [n]\backslash\sigma([k])$ *be a nonempty subset of* $[n]\backslash\sigma([k])$. *The set of points in* $\mathbb{F}_2^{2^n}$ *corresponding to* k*-canalizing functions in the variable order* $x_{\sigma(1)}, x_{\sigma(2)}, \ldots, x_{\sigma(k)}$, *such that* $c_{M\cup\sigma([k])} = 1$, *denoted by* $V_{\sigma,M}^{k\text{-canalizing}}$, *is defined by*

$$V_{\sigma,M}^{k\text{-canalizing}} = \{(c_\emptyset, \ldots, c_{[n]}) \in \mathbb{F}_2^{2^n} : c_{M\cup\sigma([k])} = 1, c_S = c_{[r_S^\sigma]_k} \prod_{i\in[r_S^\sigma]_k\backslash S} c_{M\cup\sigma([k])\backslash\{\sigma(i)\}}, \text{ for } S \subseteq [n]\}. \tag{3.8}$$

Corollary 52 is the starting point for a geometric analysis of the set of all k-canalizing functions. It provides a set of equations that have to be satisfied by the coefficient vectors of the polynomial representations of the functions. These coefficient vectors therefore form an algebraic variety in the space $\mathbb{F}_2^{2^n}$, which turns out to have nice properties. Let $I_{\sigma,M}^{k\text{-canalizing}}$ be the ideal generated by the set of equations in Eq. 3.8. That is

$$I_{\sigma,M}^{k\text{-canalizing}} = \langle c_{M\cup\sigma([k])} = 1, c_S = c_{[r_S^\sigma]_k} \prod_{i\in[r_S^\sigma]_k\backslash S} c_{M\cup\sigma([k])\backslash\{\sigma(i)\}}, \text{ for } S \subseteq [n]\rangle.$$

Directly from Theorem 2 in [25], one can show $I_{\sigma,M}^{k\text{-canalizing}}$ is a prime ideal generated by binomials, and hence, a toric ideal. Thus, $V_{\sigma,M}^{k\text{-canalizing}}$ is a toric variety.

Corollary 53. *Let* σ *be a* k*-permutation of* $[n]$. *The set of points in* $\mathbb{F}_2^{2^n}$ *corresponding to the* k*-variable nested canalizing functions in the variable order* $x_{\sigma(1)}, x_{\sigma(2)}, \ldots, x_{\sigma(k)}$, *denoted by*

$V_{\sigma,\emptyset}^{k-canalizing}$, *is defined by*

$$V_{\sigma,\emptyset}^{k-canalizing} = \{(c_\emptyset, \ldots, c_{[n]}) \in \mathbb{F}_2^{2^n} : c_{\sigma([k])} = 1, c_S = \prod_{i \in [r_S^\sigma]_k \setminus S} c_{\sigma([k]) \setminus \{\sigma(i)\}}, \textit{ for } S \subseteq \sigma([k]),$$
$$c_S = 0, \textit{ for } S \nsubseteq \sigma([k])\}.$$

Therefore, $V_{\sigma,\emptyset}^{k-\text{canalizing}}$ is also a toric variety.

Corollary 54. *Let σ be a k-permutation of $[n]$. The set of points in $\mathbb{F}_2^{2^n}$ corresponding to k-canalizing functions in the variable order $x_{\sigma(1)}, x_{\sigma(2)}, \ldots, x_{\sigma(k)}$, denoted by $V_\sigma^{k-canalizing}$, is given by*

$$V_\sigma^{k-canalizing} = \bigcup_{M \subseteq [n] \setminus \sigma([k])} V_{\sigma,M}^{k-canalizing}.$$

Corollary 55. *The set of points in $\mathbb{F}_2^{2^n}$ corresponding to k-canalizing functions in n variables, denoted by $V^{k-canalizing}$, is given by*

$$V^{k-canalizing} = \bigcup_{\sigma \in \mathfrak{S}_{n,k}} V_\sigma^{k-canalizing}.$$

3.7 Reverse Engineering with k-canalizing Functions

In this section, we will propose an algorithm that identifies all k-canalizing models that fit the given data. We will be talking about gene regulatory networks, however, the methods apply for general molecular networks, such as biochemical reaction networks, protein-protein interaction networks, etc. Suppose that the gene regulatory network that we want to reverse engineer has n genes and that we have a set D of r state transition pairs $(s_j, t_j), j = 1, \ldots, r$. The input s_j and the output t_j are binary n-tuples encoding the state of genes $x_1, \ldots, x_n$. The

goal now is to find a model

$$f = (f_1, f_2, \ldots, f_n) : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n,$$

such that

$$f(s_j) = (f_1(s_j), \ldots, f_n(s_j)) = t_j.$$

Since $f_i : \mathbb{F}_2^n \to \mathbb{F}_2^n$ is over a finite field, it is a polynomial. An algorithm that finds all models f is presented in [37]. This is done by identifying, for each gene i, the set of all possible functions for f_i. This set can be represented as the coset $f + I$, where f is a particular such function and $I \subseteq \mathbb{F}_2[x_1, \ldots, x_n]$ is the ideal of all polynomials that vanish on the input data set, that is, $I = \mathbb{I}(\{s_1, \ldots, s_r\})$. However, in general, there are $2^{2^n - r}$ different Boolean functions in this coset. Hence, it is necessary to develop certain model selection procedures. One approach to improve the model selection process is restricting the model space $f + I$ by requiring not only that the chosen model fits the data but also satisfies some other conditions. Hinkelmann and Jarrah [22] designed an algorithm to reverse engineer GRNs using only nested canalizing functions. However, nested canalizing functions are very sparse in Boolean functions, so it is possible that there exists no nested canalizing function that fits a given data set. In this section, we will generalize Hinkelmann and Jarrah's algorithm and propose an algorithm that reverse engineers gene regulatory networks using k-canalizing functions.

In Section 3.6, we gave a parametrization of k-canalizing functions in $\mathbb{F}_2^{2^n}$. One can think of such a parametrization as a set of relations on the 2^n coefficients of a Boolean function. Recall that our goal is to identify all k-canalizing functions that fit the given data. To do this, we also need a parametrization of Boolean functions that fit the data in $\mathbb{F}_2^{2^n}$.

Recall that we are given the data set $D = \{(s_j, t_j) \in \mathbb{F}_2^n \times \mathbb{F}_2^n, j = 1, \ldots, r\}$. The model

space could be presented by the set $f + I$ where, for all i,

$$f_i(x_1, \ldots, x_n) = \sum_{j=1}^{r} t_{j,i} \prod_{e=1}^{n} (1 - (x_e - s_{j,e})).$$

Here, f_i is a interpolating polynomial that fits the data for gene i, and I is the vanishing ideal for D. Explicitly, I is

$$\begin{aligned} I = \mathbb{I}(\{s_1, \ldots, s_r\}) &= \bigcap_{j=1}^{r} \mathbb{I}(s_j) = \bigcap_{j=1}^{r} \langle x_e - s_{j,e} : 1 \leq e \leq n \rangle \\ &= \bigcap_{j=1}^{r} \langle (1 - \prod_{e=1}^{n} (1 - (x_e - s_{j,e}))) \rangle \\ &= \langle \prod_{j=1}^{r} (1 - \prod_{e=1}^{n} (1 - (x_e - s_{j,e}))) \rangle. \end{aligned}$$

Now, a polynomial $g \in f_i + I$ if and only if

$$g = f_i + h(x_1, \ldots, x_n) \prod_{j=1}^{r} \left(1 - \prod_{e=1}^{n} (1 - (x_e - s_{j,e})) \right)$$

for some polynomial h, say $h = \sum_{H \subseteq [n]} b_H X_H$. By expanding the right-hand side and collecting terms, we get that

$$g = \sum_{S \subseteq [n]} w_S(\{b_H\}_{H \subseteq [n]}, D) X_S$$

, where, for $S \subseteq [n]$, the coefficient w_S is determined by the coefficients of h and the input data D. In other words, for a given data set D, we can think of $w_S^D(\{b_H\}_{H \subseteq [n]}) := w_S(\{b_H\}_{H \subseteq [n]}, D)$ as a function with variables $\{b_H\}_{H \subseteq [n]}$. Then the subset of points

$$W = \{(w_\emptyset^D(\{b_H\}_{H \subseteq [n]}), \ldots, w_{[n]}^D(\{b_H\}_{H \subseteq [n]})) \in \mathbb{F}_2^{2^n} : b_H \in \{0, 1\}, \text{ for } H = \emptyset, \ldots, [n]\}$$

is the set of Boolean functions in $\mathbb{F}_2^{2^n}$ that fit the data set D. Moreover, we can compute $\mathbb{I}(W)$

using the following theorem, whose proof follows directly from Theorem $2.4.2$ in [2].

Theorem 56. *Consider the ring homomorphism*

$$\Phi : \overline{\mathbb{F}}_2[\{c_S : S \subseteq [n]\}] \longrightarrow \overline{\mathbb{F}}_2[\{b_H : H \subseteq [n]\}]$$

given by, for $S \subseteq [n]$*,*

$$c_S \rightarrow w_S^D(\{b_H\}_{H \subseteq [n]}).$$

Then $\ker(\Phi)$ *is the ideal of all polynomials that fit the data set* D*. In particular, the rational points in the variety* $\mathbb{V}(\ker(\Phi))$ *is the set of all models that fit the data set* D*, namely* $f + I$*.*

We are interested in the set $W \bigcap V^{k-\text{canalizing}}$, the set of all k-canalizing functions that fit the data. Using the relationship between ideals and varieties, we have the following corollary.

Corollary 57. *The ideal of all* k*-canalizing functions that fit the data set* D *is* $\mathbb{I}(V^{k-canalizing}) + ker(\Phi)$.

In practice, we can compute $\ker(\Phi)$ efficiently using elimination ideals.

3.7.1 Algorithm Description

Input

Data set $D = \{(s_j, t_j) \in \mathbb{F}_2^n \times \mathbb{F}_2^n, j = 1, \ldots, r\}$. Integer k_i, $0 \leq k_i \leq n$, $1 \leq i \leq n$.

Output

For each variable x_i, the complete list of all k_i-canalizing functions interpolating the given data set.

Algorithm

The outline of the algorithm is as follows: For each i:

1. Compute $\mathbb{I}(V^{k_i-\text{canalizing}})$ in $\mathbb{F}_2[\{c_S : S \subseteq [n]\}]$.

2. Compute an interpolating polynomial $f_i = \sum_{j=1}^{r} t_{j,i} \prod_{e=1}^{n}(1-(x_e - s_{j,e}))$ that fits data D.

3. Compute the polynomial $p = \prod_{j=1}^{r}(1-\prod_{e=1}^{n}(1-(x_e - s_{j,e})))$ that generates the vanishing ideal $\mathbb{I}(\{s_1, \ldots, s_r\})$.

4. For $S \subseteq [n]$, compute the polynomial $w_S^D(\{b_H\}_{H\subseteq[n]})$, by expanding

$$\sum_{S\subseteq[n]} c_S X_S = f_i + (\sum_{H\subseteq[n]} b_H X_H)p.$$

5. Compute a Gröbner basis G of the ideal

$$\langle c_S - w_S^D(\{b_H\}_{H\subseteq[n]}) : S \subseteq [n]\rangle \subseteq \mathbb{F}_2[\{c_S\}_{S\subseteq[n]}, \{b_H\}_{H\subseteq[n]}]$$

   using lexicographical order $c_\emptyset \prec \ldots \prec c_{[n]} \prec b_\emptyset \prec \ldots \prec b_{[n]}$. The reason we pick this specific monomial order is we would like to express relations among variables $\{c_S\}$ explicitly. In fact, any lexicographical order such that $c_S \prec b_H$, $S \subseteq [n]$, $H \subseteq [n]$, will also work.

6. Concatenate generators of $G \cap \mathbb{F}_2[\{c_S\}_{S\subseteq[n]}]$ and $\mathbb{I}(V^{k_i-\text{canalizing}})$.

7. Use the primary decomposition on $\langle G\cap\mathbb{F}_2[\{c_S\}_{S\subseteq[n]}]\rangle + \mathbb{I}(V^{k_i-\text{canalizing}})$ to obtain necessary and sufficient conditions on the coefficients of all k_i-canalizing function fitting data set D.

3.8 Binary Decision Diagram of k-canalizing Functions

The interest in nested canalizing functions by electrical engineers arose because this is the precise class of functions whose "binary decision diagrams" have minimal average path length [9]. A *binary decision diagram* (BDD) is a simple data structure that can efficiently

represent a Boolean function, and it additionally serves as a convenient visual aid and a quick evaluation tool. The basic idea is to minimize the memory needed to store the function and the time steps needed to evaluate the function with respect to a fixed variable order.

A Boolean function $f(x_1, \ldots, x_n)$ can be also represented (inefficiently) as a *binary decision tree*. To do this, one needs to first specify a fixed *variable order* on $\{x_1, \ldots, x_n\}$. The binary decision tree gives a quick way to compute the function output when the values of the variables are plugged in according to this order. This is best seen by an example: consider the function

$$f(x_1, x_2, x_3) = x_1 \wedge (x_2 \vee x_3) = x_1x_2x_3 + x_1x_2 + x_1x_3$$

with variable order $x_1 < x_2 < x_3$, which means "x_1 *comes first, then* x_2*, then* x_3." The binary decision tree of this function is shown in Figure 3.1. The root vertex is labeled with the first variable (in this case, x_1). There are two outgoing edges – one corresponding to setting $x_1 = 0$ (dashed, and to the left), and the other to $x_1 = 1$ (solid, and to the right). Both of the children are labeled with the next variable in the fixed order (in this case, x_2). This process is repeated: each non-leaf node is labeled with a variable x_i and has exactly two children (nodes directly "below" it). The nodes are labeled by level, from top to bottom. Finally, each node labeled with the last variable (in this case, x_3) also has two children, but these are leaves. Notice that in general, there are 2^n leaves, each one having a unique length-n path back to the root. This path uniquely describes an evaluation of all n variables. Label each leaf with either 0 or 1 – the value of the function $f(x_1, \ldots, x_n)$ corresponding to this particular evaluation.

A binary decision tree is an extremely inefficient way to represent a Boolean function – it requires 2^n leaf nodes and $2^n - 1$ interior nodes, which is prohibitly large for functions of more than just a few variables. It has other draw-backs as well: it carries a lot of redundant information, and the trees of two n-variable functions look structurally identical. A *binary decision diagram* (BDD) can be thought of as a "reduced" version of a binary decision tree, in that it carries the

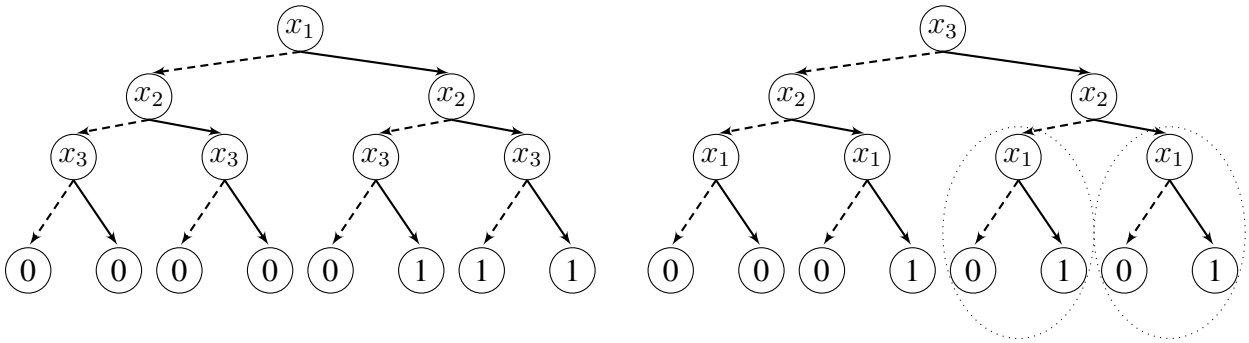

Figure 3.1: Two binary decision trees of $f = x_1x_2x_3 + x_1x_2 + x_1x_3$, with respect to variable orders $x_1 < x_2 < x_3$ (left) and $x_3 < x_2 < x_1$ (right). A dashed edge out of node x_i means that $x_i = 0$, and a solid edge out of x_i means $x_i = 1$. Each evalution of $x_1 = a_1$, $x_2 = a_2$, and $x_3 = a_3$ corresponds to a unique leaf which is labeled by $f(a_1, a_2, a_3)$.

same information but without the redundancies. For example, the tree on the left of Figure 3.2 is the BDD for the function $f = x_1x_2x_3 + x_1x_2 + x_1x_3$, with order $x_1 < x_2 < x_3$. Notice how any evaluation $f(a_1, a_2, a_3)$ can be computed in the same manner as for the binary decision tree – start at the root, and follow the paths corresponding to $x_1 = a_1$, $x_2 = a_2$, and $x_3 = a_3$ until a node labeled with 0 or 1 is reached. Also notice how the fact that f is an NCF with the variable order $x_1 < x_2 < x_3$ can be visualized from the BDD: Starting at the node labeled x_1, there is an edge $(x_1 = 0)$ down to a leaf. However, the other edge $(x_1 \neq 0)$ leads to a node labeled x_2, which also has a direct edge $(x_2 = 1)$ to a leaf. To summarize, there is a unique node at every level, and each node has a unique path to a leaf node. On the other hand, the BDD on the right of Figure 3.2 for the same function f but with respect to variable order $x_3 < x_2 < x_1$ does not have that property. Specifically, there is no edge from x_3 to a leaf. Even stronger: upon following either edge from x_3, one can still reach both 0 and 1 nodes. Therefore, x_3 is not a canalizing variable.

In any rooted tree, every node has a canonical *subtree* consisting of itself and all of its descendants. However, the BDDs in Figure 3.2 are not trees, but acyclic directed graphs. In such a structure, every node still has an analogue of a subtree that we called a *substructure*. Specifically, the substructure of v is the directed graph consisting of all of the nodes and edges

Figure 3.2: Two BDDs of $f = x_1x_2x_3 + x_1x_2 + x_1x_3$. The one on the left arises from the variable order $x_1 < x_2 < x_3$, and the one on the right arise from $x_3 < x_2 < x_1$

that can be reached via a directed path from v.

In general, the problem of how to construct a BDD given a Boolean function and a fixed variable order is difficult. If a binary decision tree has already been constructed, then it can be easily reduced to a BDD by repeatedly applying the following operations:

(i) Merge identical substructures that have the same parent node, and then eliminate that node.

(ii) Merge identical substructures that have different parents.

These two operations are applied repeatedly as long as they are applicable and the resulting graph will be a BDD. As an example of this, the binary decision tree of $f = x_1x_2x_3 + x_1x_2 + x_1x_3$ with respect to order $x_3 < x_2 < x_1$ is shown in Figure 3.1 on the right. Notice that for three of the four nodes labeled x_1, the substructures (subtrees) rooted at those nodes are identical. When the two circled subtrees are merged and their parent node (labeled x_2) is eliminated, we obtain the diagram on the left of Figure 3.3. This diagram also has two identical subtrees rooted at x_1-nodes, but with different parents. Merging these subtrees gives the diagram in the middle of Figure 3.3, which is no longer a tree. Finally, the x_1-node that has two 0-children can be eliminated, yielding the BDD which is shown on the right in Figure 3.3.

It is not obvious, but every Boolean function has, for a given ordering of variables, a unique binary decision diagram [9]. On the other hand, different variable orders of the same function generally have BDDs that are structurally different. For example, both diagrams in

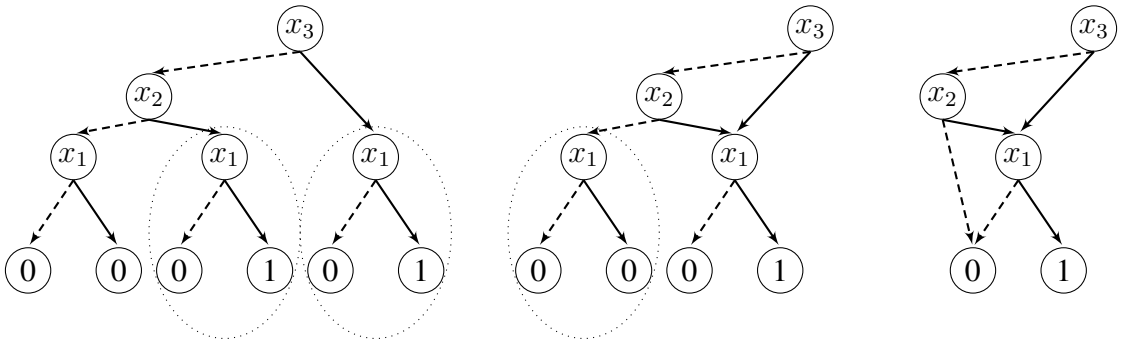

Figure 3.3: The merging process applied to the binary decision tree of $f = x_1x_2x_3 + x_1x_2 + x_1x_3$, with order $x_3 < x_2 < x_1$. The tree on the left is obtained by merging the two circled subtrees in Figure 3.1 and removing the parent vertex. The middle tree is formed by merging the two circled subtrees on the leftmost tree, and the tree on the right is formed by eliminating the x_1-labeled vertex.

Figure 3.2 are BDDs of the same function, but with respect to different variable orderings.

One of the primary utilities of BDDs is that they represent a concise representation of a Boolean function. Every evaluation of a function's variables corresponds to a path from the root to the leaves. The average path length of a BDD, taken over all 2^n evaluations, is in some sense a measure of the function's complexity. Since this depends on the variable order, we say that the *average path length* (APL) of a Boolean function is the minimal average path length of one of its BDDs, taken over all possible orderings. We denote this as APL_f, and it describes how quickly f can be evaluated on average. An NCF has the property that at every interior vertex, there is a direct edge to a leaf, which lowers the average path length considerably. Obviously, constant functions have the simplest BDDs – they would consist of just a single vertex and no edges. The number of levels of a BDD describes how many variables the function depends on. By definition, an NCF depends on all variables, and so its diagram must have n levels (or $n + 1$, if the leaves are included). The next result, from the electrical engineering community, says that NCFs are the "quickest" functions to evaluate. It may be no coincidence that these functions often arise in biological networks – they may possess some sort of evolutionary advantage.

Theorem 58 ([27])**.** *The n-variable Boolean functions with no fictitious variables that have*

minimal average path length are precisely the nested canalizing functions.

Theorem 59 ([10])**.** *The nested canalizing functions are uniquely those functions whose BDDs have the smallest APL* $(2 - \frac{1}{2^{n-1}})$ *among all functions that depend on* n*-variables.*

Example 60. $f = x_1x_2x_3 + x_1x + 2 + x_1x + 3 = x_1((x_2 + 1)(x_3 + 1) + 1)$ *is nested canalizing in the variable order* $x_1 < x_2 < x_3$*. The BDD given by this variable order is shown on the left in Figure 3.2. The average path length of this BDD achieves minimum among all possible variable order:*

$$\mathrm{APL}_f = \frac{1}{2} + \frac{2}{4} + \frac{3}{8} + \frac{3}{8} = \frac{7}{4} = 2 - \frac{1}{2^{3-1}}.$$

The key observation here is that, for a function f that is nested canalizing with respect to variable order σ, APL_f is achieved by taking the exact same variable order σ. In fact, if a function is canalizing, one should pick the canalizing variable to be the first variable in BDD in order to achieve minimum APL [46, 53]. A k-canalizing function is simply a Boolean function that is recursively canalizing for at least k-steps. Based on these observations, we can reach the following conclusion about the APL of a k-canalizing function.

Theorem 61. *If* f *is a* k*-canalizing Boolean function on* n *variables, then*

$$2 - \frac{1}{2^{k-1}} \leq \mathrm{APL}_f \leq 2 - \frac{1}{2^{k-1}} - \frac{k}{2^k} + \frac{n}{2^k}$$

Proof. Assume f is a k-canalizing Boolean function, with respect to k-permutation σ, as shown in the form of Eq. (3.2). In order to achieve minimum APL, the order of the first k variables in the BDD should be σ. Hence, $\frac{1}{2}$ of the inputs should have path length 1, $\frac{1}{4}$ of the inputs should have path length 2, etc. When the first k variables are fixed, the path length of $1 - \frac{1}{2^k}$ the inputs has been determined. All we need to compute now is the average path length of the remaining $\frac{1}{2^k}$ of the inputs, where the k canalizing variables have all taken non-canalizing inputs. When g is a constant function, the path length of all the remaining $\frac{1}{2^k}$ of the inputs is k. In this case,

$\mathrm{APL}_f = 2 - \frac{1}{2^{k-1}}$. When g is a parity function, the path length of all the remaining $\frac{1}{2^k}$ of the inputs is n. In this case, $APL_f = 2 - \frac{1}{2^{k-1}} - \frac{k}{2^k} + \frac{n}{2^k}$.

□

Chapter 4

Data Identification for Improving Gene Network Inference

4.1 Introduction

Gene regulatory networks (GRNs) in molecular biology have been classically modeled using continuous methods such as ordinary differential equations (ODEs). In the last few decades, Boolean network models have arisen as a popular alternative [4, 64]. Every Boolean function $\mathbb{F}_2^n \to \mathbb{F}_2$ is in fact, a multivariate polynomial in the ring $\mathbb{F}_2[x_1, \ldots, x_n]$. This opens the door to using tools from discrete math and computational algebra to tackle classic problems in molecular biology in a new way [38]. These types of models are called *polynomial dynamical systems* (PDSs), and many of them are not just Boolean, but over larger finite fields [61].

One classic question is how to *reverse engineer* a PDS model given partial data, e.g., time-series data [26, 37]. Generally, there are (too) many models which fit the data, and this *model space* admits an algebraic structure, similar to an affine vector space. In particular, it can be written as the coset $f + I$ in the polynomial ring $\mathbb{F}_p[x_1, \ldots, x_n]$, where $I = \mathbb{I}(V)$ is the *ideal* of models which vanish on the data set V, and f is any particular model that fits the data. This

is analogous to e.g., the structure of the solution space of a linear ODE, or a linear system of equations $Ax = b$.

The algebraic structure of the model space is key to using computational algebraic geometry to design algorithms for not only reverse-engineering, but also for *model selection*. This is the problem of choosing the "best" model from the space $f + I$, given some assumptions. Most GRNs are sparse, in the sense that they have few (less than half of all possible) connections, and their edges follow a power-law distribution [8, 65]. Thus, a model selection algorithm should return a model whose wiring diagram is sparse. One way to do this is to use Gröbner bases [15]. The main idea is to compute the unique remainder of f, given by multivariate division of f by $\mathcal{G}$, where $\mathcal{G}$ is a Gröbner basis of $\mathbb{I}(V)$.

The algorithm outlines as follows:

Input

Data set $D = \{(s_j, t_j) \in \mathbb{F}_2^n \times \mathbb{F}_2^n, j = 1, \ldots, r\}$. A monomial order $\prec$.

Algorithm

For each i:

1. Compute an interpolating polynomial f_i that fits the data D.
2. Compute a Gröbner basis $\mathcal{G}$ of the vanishing ideal $\mathbb{I}(\{s_1, \ldots, s_r\})$ with respect to $\prec$.
3. Compute the unique remainder of f_i upon division by the elements of $\mathcal{G}$.

However, $\mathbb{I}(V)$ might have multiple reduced Gröbner bases and each one is given by a specific monomial ordering. In this case, different monomial orderings yield different models.

Example 62. *Let $f = x^2 \in \mathbb{F}_3[x, y]$ and $V = \{(2, 0), (0, 1)\} \subseteq \mathbb{F}_3^2$. Then $\mathbb{I}(V)$ has two distinct reduced Gröbner bases (under different monomial orders): $\mathcal{G}_1 = \{x - y + 1, y^2 - y\}$ (with respect to $y \prec x$) and $\mathcal{G}_2 = \{x^2 + x, y - x + 2\}$ (with respect to $x \prec y$) which produce two different remainder polynomials: $\overline{f}^{\mathcal{G}_1} = -y + 1$ and $\overline{f}^{\mathcal{G}_2} = -x$.*

We saw in the above example that the choice of monomial ordering affects which model is selected from the model space. The two remainder polynomials look quite different. Applications of the above algorithm, such as network inference in [37], that use polynomial dynamical systems for modeling strongly depend on selecting minimal polynomials. Typically, it is hard to determine which term order to use. Prior knowledge of the network is often required to choose a "good" term order.

4.2 Data Identification for Unique Model

To resolve the dependency on the choice of the monomial order, E. Dimitrova and B. Stigler proposed a systematic way to add new data points to an existing data set to ensure that the ideal of points has a unique reduced Gröbner basis, yielding a unique model [14]. They also gave an algebraic characterization of the data points that need to be added. We will restate their characterization formally below (in a slightly modified way), and then summarize colloquially.

Theorem 63 ([14])**.** *Let $\mathbb{F}_p$ be a finite field where p is prime and $V \subseteq \mathbb{F}_p^n$ be a set of points for which $I = \mathbb{I}(V)$ has standard monomials $SM_1(I), \ldots, SM_m(I)$ (with respect to different monomial orders) in $R = \mathbb{F}_p[x_1, \ldots, x_n]$. Let M be the set of monomials that are in some but not all $SM_i(I)$, namely,*

$$M = \left(\bigcup_{i=1}^{m} SM_i(I) \right) \setminus \left(\bigcap_{i=1}^{m} SM_i(I) \right).$$

Finally, let

$$\mathcal{F} = \{ f \in I \mid \ LT(f) = x^\alpha \in M, \textit{ for some monomial order} \}.$$

A set of points $W \subset k^n \setminus V$ with

$$|W| = \left| \bigcup_{i=1}^{m} SM_i(I) \right| - |V|$$

is such that $\mathbb{I}(V \cup W)$ *has a unique reduced Gröbner basis if and only if* $\mathcal{F} \cap \mathbb{I}(W) = \emptyset$.

The set $\mathcal{F}$ is exactly the collection of polynomials in I that cause multiple reduced Gröbner bases. Hence, the main idea of this theorem is to "kick out" $\mathcal{F}$ from I by adding the set of points W that do not vanish on $\mathcal{F}$. $|\cup_{i=1}^{m} SM_i(I)| - |V|$ is the minimum possible number of points to add to achieve a unique reduced Gröbner basis.

Example 64. *Consider again the ideal of points* I *from Example 62. The standard monomials corresponding to the two Gröbner bases are* $SM_1(I) = \{1, y\}$ *and* $SM_2(I) = \{1, x\}$. *Also,* $|SM_1(I) \cup SM_2(I)| = 3$ *and* $|V| = 2$ *so we need to add at least* 1 *more point to the data set. Moreover, we have* $M = \{x, y\}$. *There are six monic polynomials that are in* I *and have* x *or* y *as a leading term under some term order:*

$$f_{11} = x + 2y + 1, \qquad f_{12} = x + y + y^2 + 1,$$

$$f_{13} = x + 2y^2 + 1, \qquad f_{21} = y + x^2 + 2,$$

$$f_{22} = y + x + 2x^2 + 2, \qquad f_{23} = y + 2x + 2.$$

Among the points in $\mathbb{F}_3^2 \backslash V$ *, only* $(a_1, a_2) = (0, 0)$ *and* $(2, 1)$ *are such that none of* f_{ij} *vanishes on* (a_1, a_2). *By inspection, one can verify that indeed these two points are the only ones that, added individually to* V*, generate an ideal of points with a unique reduced Gröbner basis. Notice that depending on which point is added to* V*, a different reduced Gröbner basis is produced, but in either case the standard monomials are* $\{1, x, y\} = SM_1(I) \cup SM_2(I)$.

Provided that such a W exists, $\mathcal{F} \cap \mathbb{I}(W) = \emptyset$ is both necessary and sufficient. However, generating W by explicitly finding the polynomials in $\mathcal{F}$ is impractical since the number of polynomials can be extremely large. Another problem with applying the above approach directly is in actually finding the points in W that satisfy the condition, especially since we want to be able to find all such minimal sets W. Moreover, it is possible that W, such that $|W| =$

$|\cup_{i=1}^{m} SM_i(I)| - |V|$ and $\mathcal{F} \cap \mathbb{I}(W) = \emptyset$ might not even exist [14]. Hence we propose to tackle this problem from a different angle. We are trying to give a combinatorial characterization of *all* $V \in \mathbb{F}_p^n$ such that $\mathbb{I}(V)$ has a unique reduced Gröbner basis, regardless of monomial order. The main result in this chapter is a sufficient condition for $\mathbb{I}(V)$ to have a unique reduced Gröbner basis. Before we state our main result, we will introduce some terminologies and build up the connection between staircases and Gröbner bases.

4.3 Staircases and Gröbner Bases

In this chapter, when we discuss point configurations in $\mathbb{F}_p^n$, we choose the smallest non-negative integer at each coordinate as representative. Hence we are embedding $\mathbb{F}_p^n$ into $\mathbb{N}^n$, where $\mathbb{N} = \{0, 1, 2, \ldots\}$. We denote the image of this embedding as $\mathbb{N}_p^n$.

Definition 65. *A* staircase *is a nonempty subset $\lambda \subseteq \mathbb{N}^n$, such that if $u \in \lambda$ and $v \leq u$ (coordinate-wise), then $v \in \lambda$.*

Example 66. *$\lambda_1 = \{(0,0), (1,0), (0,1)\}$ is a staircase. $\lambda_2 = \{(1,0), (0,1)\}$ is not a staircase.*

Example 67. *Exponent vectors of standard monomials of an ideal I, with respect to some monomial order $\prec$, also form a staircase, since any divisor of a standard monomial is again a standard monomial. Such staircase is called a* standard staircase *of I.*

We can also consider a staircase as a higher dimensional generalization of a Young diagram (a Young diagram is a $2D$ staircase). One important property of a staircase is that, any "layer" of a staircase is also a staircase (one dimension lower).

Definition 68. *Given a staircase $\lambda = \{u = (u_1, \ldots, u_n)\} \subseteq \mathbb{N}^n$, the i*th layer *of λ with respect to the jth coordinate is the subset $\{u \in \lambda : u_j = i\} \subseteq \lambda$. Let k be the largest integer such that $\{u \in \lambda : u_j = k\} \neq \emptyset$. Then the* height *of λ in the jth coordinate is defined to be $k+1$, denoted as $h_j(\lambda)$.*

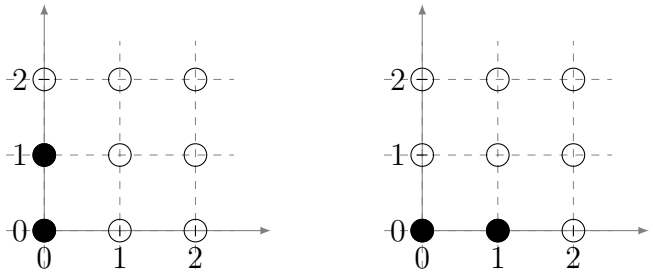

Figure 4.1: Standard staircases of the ideal I in Example 62. For a given Gröbner basis, each diagram shows the interface between the standard monomials, which are represented by black dots, and the leading terms, which are represented by white dots. A monomial is depicted via its exponent vector: $(m, n) \leftrightarrow x^m y^n$. In these examples, $SM_1 = \{1, y\}$ and $SM_2 = \{1, x\}$.

One can ask many questions about staircases since it is an interesting class of combinatorial objects. However, in this chapter we will be mainly focusing on the algebraic properties of a staircase. In particular, the connection between staircases and Gröbner bases. Here we will introduce a special type of staircases called basic staircases.

Definition 69. *Given a zero-dimensional ideal $I \subseteq \mathbb{F}_p[x_1, \ldots, x_n]$ (the dimension of $\mathbb{F}_p[x_1, \ldots, x_n]/I$ is finite), a subset $\lambda \in \mathbb{N}^n$ is* basic *for I if the congruence classes modulo I of the monomials x^v with $v \in \lambda$ form a vector space basis for the quotient space $\mathbb{F}_p[x_1, \ldots, x_n]/I$. If λ is also a staircase, then λ is called a* basic staircase *for I.*

If λ is basic then the class $[f] = f + I$ of any $f \in \mathbb{F}_p[x_1, \ldots, x_n]$ can be uniquely represented as linear combination of $\{x^v \mid v \in \lambda\}$. For a given monomial order, any polynomial $f \in \mathbb{F}_p[x_1, \ldots, x_n]$ has a unique normal form with respect to ideal I. Hence, a standard staircase of an ideal I is basic. We can check whether a set $\lambda \in \mathbb{N}_p^n$ is basic by looking at the evaluation matrix.

Definition 70. *Let $\lambda = \{u^1, \ldots, u^r\}$ be an r-subset of $\mathbb{N}_p^n$ and let $V = \{v^1, \ldots, v^s\}$ be a s-subset of $\mathbb{N}_p^n$. The* evaluation matrix $\mathbb{X}(x^\lambda, V)$ *is the s by r matrix whose element in position (i, j) is $x^{u^j}(v^i)$, the evaluation of x^{u^j} at v^i.*

Example 71. *Let $\lambda_1 = \{(0, 0), (1, 0)\}$, $\lambda_2 = \{(0, 0), (0, 1)\}$ and $V = \{(2, 0), (0, 1)\}$ be subsets*

of $\mathbb{N}_3^2$. *Then* $\mathbb{X}(x^{\lambda_1}, V) = \begin{bmatrix} 1 & 2 \\ 1 & 0 \end{bmatrix}$ *and* $\mathbb{X}(x^{\lambda_2}, V) = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$.

Theorem 72 ([5]). *Let* $\lambda \subseteq \mathbb{N}_p^n$ *be a subset and* V *be a subset of* $\mathbb{N}_p^n$. *Then* λ *is basic for* $\mathbb{I}(V)$ *if and only if the evaluation* $\mathbb{X}(x^\lambda, V)$ *is invertible.*

Example 73. *Let* $\lambda_1 = \{(0,0),(1,0)\}$, $\lambda_2 = \{(0,0),(0,1)\}$ *and* $V = \{(0,0),(1,0)\}$ *be subsets of* $\mathbb{N}_3^2$. λ_1 *is basic for* $\mathbb{I}(V)$, *since* $\mathbb{X}(x^{\lambda_1}, V) = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$ *is invertible.* λ_2 *is not basic for* $\mathbb{I}(V)$, *since* $\mathbb{X}(x^{\lambda_2}, V) = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}$ *is not invertible.*

Remark 74. *A standard staircase must be basic, while a basic staircase might not be standard.*

However, if I has a unique standard staircase (hence a unique reduced Gröbner basis), then I has a unique basic staircase.

Proposition 75. *An ideal* I *has a unique standard staircase if and only if* I *has a unique basic staircase.*

Proof. Follows directly from Proposition 2.2 in [5]. □

Based on Proposition 75, if we want to find out whether $\mathbb{I}(V)$ has a reduced Gröbner basis, we just need to check whether $\mathbb{I}(V)$ has a unique basic staircase. In other words, we can just check if there exist a unique staircase $\lambda \subseteq \mathbb{N}_p^n$, such that $\mathbb{X}(x^\lambda, V)$ is invertible.

Before we state the main theorem of this chapter, we will first introduce an equivalence relation on subset of $\mathbb{N}_p^n$ that fits well with staircases and Gröbner bases.

Definition 76. *For* $V_1, V_2 \subseteq \mathbb{N}_p^n$, *we say* V_1 *is a* linear shift *of* V_2, *denoted as* $V_1 \overset{L}{\sim} V_2$, *if there exist* $\phi = (\phi_1, \ldots, \phi_n) : \mathbb{N}_p^n \longrightarrow \mathbb{N}_p^n$, *such that* $V_1 = \phi(V_2)$, *and* $\phi_i(x) = a_i x + b_i$, $a_i \in \mathbb{F}_p^*$, $b_i \in \mathbb{F}_p$, *for* $i = 1, \ldots, n$.

Example 77. *Consider $V_1, V_2, V_3 \subseteq \mathbb{N}_3^2$, where $V_1 = \{(0,0),(0,1)\}$, $V_2 = \{(1,1),(1,2)\}$ and $V_3 = \{(1,1),(2,2)\}$. $V_1 \overset{L}{\sim} V_2$ since $V_1 = \phi(V_2)$ where $\phi = (x+1, x+1)$. However, $V_1 \overset{L}{\not\sim} V_3$ since the x-coordinates of the points in V_3 are different.*

Proposition 78. *If $V_1, V_2 \subseteq \mathbb{N}_p^n$ are both staircases and $V_1 \overset{L}{\sim} V_2$, then $V_1 = V_2$.*

Proof. By mathematical induction. □

Proposition 79. *If $V_1 \overset{L}{\sim} V_2$, then $\mathbb{I}(V_1)$ and $\mathbb{I}(V_2)$ have the same number of reduced Gröbner bases. In particular, when $\mathbb{I}(V_1)$ has a unique reduced Gröbner basis, $\mathbb{I}(V_2)$ will also have a unique reduced Gröbner basis.*

Proof. First, observe that

$$\begin{aligned}\mathbb{I}(V_2) &= \{f : f(v) = 0, \forall v \in V_2\} \\ &= \{f : f(\phi(u)) = f \circ \phi(u) = 0, \forall u \in V_1\}.\end{aligned}$$

Thus, for any $f \in \mathbb{I}(V_2)$, we have $f \circ \phi \in \mathbb{I}(V_1)$. Since ϕ is a linear shift, f and $f \circ \phi$ have the same leading monomial with respect to any monomial ordering. Therefore, $LT(\mathbb{I}(V_2)) \subseteq LT(\mathbb{I}(V_1))$. We can show $LT(\mathbb{I}(V_1)) \subseteq LT(\mathbb{I}(V_2))$ by replacing ϕ with ϕ^{-1}. Hence $LT(\mathbb{I}(V_1)) = LT(\mathbb{I}(V_2))$ with respect to any monomial ordering. Due to the one-to-one correspondence between initial ideals and reduced Gröbner bases, $\mathbb{I}(V_1)$ and $\mathbb{I}(V_2)$ have the same number of reduced Gröbner bases. □

4.4 A Sufficient Condition for Unique Reduced Gröbner basis

In this section, we will present a sufficient condition for $\mathbb{I}(V)$ to have a unique reduced Gröbner basis.

Theorem 80. *Let $\lambda \in \mathbb{N}_p^n$ and $V \in \mathbb{N}_p^n$ be two staircases. Then λ is basic for $I(V)$ if and only if $\lambda = V$.*

Proof. We first induct on number of variables: $n = 1$:

$\Rightarrow$ If $\lambda \neq V$:, since we have only one variable, we must have $|\lambda| \neq |V|$. Therefore $\mathbb{X}(x^\lambda, V)$ is not invertible since it is not a square matrix.

$\Leftarrow$ If $\lambda = V$:, then $\mathbb{X}(x^\lambda, V)$ is a square Vandermonde matrix. Since V is a set of distinct points, $\mathbb{X}(x^\lambda, V)$ is invertible.

If the inductive hypothesis holds for $n = k$, let us consider the case when $n = k + 1$:

Here we are going to induct on the height of the monomial staircase with respect to the first coordinate. Let us consider the base case when $h_1(\lambda) = 1$, in other words, for all $u \in \lambda$, we have $u_1 = 0$. That is to say all monomials of x^λ do not involve x_1.

$\Leftarrow$: If $\lambda = V$, then the first coordinate of any point in V is 0. Therefore λ and V are essentially staircases with one fewer variable (x_1). Based on our inductive hypothesis $n = k$, we have x^λ are basic monomials of $I(V)$.

$\Rightarrow$: If $\lambda \neq V$.

case 1: $|\lambda| \neq |V|$. In this case, $\mathbb{X}(x^\lambda, V)$ is not invertible as it is not square.

case 2: $h_1(V) \geq 2$. In this case, $\mathbb{X}(x^\lambda, V)$ is not invertible as at least two rows are the same.

case 3: $h_1(V) = h_1(\lambda) = 1$, while $\lambda \neq V$: In this case, the first coordinate of any point in λ and V is 0. In other words, λ and V are essentially staircases with one fewer variable (x_1). Based our inductive hypothesis for $n = k$, we have $\mathbb{X}(x^\lambda, V)$ not invertible.

Assume the inductive hypothesis holds for all monomial staircases λ with $1 \leq h_1(\lambda) \leq d$. Let us consider a staircase λ with $h_1(\lambda) = d + 1$.

$\Leftarrow$: If $\lambda = V$. Let $\lambda_0 := \{u \in \lambda : u_1 = 0\}$ denote the 0th layer of λ and $V_0 := \{v \in \lambda : v_1 = 0\}$ denote the 0th layer of V with respect to the first coordinate. Since $\lambda = V$, we have $\lambda_0 = V_0$. By inductive hypothesis, the evaluation matrix $\mathbb{X}(x^{\lambda_0}, V_0)$ is invertible. Now let us consider the evaluation matrix $\mathbb{X}(x^\lambda, V)$. We can reorder rows and columns of $\mathbb{X}(x^\lambda, V)$ so that $\mathbb{X}(x^{\lambda_0}, V_0)$ appears as the upper left submatrix of $\mathbb{X}(x^\lambda, V)$. Since the upper left submatrix of $\mathbb{X}(x^\lambda, V)$ is

invertible, after basic row and column operations, we can transform $\mathbb{X}(x^\lambda, V)$ into a block matrix of the form

$$\begin{bmatrix} I & 0 \\ 0 & \mathbb{X}(x^{\lambda\backslash\lambda_0}, V\backslash V_0) \end{bmatrix}.$$

Moreover, x_1 divides all monomials in $x^{\lambda\backslash\lambda_0}$ and for any point $v \in V\backslash V_0$, we have $v_1 \neq 0$. Therefore, $\mathbb{X}(x^{\lambda\backslash\lambda_0}, V\backslash V_0)$ is invertible if and only if $\mathbb{X}(\frac{x^{\lambda\backslash\lambda_0}}{x_1}, V\backslash V_0)$ is invertible. Note that the $\frac{x^{\lambda\backslash\lambda_0}}{x_1}$ corresponds to a staircase with height at most d and $V\backslash V_0$ is a *linear shift* of the same staircase, so X^* is invertible by the inductive hypothesis and Proposition 79. Hence the original evaluation matrix $\mathbb{X}(x^\lambda, V)$ is also invertible.

$\Rightarrow$: If $\lambda \neq V$.

If $|\lambda| \neq |V|$, then $\mathbb{X}(x^\lambda, V)$ is not invertible since it is not a square matrix.

We can reorder rows and columns so that $\mathbb{X}(x^\lambda, V)$ appears of the form:

$$\begin{bmatrix} \mathbb{X}(x^{\lambda_0}, V_0) & 0 \\ A & \mathbb{X}(x^{\lambda\backslash\lambda_0}, V\backslash V_0) \end{bmatrix}.$$

Note that row space of A is a subspace of the row space in $\mathbb{X}(x^{\lambda_0}, V_0)$. If $\mathbb{X}(x^{\lambda_0}, V_0)$ is not a square matrix:

Case 1: If $\mathbb{X}(x^{\lambda_0}, V_0)$ has more rows than columns, then rows of $[\mathbb{X}(x^{\lambda_0}, V_0), 0]$ are linearly dependent.

Case 2: If $\mathbb{X}(x^{\lambda_0}, V_0)$ has more columns than rows, then columns of $[\mathbb{X}(x^{\lambda_0}, V_0), A]^T$ are linearly dependent.

In these cases, $\mathbb{X}(x^\lambda, V)$ is not invertible.

If $\mathbb{X}(x^{\lambda_0}, V_0)$ is a square matrix but $\lambda_0 \neq V_0$, then $\mathbb{X}(x^{\lambda_0}, V_0)$ is not invertible by the inductive hypothesis. So $\mathbb{X}(x^\lambda, V)$ is not invertible.

If $\mathbb{X}(x^{\lambda_0}, V_0)$ is a square matrix and $\lambda_0 = V_0$, then we must have $\lambda\backslash\lambda_0 \neq V\backslash V_0$. Note

that $\mathbb{X}(x^{\lambda\backslash\lambda_0}, V\backslash V_0)$ is invertible if and only if $\mathbb{X}(\frac{x^{\lambda\backslash\lambda_0}}{x_1}, V\backslash V_0)$ is invertible. Since $\lambda\backslash\lambda_0 \neq V\backslash V_0$ and $\frac{x^{\lambda\backslash\lambda_0}}{x_1}$ corresponds to a staircase with height at most d and $V\backslash V_0$ is a linear shift of some other staircase, $\mathbb{X}(\frac{x^{\lambda\backslash\lambda_0}}{x_1}, V\backslash V_0)$ is not invertible by the inductive hypothesis and Proposition 79. Therefore $\mathbb{X}(x^{\lambda}, V)$ is also not invertible. □

The following proposition is an immediate result of Theorem 80.

Proposition 81. *If $V \in \mathbb{N}_p^n$ is a staircase $\lambda \in \mathbb{N}_p^n$, then $I(V)$ has a unique reduced Gröbner basis.*

Example 82. *$V = \{(0,0),(0,1),(1,0)\}$ is a staircase in $\mathbb{N}_3^2$. $\mathbb{I}(V)$ has a unique reduced Gröbner basis $\mathcal{G} = \{y^2 - y, xy, x^2 - x\}$.*

Applying Proposition 79, we get a more general condition for $\mathbb{I}(V)$ to have a unique reduced Gröbner basis.

Proposition 83. *If $V \in \mathbb{N}_p^n$ is a linear shift of some staircase $\lambda \in \mathbb{N}_p^n$, then $I(V)$ has a unique reduced Gröbner basis.*

Example 84. *$V_1 = \{(0,0),(0,1),(1,0)\}$ is a staircase in $\mathbb{N}_3^2$. $V_2 = \{(0,1),(0,2),(2,2)\}$ is a subset of $\mathbb{N}_3^2$. $V_1 \overset{L}{\sim} V_2$, since $V_2 = \phi(V_1)$, where $\phi = (2x, 2x+2)$. $\mathbb{I}(V_2)$ has a unique reduced Gröbner basis $\mathcal{G} = \{y^2 - 1, xy + x, x^2 + x\}$.*

However, V being a linear shift of a staircase is not a necessary condition for $\mathbb{I}(V)$ to have a unique reduced Gröbner basis, as shown in the previous example.

Example 85. *The set $V = \{(0,0,0),(1,0,0),(1,1,0),(1,1,1)\}$ is not a linear shift of a staircase in $\mathbb{N}_2^3$. However, $\mathbb{I}(V)$ has a unique reduced Gröbner basis $\mathcal{G} = \{z^2 + z, yz + z, y^2 + y, xz + z, xy + y, x^2 + x\}$.*

Chapter 5

Conclusions and Discussion

5.1 Significance of Results

In this work, we focused on two model selection methods and algebraic geometry arising from them.

In Chapter 3, we extended results on nested canalizing functions and derived a unique extended monomial form of arbitrary Boolean functions. This gave us a stratification of the set of n-variable Boolean functions by canalizing depth. In particular, this form encapsulates three invariants of Boolean functions: canalizing depth, dominance layer number and the noncanalizing core polynomial. By combining these three invariants, we obtained an explicit formula for the number of Boolean functions on n variables with depth k. We also introduced the notion of k-canalizing Boolean functions, which is a promising framework for modeling gene regulatory networks. We also derived an algebraic parametrization of $V^{k\text{-canalizing}}$, namely, a collection of polynomial equations that encode the relations of the coefficients of k-canalizing functions. This parametrization describes the entire space of k-canalizing functions as a geometric object, whose properties can then be studied with the tools of algebraic geometry. We generalized Hinkelmann and Jarrah's algorithm and proposed an algorithm to reverse engineer gene regulatory networks

using k-canalizing functions. We also studied k-canalizing functions from a data structure point of view and computed the lower and upper bound of the average path length of a k-canalizing function.

In Chapter 4, we studied the model selection methods that involves Gröbner bases. The model is obtained by computing the unique remainder of f, given by multivariate division of f by $\mathcal{G}$, where $\mathcal{G}$ is a Gröbner basis of $\mathbb{I}(V)$. To minimize the effect of multiple Gröbner bases, we studied the connection between staircases and Gröbner bases. We provided a necessary and sufficient condition for $\mathbb{I}(V)$ to have a unique reduced Gröbner basis, using the concept of basic staircase. We also provide a sufficient combinatorial characterization of $V \subset \mathbb{N}_p^n$ that yields to a unique reduced Gröbner basis.

5.2 Future Work

The application of discrete models to biological networks is a blossoming field, so there is still much work to be done in this area of research.

Our work in Chapter 3 motivates us to continue exploring canalization in Boolean functions. Our stratification yielded closed formulas for the number of n-variable Boolean functions of canalizing depth k. We are currently working on deriving asymptotics for the number of such functions as n and k grow large, using analytic combinatorial techniques. We will investigate well-known Boolean network models and compute the canalizing depth of the proposed functions. We will also study the impact of the stratification of all Boolean functions on random Boolean networks (RBNs). For example, it was shown in [39] that RBNs with k-canalizing functions are more stable as the parameter k increases. In [40], the authors showed that RBNs built with NCFs of extended monomial layers are less stable as the parameter r increases. Both of these studies used the average sensitivity of a Boolean function [55] as well as a tool from statistical physics called a Derrida curve [13], which measures how small errors propagate throughout the

network. It would be interesting to consider RBNs where the Boolean functions are k-canalizing and have r monomial layers, and see how the stability depends jointly on the parameters k and r. We would like to extend the aforementioned stratification results to multi-state (rather than Boolean) functions. The definition of an NCF was extended from Boolean to multi-state functions in [45], where the authors also enumerated these functions. We are interested in questions about k-canalizing functions in computer science. Boolean NCFs have been studied extensively in engineering and computer science, since they are precisely the class of Boolean functions whose corresponding binary decision diagrams are of shortest average path length [9, 27]. This simply means that NCFs can be evaluated very efficiently. In that sense, k-canalizing functions are Boolean functions with binary decision diagrams of "short" average path length, which might lead to efficient algorithms.

In Chapter 4, we provided a sufficient combinatorial characterization of $V \subset \mathbb{N}_p^n$ that yields to a unique reduced Gröbner basis. We are trying to generalize this result to a necessary and sufficient combinatorial characterization. The reason we believe such characterization should exist is because computing a reduced Gröbner basis with a given monomial ordering is equivalent to solving a minimization problem with a cost vector on the variables. Hence, having a unique Gröbner basis should be a combinatorial property of the ideal $\mathbb{I}(V)$, which should be encoded in combinatorial properties of V. Another interesting question we could ask is that: given a subset $V \subseteq \mathbb{N}_p^n$, how can we determine whether V is a linear shift of a staircase in N_p^n. If V is not a linear shift of a staircase, how far away is it from a linear shift of a staircase? We are working on a systematical way to identify subsets $W \subseteq \mathbb{F}_p^n \setminus V$, so that $V \cup W$ becomes is a linear shift of a staircase, and hence, $\mathbb{I}(V \cup W)$ has a unique reduced Gröbner basis.

While we have provided a basis for the study of algebraic geometry arising from discrete models of gene regulatory networks, there are still many directions to explore. We hope that the results presented in this work are a source of motivation for continuing the application and study of these models.

Bibliography

[1] John Abbott, A Bigatti, Martin Kreuzer, and Lorenzo Robbiano. Computing ideals of points. *Journal of Symbolic Computation*, 30(4):341–356, 2000.

[2] William Wells Adams and Philippe Loustaunau. *An introduction to Gröbner bases*. Number 3. American Mathematical Soc., 1994.

[3] Charu Aggarwal and Karthik Subbian. Evolutionary network analysis: A survey. *ACM Computing Surveys (CSUR)*, 47(1):10, 2014.

[4] R. Albert and H.G. Othmer. The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in drosophila melanogaster. *J. Theor. Biol.*, 223(1):1–18, 2003.

[5] Eric Babson, Shmuel Onn, and Rekha Thomas. The hilbert zonotope and a polynomial time algorithm for universal gröbner bases. *Advances in Applied Mathematics*, 30(3):529–544, 2003.

[6] E.A. Bender and J.T. Butler. Asymptotic aproximations for the number of fanout-free functions. *IEEE T. Comput.*, 27(12):1180–1183, 1978.

[7] Paul Brazhnik, Alberto de la Fuente, and Pedro Mendes. Gene networks: how to put the function in genomics. *TRENDS in Biotechnology*, 20(11):467–472, 2002.

[8] Z Burda, A Krzywicki, OC Martin, and M Zagorski. Distribution of essential interactions in model gene regulatory networks under mutation-selection balance. *Physical Review E*, 82(1):011908, 2010.

[9] J.T. Butler, T. Sasao, and M. Matsuura. Average path length of binary decision diagrams. *Comput., IEEE Trans.*, 54(9):1041–1053, 2005.

[10] J.T. Butler, T. Sasao, and M. Matsuura. Average path length of binary decision diagrams. *IEEE T. Comput.*, 54(9):1041–1053, 2005.

[11] David A Cox, John Little, and Donal O'shea. *Using algebraic geometry*, volume 185. Springer Science & Business Media, 2006.

[12] David A Cox, John B Little, and Henry K Schenck. *Toric varieties*. American Mathematical Soc., 2011.

[13] Bernard Derrida and Yves Pomeau. Random networks of automata: a simple annealed approximation. *EPL (Europhysics Letters)*, 1(2):45, 1986.

[14] E. Dimitrova and B. Stigler. Data identification for improving gene network inference using computational algebra. *Bull. Math. Biol.*, 76(11):2923–2940, 2014.

[15] E.S. Dimitrova, A.S. Jarrah, R. Laubenbacher, and B. Stigler. A gröbner fan method for biochemical network modeling. In *Proc. 2007 Internat. Symp. Symbolic Algebraic Computat.*, pages 122–126. ACM, 2007.

[16] Jean-Charles Faugere. A new efficient algorithm for computing gröbner bases (f 4). *Journal of pure and applied algebra*, 139(1):61–88, 1999.

[17] Shuhong Gao, Yinhua Guan, and Frank Volny IV. A new incremental algorithm for computing gröbner bases. In *Proceedings of the 2010 International Symposium on Symbolic and Algebraic Computation*, pages 13–19. ACM, 2010.

[18] Timothy S Gardner and Jeremiah J Faith. Reverse-engineering transcription control networks. *Physics of life reviews*, 2(1):65–88, 2005.

[19] Irit Gat-Viks and Ron Shamir. Chain functions and scoring functions in genetic networks. *Bioinformatics*, 19(suppl 1):i108–i117, 2003.

[20] Rüdiger Gebauer and H Michael Möller. On an installation of buchberger's algorithm. *Journal of Symbolic Computation*, 6(2):275–286, 1988.

[21] Mika Gustafsson, Michael Hörnquist, and Anna Lombardi. Constructing and analyzing a large-scale gene-to-gene regulatory network lasso-constrained inference and biological validation. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, 2(3):254–261, 2005.

[22] F. Hinkelmann and A.S. Jarrah. Inferring biologically relevant models: nested canalyzing functions. *ISRN Biomathematics*, 2012, 2012.

[23] François Jacob and Jacques Monod. Genetic regulatory mechanisms in the synthesis of proteins. *Journal of molecular biology*, 3(3):318–356, 1961.

[24] K. Jansen and M.T. Matache. Phase transition of boolean networks with partially nested canalizing functions. *Eur. Phys. J. B*, 86(7):1–11, 2013.

[25] A.S. Jarrah and R. Laubenbacher. Discrete models of biochemical networks: The toric variety of nested canalyzing functions. In *Algebraic Biology*, pages 15–22. Springer, 2007.

[26] A.S. Jarrah, R. Laubenbacher, B. Stigler, and M. Stillman. Reverse-engineering of polynomial dynamical systems. *Adv. Appl. Math.*, 39(4):477–489, 2007.

[27] A.S. Jarrah, B. Raposa, and R. Laubenbacher. Nested canalyzing, unate cascade, and polynomial functions. *Physica D*, 233(2):167–174, 2007.

[28] W. Just, I. Shmulevich, and J. Konvalina. The number and probability of canalizing functions. *Physica D*, 197(3):211–221, 2004.

[29] C. Kadelka, Y. Li, J.O. Adeyeye, and R. Laubenbacher. Nested canalizing functions and their networks. *arXiv:1411.4067*, 2014.

[30] F. Karlsson and M. Hörnquist. Order or chaos in boolean gene networks depends on the mean fraction of canalizing functions. *Physica A*, 384(2):747–757, 2007.

[31] S. Kauffman. The large scale structure and dynamics of gene control circuits: an ensemble approach. *J. Theor. Biol.*, 44(1):167–190, 1974.

[32] S. Kauffman, C. Peterson, B. Samuelsson, and C. Troein. Random boolean network models and the yeast transcriptional network. *Proc. Natl. Acad. Sci.*, 100(25):14796–14799, 2003.

[33] S. Kauffman, C. Peterson, B. Samuelsson, and C. Troein. Genetic networks with canalyzing boolean rules are always stable. *Proc. Natl. Acad. Sci.*, 101(49):17102–17107, 2004.

[34] S.A. Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *J. Theor. Biol.*, 22(3):437–467, 1969.

[35] Stuart A. Kauffman. *The origins of order: Self organization and selection in evolution*. Oxford University Press, USA, 1993.

[36] J.G. Klotz, D. Kracht, M. Bossert, and S. Schober. Canalizing boolean functions maximize mutual information. *IEEE T. Inform. Theory*, 60(4):2139–2147, 2014.

[37] R. Laubenbacher and B. Stigler. A computational algebra approach to the reverse engineering of gene regulatory networks. *J. Theor. Biol.*, 229(4):523–537, 2004.

[38] R. Laubenbacher and B. Sturmfels. Computer algebra in systems biology. *Amer. Math. Monthly*, pages 882–891, 2009.

[39] L. Layne, E. Dimitrova, and M. Macauley. Nested canalyzing depth and network stability. *Bull. Math. Biol.*, 74(2):422–433, 2012.

[40] Y. Li, J.O. Adeyeye, D. Murrugarra, B. Aguilar, and R. Laubenbacher. Boolean nested canalizing functions: A comprehensive analysis. *Theor. Comput. Sci.*, 481:24–36, 2013.

[41] R. Lidl, H. Niederreiter, and P.M. Cohn. Encyclopedia of mathematics and its applications 20: Finite fields, 1996.

[42] Ernst W Mayr and Albert R Meyer. The complexity of the word problems for commutative semigroups and polynomial ideals. *Advances in mathematics*, 46(3):305–329, 1982.

[43] A.A. Moreira and L.A.N. Amaral. Canalizing Kauffman networks: Nonergodicity and its effect on their critical behavior. *Phys. Rev. Lett.*, 94(21):218702, 2005.

[44] Henning Mortveit and Christian Reidys. *An introduction to sequential dynamical systems*. Springer Science & Business Media, 2007.

[45] R. Murrugarra, D.and Laubenbacher. The number of multistate nested canalyzing functions. *Physica D*, 241(10):929–938, 2012.

[46] Shinobu Nagayama, Alan Mishchenko, Tsutomu Sasao, and Jon T Butler. Minimization of average path length in bdds by variable reordering. Technical report, DTIC Document, 2003.

[47] Donal O'Shea, John B Little, David Archibald Cox, David Archibald Cox, and David Archibald Cox. *Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra*. Springer, 2007.

[48] T.P. Peixoto. The phase diagram of random boolean networks with nested canalizing functions. *Euro. Phys. J. B*, 78(2):187–192, 2010.

[49] Tony Puthenpurackel, AV Jayanthan, Amit Khetan, Leah Gold, and Sara Faridi. Lecture notes for the graduate school at the. 2004.

[50] Luc Raeymaekers. Dynamics of boolean networks controlled by biologically meaningful functions. *Journal of Theoretical Biology*, 218(3):331–341, 2002.

[51] C. Ray, J.K. Das, and P.P. Choudhury. On analysis and generation of some biologically important boolean functions. *arXiv:1405.2271*, 2014.

[52] R. Robeva and T. Hodge. *Mathematical concepts and methods in modern biology: using modern discrete models*. Academic Press, 2013.

[53] T Sasao, JT Butler, and M Matsuura. Average path length as a paradigm for the fast evaluation of functions represented by binary decision diagrams. Technical report, DTIC Document, 2002.

[54] T. Sasao and K. Kinoshita. On the number of fanout-free functions and unate cascade functions. *IEEE T. Comput.*, 100(1):66–72, 1979.

[55] I. Shmulevich and S.A. Kauffman. Activities and sensitivities in boolean network models. *Phys. Rev. Lett.*, 93(4):048701, 2004.

[56] Bernd Sturmfels. Grobner bases of toric varieties. *Tohoku Mathematical Journal, Second Series*, 43(2):249–261, 1991.

[57] Bernd Sturmfels, Robert Weismantel, and Günter M Ziegler. *Gröbner bases of lattices, corner polyhedra, and integer programming*. ZIB, 1994.

[58] A. Szejka and B. Drossel. Evolution of canalizing boolean networks. *Euro. Phys. J. B*, 56(4):373–380, 2007.

[59] René Thomas. Boolean formalization of genetic control circuits. *Journal of theoretical biology*, 42(3):563–585, 1973.

[60] Vesteinn Thorsson, Michael Hörnquist, Andrew F Siegel, and Leroy Hood. Reverse engineering galactose regulation in yeast through model selection. *Statistical applications in genetics and molecular biology*, 4(1), 2005.

[61] A. Veliz-Cuba, A.S. Jarrah, and R. Laubenbacher. Polynomial algebra of discrete models in systems biology. *Bioinformatics*, 26(13):1637–1643, 2010.

[62] Joachim Von Zur Gathen and Jürgen Gerhard. *Modern computer algebra*. Cambridge university press, 2013.

[63] C.H. Waddington. Canalization of development and the inheritance of acquired characters. *Nature*, 150(3811):563–565, 1942.

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

[65] H.-Y. Yeh, S.-W. Cheng, Y.-C. Lin, C.-Y. Yeh, S.-F. Lin, and V.-W. Soo. Identifying significant genetic regulatory networks in the prostate cancer from microarray data based on transcription factor analysis and conditional independency. *BMC Med. Genet.*, 2(1):70, 2009.