August 2020

Inferring Networks with Gene Knockouts and Computational Algebra

Tilly Grace Erwin
Clemson University, terwin@clemson.edu

Inferring networks with gene knockouts and computational algebra

---

A Masters Project
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Mathematics

---

by
Tilly Erwin
August 2020

Accepted by:
Dr. Matthew Macauley, Committee Chair
Dr. Michael Burr
Dr. Svetlana Poznanovikj

Abstract

The network inference problem is a significant problem in systems biology. In this paper, we will describe an approach to this problem involving computational algebra. Specifically, given an unknown Boolean function, we can create a square-free monomial or pseudomonomial ideal whose primary decomposition encodes the possible sets of variables that the function can depend on, and whether those interactions are activations or inhibitions. We apply this problem to time series data generated from a non-linear ODE, built over unknown feed-forward loops, and subject to gene knockouts.

Dedication

I dedicate this project to my Milligan College advisors: Dr. Aaron Allen, Dr. Teresa Carter, and Dr. Nate Wentzel. Thank you for believing in me and pushing me to pursue this degree. I will be forever grateful for your support and encouragement.

Acknowledgments

A special thanks to Marc Birtwistle and his lab for their willingness to share his data, as well as to the Systems Biology Journal Club for providing the forum from which this project was born.

In addition I would like to thank my Husband and my family. I could never have gotten here without their support.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

A gene regulatory network is a collection of genes, gene products, and other biomolecules that self-regulate to carry out a common purpose. Scientists have used various methods to infer or reconstruct such gene systems, as this is a central biological problem. Several years ago in the Clemson University Systems Biology Club, one faculty member in Bioengineering was working on a project to reconstruct signaling networks using a new algorithm his group had developed. His group had collected time series of gene expression levels, and also collected data under targeted gene knockouts. Their algorithm involved systems of differential equations, but it had the drawback that there were sometimes false positives due to the difficulty of separating correlation versus causation. It was sometimes unclear whether a strong correlation actually meant that one variable affected the other. After seeing an algebraic biology talk, they asked whether existing theoretical reverse engineering methods based on algebraic geometry could do better? While there is a nice theoretical framework for this, it is hard to successfully implement in practice, and it is not clear how robust it is to noise. In addition, the current framework has never been used in conjunction with data from gene knockouts. In this paper, we will give an overview of the algebraic background, methods, and talk about how they can be adopted for such a project, and specifically for the data provided by this bioengineering group.

This paper is organized as follows. In Section 2 we will present an overview of the mathematics necessary to understand the reverse engineering framework. This includes ideals, primary decomposition, Stanley-Reisner theory, and pseudomonomials. In Section 3, we will introduce the concept of an algebraic model, which includes Boolean networks as a special case. Then we will ap-

ply Stanley-Reisner theory to the problem of reverse-engineering a network from data. Finally, we will generalize this from square-free monomial ideals to pseudomonomial ideals, in order to capture the difference between activiation and inhibition. Finally, in Section 4, we will explore how well these methods apply to synthetic and biological data and how they compare to existing biological methods.

Chapter 2

Mathematical Background

2.1 Some basic algebraic geometry

A *ring* is a set $R = \mathbb{F}[x_1, \ldots, x_n]$ that is closed under addition, subtraction, and multiplication. All of the rings we will see are commutative, which means that $xy = yx$ for all x and y. Also, we will primarily be dealing with polynomial rings, where the coefficients are from a *field*, such as the real numbers $\mathbb{R}$, the complex numbers $\mathbb{C}$, or a finite field $\mathbb{F}_p = \{0, 1, \ldots, p-1\}$, where arithmetic is done modulo a prime p.

An *ideal* is a subset of a ring that that is invariant under multiplication. In other words, for any $x \in I$ and $r \in R$ the product rx is in I. Every ideal I in a polynomial ring has a corresponding algebraic variety.

Definition 1. If I is an ideal in $R = \mathbb{F}[x_1, \ldots, x_n]$, then its *variety* is

$$V(I) = \{v \in \mathbb{F}^n : f(v) = 0, \text{ for all } f \in I\}.$$

In words, the variety of an ideal is the set of solutions to a system of polynomial equations, i.e., given an ideal of polynomials, the variety specifies the points that vanish on them.

Given a variety, or any subset of $\mathbb{F}^n$, we have the concept of a function vanishing, or in other words being 0, on all of its elements. Mathematically the *vanishing ideal* of $V \subseteq \mathbb{F}^n$ is

$$I(V) = \{f \in \mathbb{F}\{x_1, ..., x_n\} : f(x) = 0 \text{ for all } v \in V\}.$$

There are many different types of ideals and rings, one of which are the primary ideals.

Definition 2. Let R be a commutative ring. An ideal P is

- *prime* if $xy \in P$ implies $x \in P$ or $y \in P$,
- *primary* if $xy \in P$ implies $x \in P$ or $y^k \in P$ for some $k \in \mathbb{N}$.

As their names suggest, prime ideals of $\mathbb{Z}$ are of the form $p\mathbb{Z}$ for some prime number p, and primary ideals are of the form $p^k\mathbb{Z}$. In many rings, ideals can always be written as an intersection of primary ideals. This is called a *primary decomposition*.

Without going into details, this construction is analogous to the factorization of integers by prime powers. For example, the "ideal" version of $360 = 2^3 \cdot 3^2 \cdot 5$ is

$$360\mathbb{Z} = 8\mathbb{Z} \cap 9\mathbb{Z} \cap 5\mathbb{Z}.$$

Though primary decompositions are not unique and can often be difficult to construct, they are quite simple for the class of *square-free monomial ideals*, which we will define in the next section.

Theorem 3 (Lasker–Noether)**.** *Every Noetherian ring has a primary decomposition.*

We do not need to define Noetherian rings here, all that is important to know is that it is a large class that contains all polynomial rings over a field.

2.2 Stanley-Reisner theory and Alexander duality

A *monomial ideal* is simply an ideal generated by monomials. Every monomial can be defined by an *exponent vector* $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}_0^n$, where

$$x^\alpha := x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}.$$

A monomial is *square-free* if each $\alpha_i \in \{0, 1\}$. Ideals generated by square-free monomials are called *sqaure-free monomial ideals*, or *Stanley-Reisner ideals*. If x^α is square-free, then α canonically describes a subset of $[n] = 1, \ldots, n$. We will slightly abuse notation here to refer to α as both a vector and subset, but it should always be clear from the context. If an ideal contains x^α and x^β,

then it also contains $x^{\alpha\cup\beta}$. Conversely, if an ideal does not contain x^α and x^β, then it also does not contain $x^{\alpha\cap\beta}$. Finally if $\sigma \supseteq \alpha$ and $x^\alpha \in I$, then $x^\sigma \in I$.

An (abstract) *simplicial complex* is a collection Δ of subsets 2^X of a set X, which is additionally closed under the operation of taking subsets, and therefore also under intersections as well. Visually, a simplicial complex can be thought of as collection of vertices, edges, triangles, and higher dimensional simplices. A face with $k+1$ vertices is k-dimensional.

Definition 4. Let X be a finite set. A *simplicial complex* is a collection $\Delta \subseteq 2^X$ satisfying

$$\alpha \in \Delta \qquad \text{and} \qquad \beta \subseteq \alpha \qquad \Longrightarrow \qquad \beta \in \Delta.$$

Elements of Δ are called *faces*, and subsets not in Δ are called *non-faces*.

Usually we will take $X = [n] = \{1, \ldots, n\}$, and write subsets as strings. For example, $\alpha = 1346$ represents $\alpha = \{1, 3, 4, 6\}$. Every simplicial complex Δ canonically defines a square-free monomial ideal, generated by the non-faces, or equivalently, just the minimal non-faces. We write this as

$$I_{\Delta^c} = \big\langle x^\alpha \mid \alpha \notin \Delta \big\rangle.$$

Conversely, every square-free ideal I defines a simplicial complex where the faces correspond to the ideal not in I. We write this as

$$\Delta_{I^c} = \{\sigma \mid x^\sigma \notin I\}$$

Stanley-Reisner theory guarantees a bijection between square-free ideals and simplicial complexes, given by

$$I \longmapsto \Delta_{I^c}, \qquad \Delta \longmapsto I_{\Delta^c}.$$

This correspondence is known as Alexander duality.

For example, consider the simplicial complex Δ on $X = \{1, 2, 3, 4, 5\}$ shown in Figure 2.1. The ideal would be:

$$I_{\Delta^c} = \langle x_1x_4, x_1x_5, x_2x_5, x_3x_5, x_2x_3x_4 \rangle.$$

Primary decompositions of square-free monomial ideals are simple and combinatorial in nature. The primary components of I_{Δ^c} correspond to the complements of the maximal faces of Δ.

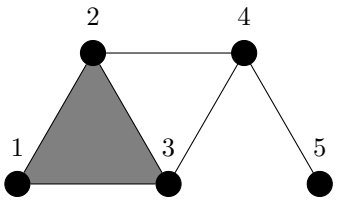


Figure 2.1: A visual representation of a simplicial complex Δ. The ideal can be generated from the non-faces. In this case the ideal would be $I_{\Delta^c} = \langle x_1x_4, x_1x_5, x_2x_5, x_3x_5, x_2x_3x_4 \rangle$

Specifically,

$$I_{\Delta^c} = \langle x^\alpha \mid \alpha \notin \Delta \rangle = \bigcap_{\substack{\sigma \in \Delta \\ \sigma \text{ max'l}}} \langle x_i \mid i \notin \sigma \rangle.$$

2.3 Pseudomonomials

Now that we have grasped the concept of a square-free monomial, we can begin to examine the concept of pseudomonomials.

Definition 5. A *pseudomonomial* in $\mathbb{F}[x_1, \ldots, x_n]$ is a product of terms of the form $x_i - a_i$, where all x_i are distinct, and $a_i \in \mathbb{F}$.

If we change variables to represent $x_i - a_i$ as y_i, then we have the appearance of a monomial. What we call pseudomonomials really should be "square-free pseudomonomials", as they always fit the definition of being square-free, however, other than [9], there has been very little work done with pseudomonomials that were not square-free and as such,we will be following the convention of dropping the square-free.

Examples of pseudomonomials include $x_1 - 1$ and $(x_1 - 1)x_2(x_3 - 5)$. In this project, all pseudomonomials are over $\mathbb{F}_3$ and moreover, our $a_i = \pm 1$.

Chapter 3

Min-sets of algebraic models

3.1 Algebraic models

3.1.1 Basic definitions

A number of phenomena from the life and physical sciences have been represented by agent or graph-based models, such as Boolean networks, cellular automata, and neural networks. Applications include everything from gene networks, disease models, blood flow, to chemical reaction networks. Many of these models involve simple functions that can be expressed algebraically as polynomials. Loosely speaking, an *algebraic model* is a collection of functions on a finite set $\mathbb{F} = \{0, 1, \ldots, n-1\}$ representing some real life phenomena. Each individual function f_i can be thought of as updating the state x_i of a particular node or entity. As such, each function is a mapping $f_i \colon \mathbb{F}^n \to \mathbb{F}$. We lose no generality in assuming that $\mathbb{F} = \mathbb{F}_p$, a finite field, because if not, we can simply expand it. An advantage of this assumption is that each function can be written as a polynomial, which opens the door to using tools from computational algebra for the analysis.

Definition 6. An *algebraic model* is a collection of functions $f_1, \ldots, f_n$, where $f_i \colon \mathbb{F}^n \to \mathbb{F}$.

Time is usually discretized, such as $t = 0, 1, 2, \ldots$. At time t, we will denote a *global system state* in $\mathbb{F}^n$ as a vector $x(t) = (x_1(t), \ldots, x_n(t))$. If we ignore time, then we can write it as $x = (x_1, \ldots, x_n)$.

The local functions in an algebraic model can be updated in several different ways, the most common being synchronously or asynchronously. A synchronous update defines a *finite dynamical*

system (FDS) map

$$f\colon \mathbb{F}^n \longrightarrow \mathbb{F}^n, \qquad f\colon x \longmapsto (f_1(x), \ldots, f_n(x)),$$

and iterating this map generates the dynamics. There are multiple ways of updating the functions asynchronously, or sequentially, and this leads to a multigraph called the *asynchronous automaton*. We will not define that here because it will not be used.

Any function from a finite set to itself, such as an FDS map, can be visualized with a directed graph, where the edges are of the form $(x, f(x))$. For an FDS, this is called the *phase space* or *state space*. The vertex set is $\mathbb{F}^n$ and every $x \in \mathbb{F}^n$ has exactly one directed edge $(x, f(x))$ from it.

When $\mathbb{F} = \mathbb{F}_2 = \{0, 1\}$, it is common to use Boolean logic to represent functions. These can also be written as polynomials, via the following equivalences:

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

This can be used to convert other Boolean operations, such as "exclusive OR" (XOR), into polynomials as well.

Example 7. Consider an algebraic model with four variables, namely $\{x_1, x_2, x_3, x_4\}$, and functions $f_1 = 1 + x_2$, $f_2 = x_1(x_2 + 1)$, $f_3 = x_1x_2x_3$, $f_4 = x_3 + x_4$. The phase space is the directed graph shown in Figure 3.1.

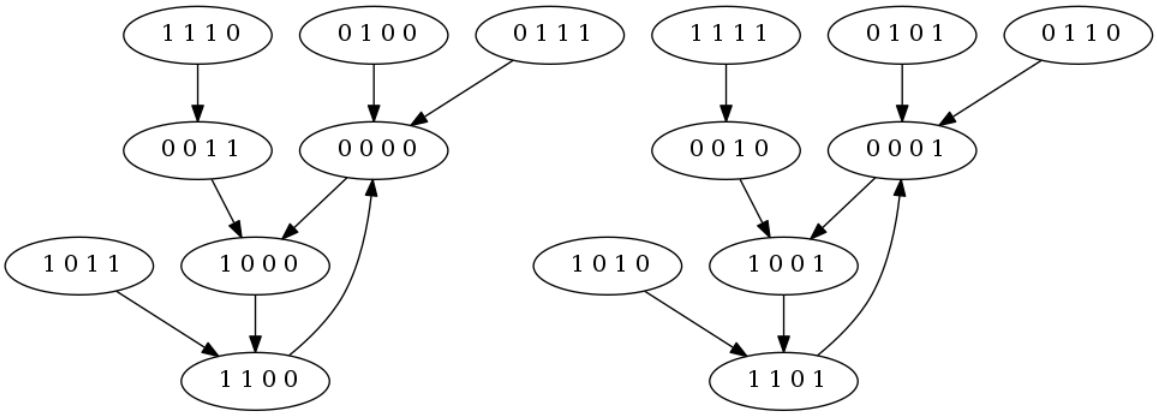


Figure 3.1: The phase space of the algebraic model from Example 7.

3.1.2 Wiring diagrams

Given an algebraic model $f_1, \ldots, f_n$, we can ask what functions depend on which variables, and we can encode this information with a *wiring diagram*. The vertex set consists of the variables $x_1, \ldots, x_n$ or just $1, \ldots, n$, and there is a directed edge from x_i to x_j if the function f_j depends on x_i. At node j, the set of incoming edges represents all variables that have an effect on the function f_j, and we call this set the *support* of the function. If our variables are indexed as, e.g., $x_1, \ldots, x_n$, then it can be convenient to speak of the support as a subset of $[n] = \{1, \ldots, n\}$. Sometimes we want to include more information such as the sign of the interaction. For example, in the Boolean case, if x_i (but not $\overline{x_i}$) appears in f_j then x_i affects f_j *positively*, and if $\overline{x_i}$ (but not x_i) appears, then it affects f_j *negatively*. Biologically, these represent activation and inhibition, respectively. Visually, we can adjust the head of the arrow to reflect this. For example, a positive edge is $i \longrightarrow j$ and a negative edge is $i \longdashv j$. For example, the wiring diagram motif on the left in Figure 3.2 shows a system of 3 variables, A, B, and C. It could arise from the function $f_C = A \wedge \overline{B}$, or from $f_C = A \vee \overline{B}$.

Figure 3.2: Wiring diagram examples.

Now consider an example of a system with two variables. If f_2 is constant, and $f_1 = x_1 + x_2$, which is simply the "exclusive OR" (XOR) of x_1 and x_2, then the wiring diagram is shown on the right in Figure 3.2.

Positive and negative interactions can be defined more generally, in any ordered or prime field $\mathbb{F}$. If $\mathbb{F} = \mathbb{F}_p$ then order the elements canonically as $0 < 1 < \cdots < p-1$. We say that that x_i affects f_j *positively* if

$$f_j(c_1, \ldots, c_{i-1}, c_i, c_{i+1}, \ldots, c_n) < f_j(c_1, \ldots, c_{i-1}, c_i', c_{i+1}, \ldots, c_n)$$

holds for some $c_i < c_i'$. Likewise, x_i affects f_j *negatively* if

$$f_j(c_1, \ldots, c_{i-1}, c_i, c_{i+1}, \ldots, c_n) > f_j(c_1, \ldots, c_{i-1}, c_i', c_{i+1}, \ldots, c_n)$$

holds for some $c_i < c_i'$. Notice that there are four possibilities: x_i could affect f_j not at all, positively but not negatively, negatively but not positively, or both positively and negatively. In terms of the wiring diagram, these correspond to no edge from x_i to x_j, a positive edge only, a negative edge only, or both a positive and negative edge, as in the XOR function.

As an example, consider the following Boolean function written two ways, both with Boolean logic and as a polynomial in $\mathbb{F}_2[x_1, \ldots, x_5]$:

$$f(x_1, x_2, x_3, x_4, x_5) = x_2 \wedge \overline{x_3} \wedge (x_4 \,\mathrm{XOR}\, x_5) = x_2(1 + x_3)(x_4 + x_5).$$

In this example, we see both a positive interaction of x_2, demonstrated by the variables and lack of the $+1$ in the algebra for x_2, as well as a negative interaction for x_3 seen by the complement of x_3 and the $x_3 + 1$ in the algebra. We also see that x_1 does not affect f at all. Finally, we see that x_3 and x_4 can affect f positively and negatively with the XOR gate. A function f_k is *unate* if there are no variables that affect f positively and negatively. Equivalently, no node has both positive and negative edges directed into x_k.

Wiring diagrams capture a key feature about algebraic models in a way that is simple to understand. Signed wiring diagrams and unate functions are well-suited to describe many biological interactions, as most individual interactions in a molecular network are either simple activations or inhibitions. Often these are apparent from their names, such as transcription factors or repressor proteins.

3.2 Reverse engineering from data

3.2.1 Model spaces of data

Sometimes we want to infer an algebraic model from partial information about the phase space, which might have been experimentally determined. There are methods to find the specific functions, but in this paper we are looking to infer just the network, as this is more biologically realistic. Here, inferring the network means determining which functions affect which variables. This in essence boils down to simply reconstructing the wiring diagram. Biologically, one example of this might be inferring a molecular network from observed data. Gene regulatory networks and protein-protein interaction networks are prime candidates because their systems are self-contained.

When finding the wiring diagram from partial data, instead of finding the "most likely" or getting one "best fit" answer, we actually will find all possible answers. One advantage of this is we know our solution is there among this collection. Of course, a drawback of this is that there might be hundreds or even thousands of solutions to sort through. While finding many answers reduces the likelihood of human or model error, it does make finding the true answer more of a challenge.

Let's first just focus on a single node j, and its unknown function f_j. What variable(s) does f_j depend on? Or equivalently, what are the incoming edges to node j in the wiring diagram? This is the question we seek to answer with computational algebraic tools. The goal is to figure out the network dependencies rather than the specific function.

A function f_j can be fully described by its truth table, i.e., an exhaustive table of all input-output pairs. Suppose we only have some, but not all, of the truth table entries. Specifically, such "partial data" will consist of pairs $(\mathbf{s}_i, f_j(\mathbf{s}_i))$ of some input vectors and their corresponding output values. Formally, we will define this to be a set of *data*, i.e.,

$$\mathcal{D} = \big\{(\mathbf{s}_1, t_1), \ldots, (\mathbf{s}_m, t_m)\big\}.$$

A function $f_j \colon \mathbb{F}^n \to \mathbb{F}$ is said to *fit the data* if $f_j(\mathbf{s}_i) = t_i$ for all $i = 1, \ldots, m$. The set of all functions that fit the data $\mathcal{D}$ is called the *model space*, and denoted

$$\mathrm{Mod}(\mathcal{D}) = \big\{f : \mathbb{F}^n \to \mathbb{F} \mid f(\mathbf{s}_i) = t_i,\ \forall i = 1, \ldots, m\big\}.$$

Example 8. Consider the partial data set in Table 3.1. The model space for this example would be

$$\mathrm{Mod}(\mathcal{D}) = \Big\{f \colon \mathbb{F}_2^4 \to \mathbb{F}_2 \mid f(0,0,1,0) = f(1,1,0,0) = f(1,1,1,1) = 0,\ f(0,1,1,0) = 1\Big\}.$$

x	0010	1100	1111	0110
$f(x)$	0	0	0	1

Table 3.1: An unknown 4-variable Boolean function with the above partial truth table.

3.2.2 Feasible and disposable sets of variables

Informally, given a data set $\mathcal{D}$, a subset $\alpha \subseteq [n]$ is *feasible* if there is some function that fits the data that depends only on variables with indices in α. The formal definition follows.

Definition 9. Let $\mathcal{D}$ be a data set. The set $\alpha \subseteq [n]$ is *feasible* if there is some f in $\text{Mod}(\mathcal{D})$ for which $\text{supp}(f) \subseteq \alpha$.

A set $\alpha \subseteq [n]$ is *disposable* if there is some function that fits the data that depends on no variables with indices in α. In other words, if we can discard all of these variables and still find a function that fits the data.

Definition 10. Let $\mathcal{D}$ be a data set. Then $\alpha \subseteq [n]$ is *disposable* if there is some f in $\text{Mod}(\mathcal{D})$ for which $\text{supp}(f) \cap \alpha = \emptyset$.

If α is not feasible with respect to $\mathcal{D}$, then we say that it is infeasible. If it is not disposable, then we say it is non-disposable. Note that feasible and disposable are *not* opposite concepts, but they are related. A set can be both feasible and disposable, or neither. Disposable sets are clearly closed under intersection and subsets, and so they form a simplicial complex, that we will denote by $\Delta_{\mathcal{D}}$. Non-disposable sets are closed under unions and supersets. Therefore, they define a square-free monomial ideal in $\mathbb{F}[x_1, \ldots, x_n]$, where the generators x^α are indexed by the non-disposable sets α. This holds because of the simple observation that if x^α and x^β are in an ideal I, then $x^{\alpha \cup \beta}$ is also in I. In other words, the supports of the set of square-free monomials in I are closed under unions.

To define our ideal of non-disposable sets, we will construct a convenient generating set. If two output values from a data set $\mathcal{D}$ differ, say $t \neq t'$, any function f that fits the data must depend on at least one coordinate where the input vectors $\mathbf{s}$ and $\mathbf{s}'$ differ. As such, if we take the product of the corresponding variables, to get a polynomial defined as

$$m(\mathbf{s}, \mathbf{s}') = \prod_{s_i \neq s'_i} x_i,$$

then $\text{supp}(m(\mathbf{s}, \mathbf{s}'))$ is non-disposable by construction. All of these polynomials generate a square-free monomial ideal that we will call the *ideal of non-disposable sets*. This follows directly from Stanley-Resiner theory [5].

Definition 11. Given a set $\mathcal{D}$ of data, its *ideal of non-disposable sets* is

$$I_{\triangle_{\mathcal{D}}^c} = \big\langle m(\mathbf{s}, \mathbf{s}')) \mid t < t' \big\rangle.$$

3.2.3 Min-sets and primary decompositions

Sometimes when we have large quantities of data or variables, we seek to cut out the data that is not actually helpful to our cause. Just as a computer scientist might prune branches from a tree to save time, we cut out the excess data by forming what is called a min-set for short. Loosely speaking, a *min-set* of $\mathcal{D}$ is a minimal set of variables on which a function that fits that data can depend.

Definition 12. A subset $\alpha \subseteq [n]$ is a *min-set* of $\mathcal{D}$ if it is a *minimal feasible set*, or equivalently, if its complement $\overline{\alpha}$ is a *maximal disposable set.*

Recall that feasible sets correspond to the complements of the faces of the simplicial complex $\Delta_{\mathcal{D}}$ of disposable sets. Therefore, we can interpret a min-set of data $\mathcal{D}$ in terms of its simplicial complex $\Delta_{\mathcal{D}}$.

Theorem 13. *A subset $\alpha \subseteq [n]$ is a min-set of $\mathcal{D}$ if its complement $\overline{\alpha}$ is a maximal face of $\Delta_{\mathcal{D}}$.*

Finally, we can express min-sets algebraically by the primary components of the ideal $I_{\Delta_{\mathcal{D}}^c}$ of non-disposable sets. Recall that the primary components are indexed by the complements of maximal faces. The complement of a maximal face will be denoted $\overline{\alpha}$.

Theorem 14. *A subset $\alpha = \{\alpha_1, \ldots, \alpha_k\} \subseteq [n]$ is a min-set of $\mathcal{D}$ if and only if $\langle \alpha_1 \ldots, \alpha_k \rangle$ is a primary component of the ideal $I_{\Delta_{\mathcal{D}}^c}$ of non-disposable sets.*

In summary, the relationship between the min-sets of $\mathcal{D}$, the faces of the simplicial complex $\Delta_{\mathcal{D}}$, and the primary components of the ideal $I_{\Delta_{\mathcal{D}}^c}$ of non-disposable sets, is illustrated by the following string of equalities:

$$I_{\Delta_{\mathcal{D}}^c} = \bigcap_{\substack{\sigma \in \Delta_{\mathcal{D}} \\ \sigma \text{ max'l}}} \big\langle x_i \mid i \notin \sigma \big\rangle \quad = \bigcap_{\substack{\sigma \text{ max'l} \\ \text{disposable}}} \big\langle x_i \mid i \notin \sigma \big\rangle \quad = \bigcap_{\substack{\alpha \text{ min'l} \\ \text{feasible}}} \big\langle x_i \mid i \in \alpha \big\rangle \quad = \bigcap_{\alpha \text{ min-set}} \big\langle x_i \mid i \in \alpha \big\rangle.$$

Example 15. For an example, let's consider again the data from Example 8, which consisted of an unknown Boolean function on four variables, where exactly 4 of the 16 entries in the truth table are

known, and are given again in Table 3.2.

x	0010	1100	1111	0110
$f(x)$	0	0	0	1

Table 3.2: An unknown 4-variable function with the above truth table.

This defines the following set of data

$$\mathcal{D} = \big\{(0010, 0), (1100, 0), (1111, 0), (0110, 1)\big\}. \tag{3.1}$$

Let $\mathbf{s}_1 = 0010$, $\mathbf{s}_2 = 1100$, $\mathbf{s}_3 = 1111$, and $\mathbf{s}_4 = 0110$ which means that $0 = t_1 = t_2 = t_3 < t_4 = 1$. There are $2^{12} = 4096$ functions that fit the data, as there are $16 - 4 = 12$ missing entries in the truth table. Next, we need to compute the monomials $m(\mathbf{s}_i, \mathbf{s}_j)$, for each pair such that $t_i < t_j$. Each of these is the product of the variables in the coordinates in which $\mathbf{s}_i$ and $\mathbf{s}_j$ differ, so

$$m(\mathbf{s}_1, \mathbf{s}_4) = x_2, \qquad m(\mathbf{s}_2, \mathbf{s}_4) = x_1x_3, \qquad m(\mathbf{s}_3, \mathbf{s}_4) = x_1x_4.$$

The ideal of non-disposable sets is thus

$$I_{\Delta^c_{\mathcal{D}}} = \big\langle m(\mathbf{s}_1, \mathbf{s}_4),\, m(\mathbf{s}_2, \mathbf{s}_4),\, m(\mathbf{s}_3, \mathbf{s}_4)\big\rangle = \big\langle x_2,\, x_1x_3,\, x_1x_4\big\rangle.$$

From this information, we can determine both the feasible and non-disposable sets. First, we can visualize the non-disposable sets in a Boolean lattice, as shown in Figure 3.3. The feasible sets are highlighted in yellow, and the non-disposable sets are highlighted in red. The minimal non-disposable sets are highlighted in darker red, and the maximal disposable sets are highlighted in darker yellow.

Taking the complement of each node results in the upside-down Boolean lattice in Figure 3.4. The complement of the non-disposable (red) nodes are the infeasible sets, and the complement of the disposable (yellow) nodes are the feasible sets. The complements of the maximal disposable sets (the dark yellow nodes) are thus the minimal feasible sets, which are the min-sets by definition.

We can then see that the min-sets are $\{x_1, x_2\}$ and $\{x_2, x_3, x_4\}$. By Stanley-Reisner theory, these are the generators of the primary components of the ideal of non-disposable sets. In other

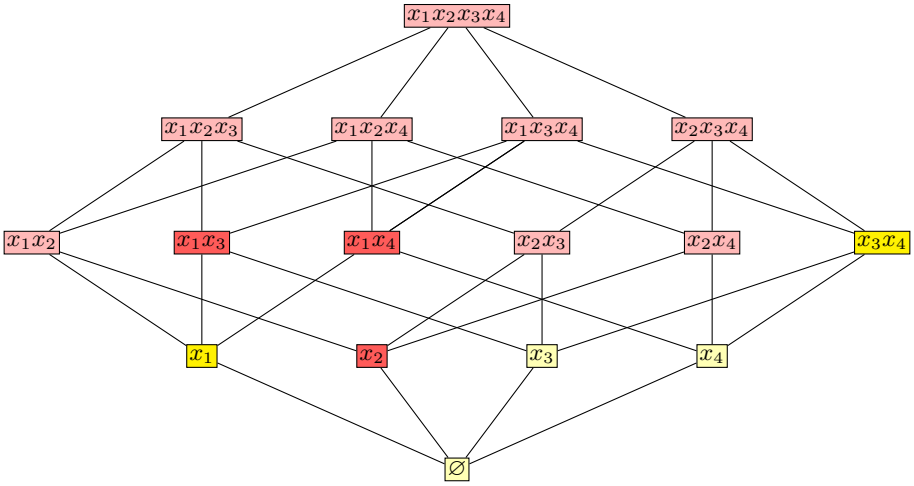


Figure 3.3: The red nodes are the non-disposable sets. The darker one are the minimal non-disposable sets, which represent the generators of the ideal $I_{\Delta_{\mathcal{D}}^c}$. The yellow nodes are the disposable sets, and the maximal ones are darker.

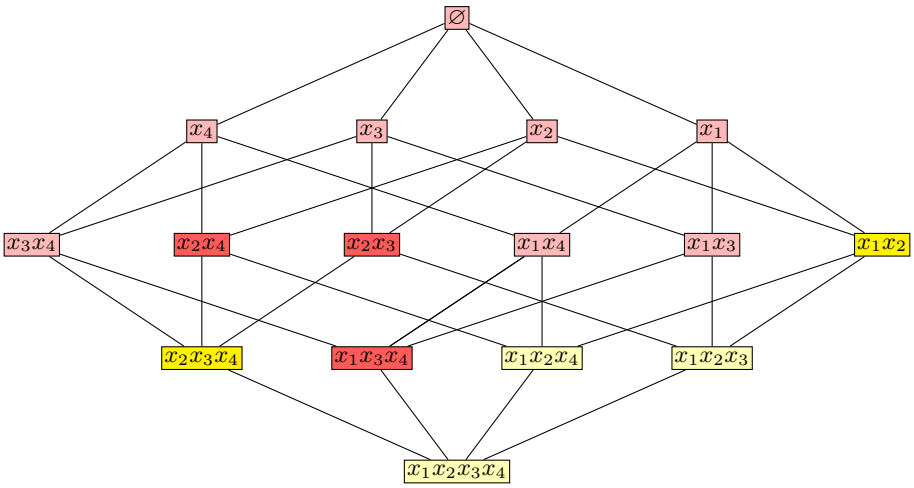


Figure 3.4: The complement of the previous diagram. The complement of the disposable sets are the feasible sets, which are in yellow. The minimal feasible sets are darker, and these are the *min-sets*.

words,

$$I_{\Delta_{\mathcal{D}}^c} = \big\langle m(\mathbf{s}_1, \mathbf{s}_4),\, m(\mathbf{s}_2, \mathbf{s}_4),\, m(\mathbf{s}_3, \mathbf{s}_4) \big\rangle = \big\langle x_2,\, x_1x_3,\, x_1x_4 \big\rangle = \langle x_1, x_2 \rangle \cap \langle x_2, x_3, x_4 \rangle.$$

This means that any 4-variable Boolean function with partial truth table given in Table 3.2 must depend minimally on either $\{x_1, x_2\}$ or $\{x_2, x_3, x_4\}$.

Instead of constructing the Boolean lattices to find the min-sets like we did in Figures 3.3 and 3.4, we could have alternatively used a computational algebra software package such as "Macaulay2"[3] or "Sage"[4] to compute the primary decomposition of the ideal of non-disposable sets. For example, using Macaulay2, the input of

```
R = ZZ/2[x1,x2,x3,x4]
I_nonDisp = ideal(x2,x1*x3,x1*x4)
primaryDecomposition I_nonDisp
```

gives an output of

```
R
PolynomialRing
Ideal of R
{ideal (x1,x2), ideal (x2, x3, x4) }
List
```

Now that we have the min-sets, we can examine functions that fit the data and determine which one is the best fit. Consider the case of the min-set $\{x_1, x_2\}$. This tells us that there exists a function fitting the data that only depends on x_1 and x_2. There are $2^{2^2} = 16$ possible Boolean functions on two variables. Of these, two are constant, two depend only on x_1, and two depend only on x_2. This leaves ten functions that depend on both x_1 and x_2. From the data, we can clearly see that $f(x) = 1$ when $x_1 = 0$ and $x_2 = 1$. This restriction gives us that the function must be $\overline{x_1} \wedge x_2$, as shown in Table 3.3.

$x = x_1x_2$	00	11	11	01
$f(x)$	0	0	0	1

Table 3.3: The function $f(x_1, x_2) = \overline{x_1} \wedge x_2$ fits the data from Table 3.2.

Next consider the three variable min-set $\{x_2, x_3, x_4\}$. If we restrict the the truth table to these values, we get Table 3.4. A truth table on three variables has $2^3 = 8$ entries, and so there are

$x = x_2x_3x_4$	001	100	111	110
$f(x)$	0	0	0	1

Table 3.4: The function $f(x_2, x_3, x_4) = x_2 \wedge x_3 \wedge \overline{x_4}$ fits the data from Table 3.2.

$2^{2^3} = 256$ possible Boolean functions. However, of these, two are constant and six depend on only one variable; such as $f(x_i) = x_i$, and $f(x_i) = x_i + 1$. There are ten Boolean functions that depend

on exactly two variables, and there are $\binom{3}{2} = 3$ possible pairs of variables to choose. Overall, this means that there are 30 functions that depend on exactly two variables, leaving us 218 functions to consider. Once again, since the only specified time that $f(x) = 1$ is if $x_2x_3x_4 = 110$, we can narrow this down to a single function, $x_2 \wedge x_3 \wedge \overline{x_4}$. This is not always the case however, as we will often have partial data where more than one function is possible. When this occurs, finding the signed min-set is the most helpful course of action.

3.3 Signed min-sets

The framework in the previous section does not take into account the sign of the interactions. If we restrict our view to unate functions, it should be clear how to define signed min-sets. Each variable x_i either does not appear, appears as a positive interaction, or as a negative interaction. Motivated by the Boolean setting, we will denote these last two cases by x_i and $\overline{x_i}$, respectively. For example, a min-set $\{x_1, x_2\}$ means that the function depends on at least x_1 and x_2, whereas a signed min-set $\{x_1, \overline{x_2}\}$ means that the function depends on at least x_1 in a positive manner (x_1 activates the node) and x_2 in a negative or inhibitory manner. The next natural question is how to find the signed min-sets. In the ordinary or "non-signed" case, we would find the primary decomposition and use it to find the ideal of non-disposable sets. Monomials would be used to describe subsets of variables, without any regard to which interactions are positive or negative. One way we can encode this into signed min-sets is to actually use pseudomonomials instead of monomials. While a monomial is a product of x_i's, a *pseudomonomial* is a product of $(x_i - a_i)$'s. We will use $(x_i - 1)$ to represent a positive change (activation) and $(x_i + 1)$ to represent a negative change (inhibition). To put it in the notation above, $x_i - 1$ would correspond to x_i and $x_i + 1$ would correspond to $\overline{x_i}$. By considering signs, we can understand biological systems in more detail, such as whether a gene product is acting as an activator or as a repressor. In addition, when working backwards from data, by including the signs, we can infer not only which variables a function is dependent on, but whether it is an activation or inhibition.

We can define a discrete analogue of the partial derivative. Assume $s_i, s_i' \in \mathbb{F}_p = \{0, 1, \ldots, p-$

1}, which is ordered canonically. Then define

$$\partial_i(\mathbf{s}, \mathbf{s}') = \begin{cases} 1 & s_i' > s_i \\ -1 & s_i' < s_i \\ 0 & s_i' = s_i. \end{cases}$$

When we were finding min-sets, we defined monomials $m(\mathbf{s}_i, \mathbf{s}_j)$ whose supports were non-disposable by construction. Similarly, we can define a "signed version" of this using pseudomonomials. Specifically, define

$$p(\mathbf{s}, \mathbf{s}') = \prod_{s_i \neq s_i'} \big(x_i - \partial(\mathbf{s}, \mathbf{s}')\big).$$

While the support of $p(\mathbf{s}, \mathbf{s}')$ is non-disposable just like $m(\mathbf{s}, \mathbf{s}')$, this product encodes more, because it tells us the signs of the individual interactions. In the unsigned case, these polynomials generate the ideal $I_{\triangle_{\mathcal{D}}^c}$ of non-disposable sets, and by Stanley-Reisner theory, the primary components correspond to the min-sets. It turns out that an analogous result holds for the signed case, but it does not follow from Stanley-Reisner theory, but rather had to be derived from scratch [10].

Definition 16. Let $\mathcal{D}$ be a data set. Define the *ideal of signed non-disposable sets* as

$$J_{\triangle_{\mathcal{D}}^c} = \big\langle p(\mathbf{s}, \mathbf{s}') \mid t < t' \big\rangle.$$

In the unsigned min-set case, if a primary component is $\langle x_{i_1}, \ldots, x_{i_k} \rangle$, then the corresponding min-set is $\{x_{i_1}, \ldots, x_{i_k}\}$. Similarly, for a signed primary component

$$\big\langle x_{i_1} \pm 1, \ldots, x_{i_k} \pm 1 \big\rangle, \tag{3.2}$$

the corresponding unsigned min-set is still $\{x_{i_1}, \ldots, x_{i_k}\}$, and the signed min-set is formed by replacing x_{i_j} with $\overline{x_{i_j}}$ if $x_{i_j} + 1$ appears in Eq. (3.2).

Let's return to the data set $\mathcal{D} = \big\{(0010, 0), (1100, 0), (1111, 0), (0110, 1)\big\}$ from Example 15, which had two unsigned min-sets, $\{x_1, x_2\}$ and $\{x_2, x_3, x_4\}$, and compute the signed min-sets. From above, we can see that $\partial_2(\mathbf{s}_1, \mathbf{s}_4) = 1$, thus x_2 becomes $x_2 - 1$ in the signed case. Similarly, $\partial_1(\mathbf{s}_2, \mathbf{s}_4) = -1$ and $\partial_3(\mathbf{s}_2, \mathbf{s}_4) = 1$, thus $x_1 x_3$ becomes $(x_1 + 1)(x_3 - 1)$. Finally, $\partial_1(\mathbf{s}_3, \mathbf{s}_4) = -1$, and $\partial_4(\mathbf{s}_3, \mathbf{s}_4) = -1$, and as such $x_1 x_4$ becomes $(x_1 + 1)(x_4 + 1)$. Thus, our pseudomonomials $p(\mathbf{s}_i, \mathbf{s}_j)$

are

$$p(\mathbf{s}_1, \mathbf{s}_4) = x_2 - 1, \qquad p(\mathbf{s}_2, \mathbf{s}_4) = (x_1 + 1)(x_3 - 1), \qquad p(\mathbf{s}_3, \mathbf{s}_4) = (x_1 + 1)(x_4 + 1).$$

Putting this into "Macaulay2" to compute the primary decomposition yields

$$\begin{aligned} J_{\triangle_{\mathcal{D}}^c} &= \big\langle p(\mathbf{s}_1, \mathbf{s}_4),\, p(\mathbf{s}_2, \mathbf{s}_4),\, p(\mathbf{s}_3, \mathbf{s}_4) \big\rangle \\ &= \big\langle (x_2 - 1), (x_1 + 1)(x_3 - 1), (x_1 + 1)(x_4 + 1) \big\rangle \\ &= \big\langle x_1 + 1,\, x_2 - 1 \big\rangle \cap \big\langle x_2 - 1,\, x_3 - 1,\, x_4 + 1 \big\rangle. \end{aligned}$$

Thus, the signed min-sets are $\{\overline{x_1}, x_2\}$ and $\{x_2, x_3, \overline{x_4}\}$. This concurs with the results we found earlier of the functions $f = \overline{x_1} \wedge x_2$ and $\{x_2, x_3, x_4\}$, $f = x_2 \wedge x_3 \wedge \overline{x_4}$ that fit the data from Table 3.2.

While the signed min-sets give us the same variable dependencies, they actually show significantly more information about what is going on in the system, as we can see exactly which variables are positive and which are negative. If this were a biological system then we would then know which variables were activators and which were inhibitors, and that could be very helpful in inferring unknown gene networks.

Chapter 4

Applications to biological networks

4.1 Existing network reconstruction methods with perturbations

One challenging problem that has become a widely researched area in systems biology is how to infer gene network systems from experimental data. One broad approach is to analyze data resulting from perturbations of the system.

In a paper published in 2002 [6], the authors developed an approach called *Modular Response Analysis* (MRA), which analyzed experimental data collected from systematic perturbations at fixed snapshots in time. Two years later, a variant of this, called *Dynamic Modular Response Analysis* (DMRA) [8], analyzed systematic perturbation data but over a time series. Though DMRA is an improvement over MRA, one significant drawback, which has prevented it from going mainstream, is that it does not handle noise in the data well.

In a recent 2018 preprint [7], a new time series perturbation method was proposed, and shown to be more robust to noise. In this paper, the network dynamics were modeled by a system of differential equations. The strength of the interaction of the edges in the network are determined

by the Jacobian matrix. Recall that if we have a 2×2 system

$$\frac{dx_1}{dt} = f_1(x_1, x_2)$$
$$\frac{dx_2}{dt} = f_2(x_1, x_2)$$

then its Jacobian is

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix} = \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix}.$$

The coefficient F_{ij} determines strength of the interaction and is used to draw conclusions about dependencies between variables. For example, the sign distinguishes activation vs. inhibition, and interactions are ignored if they are below a certain threshold.

Determining the edges in the network was done by solving the system – i.e., the predictor was determined by a system of linear equations and then they solved for the edge weights directly. However, when noise that is common in these biological settings was introduced, solving directly had to be replaced with a least squares estimation. The authors of [7] test and validate their method using synthetic two and three node networks constructed in Matlab, and then knocking out individual nodes, both with and without noise. This method turns out to be quite accurate, especially with noise. They then apply their methods to reconstruct the ERK and AKT pathways, which are important in the study of mammalian signaling.

One disadvantage of this technique is that there can sometimes be a causation vs. correlation error, meaning that a strong correlation between two variables might be coincidental, or the result of a third variable, and not because they depend on each other functionally. This can cause false positives in the network, and is where algebraic techniques can potentially be useful. Algebraic models are tailor made for determining precisely which functions depend on which variables, thus eliminating the false positives. It also seems like they might be able to differentiate between AND and OR gates in the individual functions. Of course, a downside is that they theory is difficult to apply to real data, and it is not clear how robust to noise it will be. *One of the goals of this section is to try to apply the previously developed algebraic methods and see how they compare with the published methods in [7].*

4.2 Processing of time series data

All time series data in our setting, whether experimentally determined or synthetically generated, consists of a finite number of tuples of floating point values, corresponding to, e.g., gene expression levels, or protein and enzyme concentrations. Such data can be easily graphed in software such as Matlab, Mathematica, or R, all of which have the capability of quickly finding the best fitting polynomial or piecewise linear function. From this, one can also infer the general trend of said data, such as whether it is increasing, decreasing, or roughly constant. For example, consider the time series data graphs shown in Figure 4.1, which were rendered by Matlab from synthetic data generated in [7].

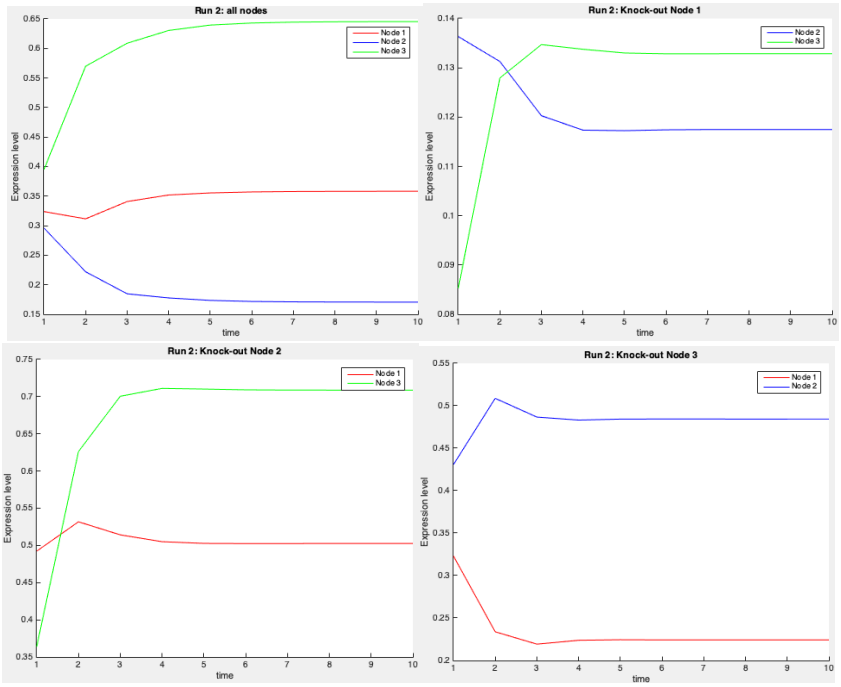


Figure 4.1: Each of these four graphs were made in Matlab using synthetic time series data from [7].

There are many ways to discretize data, for example, into Boolean, ternary, or some other finite set $\mathbb{Z}_k = \{0, 1, \ldots, k-1\}$. One example of how to do this, in the setting of algebraic models, was published in [2]. Another way to approximate, or discretize that data is to just look at general trends: up, down, or flat. Figure 4.2 illustrates this for the example from Figure 4.1.

The trend data in Figure 4.1 can be encoded over $\mathbb{F}_3 = \{0, 1, 2\}$ by sectioning the output into 3 parts, namely high, medium, and low, corresponding to $2, 1$ and 0, respectively. Clearly, $\mathbb{F}_2$ will

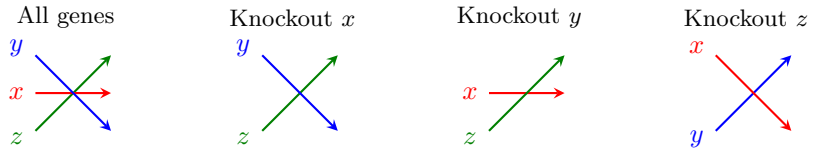


Figure 4.2: The arrows show the general trends in the time series data from Figure 4.1

not suffice for this, but also, we only need $\mathbb{F}_3$, because we are examining the general trends. Small variances in the data are often caused by noise, so discretizing the data into more than three levels may actually lead to false positives. For example, it seems reasonable that the red line (Node 1) in the two left-most graphs in Figure 4.1 should be considered constant. Discretizing into more levels might pick up small deviations that we do not want to consider. Figure 4.3 shows the dizcretization of the trend data in 4.2. An arrow pointing downward will be labeled as $(2, 1, 0)$, whereas an arrow pointing upward will be labeled as $(0, 1, 2)$. Any constant arrows will be labeled based on what level they remain constant on, as such the choices are $(0, 0, 0)$, $(1, 1, 1)$, or $(2, 2, 2)$.

t	x	y	z
0	1	2	0
1	1	1	1
2	1	0	2

t	y	z
0	2	0
1	1	1
2	0	2

t	x	z
0	1	0
1	1	1
2	1	2

t	x	y
0	2	0
1	1	1
2	0	2

Figure 4.3: Discretized time series of the synthetic data from in Figures 4.1 and 4.2.

4.3 Min-sets from time series data

Once we have constructed and discretized the general trend data, our next goal is to represent it by pseudomonomial ideals, and then find the (signed) min-sets. In an algebraic model we use $\mathbb{F} = \{0, \ldots, p-1\}$ but here we want to emphasis the positive and negative signs, and as such, as in Section 3, we will work over $\mathbb{F}_3 = \{0, 1, -1\}$.

Recall that we encode a change in the i^{th} coordinate by $\partial_i(\mathbf{s}, \mathbf{s}') \in \mathbb{F}_3$, with a value of 1 meaning increase, -1 meaning decrease, and 0 meaning constant. For example, a time series data with a positive trend at node i is represented by $x_i - 1$, just as we represented an activation earlier, whereas a negative trend is represented by $x_i + 1$, an inhibition. Recall that our individual

pseudomonomials are defined as

$$p_j(x) = \prod_{a_i \neq 0} (x_i - a_i).$$

	t	x	y	z
$\vec{\mathbf{s}}_1$	0	1	2	0
$\vec{\mathbf{t}}_1 = \vec{\mathbf{s}}_2$	1	1	1	1
$\vec{\mathbf{t}}_2$	2	1	0	2

Figure 4.4: We can see how to break down the data by labeling the input and output vectors.

We start with the trend data and write it in terms of input-output *vectors*. Then we can break it into n individual data sets of input vectors and output values. For example, the discretized data from Figure 4.4 is

$$\begin{aligned}
\mathcal{D} &= \{(\mathbf{s}_1, \mathbf{t}_1), (\mathbf{s}_2, \mathbf{t}_2)\} \\
\mathcal{D}_x &= \big\{(\mathbf{s}_1, t_{11}), (\mathbf{s}_2, t_{21})\big\} = \big\{((1,2,0),1), ((1,1,1),1)\big\} \\
\mathcal{D}_y &= \big\{(\mathbf{s}_2, t_{12}), (\mathbf{s}_2, t_{22})\big\} = \big\{((1,2,0),1), ((1,1,1),0)\big\} \\
\mathcal{D}_z &= \big\{(\mathbf{s}_2, t_{13}), (\mathbf{s}_2, t_{23})\big\} = \big\{((1,2,0),1), ((1,1,1),2)\big\}.
\end{aligned}$$

However, looking at just the time series data when all nodes are online is only part of the picture. We must also consider the gene knockout data. If we have a system with three nodes, then we will have four time series to examine: One with all three nodes, and the ones with each of the individual genes knocked out. Once we analyze the trends for each node in each graph, we can encode each with a pseudomonomial, and construct the ideal of signed non-disposable sets. Finally, we use a software package to compute the primary decomposition and get the min-sets.

Example 17. Consider again the system of three nodes x, y, z from Figure 4.3, and graphed in Figure 4.1 with the trends in Figure 4.2. Note that we have four graphs here, meaning that we have the complete knockout data.

Adding gene knockouts to the data above yields:

$$\mathcal{D}_x = \big\{((1,2,0),1),((1,1,1),1),((1,0,0),1),((1,0,1),1),((2,0,0),1),((1,1,0),0)\big\}$$
$$\mathcal{D}_y = \big\{((1,2,0),1),((1,1,1),0),((2,0),1),((0,2,0),1),((0,1,1),0),((2,0,0),1),((1,1,0),2)\big\}$$
$$\mathcal{D}_z = \big\{((1,2,0),1),((1,1,1),2),((0,2,0),1),((0,1,1),2),((1,0,0),1),((1,0,1),2)\big\}.$$

From these trends we can assign a pseudomonomial $x_i \pm 1$ to each node, resulting in the following 3 ideals:

$$J_x = \big\langle (x+1)(y-1) \big\rangle, \qquad J_y = \big\langle (y+1)(z-1), (x+1)(y-1) \big\rangle, \qquad J_z = \big\langle (y+1)(z-1), (z-1) \big\rangle.$$

Which, when plugged into Macaulay2, yields that the min-sets are:

- Gene x: $\{x+1\}$, $\{y+1\}$
- Gene y: $\{x+1, z+1\}$, $\{y-1\}$
- Gene z: $\{z-1\}$.

We should note here that if the data is generated by non-unate functions, then the supports of the function may not be reflected by the signed min-sets.

4.4 Inferring a feed-forward loop

One of the synthetic networks that the authors of [7] tested their algorithm on is a feed-forward loop where the types of interactions – activation vs. inhibition, and AND vs. OR gate, are unknown. There are 8 such feed-forward loops, and 16 cases in total because each one can be an AND or an OR. We will describe these below.

We contacted the authors and received the full time series data for all 16 different networks. These were generated from non-linear differential equations, and each node additionally had a decay term. Our goal was to determine which network was which purely algebraically. Each network consists of 3 nodes and can be described with 2 functions, f_2 and f_3, because f_1 is a source not regulated by the other nodes, other than an artificial external force that remained constant throughout. The

function f_2 is dependent on node 1 and as such, has 2 options, excluding the decay term: $f_2 = x_1$, and $f_2 = \overline{x_1}$. Function f_3 depends on both x_1 and x_2 and as such, has 8 different choices:

$$f_3 = \pm x_1 \Diamond \pm x_2,$$

where $\pm x_i$ is a placeholder for either x_i or $\overline{x_i}$, and $\Diamond$ is a placeholder for either $\wedge$ or $\vee$, The 8 choices for possible wiring diagram, which describe these 16 different feed-forward loops, are depicted in Figure 4.5. Each data set, and wiring diagram, was labeled by the authors of [7] with a string of the form $t_1t_2t_2t_4$, where $t_i \in \{1, 2\}$.

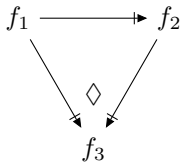


Figure 4.5: The wiring diagram for the [7] data is one of 16 different feed-forward loops. Each edge can be either positive or negative, and the function f_3 can either be AND or OR.

Our first task was to graph this data in Matlab. Figure 4.6 shows these graphs for two of the 16 data sets, data set 1111 and data set 2112. The other 14 graph sets can be found in the Appendix.

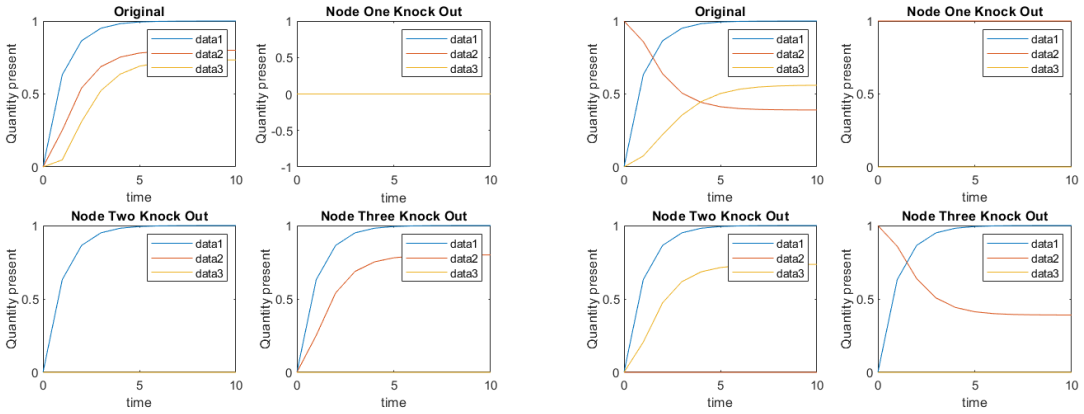


Figure 4.6: These eight graphs were generated in Matlab from data set 1111 (left) and data set 2112 (right) from [7].

From there, we were able to find the trends and generate the min-sets for each data set. In

this section, we will carry out the details for data sets 1111 and 2112.

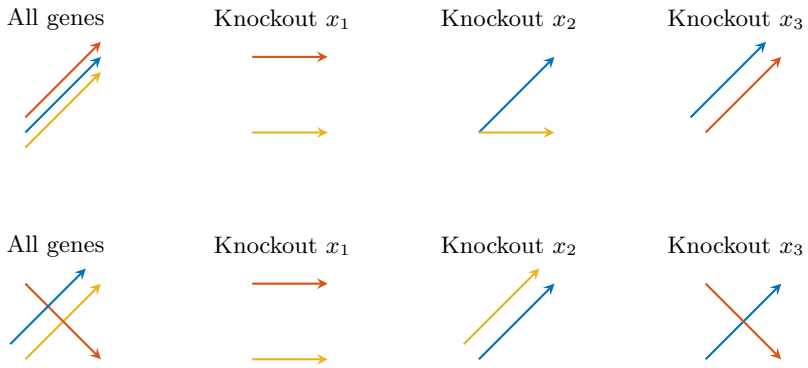


Figure 4.7: The arrows show the general trends in the time series data from Figure 4.6. The red arrows correspond to x_1, the blue arrows to x_2, and the yellow arrows to x_3.

Next, we discretize the trend data from Figure 4.7 over $\mathbb{F}_3$, which is shown in Figure 4.8. The other 14 cases can be found in the Appendix.

t	x_1	x_2	x_3
0	0	0	0
1	1	1	1
2	2	2	2

t	x_2	x_3
0	2	0
1	2	0
2	2	0

t	x_1	x_3
0	0	0
1	1	0
2	2	0

t	x_1	x_2
0	0	0
1	1	1
2	2	2

t	x_1	x_2	x_3
0	0	2	0
1	1	1	1
2	2	0	2

t	x_2	x_3
0	2	0
1	2	0
2	2	0

t	x_1	x_3
0	0	0
1	1	1
2	2	2

t	x_1	x_2
0	0	2
1	1	1
2	2	0

Figure 4.8: Time series data for data sets 1111 and 2112.

Using the discretized data from Figure 4.8, the signed ideals of non-disposable sets, for data set 1111, are

$$J_{x_1} = \langle (x_1-1)(x_2-1)(x_3-1), (x_1-1), (x_1-1)(x_2-1) \rangle$$
$$J_{x_2} = \langle (x_1-1)(x_2-1)(x_3-1), (x_1-1)(x_2-1) \rangle,$$
$$J_{x_3} = \langle (x_1-1)(x_2-1)(x_3-1) \rangle.$$

Figure 4.9: Wiring diagrams for data sets 1111 (left) and 2112 (right

For data set 2112, the signed ideals of non-disposable sets are

$$J_{x_1} = \big\langle (x_1-1)(x_2+1)(x_3-1), (x_1-1)(x_3-1), (x_1-1)(x_2-1) \big\rangle,$$
$$J_{x_2} = \big\langle (x_1-1)(x_2+1)(x_3-1), (x_1-1)(x_2+1) \big\rangle,$$
$$J_{x_3} = \big\langle (x_1-1)(x_2+1)(x_3-1), (x_1-1)(x_3-1) \big\rangle.$$

We used computational algebra software, in this case [3], to compute the primary decompositions of these ideas, and the primary components give us the signed min-sets. For data set 1111, these are

- Gene x_1: $\{x_1 - 1\}$
- Gene x_2: $\{x_1 - 1\}$, $\{x_2 - 1\}$
- Gene x_3: $\{x_1 - 1\}$,$\{x_2 - 1\}$,$\{x_3 - 1\}$,

and for data set 2112, they are

- Gene x_1: $\{x_1 - 1\}$,$\{x_2 - 1, x_3 - 1\}$
- Gene x_2: $\{x_1 - 1\}$, $\{x_2 + 1\}$
- Gene x_3: $\{x_1 - 1\}$,$\{x_3 - 1\}$.

The other 14 cases can be found in the Appendix.

In this case the functions (f_2, f_3) are $(x_1, x_1 \wedge x_2)$ and $(\overline{x_1}, x_1 \vee x_2)$ respectively. These functions have wiring diagrams shown in Figure 4.9.

Table 4.1 shows the summarized results of all 16 data sets. The graphs, discretized data, and ideals for the other 14 are in the Appendix. Also given are the corresponding wiring diagrams.

Equation (f_2, f_3)	**data set**	**Min-Set for** x_2	**Min-Set for** x_3
$(x_1, x_1 \wedge x_2)$	1111	$\{x_1\}, \{x_2\}$	$\{x_1\}, \{x_2\}, \{x_3\}$
$(x_1, x_1 \wedge \overline{x_2})$	1121	$\{x_1\}$	$\{x_1\}, \{x_2\}, \{x_3\}$
$(x_1, x_1 \vee x_2)$	1211	$\{x_1\}, \{x_2\}$	$\{x_1\}, \{x_3\}$
$(x_1, x_1 \vee \overline{x_2})$	1112	$\{x_1\}, \{x_2\}$	$\{x_1\}, \{x_3\}$
$(x_1, \overline{x_1} \vee x_2)$	1212	$\{x_1\}, \{x_2\}$	$\{x_1\}, \{x_2\}, \{x_3\}$
$(x_1, \overline{x_1} \wedge \overline{x_2})$	1222	$\{x_1\}, \{x_2\}$	$\{x_1\}, \{\overline{x_3}\}$
$(x_1, \overline{x_1} \wedge x_2)$	1122	$\{x_1\}, \{x_2\}$	$\{x_1\}, \{\overline{x_3}\}$
$(x_1, \overline{x_1} \vee \overline{x_2})$	1221	$\{x_1\}, \{x_2\}$	$\{x_1\}, \{\overline{x_3}\}$
$(\overline{x_1}, x_1 \vee x_2)$	2211	$\{x_1\}, \{\overline{x_2}\}$	$\{x_1\}, \{x_3\}$
$(\overline{x_1}, x_1 \wedge \overline{x_2})$	2111	$\{x_1\}, \{\overline{x_2}\}$	$\{x_1\}, \{\overline{x_2}\}, \{x_3\}$
$(\overline{x_1}, x_1 \vee x_2)$	2112	$\{x_1\}, \{\overline{x_2}\}$	$\{x_1\}, \{x_3\}$
$(\overline{x_1}, x_1 \wedge \overline{x_2})$	2122	$\{x_1\}, \{\overline{x_2}\}$	$\{x_1\}, \{\overline{x_3}\}$
$(\overline{x_1}, \overline{x_1} \vee x_2)$	2121	$\{x_1\}, \{\overline{x_2}\}$	$\{x_1\}, \{\overline{x_2}\}, \{\overline{x_3}\}$
$(\overline{x_1}, \overline{x_1} \wedge \overline{x_2})$	2212	$\{x_1\}, \{\overline{x_2}\}$	$\{x_1\}, \{\overline{x_2}\}, \{\overline{x_3}\}$
$(\overline{x_1}, \overline{x_1} \wedge x_2)$	2221	$\{x_1\}, \{\overline{x_2}\}$	$\{x_1\}, \{\overline{x_3}\}$
$(\overline{x_1}, \overline{x_1} \vee \overline{x_2})$	2222	$\{x_1\}, \{\overline{x_2}\}$	$\{x_1\}, \{\overline{x_3}\}$

Table 4.1: A summary of all 16 data sets.

4.5 Final thoughts

The best case scenario of this experiment would have been that the min-sets defined one unambiguous function for each network. While this was, not surprisingly, not the case, there can still be insight gleaned from these min-sets, as well as some considerations to make future attempts more accurate.

The first item to consider is how to capture the behavior of a gene that is "On" but constant. Our current method, based on the published min-set algorithms in [5] and [10], does not capture a variable x_i if it does not change as the output changes. However, this can be misleading because a gene remaining "On" can influence the system even if it remains constant. Figuring out how to incorporate this into a reverse-engineering algorithm is an interesting open-ended question.

The second item to consider is how the data is generated. It is important to note that in the above data, gene x_1 was artificially stimulated during the start of the experiment, and therefore the min-sets for gene x_1 might not be reliable indicators. For this reason they have been excluded from Table 4.1 and the other 14 data sets in the Appendix. They are included for the examples in the previous two sections for completeness. Artificial starting values and simulations can be a source of false positives in our current method. For example, an introduction of a gene product at concentration levels above its steady state could cause the false appearance of an inhibition as it

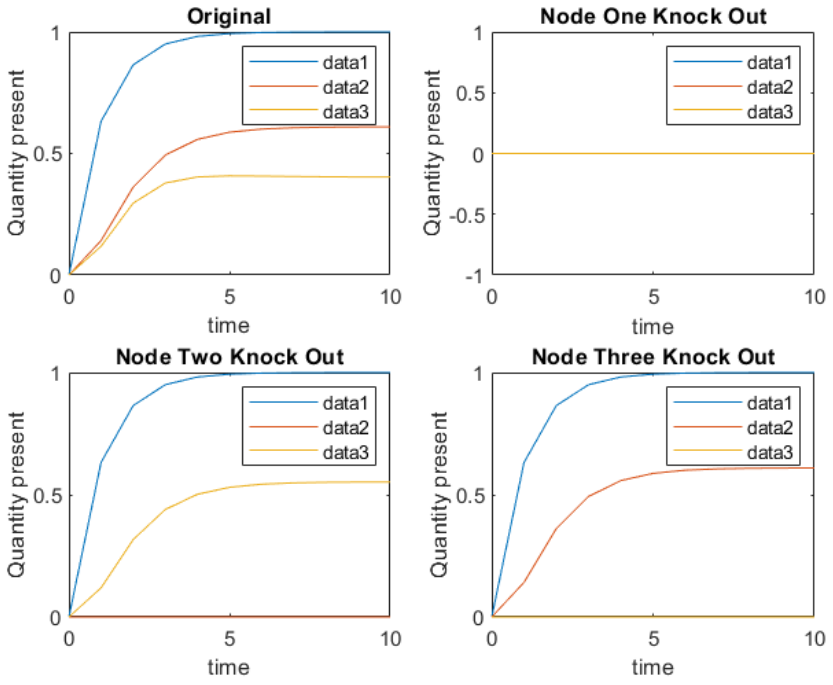
drops to its steady state and then remains constant. In other words, the natural decay term could have a false-positive effect.

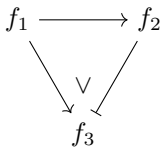
Finally, in a very different direction, while a number of results have been proven for pseudomonomials, since their appearance in the mathematical biology literature around 2012 [1, 10] a "signed version" of Stanley-Reisner theory has yet to be fully developed, and is quite fertile ground for future exploration. In other words, it seems clear that there are strong combinatorial reasons why pseudomonomial ideals, and primary decompositions, behave analogously to regular monomial ideals, but this is still largely unknown.

Algebraic methods in mathematical biology are still in their infancy. It is major challenge to connect the beautiful theoretic results to actual messy biological data. In this paper, we explored this and saw first-hand some of the challenges. Further research into the three aforementioned items is crucial to developing more robust and useful algebraic methods for analyzing biological data.

Chapter 5

Appendix





t	x_1	x_2	x_3
0	0	0	0
1	1	1	1
2	2	2	2

t	x_2	x_3
0	0	0
1	0	0
2	0	0

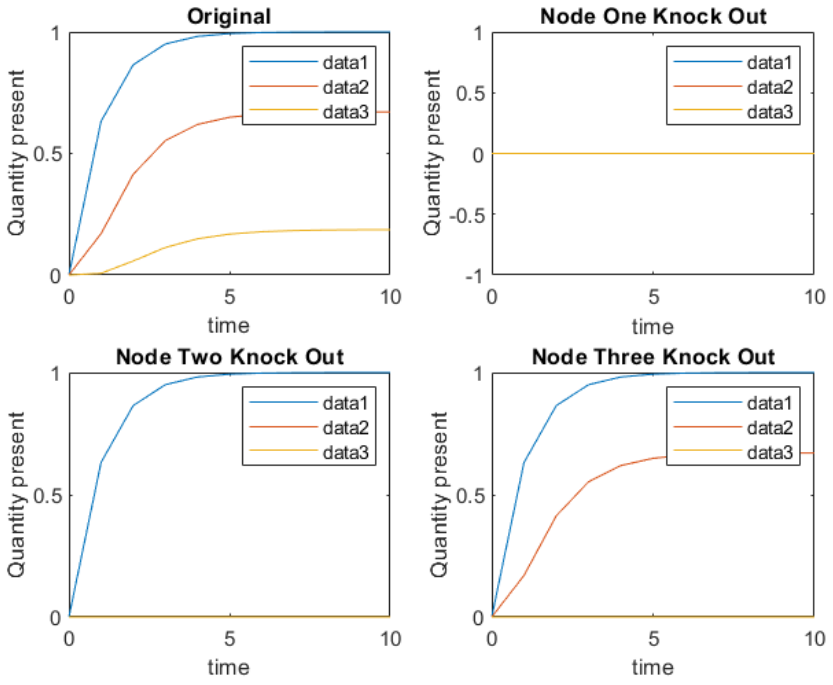
t	x_1	x_3
0	0	0
1	1	1
2	2	2

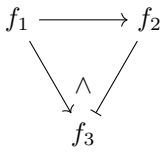
t	x_1	x_2
0	0	0
1	1	1
2	2	2

$J_{x_2} = \langle (x_1-1)(x_2-1)(x_3-1), (x_1-1)(x_2-1) \rangle, \qquad J_{x_3} = \langle (x_1-1)(x_2-1)(x_3-1), (x_1-1)(x_3-1) \rangle.$

- Gene x_2: $\{x_1 - 1\}$, $\{x_2 - 1\}$
- Gene x_3: $\{x_1 - 1\}$, $\{x_2 - 1\}$

Figure 5.1: Data set 1112





t	x_1	x_2	x_3
0	0	0	0
1	1	1	1
2	2	2	2

t	x_2	x_3
0	0	0
1	0	0
2	0	0

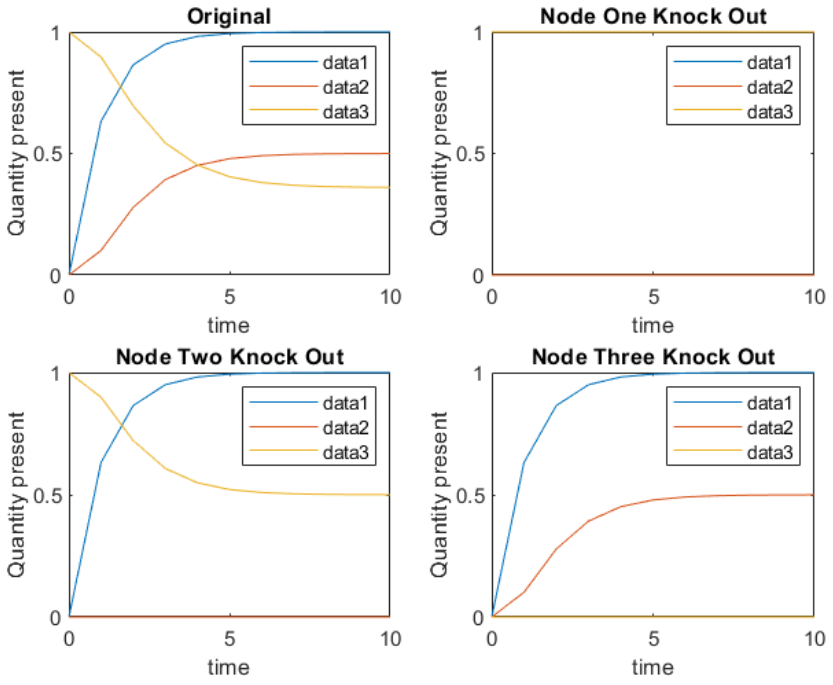
t	x_1	x_3
0	0	0
1	1	0
2	2	0

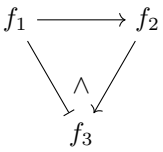
t	x_1	x_2
0	0	0
1	1	1
2	2	2

$J_{x_2} = \langle (x_1-1)(x_2-1)(x_3-1), (x_1-1), (x_1-1)(x_2-1) \rangle, \qquad J_{x_3} = \langle (x_1-1)(x_2-1)(x_3-1) \rangle.$

- Gene x_2: $\{x_1 - 1\}$
- Gene x_3: $\{x_1 - 1\}$, $\{x_2 - 1\}$,$\{x_3 - 1\}$

Figure 5.2: Data set 1121





t	x_1	x_2	x_3
0	0	0	2
1	1	1	1
2	2	2	0

t	x_2	x_3
0	0	2
1	0	2
2	0	2

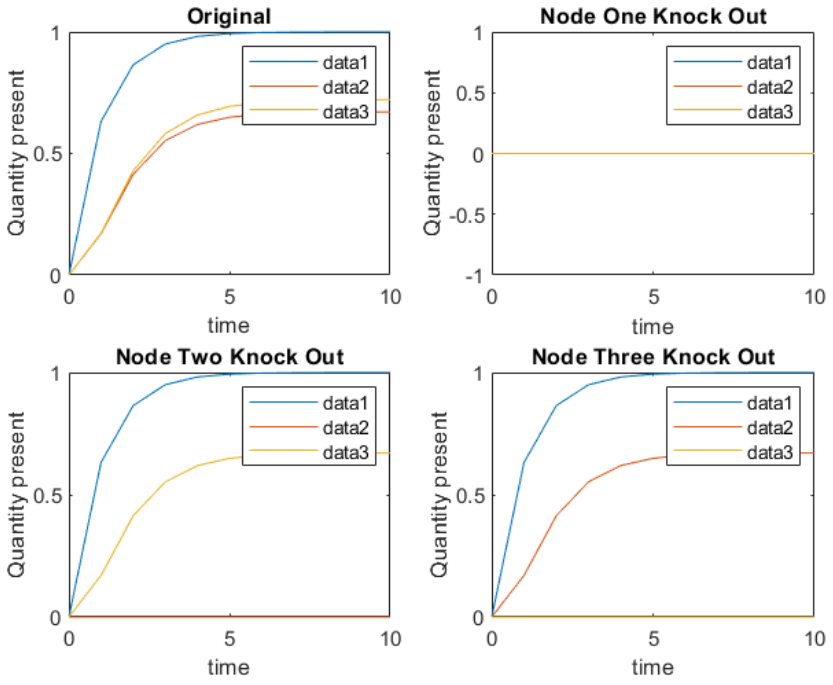
t	x_1	x_3
0	0	2
1	1	1
2	2	0

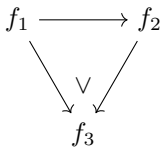
t	x_1	x_2
0	0	0
1	1	1
2	2	2

$$J_{x_2} = \langle (x_1-1)(x_2-1)(x_3+1), (x_1-1)(x_2-1) \rangle, \qquad J_{x_3} = \langle (x_1-1)(x_2-1)(x_3+1), (x_1-1)(x_3+1) \rangle.$$

- Gene x_2: $\{x_1 - 1\}$,$\{x_2 - 1\}$
- Gene x_3: $\{x_1 - 1\}$, $\{x_3 + 1\}$

Figure 5.3: Data set 1122





t	x_1	x_2	x_3
0	0	0	0
1	1	1	1
2	2	2	2

t	x_2	x_3
0	0	0
1	0	0
2	0	0

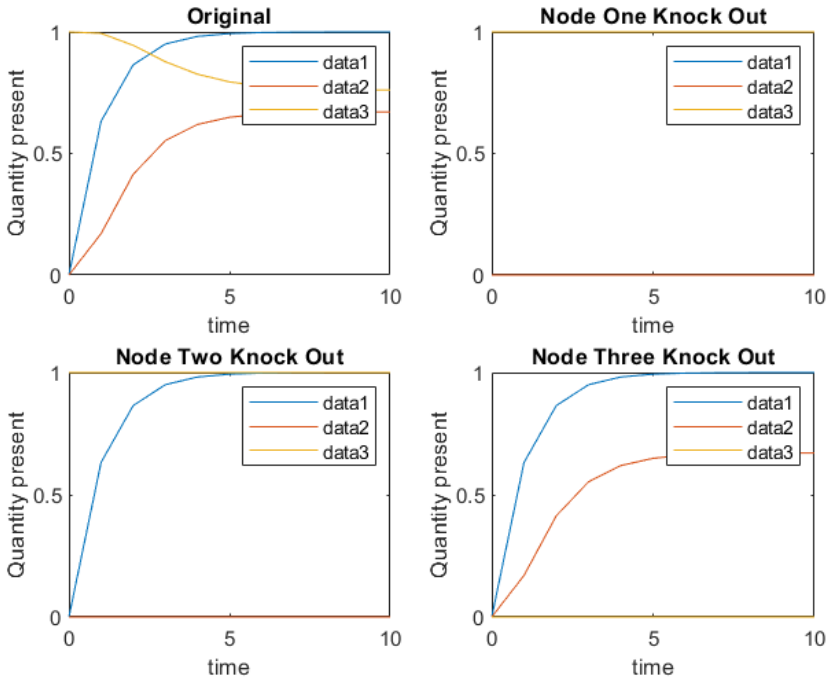
t	x_1	x_3
0	0	0
1	1	1
2	2	2

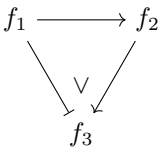
t	x_1	x_2
0	0	0
1	1	1
2	2	2

$J_{x_2} = \langle (x_1-1)(x_2-1)(x_3-1), (x_1-1)(x_2-1) \rangle, \qquad J_{x_3} = \langle (x_1-1)(x_2-1)(x_3-1), (x_1-1)(x_3-1) \rangle.$

- Gene x_2: $\{x_1 - 1\}$,$\{x_2 - 1\}$
- Gene x_3: $\{x_1 - 1\}$, $\{x_3 - 1\}$

Figure 5.4: Data set 1211





t	x_1	x_2	x_3
0	0	0	2
1	1	1	1
2	2	2	0

t	x_2	x_3
0	0	2
1	0	2
2	0	2

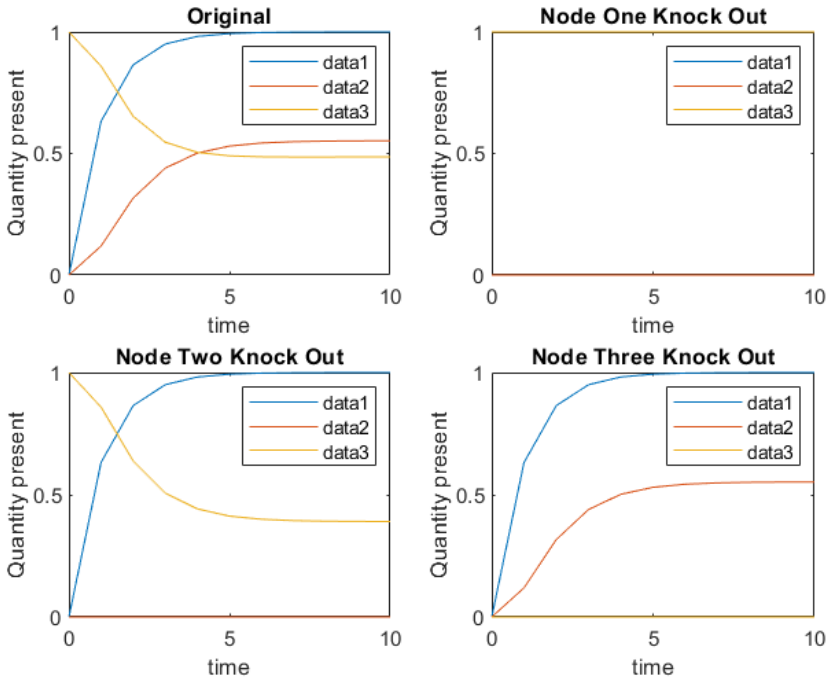
t	x_1	x_3
0	0	2
1	1	2
2	2	2

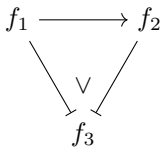
t	x_1	x_2
0	0	0
1	1	1
2	2	2

$J_{x_2} = \langle (x_1-1)(x_2-1)(x_3+1), (x_1-1)(x_2-1) \rangle, \qquad J_{x_3} = \langle (x_1-1)(x_2-1)(x_3+1) \rangle.$

- Gene x_2: $\{x_1-1\}$,$\{x_2-1\}$
- Gene x_3: $\{x_1-1\}$,$\{x_2-1\}$, $\{x_3-1\}$

Figure 5.5: Data set 1212





t	x_1	x_2	x_3
0	0	0	2
1	1	1	1
2	2	2	0

t	x_2	x_3
0	0	2
1	0	2
2	0	2

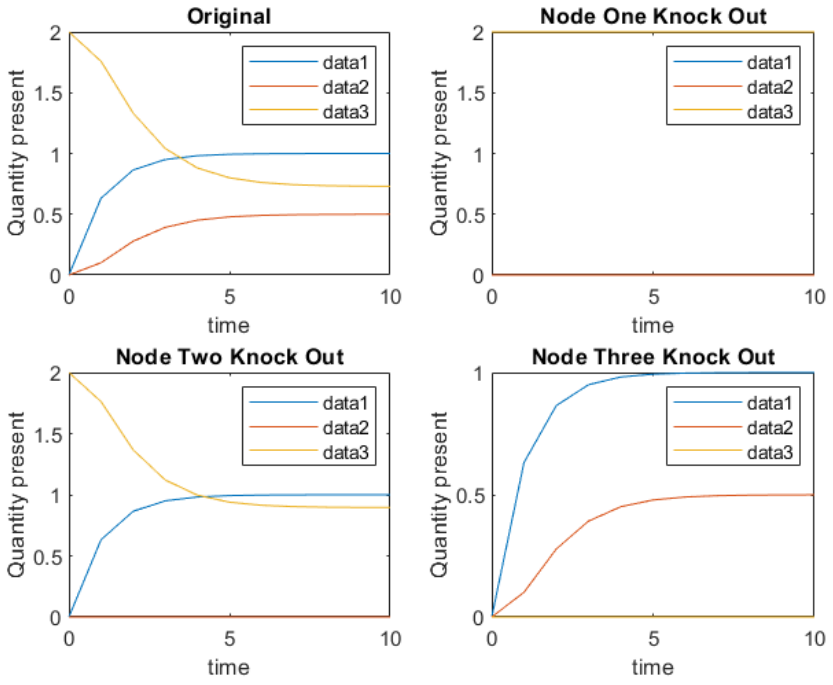
t	x_1	x_3
0	0	2
1	1	1
2	2	0

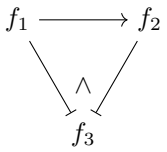
t	x_1	x_2
0	0	0
1	1	1
2	2	2

$J_{x_2} = \langle (x_1-1)(x_2-1)(x_3+1), (x_1-1)(x_2-1) \rangle, \qquad J_{x_3} = \langle (x_1-1)(x_2-1)(x_3+1), (x_1-1)(x_3+1) \rangle.$

- Gene x_2: $\{x_1 - 1\}$,$\{x_2 - 1\}$
- Gene x_3: $\{x_1 - 1\}$,$\{x_3 + 1\}$

Figure 5.6: Data set 1221





t	x_1	x_2	x_3
0	0	0	2
1	1	1	1
2	2	2	0

t	x_2	x_3
0	0	2
1	0	2
2	0	2

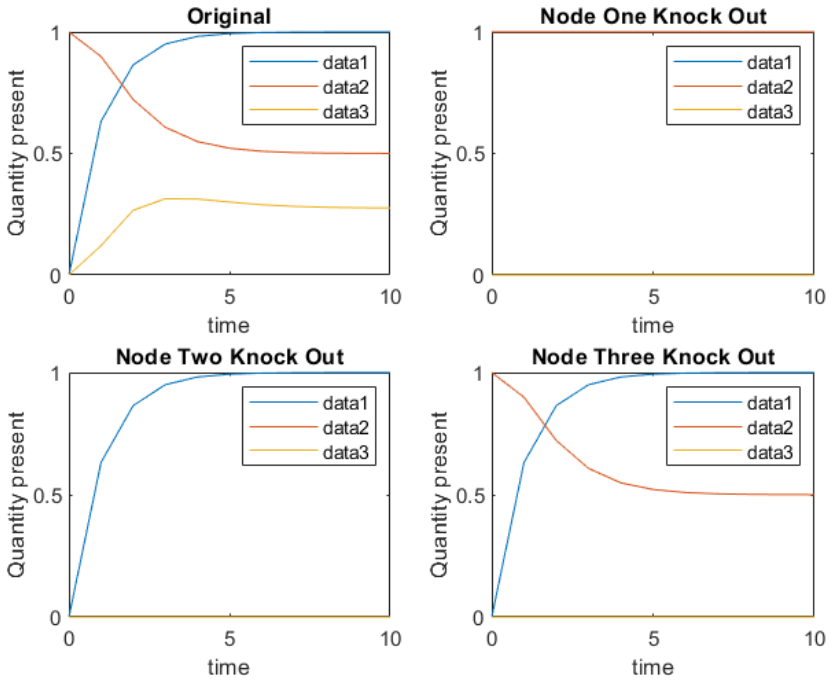
t	x_1	x_3
0	0	2
1	1	1
2	2	0

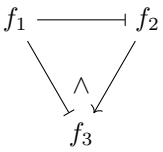
t	x_1	x_2
0	0	0
1	1	1
2	2	2

$J_{x_2} = \langle (x_1-1)(x_2-1)(x_3+1), (x_1-1)(x_2-1) \rangle, \qquad J_{x_3} = \langle (x_1-1)(x_2-1)(x_3+1), (x_1-1)(x_3+1) \rangle.$

- Gene x_2: $\{x_1 - 1\}$,$\{x_2 - 1\}$
- Gene x_3: $\{x_1 - 1\}$,$\{x_3 + 1\}$

Figure 5.7: Data set 1222





t	x_1	x_2	x_3
0	0	2	0
1	1	1	1
2	2	0	2

t	x_2	x_3
0	2	0
1	2	0
2	2	0

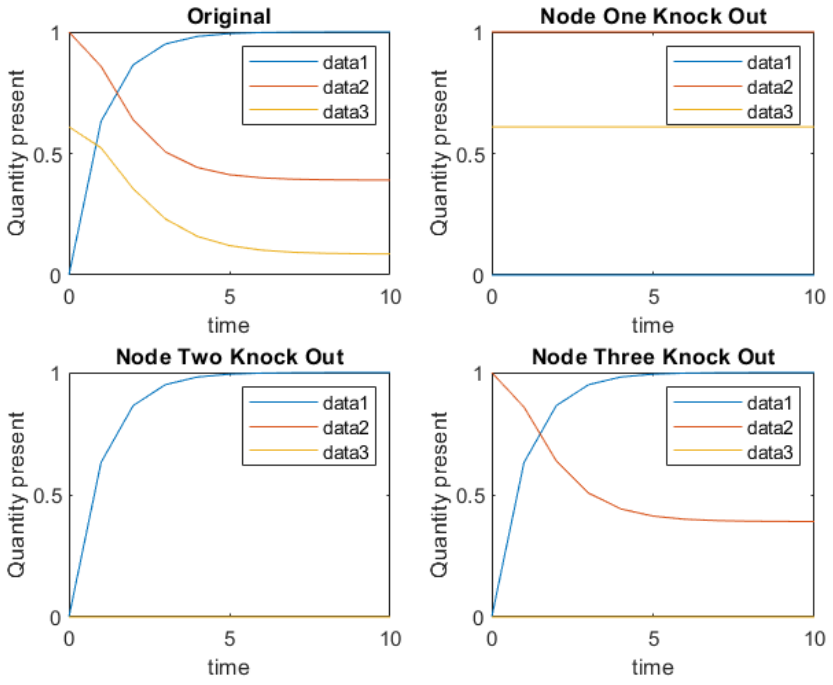
t	x_1	x_3
0	0	0
1	1	0
2	2	0

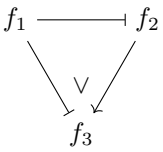
t	x_1	x_2
0	0	2
1	1	1
2	2	0

$J_{x_2} = \langle (x_1-1)(x_2+1)(x_3-1), (x_1-1)(x_2+1) \rangle, \qquad J_{x_3} = \langle (x_1-1)(x_2+1)(x_3-1) \rangle.$

- Gene x_2: $\{x_1-1\}$,$\{x_2+1\}$
- Gene x_3: $\{x_1-1\}$,$\{x_2+1\}$,$\{x_3-1\}$

Figure 5.8: Data set 2111





t	x_1	x_2	x_3
0	0	2	2
1	1	1	1
2	2	0	0

t	x_2	x_3
0	2	1
1	2	1
2	2	1

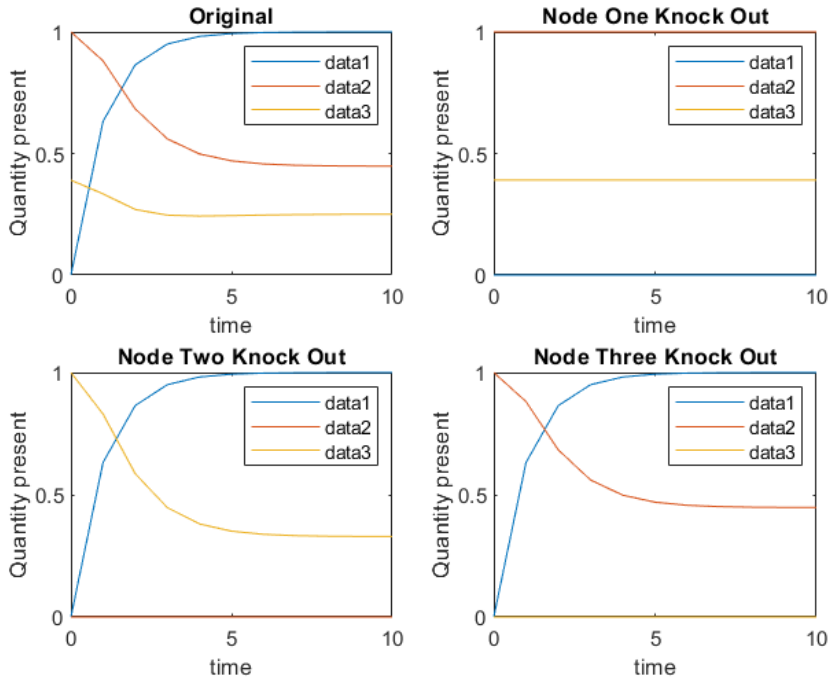
t	x_1	x_3
0	0	0
1	1	0
2	2	0

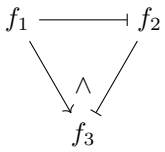
t	x_1	x_2
0	0	2
1	1	1
2	2	0

$J_{x_2} = \langle (x_1-1)(x_2+1)(x_3+1), (x_1-1)(x_2+1) \rangle, \qquad J_{x_3} = \langle (x_1-1)(x_2+1)(x_3+1) \rangle.$

- Gene x_2: $\{x_1-1\}$,$\{x_2+1\}$
- Gene x_3: $\{x_1-1\}$,$\{x_2+1\}$,$\{x_3+1\}$

Figure 5.9: Data set 2121





t	x_1	x_2	x_3
0	0	2	2
1	1	1	1
2	2	0	0

t	x_2	x_3
0	2	1
1	2	1
2	2	1

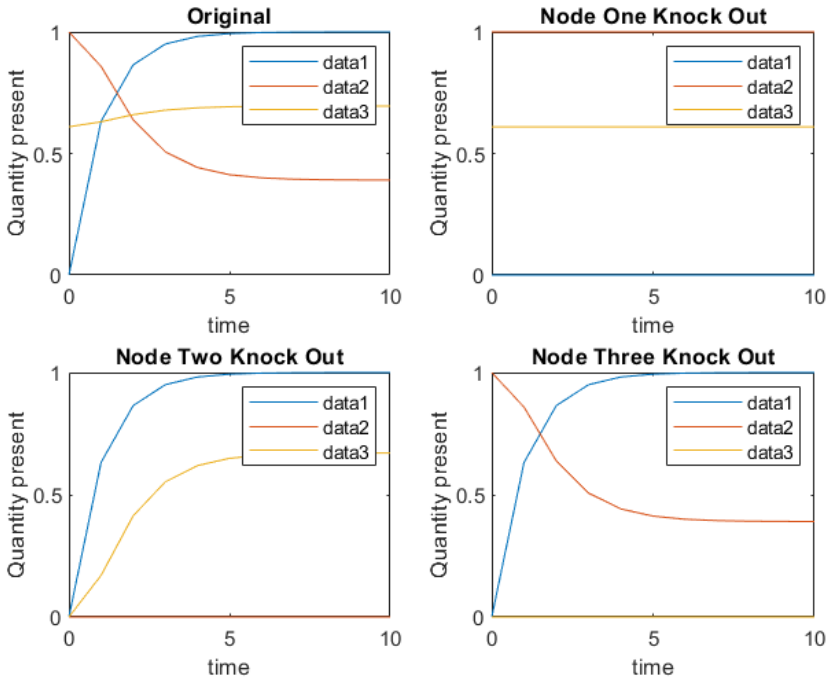
t	x_1	x_3
0	0	2
1	1	1
2	2	0

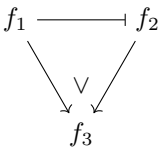
t	x_1	x_2
0	0	2
1	1	1
2	2	0

$J_{x_2} = \langle (x_1-1)(x_2+1)(x_3+1), (x_1-1)(x_2+1) \rangle, \qquad J_{x_3} = \langle (x_1-1)(x_2+1)(x_3+1), (x_1-1)(x_3+1) \rangle.$

- Gene x_2: $\{x_1 - 1\}$,$\{x_2 + 1\}$
- Gene x_3: $\{x_1 - 1\}$,$\{x_3 + 1\}$

Figure 5.10: Data set 2122





t	x_1	x_2	x_3
0	0	2	0
1	1	1	1
2	2	0	2

t	x_2	x_3
0	2	1
1	2	1
2	2	1

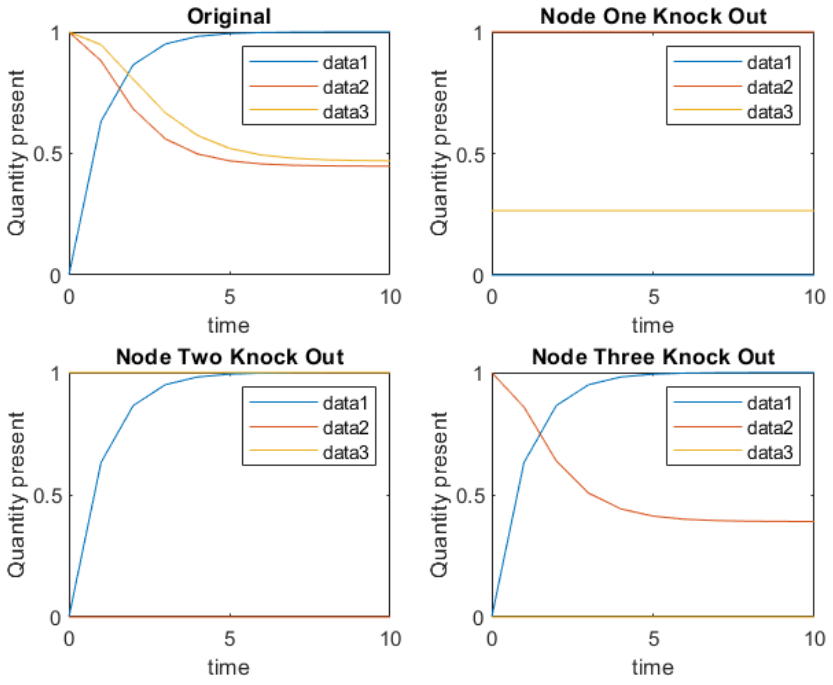
t	x_1	x_3
0	0	0
1	1	1
2	2	2

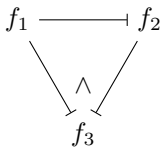
t	x_1	x_2
0	0	2
1	1	1
2	2	0

$J_{x_2} = \langle (x_1-1)(x_2+1)(x_3-1), (x_1-1)(x_2+1) \rangle, \qquad J_{x_3} = \langle (x_1-1)(x_2+1)(x_3-1), (x_1-1)(x_3-1) \rangle.$

- Gene x_2: $\{x_1 - 1\}$,$\{x_2 + 1\}$
- Gene x_3: $\{x_1 - 1\}$,$\{x_3 - 1\}$

Figure 5.11: Data set 2211





t	x_1	x_2	x_3
0	0	2	2
1	1	1	1
2	2	0	0

t	x_2	x_3
0	2	0
1	2	0
2	2	0

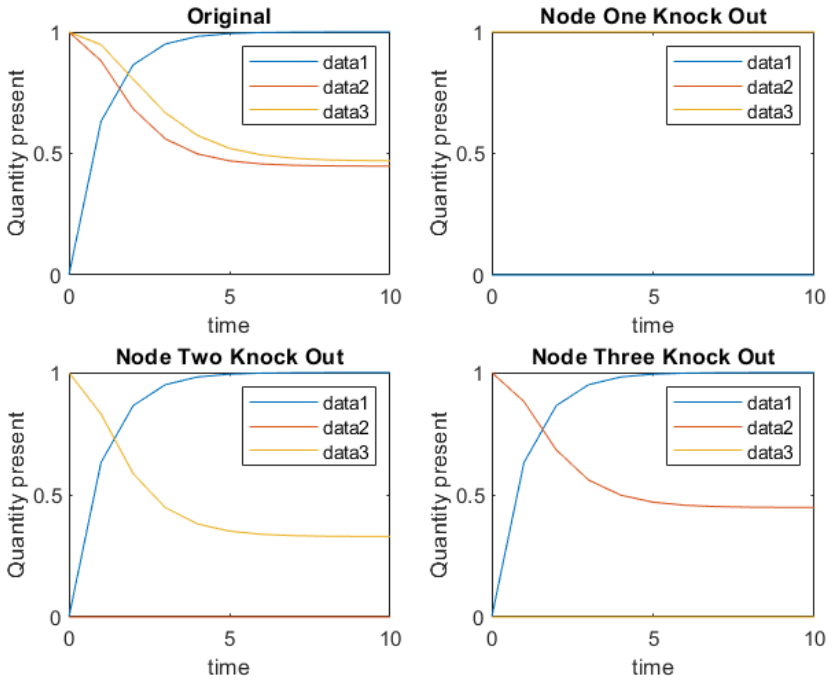
t	x_1	x_3
0	0	2
1	1	2
2	2	2

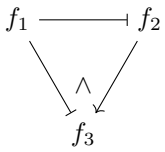
t	x_1	x_2
0	0	2
1	1	1
2	2	0

$J_{x_2} = \langle (x_1-1)(x_2+1)(x_3+1), (x_1-1)(x_2+1) \rangle, \qquad J_{x_3} = \langle (x_1-1)(x_2+1)(x_3+1) \rangle.$

- Gene x_2: $\{x_1-1\}$,$\{x_2+1\}$
- Gene x_3: $\{x_1-1\}$,$\{x_2+1\}$,$\{x_3+1\}$

Figure 5.12: Data set 2212





t	x_1	x_2	x_3
0	0	2	2
1	1	1	1
2	2	0	0

t	x_2	x_3
0	2	0
1	2	0
2	2	0

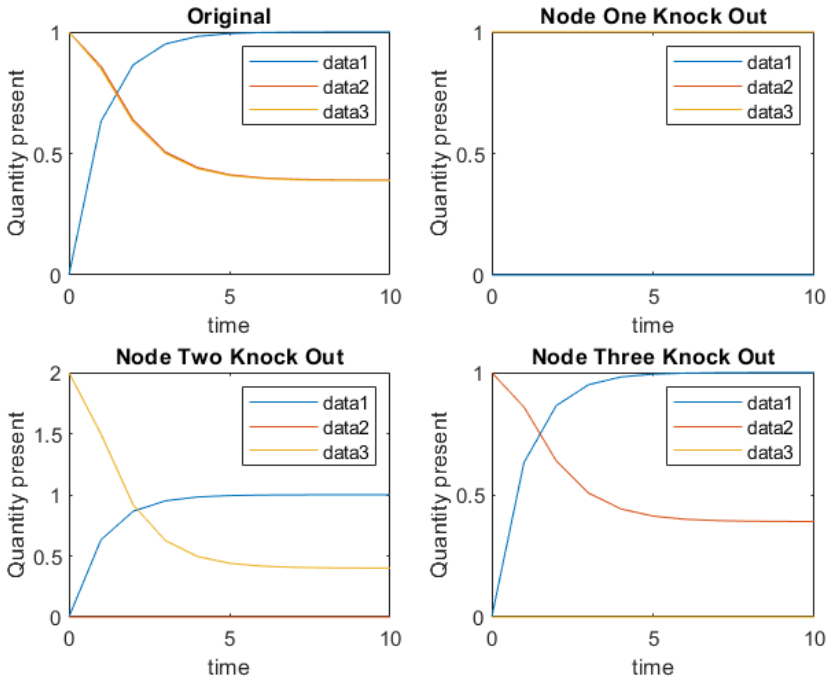
t	x_1	x_3
0	0	2
1	1	1
2	2	0

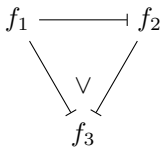
t	x_1	x_2
0	0	2
1	1	1
2	2	0

$J_{x_2} = \langle (x_1-1)(x_2+1)(x_3+1), (x_1-1)(x_2+1) \rangle, \qquad J_{x_3} = \langle (x_1-1)(x_2+1)(x_3+1), (x_1-1)(x_3+1) \rangle.$

- Gene x_2: $\{x_1 - 1\}$,$\{x_2 + 1\}$
- Gene x_3: $\{x_1 - 1\}$,$\{x_3 + 1\}$

Figure 5.13: Data set 2221





t	x_1	x_2	x_3
0	0	2	2
1	1	1	1
2	2	0	0

t	x_2	x_3
0	2	2
1	2	2
2	2	2

t	x_1	x_3
0	0	2
1	1	1
2	2	0

t	x_1	x_2
0	0	2
1	1	1
2	2	0

$J_{x_2} = \langle (x_1-1)(x_2+1)(x_3+1), (x_1-1)(x_2+1) \rangle, \qquad J_{x_3} = \langle (x_1-1)(x_2+1)(x_3+1), (x_1-1)(x_3+1) \rangle.$

- Gene x_2: $\{x_1 - 1\}$,$\{x_2 + 1\}$
- Gene x_3: $\{x_1 - 1\}$,$\{x_3 + 1\}$

Figure 5.14: Data set 2222

Bibliography

[1] Carina Curto, Vladimir Itskov, Alan Veliz-Cuba, and Nora Youngs. The neural ring: an algebraic tool for analyzing the intrinsic structure of neural codes. *Bulletin of Mathematical biology*, 75(9):1571–1611, 2013.

[2] Elena S Dimitrova, M Paola Vera Licona, John McGee, and Reinhard Laubenbacher. Discretization of time series data. *J. Comput. Biol*, 17(6):853–868, 2010.

[3] D.R. Grayson and M.E. Stillman. Macaulay2, a software system for research in algebraic geometry. Available at `http://www2.macaulay2.com/Macaulay2/`, 2020.

[4] SageMath Inc. *CoCalc Collaborative Computation Online*, 2020. `https://cocalc.com/`.

[5] Abdul Salam Jarrah, Reinhard Laubenbacher, Brandilyn Stigler, and Michael Stillman. Reverse-engineering of polynomial dynamical systems. *Advances in Applied Mathematics*, 39(4):477–489, 2007.

[6] Boris N Kholodenko, Anatoly Kiyatkin, Frank J Bruggeman, Eduardo Sontag, Hans V Westerhoff, and Jan B Hoek. Untangling the wires: a strategy to trace functional interactions in signaling and gene networks. *Proc. Natl. Acad. Sci.*, 99(20):12841–12846, 2002.

[7] Gregory R Smith, Mehdi Bouhaddou, Alan D Stern, Caitlin M Anglin, Orrod M Zadeh, Jake Erskin, and Marc Birtwistle. Network reconstruction from perturbation time course data. *BioRxiv*, page 341008, 2018.

[8] Eduardo Sontag, Anatoly Kiyatkin, and Boris N Kholodenko. Inferring dynamic architecture of cellular networks using time series of gene expression, protein and metabolite data. *Bioinformatics*, 20(12):1877–1886, 2004.

[9] Sandra Annie Tsiorintsoa. Pseudo-monomials in algebraic biology. Master's thesis, African Institute for Mathematical Sciences, South Africa, 2018.

[10] Alan Veliz-Cuba. An algebraic approach to reverse engineering finite dynamical systems arising from biology. *SIAM Journal on Applied Dynamical Systems*, 11(1):31–48, 2012.