An Algebraic Approach to Reverse Engineering with an Application to Biochemical Networks

Brandilyn S. Stigler

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

Doctor of Philosophy
in
Mathematics

Reinhard C. Laubenbacher, Chair
Christopher Beattie
Abdul Jarrah
Pedro J. Mendes

July 16, 2005
Blacksburg, Virginia

Keywords: Reverse engineering, gene regulatory networks, computational algebra, discrete modeling, polynomial dynamical systems

An Algebraic Approach to Reverse Engineering with an Application to Biochemical Networks

Brandilyn S. Stigler

(ABSTRACT)

One goal of systems biology is to predict and modify the behavior of biological networks by accurately monitoring and modeling their responses to certain types of perturbations. The construction of mathematical models based on observation of these responses, referred to as reverse engineering, is an important step in elucidating the structure and dynamics of such networks. Continuous models, described by systems of differential equations, have been used to reverse engineer biochemical networks. Of increasing interest is the use of discrete models, which may provide a conceptual description of the network.

In this dissertation we introduce a discrete modeling approach, rooted in computational algebra, to reverse-engineer networks from experimental time series data. The algebraic method uses algorithmic tools, including Gröbner-basis techniques, to build the set of all discrete models that fit time series data and to select minimal models from this set. The models used in this work are discrete-time finite dynamical systems, which, when defined over a finite field, are described by systems of polynomial functions. We present novel reverse-engineering algorithms for discrete models, where each algorithm is suitable for different amounts and types of data. We demonstrate the effectiveness of the algorithms on simulated networks and conclude with a description of an ongoing project to reverse-engineer a real gene regulatory network in yeast.

This research was supported by the National Institute of General Medical Sciences grant number RO1 GM068947-01.

Dedication

I dedicate this work

to my dad, who taught me to seize opportunities;

to my advisor, who taught me what it takes to love one's work;

to my husband, who teaches me to see the best in others and myself.

Acknowledgments

I first offer my humble gratitude to God: it is through His persistence, patience, and love that have carried me to where I am.

I give utmost thanks to my advisor, Reinhard Laubenbacher. You have been an incredible mentor and friend through these years. You saw potential in me when my vision was weak and provided me the encouragement and environment to be successful.

I thank the members of my committee, Christopher Beattie, Abdul Jarrah, and Pedro Mendes, for their helpful comments and discussions, as well as for their insightful questions about my work.

I thank the colleagues and friends, whom I have come to hold dear, at the Virginia Bioinformatics Institute (VBI). You made my transition to the worlds of mathematical biology, computational biology, and bioinformatics possible. My experience at VBI has instilled in me a strong sense of community and respect for all mathematicians and scientists. In particular I extend my thanks to current and former members of the Applied Discrete Mathematics Group and the Yeast Systems Biology Group, in particular, Diogo Camacho, Monica Castro-Simmons Autumn Clapp, Miguel Colón, Omar Colón-Reyes, Alberto de la Fuente, Edgar Delgado Eckert, Elena Dimitrova, Karen Duca, Dana Eckert, Luis García, Stefan Hoops, John McGee, Ana Martins, Karen Schlauch, Wei Sha, Jignesh Shah, Vladimir Shulaev, Hussein Vastani, Leepika Tuli, and Paola Vera Licona.

I thank the professors at Virginia Tech (VT) and at New Mexico State University (NMSU) who gave me the education necessary for pursuing my research. I extend a special thanks to those graduate students in the mathematics departments who enriched my academic and personal life, especially the members of the revitalized SIAM Student Chapter at Virginia Tech, including José María Menéndez Gómez and Mark Pierson. A special thanks goes to two women who were instrumental as graduate staff and fiercely competent as female role models: Rose Marquez (NMSU) and Hannah Swiger (VT).

Without the existence of student research programs, I would not have known how exciting research could be as a a young mathematician. I first thank Joaquin Loustanau for introducing such programs to me. I thank all the helping hands with SIMU, especially Ivelisse Rubio and Herbert Medina. This experience made my transition into graduate school smoother

and showed me the door to the wonderful community that is SACNAS. I also thank Ricardo Cortez for helping me take my work to a new country. I thank two important women in life, Rebecca García for introducing me to research the fun way, and Olgamary Rivera-Marrero, for helping me realize our own student research program.

Next I give my thanks to the family that raised me, for without my family I would not be here. To Jesusita and Mamalía, I thank you for providing me a stable home and a solid foundation. To Dad, Cheri, Sean, Shaina, and J. Allen, I thank you for teaching me how important Family is; for the love and support to be who I am now; for always demanding the best from me; and for the joy and peace I have in me to be happy and successful in life. I could not have asked for a more wonderful and spirited family. To Mom, Mandy, and Jacqui, I thank you for the love, laughter, and lessons you have given and continue to give me. To my uncle Terry, I thank you for challenging me: I will never forget how to prove that $\sqrt{2}$ is irrational.

Finally I give God thanks for the family that will see me through the next stage in life. To Dustin and Ely, I thank you for your love, your support, your unending patience, and your ability to see the best in me. I am truly blessed by your presence in my life and am comforted that we will be together for the upcoming journey.

Contents

List of Figures

List of Tables

Chapter 1

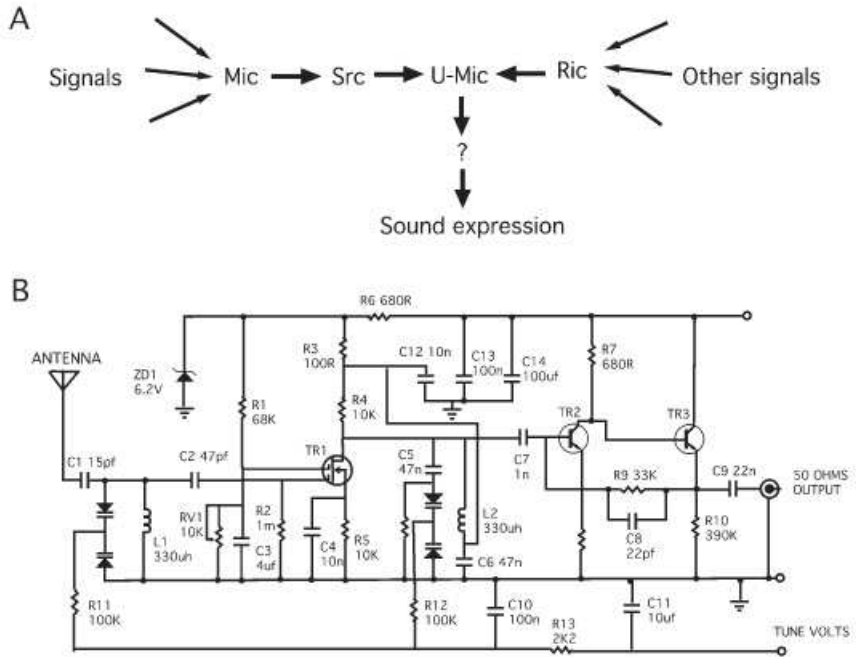
Introduction

Biology is the study of life processes of living organisms, from the metabolism of single-celled amoeba to migration patterns in flocks of birds. The prevalent modes of scientific discovery in biology in the last two centuries are the so-called reductionist and integrative approaches ([12]). In the reductionist mode, questions at all levels of living organization are reduced to the molecular level, whereas in the integrative mode, the focus is on the interaction between parts of an organism at multiple levels ([12], [35]).

In the 20th century, experimental as well as theoretical biologists used reductionist approaches to model and analyze biological systems ([11]; Ch. 1, [12]). For example, to understand breeding patterns in fish, one would study their genetic makeup, describe connections between the genes or proteins by way of diagrams, and use the diagrams to infer a mechanism for breeding. While this approach has been successful, Hiroaki Kitano ([35]) argued that simply knowing the molecular parts of an organism and how they are connected does not elucidate *how* the organism behaves, just as identifying the atoms in a dishwasher does not reveal how it works. The limitations of the reductionist paradigm are that the focus is on the identification of molecular "parts," in which the parts are viewed in isolation from the rest of the system, and that the cataloging of parts does not aid in discovering mechanisms for control ([35], [54]). Moreover, until recently, technological capabilities have limited the *number* of parts that could be studied.

Yuri Lazebnik ([39]) argued further that biologists need a language, similar to that in engineering or mathematics, to formalize the process of modeling and analysis. While mathematical models, including those described by systems of differential equations, have been extremely useful in the reductionist regime, the lack of rigorous mathematics training in life science curricula have encouraged biologists to use descriptive models over those based on a formal language ([33], [43]). In [39], Lazebnik illustrated the limitations of using descriptive models such as those based on graphics, by contrasting a graphical representation of a radio with a formal model based on engineering principles (see Figure 1.1).

Figure 1.1: Two representations of a radio, as presented in [39]. **A.** A graphical view of the connections between parts of a radio. **B.** A formal, quantitative view which includes size and capacity of the parts.



While biologists are still largely being trained in the reductionist paradigm ([11]), advances in the last two decades in high-throughput technology, such as DNA sequencing ([2], [45]), have provided experimentalists an ability to view large collections of the biomolecules of an organism. In fact, at the 2005 Joint Mathematics Meeting of the AMS and the MAA, John Whitmarsh, Health Scientist Administrator at the National Institute for General Medical Sciences of the National Institutes of Health, pointed out the transformation that the life sciences are currently experiencing. He stated that "biology is moving from being a descriptive science to being a quantitative [and ultimately, predictive] science" ([45], [57]). This paradigm shift is evidence that the purpose for institutes with such goals, including the Institute for Systems Biology, is being realized ([5]).

Technological advances in the life sciences have triggered an explosion of experimental data, representing the activity of a variety of biochemicals such as genes, proteins, and metabolites in living organisms. With this abundance of information has come the ability to gain knowledge about the underlying system, thereby catalyzing the return to an integrative paradigm. Systems biology focuses on the structure and dynamics of biological systems with an emphasis on prediction. The paradigm within this emerging science to develop an understanding of biological systems is to perturb the system at the molecular level, to accurately monitor the responses to the perturbations, and to formulate mathematical models that incorporate diverse data types (observations) and describe the structure and behavior of the system with

regard to the perturbations ([33]). Hence systems biology requires continued technological developments to address questions at the systems level ([35], [50]).

Having a comprehensive understanding of a biological system, as opposed to merely having a record of its parts, may be achievable by focusing on the following aspects of the system ([35]): its biochemical structure, function, and dynamics. In addition, methods for constructing, modifying, and controlling a biological system may be required for further understanding. Due to current technology, large amounts of data, such as whole genome sequences, gene ontologies, and multi-scale structures of proteins, are being collected from laboratories around the world and stored in ever-growing databases, such as GenBank ([3]), GEO ([25]), and Swiss-Prot ([46]). Present challenges are to integrate data from multiple sources and to model and analyze a system at disparate scales ([45]). Success at data integration and multi-scale modeling may lead to universal data standards, robust predictive modeling and validation in biomedical research, and scalable modeling methods ([45]).

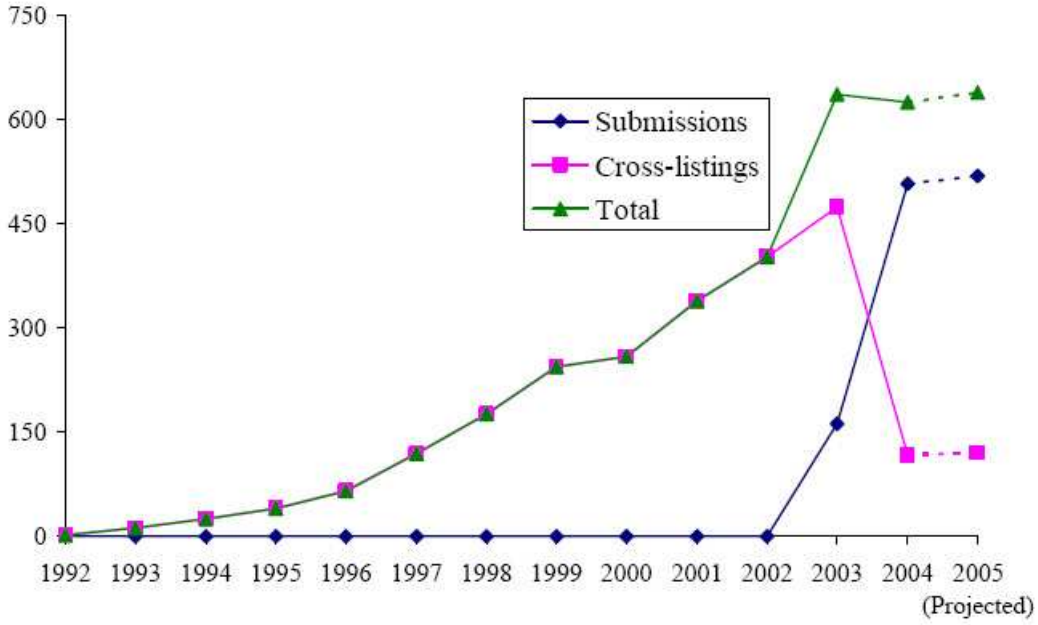
1.1 The Need for Mathematical Models

The persistent trend in science is to use models to understand systems, particularly for those that exist in nature. As defined by L. von Bertalanffy in [55] (p.56), a *system* is a set of interacting parts, which we also call *nodes*. In molecular biology, a system may be an assembly of biochemicals involved in chemical reactions carrying out the cellular life processes of an organism, such as metabolism. Models must be equipped not only to characterize the parts of the system, but also the interactions among the parts. Those written in a formal language, such as mathematics, can be used to simulate the behavior or dynamics of systems of interest. Such models have strength in their ability to aid the scientist in formulating hypotheses or making predictions about the system, which can be tested through experimentation. The importance of this process is the potential for uncovering salient features of the system.

The shift to integrative biology has brought mathematics to the forefront as an essential resource for modeling and analysis ([43]). In fact, this can be seen in the increase of quantitative biology articles posted on `arXiv.org` in the last decade (see Figure 1.2). There is a variety of dynamic modeling tools that are currently being exploited in systems biology, which can be categorized as follows ([33]). One class of models consists of systems of differential equations, which are applied to the modeling of biochemical reactions. In this setting variables represent continuous concentrations of biochemicals and for each variable, there is a differential equation that simulates the time evolution of the reaction in terms of continuous changes of concentrations. Continuous models can encode low-level mechanistic properties of systems. The limitation of continuous approaches is that typically the systems of differential equations are too complex to be solved analytically. Therefore, only approximate, numerical solutions can be found, where solutions represent concentrations of the biochemicals involved in the reactions. Examples of continuous models include ordinary,

partial, and delay differential equations (see [33] and [43]).

Figure 1.2: The number of *q-bio* articles posted annually on `arXiv.org`.



Measurements from biological experiments contain noise/error, due to biological variability in the organism or technological constraints of the instruments used for data collection. One approach to resolve this problem is to quantize or discretize the data into a finite set of labels representing groupings of continuous values ([21]). It is also the case that experimental data are discretized when using continuous methods in order to perform model validation (for example, see [9]).

In response to these issues, a different class of modeling methods has been used, which consists of discrete circuits characterized by systems of discrete-valued functions. Discrete models represent biochemical systems through graphs associated to the functions. Vertices of the graph correspond to biochemicals in the system, which taken on discrete quantities or levels, and edges depict interactions between the biochemicals affecting their levels. Discrete models provide a high-level qualitative view of systems. The limitation of discrete approaches lies in the simplicity of the models, as compared to their continuous counterparts. Another issue is that in practice, the number of discrete quantities or levels that is often used is two, to signify "presence"/"absence" or "activity"/"inactivity" of the biochemicals, further limiting the scope of the models. Examples of discrete models are Bayesian belief networks, neural networks, Boolean models, including probabilistic and random Boolean models, multi-logic models, and Glass networks (see [19], [24]).

There are other categories of dynamic mathematical models, including deterministic/stochastic and linear/nonlinear (for a review, see [19]). Regardless of the framework, mathematical

models are an essential component in studying biological systems, as they lend themselves naturally to explicitly describing the connectivity structure of the nodes in the system, as well as to methodically analyze the behavior of the nodes ([19]).

1.2 From Systems to Models to Systems

Given a modeling framework, there are a variety of ways to construct a model and can be broadly characterized as follows. The first type of method constructs a mathematical model from an existing, yet incomplete "model." The existing model is oftentimes manifested as a conceptual description of the system, arising from extensive knowledge and years of research ([20]). With these premises, a skeleton of a model is selected from a family of models, such as the collection of SIR models which are used in modeling epidemics including influenza and AIDS. Then a specific model is built by constraining the system so that the model behaves in a predetermined way. The mathematical model can then be used to simulate the behavior of the system. Such a method is referred to as forward engineering or bottom-up modeling as one starts from first principles (the conceptual model) and designs a system (or a simulation of the system) ([33]). The challenge in this modeling approach is to identify parameters by which to impose constraints on the system.

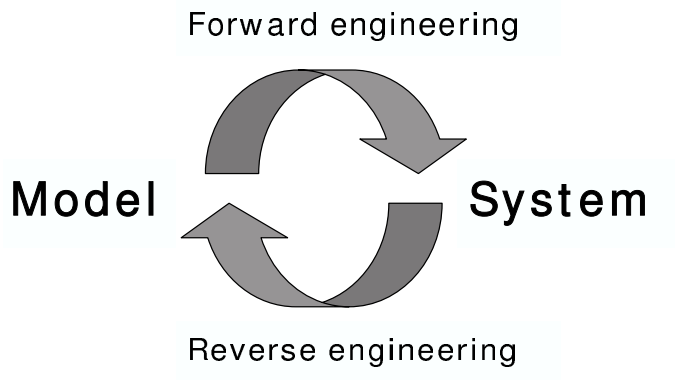
The second type of method builds a mathematical model from observations of the system in response to well-constructed perturbations. In molecular biology the observations may be measurements of concentrations of biomolecules, recorded at pre-determined time intervals, resulting in time series of experimental data (for example, see [58]). A model is built from the observations and is adjusted to fit the observations. As in the previous case, the mathematical model is a representation of the system which can be used to identify key features, including system dynamics. This process of discovery is called reverse engineering or top-down modeling: the starting point is the system (the observations) and the result is a model ([20], [33]). The challenge in this paradigm is the development of algorithmic tools for constructing models from data, which at present are enumeration techniques (see Chapter 2). The relationship between systems and their models can be seen in Figure 1.3.

Observations of biological systems at the molecular level are ever abundant ([20], [33]). A key goal within systems biology is to gain insight into the structure and dynamics of systems, there is a growing need for reverse engineering methods, which by their nature translate observations into predictive models.

1.3 Reverse Engineering

Informally, *reverse engineering* is the process of discovering the behavior and connectivity of the parts of a system, given observations of the system over time ([20], [27], [52], [58]). Local

Figure 1.3: A graphical view of the relationship between systems and their models.



behavior of a node in the system, when described in relation to other nodes, may give rise to relationships or interactions between them. Therefore, reconstructing local interactions may lead to identification of global, system-level behavior ([55], p.55).

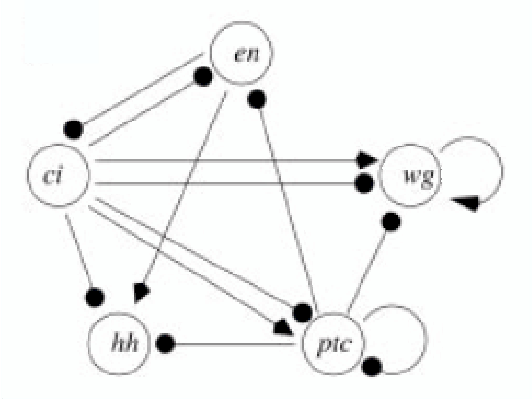
Connectivity information can be depicted in a *wiring diagram*. These diagrams are typically represented by graphs, such as Figure 1.4 from [52], where vertices are the nodes of a system and edges indicate the direction and perhaps the type of interaction between the nodes. As observations in experimental biology are recorded at pre-determined times, the dynamics of a biological system is manifested as a *time series* of values representing the level or amount of the nodes in the system.

The structure and dynamics of a model can also be discussed and compared to the system under investigation. As with real systems, a wiring diagram can be associated to a model, where the diagram is created from the functions defining the model in such a way that analogies between the system and the model can be made. The dynamics of a model can also be viewed graphically, where the representation depends on the type of model. If the model is continuous, the dynamics is typically depicted as the graph of the functions defining the model, that is, the values of the functions are plotted against time. For discrete models, this information is stored in a vertex graph, in which vertices are the possible states of the system and edges represent transition between the states where the transitions are given by iteration of the functions defining the model.

Here we focus on the reverse engineering of biochemical systems using discrete models, where the models are discrete-time discrete-state dynamical systems. In this setting the general reverse-engineering problem can be posed as follows.

Problem 1 (Reverse Engineering). *Let $T = (\mathbf{s}_1, \ldots, \mathbf{s}_t)$ be a time series of observations in a set X for a system with n nodes. Find a function $f : X^n \to X^n$ such that*

Figure 1.4: A wiring diagram of a system of 5 genes in the fruit fly *D. melanogaster*, as presented in Tegnér *et al.* ([52]). Edges with arrows denote activation, whereas edges with circles denote inhibition.



1. (Connectivity) *the wiring diagrams of the system and f are consistent, and*
2. (Dynamics) $f(\mathbf{s}_i) = \mathbf{s}_{i+1}$ *for each* $i < t$.

In this discourse we present a method, from the point of view of computational algebra, to construct *polynomial dynamical systems* (PDSs) from time series of discrete data. Next is a brief review of existing mathematical methods for reverse engineering. In order to provide an inclusive discussion for the algebraic method, concepts from a variety of fields will be introduced, which is the objective of chapters 3, 4, 5. In these chapters, we introduce some basic concepts from molecular biology, computational commutative algebra and algebraic geometry, and finite dynamical systems theory. Chapter 6 is devoted to the novel method of reverse engineering. The approach is presented as an algorithm, implemented in the symbolic computation softwares *Macaulay 2* and CoCoA, in which a certain class of PDSs for biochemical networks described by time series of experimental data is constructed. Chapter 7 contains some applications of the method to simulated biochemical networks and results are reported. The final chapter closes with a discussion of the implications of the algebraic method and a plan for future work.

Chapter 2

Mathematical Advances in Reverse Engineering

In recent years mathematics has come to the forefront as a viable resource for algorithmic techniques to reverse-engineer systems of interactions among entities in biological systems. Particular emphasis has been placed on gene regulatory networks, comprised of genes, proteins, and metabolites that regulate each other's production. Mathematical techniques for reverse engineering range from the continuous to the discrete. Continuous methods, such as the one developed in [58], construct all possible linear systems of differential equations that agree with the data and use some criterion to select a representative model. Many discrete methods, on the other hand, construct qualitative models, such as those described by Boolean functions in [7], which take into account the amount of information stored in the data. Next we present various modeling frameworks and provide a review of existing methods for each framework.

2.1 Bayesian Networks

A Bayesian network (G, θ) is a representation of a joint probability distribution that is comprised of a graph G and a probability distribution θ. G is a directed acyclic graph where vertices correspond to random network variables, whether continuous or discrete, and directed edges correspond to dependencies between variables. θ describes a conditional distribution for each variable of the network, given its parents as defined by the relations in G. Together they capture the conditional independence relations between the variables.

Friedman *et al.* ([26]) proposed Bayesian networks to infer causal dependencies between genes in gene regulatory networks. The goal is to estimate the posterior probability of chosen features being inherent in the network, given the data. Estimation of these probabilities gives rise to a class of Bayesian networks which are consistent with the data. To minimize the

model space, the authors employed bootstrapping methods to generate perturbations of the original data set, which help to determine the effect of noise on the learning process. The authors then used learning algorithms to search the model space and select a Bayesian network based on fixed selection criteria.

They applied this learning approach to 76 mRNA expression measurements of 6177 genes in yeast. They found that the Bayesian network learned from the data is stable to perturbations in the data. However, their analysis is sensitive to the choice of Bayesian network from the selection process. The authors do not provide a complexity analysis of the proposed algorithm, though they do report that the problem of finding a network with maximal score is NP-hard.

An extension of this work was proposed in [47] by Pe'er *et al.* to reverse-engineer significant subnetworks of interacting genes and was applied to yeast gene expression data. In [29], the concept of a *dynamic Bayesian network* (DBN) was introduced by Hartemink *et al.* to deal with time-dependent data. Because causal dependencies are viewed as unidirectional in the Bayesian framework, Bayesian networks are not equipped to model feedback loops, for example, which can occur over time. Therefore, the authors proposed the use of dynamic Bayesian networks for reverse-engineering gene regulatory networks from time series data. They used DBNs to model the galactose metabolic pathway in yeast from gene expression data with promising results.

2.2 Boolean Networks

A Boolean network $G(V, F)$ is a set $V = \{v_1, \ldots, v_n\}$ of vertices representing nodes of the network, together with a collection $F = (f_1, \ldots, f_n)$ of Boolean functions assigned to each node. Boolean networks were first used in the life sciences in the 1960s, when Stuart Kauffman introduced them to model regulatory networks as logical switching networks. Since then Boolean networks have received limited attention in the life sciences, although there are numerous theoretical results proved about them.

Liang *et al.* proposed an information-theoretic algorithm for constructing Boolean networks from Boolean time series data ([40]). The algorithm constructs both the global function as well as the wiring diagram that carry the maximal amount of information, as defined by the Shannon entropy and mutual information measures. The Shannon entropy is the measure of the information stored in a given set of data and is defined in terms of the probability of observing a particular state in the data. Its closed form is the function

$$H(x_1, \ldots, x_m) = -\sum_{i=1}^{2^m} p_i \log_2(p_i)$$

where p_i is the probability of observing state i in the data set restricted to the variables $x_1, \ldots, x_m$ and state i is one of the possible 2^m 0-1 combinations for m variables. Given a set

of state transitions, the algorithm REVEAL computes the entropy and mutual information for all input-output combinations of the form $H(\textit{input set}, \textit{output variable})$, where *input set* consists of any set of variables and the mutual information is defined by

$$M(I, o) = H(I) + H(o) - H(I, o).$$

for $I = \textit{input set}$ and $o = \textit{output variable}$. For computational feasibility, they restrict the size of I to be at most 3. For each variable, the I-o pair with the greatest mutual information gives rise to the most likely Boolean function, on at most 3 variables, associated to the variable which is consistent with the given data set. The Boolean network is the collection of Boolean functions selected by the algorithm together with the corresponding wiring diagram.

The worst-case complexity of REVEAL is the computation of the probability of each of the possible 2^n state transitions for every wiring diagram out of the total $\binom{n}{k}$ possibilities, where n is the number of nodes and k is the maximum number of inputs allowed per node. However, empirically the authors found that the algorithm performs better than in the worst case. For a 50-node network and inclusion of up to 100 state transitions for $k = 1, 2, 3$, the number e of misidentified solutions obtained by REVEAL decreased exponentially, with $e < 10^{-8}$ for $k = 3$ and maximal state transitions. Based on our literature search, we found that REVEAL has only been applied to synthetic data sets.

In [6], Akutsu *et al.* proposed the use of gene disruptions (knock-outs) and gene over-expressions to identify gene networks with Boolean networks. As with [40], the authors provide theoretical results and have not applied their method to real systems. Other algorithms, together with complexity analyses, are described in [8].

A reconstruction technique that allows for uncertainty in the data is given in [30], in which Hashimoto *et al.* introduced *probabilistic Boolean networks* (PBNs), a Bayesian adaptation of standard Boolean networks. The authors proposed an algorithm for constructing PBNs to represent a subnetwork of a larger gene regulatory network. They applied their method to gene expression data from human melanoma and glioma and were successful at making inferences about pathways in the networks.

2.3 Ordinary Differential Equations

Systems of ordinary differential equations (ODEs) have played a tremendous role in modeling biological networks. For a network of n nodes, a system of ODEs is a collection of simultaneous equalities

$$\begin{aligned} \frac{dx_1(t)}{dt} &= f_1(x_1(t), \ldots, x_n(t)) \\ &\vdots \\ \frac{dx_n(t)}{dt} &= f_n(x_1(t), \ldots, x_n(t)) \end{aligned}$$

where each $x_i(t)$ is a function of time representing node i.

Yeung *et al.* described a method to reverse-engineer gene networks with linear ODEs ([58]). The models are of the form

$$\frac{dX}{dt} = AX + B \tag{2.1}$$

where X is an $(n \times m)$-matrix of m mRNA measurements for n genes; A is an $(n \times n)$-matrix of strengths of interactions between the genes, including self-degradation rates; and B is an $(n \times m)$-matrix of external stimuli. In their paper, they describe an algorithm to reverse-engineer the wiring diagram of a gene network given time series of mRNA concentrations and values for the external stimuli for systems operating near a steady state. The wiring diagram is obtained through identification of the entries of A. Since $\frac{dX}{dt}$, X, and B are given, the goal is to solve the equation for A.

If X is invertible (*i.e.*, $m = n$ and X has full rank), then a solution can be found. In general, however, $m \ll n$ resulting in a vastly underdetermined system. So, the authors apply singular value decomposition to X so that

$$X = VWU^T,$$

where U and V are orthogonal and W is a diagonal matrix, and Equation 2.1 can be solved for A. A solution to the equation is then

$$A = A_0 = \left(\frac{dX}{dt} - B\right) UW^{-1}V^T$$

with nullspace CV^T, for some matrix C.

In fact, $A = A_0 + CV^T$ represents all feasible solutions that are consistent with the given data, for different choices of C. The last step of their method is to choose one solution from this set. Since the authors are interested in applying their method to gene networks, which they assume to be sparse, the selection criterion employed is sparseness of the matrix A; that is, the entries of C should be chosen so that the number of 0 entries in A is maximized. They accomplish this through L_1 regression on the equation

$$CV^T = -A_0 = 0$$

where 0-entries are maximized.

They applied this reverse-engineering algorithm to *in numero* experiments for gene networks of size 10 to 1000 and up to 100 measurements. The data sets included linear and nonlinear interactions between genes. The authors found that the algorithm requires $m = O(\log n)$ experimental measurements and $O(n^4)$ time to correctly identify the wiring diagram.

In the following year the same authors, together with a colleague, presented in [52] an alternate reverse-engineering algorithm, again using linear ODEs. Tegnér *et al.* described a

procedure to generate experimental data of perturbations appropriate for their new algorithm based on selection of genes whose expression changed least, as well as those whose connections are most uncertain. With the generated data, they used linear algebra techniques to estimate the entries of A, as above. The authors applied this algorithm to *in numero* experiments, as well as to data generated from a nonlinear model of a *Drosophila* segment polarity network for which they were able to reconstruct most of the wiring diagram.

Another example of the use of linear ODEs is presented in [17]. Chen *et al.* proposed four models for gene and protein expression data: one for both data types, one for gene expression only, another for protein expression only, and a fourth for gene expression data with time delays. They describe two algorithms for constructing such models from data. One of their algorithms is reported to be NP-complete; however, if the number k of inputs per node is fixed, then the algorithm requires $O(n^{k+1})$ time. The methods were not applied to real systems and only theoretical results were provided.

While ODE models represent temporal behavior of entities in a network, they can be extended to include spatial information of the entities as well. In this setup, such models are systems of partial differential equations. The additional sophistication comes at the cost of complexity in solving. Because of their computational expense, they have received little attention in the life sciences and we do not discuss them in this exposition.

2.4 Qualitative Networks

Qualitative networks are depicted as digraphs $G = (V, E)$, in which the edges (activation/inhibition) are governed by simple ordinary differential equations. Let the elements of $V = \{v_1, \ldots, v_n\}$ represent network nodes and $X_i(t)$ (or X_i, for short) be the expression value of v_i at time t. An activation edge $v_j \to v_i$ is in E iff

$$\frac{dX_i}{dt} > 0 \text{ if } X_j > 0 \text{ or } \frac{dX_i}{dt} < 0 \text{ if } X_j < 0.$$

Similarly an inhibition edge $v_j \multimap v_i$ is in E iff

$$\frac{dX_i}{dt} < 0 \text{ if } X_j > 0 \text{ or } \frac{dX_i}{dt} > 0 \text{ if } X_j < 0.$$

Akutsu *et al.* assert that qualitative networks can be used to reverse-engineer gene networks ([8]). Given a set of Boolean time series data, the proposed algorithm QNET-1 constructs a complete directed graph and removes edges that are not consistent with the given data. QNET-1 is reported to uniquely identify the Boolean network with probability at least $1-\frac{1}{n^\alpha}$, for a fixed constant $\alpha > 1$, in $O(n^2 m)$ time. For complete identification of the network, the algorithm requires $O(\alpha \log n)$ time series, beginning with initial values chosen uniformly randomly from $\{\pm 1\}$. An extension QNET-2 has been implemented for monotonic functions

Table 2.1: A summary of reverse-engineering methods. Each row of the table lists a sampling of methods that exist for each type of model, as well as indicate some properties of the models and the methods. c/d refers to the type of model with c = continuous and d = discrete. F/G refers to what is constructed by the method where F = dynamic model and G = wiring diagram. A = existence of an algorithm, RD = application of method to real data.

Modeling framework	c/d	References	F/G	A	RD
Bayesian network	c,d	[26], [47]	G	yes	yes
– Dynamic BN	c,d	[29]	G	yes	yes
Boolean network	d	[7], [8], [6], [40]	F, G	yes	no
– Probabilistic BN	d	[51]	F, G	no	no
ODE	c	[52]	F, G	yes	yes
	c	[17], [58]	F, G	yes	no
Qualitative network	d	[8]	G	yes	no

(right-hand sides of the differential equations). Based on our search, we found that both algorithms have been tested on synthetic data only.

While this exposition is by no means complete, its purpose was to simply provide the reader with some background in a variety of reverse-engineering methods. Table 2.1 contains a summary of the methods discussed above. For a review of other existing reverse-engineering methods, see [20], [19].

Chapter 3

Biology

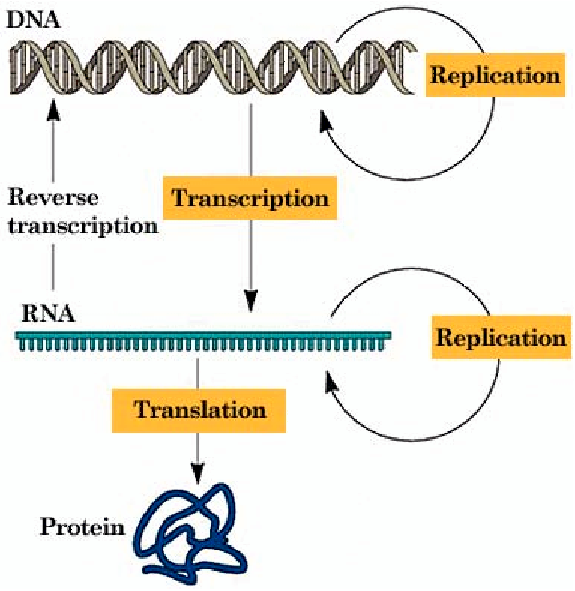
All definitions provided in this chapter can be found in [15], unless otherwise stated.

3.1 Concepts from Molecular Biology

Biology is the study of all life processes, both chemical and physical. *Metabolism*, which refers to the chemical processes that maintain the functioning of an organism, is responsible for the extraction of energy from nutrients and the biosynthesis of chemicals required for survival, including DNA and proteins. Metabolic processes are organized into *pathways* and can be characterized as *catabolism*, the destructive phase of metabolism, and *anabolism*, the constructive phase. Through these two phases, all other chemical reactions have the energy available in order to function, including the processing of genetic information.

Deoxyribonucleic acid (DNA) is a cellular molecule whose double helical structure encodes all of the genetic information of an organism. *Genes* are sequences of chemicals, called *nucleic acids*, in DNA which roughly correspond to genetic functions within the DNA. The molecules that carry out those functions are called *proteins*. The first step in processing genetic information is *transcription*, in which the information is "read" or copied from the gene and recorded in a molecule, called *messenger RNA* (mRNA). It is proposed that the purpose of creating a copy of the gene is to minimize the exposure of DNA and the potential for DNA damage. The second step is *translation*, in which a protein is constructed from building blocks (*amino acids*) according to the structure encoded in the gene. The relationship among these molecules is encapsulated in the central dogma of molecular biology, illustrated in Figure 3.1.

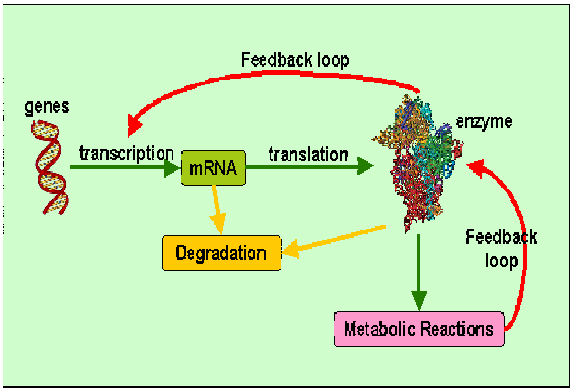
Figure 3.1: Central dogma of molecular biology, as presented in [53].



3.2 Gene Regulatory Networks

While the main flow of genetic information is from DNA to mRNA to protein, proteins are responsible for the structure, function, and regulation of cells, tissues, and organs of an organism. In particular proteins play a big part in regulating the *expression* of genes, that is, their transcription, either by acting as a mechanism for transcription or by catalyzing the reaction. *Gene regulatory networks* (GRNs) are collections of genes and their products, together with the interactions between them that collectively carry out cellular functions ([49]).

Figure 3.2: Simplified view of a gene regulatory network, as presented in [32].



The main components of a GRN are the genes, proteins, and metabolites involved in a particular metabolic pathway; see Figure 3.2 for an overview. As we saw above, both genes and proteins are derived from DNA, whereas metabolites are small biomolecules (in comparison to proteins and DNA, for example) that are involved in metabolic processes; including intermediate and waste products. Examples of common metabolites found freely available in cells are water; ATP, a molecule used for storing and releasing energy; and glucose.

Chapter 4

Algebra

Unless otherwise noted, all definitions and theorems in this section were taken from [18] and [23] and alternatively can be found in most standard commutative algebra, computational algebra or algebraic geometry textbooks.

4.1 Concepts from Ring Theory

Definition 1. *A* ring*, denoted by* $(R; +, \times; 0, 1)$*, is a set* R *with two binary operations, generically called addition and multiplication, and elements 0, 1 of* R *associated to the operations such that the following hold:* R *is closed under the operations; the operations are associative and distributive;* R *has an additive identity (0) and a multiplicative identity (1); addition is commutative and all elements of* R *have additive inverses. A ring is* commutative *if multiplication is commutative.*

Definition 2. $(R; +, \times; 0, 1)$ *is a* field *if it is a commutative ring and all nonzero elements of* R *have multiplicative inverses.*

We will denote a ring or field by its defining set R, unless it is necessary to specify its operations and identities.

Example 1. *The set of integers,* $\mathbb{Z}$*, with usual addition and multiplication form a commutative ring, as do the set of rationals,* $\mathbb{Q}$*. The rationals form a field, whereas* $\mathbb{Z}$ *does not. The set of integers modulo 4, denoted* $\mathbb{Z}/4\mathbb{Z}$ *is a ring, but not a field; however,* $\mathbb{Z}/5\mathbb{Z}$ *is a field. In fact, sets of the form* $\mathbb{Z}/p\mathbb{Z}$*, where* p *is prime, are fields; these we denote by* $\mathbb{F}_p$*.*

Definition 3. *Let* R *be a commutative ring. A* polynomial ring $R[x_1, \ldots, x_n]$ *in* n indeterminates *or* variables *is the set of finite sums* $\sum_{c \in R} c x_1^{a_1} \cdots x_n^{a_n}$*, called* polynomials*. Elements* $c x_1^{a_1} \cdots x_n^{a_n}$ *are called* terms*, consisting of a* coefficient c*, a* monomial $x_1^{a_1} \cdots x_n^{a_n}$*, and a vector of* exponents $a = (a_1, \ldots, a_n) \in \mathbb{Z}_{\geq 0}^n$*.*

While we do not include the proof, it should be noted that polynomial rings are in fact rings with usual polynomial addition and multiplication. We will focus on polynomial rings over fields as they play an important role in this work. For the remaining discussion in this chapter, we assume that k is a field. For simplicity, we denote $\{x_1, \ldots, x_n\}$ by $\mathbf{x}$ and a monomial $x_1^{a_1} \cdots x_n^{a_n}$ by $\mathbf{x}^a$.

The algebraic structure of a ring induces the following type of subset.

Definition 4. *Let R be a ring. A subset $I \subset R$ is an* ideal *if $0 \in I$, $a+b \in I$ for all $a, b \in I$, and $ra \in I$ for all $r \in R$ and $a \in I$.*

Example 2. *The set $2\mathbb{Z}$ of even integers is an ideal of $\mathbb{Z}$, whereas it is not an ideal of the reals $\mathbb{R}$. The set $\langle x \rangle := \{xf : f \in k[x, y]\}$ of polynomial multiples of x is an ideal of the ring $k[x, y]$.*

In the last example, we introduced the notion of a generating set. We say that an ideal I of $k[\mathbf{x}]$ is generated by polynomials $f_1, \ldots, f_s$ if $I = \{\sum_{i=1}^{s} h_i f_i : h_i \in k[\mathbf{x}]\}$; furthermore we write $I = \langle f_1, \ldots, f_s \rangle$.

The following concepts are required for the main result of this section, which is a ring-theoretic version of the classic Chinese Remainder Theorem.

Definition 5 (Operations on Ideals). *Let $I, J \subset R$ be ideals.*

- $I \bigcap J := \{a \in k[\mathbf{x}] : a \in I, a \in J\}$.
- $I + J := \{f + g : a \in I, b \in J\}$.
- $IJ := \{\Sigma_{finite} ab : a \in I, b \in J\}$.

Each of the above operations results in an ideal of the ring R.

Definition 6. *An ideal $I \subset R$ is said to be* maximal *if for any ideal J containing I, either $J = I$ or $J = R$. Two ideals I, J are* comaximal *if $I + J = R$.*

In other words, ideals I, J are comaximal if there are $a \in I, b \in J$ such that $a + b = 1$. An important point is that only the ideal $R = \langle 1 \rangle$ contains the multiplicative identity: if an ideal contains 1, then any element r can be written as $r \cdot 1$ and hence the ideal contains all elements of the ring.

Theorem 1 (Chinese Remainder Theorem). *Let R be a ring and $I_1, \ldots, I_t$ be pairwise comaximal ideals of R. Consider the canonical homomorphism*

$$\phi : R \to R/I_1 \times \cdots \times R/I_t.$$

Then ϕ is surjective with kernel $\bigcap I_i = I_1 \cdots I_t$.

While the proof can be found in a standard abstract algebra textbook, for example [23], we include it here as it presents some concepts that will become useful later.

Proof. First we show that if I, J are comaximal, then $IJ = I \bigcap J$. By construction of the product of ideals, $IJ \subset I \bigcap J$. Let $c \in I \bigcap J$. By comaximality, there are $a \in I, b \in J$ with $a + b = 1$. It follows that $c = ca + cb \in I \bigcap J$, achieving equality.

Next we show that the ideals I_i and $\bigcap_{j \neq i} I_j$ are comaximal. As I_i is comaximal with all other ideals, then choose $a_j \in I_i$ and $b_j \in I_j, j \neq i$ such that $a_j + b_j = 1$. Then we have that

$$1 = \prod_{j \neq i} (a_j + b_j).$$

We notice that all terms of the expanded product are elements of I_i, except for the last term $\prod_{j \neq i} b_j$, which is in $\bigcap_{j \neq i} I_j$. Therefore $1 = \prod_{j \neq i}(a_j + b_j) \in I_i + \bigcap_{j \neq i} I_j$ and the ideals are comaximal.

To prove surjectivity, let $r = (r_1 + I_1, \ldots, r_t + I_t) \in R/I_1 \times \cdots \times R/I_t$. Let $J_i = \bigcap_{j \neq i} I_j$. Then for every I_i, J_i, there are $a_i \in I_i, b_i \in J_i$ with $a_i + b_i = 1$. Note that since $b_i \in J_i \subset I_i$ for all $j \neq i$, then $r_i b_i \in I_i$, therefore $r_i b_i = 0 + I_j, j \neq i$. It is also true that $b_i = 1 - a_i$ and so $r_i b_i = r_i - r_i a_i = r_i + I_i$. It follows then that

$$\begin{aligned} \phi\left(\sum r_i b_i\right) &= \left(\sum r_i b_i + I_1, \ldots, \sum r_i b_i + I_t\right) \\ &= (r_1 + I_1, \ldots, r_t + I_t) \\ &= r \end{aligned}$$

Since $a \in \ker \phi$ for $a \in R$ iff $a \in I_i$ for every i, then $\ker \phi = \bigcap I_i = I_1 \ldots I_t$.

□

4.2 Concepts from Algebraic Geometry

Algebraic geometry is the study of the relationship between systems of polynomial equations and the set of their simultaneous solutions. A system is posed as an ideal and the set of common solutions as a *variety*. In this section we introduce some properties of ideals and varieties and present well-known theorems which describe their relationship.

Definition 7. *A subset $V \subset k^n$ is an* affine variety *if it is the set of all simultaneous zeroes of some collection of polynomials, $f_1, \ldots, f_s \in k[x_1, \ldots, x_n]$.*

Example 3. *Consider the solutions of the system of equations*

$$\begin{aligned} x^2 + y^2 &= 1 \\ x^2 - 1 &= y \end{aligned}$$

corresponding to the intersection of the unit circle centered at the origin and the standard parabola with vertex at $x = -1$. If we view these equations as polynomials $x^2 + y^2 - 1, x^2 - 1 - y \in \mathbb{R}[x, y]$, then the variety of $\{x^2 + y^2 - 1, x^2 - 1 - y\}$ is $V = \{(\pm 1, 0), (0, -1)\}$, that is, the 3 points of intersection.

For a variety V, let $\mathbf{I}$ be the map

$$\mathbf{I}(V) = \{f \in k[\mathbf{x}] : f(a) = 0 \text{ for all } a \in V\}$$

that sends varieties to ideals. For an ideal I, let $\mathbf{V}$ be the map

$$\mathbf{V}(I) = \{a \in k^n : f(a) = 0 \text{ for all } f \in I\}$$

that sends ideals to varieties. These maps present a natural relationship between ideals and varieties. Given a variety V, we always have that $V = \mathbf{V}(\mathbf{I}(V))$. However, for an ideal I, $I \subset \mathbf{I}(\mathbf{V}(I))$, with equality achieved only for certain types of ideals. For example, consider the ideal $I = \langle x^2 \rangle$ in $\mathbf{Q}[x]$. The variety of I is the singleton 0. The ideal of 0 is $\langle x \rangle$. Yet,$\langle x^2 \rangle \subsetneq \langle x^2 \rangle$. The following theorem summarizes this relationship.

Theorem 2 (Ideal-Variety Correspondence). *For any field k, the maps $\mathbf{I}$ and $\mathbf{V}$ are inclusion-reversing and $\mathbf{I}$ is injective. Moreover, if k is algebraically closed, then $\mathbf{I}$ and $\mathbf{V}$ are bijections.*

For I, J ideals, if $I \subset J$, then we have that $\mathbf{V}(I) \supset \mathbf{V}(J)$. It also follows that for all varieties V, W with $V \subset W$, $\mathbf{I}(V) \supset \mathbf{I}(W)$. Intuitively, this means that the larger the system of polynomials, the fewer the number of common solutions, and vice versa. The second part of the statement is true since the inclusion relationship gives us that $\mathbf{V}(\mathbf{I}(V)) = V$ for any variety V. We also always have that $I \subset \mathbf{I}(\mathbf{V}(I)$ for any ideal I.

The ideal $\mathbf{I}(V)$ is called the *ideal of points of V* and can be interpreted as the set of all polynomials that vanish on all points in V.

The Ideal-Variety Correspondence also tells us how unions and intersections behave with respect to the maps $\mathbf{I}$ and $\mathbf{V}$, which is summarized in the following statement.

Corollary 1. *Let I, J be ideals and V, W be varieties. Then*

- $\mathbf{V}(I \bigcap J) = \mathbf{V}(I) \bigcup \mathbf{V}(J)$
- $\mathbf{I}(V \bigcup W) = \mathbf{I}(V) \bigcap \mathbf{I}(W)$.

4.3 Concepts from Gröbner Basis Theory

Many problems in ring theory and algebraic geometry require knowing whether a ring element is contained in an ideal, the so-called ideal membership problem. For example, let $f = x^4 - 1$,

$g = x + 1$, and $I = \langle x^2 + 1 \rangle$. By factoring f, we can see that $f \in I$, but $g \notin I$ since all elements of I are polynomial multiples of a quadratic polynomial. Another way to see this is to divide f and g by $x^2 + 1$. If the remainder is 0, the dividend is in I; otherwise, it is not. So the ideal membership problem is reduced to polynomial division.

In the polynomial ring $k[x]$ in one variable, monomials have a natural ordering given by the ordering on the exponents in $\mathbb{Z}_{\geq 0}$. So comparing monomials is the same as comparing their exponents: $x^3 > x^2$ since $3 > 2$. However, when we move to polynomial rings in $n > 1$ indeterminates, we lose the natural ordering on the exponents in $\mathbb{Z}^n_{\geq 0}$. Intuitively it is no longer clear how to order monomials, such as x^2y and xy^2. Furthermore, long division of polynomials in several variables is not unique for different choices of monomial orders. To see how multivariate long division is defined, see Chapter 2.3 in [18]; an example is provided below.

Example 4. *Consider the polynomials $f = x^2 + x + y$ and $g = x + y$ in $\mathbb{R}[x, y]$. Division of f by g yields different remainders given different rules for ordering their monomials. Recall that division proceeds in descending order. If we use a* lexicographic *ordering with $y > x$, then y is the leading term of f and $f \div g$ produces a quotient of 1 and a remainder of x^2. However, if we use a* graded *ordering, in which high-degree monomials are largest, with $x > y$, then x^2 is the leading term of f and the division yields a quotient of $x - y + 1$ and remainder y^2.*

The last example demonstrates that polynomial division is dependent on the choice of monomial order, which we now define.

Definition 8. *Let k be a field. A* monomial ordering *(or* term order*) on $k[x_1, \ldots, x_n]$ is a relation $>$ on the set of monomials $\mathbf{x}^a$ such that $>$ is a total ordering,*

$$\mathbf{x}^a > \mathbf{x}^b \implies \mathbf{x}^a\mathbf{x}^c > \mathbf{x}^b\mathbf{x}^c$$

for any monomial $\mathbf{x}^c$, and $>$ is a well-ordering; i.e., every nonempty subset of monomials has a smallest element under $>$.

Note: Technically, term orders and monomial orderings differ in that the former is an ordering on the *terms* (monomial and coefficient) of a polynomial; however, here we make no distinction.

While there are an infinite number of term orders, we will primarily focus on two types, whose definitions are below.

Definition 9.

1. *(Lexicographic) Let $\mathbf{x}^a, \mathbf{x}^b \in k[x_1, \ldots, x_n]$ be two monomials. Then $\mathbf{x}^a >_{lex} \mathbf{x}^b$ if the first nonzero entry of the vector difference $a - b$ is positive.*

2. *(Graded Reverse Lexicographic) Let* $\mathbf{x}^a, \mathbf{x}^b \in k[x_1, \ldots, x_n]$ *be two monomials with* $a = (a_1, \ldots, a_n), b = (b_1, \ldots, b_n)$. *Then* $\mathbf{x}^a >_{grevlex} \mathbf{x}^b$ *if*

$$|a| = \sum a_i > |b| = \sum b_i,$$

or if $|a| = |b|$ *and the last nonzero entry of the vector difference* $a - b$ *is negative.*

Example 5. *Let* k *be a field and consider the monomials* $x^2y^3, x^4, x^5 \in k[x, y]$. *Suppose* $x > y$. *In the* lex *ordering,* $x^4 >_{lex} x^2y^3$ *since the first nonzero entry of*

$$(4, 0) - (2, 3) = (2, -3)$$

is positive. In the grevlex *ordering,* $x^2y^3 >_{grevlex} x^4$ *since*

$$|(2, 3)| = 5 > |(4, 0)| = 4.$$

On the other hand $x^5 >_{grevlex} x^2y^3$ *since* $|(5, 0)| = |(2, 3)| = 5$ *and the last nonzero entry of*

$$(5, 0) - (2, 3) = (3, -3)$$

is negative.

Now suppose that $y > x$. *Then* $y^3x^2 >_{grevlex} x^5$ *since the last nonzero entry of*

$$(3, 2) - (0, 5) = (3, -3)$$

is negative.

The ordering of the variables plays a crucial role in determining a term order. For instance, for every permutation of the variables, there is a corresponding grevlex ordering and there are $n!$ grevlex orderings for a polynomial ring in n indeterminates. The same is true for lex, as well as for all other monomial orders. We call the initial ordering of the variables a *variable order.*

We also saw in Example 4 that there is a dependence of polynomial division on the choice of term order. Given a fixed order, then the division of two polynomials proceeds as expected, producing unique results as there is no ambiguity in the *leading terms*; that is, the largest term of a polynomial under the term order.

Example 6. *Consider the division of* $x^5 + yz$ *by* $x - yz$. *For illustration purposes, assume that* $x > y > z$. *We will see that the division process is quite different for lex and for grevlex. To begin, we will divide the polynomials using grevlex.*

$$\begin{array}{rl|l} & -1 & \\ \hline -yz + x &) \; x^5 + yz & \longrightarrow \quad \underline{\textit{Remainder}} \\ \hline & yz & \\ & \text{- } (yz - x) & \\ \hline & x & \longrightarrow \quad x \\ \hline & 0 & \end{array}$$

Therefore the remainder of $x^5 + yz$ when divided by $-yz + x$ in the given grevlex ordering is $x^5 + x$.

If we now use lex, we will get a very different quotient and remainder.

$$
\begin{array}{rl}
 & \quad x^4 + x^3yz + x^2y^2z^2 + xy^3z^3 + y^4z^4 \qquad\qquad \underline{\text{Remainder}} \\
x - yz & \;) \; x^5 + yz \\
 & \;\text{-}\; (x^4 - x^4yz) \\
\hline
 & \qquad x^4yz + yz \\
 & \qquad\; \text{-}\; (x^4yz - x^3y^2z^2) \\
\hline
 & \qquad\qquad \vdots \\
\hline
 & \qquad y^5z^z + yz \qquad\qquad \longrightarrow \quad y^5z^z + yz \\
\hline
 & \qquad\quad 0
\end{array}
$$

In this case, $x^5 + yz$ divided by $x - yz$ yields a remainder of $y^5z^z + yz$.

One difference between the two term orders is that in lex, any term involving x, the largest variable, will get divided out first, whereas in grevlex (or any graded ordering, for that matter) terms of highest degree get divided out first.

A problem encountered when dividing multivariate polynomials, even if a term order has been established, is that performing successive divisions of a polynomial using different polynomials often yields nonunique results.

Say we want to divide the polynomial $f = x^5 + yz$ from the example above, by $g = x - yz$ and $h = x^4 + 1$ using lex with $x > y > z$. If we divide f by g first, we get a remainder that is a polynomial in y and z only. Since x is the largest variable, division cannot proceed with h and so f divided by g then h gives a remainder of $y^5z^z + yz$. However if we divide in the opposite order, that is by h first and then by g, we get a remainder of 0.

In terms of the ideal membership problem, this shows that $f \in \langle g, h\rangle$; though if we had not divided cleverly, it is possible that we would not have realized it. The problem lies in that the leading terms of the generators do not divide the leading terms of all elements in $\langle g, h\rangle$. What is needed is a "nice" generating set.

Definition 10. *Let $>$ be a monomial order for $k[\mathbf{x}]$ and let I be an ideal in the ring. A finite subset $G = \{g_1, \ldots, g_m\} \subset I$ is a* Gröbner basis *for I if the leading term of any $f \in I$ is divisible by one of the leading terms $LT(g_i)$ under $>$.*

Theorem 3. *Let $>$ be a monomial order for $k[\mathbf{x}]$. Every nonzero ideal $I \subset k[\mathbf{x}]$ has a Gröbner basis G. Moreover G is a generating set for I.*

Example 7. *Consider the ideal I generated by the polynomials $f = x^2 + x + y$ and $g = x + y$ from Example 4. The question is to decide whether $h = y^3$ is an element of I. As every element of I is of the form $af + bg$ for some $a, b \in \mathbb{R}[x, y]$, then we must check whether h can*

be written in this way; i.e. divide h by f and g and check for 0 remainder. If $x > y$, then division is not possible since the leading terms of f and g involve x. However, it is true that

$$y^3 = yf + (-xy + y^2 - y)g.$$

So we cannot solve the ideal membership problem in this case, because $\{f, g\}$ is not a Gröbner basis for I.

Consider the set $\{x + y, y^2\}$. We know this to be a subset of I and it can even be shown that the leading term of any $f \in I$ is divisible by x or y^2. Therefore, G is a Gröbner basis for I and now it is clear that $h \in I$.

Definition 11. *Let G be a Gröbner basis for an ideal $I \subset k[\mathbf{x}]$ and let $f \in k[\mathbf{x}]$. The* normal form *of f with respect to G, denoted $NF(f, G)$, is the remainder of f under division by the elements of G.*

To solve the ideal membership problem, we state the following well-known result.

Theorem 4. *Let G be a Gröbner basis for an ideal $I \subset k[\mathbf{x}]$ and let $f \in k[\mathbf{x}]$. Then $f \in I$ iff $NF(f, G) = 0$.*

Given a Gröbner basis G of an ideal $I \subset k[\mathbf{x}]$ with respect to a term order $>$, the quotient ring $k[\mathbf{x}]/I$ and the set of leading terms *not* in $LT(G)$ have a special relationship. Viewing the quotient ring as a vector space over k, the elements $\{\mathbf{x}^a : \mathbf{x}^a \notin \langle LT(G)\rangle\}$ form a basis for $k[\mathbf{x}]/I$; this set is called the set of *standard monomials* for I with respect to $>$. In fact, for the same ideal, there may be a different set of associated standard monomials for different term orders; however, the number of standard monomials is invariant.

The task of computing Gröbner bases requires the calculation of polynomials that allow for cancellation of leading terms. Called *S-polynomials*, they are built from all pairs of elements in the given generating set for an ideal and added to the set if certain criteria are met. The naïve approach is known to be exponential in the number of variables ([14]). However, there are a number of improved algorithms with better complexity. We conclude this chapter with a well-known algorithm of H. Möller and B. Buchberger ([44]) for computing Gröbner bases, which is quadratic in the number of variables and cubic in the number of points ([48]). Before presenting the algorithm, we provide one last definition.

Definition 12. *Let $\{p_1, \ldots, p_s\}$ be a set of distinct points in k^n. A polynomial $s_{p_i} \in k[\mathbf{x}]$ is called a* separator *of p_i if $s_{p_i}(p_i) = 1$ and $s_{p_i}(p_j) = 0$ for all $j \neq i$.*

BM: Algorithm for computing separators for a variety,
a Gröbner basis for the ideal of points, and a set of standard monomials
Input: $V = \{p_1, \ldots, p_s\}$ variety, σ = term order
Output: S = separators of V, GB = Gröbner basis of $\mathbf{I}(V)$,
SM = set of standard monomials with respect to σ

Algorithm:
S = ∅; GB = ∅; SM = ∅;
r = 0
L = [1] –L contains candidates for std monom.
while L ≠ 0
t = min(L) –smallest monomial in L
L = L \ {t}
f = t - $\sum_{i=1}^{s}$t($p_{\pi(i)}$)s_i –first point for which f does not vanish
if f vanishes on V –then f is in the ideal
GB = GB ⋃ f
L = L \ {multiples of t} –throw away multiples, we want a reduced basis
else
SM = SM ⋃ {t} –the monomial t does not divide any LT in the ideal
r = r + 1
π(r) = min{i : f(p_i) ≠ 0}
s_r = f($p_{\pi(r)}$)$^{-1}$f –a partial separator
S = S ⋃ {s_r}
for every i = 1..r-1
s_i = s_i - s_i($p_{\pi(r)}$)s_r
endfor
L = L ⋃ {x_it:i=1..n} \ LT(GB)
endif
endwhile
return S, GB, SM

We compute a small example to illustrate the various steps of the algorithm BM.

Consider the variety $V = \{(0,1),(1,0)\}$ under any term order with $x_1 < x_2$. We will execute each step of the algorithm and show what the value of each variable is.

$S=0; GB=0; SM=0; r=0; L=\{1\}$			
First pass	Second pass	Third pass	Fourth pass
enter while	*enter while*	*enter while*	*enter while*
$t=1$	$t=x_1$	$t=x_2$	$t=x_1^2$
$L=\emptyset$	$L=\{x_2\}$	$L=\{x_1^2, x_2x_1\}$	$L=\emptyset$
$f=1$	$f=x_1-1$	$f=x_2+x_1-1$	$f=x_1^2-x_1$
		enter if	*enter if*
		$GB=\{x_2+x_1-1\}$	$GB=\{x_2+x_1-1, x_1^2-x_1\}$
		$L=\{x_1^2\}$	$L=\emptyset$
enter else	*enter else*		
$SM=\{1\}$	$SM=\{1, x_1\}$		
$r=1$	$r=2$		
$\pi(1)=1$	$\pi(2)=2$		
$s_1=1$	$s_2=-x_1+1$		
$S=\{1\}$	$S=\{1, -x_1+1\}$		
	enter for		
	$s_1=x_1$		
$L=\{x_1, x_2\}$	$L=\{x_2, x_1^2, x_2x_1\}$		

When the algorithm finishes, the following are returned:

$$S=\{s_1, s_2\}=\{x_1, -x_1+1\}, GB=\{x_2+x_1-1, x_1^2-x_1\}, SM=\{1, x_1\}.$$

Chapter 5

Dynamical Systems

In Chapter 2, we described a number of different reverse-engineering methods primarily for modeling gene regulatory networks, ranging from continuous to discrete. While the behavior of a biological system may be seen as continuous, in that it moves continuously from one state to another, the technology to record observations of the system, such as microarray chips, is most certainly not continuous. In fact, dynamic models constructed from reverse-engineering methods must fit discrete instances of a continuous process. Moreover, to validate models that use continuous data often requires discretization of the data so that comparisons may be made (for example, see [9], [56]).

In this chapter, we explore *discrete-time dynamical systems*, which we simply call dynamical systems. We introduce a number of concepts related to this type of system and then the stage is set for presentation of our algebraic reverse-engineering algorithm.

Definition 13. *A* dynamical system on n nodes *is a triple* (N, X, F) *with the following properties:*

1. N *is a set of* n *variables;*
2. X *is a set of values, called* states*, that the variables can take; and*
3. $F = (f_1, \ldots, f_n) : X^n \to X^n$ *is a function in terms of the variables in* N*, called the* global function*, and each* $f_i : X^n \to X$ *is the function, called a* transition function*, associated to node* i*.*

In the above definition, the state set can be specialized to a cartesian product of sets $X_1 \times \cdots \times X_n$, in which each X_i is a state set for a node of the dynamical system. For simplicity, we assume that all nodes have the same state set.

Example 8. *Consider a dynamical system $\mathcal{S}_3 = (N, X, F)$ on 3 nodes with $N = \{x_1, x_2, x_3\}$ and $X = \mathbb{R}$. Let $F = (f_1, f_2, f_3)$ be the global function given by*

$$\begin{aligned} f_1(x_1, x_2, x_3) &= 0.1x_1^2 \\ f_2(x_1, x_2, x_3) &= \frac{x_1 + x_3}{x_2} \\ f_3(x_1, x_2, x_3) &= 2^{x_2} + 1. \end{aligned}$$

Then F evaluated at the state $(1, 1, 1)$ is calculated as follows:

$$F(1, 1, 1) = (f_1(1, 1, 1), f_2(1, 1, 1), f_3(1, 1, 1)) = (0.1, 2, 3),$$

giving the state transition $(1, 1, 1) \mapsto (0.1, 2, 3)$.

It is important to notice that state transitions come about through synchronous updating of the variables. In applications, it may be the case that the variables should be updated at different times. Such a phenomenon occurs when the nodes of the system operate at heterogeneous time scales. The theory of sequential dynamical systems provides an alternative way to describe these types of systems. For an exposition, see [37].

Two key features are well-characterized by the global function F: the structure, given by the wiring diagram of F, and the dynamics, arising from iteration of F. If the state set is finite, then the dynamics can be completely described through a graph called the *state space*, which will be defined in Chapter 6. However, if the state set is infinite, then the dynamics can only be partially identified since the global function is usually too complex and high dimensional to be studied analytically.

The importance of these features will motivate the particular characterization of reverse engineering we investigate. Since the input of any reverse-engineering program is a collection of measurements or observations of a system, we must specify the form of the data. For the purpose of constructing models from time-dependent measurements, which is often the case for gene expression data, we focus on special input-output pairs called *time series*.

Definition 14. *A* time series $T = (\mathbf{s}_1, \ldots, \mathbf{s}_t)$ *for a dynamical system $\mathcal{S}_n = (N, X, F)$ is a sequence of elements of X^n with the following property: for each $i \geq 1$, we have $F(\mathbf{s}_i) = \mathbf{s}_{i+1}$. The* time points $\mathbf{s}_i = (s_{i1}, \ldots, s_{in})$ *are n-tuples with coordinates representing the state of each node at time i. If $t < \infty$, then we say that the time series has* length t.

Example 9. *For the dynamical system $\mathcal{S}_3$ from Example 8, the following is a time series of length 4:*

$$(1, 1, 1) \mapsto (0.1, 2, 3) \mapsto (0.001, \frac{31}{20}, 5) \mapsto (10^{-6}, \frac{5001}{1550}, 2^{1.55} + 1).$$

Each time point corresponds to an observation of the system at a particular time unit and each consecutive pair $(\mathbf{s}_i, \mathbf{s}_{i+1})$ of time points gives rise to a state transition via iteration of F:

$$\mathbf{s}_i \mapsto \mathbf{s}_{i+1}.$$

In fact, the time points have a natural ordering: for each $i < t$, $\mathbf{s}_i < \mathbf{s}_{i+1}$, meaning that the observation at time i occurred before the observation at time $i+1$. If there are time series states $\mathbf{s}_i = \mathbf{s}_j$ with $i < j$, then we say that the time series has a *limit cycle of length* $j - i$. In this case, if $j = i + 1$, then the time series has a fixed point.

Time series of experimental data are collected via technological procedures, in which only a finite number of samples can be extracted; therefore, in this discourse, we will only consider time series of finite length.

In applications, if a time series is a sequence of measurements for a biological system in its natural state, we call this a *wildtype* time series. Additionally, it is often the case that a time series comes from observation of a system in a perturbed state. If the perturbation is one such that one node of the system is completely inactivated, we call such a data set a *knockout* time series. These time series can be characterized as those with the property that there is $1 \leq j \leq n$, corresponding to the inactive node, such that

$$\mathbf{s}_i = (s_{i1}, \ldots, s_{i,j-1}, 0, s_{i,j+1}, \ldots, s_{in})$$

for all $1 \leq i \leq t$. These types of perturbation data will become important in Chapter 4.

5.1 Finite Dynamical Systems

In this section we will explore properties of systems with finite state sets. Under special conditions, the structure and dynamics of such systems can be studied using algebraic methods.

Definition 15. *Let X be a finite set. A* finite dynamical system (FDS) of dimension n *is a function $F = (f_1, \ldots, f_n) : X^n \to X^n$ with each $f_i : X^n \to X$ called a* local function.

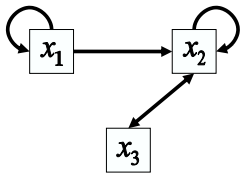
Note that $F(\mathbf{s}) = (f_1(\mathbf{s}), \ldots, f_n(\mathbf{s}))$ for $\mathbf{s} \in X^n$. Unless otherwise noted, all FDSs are n-dimensional, for a fixed integer $n > 0$.

While general set functions do not have much structure, if there is a constraint on the size of the state set X, then the local functions of an FDS can be expressed as *polynomial functions*. By stipulating that the cardinality of the state set is a power of a prime number, we can impose the algebraic structure of a finite field on the set. The primality condition allows us to exploit the following theorem (see [41], p. 369), which characterizes functions defined over finite fields.

Theorem 5 (Lidl and Niederreiter). *Let k be a finite field. Then every function $f : k^n \to k$ is a polynomial of degree at most n.*

If the state set for an FDS F is a finite field, then we call F a *polynomial dynamical system* (PDS). We denote its state set as k to distinguish it as a finite field. Below we will explore some properties of PDSs.

Figure 5.1: The dependency graph for the system in Example 11.



5.2 Polynomial Dynamical Systems

Above we saw that the wiring diagram referred to a graph depicting the connectivity structure of the system. For PDSs, we call this graph a *dependency graph*. We use both terms interchangeably. An example of a dependency graph is depicted in Figure 5.1. Dependency graphs associated to PDSs can be constructed in terms of the *support* of their local functions; that is, the collection of variables in the function.

Definition 16. *Let $f \in k[x_1, \ldots, x_n]$. The* support of f*, denoted by supp(f), is a subset $\{x_{i_1}, \ldots, x_{i_m}\}$ of $\{x_1, \ldots, x_n\}$ such that m is the smallest integer with $f \in k[x_{i_1}, \ldots, x_{i_m}]$.*

Example 10. *Let $f \in k[x, y, z]$. If $f = x^2y + 3y$, then supp$(f) = \{x, y\}$. If $f = a$, for some $a \in k$, then supp$(f) = \{\}$.*

Definition 17. *Let F be an n-dimensional PDS. The* dependency graph *of F, denoted by $\mathcal{D}(F)$ is a directed graph (V, E) where $V := \{x_1, \ldots, x_n\}$ and $E = \{(x_i, x_j) : x_i \in$ supp$(f_j)\}$.*

Example 11. *Let $k = \mathbb{F}_5$ and $F = (f_1, f_2, f_3) : k^3 \to k^3$ be the 3-dimensional PDS with local functions*

$$\begin{aligned} f_1(x_1, x_2, x_3) &= x_1^3 + 2x_1 + 4 \\ f_2(x_1, x_2, x_3) &= 3x_1^2 + x_2x_3 + x_3 \\ f_3(x_1, x_2, x_3) &= 2x_2 + 1. \end{aligned}$$

The dependency graph of F is given in Figure 5.1.

From the definition, we see that polynomial dynamical systems give rise to directed graphs on n vertices through the construction of a dependency graph. However the converse is also true: any directed graph on n vertices can be the dependency graph for a PDS.

Theorem 6. *Let D be the mapping from the set of n-dimensional PDSs to the set of directed graphs on n vertices that associates dependency graphs to PDSs. Then D is a surjective mapping.*

Proof. Let $G = (V, E)$ be a digraph with $|V| = n$. We can assume that the vertices of V are labeled as integers $1, \ldots, n$. Denote by E_i the set $\{(v_1, i), \ldots, (v_m, i)\} \subset E$ of incoming edges for a vertex i. Define $f_i \in k[x_1, \ldots, x_n]$ to be the polynomial $f_i = \sum_{j=1}^{m} x_{v_j}$. Then $F = (f_1, \ldots, f_n)$ is a PDS with $D(F) = G$. □

It is not true that D is injective. Let $F_1 = (f_1^1, \ldots, f_n^1)$ and $F_2 = (f_1^2, \ldots, f_n^2)$ be two PDSs with $supp(f_i^1) \neq supp(f_i^2)$ for some $i \leq n$. Then their dependency graphs differ on vertex i.

The above representation of PDSs lend them to be viewed as systems, as defined in Chapter 2. Iteration of a PDS produces its *dynamics* in the same way that iterating the global function of a system produces the dynamics of the system. Because the state set of a PDS is finite, we can represent the dynamics through a *finite* graph with $|X|^n$ vertices.

Definition 18. *Let F be an n-dimensional PDS. The* state space graph *of F, denoted by $\mathcal{S}(F)$, is a directed graph (V, E) where $V := k^n$ and $E = \{(a, b) : a, b \in V \text{ and } F(a) = b\}$.*

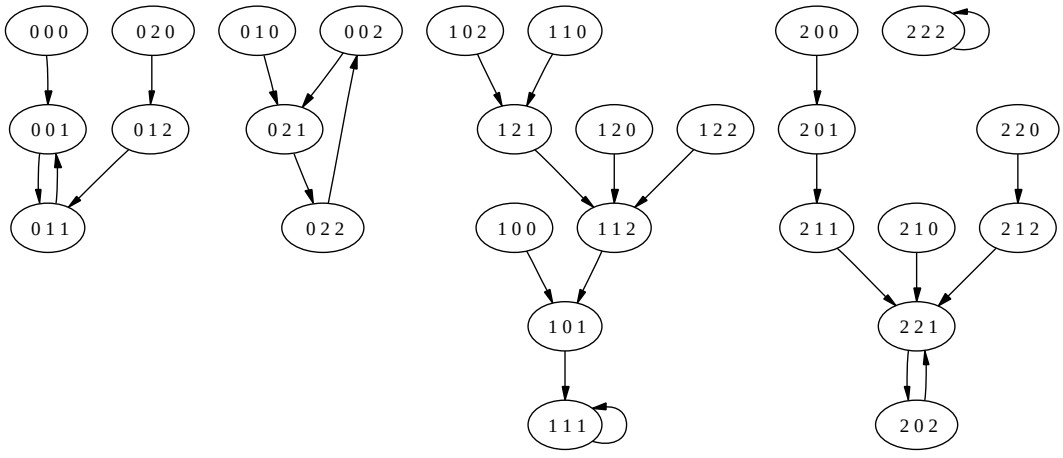
As with the wiring diagram, the edges of the state space graph (or state space, for short) represent state transitions of the function F. Below we provide an example of a state space, generated by the visualization tool DVD ([34]).

Example 12. *Let $k = \mathbb{F}_3$ and $F = (f_1, f_2, f_3) : k^3 \to k^3$ be the 3-dimensional PDS with local functions*

$$\begin{aligned} f_1(x_1, x_2, x_3) &= x_1 \\ f_2(x_1, x_2, x_3) &= x_1x_2x_3 + 2x_2 + x_3 \\ f_3(x_1, x_2, x_3) &= 2x_2^2 + x_2 + 1. \end{aligned}$$

Notice that it has the same dependency graph of F in Example 11. Its state space is given in Figure 5.2.

Figure 5.2: State space for PDS in Example 12.



PDSs over a finite field k give rise to a second class $\mathcal{G}$ of graphs, namely directed graphs on $|k|^n$ vertices such that the outdegree of each vertex is exactly 1. It follows that the indegree of each vertex is at least 1. We will see that the correspondence between the set $\mathcal{P}_k$ of PDSs over k and $\mathcal{G}$ is a bijection.

Theorem 7. *Let $S : \mathcal{P}_k \to \mathcal{G}$ be the mapping $F \mapsto \mathcal{S}(F)$. Then S is bijective.*

Proof. Let $F_1 = (f_1^1, \ldots, f_n^1), F_2 = (f_1^2, \ldots, f_n^2) \in \mathcal{P}_k$ with $F_1 \neq F_2$. Then there is $a \in k_n$ such that $F_1(a) \neq F_2(a)$. This implies that $\mathcal{S}(F_1) \neq \mathcal{S}(F_2)$ since $(a, F_1(a)) \neq (a, F_2(a))$. So S is one-to-one.

Let $G = (V, E) \in \mathcal{G}$ and for $1 \leq i \leq n$ define $\pi_i : k^n \to k$ to be the i-th projection $a = (a_1, \ldots a_n) \mapsto a_i$ for each $a \in k^n$. Consider the function $f_i : k^n \to k$ defined as $f(a) = \pi_i(b)$ for each $(a, b) \in E$. By Theorem 5, f_i is a polynomial. Hence $F = (f_1, \ldots, f_n)$ is a PDS with state space G and S is onto, thus concluding the proof. □

In summary an n-dimensional PDS $F : k^n \to k^n$ is a system $(\{x_1, \ldots, x_n\}, k, F)$ with wiring diagram $\mathcal{D}(F)$ and state space $\mathcal{S}(F)$. The next chapter is devoted to the construction of PDSs from discrete time series.

Chapter 6

An Algebraic Approach to Reverse Engineering

As we saw in the preceding chapter, there have been a number of developments in reverse-engineering algorithms. In this chapter, we will focus on reverse engineering from the context of discrete modeling.

Measurements from biological experiments often contain noise or error, due to biological variability of the organism and technological constraints of the instruments used for data collection. Furthermore, the number of data samples (i.e., time points) that are collected is limited because of the sizable costs in conducting biological experiments, rendering statistics inappropriate. One approach to resolve these issues is to discretize the data ([21]). In fact a number of reverse-engineering methods require discrete data, as seen in the last chapter. In this setting, we define the discrete reverse-engineering problem.

6.1 A Discrete Formalism of Reverse Engineering

Problem 2 (Discrete Reverse Engineering). *Let $T = (\mathbf{s}_1, \ldots, \mathbf{s}_t)$ be a time series of values in a finite field k for a system on n nodes. Let G denote the wiring diagram of the system. Find a PDS $F : k^n \to k^n$ such that*

1. (Connectivity) $G \subseteq \mathcal{D}(F)$, *and*
2. (Dynamics) $F(\mathbf{s}_i) = \mathbf{s}_{i+1}$ *for each* $i < t$.

In this setting, we call f_i an i-th *interpolator of* T, as it interpolates all pairs $(\mathbf{s}_j, s_{j+1,i})$; that is, it interpolates the i-th coordinate of every time point in T.

Table 6.1: REV-ENG: General algorithm for one time series

Input: $T = \{\mathbf{s}_1, \ldots, \mathbf{s}_t\}$ time series, *t-order* = term order with variable order
Output: F = PDS for T
Algorithm:
DataPoints := $T \backslash \{\mathbf{s}_t\}$
Sep := BM-SEP(DataPoints, *t-order*)
for each $i = 1 \ldots n$
Values := $\{s_{2i}, ..., s_{ti}\}$
$f_i := \sum_{j=1}^{t-1} \text{Values}_j \text{Sep}_j$
endfor
return PDS $F = (f_1, ..f_n)$

Recall that the goal of a reverse-engineering program is to infer the connectivity structure and dynamics of a system, *given only observations of the system.* The observations are usually time series of state transitions, collected from experimental procedures. They may also include interactions among the nodes of the system. The complete wiring diagram for a system may not be known; however, the links or edges that are known should be inferred by a reverse-engineering method. It may be the case, though, that indirect interactions are detected by the method. This arises when the data are not sufficient for describing the the structure and dynamics of a system. Therefore we impose a containment relation between the dependency graph of f and the wiring diagram G, but not a strict equality.

In this discrete setting we model biochemical systems as PDSs. Below we describe a reverse-engineering method for building PDSs given time series of discrete data. The algorithm constructs all PDSs for the data and then uses a minimality criterion to select one PDS from the set. A unique feature of this method is the ability to construct all PDSs that satisfy the given time series, as well as some portions of the wiring diagram. It must be emphasized that the set of PDSs is not constructed via enumeration. On the contrary, this is accomplished by way of a Gröbner basis, which we discussed in Chapter 4. We will show that each transition function of a given system may be reverse-engineered individually.

The algorithm is a discrete analog of the ODE method presented in [58] (see Chapter 2.3). We briefly remind the reader of the general flow of their method: first a particular solution p to the problem is obtained and then the family of homogeneous solutions, H, is constructed. Then any solution to the problem is of the form $p + h$, for some $h \in H$.

6.2 An Algebraic Reverse-Engineering Algorithm

In Table 6.1 are the steps for an algorithm given one time series of discrete states.

Consider the set of time points in $T \subset k^n$. The algorithm constructs an interpolating

polynomial, together with a set of vanishing polynomials on T. First we consider the set $V = T\backslash\{\mathbf{s}_t\}$, which can be viewed as an affine variety in k^n. The Ideal-Variety Correspondence provides a way to associate an ideal to V, namely the ideal $I = \mathbf{I}(V)$ of polynomials that vanish on V.

If we construct one interpolating polynomial f_i for each node, then the set

For each node j, we construct one local function f_j that maps each time point $\mathbf{s}_i$ to the corresponding entry $s_{i+1,j}$ of the next time point. So $(f_1, \ldots, f_n)$ is one such PDS that interpolates the time points in T. It follows that the polynomial $f_j + h$ is identical to f_j as a function on T, for any vanishing polynomial $h \in I$. Therefore, the set

$$(f_1, \ldots, f_n) + I^n := \{(f_1 + h_1, \ldots, f_n + h_n) : h_i \in I\}$$

represents all PDSs that fit T. The task is then to select a minimal polynomial dynamical system from this set, which we discuss later in the section.

Let BM-SEP be the BM algorithm where only the set of separators is returned. Recall that the BM algorithm, described in Chapter 4.3, computes separators for a set $\{\mathbf{s}_1, \ldots, \mathbf{s}_m\}$ of points in k^n. Each separator is a polynomial $r_i(x_1, \ldots, x_n)$ and is associated to a point $\mathbf{s}_i$; that is, $r_i(\mathbf{s}_i) = 1$ and $r_i(\mathbf{s}_j) = 0$ for all other points $\mathbf{s}_j$. So a polynomial f that maps points $\mathbf{s}_i$ to corresponding values b_i can be written as

$$f(x_1, \ldots, x_n) \;=\; \sum_{i=1}^{m} b_i r_j(x_1, \ldots, x_n). \tag{6.1}$$

When computed, each separator is reduced with respect to the Gröbner basis G of $\mathbf{I}(V)$. In fact, an interpolator is also reduced if written as in Equation 6.1. Therefore, we say that a PDS $F = (f_1, \ldots, f_n)$ is *minimal* if each f_i is reduced with respect to G.

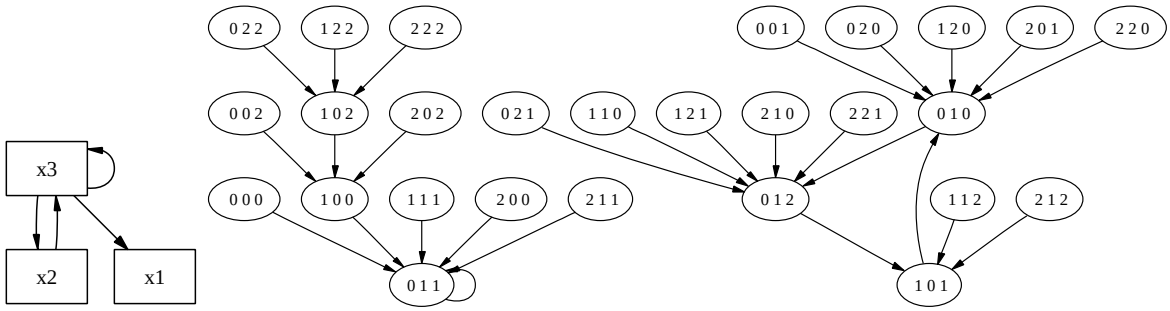
We illustrate the algorithm with a small artificial gene network $\mathcal{S}_3 = (\{x, y, z\}, \mathbb{F}_3, F)$. The elements of $\mathbb{F}_3$ can be interpreted as the states *increase in gene expression* (1), *no change in expression* (0), and *decrease in expression* (-1). Consider the following time series of state transitions for three genes, labeled x, y, and z:

Time	x	y	z
1	-1	-1	-1
2	1	0	-1
3	1	0	0
4	0	1	1
5	0	1	1

Take

$$V = \{(-1, -1, -1), (1, 0, -1), (1, 0, 0), (0, 1, 1)\}$$

Figure 6.1: Dependency graph and state space for $\mathcal{S}_3$.



and assume that all computations will be performed under the grevlex ordering with $x > y > z$. Using the BM-SEP algorithm, the computed separators associated to each point are

$$\begin{aligned} r_1(x,y,z) &= -z^2 - y - z \\ r_2(x,y,z) &= y - z \\ r_3(x,y,z) &= -z^2 + 1 \\ r_4(x,y,z) &= -z^2 - z \end{aligned}$$

One can verify that these polynomials are in fact separators of the points in V. Consider the set of values $\{1,1,0,0\}$ corresponding to the first column of data; that is, the set of states for gene x. Then we write

$$\begin{aligned} f_1(x,y,z) &= 1 \cdot r_1(x,y,z) + 1 \cdot r_2(x,y,z) + 0 \cdot r_3(x,y,z) + 0 \cdot r_4(x,y,z) \\ &= -z^2 + z. \end{aligned}$$

Computing the other two polynomials in this way, we get

$$\begin{aligned} f_1(x,y,z) &= -z^2 + z \\ f_2(x,y,z) &= z^2 - z + 1 \\ f_3(x,y,z) &= -z^2 + y + 1. \end{aligned}$$

While the algorithm REV-ENG does not output the Gröbner basis of the ideal, it is still computed by BM-SEP and so we include it here for the purpose of illustration.

$$G = GB(\mathbf{I}(V)) = \{x + y - 1, yz - z^2 + y - z, y^2 - z^2 + y - z, z^3 - z\}.$$

Notice that if we add a polynomial multiple of any element of G to one of the f_i produces another interpolating function.

6.3 Complexity Analysis of REV-ENG

We include the steps of the algorithm REV-ENG for ease of explanation.

1. DataPoints := $T \backslash \{\mathbf{s}_t\}$
2. Sep := BM-SEP(DataPoints, *t-order*)
3. for each $i = 1 \ldots n$
4. Values := $\{s_{2i}, ..s_{ti}\}$
5. $f_i := \sum_{j=1}^{t-1} \text{Values}_j \text{Sep}_j$
6. endfor

Let t be the number of time points, n the number of nodes, and p the characteristic of the field k.

Line 1. Constructing the *DataPoints* list takes $O(nt)$ time.

Line 2. As stated in [4], this step requires

$$O(t^2(g(t,n)+t)(\log p)^2 + t^2n^2)$$

time, where $g(t,n)$ is $O(t^{\frac{n-1}{n}})$ ([13]). Since g is sublinear in t, the worst-case complexity of this step is quadratic in the number of variables and cubic in the number of data points ([48]).

Line 4. To construct the set *Values* requires $\theta(t)$ time.

Line 5. Since there are $\theta(t)$ separators, construction of each f_i requires $\theta(t)$ time.

Line 3-6. Execution of the *for* loop requires $\theta(nt)$ time.

In summary, the worst-case complexity of REV-ENG is

$$O(n) + O(t^2(g(t,n)+t)(\log p)^2 + t^2n^2) + O(nt).$$

In practice, $n \gg t$ and so the complexity is dominated by the quadratic term in n.

6.4 Extensions of the General Algorithm

The algorithm REV-ENG is designed to compute a minimal PDS given a time series of discrete states, where the minimality condition is subject to a term order. From this PDS, the connectivity and dynamics can be inferred from the dependency graph and state space. One drawback of the algorithm is its dependence on a term order, in that an artificial relationship is imposed on the nodes of a system by way of the variable order. As we saw

in Chapter 4.3, the normal form of a polynomial can be different for different choices of a term order. More importantly, the dependency graph of a PDS can change for different choices of a term order, thereby affecting the process of inference. In this section we present a modification of REV-ENG to counteract this dependency. We also introduce two other modifications of the general algorithm to account for multiple time series, including those from perturbations (knockouts) of the system.

In applications, it may be the case that some information is known about the wiring diagram of a biological system of interest. This information may simply be the identification of interaction between nodes of the system, but may also include detailed information about the strength and type of interaction. The knowledge of interaction can be incorporated into the reverse-engineering process by specifying a variable order. A variable order can be assigned for each node x in the following way: the variables that are adjacent to x are ordered least and the rest of the variables are ordered greatest. Then a lexicographic or elimination ordering can be used. As we saw in Chapter 4.3, computations using a lexicographic order will result in normal forms that are in terms of the variables ordered least and potentially of high degrees.

The following algorithm makes use of this capability and takes as input one time series and a set of variable orders.

REV-ENG-D: Algorithm for one time series and fixed variable orders.

Input: $T = \{\mathbf{s}_1, \ldots, \mathbf{s}_t\}$ time series, VOrder = set of variable orders, *t-order* = term order
Output: F = PDS for T

Algorithm:
DataPoints := $T \backslash \{\mathbf{s}_t\}$
TOrder := $\{(t\text{-}order \text{ with } vo) : vo \in \text{VOrder}\}$
for each $i = 1 \ldots n$
Sep := BM-SEP(DataPoints, TOrder_i)
Values := $\{s_{2i}, ..s_{ti}\}$
$f_i := \sum_{j=1}^{t-1} \text{Values}_j \text{Sep}_j$
endfor
return PDS $F = (f_1, ..f_n)$

As mentioned above, fixing a term order imposes relations on the variables. Through the last example, we saw that the variable order can be chosen appropriately so that the dependency graph of the reverse-engineered PDS is similar to a given wiring diagram. In the situation where no information about the wiring diagram is known, then a default variable order is used by the algorithm REV-ENG. However, it would be advantageous to be able to make inferences about the structure and dynamics of a system for different choices of variable orders. The algorithm REV-ENG-R produces a PDS for m random variable orders, for a

fixed value of m. A slightly modified version of this algorithm was proposed by Allen *et al.* in [10], in which only a wiring diagram is constructed from the data. We extended the algorithm to construct a full PDS.

REV-ENG-R: Algorithm for one time series and random variable orders.

Input: $T = \{\mathbf{s}_1, \ldots, \mathbf{s}_t\}$ time series, m = #variable orders, *t-order* = term order
Output: F = PDS for T

Algorithm:
DataPoints := $T \backslash \{\mathbf{s}_t\}$
$V = n \times n$ matrix
for each $c = 1 \ldots m$
randomvo = random variable order
Sep := BM-SEP(DataPoints, *t-order* with *randomvo*)
for each $i = 1 \ldots n$
Values := $\{s_{2i}, ..s_{ti}\}$
$f_i := \sum_{j=1}^{t-1} \text{Values}_j \text{Sep}_j$
row $V[i] = supp(f_i)$
endfor
endfor
REV-ENG-D(T, V, *t-order*)
return PDS $F = (f_1, ..f_n)$

Up to this point, we have only discussed reverse engineering from a single time series. In general any one time series will vastly underdetermine a system. Incorporating information from several experiments is vital for making inferences, especially when those experiments correspond to perturbations of the system. The last algorithm accepts multiple time series, where the time series may be wildtype time series, those collected from a system operating under a normal conditions, or knockout time series, those corresponding to a perturbation in which the interactions of one node of a system are blocked or "knocked out."

REV-ENG-M: Algorithm for multiple time series.

Input: $WT = \{wt_1, .., wt_l\}$ set of wildtype time series,
$KO = \{(j, ko_j) : j \in A \subset \mathcal{P}(N)\}$ set of knockout time series with indices,
t-order = term order
Output: F = PDS for T

Algorithm:
for each $i = 1 \dots n$
Data = $WT \bigcup (KO \backslash \{(i, ko_i)\})$
DataPoints = $WT \bigcup (KO \backslash \{(i, ko_i)\})$ minus last time point in each series
Sep := BM-SEP(DataPoints$_i$, *t-order*)
Values := $\{s_{ki} : k > 1, \mathbf{s}_k = (s_{k1}, \dots, s_{ki}, \dots, s_{kn}) \in \text{Data}\}$
$f_i := \sum_{j \in Values} \text{Values}_j \text{Sep}_j$
endfor
return PDS $F = (f_1, .. f_n)$

All of the algorithms have been implemented in *Macaulay2* ([28]) and a toolbox for building PDSs from discrete time series data is currently under development with Michael Stillman, one of the authors of *Macaulay2*. A web-based version of the algorithms is available at [22].

6.4.1 Complexity Analysis

Let t, n, and p be as above, and let m be the number of variable orders.

REV-ENG-D

We have already analyzed the complexity for all steps, except for construction of the set TOrder. Its construction requires $O(nm)$ time. Overall, this algorithm has worst-case complexity of

$$\begin{aligned} &O(nt) + O(nm) + O(n)[O(B) + O(t) + O(t)] \\ &= O(nt) + O(nm) + O(nB) \end{aligned}$$

where $O(B)$ is the complexity for the BM-SEP algorithm as reported in Section 6.3. Since $O(nB)$ is the dominating term, REV-ENG-D is cubic in both n and t.

REV-ENG-R

The construction of the matrix V is $O(n^2)$. Creating a random variable order requires $O(n)$ time, as well as updating row i of V. Therefore, this algorithm has worst-case complexity of

$$O(nt) + O(n^2) + O(m)[O(n) + O(B) + O(n)(O(t) + O(t) + O(n))] + O(RD)$$

$$= O(nt) + O(nm) + O(B(n+m)) + O(mnt) + O(mn^2)$$

where $O(RD)$ is the complexity for the REV-ENG-D algorithm. In practice $O(n) = O(m)$ and as above, the complexity of REV-ENG-R is dominated by $O(nB)$.

REV-ENG-M

Here we assume that the maximum length of any time series is n data coming from experiments will typically have fewer time points than variables. Construction of the sets Data and DataPoints requires about the same number of steps, namely $O(n^2(l+|A|))$, where $|A|$ indicates the number of knockout time series. Computation of the set Values takes $O(n(l+|A|)$ time. The complexity of all other steps has been previously computed. Therefore, we conclude that the complexity of REV-ENG-M is

$$O(n)[O(n^2(l+|A|)) + O(B) + O(n(l+|A|)) + O(n(l+|A|))].$$

with $O(nB)$ being the dominating term, in practice.

In summary, the worst-case complexity of each algorithm is dominated by $O(nB)$, where $O(B)$ is known to be cubic in t and quadratic in n.

6.5 Some Theoretical Considerations

In this section we provide conditions for determining if the first part of the Reverse-Engineering Problem can be solved; that is, when the dependency graph of a constructed PDS contains all known interactions.

Suppose we are given a time series $T = (\mathbf{s}_1, \ldots, \mathbf{s}_t)$ and a wiring diagram (W, E). Consider the variety $V = \{\mathbf{s}_i : i < t\}$ of all but the last time point. Let $W_i = \{x_j : (x_j, x_i) \in E\}$ be the set of edges adjacent to node i in the wiring diagram. Under certain conditions, the elements of W_i will be standard monomials and thus will have a chance to be in the support of some polynomial, in particular the interpolator for node i. If the variables in W_i are ordered least, then either a lexicographic or elimination ordering may be used in the reverse-engineering algorithms to increase the likelihood of the desired variables appearing in the i-th interpolator.

Proposition 1. *Let $X = \{x_1, \ldots, x_n\}$ and G be a Gröbner basis for an ideal $I \subset k[X]$ with respect to a fixed term order $>$ and let $W \subset X$. Suppose that for every $x \in W$ there is $m > 1$ such that $x^m \in LT(G)$ and for every $y \in X \backslash W$, we have $y \in LT(G)$. Then $W \subset SM(G)$.*

Proof. Take x and y as in the statement of the proposition. Recall that $LT(G)$ is the set of leading terms of the elements of G and $SM(G)$ the set of standard monomials for I with respect to G. If $y \in LT(G)$, then y is not in the set of standard monomials. If $x^m \in LT(G)$

for $m > 1$ but not for $m = 1$, then $x^{m'}$ is a standard monomial for every $m' < m$; in particular x is a standard monomial. Therefore $W \subset SM(G)$. □

If the support of an interpolator f_i is contained in the set W_i of variables adjacent to a fixed node i, then all edges identified by the function are correct. It may be the case, however, that not all known edges have been identified, given the data. If $supp(f_i) \subsetneq W_i$, then more data points are needed to infer the missing edges (see [36] for more details).

Proposition 2. *Let V be a variety and G be a Gröbner basis for the ideal of points in V. If $W \subset SM(G)$ and $|W| = |V| - 1$, then $supp(f) \subset W$ for all nonconstant $f \in k[\mathbf{x}]$.*

Proof. Since $|V| = |SM(G)|$, then if W has one less element than V, it must be that W contains all standard monomials $\neq id$. Therefore, $supp(f) \subset W$ for any nonconstant polynomial $f \in k[\mathbf{x}]/\mathbf{I}(V)$. □

Given a system, the data derived from the system can determine local properties of its wiring diagram. Specifically if there is a node i that is constant on at least all but the last time point, then that node will have no outgoing edges since the variable x_i will not appear in the normal form for any polynomial under any term order. Similarly if the product of two columns is constant on at least all but the last time point, then $x_i x_j$ is not a standard monomial for any term order.

Proposition 3. *Let $V \subsetneq k^n$ be a variety and G be a Gröbner basis for $\mathbf{I}(V)$. Suppose there is a coordinate i such that for every $\mathbf{a} \in V$, $\pi_i(\mathbf{a}) = c$ for some $c \in k$. Then for every $f \in k[x_1, \ldots, x_n]$, $NF(f, G)$ does not contain the variable x_i.*

Proof. Recall that π_i is the mapping of an n-tuple onto its ith coordinate. Suppose there is i such that $\pi_i(\mathbf{a}) = c$ some $c \in k$. This holds iff $x_i - c \in I = \mathbf{I}(V)$. This element is also in G by definition of a Gröbner basis: there is $g \in G$ such that $LT(g)$ divides $LT(x_i - c) = x_i$. Then it follows that $x_i - c \in G$ iff $LT(x_i - c) = x_i$ is not a standard monomial. Since the standard monomials form a basis for $k[x_1, \ldots, x_n]/I$, then $supp(NF(f, G)) \bigcap \{x_i\} = \emptyset$ for any $f \in k[x_1, \ldots, x_n]$. □

Proposition 4. *Let $V \subsetneq k^n$ be a variety and G be a Gröbner basis for $\mathbf{I}(V)$. Suppose there are coordinates i, j such that for every $\mathbf{a} \in V$, $\pi_i(\mathbf{a})\pi_j(\mathbf{a}) = c$ for some $c \in k$. Then for every $f \in k[x_1, \ldots, x_n]$, $NF(f, G)$ does not contain the monomial $x_i x_j$.*

Proof. Suppose that for i, j fixed, $x_i x_j(\mathbf{a}) = c$ for all $\mathbf{a} \in V$. Let $I = \mathbf{I}(V)$. Then $x_i x_j \in LT(I)$ iff $x_i x_j \notin SM(G)$, the standard monomials for I with respect to G. As $x_i x_j$ is not a standard monomial, then $x_i x_j \notin supp(NF(f, G))$. □

When a variable is constant, no information about which other variables it affects can be extracted from a model. Especially if the constant variable is thought to have a substantial

impact on the regulation of the system, an experiment in which this variable is changing should be proposed.

Chapter 7

Applications and Results

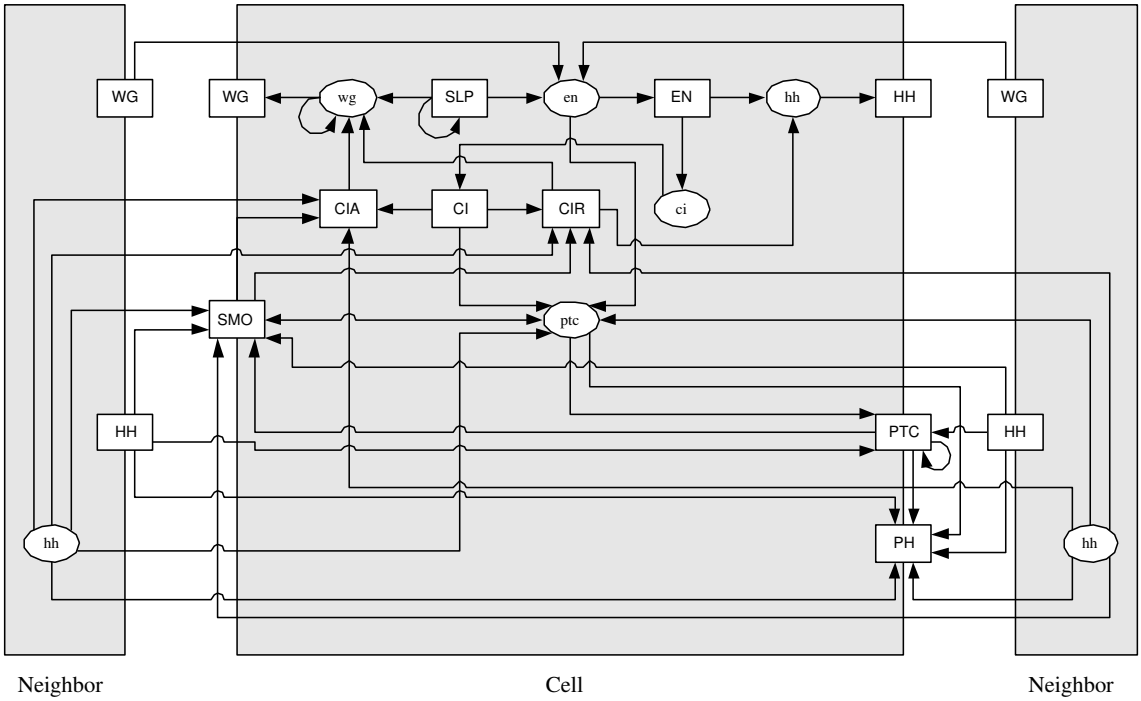
Here we focus on the application of the algorithms described in the previous chapter to two simulated networks. The first application serves to validate the algebraic methods on a published Boolean network for a gene regulatory network in the fruitfly *Drosophila melanogaster*. The second application is for discovery purposes, in which a simulated, but sufficiently complex system is investigated. We close the chapter with a description of an ongoing project to apply the algebraic methods to a real GRN in the yeast *Sacchromyces cerevisiae*.

7.1 Application to Simulated Data for Validation

When designing modeling tools, it is important to test them on systems for which much is known so that the amount of information that is identified by the model can be measured. To this end, we validated our method by applying it to a simulated dataset from a well-studied network of segment polarity genes in *D. melanogaster* embryo. In [9] a Boolean model was proposed for the network of 5 genes and their associated proteins. The network consists of a ring of 12 interconnected cells, grouped into 3 parasegment primordia, in which the genes are expressed in every fourth cell. Our goal was to reverse-engineer the network of interactions between molecular species, as well as the dynamics through identification of additive and nonadditive interactions. Note that for our purposes, it is irrelevant whether the Boolean model is indeed correct.

The genes represented in the Boolean model are *wingless* (*wg*), *engrailed* (*en*), *hedgehog* (*hh*), *patched* (*ptc*), and *cubitus interruptus* (*ci*). Also included are the proteins encoded by these 5 genes, as well as 2 other compounds (*smoothened* protein, denoted by SMO, and *sloppy-paired* proteins denoted as one compound SLP), constituting 15 distinct molecular species. Figure 7.1 depicts the wiring diagram of connections in the Boolean model.

Figure 7.1: The graph of interactions in one cell of $\mathcal{N}$ with cellular interactions, as presented in [38]. Ovals = mRNAs, rectangles = proteins. SLP denotes a protein which is believed to activate the segment polarity genes depicted in the model (Cadigan *et al.*, 1994). PH is a protein complex formed by the binding of HH to PTC (Ingham and McMahon, 2001). The protein SMO is encoded by the gene *smoothened*. Because its transcription is not regulated by any molecular species in the model, *smoothened* is not represented.



In the graph, nodes represent mRNAs and proteins. An edge between nodes indicates that the node at the tail is involved in the regulation of the head node. For example, an edge A→B between protein nodes A and B implies that A regulates the synthesis of B, whereas an edge A→b from protein A to mRNA b implies that A regulates the synthesis of b, that is, the transcription of gene b. Note that edges denote the existence of regulation, not the type, whether activation or inhibition. Table 7.1 lists the polynomial representations of the Boolean functions that accompany the model in Figure 7.1. The Boolean functions, given in polynomial form, can be found in Table 7.1.

Table 7.1: Polynomial representations of the Boolean functions in $\mathcal{N}$, together with the legend of variable names, as given in [38]. The subscript i denotes a particular cell of the ring.

$$
\begin{array}{lcl}
F_1 & = & x_1 \\
F_2 & = & (x_{15}+1)\left(x_1x_{14}+x_2\left(x_1+x_{14}+x_1x_{14}\right)+x_1x_2x_{14}\left(x_1+x_{14}+x_1x_{14}\right)\right) \\
F_3 & = & x_2 \\
F_4 & = & \left(x_{16}+x_{17}+x_{16}x_{17}\right)(x_1+1) \\
F_5 & = & x_4 \\
F_6 & = & x_5\left(x_{15}+1\right) \\
F_7 & = & x_6 \\
F_8 & = & x_{13}\left(\left(x_{11}+x_{20}+x_{11}x_{20}\right)+x_{21}+\left(x_{11}+x_{20}+x_{11}x_{20}\right)x_{21}\right)(x_4+1) \\
 & & \left(x_{13}\left(x_{11}+1\right)\left(x_{20}+1\right)\left(x_{21}+1\right)+1\right) \\
F_9 & = & x_8+x_9\left(x_{18}+1\right)\left(x_{19}+1\right)+x_8x_9\left(x_{18}+1\right)\left(x_{19}+1\right) \\
F_{10} & = & \left(x_8+x_9\left(x_{18}+1\right)\left(x_{19}+1\right)+x_8x_9\left(x_{18}+1\right)\left(x_{19}+1\right)\right)\left(x_{20}+x_{21}+x_{20}x_{21}\right) \\
F_{11} & = & x_8+x_9Y+x_8x_9Y+1+x_{20}+\left(\left(x_8+x_9Y+x_8x_9Y+1\right)x_{20}\right)+x_{21} \\
 & & +\left(x_8+x_9Y+x_8x_9Y+1+x_{20}+\left(x_8+x_9Y+x_8x_9Y+1\right)x_{20}\right)x_{21} \\
F_{12} & = & x_5+1 \\
F_{13} & = & x_{12} \\
F_{14} & = & x_{13}\left(\left(x_{11}+x_{20}+x_{11}x_{20}\right)+x_{21}+\left(x_{11}+x_{20}+x_{11}x_{20}\right)x_{21}\right) \\
F_{15} & = & x_{13}\left(x_{11}+1\right)\left(x_{20}+1\right)\left(x_{21}+1\right)
\end{array}
$$

SLP_i	wg_i	WG_i	en_i	EN_i	hh_i	HH_i	ptc_i
x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
PTC_i	PH_i	SMO_i	ci_i	CI_i	CIA_i	CIR_i	
x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	
WG_{i-1}	WG_{i+1}	HH_{i-1}	HH_{i+1}	hh_{i-1}	hh_{i+1}	Y	
x_{16}	x_{17}	x_{18}	x_{19}	x_{20}	x_{21}	$(x_{18}+1)(x_{19}+1)$	

The Boolean model, denoted by $\mathcal{N}$, consists of 60 nodes: 15 mRNAs and proteins indexed by each of the four cells. Each cell of the ring is assumed to have the same network of segment polarity genes. For the purpose of reverse engineering, we consider the network as containing

15 nodes, representing one of the four cells, and each node has an initial configuration for each cell. To account for intercellular connections, we include 6 extra variables. We focus our efforts on inferring the wiring diagram for the 15 variables from time series generated by the Boolean network, though we also report various findings in relation to the dynamics.

We used the Boolean initializations for the wildtypes published in [9] for the 5 genes and generated times series using the Boolean functions. All of the initializations terminate in steady states when evaluated by the Boolean functions in Table 7.1, as reported by [9]. Using these data, we applied the REV-ENG algorithm with the term order grevlex with $x_1 > \cdots > x_{21}$, resulting in the following PDS:

$$\begin{aligned}
f_1 &= x_1 \\
f_2 &= x_2 \\
f_3 &= x_2 \\
f_4 &= x_{16} \\
f_5 &= x_4 \\
f_6 &= x_5 \\
f_7 &= x_{12} + 1 \\
f_8 &= x_9 + x_{11} + x_{16} + x_{17} + x_{18} + x_{19} \\
f_9 &= x_8 + x_{17} \\
f_{10} &= x_{20} + x_{21} \\
f_{11} &= x_8 + x_{17} + x_{20} + x_{21} + 1 \\
f_{12} &= x_5 + 1 \\
f_{13} &= x_{12} \\
f_{14} &= x_{13} + x_{17} \\
f_{15} &= x_{17}
\end{aligned}$$

Quick inspection reveals that our model from the general reverse-engineering algorithm produces the following results: 16 of the 44 edges were correctly identified with 10 false positives, a detection rate of 36%. We note here that the Boolean functions $F_1, F_3, F_5, F_7, F_{12}$, and F_{13} were completely identified (40% detection rate). The remaining 9 functions were inferred to be linear, whereas the actual Boolean functions are of higher degree, ranging from 2 to 6.

The size of the state space is 2^{21}, involving multiple components. Any single trajectory in that space vastly underdetermines the network. Therefore we include knock-out time series for each gene in the network. Altogether we used 24 time series: one for the wildtype for each cell and one for each gene knock-out for each cell. As the length of each time series is at most 8 time steps, constituting a total of 127 time points, the data still comprises only a minuscule fraction of the state space, less than $(6.06 \times 10^{-3})\%$ of 2^{21} total states.

7.1.1 REV-ENG-M

To simulate an experiment in which node x_i representing a gene is knocked out, we set its corresponding transition function f_i in Table 7.1 to 0 and kept all other functions the same. When applicable, we also set the corresponding functions in neighboring cells equal to 0. For example, to simulate the knock out of the hedgehog gene, we set $f_6 = 0$, $f_{20} = 0$, and $f_{21} = 0$, where f_{20} and f_{21} are the functions associated to the gene in neighboring cells. We also set the i-th entry, corresponding to the initial mRNA concentration for x_i, in the wildtype initialization to 0. For each knock-out, we generated a new time series, which also ended in a steady state, by iteration of the functions given the modified initializations.

For this experiment, we used the same term order as above. An application of the REV-ENG-M algorithm for multiple time series and one term order, resulted in increased true, as well as false positives. The algorithm detected 51 total edges, where 37 were correctly identified. The Boolean functions $F_1, F_3, F_5, F_7, F_{12}$, and F_{13} have been correctly identified using this one term order. The functions F_4 and F_6 cannot be completely identified since they have terms in the Gröbner basis of the ideal of points: these terms cannot be identified with the given data. This point will be discussed in greater detail in the next section. What remains to be done is to identify the connectivity relations and dynamics for the other 7 functions.

As we saw in Chapter 4.3, the effect of a variable ordering is that the "cheaper" variables, those that are ordered least, are used preferentially in computing interpolators. Since we aim to reverse-engineer the wiring diagram of the Boolean model, it is especially important not to impose an artificial ordering on the variables. In order to counteract this dependency, we used the algorithms REV-ENG-M, together with REV-ENG-D for fixed variable orders. We also applied REV-ENG-M/REV-ENG-R to test the effectiveness of using random variable orders.

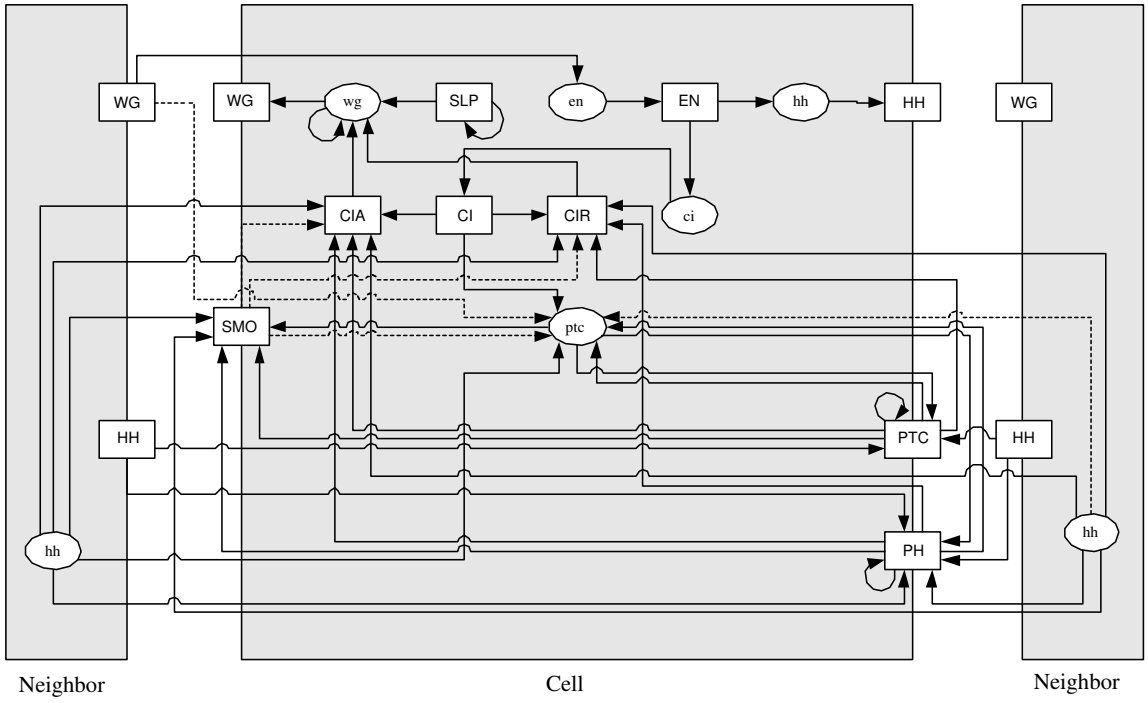
7.1.2 REV-ENG-M/REV-ENG-D

For the REV-ENG-M/REV-ENG-D experiment, we used the following four variable orders to define four grevlex term orders:

$$x_1 > \cdots > x_{21} \text{ (default order)}$$
$$x_1 < \cdots < x_{21} \text{ (reverse order)}$$

and two other orders making the "interior" variables greatest and least. The dependency graph of the PDS that is output has 41 edges, where 33 are common to the wiring diagram of the Boolean model. If we allow for partial detection, then the results improve slightly. In 3 of the 4 variable orders used, 46 edges are in the dependency graph, where 37 are correctly identified (see Figure 7.2). For this experiment, we provide a detailed account of some of the false positives and true negatives.

Figure 7.2: The dependency graph of the PDS built using REV-ENG-M/REV-ENG-D with the wildtype and knockout time series, as presented in [38]. Solid lines are links that appear for all 4 variable orders, whereas dashed lines are links that appear for 3 of the 4 variable orders.



In determining which biomolecules affect the transcription of the gene *hh*, represented by function 6, we found a polynomial function that involves fewer terms than its counterpart in the Boolean model. Specifically, the function $f_6 = x_5$ is in terms of the variable representing EN only, instead of both EN and CIR proteins. It correctly interpolates all time points in the data generated by the corresponding Boolean function $F_6 = x_5(x_{15}+1)$. The discrepancy lies in the fact that $x_{15} + 1$ is an element of the Gröbner basis G for the ideal of points. However, links whose effects are not reflected in the given data are not detectable by any reverse-engineering method unless prior information about the link is given. In this case, the variable 15, representing the protein CIR, always takes on the value 1 on all data sets, and its effect on EN is not detectable; we saw this phenomenon in Chapter 6.5. Similarly, the Boolean function F_4 for *en* also contains such terms that are in G, which accounts for lack of regulation detection.

In f_{10} for the protein complex PH, we detected 5 of the 6 of the appropriate molecules as regulators and failed to identify regulation by $x_9 =$ PTC. For every variable order, terms of the form $x_9x_j + x_j$ or x_9x_j, for nearly half of the variables x_j, can be found in the Gröbner basis of the ideal of points for f_{10}. We also identified x_{10} as its own regulator. Here we refer to the network to understand the discrepancy. PH is a protein complex formed by the binding of HH from adjacent cells to the receptor PTC. In [9] the authors assumed in their model that this binding occurs instantaneously since it is known that the reaction occurs faster than transcription or translation (which they also presuppose to require 1 time unit for completion). Therefore, we attribute the misidentification to the binding rate not being properly represented in the data and call this an *indirect effect*. Similarly for the function F_{11}, we detected an indirect effect from extracellular *hh*, as well as the correct direct effects from 3 other molecules.

Next we focus on reverse-engineering the dynamics of the Boolean network. As pointed out above, the functions in the Boolean model contain terms that evaluate to 0 on all input data, and so we are unable to detect the corresponding relationships. To compare the dynamics predicted by our model with the Boolean network, one approach is to compute the normal forms of the polynomials in Table 7.1 with respect to the ideal of time points. As the reduction depends on a term order, for each choice of term order, the normal forms of the transition functions in the Boolean network and the local functions reverse-engineered PDS agree exactly. However, this observation occurs for the following reason.

Let $D = \{(a_1, b_1), \ldots, (a_t, b_t)\}$ be a collection of input-output pairs and suppose that f, g are two polynomials that interpolate D. For each $1 \leq i \leq t$, $f(a_i) = g(a_i) = b_i$. Then the polynomial $f - g$ vanishes on all a_i and $f - g \in \mathbf{I}(a_1, \ldots, a_t)$. Since reduction with respect to a Gröbner basis is unique, we have that f and g are equivalent after reduction by being equal on the data.

The dependence of the algebraic methods on a term order may result in the particular form of the reverse-engineered functions to be not directly interpretable with respect to regulatory relationships. Here we extract information about network dynamics from terms common to

the reverse-engineered functions for the multiple term orderings used.

For each term ordering, the model constructed only from the wildtype is linear, whereas using the 4 term orders mentioned above, we found 19 terms consisting of a single variable, in which 10 are true positives. These terms, which we call "single interactions," account for 77% of the linear terms in the Boolean network for one cell of the ring. However, the degrees of the polynomial functions in $\mathcal{N}$ range from 1 to 6. Incorporating knock-out data yields more comprehensive results, highlighted in the following discussion.

In all models built from the knock-out time series, there are 18 linear terms. Of these, 12 are in one cell of $\mathcal{N}$, accounting for 92% of the linear terms present. Specifically, the linear terms in the functions for *hh* and for all the proteins, excluding the complex PH and the transcriptional forms CIA and CIR of the protein CI, were completely identified. In three of the four models, we found 21 linear terms, of which 12 are in one cell of the Boolean model.

As distinct from the models built from wildtype data only, there are nonlinear terms in the models from the knock-out data. These nonlinear or cross terms can be considered as "cooperative interactions" between nodes. For the protein SMO, we found that its synthesis depends on the cooperative interaction between the genes *ptc* and extracellular *hh*. Specifically, the terms x_8x_{20} and x_8x_{21} appear in the polynomial function for SMO for all term orders used, of which both appear in the corresponding Boolean function. In the function describing transcription of *wg*, the term x_1x_{14} is common to all models and x_2x_{15} appears in three of the four models. Both of these products are terms in the Boolean function for *wg*. In fact, we found 3 nonlinear terms common to all models. In three of the four models, there are 11 nonlinear terms, of which 8 are in $\mathcal{N}$, accounting for 27% of the quadratic terms. While $\mathcal{N}$ contains polynomial functions of degree as high as 6 involving 77 superquadratic terms, our method did not find interactions of degree higher than 2. A summary of these results is displayed in Table 7.2.

Now we apply REV-ENG-M/REV-ENG-R to test the predictions using random variable orders.

7.1.3 REV-ENG-M/REV-ENG-R

To perform this analysis, we first computed 20 and subsequently 100 lexicographic term orders with random variable orders. The purpose in fixing the term order is the following: since we are mostly interested in reconstructing the wiring diagram for the Boolean model $\mathcal{N}$, choosing a lex ordering will favor the least number of variables in the normal forms of the interpolators. For each of the two experiments, we computed the matrix V, whose entries (i, j) indicate the number of times a variable x_j appeared in an interpolator f_i. Using row j of the matrix, we inferred a variable order for f_i. Using these orders, we defined a grevlex and a lex term ordering and computed local functions for the remaining variables $x_2, x_8, x_9, x_{10}, x_{11}, x_{14}$, and x_{15}. Altogether we computed local functions for the seven vari-

Table 7.2: Performance of dynamics detection for one cell of $\mathcal{N}$, as given in [38]. Single interactions = degree-one terms; cooperative interactions = degree-two terms. 4 TO denotes results for all 4 term orders used, whereas 3 TO denotes results for any 3 of the 4 term orders used.

Total single interactions in $\mathcal{N}$		13
Total cooperative interactions in $\mathcal{N}$		30
Single interactions	4 TO	3 TO
Total predicted	18	21
True positives	12	12
False positives	6	9
Cooperative interactions	4 TO	3 TO
Total predicted	3	11
True positives	3	8
False positives	0	3

ables using the four defined term orders, which we denote by lex20, lex100, grevlex20, and grevlex100. Since the dependency graph of a polynomial dynamical system is defined by the support of each local function, we report our findings in this context.

In all four orders, $supp(f_2) = supp(F_2) = \{x_1, x_2, x_{14}, x_{15}\}$. That is to say that the REV-ENG-M/REV-ENG-R method identified all 4 incoming edges for x_2. In five of the experiments, the edges adjacent to x_9 were also identified, namely $supp(f_9) = supp(F_9) = \{x_8, x_9, x_{18}, x_{19}\}$, and in the lex100 experiment, $supp(f_9) = \{x_8, x_9, x_{11}, x_{20}, x_{21}\}$. Further, in all experiments, $supp(f_{14}) = supp(f_{15})$. Since the support for these two variables is the same, we only consider one of them for the remaining analysis. Therefore, we focus our attention on the variables x_8, x_{10}, x_{11}, and x_{14}.

Using a grevlex order, as we increase the number of variable orders, the number of correct edges remains the same when compared to the PDS constructed from the wildtype and knockout time series under 1 grevlex order. However, the total number of edges predicted decreases as we add more orders. The advantage of incorporating multiple term orders seems to be that the number of false positives decreases, which supports the hypothesis that the use of multiple orders counteracts relations imposed by the order. We find similar results with the lex orders. Figure 7.3 illustrates the findings for the grevlex experiments.

To construct a dependency graph, we use a lexicographic ordering to minimize the number of edges in the graph. Below is a tabular representation of the dependency graph, in comparison to that of the Boolean model.

Other results, including the effect of noise on the general algorithm, can be found in [38].

Figure 7.3: Comparison of edges predicted using 1, 20, 100 grevlex orders. **a.** Total number of predicted edges. **b.** Number of correct edges.

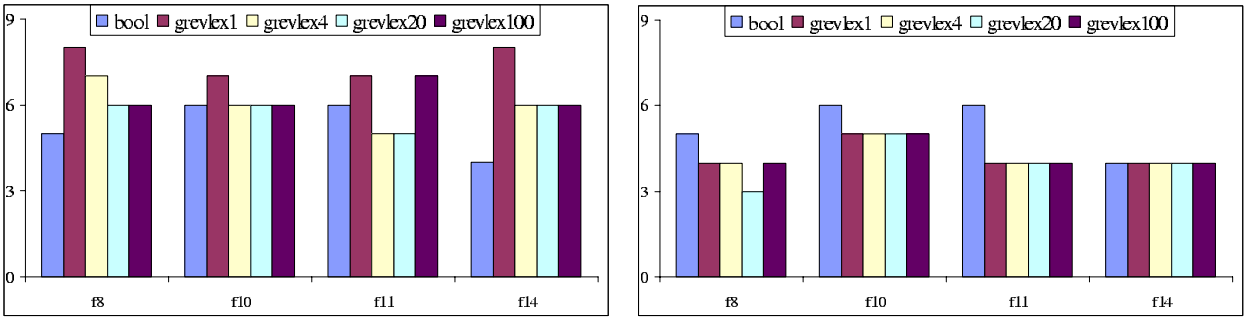
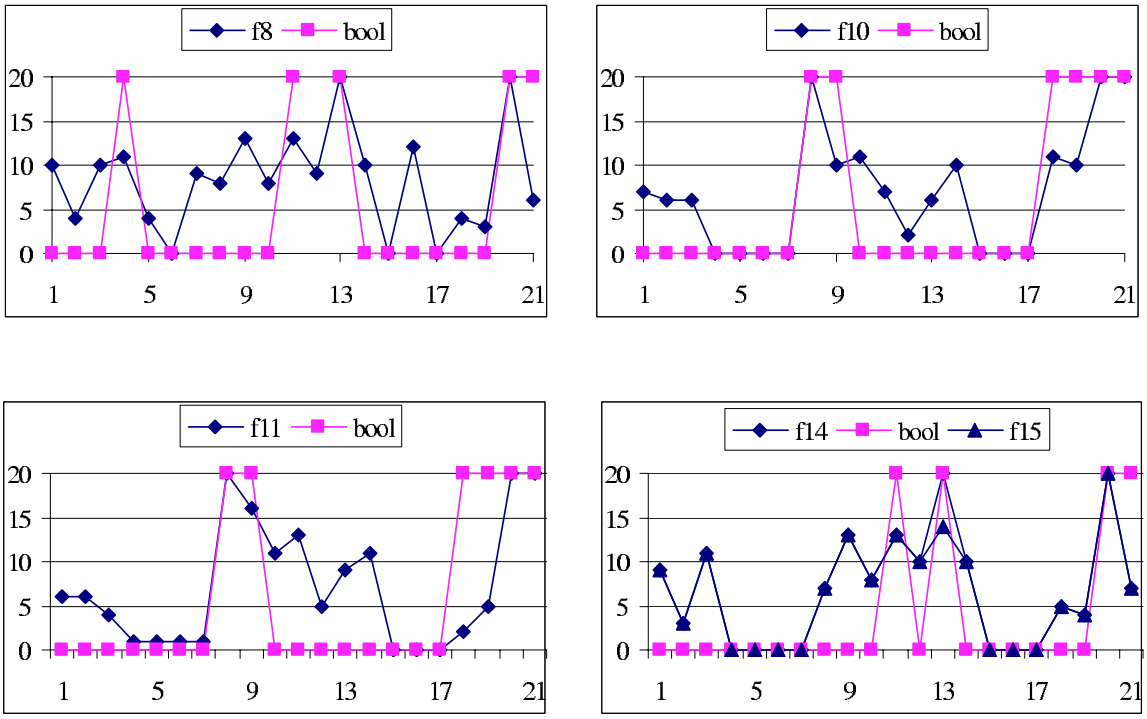


Table 7.3: Comparison of the wiring diagrams of the Boolean model and the reconstructed model, given in a tabular representation. A number in the table indicates the index of a variable. For example, line 6 should be interpreted as "function 6 in the Boolean model is in terms of x_5, x_{15}, and in the reverse-engineered model it is in terms of x_5." Indices in italics represent misidentified edges.

	Boolean model	lex20
f1	1	1
f2	1 2 14 15	1 2 14 15
f3	2	2
f4	1 16 17	16
f5	4	4
f6	5 15	5
f7	6	6
f8	4 11 13 20 21	*3* 11 13 *14* *16* 20
f9	8 9 18 19	8 9 18 19
f10	8 9 18 19 20 21	8 *10* 20 21
f11	8 9 18 19 20 21	8 9 *10* *11* *14* 20 21
f12	5	5
f13	12	12
f14	11 13 20 21	*3* 11 13 *14* 20
f15	11 13 20 21	*3* 11 13 *14* 20

Figure 7.4: Comparison of number of occurrences of variables in each function over 20 lex orderings. Variable indices are on horizontal axes and number of occurrences are on vertical axes.



Another method to reconstruct the wiring diagram is to use the matrices V for the 20- and 100-variable-order experiments with a lex ordering. If an entry (i, j) exceeds some threshold of the number of random orders, then we add the edge (x_j, x_i) to the dependency graph. If we use a threshold of 75%, then this criterion for selecting edges produces the following results. In both lex experiments, 18 of the 33 edges are correctly identified. Note that with this selection process, there are no false positives. Figure 7.4 shows these results.

7.2 Application to Simulated Data for Discovery

A simulated network of 20 genes, 23 proteins, and 16 metabolites was generated in *Gepasi* ([1]) by Pedro Mendes, director of the Biochemical Networks Modeling Group at the Virginia Bioinformatics Institute (VBI). This network mimics the response in yeast to oxidative stress. The network is described by a system of 100+ ordinary differential equations and wildtype time series of 50 time points was generated from solutions of the ODEs. Seven mutated networks were created from the original by setting to 0 the differential equation and initialization associated to one of the 20 genes; these networks simulate the network with a gene knockout. Similarly, time series of length 50 were generated from the mutant

networks. Each of the eight networks was subject to three perturbations of two metabolites. A partial wiring diagram for the network, which for historical reasons we call the Claytor network, is given in Figure 7.5. What is not included are all interactions of the type $g_i \to p_i$ which signify that a gene has a regulatory affect on its own protein.

In order to use the algebraic reverse-engineering algorithms proposed in this work, we first discretized the data using the method described in [21], which uses a graph- and information-theoretic technique to cluster data points. This method was chosen since it can discretize data to a prime number of states and is therefore appropriate for algebraic model over finite fields. The data were discretized to states in $\mathbb{F}_{11}$ and ten nodes were found to be constant: nodes 1, 3, 19, 23, 44, 45, 50, 57, 58, and 59. Therefore, we restrict our attention to the remaining 49 nodes.

We applied REV-ENG-M/REV-ENG-R to the wildtype and knockout time series corresponding to one perturbation condition: we did not use the other data since inconsistencies arose in the state transitions after discretization. We used 20 lexicographic orders with random variable orders; we did not use other term orders due to computational expense.

By studying the matrix V from the REV-ENG-R algorithm, we were able to detect the following edges by choosing the variables that appeared most frequently. For this discussion, the notation i (j) means that i is a label 1..20 for genes, 1..23 for proteins, and 1..24 for metabolites (some labels are in fact missing for metabolites) and j is a label 1..59 representing a variable. Below we summarize our results.

- We correctly identified nodes 6, 18, 21, and 36 as having no input.
- Genes 2, 4, and 20, as well as proteins 20 (40) and 22 (42), are regulated by gene 15 and protein 15 (35), all of which are consistent with the given wiring diagram.
- We find that genes 11, 12, 13, 14, 15, and 16 and proteins 2 (22), 4 (24), 5 (25), 11 (31), 14 (34), 15 (35), 17 (37), 21 (41), and 23 (43) regulate themselves; additionally genes 12 and 17 are regulated by their corresponding proteins.
- Proteins 6, 10, 12, 13, and 18 are regulated by their encoding genes. Protein 19 (39) is regulated by both itself and it corresponding gene.
- For gene 16, we also find that it is regulated by protein 15 (35), a more direct effect than what is given in the wiring diagram.
- Metabolites 11 and 12 affect each other, as is reflected in the cycle containing both.
- Metabolites 3, 7, 13, and 17 are affected by gene 11, which currently cannot be supported from the wiring diagram.

We do not have conclusive results for nodes 5, 7-10, 27-29, 47, 48, 51, and 55. Moreover, there are other links in the wiring diagram which cannot be supported or ruled out with the given data.

Figure 7.5: A partial wiring diagram for the Claytor network.

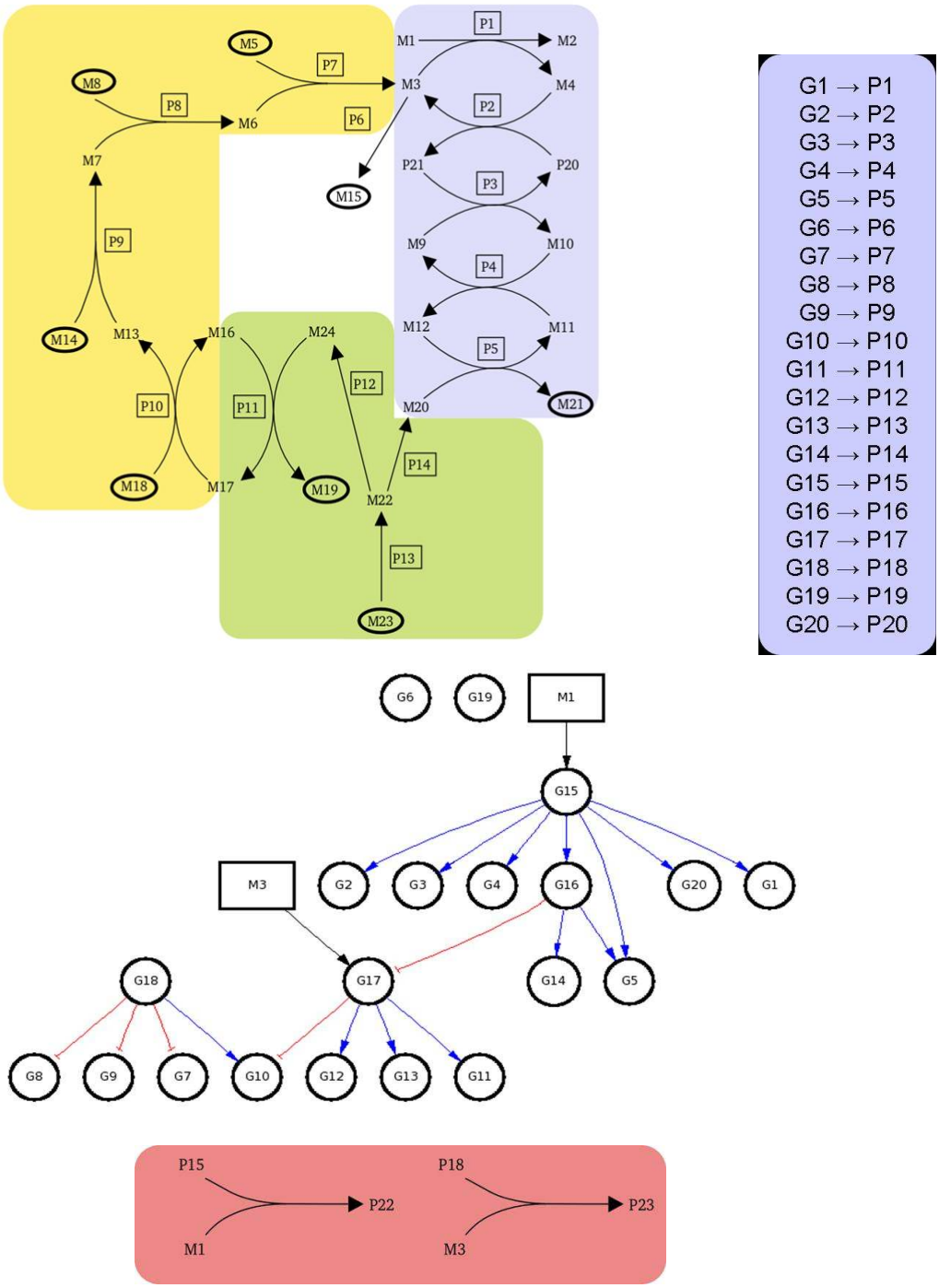


Table 7.4: Reverse-engineered wiring diagram for Claytor network with 20 random lexicographic orders. A line with an index in bold represents a variable that appeared in the interpolating polynomial for every variable order used.

f1	–	f31	**12 13 14 16** 22 **31 33**
f2	15 22 35	f32	12 **31**
f3	–	f33	13 **14 24**
f4	15 35	f34	**24** 34
f5	**24**	f35	**22** 35
f6	0	f36	0
f7	**32**	f37	13 22 31 37
f8	0	f38	0
f9	**22 24**	f39	19 39
f10	11 **12** 31	f40	15 35
f11	11	f41	**41 52**
f12	12 **31 32**	f42	15 35
f13	**13** 22 **31**	f43	22 43
f14	13 **14** 22 **25 27 31 52**	f44	–
f15	15 18 21	f45	–
f16	9 12 16 22 **31** 32 35	f46	11
f17	22 37	f47	0
f18	0	f48	0
f19	–	f49	11
f20	15 35	f50	–
f21	0	f51	**53**
f22	11 **12 13 14 22 24 25** 31 **35** 37 **52**	f52	13 **14 22 41 52 53**
f23	–	f53	**24 41 51 52 53**
f24	22 **24 31** 34 35	f54	11
f25	**13 22 25** 31 **33** 37	f55	0
f26	0	f56	22
f27	**31**	f57	–
f28	0	f58	–
f29	0	f59	–
f30	10 11 31		

7.3 Application to Real Data for Discovery

The techniques developed in this work are being applied to a project, directed by Reinhard Laubenbacher, Pedro Mendes, and Vladimir Shulaev at VBI, which involves computational and theoretical life scientists and mathematicians. This project, funded by an NSF/NIH joint initiative, is aimed at the design of a mathematical modeling framework for the investigation of oxidative stress response in the yeast *Sacchromyces cerevisiae*. Oxidative stress, which affects all aerobic organisms including humans, has been implicated in a number of human degenerative conditions, such as Parkinson's and cardiovascular diseases.

Laboratory experiments are being designed for the sole purpose of generating data for modeling, an uncommon paradigm in life science projects. The data being collected are time series of concentrations of mRNAs, proteins, and metabolites in wildtype yeast, as well as yeast mutants corresponding to single gene knockouts.

The central focus of the project is on the discovery of mathematical tools that are appropriate for the reverse-engineering of biochemical networks. Two modeling techniques will be used in this project: a continuous method, developed by P. Mendes' research group, where the models are linear systems of ordinary differential equations; and a discrete method, namely our reverse-engineering method based on PDSs. One goal is to develop modeling tools which can integrate diverse data types. A unique feature of the project is the development of an interface between the continuous and discrete models.

Chapter 8

Discussion and Future Work

In this dissertation, we presented a collection novel methods for reverse-engineering biochemical networks given discrete time series. The methods made use of algorithmic techniques in computational algebra, in particular Gröbner basis theory. A distinctive feature of our approach is the ability to construct all polynomial dynamical systems that interpolate the data.

To summarize, one PDS that fits the given time series is found. Then the set of local functions that vanish on the data is constructed by way of a Gröbner basis which does not require enumeration. A selection protocol based on minimality of the interpolating polynomials is employed, where the polynomials are minimal with respect to the Gröbner basis of the ideal of the given data points. The use of PDSs allows for each node of the system to be reverse-engineered individually, making the method appropriate for execution on distributed computer networks.

Because we use Gröbner bases to describe the set of all solutions, a term order must be fixed *a priori*. This feature allows for certain types of biological information to be built into the selection process. For example, information revealing the flow of interaction between two nodes can be incorporated into an ordering of the variables, which affects the reduction by the ideal of points. If gene A affects the regulation of gene B, then variable x_A representing A will be less than those variables not affecting B in the ordering and the reduced transition function for B will more likely be in terms of x_A than the nonaffecting variables. If there are further restrictions on the interactions, this information may be encoded in an ordering on the terms, such as an elimination ordering. In any case, a *minimal* PDS is chosen, in which each transition function contains no terms that vanish on all data points.

The use of Gröbner bases in polynomial interpolation has also been explored in algebraic statistics. In fact, the general algorithm outlined in Chapter 6 is very similar to that in [16]. In their article, the authors describe a theory in which algebraic geometry is applied to a range of problems in Design of Experiments, a branch of statistics. The novelty of our work

is the application of computational algebra and algebraic geometry to systems biology.

We demonstrated the effectiveness of inferring wiring diagrams for two different simulated networks. Incorporating multiple types of data, such as wildtype and knockout time series, dramatically improves the ability of the algorithms to detect correct edges. Using multiple variable orders minimizes the number of false positives. We also provided some theoretical results for determining when the reverse-engineering problem can be solved.

An aspect crucial to the performance of the reverse-engineering methods is the "goodness" of the data. Areas such as Design of Experiments and system identification in engineering have existing theoretical tools for analyzing data sets. For example, a theory of system identifiability, described in [42], outlines criteria for determining how effective a data set will be for the process of identification. Having a complete description of the input will only increase the applicability of our methods to biological systems by aiding the life scientist in designing experiments appropriate for modeling.

One goal for the future is to identify properties of data sets that make them suitable for algebraic reverse-engineering. Some questions to be addressed are the types of data that are appropriate for the methods, as well as the amount of data that is required for the methods to be effective. We will also look for improvements to the algorithms, such as decreased time complexity and scalability.

Finally we leave the reader with a look to the future from [31].

> A new generation of empiricists with stronger quantitative skills and of theoreticians with an appreciation for the empirical structure of biological processes will facilitate a bright future for the application of mathematics to solving biological problems.

Bibliography

[1] *GEPASI*, Available at `http://www.gepasi.org`.

[2] *ISCB enters new era*, Available at `http://www.iscb.org/history.shtml`, 2003.

[3] *GenBank Database*, Available at `http://www.psc.edu/general/software/packages/genbank/genbank.html`, 2005.

[4] J. Abbott, A. Bigatti, M. Kreuzer, and L. Robbiano, *Computing ideals of points*, JSYMC **30** (2000), no. 4, 341–356.

[5] A. Agrawal, *New institute to study systems biology*, Nature Biotechnology **17** (1999), no. 8, 743–744.

[6] T. Akutsu, S. Kuhara, O. Maruyama, and S. Miyano, *A system for identifying genetic networks from gene expression patterns produced by gene disruptions and overexpressions*, Genome Informatics (Tokyo) (K. Asai, S. Miyano, and T. Takagi, eds.), vol. 9, Universal Academy Press, 1998, same as ..Identification of genetic networks by strategic gene disruptions and gene orverepressions, Theoretical Computer Science 298 (2003) 235-251, pp. 151–160.

[7] T. Akutsu, S. Miyano, and S. Kuhara, *Identification of genetic networks from a small number of gene expression patterns under the boolean network model*, Proceedings of the Pacific Symposium on Biocomputing (Singapore) (A. Dunker L. Hunter R. Altman, K. Lauderdale and T. Klein, eds.), vol. 4, World Scientific Press, 1999, pp. 17–28.

[8] ———, *Inferring qualitative relations in genetic networks and metabolic pathways*, Bioinformatics **16** (2000), no. 8, 727–734.

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

[10] E. Allen, J. Fetrow, L. Daniel, S. Thomas, and D. John, *Algebraic dependency models of protein signal transduction networks from time-series data*, Journal of Theoretical Biology (2005), In press.

[11] R. Altman, *Challenges for intelligent systems in biology*, IEEE Intelligent Systems (2001), 14–18.

[12] K. Autumn, *Performance at low temperature and the evolution of nocturnality in lizards*, Integrative biology, Berkeley, University of California, 1995, 161 pages.

[13] D. Berman, *The number of generators of a colength n ideal in a power series ring*, Journal of Algebra **73** (1981), 156–166.

[14] B. Buchberger, *A note on the complexity of constructing groebner-bases*, Computer Algebra: Proceedings of EUROCAL 83 (J. A. von Hulzen, ed.), Lecture Notes in Computer Science, vol. 162, Springer, 1983, pp. 137–145.

[15] L. Buehler, *What is life?: A lifescience educational forum*, Available at `http://www.whatislife.com`, 2005.

[16] M. Caboara and L. Robbiano, *Families of estimable terms*, Proceedings of the 2001 International Symposium on Symbolic and Algebraic Computation (London, Ontario, Canada) (E. Kaltofen and G. Villard, eds.), ACM Press, 2001, pp. 56–63.

[17] T. Chen, H. He, and G. Church, *Modeling gene expression with differential equations*, Proceedings of the Pacific Symposium on Biocomputing (Singapore) (R. Altman, A. Dunker, L. Hunter, and T. Klein, eds.), vol. 4, World Scientific Press, 1999, pp. 29–40.

[18] D. Cox, J. Little, and D. O'Shea, *Ideals, varieties, and algorithms*, Springer Verlag, New York, 1997.

[19] H. de Jong, *Modeling and simulation of genetic regulatory systems: A literature review*, Journal of Computational Biology **9** (2002), 67–103.

[20] P. D'haeseleer, S. Liang, and R. Somogyi, *Genetic network inference: From co-expression clustering to reverse engineering*, Bioinformatics **16** (2000), no. 8, 707–726.

[21] E. Dimitrova, R. Laubenbacher, and J. McGee, *Discretization of time series data*, Submitted, 2005.

[22] E. Dimitrova, R. Laubenbacher, B. Stigler, and P. Vera Licona, *Polynome: Polynomial models of biological systems*, Available at `http://polymath.vbi.vt.edu/rev-eng/reveng.php`, 2005.

[23] D. Dummit and R. Foote, *Abstract algebra*, Prentice Hall, New Jersey, 1991.

[24] R. Edwards, *Analysis of continuous-time switching networks*, Physica D **146** (2000), 165–199.

[25] National Center for Biotechnology Information, *GEO: Gene expression omnibus*, Available at `http://www.ncbi.nlm.nih.gov/geo`, 2005.

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

[27] T. Gardner, D. di Bernardo, D. Lorenz, and J. Collins, *Inferring genetic networks and compound mode of action via expression profiling*, Science **301** (2003), 102–105.

[28] D. Grayson and M. Stillman, *Macaulay 2, a software system for research in algebraic geometry*, Available at `http://www.math.uiuc.edu/Macaulay2/`.

[29] A. Hartemink, D. Gifford, T. Jaakkola, and R. Young, *Using graphical models and genomic expression data to statistically validate models of genetic regulatory networks*, Proceedings of the Pacific Symposium on Biocomputing (Singapore) (R. Altman, A. Dunker, L. Hunter, and T. Klein, eds.), vol. 6, World Scientific Press, 2001, pp. 422–433.

[30] R. Hashimoto, S. Kim, I. Shmulevich, W. Zhang, M. Bittner, and E. Dougherty, *Growing genetic regulatory networks from seed genes*, Bioinformatics **20** (2004), no. 8, 1241–1247.

[31] A. Hastings and M. Palmer, *A bright future for biologists and mathematicians?*, Science **299** (2003), 2003–2004.

[32] Y. Huang, *Cellular regulatory network identification and metabolic engineering*, Available at `http://chem1.eng.wayne.edu/~yhuang/BioModeling.htm`, Research description.

[33] T. Ideker, T. Galitski, and L. Hood, *A new approach to decoding life: Systems biology*, Annual Review of Genomics and Human Genetics **2** (2001), 343–372.

[34] A. Jarrah, R. Laubenbacher, and H. Vastani, *DVD: Discrete visualizer of dynamics*, Available at `http://dvd.vbi.vt.edu`.

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

[36] B. Krupa, *On the number of experiments required to find the causal structure of complex systems*, Journal of Theoretical Biology **219** (2002), 257–267.

[37] R. Laubenbacher and B. Pareigis, *Decomposition and simulation of sequential dynamical systems*, Advances in Applied Mathematics **30** (2003), 655–678.

[38] R. Laubenbacher and B. Stigler, *A computational algebra approach to the reverse engineering of gene regulatory networks*, Journal of Theoretical Biology **229** (2004), 523–537.

[39] Y. Lazebnik, *Can a biologist fix a radio? Or, what I learned while studying apoptosis*, Cancer Cell **2** (2002), 179–182.

[40] S. Liang, S. Fuhrman, and R. Somogyi, *REVEAL, a general reverse engineering algorithm for inference of genetic network architectures*, Proceedings of the Pacific Symposium on Biocomputing (Singapore) (R. Altman, A. Dunker, L. Hunter, and T. Klein, eds.), vol. 3, World Scientific Press, 1998, pp. 18–29.

[41] R. Lidl and H. Niederreiter, *Finite fields*, 2nd ed., Encyclopedia of Mathematics and its Applications, vol. 20, Cambridge University Press, New York, 1997.

[42] L. Ljung, *System identification: Theory for the user*, 2nd ed., PTR Prentice Hall Information and System Science Series, Prentice Hall PTR, 1999.

[43] R. May, *Uses and abuses of mathematics in biology*, Science **303** (2004), 790–793.

[44] H. Möller and B. Buchberger, *The construction of multivariate polynomials with preassigned zeros*, Computer Algebra, Lecture Notes in Computer Science, vol. 144, Springer, Berlin, 1982, p. 2431.

[45] R. Morris, C. Bean, G. Farber, D. Gallahan, E. Jakobsson, Y. Liu, P. Lyster, G. Peng, F. Roberts, M. Twery, J. Whitmarsh, and K. Skinner, *Digital biology: An emerging and promising discipline*, TRENDS in Biotechnology **23** (2005), 113–117.

[46] Swiss Institute of Bioinformatics, *Swiss-prot protein knowledgebase*, Available at `http://us.expasy.org/sprot`, 2005.

[47] D. Pe'er, A. Regev, G. Elidan, and N. Friedman, *Inferring subnetworks from perturbed expression profiles*, Bioinformatics **17** (2001), S215–S224, Supplement 1.

[48] L. Robbiano, *Gröbner bases and statistics*, ch. Gröbner Bases and Applications, pp. 179–204, Cambridge University Press, New York, 1998.

[49] R. Shamir, I. Bogudlov, and V. Koushnir, *Genetic networks*, Available at `http://www.math.tau.ac.il/~rshamir/algmb/00/scribe00/html/lec14/`.

[50] M. Sherman, *Cell by cell: Moving biology toward a more predictive future*, Science & Technology Review, Lawrence Livermore National Laboratory, 2005.

[51] I. Shmulevich, E. Dougherty, S. Kim, and W. Zhang, *Probabilistic boolean networks: A rule-based uncertainty model for gene regulatory networks*, Bioinformatics **18** (2002), no. 2, 261–274.

[52] J. Tegnér, M. Yeung, J. Hasty, and J. Collins, *Reverse engineering gene networks: Integrating genetic perturbations with dynamical modeling*, Proceedings of the National Academy of Science of the United States of America **100** (2003), no. 10, 5944–5949.

[53] COMET Development Laboratory at University of Vermont, *Cell and molecular biology module: Prokaryotic genomics and proteomics*, Available at `http://cats.med.uvm.edu/cats_teachingmod/microbiology/courses/genomics/genomics_frameset.html`, Online teaching module offered through Department of Microbiology and Molecular Genetics.

[54] C. Tomlin and J. Axelrod, *Understanding biology by reverse engineering the control*, Proceedings of the National Academy of Science of the United States of America **102** (2005), no. 12, 4219–4220.

[55] L. von Bertalanffy, *General system theory*, 3rd ed., George Braziller, Inc., New York, 1968.

[56] G. von Dassow, E. Meir, E. Munro, and G. Odell, *The segment polarity network is a robust developmental module*, Nature **406** (2000), 188–192.

[57] J. Whitmarsh, *The need for mathematics in biomedical research*, 2005, Presentation in the Joint Mathematics Meetings of the AMS and the MAA, AMS Special Session on Mathematical Sciences Contributions to the Biomedical Sciences II.

[58] M. Yeung, J. Tegnér, and J. Collins, *Reverse engineering gene networks using singular value decomposition and robust regression*, Proceedings of the National Academy of Science of the United States of America **99** (2002), no. 9, 6163–6168.

Vita

A native of west Texas, Brandilyn Stigler grew up with second-generational roots in Mexico and India. The oldest of 6 children, she is the first in her family to earn a graduate degree. Following high school, she attended New Mexico State University, where she received a B.S. and an M.S. in mathematics, as well as minors in Spanish and computer science. She continued her education at Virginia Tech, earning a PhD in mathematics in 2005 for her interdisciplinary work at the Virginia Bioinformatics Institute. She is currently a Postdoctoral Fellow at the Mathematical Biosciences Institute at the Ohio State University. She is a member of American Mathematical Society, Association for Women in Mathematics, Society for Advancement of Chicanos and Native Americans in Science, and Society for Industrial and Applied Mathematics.