Proceedings of Symposia in Applied Mathematics

# Polynomial Dynamical Systems in Systems Biology

Brandilyn Stigler

*"Biology is moving from being a descriptive science to being a quantitative science."*
*John Whitmarsh, National Institutes of Health*
*2005 Joint Mathematics Meeting*

Abstract. Research in systems biology focuses on the prediction and control of the behavior of biochemical networks through the use of mathematical models that accurately capture the responses of the network to certain types of perturbations. The construction of models based on the observations of these responses, referred to as *reverse engineering*, is an important step in discovering the structure and dynamics of such networks. Recently polynomial dynamical systems over finite fields were introduced as a class of models for reverse engineering. These lecture notes, which are largely excerpts of the author's PhD thesis, are intended to provide the reader with key concepts related to reverse engineering using polynomial dynamical systems.

## 1. Introduction: What is Systems Biology?

The end of the twentieth century brought several technological breakthroughs in molecular biology, which led to the introduction of a new approach to the study of the molecular mechanisms that underlie the functioning of individual cells and whole organisms: systems biology. New sequencing technologies have shifted the focus from studying the DNA making up single genes to the determination of whole genomes. The field of bioinformatics, for instance, arose in part out of the need to store, search, and compare sequence data from an ever increasing number of organisms. DNA microarray chip technology has allowed quantitative measurements of the activity level of all genes in a cell extract, in response to an environmental perturbation. Likewise, it is now possible to make large-scale measurements of protein concentrations in cells. Innovative imaging technologies allow the determination of the spatial distribution of proteins in a single cell. The data that can be generated with all these technologies bring the goal of studying large-scale molecular networks

2000 *Mathematics Subject Classification.* Primary 13P10; Secondary 92C40.

*Key words and phrases.* Reverse engineering, gene regulatory networks, computational algebra, discrete modeling, polynomial dynamical systems.

This work was supported in part by NSF/NIGMS Grant #RO1 GM068947-01 and NSF Agreement #0112050.

©0000 (copyright holder)

within reach. Rather than studying the interactions of a handful of genes, we have data available to study whole gene regulatory networks. Instead of dividing cellular metabolism into individual pathways and focusing on the parts of the network, we can attempt to construct whole metabolic networks. The language, concepts, and tools of mathematics play a central role in this paradigm shift. A wide range of mathematical fields can and do contribute to the development of systems biology. In fact, the goal of constructing mathematical models of biological networks at the system level can be taken as a possible definition of the field of systems biology [**Kit02**].

There is a variety of modeling frameworks that are currently being exploited in systems biology. Ideker *et al.* [**IL03**] provide a categorization of modeling paradigms along a scale. On one end of the scale are abstracted or high-level models, such as statistical models, which encode the structure of a system by providing information about the components in the system and relationships between its components. These models are used to represent qualitative properties of a system, such as the existence of components that make up a pathway of interest and may also define correlations between components in a system. Examples of high-level models are Bayesian and neural networks [**dJ02, Edw00**].

At the other end, low-level models provide more detail, such as the mechanisms governing the relationships. These models are often represented as dynamical systems, which can capture quantitative features. Mechanistic properties of interactions among the components of the system can be encoded in models through the use of reaction rates and binding constants. In contrast to high-level models, low-level models are used to describe both the structure and dynamics of systems. Examples of low-level models include ordinary, partial, and delay differential equations [**IGH01, May04**].

These two classes represent the extremes of a spectrum of models, ranging in complexity from the abstracted to the specified. *Finite dynamical systems* (FDSs) are an intermediate class which provide a qualitative or coarse-grained view of both the structure and dynamics of a biological system. Examples of finite dynamical systems include Boolean, dynamic Bayesian, Glass, and logicical networks [**dJ02, HGJY01**]. For a more comprehensive review, see [**dJ02, DLS00, May04, Sti05a**].

Once we choose a modeling framework, there are various ways to construct models which can be broadly characterized as follows. One approach is to construct a mathematical model from a qualitative conceptual description based on extensive knowledge of the system [**DWFS99**]. With this prior knowledge, a model class is selected, such as the collection of SIR models which are used in mathematical epidemiology. 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 description) and designs a system (or a simulation of the system) [**IGH01**]. The challenge in this modeling approach is to identify model parameters.

A second approach is to build a mathematical model from observations of the system in response to well-constructed perturbations. In molecular biology the observations are often measurements of concentrations of biochemicals, recorded at pre-determined time intervals, resulting in time series of experimental data (see,

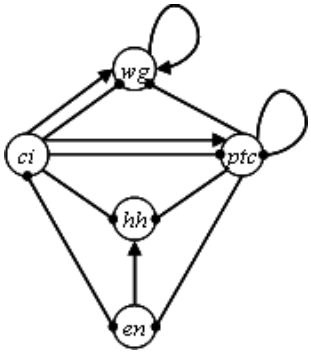


FIGURE 1. The wiring diagram of a network of 5 genes in the fruit fly *D. melanogaster*, as seen in Tegnér *et al.* [**TYHC03**]. Edges with arrows := activation; edges with circles := inhibition.

e.g., [**YTC02**]). A model is built from the observations and is adjusted to fit the observations. As with the previous approach, 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 [**DWFS99, IGH01**]. The challenge in this paradigm is the development of algorithmic tools for constructing predictive models, including model structure and parameters, from experimental data.

In the context of dynamical systems, reverse engineering is the process of discovering the dynamic behavior and connectivity structure of a system, given observations of the system [**DLS00, GdBLC03, TYHC03, YTC02**]. Local 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 (see [**vB68**], p.55).

The dependency structure can be depicted in a *wiring diagram*. These diagrams are typically directed graphs, such as Figure 1 from [**TYHC03**], where vertices represent system variables, in this case biochemicals, and edges indicate the direction and perhaps the type of interaction between the biochemicals. As observations from an experiment are recorded at fixed times, the dynamics of a biochemical network is manifested as a *time series* of values representing fluctuations in the concentration of the biochemicals in the system. The dynamics can also be represented graphically, via a *state space graph* (see Section 2 for definitions).

Observations of biological systems at the molecular level are increasingly abundant [**DWFS99, IGH01**]. A key goal of systems biology is to gain insight into the structure and dynamics of systems, so there is a growing need for reverse-engineering methods, which by their nature translate observations into predictive models. While there are algorithmic approaches for constructing low- and high-level models (for example, [**FLNP00, PREF01, TYHC03, YTC02**]), there are surprisingly few for FDSs. Those that exist for Boolean networks are based on

enumeration of large numbers of models with constraints (see [**LFS98**]). An algorithmic method for constructing a special class of FDSs, called *polynomial dynamical systems*, has been proposed in [**LS04**] and [**Sti05a**], which we present. Specifically, we present a collection of methods to construct polynomial dynamical systems (PDSs) from time series of discrete data, using tools and concepts from computational algebra.

Next we define polynomial dynamical systems. Section 3 is devoted to the algorithms and Section 4 contains results from application of the algorithms to simulated data. We close with a discussion of future directions and summary of resources.

## 2. Finite Dynamical Systems

DEFINITION 2.1. Let $X$ be a finite set, called a *state set* and its elements *states*. A *finite dynamical system* (FDS) on $n$ nodes is a function $F = (f_1, \ldots, f_n) : X^n \to X^n$ with each $f_i : X^n \to X$ called the *transition function* associated to node $i$. An FDS on $n$ nodes is called *$n$-dimensional*.

Function evaluation is defined as $F(\mathbf{s}) := (f_1(\mathbf{s}), \ldots, f_n(\mathbf{s}))$ for every $\mathbf{s} \in X^n$. Unless otherwise noted, all FDSs are $n$-dimensional, for a fixed integer $n > 0$.

An FDS $F = (f_1, \ldots, f_n)$ can be defined over a Cartesian product of state sets $X_1 \times \cdots \times X_n$, where each transition function $f_i$ takes values from $X_i$. Since we are interested in the application of FDSs to molecular biology, we make the assumption that all biochemicals take values from the same state set.

Let $\mathbf{s} = (s_1, \ldots, s_n), \mathbf{t} = (t_1, \ldots, t_n) \in X^n$. We say that the pair $(\mathbf{s}, \mathbf{t})$ is a *state transition* of an FDS $F : X^n \to X^n$ if $F(\mathbf{s}) = \mathbf{t}$. Further, we write $\mathbf{s} \mapsto \mathbf{t}$. A sequence $\mathbf{s}_1 \mapsto \cdots \mapsto \mathbf{s}_m$ of state transitions for $m \geq 1$ is called a *trajectory*. If $\mathbf{s}_1 = \mathbf{s}_m$, then the trajectory is a *limit cycle of length $m$*. When $m = 1$, the limit cycle is called a *fixed point*.

By their construction, FDSs have an ordered dynamic structure in that *all* trajectories end in limit cycles. This is due to the fact that the transition functions are defined over finite state sets and function evaluation is a deterministic process. On the other hand, FDSs in general admit no *algebraic* structure as they are defined by arbitrary set functions. This lack of structure renders the algorithmic construction of such functions computationally intractable for moderately sized system. However, constraining the state set will allow us to impose algebraic structure on FDSs.

**2.1. Polynomial Dynamical Systems.** Let $|X|$ denote the cardinality of a state set $X$. If $|X|$ is (a power of) a prime, then we can view $X$ as a finite field with usual modular arithmetic. We let $k$ denote a state set satisfying the primality condition to distinguish it as a finite field. Primality allows us to exploit the following theorem, which characterizes functions defined over finite fields [**LN97**].

THEOREM 2.2 (Generalized Lagrange Interpolation). *Let $k$ be a finite field. Then every function $f : k^n \to k$ can be represented as a polynomial of degree at most $n$.*

In fact we can think of each transition function of an FDS as an element of a polynomial ring $k[x_1, \ldots, x_n]$.

DEFINITION 2.3. Let $k$ be a finite field. An FDS $F = (f_1, \ldots, f_n) : k^n \to k^n$ over $k$ is a *polynomial dynamical system* (PDS).

EXAMPLE 2.4. Consider a 3-dimensional PDS $F = (f_1, f_2, f_3) : \mathbb{F}_5 \to \mathbb{F}_5$ given by

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

where $\mathbb{F}_5 (\cong \mathbb{Z}/5\mathbb{Z})$ is the field on 5 elements. Then $F$ evaluated at the state $(1, 1, 1)$ is calculated as follows:

$$F(1, 1, 1) = (f_1(1, 1, 1), f_2(1, 1, 1), f_3(1, 1, 1)) = (4, 4, 1),$$

giving the state transition $(1, 1, 1) \mapsto (4, 4, 1)$.

EXAMPLE 2.5. Let $p$ be any prime integer. The formal expressions $x$ and $x^p$ are distinct polynomials in the ring $R = \mathbb{F}_p[x]$; however, $x = x^p$ as functions under mod-$p$ arithmetic. Therefore, we can view the expressions as elements in quotient ring $R/\langle x^p - x\rangle$.

In this setting, state transitions are computed synchronously. 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 dynamical 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 [**LP03**].

**2.2. Properties of PDSs.** The two key features of biological systems that are of interest are the connectivity structure, typically represented as a wiring diagram (see Figure 1 for an example), and the dynamics, that is, the behavior of the system. These features are encoded in a PDS $F$: the structure is given by a graph constructed from the form of the transition functions, and the dynamics by a graph arising from iteration of $F$.

DEFINITION 2.6. Let $f \in k[x_1, \ldots, x_n]$. The *support of* $f$, denoted by $supp(f)$, is the smallest subset $\{x_{i_1}, \ldots, x_{i_m}\}$ of $\{x_1, \ldots, x_n\}$ such that $f \in k[x_{i_1}, \ldots, x_{i_m}]$.

EXAMPLE 2.7. 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 $supp(f) = \emptyset$.

DEFINITION 2.8. Let $F$ be an $n$-dimensional PDS. The *dependency graph* of $F$, denoted by $D(F)$ 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 2.9. Let $k = \mathbb{F}_3$ and $F = (f_1, f_2, f_3) : k^3 \to k^3$ be the 3-dimensional PDS with transition functions

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

The dependency graph of $F$ is given in Figure 2.

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 viewed as the dependency graph for a PDS.

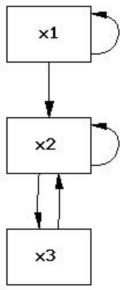


FIGURE 2. The dependency graph for the PDS in Example 2.9.

THEOREM 2.10. *Let $\mathfrak{D}$ be the mapping from the set of $n$-dimensional PDSs to the set of directed graphs on $n$ vertices that sends a PDS $F$ to its dependency graph $D(F)$. Then $\mathfrak{D}$ is a surjective mapping.*

PROOF. Let $(V, E)$ 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 $D(F) = (V, E)$. □

Notice that the construction of $F$ is not unique in that we could have defined $f_i$ as any polynomial function in terms of the variables $x_{v_1}, \ldots, x_{v_m}$. Hence, $\mathfrak{D}$ is not injective.

Iteration of a PDS produces its *dynamics*, as we define below. Because the state set of a PDS is finite, we can represent the dynamics by a finite graph with $|k|^n$ vertices.

DEFINITION 2.11. Let $F$ be an $n$-dimensional PDS. The *state space graph* of $F$, denoted by $S(F)$, is a directed graph $(V, E)$ with $V := k^n$ and $E := \{(a, b) : a, b \in V \text{ and } F(a) = b\}$.

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 [**JLV**].

EXAMPLE 2.12. The state space for the PDS in Example 2.9 is given in Figure 3 and has two fixed points, two limit cycles of length 2, and one limit cycle of length 3.

EXAMPLE 2.13. While the PDSs in Examples 2.4 and 2.9 have the same dependency graphs, their state spaces differ: their cardinalities are $5^3$ and $3^3$, respectively.

The set $\mathcal{P}_k$ of PDSs over a finite field $k$ gives rise to a second class of directed graphs. Let $\mathcal{G}$ be the set of directed graphs on $|k|^n$ vertices such that the out-degree of each vertex is exactly 1. Define $\mathfrak{S} : \mathcal{P}_k \to \mathcal{G}$ by $F \mapsto S(F)$.

THEOREM 2.14. *Then mapping $\mathfrak{S}$ 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 $S(F_1) \neq S(F_2)$ since $(a, F_1(a)) \neq (a, F_2(a))$. So $\mathfrak{S}$ is one-to-one.

Let $(V, E) \in \mathcal{G}$ 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

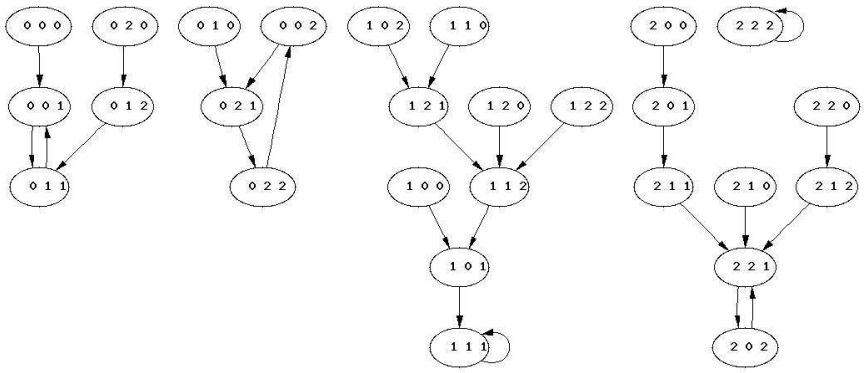


FIGURE 3. The state space for the PDS in Example 2.9.

as $f(a) = \pi_i(b)$ for each $(a, b) \in E$. By Theorem 2.2, $f_i$ is a polynomial. Hence $F = (f_1, \ldots, f_n)$ is a PDS with state space $(V, E)$ and $\mathfrak{S}$ is onto, thus concluding the proof. □

In summary an $n$-dimensional PDS is a function $F = (f_1, \ldots, f_n) : k^n \to k^n$ with dependency graph $D(F)$ and state space $S(F)$. While the state space associated to a PDS $F$ uniquely determines $F$, its dependency graph does not. In certain settings, more is known about the structure of polynomial dynamical systems, particularly for Boolean [**GJL01**], linear [**JLVL06, HT05**], and monomial dynamical systems [**CRLP05, CRJLS07**].

## 3. Reverse Engineering using PDSs

As mentioned in the introduction, reverse engineering is the process of inferring network structure and dynamics from observations. In molecular biology, observations are often transition pairs $(\mathbf{s}_i, \mathbf{t}_i) \in \mathbb{R}^n \times \mathbb{R}^n$ corresponding to the state of a biochemical network before and after a perturbation. The number of such transition pairs tends to be small (on the order of tens) due to the high cost in conducting biological experiments. Whenever $\mathbf{s}_{i+1} = \mathbf{t}_i$ for $(\mathbf{s}_i, \mathbf{t}_i)$ and $(\mathbf{s}_{i+1}, \mathbf{t}_{i+1})$, then the collection of contiguous pairs is considered to be a *time series*.

**3.1. Data Discretization.** In order to use PDSs to model biochemical networks, the data must be discrete. Since experimental data are typically real-valued, we must first preprocess the data by discretizing them; that is, we must map them to elements of a finite field. One approach is to choose a small number of thresholds and cluster the experimental values according to the thresholds, as is done in [**AFD$^+$06**]. However, this requires knowing the number of thresholds to use and explicitly defining the thresholds. Another approach is to use statistical techniques to calculate features of the data, such as its central tendency and variance. However, experimental data sets tend to have few data points and rarely more than 3 replicates.

Given these constraints, we have chosen a method which works well with few data points and does not require replicates or user-defined thresholds. This method developed by Dimitrova *et al.* [**DLM05, Dim06**] uses unsupervised single-linkage clustering which balances a small number of discrete states to reduce computational

complexity and having enough discrete states to maintain the information content in the data. We provide an example in Section 3.4.

**3.2. Solving the Reverse-engineering Problem.** Let $k$ be a finite field and $D = \{(\mathbf{s}_1, \mathbf{t}_1), \ldots, (\mathbf{s}_m, \mathbf{t}_m) \mid \mathbf{s}_i, \mathbf{t}_i \in k^n, 1 \leq i \leq m\}$ be a data set for a biochemical network on $n$ nodes.

PROBLEM 3.1 (Reverse Engineering). Find a PDS $F : k^n \to k^n$ such that

(1) (*Data Fitting*) $F(\mathbf{s}_i) = \mathbf{t}_i$ for $1 \leq i \leq m$.
(2) (*Structure*) $F$ is consistent with known network topology.
(3) (*Dynamics*) $F$ is consistent with known dynamic behavior (*e.g.*, limit cycles).

Since a PDS $F = (f_1, \ldots, f_n)$ consists of coordinate functions, to solve (1) of Problem 3.1 we can instead consider the problem of reverse engineering the transition functions $f_j$ simultaneously. That is, we aim to find functions $f_1, \ldots, f_n : k^n \to k$ such that $f_j(\mathbf{s}_i) = t_{ij}$ for all $1 \leq i \leq m$ and $1 \leq j \leq n$, and where each $\mathbf{t}_i = (t_{i1}, \ldots, t_{in})$. We fix a coordinate $j$ and let $D_j = \{(\mathbf{s}_1, t_1), \ldots, (\mathbf{s}_m, t_m) \mid \mathbf{s}_i \in k^n, t_i \in k, 1 \leq i \leq m\}$ denote the data for node $j$.

Suppose there are two such functions $f, g : k^n \to k$ such that $f(\mathbf{s}_i) = t_i = g(\mathbf{s}_i)$ for all $i$. Then $(f-g)(\mathbf{s}_i) = 0$ for all $i$ and we can write $g$ as the function $f-(f-g)$ on the given data $D_j$. So the problem of finding interpolating functions is reduced to finding one, say $f$; any other can be written as a sum of $f$ and some function which vanishes on $\mathbf{s}_1, \ldots, \mathbf{s}_m$. We call a function $g$ with the property that $g(\mathbf{s}_i) = 0$ for all $i$ a *vanishing function*.

Next we show that the reverse-engineering problem can be expressed in the language of computational algebra, which provides a convenient framework for finding PDSs.

**3.3. Connections to Computational Algebra.** All terms in italics and well-known results in this subsection can be found in most introductory commutative algebra, computational algebra or algebraic geometry textbooks. However, we refer the reader to [**CLO97**] and [**DF91**] for the specific definitions and statements of theorems used below. More definitions and results are provided in Appendix B.

Let $k$ be a field. Any finite collection of points in $k^n$ is an *affine variety* and the set of polynomials in $R = k[x_1, \ldots, x_n]$ that vanish on the points has the algebraic structure of an *ideal*. We let $V = \{\mathbf{s}_1, \ldots, \mathbf{s}_m\}$ be the set of *input points*[1] and $I(\mathbf{s}_i) = \{f \in R : f(\mathbf{s}_i) = 0\}$, the set of *vanishing functions*[1] for $\mathbf{s}_i$. In the literature, $I(V)$ is often called the *ideal of the points in* $V$.

Now we state the following well-known theorem, which will be useful for solving the interpolation problem (1).

THEOREM 3.2 (Chinese Remainder Theorem). *Let $R$ be a ring and $I_1, \ldots, I_m$ be pairwise comaximal ideals of $R$. Let $\phi : R \to R/I_1 \times \cdots \times R/I_m$ be the homomorphism*

$$f \mapsto (f + I_1, \ldots, f + I_m).$$

*Then $\phi$ is surjective with kernel $\bigcap_i I_i = I_1 \cdots I_m$.*

[1]These terms are defined by the author.

In essence, the Chinese Remainder Theorem provides the existence of a solution to (1), which is unique up to the intersection of the ideals of the points in $V$, that is, the set of vanishing functions for $V$. It is known that the ideals $I(\mathbf{s}_i)$ are generated by the binomials $x_1 - s_{i1}, \ldots, x_n - s_{in}$ and so are *maximal*. Since $\bigcap_i I(\mathbf{s}_i) = I(\mathbf{s}_1) \cdots I(\mathbf{s}_m)$, then the intersection ideal is generated by all pairwise products of the binomials. However, for the purpose of reverse engineering, the choice of representative is essential, as we will see in the next section. Special generating sets, called *Gröbner bases*, provide a natural representative of $\bigcap_i I(\mathbf{s}_i)$.

Let $R = k[x_1, \ldots, x_n]$ and $\mathbf{x}^a$ denote a monomial $x_1^{a_1} \cdots x_n^{a_n} \in R$.

DEFINITION 3.3. A *term order* on $R$ is a relation $>$ on the set of monomials $\mathbf{x}^a$ such that $>$ is a total ordering and a well-ordering.

We call the initial ordering of the variables in the declaration of a term order a *variable order*.

Given a term order $>$, we can uniquely identify the largest term of a polynomial $f$ according to $>$, called the *leading term* of $f$ and denoted by $LT(f)$.

DEFINITION 3.4. Let $>$ be a term order and $I \subset R$ an ideal. A finite subset $G = \{g_1, \ldots, g_r\} \subset I$ is a *Gröbner basis* for $I$ if for any $f \in I$ there is $g_i \in G$ such that $LT(g_i)$ divides $LT(f)$ under $>$. $G$ is *reduced* if every $g \in G$ is monic and no term of $g_i$ is divisible by any $LT(g_j)$ for $1 \leq i \neq j \leq r$.

THEOREM 3.5. *Let $>$ be a term order on $R$. Every nonzero ideal $I \subset R$ has a Gröbner basis $G$. Moreover $G$ is a generating set for $I$. If $G$ is reduced, then it is unique.*

REMARK 3.6. In the remaining discussion, we will consider all Gröbner bases to be reduced, unless otherwise stated.

While most elementary operations on polynomials can be performed independent of the term order, it is not the case with multivariate polynomial division. Dividing a polynomial by a set of polynomials may lead to different remainders, depending on the order in which division was executed. However, when dividing by a Gröbner basis the remainder is always unique.

DEFINITION 3.7. Let $G$ be a Gröbner basis for an ideal $I \subset R$ and let $f \in R$. The *normal form* of $f$ with respect to $G$, denoted by $NF(f, G)$, is the remainder of $f$ on division by the elements of $G$. If $f = NF(f, G)$, then $f$ is *reduced* with respect to $G$.

THEOREM 3.8. *Let $G$ be a Gröbner basis for $I \subset R$ and let $f \in R$. Then $NF(f, G)$ is unique.*

Given an ideal $I$ and the Gröbner basis $G$ for $I$ with respect to a term order $>$, the set $\{\mathbf{x}^a \in R : \mathbf{x}^a \notin \langle LT(G) \rangle\}$ forms a basis for $R/I$ and is called the set of *standard monomials*; let $SM(G)$ denote this set. The normal form $NF(f, G)$ for any polynomial $f \in R$ is well characterized in that it can be written in terms of the standard monomials. Furthermore polynomials expressed as a linear combination of standard monomials are reduced with respect to $G$. See Appendix B for other properties of Gröbner bases.

**3.4. The REV-ENG Algorithm.** The algorithm presented in [**LS04**] and [**Sti05a**] constructs all PDSs for a given data set and then uses the techniques described above to select a "minimal" PDS from the set. The set of PDSs is not constructed via enumeration; instead this is accomplished by way of a Gröbner basis. It is a discrete analog of the ODE method presented in [**YTC02**], in which a particular solution $p$ to the problem is obtained, the family $H$ of homogeneous solutions is constructed, and any solution to the problem is of the form $p + h$, for some $h \in H$. We will see that each transition function of a given network may be reverse engineered individually.

Consider a data set

$$D = \{(\mathbf{s}_1, \mathbf{t}_1), \ldots, (\mathbf{s}_m, \mathbf{t}_m) \mid \mathbf{s}_i, \mathbf{t}_i \in k^n, 1 \leq i \leq m\}.$$

Let $V$ be the set of input points of $D$, which we will view as an affine variety in $k^n$. Fix a term order $>$ on $R$. We first construct the ideal $I = I(V)$ of polynomials that vanish on $V$ and its Gröbner basis $G$ with respect to $>$. If we have an interpolator $f_j$ for each node $j$, then the set

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

represents all PDSs that fit $D$, where $F = (f_1, \ldots, f_n)$. The task is then to select a minimal polynomial dynamical system from this set, which we discuss later in the section.

Let $D_j = \{(\mathbf{s}_1, t_1), \ldots, (\mathbf{s}_m, t_m) \mid \mathbf{s}_i, \in k^n, t_i \in k, 1 \leq i \leq m\}$ be the data set for node $j$. Given $\mathbf{t} = (t_1 + I(\mathbf{s}_1), \ldots, t_m + I(\mathbf{s}_m))$, where $\mathbf{s}_i \mapsto t_i$, Theorem 3.2 tells us that there exists $f \in R$ such that $\phi(f) = \mathbf{t}$. In fact, the proof of the theorem (see [**CLO97**]) constructs such a function

$$f(x_1, \ldots, x_n) = \sum_{i=1}^{m} t_i r_i(x_1, \ldots, x_n)$$

with each polynomial $r_i(x_1, \ldots, x_n)$ 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 $\mathbf{s}_j$. The polynomials $r_i$ are called *separators* of the points in $V$. Moreover the normal form $NF(f, G)$ has the property that it contains no terms which vanish on the input points. From the point of view of applications, this is desirable since vanishing terms correspond to unobservable phenomena. If we set $f_j = NF(f, G)$, then $(f_1, \ldots, f_n)$ is a *minimal* PDS which fits the given data $D$, where by minimal we mean that each $f_j$ is reduced with respect to $G$. All other PDSs differ by elements of $\bigcap_i I(\mathbf{s}_i)$.

In general, computing Gröbner bases of ideals is doubly exponential in the number of indeterminates. However, in our setting we aim to compute the Gröbner basis of a *zero-dimensional* ideal (an ideal of points) for which a polynomial-time algorithm exists, namely the Buchberger-Möller algorithm (BMA). Given a variety $V$ and a term order $>$, the BMA computes the reduced Gröbner basis $G$ of the ideal $I(V)$, the associated standard monomials, and the separators, reduced with respect to $G$, of the points in $V$.

Let BM-SEP be the subroutine of the BMA which computes the reduced separators of the input data points. In Table 1 is pseudo-code for the REV-ENG algorithm given a data set over a finite field. The algorithm was first described in [**LS04**] and extensions have been proposed [**AFD$^+$06, JLSS06, Sti05a**], some of which we introduce below. All of the presented algorithms have been implemented

TABLE 1. REV-ENG

**Input**: $D = \{(\mathbf{s}_1, \mathbf{t}_1), \ldots, (\mathbf{s}_m, \mathbf{t}_m)\}$ data set, *t-order* = term order
**Output**: $F$ minimal PDS for $D$

**Algorithm**:
DataPoints := $\{\mathbf{s}_1, \ldots, \mathbf{s}_m\}$
Sep := BM-SEP(DataPoints, *t-order*)
for each $i = 1 \ldots n$
Values := $\{\mathbf{t}_1, \ldots, \mathbf{t}_m\}$
$f_i := \sum_{j=1}^{t-1} \text{Values}_j \text{Sep}_j$
endfor
**return** Minimal PDS $F = (f_1, ..f_n)$

in the computational algebra software package *Macaulay 2* [**GS**]; Polynome, a web version, is available at [**Sti05b**].

THEOREM 3.9. *Given a data set $D$ over a finite field, REV-ENG returns a minimal PDS that fits $D$. Moreover its worst-time complexity is that of the BM algorithm.*

For a proof, see [**Sti05a**].

We illustrate the algorithm with a small artificial 3-gene network with states in $\mathbb{R}$. Consider the following time series of state transitions for the genes, labeled $x$, $y$, and $z$.

| *Time* | $x$ | $y$ | $z$ |
|---|---|---|---|
| 1 | 0.9 | 0.92 | 0.89 |
| 2 | 0.53 | 0.26 | 0.93 |
| 3 | 0.47 | 0.27 | 0.26 |
| 4 | 0.27 | 0.39 | 0.41 |
| 5 | 0.24 | 0.37 | 0.42 |

After discretization, we get states in $\mathbb{F}_3$, where the elements of $\mathbb{F}_3$ can be interpreted as the states *above average gene expression* (2), *average expression* (1), and *below average expression* (0).

| *Time* | $x$ | $y$ | $z$ |
|---|---|---|---|
| 1 | 2 | 2 | 2 |
| 2 | 1 | 0 | 2 |
| 3 | 1 | 0 | 0 |
| 4 | 0 | 1 | 1 |
| 5 | 0 | 1 | 1 |

The data set consists of transition pairs

$$\{((2,2,2),(1,0,2)),$$
$$((1,0,2),(1,0,0)),$$
$$((1,0,0),(0,1,1)),$$
$$((0,1,1),(0,1,1))\}$$

with input data points

$$V = \{(2,2,2),(1,0,2),(1,0,0),(0,1,1)\}.$$

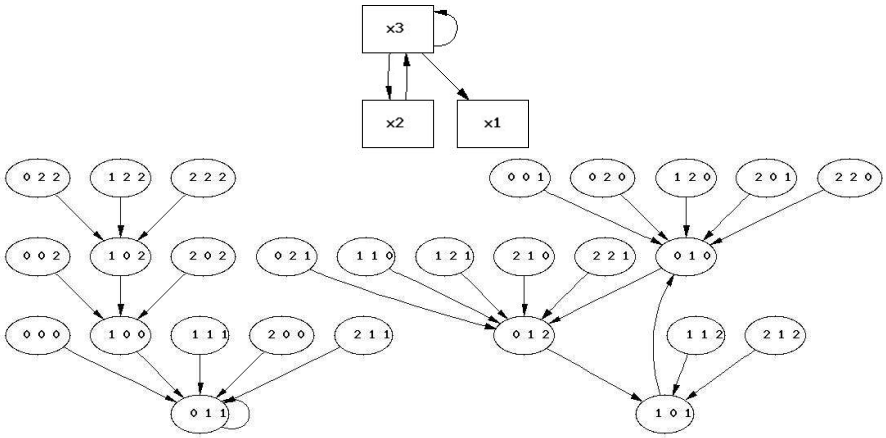


FIGURE 4. The dependency graph and state space for the 3-dimensional PDS $F$. Here x1 := $x$, x2 := $y$, and x3 := $z$.

Assume that all computations will be performed under the graded reverse lexicographic (grevlex) ordering with $x > y > z$; see Appendix B for a definition. The algorithm computes the following separators of the points in $V$:

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

One can verify that these polynomials are in fact separators of the points in $V$.[1] Consider the set of output values $\{1, 1, 0, 0\}$ for $x$. Then we write

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

Computing the other two polynomials in this way, the algorithm returns the PDS $F = (f_1, f_2, f_3)$ where

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

While REV-ENG does not output the Gröbner basis of the ideal of the points in $V$, we include it here for the purpose of illustration.

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

Notice that adding a polynomial multiple of any element of $G$ to one of the $f_i$ produces another interpolating function since every element of $G$ evaluates to 0 on the points in $V$.

We see that $F$ satisfies (1) of Problem 3.1 and has the dependency graph and state space as in Figure 4. If Condition (2) is not satisfied, for example if we believe

[1]For instance, one can check that $r_2(1, 0, 2) = 1$ and $r_2(\mathbf{s}) = 0$ for all other points $\mathbf{s}$ in $V$.

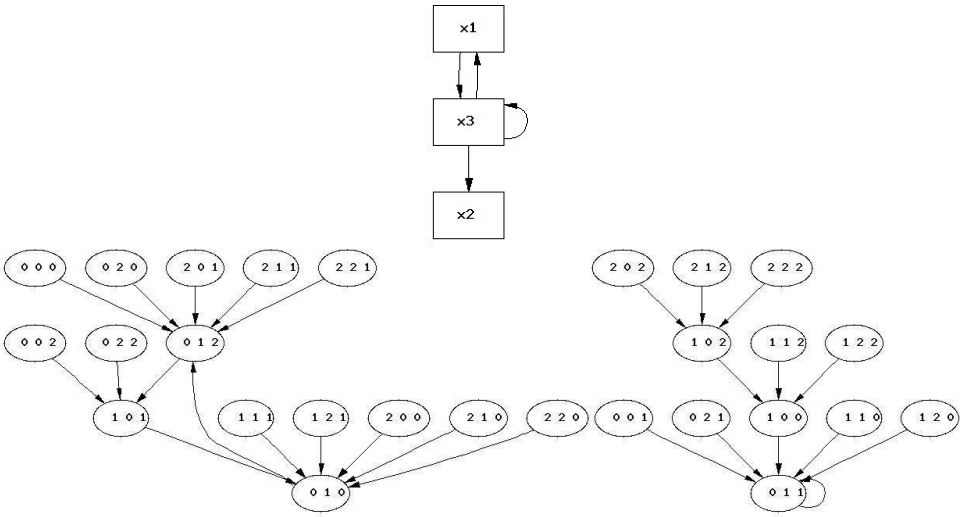


FIGURE 5. The dependency graph and state space for the PDS $F' = (f_1, f_2, f_3')$.

that $x$ is an input for $z$ instead of $y$, then we can add $2(x + y + 2)$ to $f_3$ to get

$$f_3'(x, y, z) = 2z^2 + 2x + 2.$$

Now we get the dependency graph and state space for the new PDS $F'$ as in Figure 5. Notice that the state spaces of $F$ and $F'$ coincide on the trajectory starting at (2,2,2), that is on the given data, but differ on the most of the other states. If Condition (3) is not satisfied, we can continue to modify the PDS until we get the desired dynamics.

**3.5. Extensions of REV-ENG.** The algorithm REV-ENG is designed to compute a minimal PDS given a set of state transitions, where the minimality condition is subject to a term order. However, this dependence is unfavorable because an artificial relationship is imposed on the nodes of a system by way of the variable order. The normal form of a polynomial can be different for different choices of a term order (see Appendix B for examples). 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 subsection we describe three extensions to the general algorithm, two which address the term-order dependency and one which incorporates multiple time series.

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 existence 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 the tails of the edges incident to $x$ are ordered least and the rest of the variables are ordered greatest. Then a lexicographic or an elimination ordering can be used (see Appendix B for definitions). Computations

using a lexicographic (lex) order will result in normal forms that are in terms of the variables ordered least and potentially of high degrees (see Example B.4).

The algorithm REV-ENG-D, presented in [**Sti05a**], makes use of this capability. It takes as input one data set and a set of $n$ variable orders, one for each node. It computes the interpolators with respect to the given term order and $i$-th variable order. For a description of the algorithm, see [**Sti05a**].

In the situation where no information about the connectivity structure of the system 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. In [**AFD$^+$06**], Allen *et al.* extended REV-ENG to include random variable orders for reverse engineering the wiring diagram for a protein-signalling network from one discrete time series. Their algorithm compiles a summary table for the support of each interpolator computed from REV-ENG; the output is a consensus dependency graph.

We have modified the Allen method to construct a PDS (dependency graph and state space); see Algorithm REV-ENG-R in [**Sti05a**]. The input is a data set, a term order, and positive integer $r$. This procedure generates $r$ random variable orders and then builds a PDS for each of them with the given term order. For each $c = 1, \ldots, r$ and $i = 1, \ldots, n$, it records the variables that are in the support of an interpolator $f_i$ in an $n \times n$-matrix $M$. Once $c = r$, then the $i$-th row of the resulting matrix $M$ can be viewed as a summary of the number of times each variable appeared as a support element in the computation of the interpolator $f_i$. A set of variable orders can be constructed from $M$ by ordering the variables according to their entries in the matrix. The final step is to compute a PDS with this set of variable orders, which is input into REV-ENG-D.

Up to this point, we have only discussed reverse engineering from one set of state transitions. In general any single data set 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 algorithm described below accepts data sets from different types of experiments, where the data may come from so-called wildtype or knockout experiments, whereas the aforementioned algorithms require data sets from a single experiment, namely a wildtype experiment.

If the data are measurements of a biochemical network in its natural state, we call this a *wildtype* data set. If the data come from observations of a network in a perturbed state, where the perturbation is one such that one node of the network is completely inactivated, we call it a *knockout* data set.

For any $\mathbf{s} = (s_1, \ldots, s_n) \in k^n$, define

$$\mathbf{s}^i := (s_1, \ldots, s_{i-1}, 0, s_{i+1}, \ldots, s_n)$$

Then a knockout data set for node $i$ can be characterized as follows:

$$ko_i = \{(\mathbf{s}_1^i, \mathbf{t}_1^i), \ldots, (\mathbf{s}_m^i, \mathbf{t}_m^i)\}.$$

An *indexed knockout time series* is a pair $(i, ko_i)$ for $1 \leq i \leq n$ where $ko_i$ is a knockout data set for node $i$.

Consider a set of wildtype data and a set of indexed knockout time series. The data set used to compute an interpolator $f_i$ consists of the the wildtype data set, together with the all knockout data sets $ko_j$ with $j \neq i$. We give the algorithm

below. In Section 4 we provide an application of the algorithms to a simulated biochemical network.

REV-ENG-M: Algorithm for knockout data.

**Input**: $D = WT \cup KO$ data set where
$WT = \{(\mathbf{s}_1, \mathbf{t}_1), \ldots, (\mathbf{s}_m, \mathbf{t}_m)\}$ set of wildtype data,
$KO = \{(j, ko_j) : j \in A \subset \mathbb{N}\}$ set of indexed knockout data,
*t-order* = term order

**Output**: $F$ minimal PDS for $D$

**Algorithm**:
for each $i = 1 \ldots n$
DataPoints := $\{\mathbf{s}_1, \ldots, \mathbf{s}_m\} \cup \bigcup_{j \neq i} \{\text{inputs from } (j, ko_j)\}$
Sep := BM-SEP(DataPoints$_i$, *t-order*)
Values := $\{\mathbf{t}_1, \ldots, \mathbf{t}_m\} \cup \bigcup_{j \neq i} \{\text{outputs from } (j, ko_j)\}$
$f_i := \sum_{j \in Values} \text{Values}_j \text{Sep}_j$
endfor

**return** Minimal PDS $F = (f_1, ..f_n)$

The worst-time complexity of the algorithms is polynomial in the number of data points and variables. For a complexity analysis of all the algorithms introduced in this section, see [**Sti05a**].

**3.6. Reverse Engineering Wiring Diagrams.** In this subsection we provide conditions for determining if the Condition (2) of Problem 3.1 can be satisfied; that is, when the dependency graph of a constructed PDS contains all known interactions (also available in [**Sti05a**]).

Suppose we are given a data set $D = \{(\mathbf{s}_1, \mathbf{t}_1), \ldots, (\mathbf{s}_m, \mathbf{t}_m)\}$ and a dependency graph $(W, E)$. Consider the variety $V = \{\mathbf{s}_i : 1 \leq i \leq m\}$ of inputs from $D$. 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 (basis elements of $R/I(V)$) 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 interpolator.

PROPOSITION 3.10. *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 a smallest $m > 1$ 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 a minimal $m > 1$ 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 [**Kru02**] for more details).

PROPOSITION 3.11. *Let $V$ be a variety and $G$ be a Gröbner basis for $I(V)$. If $W \subset SM(G)$ and $|W| = |V| - 1$, then $supp(f) \subset W$ for all nonconstant $f \in R$.*

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 $f \in R/I(V)$. □

Data collected from a biochemical network can determine local properties of its structure. If $x_i$ is constant on all inputs, then node $i$ will have no outgoing edges since $x_i$ will not appear in the normal form for any polynomial under any term order. Similarly if the product of two variables $x_i, x_j$ is constant on all points in $V$, then $x_i x_j$ is not a standard monomial for any term order.

PROPOSITION 3.12. *Let $V \subsetneq k^n$ be a variety and $G$ be a Gröbner basis for $I(V)$. Let $\pi_i$ be the projection map of a point in $k^n$ onto its $i$-th coordinate. 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 $f \in R$, $NF(f, G)$ does not contain the variable $x_i$.*

PROOF. Suppose there is $i$ such that $\pi_i(\mathbf{a}) = c$ some $c \in k$. This holds iff $x_i - c \in I = \mathbf{I}(V)$. There is $g \in G$ with $LT(g)|LT(x_i - c)$. It follows that $LT(x_i - c) = x_i$ is not a standard monomial. Since the standard monomials form a basis for $R/I$, then $supp(NF(f, G)) \bigcap \{x_i\} = \emptyset$ for any $f \in R$. □

PROPOSITION 3.13. *Let $V \subsetneq k^n$ be a variety and $G$ be a Gröbner basis for $I(V)$. 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 $f \in R$, $NF(f, G)$ does not contain $x_i x_j$.*

PROOF. Suppose that for $i, j$ fixed, $x_i x_j(\mathbf{a}) = c$ for all $\mathbf{a} \in V$. Let $I = I(V)$. Then $x_i x_j \in LT(I)$ iff $x_i x_j \notin SM(G)$. 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.

## 4. Applications and Results

We validated our method by applying it to a simulated dataset from a well-studied embryonal network of segment polarity genes in the fruit fly *Drosophila melanogaster*. In [**AO03**] a Boolean model, a PDS over $\mathbb{F}_2$, was proposed for the network of 5 genes and their associated proteins. Our goal was to reverse engineer the dependency graph, as well as the certain features of the state space, including fixed points. Note that it is irrelevant whether the Boolean model is biologically correct. The following results can be found in detail in [**LS04**].

The genes represented in the Boolean model $\mathcal{N}$ 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 *smoothened* protein, denoted by SMO, and *sloppy-paired* proteins denoted as one compound SLP, constituting 15 distinct molecular species. For more details about the network and the corresponding Boolean model, see [**AO03**].

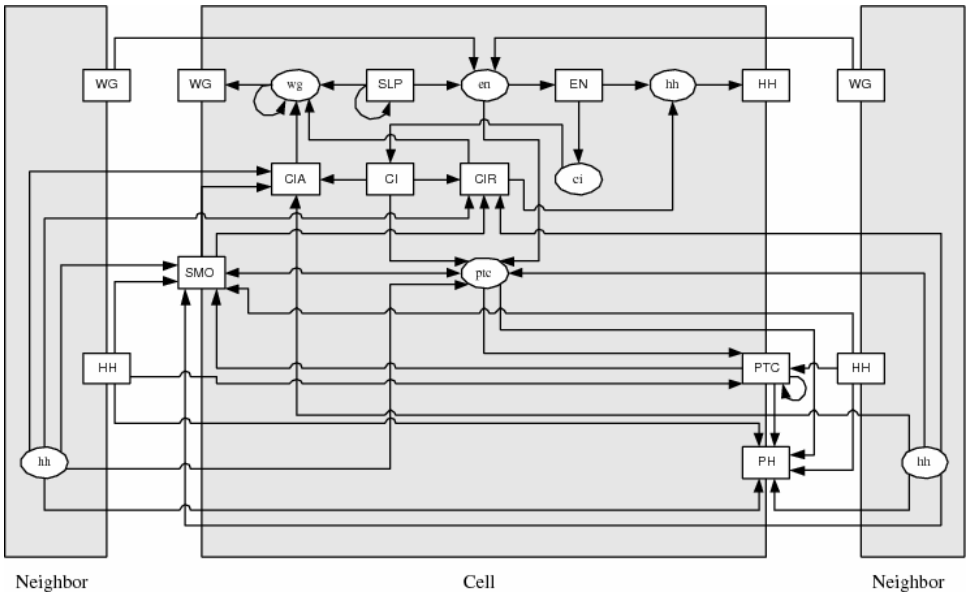


FIGURE 6. The dependency graph of $\mathcal{N}$. Ovals := mRNAs, rectangles := proteins.

In the dependency graph (Figure 6) for $\mathcal{N}$, 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 proteins A and B implies that A regulates the synthesis of B, whereas A→$b$ from protein A to mRNA $b$ implies that A regulates the transcription of gene $b$. Edges denote existence of regulation, not the type, whether activation or inhibition. To account for intercellular connections, we included 6 extra variables. We focus on the 15 variables representing the genes and proteins that constitute the polarity network in one embryonal cell. Table 4 lists the polynomial representations of the Boolean functions that define $\mathcal{N}$.

We used the wildtype Boolean initializations presented in [**AO03**] for the 5 genes and generated times series using the published Boolean functions. As reported by [**AO03**], all initializations terminate in fixed points when evaluated by the Boolean functions. 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:

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

Quick inspection reveals that our model from the general reverse-engineering algorithm produces minimal results, with 16 of the 44 edges correctly identified and 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. To improve the method's performance, we incorporated knock-out data.

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 4 to 0 and kept all other functions the same. When applicable, we also set the corresponding functions in

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

| $\mathrm{SLP}_i$ | $wg_i$ | $\mathrm{WG}_i$ | $en_i$ | $\mathrm{EN}_i$ | $hh_i$ | $\mathrm{HH}_i$ | $ptc_i$ |
|---|---|---|---|---|---|---|---|
| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | $x_8$ |
| $\mathrm{PTC}_i$ | $\mathrm{PH}_i$ | $\mathrm{SMO}_i$ | $ci_i$ | $\mathrm{CI}_i$ | $\mathrm{CIA}_i$ | $\mathrm{CIR}_i$ | |
| $x_9$ | $x_{10}$ | $x_{11}$ | $x_{12}$ | $x_{13}$ | $x_{14}$ | $x_{15}$ | |
| $\mathrm{WG}_{i-1}$ | $\mathrm{WG}_{i+1}$ | $\mathrm{HH}_{i-1}$ | $\mathrm{HH}_{i+1}$ | $hh_{i-1}$ | $hh_{i+1}$ | $Y$ | |
| $x_{16}$ | $x_{17}$ | $x_{18}$ | $x_{19}$ | $x_{20}$ | $x_{21}$ | $(x_{18}+1)(x_{19}+1)$ | |

TABLE 2. Polynomial representations of the Boolean functions in $\mathcal{N}$, together with the legend of variable names. The subscript $i$ refers to the cell of interest, $i-1$ the cell anterior to cell $i$, and $i+1$ the cell posterior to cell $i$.

neighboring cells equal to 0. For example, to simulate the knock out of 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 fixed point, by iteration of the functions given the modified initializations.

The effect of a variable ordering is that the "cheaper" variables, those that are ordered least, are used preferentially in computing interpolators (see Appendix B). Since we aim to reverse engineer the dependency graph 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 (results not shown here can be found in [**Sti05a**]).

**4.1. 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 dependency graph 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). For this experiment, we provide a detailed account of some of the false positives and true negatives.

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 a proof of this phenomenon in Section 3.6. 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 [**AO03**] the authors assumed in their model that this binding occurs instantaneously since it is known that the reaction occurs faster than

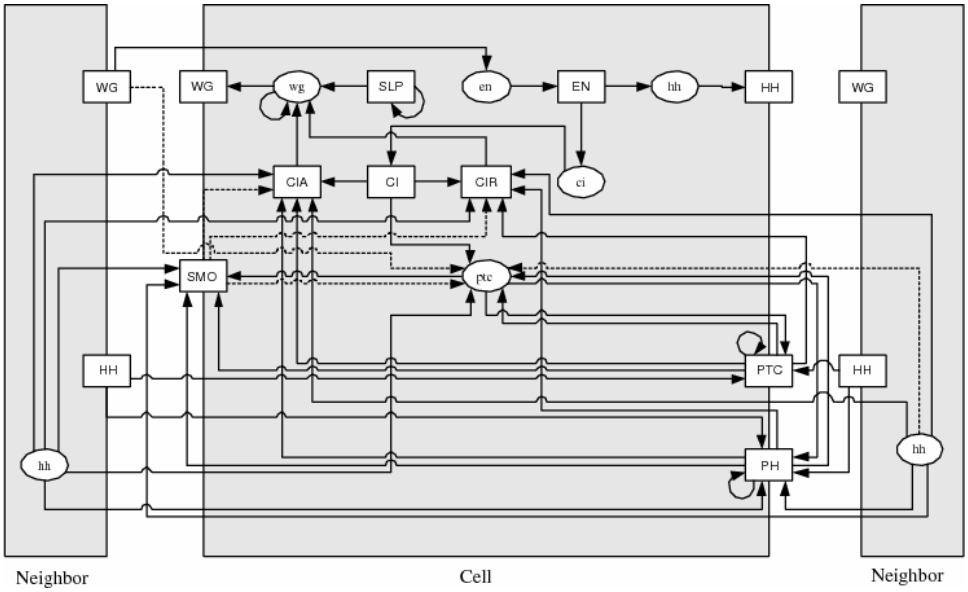


FIGURE 7. The dependency graph of the PDS built using REVENG-M/REVENG-D with the wildtype and knockout time series. Solid lines are links that appear for all 4 variable orders, whereas dashed lines are links that appear for 3 of the 4 variable orders.

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 PDS with the dynamics of $\mathcal{N}$, one approach is to compute the normal forms of the polynomials in Table 4 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 Boolean functions and the transition functions of the reverse-engineered PDS agree exactly. However, this observation occurs for the following reason.

Let $D = \{(\mathbf{s}_1, \mathbf{t}_1), \ldots, (\mathbf{s}_m, \mathbf{t}_m)\}$ be a collection of input-output pairs and suppose that $f, g$ are two polynomials that interpolate $D$. For each $1 \leq i \leq m, 1 \leq j \leq n$, we have that $f(\mathbf{s}_i) = g(\mathbf{s}_i) = t_{ij}$. Then the polynomial $f - g$ vanishes on all $\mathbf{s}_i$ and $f - g \in I(\mathbf{s}_1, \ldots, \mathbf{s}_t)$. 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 our method 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. We therefore proceed to extract information about

| **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 |

TABLE 3. Performance of dynamics detection for $\mathcal{N}$. 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.

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 model. 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 $\mathcal{N}$, 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 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. We call nonlinear terms "cooperative interactions." 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 3.

## 5. Discussion

We presented a collection of novel methods for reverse engineering biochemical networks given discrete data. The methods made use of algorithmic techniques in computational algebra, particularly in Gröbner basis theory. A distinctive feature of our approach is the ability to construct all polynomial dynamical systems that interpolate the data. This is accomplished by way of a Gröbner basis and does not require enumeration. A selection protocol based on minimality of the interpolating polynomials has been employed, where the polynomials are minimal with respect to the Gröbner basis of the ideal of the input 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 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 input data points.

The use of Gröbner bases in polynomial interpolation has also been explored in algebraic statistics. In fact, the REV-ENG algorithm is very similar to that in [**CR01**]. 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 to systems biology.

We demonstrated the effectiveness of inferring wiring diagrams for a simulated biochemical network. Incorporating data from different experiments, in particular wildtype and knockout experiments, dramatically improves 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 dynamics criterion of the reverse-engineering problem could be solved.

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. In fact there have been advancements in resolving the number of required data points. Dimitrova [**Dim06**] proposed an algorithm to minimally augment a given data set so that there is a unique PDS that fits the data. In the case of random data, Just [**Jus06**] provided some theoretical bounds on the required number of points. We are also working on improvements to the algorithms that decrease time complexity and increase scalability; see [**JLSS06, JS06**].

Finally we leave the reader with a look to the future from [**HP03**].

> 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.

## References


[ABKR00] J. Abbott, A. Bigatti, M. Kreuzer, and L. Robbiano, *Computing ideals of points*, Journal of Symbolic Computation **30** (2000), no. 4, 341–356.

[AFD+06] 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 **238** (2006), no. 2, 317–330.

[AL94] W. Adams and P. Loustaunau, *An introduction to gröbner bases*, Graduate Studies in Mathematics, vol. 3, American Mathematical Society, 1994.

[AO03] 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.

[aUoV] COMET Development Laboratory at University of Vermont, *Teaching modules*, Available at `http://cats.med.uvm.edu/cats_teachingmod/teachingmodules.html`, Online teaching modules offered through Department of Microbiology and Molecular Genetics.

[BM82] B. Buchberger and M. Möller, *The construction of multivariate polynomials with preassigned zeroes*, Computer Algebra: EUROCAM '82 (J. Calmet, ed.), Lecture Notes in Computer Science, vol. 144, Springer Berlin, 1982, pp. 24–31.

[Buc83] B. Buchberger, *A note on the complexity of constructing Groebner-Bases*, Computer Algebra: Proceedings of EUROCAL 83 (J. von Hulzen, ed.), Lecture Notes in Computer Science, vol. 162, Springer Berlin, 1983, pp. 137–145.

[Bue05] L. Buehler, *What is life?: A lifescience educational forum*, Available at `http://www.whatislife.com`, 2005.

[CLO97] D. Cox, J. Little, and D. O'Shea, *Ideals, varieties, and algorithms*, Springer Verlag, New York, 1997.

[CoC] CoCoATeam, CoCoA*: a system for doing Computations in Commutative Algebra*, Available at `http://cocoa.dima.unige.it`.

[CR01] 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.

[CRJLS07] O. Colón-Reyes, A. Jarrah, R. Laubenbacher, and B. Sturmfels, *Monomial dynamical systems over finite fields*, To appear, 2007.

[CRLP05] O. Colón-Reyes, R. Laubenbacher, and B. Pareigis, *Boolean monomial dynamical systems*, Annals of Combinatorics **8** (2005), no. 4, 425–439.

[DF91] D. Dummit and R. Foote, *Abstract algebra*, Prentice Hall, New Jersey, 1991.

[Dim06] E. Dimitrova, *Polynomial models for systems biology: Data discretization and term order effect on dynamics*, Ph.D. thesis, Virginia Polytechnic Institute and State University, 2006.

[dJ02] H. de Jong, *Modeling and simulation of genetic regulatory systems: A literature review*, Journal of Computational Biology **9** (2002), 67–103.

[DLM05] E. Dimitrova, R. Laubenbacher, and J. McGee, *Discretization of time series data*, Submitted, 2005.

[DLS00] 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.

[DWFS99] P. D'haeseleer, X. Wen, S. Fuhrman, and R. Somogyi, *Linear modeling of mrna expression levels during cns development and injury*, 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. 41–52.

[Edw00] R. Edwards, *Analysis of continuous-time switching networks*, Physica D **146** (2000), 165–199.

[Far01] M. Farabee, *Online biology book*, Available at `http://ridge.icu.ac.jp/biobk/biobooktoc.html`, 2001.

[fBI05a] National Center for Biotechnology Information, *Genbank database*, Available at `http://www.psc.edu/general/software/packages/ genbank/genbank.html`, 2005.

[fBI05b] ———, *Geo: Gene expression omnibus*, Available at `http://www.ncbi.nlm.nih.gov/geo`, 2005.

[FLNP00] 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.

[Gal04] M. Galperin, *The molecular biology database collection: 2004 update*, Nucleic Acids Research **32** (2004), D3–D22.

[GdBLC03] 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.

[GJL01] L. García, A. Jarrah, and R. Laubenbacher, *Classification of finite dynamical systems*, Available at `http://arxiv.org/abs/math.DS/0112216`, 2001.

[GPS01] G.-M. Greuel, G. Pfister, and H. Schönemann, *Singular 2.0*, A computer algebra system for polynomial computations, Centre for Computer Algebra, University of Kaiserslautern, 2001, Available at `http://www.singular.uni-kl.de`.

[GRS03] S. Gao, V. Rodrigues, and J. Stroomer, *Gröbner basis structure of finite sets of points*, Available at `http://citeseer.ist.psu.edu/gao03grbner.html`, 2003.

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

[HBS+04] A. Haubelt, E. Bullinger, T. Sauter, F. Allgöwer, and E. Gilles, *A glossary for systems biology*, Available at `http://sysbio.ist.uni-stuttgart.de/projects/glossary`, 2004.

[HGJY01] 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.

[HP03] A. Hastings and M. Palmer, *A bright future for biologists and mathematicians?*, Science **299** (2003), 2003–2004.

[HT05] R. Hernández-Toledo, *Linear finite dynamical systems*, Communications in Algebra **33** (2005), 2977–2989.

[IGH01] 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.

[IL03] T. Ideker and D. Lauffenburger, *Building with a scaffold: Emerging strategies for high- to low-level cellular modeling*, Trends in Biotechnology **21** (2003), 255–262.

[JLSS06] A. Jarrah, R. Laubenbacher, B. Stigler, and M. Stillman, *Reverse-engineering of polynomial dynamical systems*, Advances in Applied Mathematics (2006), In Press.

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

[JLVL06] A. Jarrah, R. Laubenbacher, and P. Vera-Licona, *An efficient algorithm for finding the phase space structure of linear finite dynamical systems*, Preprint, 2006.

[JS06] W. Just and B. Stigler, *Computing Gröbner bases of ideals of few points in high dimensions*, Communications in Computer Algebra **40** (2006), no. 3, 65–96.

[Jus06] W. Just, *Reverse engineering discrete dynamical systems from data sets with random input vectors*, Journal of Computational Biology **13** (2006), no. 8, 1435–1456.

[Kit02] H. Kitano, *Systems biology: A brief overview*, Science **295** (2002), 1662–1664.

[Kru02] B. Krupa, *On the number of experiments required to find the causal structure of complex systems*, Journal of Theoretical Biology **219** (2002), 257–267.

[LFS98] 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.

[LN97] R. Lidl and H. Niederreiter, *Finite fields*, 2nd ed., Encyclopedia of Mathematics and its Applications, vol. 20, Cambridge University Press, New York, 1997.

[LP01] R. Laubenbacher and B. Pareigis, *Equivalence relations on finite dynamical systems*, Advances in Applied Mathematics **26** (2001), no. 3, 237–251.

[LP03] ———, *Decomposition and simulation of sequential dynamical systems*, Advances in Applied Mathematics **30** (2003), 655–678.

[LS04] 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.

[Map] Maplesoft, Available at `http://www.maplesoft.com/`.

[May04] R. May, *Uses and abuses of mathematics in biology*, Science **303** (2004), 790–793.

[Med04] Annenberg Media, *Rediscovering biology: Molecular to global perspectives*, Available at `http://www.learner.org/channel/courses/biology`, 2004.

[MMM93] M. Marinari, H.M. Möller, and T. Mora, *Gröbner bases of ideals defined by functionals with an application to ideals of projective points*, Applicable Algebra in Engineering, Communication and Computing **4** (1993), 103–145.

[MR93] T. Mora and L. Robbiano, *Points in affine and projective spaces*, Computational Algebraic Geometry and Commutative Algebra, Cortona-91 (D. Eisenbud and L. Robbiano, eds.), Symposia Mathematica, vol. 34, Cambridge University Press, 1993, pp. 106–150.

[oB05] Swiss Institute of Bioinformatics, *Swiss-prot protein knowledgebase*, Available at `http://us.expasy.org/sprot`, 2005.

[PREF01] D. Pe'er, A. Regev, G. Elidan, and N. Friedman, *Inferring subnetworks from perturbed expression profiles*, Bioinformatics **17** (2001), S215–S224, Supplement 1.

[Rob98] L. Robbiano, *Gröbner bases and statistics*, Gröbner Bases and Applications (New York) (B. Buchberger and F. Winkler, eds.), London Mathematical Society Lecture Notes Series, vol. 251, Cambridge University Press, 1998, pp. 179–204.

[Sti05a] B. Stigler, *An algebraic approach to reverse engineering with an application to biochemical networks*, Ph.D. thesis, Virginia Polytechnic Institute and State University, 2005.

[Sti05b] ———, *Polynome: Polynomial models of biological systems*, Available at `http://polymath.vbi.vt.edu/rev-eng/reveng.php`, 2005.

[TYHC03] 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.

[vB68] L. von Bertalanffy, *General system theory*, third ed., George Braziller, Inc., New York, 1968.

[YTC02] 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.

## Appendix A. Resources

For certain sections of these notes, we have included a listing of resources.

Introduction: What is Systems Biology?

- For a description of the systems-biology paradigm and examples of projects, see [**IGH01, Kit02**].
- Biological databases
  - GenBank: DNA sequence database [**fBI05a**]
  - Gene Expression Omnibus: repository for high-throughput gene expressions and hybridization arrays [**fBI05b**]
  - Swiss-Prot: curated protein sequence database [**oB05**]
  - For other databases, see the Molecular Biology Database Collection [**Gal04**].
- Molecular biology: [**Far01, aUoV, Med04, Bue05**]
- Systems biology: [**dJ02, DLS00, IGH01, HBS$^+$04**].

Finite Dynamical Systems

- For uses of PDSs in systems biology, see [**AFD$^+$06, LS04, Sti05a**].
- For a classification of FDSs, see [**CRLP05, CRJLS07, GJL01, JLVL06, HT05, LP01, LP03**].
- Modeling and visualization software: Polynome [**Sti05b**] and DVD [**JLV**].

Reverse Engineering using PDSs

- Discretization software: [**DLM05**]
- Connections to computational algebra
  - Abstract algebra: [**DF91**]
  - Gröbner-basis theory: [**AL94, CLO97**]
  - Algebraic geometry: [**CLO97**]
- Reverse-engineering software: Polynome [**Sti05b**]
- Software packages for computing with Gröbner bases
  - CoCoA [**CoC**]
  - *Macaulay 2* [**GS**]
  - Maple [**Map**]
  - Singular [**GPS01**]

## Appendix B. 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.

As division operates on the largest term of a polynomial, it is important to first establish an ordering on the monomials. In the polynomial ring 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 $xy^2$. What is needed is a total ordering on all possible monomials in $R = k[x_1, \ldots, x_n]$, where $k$ is any field.

Definition B.1. A *monomial ordering* (or *term order*) on $R$ 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 $>$.

While there are an infinite number of term orders, we will primarily focus on three types, whose characterizations are given below.

Definition B.2.

(1) (Lexicographic) Let $\mathbf{x}^a, \mathbf{x}^b \in R$ 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 $\mathbf{x}^a, \mathbf{x}^b \in R$ 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.
(3) ($i$-th Elimination) Let $1 \leq i \leq n$. Then a monomial order $>_i$ on $R$ is of *$i$-elimination type* if any monomial involving $x_1, \ldots, x_i$ is greater than all monomials in $k[x_{i+1}, \ldots, x_n]$.

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 types of monomial orders. We call the initial ordering of the variables a *variable order*.

Example B.3. Let $k$ be a field and consider 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.

Polynomial division is dependent 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. However, we will see in the following example that changing term orders produces different remainders. To see how multivariate polynomial division is defined, see Chapter 2.3 in [**CLO97**].

Example B.4. 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.

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

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.

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

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.

Another problem that is encountered is when a polynomial is divided successively by a set of polynomials. Say we want to divide the polynomial $f = x^5 + yz$ by $\{g = x - yz, h = x^4 + 1\}$ 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 $y^5 z^z + yz$. However if we divide in the opposite order, that is by $h$ first and then by $g$, we get a remainder of 0. What this means is that the modulus operation % is not associative: $(f\%g)\%h \neq (f\%h)\%g$.

In terms of the ideal membership problem, we see 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 elements of the generating set do not lend themselves for division: their leading terms do not divide the leading terms of all elements in $\langle g, h\rangle$. What is needed is a "nice" generating set so that polynomial division can be performed unambiguously.

Definition B.5. Let $>$ be a monomial order on $R$ and let $I \subset R$ be an ideal. A finite subset $G = \{g_1, \ldots, g_m\} \subset I$ is a *Gröbner basis* for $I$ if for any $f \in I$ there is $g_i \in G$ such that $LT(g_i)|LT(f)$ under $>$, or equivalently, $\langle LT(g) : g \in G\rangle = \langle LT(f) : f \in I\rangle$.

Theorem B.6. *Let $>$ be a monomial order on $R$. Every nonzero ideal $I \subset R$ has a Gröbner basis $G$. Moreover $G$ is a generating set for $I$.*

Definition B.7. Let $G$ be a Gröbner basis for an ideal $I \subset R$ and let $f \in R$. The *normal form* of $f$ with respect to $G$, denoted by $NF(f, G)$, is the remainder of $f$ on division by the elements of $G$.

By construction, Gröbner basis elements are chosen so that their leading terms divide the leading terms of all other elements of a given ideal. This property makes it so that polynomial division is well defined.

Theorem B.8. *Let $G$ be a Gröbner basis for an ideal $I \subset R$ and let $f \in R$. Then $NF(f, G)$ is unique.*

The following well-known result is used to solve the ideal membership problem.

Theorem B.9. *Let $G$ be a Gröbner basis for an ideal $I \subset R$ and let $f \in R$. Then $f \in I$ iff $NF(f, G) = 0$.*

Example B.10. Let $>$ be the grevlex order with variable order $x > y$. Consider the ideal $I$ generated by the polynomials $f = x^2 + x + y$ and $g = x + y$. 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 $a, b \in \mathbb{Q}[x, y]$, 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. Division is not possible since the leading terms of $f$ and $g$ involve $x$. However, it is true that

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

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

Consider the set $G = \{x + y, y^2 + x + y\}$. Note that $y^2 = f - g - (x - y)g$. We see that $G$ is 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 $h \in I$ since $NF(h, G) = 0$.

EXAMPLE B.11. Consider the ideal $I = \langle x^2 + x + y, x + y \rangle \subset \mathbb{Q}[x, y]$ from Example B.10. Let $>$ be the grevlex order with $x > y$. We saw above that $\{y^2+x+y, x+y\}$ is a Gröbner basis for $I$. So is the set $\{y^2, x+y\}$ as it satisfies the conditions for being a Gröbner basis. What this illustrates is that Gröbner bases are not unique in general. However, if a Gröbner basis is *reduced*, then it is unique.

DEFINITION B.12. A Gröbner basis $G = \{g_1, \ldots, g_m\}$ is a *reduced* Gröbner basis if every $g \in G$ is monic (leading coefficient is 1) and no term of $g_i$ is divisible by any $LT(g_j)$ for $1 \leq i \neq j \leq m$.

THEOREM B.13. *Every nonzero ideal in $R$ has a unique reduced Gröbner basis with respect to a fixed term order.*

In Example B.11, the set $\{y^2, x + y\}$ is the reduced Gröbner basis for $I$. In the rest of this discourse, a Gröbner basis will be taken to be reduced.

Recall that the transition functions of a PDS are polynomials in $R$. As functions defined on a data set $V$, however, they are elements of the quotient ring $k[V] := R/I(V)$, called the *coordinate ring*. Given a Gröbner basis $G$ of an ideal $I \subset R$ with respect to a term order $>$, we can view the quotient ring as a vector space over $k$. Then the set $SM(G) := \{\mathbf{x}^a : \mathbf{x}^a \notin \langle LT(G) \rangle\}$ forms a basis for $R/I$; this set is called the set of *standard* or *basis 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.

THEOREM B.14. *Let $V$ be a variety in $k^n$ and consider $I = I(V) \subset R$. Let $G$ be a (reduced) Gröbner basis for $I$. Then $SM(G)$ is a basis for $R/I$ as a vector space over $k$ and $|V| = |SM(G)|$.*

The former statement is true for any field $k$. The latter, however, is true only for perfect fields, which include algebraically closed fields and finite fields. (See [**GRS03**].)

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 original algorithm to compute Gröbner bases was proposed by Bruno Buchberger and is known to be exponential in the number of variables [**Buc83**]. However, there are a number of improvements with better complexity (for example, see [**Jus06**]). The Buchberger-Möller algorithm is quadratic in the number of variables and cubic in the number of points [**Rob98**]. It has been implemented in various computer algebra systems, including CoCoA[**CoC**] and *Macaulay 2* [**GS**]. See Appendix C for the algorithm and an example.

## Appendix C. The Buchberger-Möller Algorithm

Given a variety $V$ and a term order $>$, the Buchberger-Möller algorithm (BMA) computes the reduced Gröbner basis $G$ for the ideal $I(V)$ with respect to $>$, the set $SM(G)$ of standard monomials for $G$, and the set of reduced separators of the points in $V$ [**BM82**]. The BMA has worst-case complexity $O(nm^3 + n^2m^2)$, where $n$ is the number of variables and $m$ is the number of points in $V$; for details see [**ABKR00, MMM93, MR93**].

**Input**: $V = \{p_1, \ldots, p_m\}$ a variety, >= a term order
**Output**: S = separators of $V$, G = Gröbner basis of $I(V)$, SM = set of standard monomials

**Algorithm**:
S = ∅; G = ∅; SM = ∅;
$r = 0$
L = [1] –L=candidate std. mon.
while L ≠ 0
t = min(L) –smallest monomial in L
L = L \ {t}
f = t $-\sum_{i=1}^{s}$ t($p_{\pi(i)}$)$s_i$ –$1^{st} p$ : f($p$)≠0; indx calc. in "else"
if f vanishes on $V$ –then f is in the ideal
G = G ⋃ f
L = L \ {multiples of t} –remove multiples; want G red.
else
SM = SM ⋃ {t} –t ∤ any LT in the ideal
$r = r+ 1$
$\pi(r) = \min\{i \mid \mathrm{f}(p_i) \neq 0\}$
$s_r = \mathrm{f}(p_{\pi(r)})^{-1}\mathrm{f}$ –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 ⋃ {$x_i$t | $i$=1..n} \ LT(G)
endif
endwhile
**return** S, G, SM

Next we compute a small example to illustrate the various steps of the BMA. Let $V$ be the variety $V = \{(0,1),(1,0)\} \subset (\mathbb{F}_2)^2$ and consider the ideal $I(V) \subset \mathbb{F}_2[x_1, x_2]$ under any term order with $x_1 < x_2$. We will execute each step of the BMA and show the result of each pass through the algorithm. At termination, the following sets are returned:

$$\begin{aligned} S &= \{s_1, s_2\} = \{x_1, -x_1 + 1\}, \\ G &= \{x_2 + x_1 - 1, x_1^2 - x_1\}, \\ SM &= \{1, x_1\}. \end{aligned}$$

| $S=0; G=0; SM=0; r=0; L=\{1\}$ | | | |
|---|---|---|---|
| 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* |
| | | $G=\{x_2+x_1-1\}$ | $G=\{x_2+x_1-1, x_1^2-x_1\}$ |
| | | $L=\{x_1^2\}$ | $L=\emptyset$ |
| *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* | | |
| | $s_1=x_1$ | | |
| $L=\{x_1, x_2\}$ | $L=\{x_2, x_1^2, x_2x_1\}$ | | |

Mathematical Biosciences Institute, The Ohio State University, Columbus, Ohio 43210

*E-mail address*: bstigler@mbi.osu.edu