CHAPTER SEVEN

Algebraic Models of Biochemical Networks

Reinhard Laubenbacher *and* Abdul Salam Jarrah

Contents

Abstract

With the rise of systems biology as an important paradigm in the life sciences and the availability and increasingly good quality of high-throughput molecular data, the role of mathematical models has become central in the understanding of the relationship between structure and function of organisms. This chapter focuses on a particular type of models, so-called algebraic models, which are generalizations of Boolean networks. It provides examples of such models and discusses several available methods to construct such models from high-throughput time course data. One specific such method, *Polynome*, is discussed in detail.

Virginia Bioinformatics Institute at Virginia Tech, Blacksburg, Virginia, USA

Methods in Enzymology, Volume 467 © 2009 Elsevier Inc.
ISSN 0076-6879, DOI: 10.1016/S0076-6879(09)67007-5 All rights reserved.

1. Introduction

The advent of functional genomics has enabled the molecular biosciences to come a long way towards characterizing the molecular constituents of life. Yet, the challenge for biology overall is to understand how organisms function. By discovering how function arises in dynamic interactions, systems biology addresses the missing links between molecules and physiology.

(Bruggemann and Westerhoff 2006)

With the rise of systems biology as an important paradigm in the life sciences and the availability and increasingly good quality of high-throughput molecular data, the role of mathematical models has become central in the understanding of the relationship between structure and function of organisms. It is by now well understood that fundamental intracellular processes such as metabolism, signaling, or various stress responses, can be conceptualized as complex dynamic networks of interlinked molecular species that interact in nonlinear ways with each other and with the extracellular environment. Available data, such as transcriptional data obtained from DNA microarrays or high-throughput sequencing machines, complemented by single-cell measurements, are approaching a quality and quantity that makes it feasible to construct detailed mechanistic models of these networks.

There are two fundamental approaches one can take to the construction of mathematical or statistical network models. For each approach the first step is to choose an appropriate model type, which might be, for example, a dynamic Bayesian network (DBN) or a system of ordinary differential equations (ODE). The traditional *bottom-up* approach begins with a ''parts list,'' that is, a list of the molecular species to be included in the model, together with information from the literature about how the species interact. For each model type, there will then likely be some model parameters that are unknown. These will then either be estimated or, if available, fitted to experimental data. The result is typically a detailed mechanistic model of the network. That is, the selected parts are assembled to a larger system.

Systems biology has provided another, so-called *top-down* approach, which attempts to obtain an unbiased view of the underlying network from high-throughput experimental data alone, using statistical or mathematical network inference tools. The advantage of this approach is that the results are not biased by a perception or presumed knowledge of which parts are important and how they work together. The disadvantage is that the resulting model is most likely phenomenological, without a detailed mechanistic structure. It is also the case that at this time the available data sets for this approach are rather small, compared to the number of network nodes, so that the resulting network inference problem is underdetermined.

The most beneficial approach, therefore, is to combine both methods by using the available information about the network as prior information for an appropriate network inference method. Thus, network inference becomes an extreme case of parameter estimation, in which no parameters are specified. The development of appropriate network inference methods has become an important research field within computational systems biology.

The goal of this chapter is to illustrate parameter estimation and network inference using a particular type of model, which we will call *algebraic model*. Boolean networks, which are being used increasingly as models for biological networks, represent the simplest instance. First, in Section 2, we will provide an introduction to computational systems biology and give some detailed examples of algebraic models. In the following section, we provide an introduction to network inference within several different model paradigms and provide details of several methods. Finally, in the last section, we talk about network inference as it pertains specifically to algebraic models. Few inference methods provide readily available software. We describe one of those, *Polynome*, in enough detail so that the reader can explore the software via the available Web interface.

2. Computational Systems Biology

To understand how molecular networks function it is imperative that we understand the dynamic interactions between their parts. Gene regulatory networks give an important example. They are commonly represented mathematically as so-called directed graphs, whose nodes are genes and sometimes proteins. There is an arrow from gene A to gene B, if A contributes to the regulation of B in some way. The arrow typically indicates whether this contribution is an activation or an inhibition. It is important to understand that such a network provides an abstract representation of gene regulation in the sense that the actual regulation is not direct but could involve a fairly long chain of biochemical reactions, involving mRNA from gene A and/or a protein A codes for.

As an example, we consider the *lac* operon in *E. coli*, one of the earliest and best understood examples of gene regulation. We will use this gene network as a running example for the entire chapter. *E. coli* prefers glucose as a growth medium, and the operon genes allow *E. coli* to metabolize lactose in the absence of glucose. When glucose is present it is observed that the enzymes involved in lactose metabolism have very low activity, even if intracellular lactose is present. In the absence of glucose, lactose metabolism is induced through expression of the *lac* operon genes. Figure 7.1 shows a representation of the basic biological mechanisms of the network, commonly referred to as a "cartoon" representation.

While this representation is very intuitive, to understand the biology, a network representation of the system is a first step toward the construction of a dynamic mathematical model. Such a representation is given in Fig. 7.2.

The dynamic properties of this network are determined by two control mechanisms, one positive, leading to induction of the operon, and another

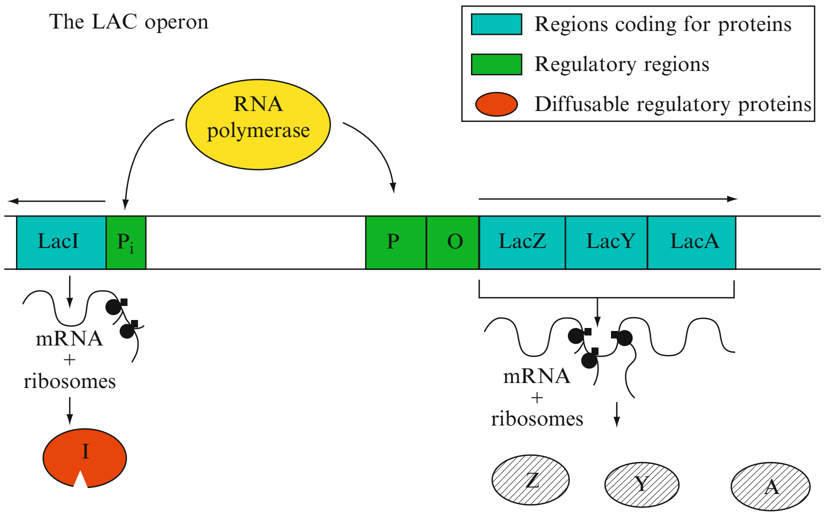


Figure 7.1 The *lac* operon (from http://www.uic.edu/classes/bios/bios100/lecturesf04am/lect15.htm).

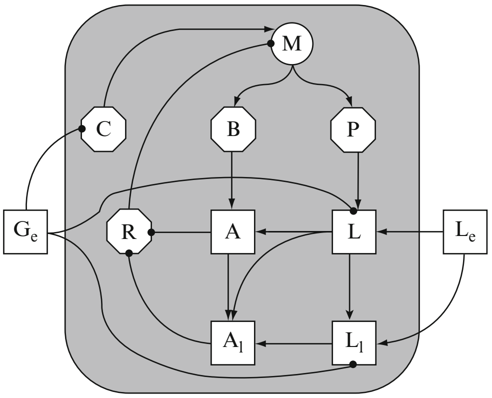


Figure 7.2 The *lac* operon network (from Stigler and Veliz-Cuba, 2009).

one is negative, leading to the repression of the operon mechanism. The negative control is initiated by glucose, through two modes of action. In the absence of intracellular glucose, the catabolite activator protein CAP forms a complex with another protein, cAMP, which binds to a site upstream of the *lac* promoter region, enhancing transcription of the *lac* genes. Intracellular glucose inhibits cAMP synthesis and thereby gene transcription. Furthermore, extracellular glucose inhibits the uptake of lactose into the cell via lactose permease, one of the proteins in the *lac* operon. The positive control operates via the action of lactose permease, increasing intracellular lactose, and by disabling of the *lac* repressor protein via increased production of allolactose. To understand how these two feedback loops, one positive and the other negative, work together to determine the dynamic properties of this network it is necessary to construct a mathematical model that captures this synergistic interplay of positive and negative feedback.

Several different mathematical modeling frameworks are available for this purpose. The most commonly used type of model for molecular networks is a system of ODE, one equation for each of the nodes in the network. Each equation describes the rate of change of the concentration of the corresponding molecular species over time, as a function of other network nodes involved in its regulation. As an example, we present a very simplified differential equations model of the *lac* operon taken from Section 5.2 of deBoer. This is an example of a model which was referred to in the introduction as "bottom-up." The model includes only the repressor R, the *lac* operon mRNA M, and allolactose A. The three equations are given below:

$$R = 1/(1 + A^n),$$

$$\mathrm{d}M/\mathrm{d}t = c_0 + c(1 - R) - \gamma M,$$

$$\mathrm{d}A/\mathrm{d}t = ML - \delta A - vMA/(h + A).$$

Here, c_0, c, γ, v, δ, h, and L are certain model parameters, n is a fixed positive integer, and the concentrations R, M, and A are functions of time t. The model does not distinguish between intracellular and extracellular lactose, both denoted by L. It is assumed further that the enzyme β-galactosidase is proportional to the operon activity M and is not represented explicitly. The repressor concentration R is represented by a so-called Hill function, which has a sigmoid-shaped graph: the larger the Hill coefficient n, the steeper the shape of the sigmoid function. The constant c_0 represents the baseline activity of the operon transcript M, and the term γM represents degradation. The concentration of allolactose A grows with M, assuming that lactose L is present. Its degradation term is represented by a Michaelis–Menten type enzyme substrate reaction composed of two terms. The various model parameters can be estimated to fit experimental data, using parameter estimation algorithms.

This model is not detailed enough to incorporate the two different feedback loops discussed earlier, but will serve as an illustration of the kind of information a dynamic model can provide. The most important information typically obtained from a model is about the steady states of the network, that is, network states at which all derivatives are equal to 0, so that the system remains in the steady state, once it reaches it. A detailed analysis of this model can be found in deBoer and also in Laubenbacher and Sturmfels (2009). Such an analysis shows that the model has three steady states, two stable and one unstable. This is what one would expect from the biology. Depending on the lactose concentration, the operon is either "on" or "off," resulting in two stable steady states. For bacteria growing in an environment with an intermediate lactose concentration, it has been shown that the operon can be in either one of these two states, and which one is attained depends on the environmental history a particular bacterium has experienced, a form of hysteresis. This behavior corresponds to the unstable steady state. Thus, the model dynamics agrees with experimental observations.

Another modeling framework that has gained increasing prominence is that of Boolean networks, initially introduced to biology by Kauffman (1969) as a model for genetic control of cell differentiation. Since then, a wide array of such models has been published, as discussed in more detail below. They represent the simplest examples of what is referred to in the title of this chapter as *algebraic models*. They are particularly useful in cases when the quantity or quality of available experimental data is not sufficient to build a meaningful differential equations model. Furthermore, algebraic models are just one step removed from the way a biologist would describe the mechanisms of a molecular network, so that they are quite intuitive and accessible to researchers without mathematical background. In particular, this makes them a useful teaching tool for students in the life sciences (Robeva and Laubenbacher, 2009). To illustrate the concept, we present here a Boolean model of the *lac* operon, taken from Stigler and Veliz-Cuba (2009).

A Boolean network consists of a collection of nodes or variables, each of which can take on two states, commonly represented as ON/OFF or 1/0. Each node has attached to it a Boolean function that describes how the node depends on some or all of the other nodes. Time progresses in discrete steps. For a given state of the network at time $t = 0$, the state at time $t = 1$ is determined by evaluating all the Boolean functions at this state.

Example 7.1

We provide a simple example to illustrate the concept. Consider a network with four nodes, $x_1, \ldots, x_4$. Let the corresponding Boolean functions be

$$f_1 = x_1 \text{ AND } x_3, \quad f_2 = x_2 \text{ OR } (\text{NOT } x_4), \quad f_3 = x_1, \quad f_4 = x_1 \text{ OR } x_2.$$

Then this Boolean network can be described by the function

$$f = (f_1, \ldots, f_4) : \{0, 1\}^4 \to \{0, 1\}^4,$$

where

$$f(x_1, \ldots, x_4) = (x_1 \text{ AND } x_3, x_2 \text{ OR } (\text{NOT } x_4), x_1, x_1 \text{ OR } x_2). \qquad (7.1)$$

As a concrete example, $f(0, 1, 1, 0) = (0, 1, 0, 1)$. Here, $\{0, 1\}^4$ represents the set of all binary 4-tuples, of which there are 16. The dependencies among the variables can be represented by the directed graph in Fig. 7.3, representing the wiring diagram of the network.

The dynamics of the network is represented by another directed graph, the discrete analog of the phase space of a system of differential equations, given in Fig. 7.4.

The nodes of the graph represent the 16 possible states of the network. There is a directed arrow from state $\boldsymbol{a}$ to state $\boldsymbol{b}$ if $f(\boldsymbol{a}) = \boldsymbol{b}$. The network has three steady states, (0, 1, 0, 1), (1, 1, 1, 1), and (1, 0, 1, 1). (In general, there might be periodic points as well.)

A Boolean model of the lac operon. The following model is presented in Stigler and Veliz-Cuba (2009). In addition to the three variables M (mRNA for the 3 *lac* genes), R (the repressor protein), and A (allolactose) in the previous ODE model, we need to include these additional variables:

- *Lac* permease (P)
- β-Galactosidase (B)
- Catabolite activator protein CAP (C)
- Lactose (L)
- Low concentrations of lactose (L_{low}) and allolactose (A_{low})

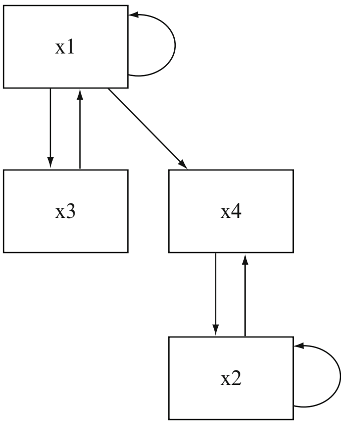


Figure 7.3 Wiring diagram for Boolean network in Example 7.1 (constructed using the software tool DVD; http://dvd.vbi.vt.edu).

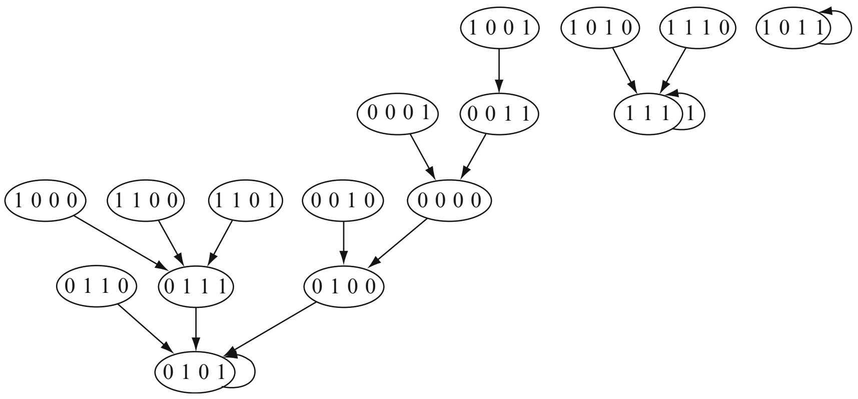


Figure 7.4 Dynamics of the Boolean network in Example 7.1 (constructed using the software tool DVD; http://dvd.vbi.vt.edu).

The last two variables are needed for the model to be accurate, since we need to allow for three, rather than two, possible concentration levels of lactose and allolactose: absent, low, and high. Introducing additional binary variables to account for the three states avoids the introduction of variables with more than two possible states. (Models with multistate variables will be discussed below.) The model depends on two external parameters: a, representing the concentration of external lactose and g, representing the concentration of external glucose. They can both be set to the values 0 and 1, providing four different choices. The interactions between these different molecular species are described in Stigler and Veliz-Cuba (2009) by the following Boolean functions:

$$\begin{gathered}
f_M = (\text{NOT}\,R)\,\text{AND}\,C,\\
f_P = M,\\
f_B = M,\\
f_C = \text{NOT}\,g,\\
f_R = (\text{NOT}\,A)\,\text{AND}\,(\text{NOT}\,A_{\text{low}}),\\
f_A = L\,\text{AND}\,B,\\
f_{A\text{low}} = A\,\text{OR}\,L\,\text{OR}\,L_{\text{low}},\\
f_L = (\text{NOT}\,g)\,\text{AND}\,P\,\text{AND}\,a,\\
f_{L\text{low}} = (\text{NOT}\,g)\,\text{AND}\,(L\,\text{OR}\,a).
\end{gathered} \tag{7.2}$$

To understand these Boolean statements and how they assemble to a mathematical network model, consider the first one. It represents a rule that describes how the concentration of *lac* mRNA evolves over time. To make the time dependence explicit, one could write $f_M(t+1) = (\text{NOT } R(t))$ AND $C(t)$ (similarly for the other functions). This is to be interpreted as saying that the *lac* genes are transcribed at time $t+1$ if the repressor protein

R is absent at time *t* and the catabolite activator protein *C* is present at time *t*. The interpretation of the other functions is similar. One of the model assumptions is that transcription and translation of a gene happens in one time step, and so does degradation of mRNA and proteins.

Thus, choosing the above ordering of these nine variables, a state of the *lac* operon network is given by a binary 9-tuple, such as (0, 1, 1, 0, 0, 1, 0, 1, 1). The above Boolean functions can be used as the coordinate functions of a time-discrete dynamical system on $\{0, 1\}^9$, that is, a function

$$f = (f_M, f_P, f_B, f_C, f_R, f_A, f_{A\text{low}}, f_L, f_{L\text{low}}) : \{0,1\}^9 \rightarrow \{0,1\}^9.$$

For a given network state $\boldsymbol{a}$ in $\{0, 1\}^9$, the function f is evaluated as

$$f(a) = (f_M(a), f_P(a), f_B(a), f_C(a), f_R(a), f_A(a), f_{A\,\text{low}}(a), f_L(a), f_{L\,\text{low}}(a))$$

to give the next network state. For the exemplary network state above, and the parameter setting $a = 1, g = 0$, we obtain

$$f(0,1,1,0,0,1,0,1,1) = (0,0,0,1,0,1,1,1,1).$$

It is shown in Stigler and Veliz-Cuba (2009) that this model captures all essential features of the *lac* operon, demonstrated through several other published ODE models. One of these is its bistability, which was already discussed in the simple model above. It is shown that for each of the four possible parameter settings the models attains a steady state, corresponding to the switch-like nature of the network. It is in principle possible to compute the entire phase space of the model, using a software package such as DVD (http://dvd.vbi.vt.edu). The space, which contains 2^9 states, is too large to be visualized, however.

We have discussed this particular model in some detail, for three reasons. It represents a model of an interesting network that continues to be the object of ongoing research, yet is simple enough to fit the space constraints of this chapter. For the reader unfamiliar with algebraic models of this type, it provides a detailed realistic example to explore. Finally, we will use this particular model subsequently to demonstrate a particular network inference method.

Boolean network models of biological systems are the most common type of discrete model used, including gene regulatory networks such as the cell cycle in mammalian cells (Faure *et al.*, 2006), in budding yeast (Li *et al.*, 2004) and fusion yeast (Davidich and Bornholdt, 2007), and the metabolic networks in *E. coli* (Samal and Jain, 2008) and in *Saccharomyces cerevisiae* (Herrgard *et al.*, 2006).

Also, Boolean network models of signaling networks have recently been used to provide insights into different mechanisms such as the molecular neurotransmitter signaling pathway (Gupta *et al.*, 2007), the T cell receptor signaling pathways (Saez-Rodriguez *et al.*, 2007), the signaling network for the long-term survival of cytotoxic T lymphocytes in humans (Zhang *et al.*, 2008), and the abscisic acid signaling pathway (Li *et al.*, 2006a).

A more general type of discrete model, so-called *logical models*, was introduced by the geneticist (Thomas and D'Ari, 1989) for the study of gene regulatory networks. Since then they have been developed further, with published models of the cell-fate determination in *Arabidopsis thaliana* (Espinosa-Soto *et al.*, 2004), the root hair regulatory network (Mendoza and Alvarez-Buylla, 2000), the Hh signaling pathway (Gonzalez *et al.*, 2008), the gap gene network in Drosophila (Sanchez and Thieffry, 2001), and the differentiation process in T helper cells (Mendoza, 2006), to name a few.

A logical model consists of a collection of variables $x_1, \ldots, x_n$, representing molecular species such as mRNA, where variable x_j takes values in a finite set S_j. The number of elements in S_j corresponds to the number of different concentrations of x_j that trigger different modes of action. For instance, when transcribed at a low level, a gene might perform a different role than when it is expressed at a high level, resulting in three states: absent, low, and high. The variables are linked in a graph, the *dependency graph* or *wiring diagram* of the model, as in the Boolean case, by directed edges, indicating a regulatory action of the source of the edge on the target. Each edge is equipped with a sign $+/-$ (indicating activation or inhibition) and a weight. The weight indicates the level of transcription of the source node required to activate the regulatory action. The state transitions of each node are given by a table involving a list of so-called logical parameters. The dynamics of the system is encoded by the *state space graph*, again as in the Boolean case, whose edges indicate state transitions of the system. An additional structural feature of this model type is that the variables can be updated sequentially, rather than in parallel, as in the previously discussed Boolean model. This feature allows the inclusion of different time scales and stochastic features that lead to asynchronous updating of variables. The choice of different update orders at any given update step can result in a different transition. To illustrate the logical model framework, we briefly describe the T cell differentiation model in Mendoza (2006).

As they differentiate from a common precursor called T0, two distinct functional subsets T1 and T2 of T helper cells have been identified in the late 1980s. T1 cells secret IFNγ, which promotes more T1 differentiation while inhibiting the differentiation into T2. On the other hand, T2 cells secret IL-4, a cytokine which promotes T2 differentiation and inhibits that of T1. The most general logical model is presented in Mendoza (2006), where the gene regulatory network of T1/T2 differentiation is synthesized from published experimental data. The multilevel network includes 19 genes and four stimuli and interactions at the inter- and intracellular levels.

Some of the nodes are assumed to be Boolean while a few others are multistates (*Low*, *Medium*, or *High*). Based on the value at the source of an edge being above or below some threshold, that edge is considered active or not, in the sense, that it will contribute to changing the value at the sink of that edge. When there is more than one incoming edge, the combinations of different active incoming edges and their thresholds are assembled into a logical function.

It is worth mentioning briefly that the algebraic model framework is well suited for the study of the important relationship between the structure and the dynamics of a biological network. In systems biology, the work of Uri Alon has drawn a lot of attention to this topic, summarized in his book (Alon, 2006). An important focus of Alon's work has been the effect of so-called network motifs, such as feed-forward loops, on dynamics. Another topic of study in systems biology has been the logical structure of gene regulatory and other networks. In the context of Boolean networks, significant work has been devoted to identifying features of Boolean functions that make them particularly suitable for the modeling of gene regulation and metabolism. As an example, we present here a summary of the work on so-called *nested canalyzing functions* (NCFs), a particular class of Boolean functions that appear frequently in systems biology models.

Biological systems in general and biochemical networks in particular are robust against noise and perturbations, and, at the same time, can evolve and adapt to different environments (Balleza *et al.*, 2008). Therefore, a realistic model of any such system must possess these properties, and hence the update functions for the network nodes cannot be arbitrary. Different classes of Boolean functions have been suggested as biologically relevant models of regulatory mechanisms: biologically meaningful functions (Raeymaekers, 2002), post functions (Shmulevich *et al.*, 2003b), and chain functions (Gat-Viks and Shamir, 2003). However, the class that has received the most attention is that of NCFs, introduced by Kauffman *et al.* (2004) for gene regulatory networks. We first give some precise definitions.

Definition 7.1

A Boolean function $g(x_1, \ldots, x_n)\colon \{0, 1\}^n \rightarrow \{0, 1\}^n$ is called *canalyzing* in the variable x_i with the input value a_i and output value b_i if x_i appears in g and

$$g(x_1, \ldots, x_{i-1}, a_i, x_{i+1}, \ldots, x_n) = b_i$$

for all inputs of all variables x_j and $j \neq i$.

The definition is reminiscent of the concept of "canalization" introduced by the geneticist (Waddington, 1942) to represent the ability of a genotype to produce the same phenotype regardless of environmental variability.

Definition 7.2

Let f be a Boolean function in n variables.

Let σ be a permutation of the set $\{1, \ldots, n\}$. The function f is a *nested canalyzing function (NCF) in the variable order* $x_{\sigma(1)}, \ldots, x_{\sigma(n)}$ with canalyzing input values $a_1, \ldots, a_n$ and canalyzed output values $b_1, \ldots, b_n$, if

$$\begin{aligned} f(x_1, \ldots, x_n) &= b_1 \quad \text{if } x_{\sigma(1)} = a_1, \\ &= b_2 \quad \text{if } x_{\sigma(1)} \neq a_1 \quad \text{and} \quad x_{\sigma(2)} = a_2, \\ &\ldots \\ &= b_{n-1} \, \text{if } x_{\sigma(1)} \neq a_1, \ldots, x_{\sigma(n-1)} \neq a_{n-1}, \text{ and } x_{\sigma(n)} = a_n, \\ &= b_n \quad \text{if } x_{\sigma(1)} \neq a_1, \ldots, x_{\sigma(n-1)} \neq a_{n-1}, \text{ and } x_{\sigma(n)} \neq a_n. \end{aligned}$$

The function f is nested canalyzing if it is nested canalyzing for some variable ordering σ.

As an example, the function $f(x,y,z) = x$ AND (NOT y) AND z is nested canalyzing in the variable order x,y,z with canalyzing values 0,1,0 and canalyzed values 0,0,0, respectively. However, the function $f(x,y,z,w) = x$ AND y AND (z OR w) is not nested canalyzing because, if $x = 1$ and $y = 1$, then the value of the function is not constant for any input values for either z or w.

One important characteristic of NCFs is that they exhibit a stabilizing effect on the dynamics of a system. That is, small perturbations of an initial state should not grow in time and must eventually end up in the same attractor of the initial state. The stability is typically measured using so-called Derrida plots which monitor the Hamming distance between a random initial state and its perturbed state as both evolve over time. If the Hamming distance decreases over time, the system is considered stable. The slope of the Derrida curve is used as a numerical measure of stability. Roughly speaking, the phase space of a stable system has few components and the limit cycle of each component is short.

Example 7.2

Consider the Boolean networks.

$$f = (x_4, x_3 \,\text{XOR}\, x_4, x_2 \,\text{XOR}\, x_4, x_1 \,\text{XOR}\, x_2 \,\text{XOR}\, x_3) : \{0,1\}^4 \to \{0,1\}^4;$$

$$g = (x_4, x_3 \,\text{AND}\, x_4, x_2 \,\text{AND}\, x_4, x_1 \,\text{AND}\, x_2 \,\text{AND}\, x_3) : \{0,1\}^4 \to \{0,1\}^4.$$

Notice that f and g have the same dependency graph and that g is a Boolean network constructed with NCFs while f is not. It is clear that the phase space of g in Fig. 7.5 has fewer components and much shorter limit cycles compared to the phase space of f in Fig. 7.6, and therefore g should be considered more stable than f.

In Kauffman *et al.* (2004), the authors studied the dynamics of nested canalyzing Boolean networks over a variety of dependency graphs. That is,

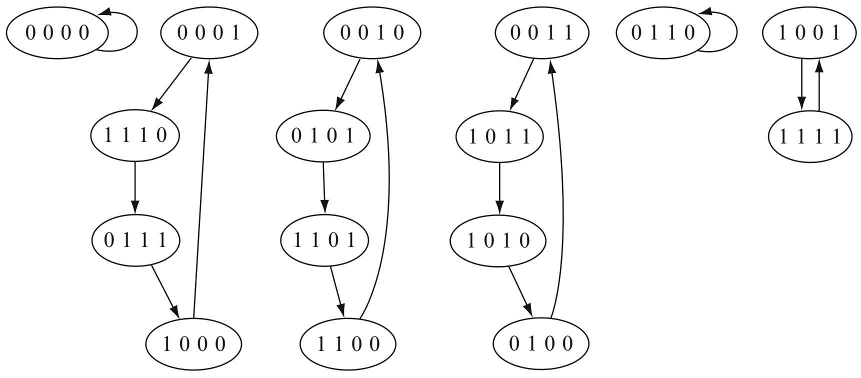


Figure 7.5 Phase space of the network *f* in Example 7.2.

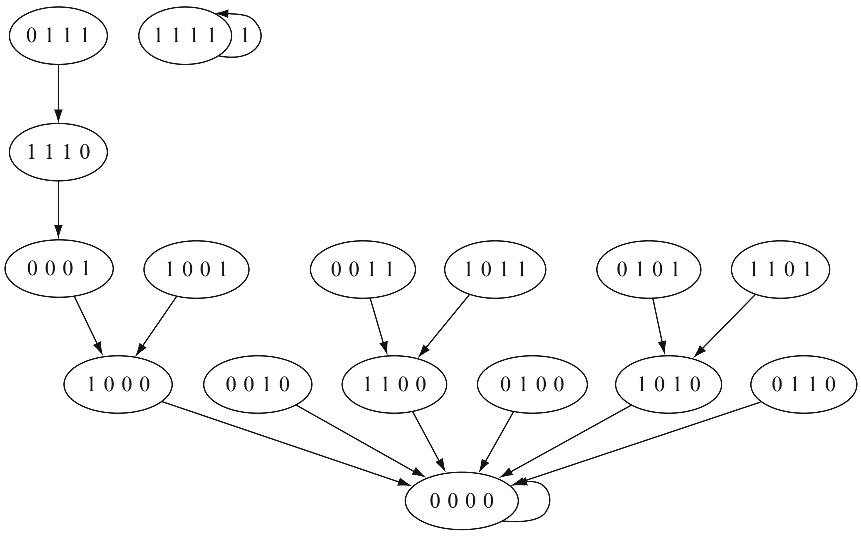


Figure 7.6 Phase space of network *g* in Example 7.2.

for a given random graph on n nodes, where the in-degree of each node is chosen at random between 0 and k, where $k < n + 1$, a NCF is assigned to each node in terms of the in-degree variables of that node. The dynamics of these networks were then analyzed and the stability measured using Derrida plots. It is shown that nested canalyzing networks are remarkably stable regardless of the in-degree distribution and that the stability increases as the average number of inputs of each node increases.

An extensive analysis of available biological data on gene regulations (about 150 genes) showed that 139 of them are regulated by canalyzing functions (Harris *et al.*, 2002; Nikolayewaa *et al.*, 2007). In Kauffman *et al.* (2004) and Nikolayewaa *et al.* (2007) it was shown that 133 of the 139 are, in fact, nested canalyzing.

Most published molecular networks are given in the form of a wiring diagram, or dependency graph, constructed from experiments and prior published knowledge. However, for most of the molecular species in the network, little knowledge, if any, could be deduced about their regulatory mechanisms, for instance in the gene transcription networks in yeast (Herrgard *et al.*, 2006) and *E. coli* (Barrett *et al.*, 2005).

Each one of these networks contains more than 1000 genes. Kauffman *et al.* (2003) investigated the effect of the topology of a subnetwork of the yeast transcriptional network where many of the transcriptional rules are not known. They generated ensembles of different models where all models have the same dependency graph. Their heuristic results imply that the dynamics of those models that used only NCFs were far more stable than the randomly generated models. Since it is already established that the yeast transcriptional network is stable, this suggests that the unknown interaction rules are very likely NCFs.

In Balleza *et al.* (2008), the whole transcriptional network of yeast, which has 3459 genes as well as the transcriptional networks of *E. coli* (1481 genes) and *B. subtillis* (840 genes) have been analyzed in a similar fashion, with similar findings. These heuristic and statistical results show that the class of NCFs is very important in systems biology. We showed in Jarrah *et al.* (2007) that this class is identical to the class of so-called unate cascade Boolean functions, which has been studied extensively in engineering and computer science. It was shown in Butler *et al.* (2005) that this class produces the binary decision diagrams with shortest average path length. Thus, a more detailed mathematical study of this class of functions has applications to problems in engineering as well.

In this section, we have shown that algebraic models, in particular Boolean network models, play an important role in systems biology, as models for a variety of molecular networks. They also are very useful in studying more theoretical questions, such as design principles for molecular networks. In Section 3, we will show how to construct such models from experimental data, in the top-down fashion discussed earlier.

3. Network Inference

In 2006 the first "Dialogue on Reverse-Engineering Assessment and Methods (DREAM)" workshop was held, supported in part by the NIH Roadmap Initiative. The rationale for this workshop is captured on the DREAM Web site (http://wiki.c2b2.columbia.edu/dream/index.php/

The_DREAM_Project): "The endless complexities of biological systems are orchestrated by intricate networks comprising thousands of interacting molecular species, including DNA, RNA, proteins, and smaller molecules. The goal of systems biology is to map these networks in ways that provide both fundamental understanding and new possibilities for therapy. However, although modern tools can provide rich data sets by simultaneously monitoring thousands of different types of molecules, discerning the nature of the underlying network from these observations—reverse engineering—remains a daunting challenge."

Traditionally, models of molecular regulatory systems in cells have been created *bottom-up*, where the model is constructed piece by piece by adding new components and characterizing their interactions with other molecules in the model. This process requires that the molecular interactions have been well characterized, usually through quantitative numerical values for kinetic parameters. Note that the construction of such models is biased toward molecular components that have already been associated with the phenomenon. Still, modeling can be of great help in this bottom-up process, by revealing whether the current knowledge about the system is able to replicate its *in vivo* behavior. This modeling approach is well suited to complement experimental approaches in biochemistry and molecular biology, since models thus created can serve to validate the mechanisms determined *in vitro* by attempting to simulate the behaviors of intact cells. While this approach has been dominant in cellular modeling, it does not scale very well to genome-wide studies, since it requires that proteins be purified and studied in isolation. This is not a practical endeavor due to its large scale, but especially because a large number of proteins act on small molecules that are not available in purified form, as would be required for *in vitro* studies.

With the completion of the human genome sequence and the accumulation of other fully sequenced genomes, research is moving away from the molecular biology paradigm to an approach characterized by large-scale molecular profiling and *in vivo* experiments (or, if not truly *in vivo*, at least, *in situ*, where experiments are carried out with intact cells). Technologies such as transcript profiling with microarrays, protein profiling with 2D gels and mass spectrometry, and metabolite profiling with chromatography and mass spectrometry, produce measurements that are large-scale characterizations of the state of the biological material probed. Other new large-scale technologies are also able to uncover groups of interacting molecules, delineating interaction networks. All these experimental methods are data rich, and it has been recognized (e.g., Brenner, 1997; Kell, 2004; Loomis and Sternberg, 1995) that modeling is necessary to transform such data into knowledge. A new modeling approach is needed for large-scale profiling experiments. Such a *top-down* approach starts with little knowledge about the system, capturing at first only a coarse-grained image of the system with only a few variables. Then, through iterations of simulation and experiment, the number of variables in the model is increased. At each iteration, novel experiments will be suggested by simulations of the model that provide data to improve it further, leading to a higher resolution in terms of mechanisms. While the processes of bottom-up and top-down modeling are distinct, both have as an objective the identification of molecular mechanisms responsible for cell behavior. Their main difference is that the construction of top-down models is biased by the data of the large-scale profiles, while bottom-up models are biased by the preexisting knowledge of particular molecules and mechanisms.

While top-down modeling makes use of genome-wide profiling data, it is conceptually very different from other genome-wide data analysis approaches. Top-down modeling needs data produced by experiments suitable for the approach. One should not expect that a random combination of arbitrary molecular snapshots would be of much use for the top-down modeling process. Sometimes they may serve some purpose (e.g., variable selection) but overall, top-down modeling requires perturbation experiments carried out with appropriate controls. In the face of modern experimental methods, the development of an effective top-down modeling strategy is crucial. Furthermore, we believe that a combination of top-down and bottom-up approaches will eventually have to be used. An example of a first step in this direction is the apoptosis model in Bentele *et al.* (2004).

A variety of different network inference methods have been proposed in recent years, using different modeling frameworks, requiring different types and quantities of input data, and providing varying amounts of information about the system to be modeled. There are fundamentally three pieces of information one wants to know about a molecular network: (i) its wiring diagram, that is, the causal dependencies among the network nodes, for example, gene activation or repression; (ii) the "logical" structure of the interactions between the nodes, for example, multiplicative or additive interaction of transcription factors; and (iii) the dynamics of the network, for example, the number of steady states. At one end of the model spectrum are statistical models that capture correlations among network variables. These models might be called *high level* (Ideker and Lauffenburger, 2003). The output of methods at the other end of the spectrum is a system of ODE which models network dynamics, provides a wiring diagram of variable dependencies as well as a mechanistic description of node interactions. In-between is a range of model types such as information-theory-based models, difference equations, Boolean networks, and multistate discrete models.

The literature on top-down modeling, or network inference, has grown considerably in the last few years, and we provide here a brief and necessarily incomplete review. The majority of new methods that have appeared utilize statistical tools. At the high-level end of the spectrum recent work has focused on the inference of relevance networks, first introduced in

Butte *et al.* (2000). Using pairwise correlations of gene expression profiles and appropriate threshold choices, an undirected network of connections is inferred. Partial correlations are considered in de la Fuente *et al.* (2004) for the same purpose. In Rice *et al.* (2005) conditional correlations using gene perturbations are used to assign directionality and functionality to the edges in the network and to reduce the number of indirect connections. Another modification is the use of time-delayed correlations in Li *et al.* (2006b) to improve inference. Using mutual information instead of correlation, together with information-theoretic tools, the ARACNE algorithm in Margolin *et al.* (2006) reports an improvement in eliminating indirect edges in the network.

Probably the largest part of the recent literature on statistical models is focused on the use of DBNs for reverse-engineering. Originally proposed in Friedman *et al.* (2000), the use of causal Bayesian network methods has evolved to focus on (DBNs), to avoid the limitation of not capturing feedback loops in the network. These can be thought of as a sequence in time of Bayesian networks that can represent feedback loops over time, despite the fact that each of the Bayesian networks is an acyclic directed graph. A variety of DBN algorithms and software packages have been published, see, for example, Beal *et al.* (2005), Dojer *et al.* (2006), Friedman (2004), Nariai *et al.* (2005), Pournara and Wernisch (2004), Yu *et al.* (2004), and Zou and Conzen (2005). Probably the largest challenge to DBN methods, as to all other methods, is the typically small sample sizes available for microarray data. One proposed way to meet this challenge is by bootstrapping, that is, the generation of synthetic data with a similar distribution as the experimental data; see, for example, Pe'er *et al.* (2001).

These methods all provide as output a wiring diagram, in which each edge represents a statistically significant relationship between the two network nodes it connects. Other approaches that result in a wiring diagram includes (Tringe *et al.*, 2004), building on prior work by Wagner (2001, 2004). If time course data are available it is useful to obtain a dynamic model of the network. There have been some recent results in this direction using Boolean network models, first introduced in Kauffman (1969). Each of the methods in Mehra *et al.* (2004), Kim *et al.* (2007), and Martin *et al.* (2007) either modifies or provides an alternative to the original Boolean inference methods in Liang *et al.* (1998), Akutsu *et al.* (1999, 2000a,b), and Ideker *et al.* (2000). Moving farther toward mechanistic models, an interesting network inference method resulting in a Markov chain model can be found in Ernst *et al.* (2007). Finally, methods using systems of differential equations include (Andrec *et al.*, 2005; Bansal *et al.*, 2006, 2007; Chang *et al.*, 2005; Deng *et al.*, 2005; Gadkar *et al.*, 2005; Gardner *et al.*, 2003; Kim *et al.*, 2007; Yeung *et al.*, 2002). Reverse-engineering methods have also been developed for the S-system formalism of Savageau (1991), which is a special case of ODE-based modeling, such as Kimura *et al.* (2005), Marino and Voit, (2006), and Thomas *et al.*, (2004).

Validating reverse-engineering methods and comparing their performance is very difficult at this time. Typically, each method is validated using data from simulated networks of different sizes and more or less realistic architecture. This is a crucial first step for any method, since it is important to measure its accuracy against a known network. The most common organism used for validation is yeast, with a wide collection of published data sets (very few of which are time course data). One of the key problems in comparing different methods is that they typically have different data requirements, ranging from time course data to steady-state measurements. Some methods require very specific perturbation experiments, whereas others need only a collection of single measurements. Some methods use continuous data, others, for instance most Bayesian network methods, require discretized data. Some methods take into account information such as binding site motifs. At present, there is no agreed-upon suite of test networks that might provide a more objective comparison. Nonetheless, comparisons are beginning to be made (Bansal *et al.*, 2007; Kremling *et al.*, 2004; Werhli *et al.*, 2006), but there too it is hard to correctly interpret the results. Most methods show some success with specialized data sets and particular organisms, but many theoretical and practical challenges remain in all cases. The stated goal of the DREAM effort mentioned in the beginning is to develop precisely such a set of benchmark data sets that can be used as a guide for method developers and a way to carry out more systematic comparisons.

We briefly describe two such methods here, one using parameter estimation for systems of differential equations as the principal tool, the other using statistical methods. A comparison of different methods, including these two, was done in Camacho *et al.* (2007). In the Section 4, we describe in detail a method that has as output either a wiring diagram or a Boolean network, using the interpolation of data by a Boolean network as the main tool. First, we describe a reverse-engineering method that uses multiple regression, proposed by Gardner *et al.* (2003), which is similar to de la Fuente and Mendes (2002) and Yeung *et al.* (2002). The method uses linear regression and requires data that are obtained by perturbing the variables of the network around a reference steady state. A crucial assumption of the method is that molecular networks are sparse, that is, each variable is regulated only by a few others. This method assumes no more than three regulatory inputs per node. The network is then recovered using multiple regression of the data. It estimates the coefficients in the Jacobian matrix of a generic system of linear differential equations representing the rates of change of the different variables. (Recall that the Jacobian matrix of a linear system of differential equations has as entry in position (i, j) the coefficient of the variable x_i in the equation for variable x_j.) The assumption that the wiring diagram of the network is sparse translates into the assumption that only a few of the entries of this matrix are different from 0.

The second method, originally published in Hartemink *et al.* (2002), uses the framework of so-called *DBNs*, a type of statistical model that gives as output a directed graph depicting causal dependency relations between the variables. These dependency relations are computed in terms of a time evolution of joint probability distributions on the variables, viewed as discrete random variables. The data required for this reverse-engineering method are time courses representing temporal responses to perturbations of the system from a steady state. The method has been implemented in the software package BANJO, described in Bernard and Hartemink (2005).

4. Reverse-Engineering of Discrete Models: An Example

4.1. Boolean networks: Deterministic and stochastic

As mentioned in the introduction, Boolean networks were first proposed as models for gene regulatory networks by Kauffman (1969), where a gene is considered either expressed (1) or not expressed (0), and the state of a gene is determined by the states of its immediate neighbors in the network. One interpretation of the different attractors (steady states, in particular) could be as the different phenotypes into which a cell will differentiate, starting from an arbitrary initialization. As there is evidence that gene regulation as well as metabolism could be stochastic, Boolean networks, which are deterministic, have been generalized to account for stochasticity. *Boolean networks with perturbations* (BNps) and *Probabilistic Boolean networks* (PBNs) have been developed for modeling noisy gene regulatory networks, see, for example, Akutsu *et al.* (2000a,b), Shmulevich *et al.* (2002, p. 225), and Yu *et al.* (2004).

Definition 7.3

A *Boolean network with perturbations* (BNp) is a Boolean network where, at each time step, the state of a randomly chosen node is flipped with some probability p. That is, if the current state of the network is $x = (x_1, \ldots, x_n)$, the next state y is determined as follows. A unit vector $e = (e_1, \ldots, e_n)$ is chosen at random where e_i is zero for all but one coordinate j for which $e_j = 1$. Then $y = f(x) + e$ with probability p and $y = f(x)$ with probability $1{-}p$. In particular, in a BNp, one and only one node could be perturbed with probability p at each time step. That is, the phase space of a BNp is a labeled directed graph, where the vertices are all possible states of the network, and the label of an edge (x,y) is the probability that $y = f(x) + e$

for some unit vector e. It is clear that the out-degree of each node x is n, where the edge (x,y) is labeled with $1-p$ if $y = f(x)$, and p if $y = f(x) + e$ for some unit vector e.

Definition 7.4

A PBN is a Boolean network in which each node in the network could possibly have more than one update function, in the form of a family of Boolean functions, together with a probability distribution. When it is time to update the state of a node, a function is chosen at random from the family of that node and is used to decide its new state. Namely, for each node x_i in the network, let $\{f, g, \ldots\}$ be the set of local functions, where the probability of choosing f is p, that of choosing g is q, etc. and $p + q + \ldots = 1$. Then the phase space of the network is a labeled directed graph with a directed edge (x,y), if, for all i, we have $y_i = f(x)$, for some local update function f for node i. The label on the edge is the product of the probabilities of each of the coordinates, where the probability of a coordinate is the sum of all probabilities p such that $y_i = f(x)$.

Example 7.3

Consider the network on three nodes in Fig. 7.7 below, and, let $f_{11} = x_1$ OR x_2, $f_{12} = x_2$ AND x_3, with probabilities 0.7, and 0.3, respectively. Suppose node 2 has only one local function, $f_2 = x_2$ OR x_3, and suppose node 3 has two functions: $f_{31} = x_1$ AND x_2 with probability 0.4, and $f_{32} = x_2$ with probability 0.6. The phase space of this network is depicted in Fig. 7.8.

Another way to introduce stochasticity into a deterministic model is by updating the network nodes asynchronously, and, at each time step, the order at which they are updated is chosen at random. These networks are

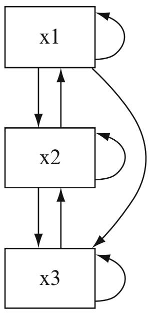


Figure 7.7 The wiring diagram of the probabilistic Boolean network in Example 7.3.

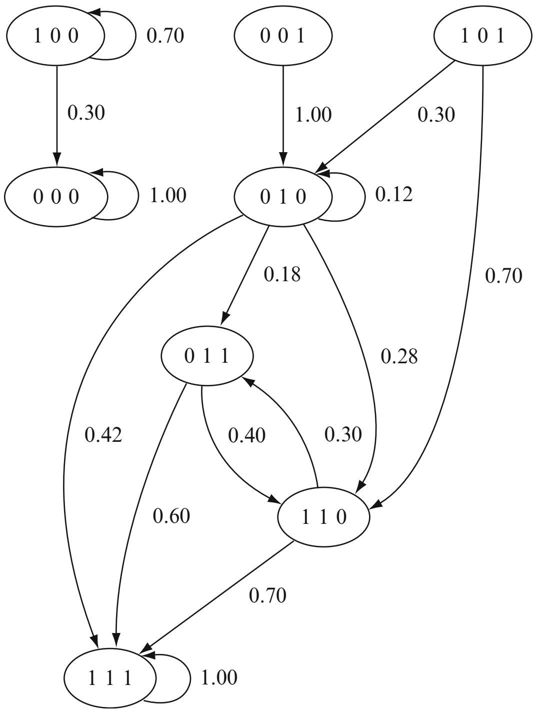


Figure 7.8 **The phase space of the probabilistic Boolean network in Example 7.3. Notice that there are three fixed points: 000 and 111 with probability 1, while the state 010 is a fixed point with probability 0.12. Furthermore, the two states 011 and 110 form a limit cycle with probability 0.12.**

clearly biologically relevant as they capture the order at which different events and processes take place and hence could change the outcome. In Chaves *et al.* (2005), the authors present a stochastic update-order model of the *segment polarity network* in Drosophila. The model captures aspects of this biological process that the original model in Albert and Othmer (2003) did not account for. One way to accomplish update-stochastic simulation of deterministic Boolean networks is to represent them as PBNs, where each node has two local update functions; the original one and the identity function, with appropriate choice of probabilities. This approach is implemented in the parameter estimation package *Polynome*, which we discuss below. Next, we briefly review some of the known network inference methods for BNp and PBNs.

It goes without saying that, to infer a Boolean network from experimental data sets, one has to start by assuming that each node in the network can only be in one of two states at any given time. In particular, the data used for inferring the network must also be binary, and hence the experimental data first have to be discretized into two qualitative states. There are several different methods for discretizing continuous data, see, for example Dimitrova *et al.* (2008). However, for the Boolean case, all methods come down to deciding the proper threshold that should be used to decide if a given molecular species is present (1) or absent (0). For DNA microarray data, this may be done by, for example, choosing a fold change above which a gene is considered upregulated compared to a control value. Or it may be done by inspection of a time course.

4.2. Inferring Boolean networks

As we described above, Boolean networks have emerged as a powerful framework for modeling and simulating gene regulatory networks. Therefore, it is natural to infer these networks from experimental data, and different methods have been proposed. Liang *et al.* (1998) pioneered this approach with the algorithm REVEAL, where information-theoretic principles are applied to reduce the search space. In Akutsu *et al.* (1999), the authors proposed a simple algorithm for identifying a Boolean network from a data set assuming that the in-degree of each node is relatively small. They discussed requirement on the data for such networks to exist. Recently, Martin *et al.* (2007) presented an algorithm for identifying all activation–inhibition Boolean networks (here each edge is either an activator or a strong inhibitor) from a given data set. Here, too, a small upper bound for the in-degree of each node is assumed. The dynamics of the identified Boolean networks are then used to shed light on the biological system.

Using the Boolean framework, Ideker *et al.* presented a method that identifies a minimal wiring diagram of the network from time course data (Ideker *et al.*, 2002). The network is minimal in the sense that each edge in the network is essential to reproduce the time course data.

4.3. Inferring stochastic Boolean networks

Deterministic Boolean network models seem inadequate for modeling some biological systems, as uncertainty is a prominent feature of many known systems. This is due either to hidden variables, intrinsic or extrinsic noise, or measurement noise. Different algorithms have been proposed for inferring stochastic Boolean networks within the framework of PBNs (Ching, 2005; Shmulevich *et al.*, 2002, 2003a) or BNps (Yu *et al.*, 2004; Akutsu *et al.*, 2000a,b).

Shmulevich and his collaborators developed an inference method that identifies a set of local functions of a given node using either time course data or a set of steady states (Shmulevich *et al.*, 2002). For each node x_i in the network, a set of local Boolean functions $X_i = \{f, g, \ldots\}$ is assigned with probabilities $P_i = \{p, q, \ldots\}$. The set X_i and P_i correspond to the highest coefficients of determination (CoD) of the node x_i relative to randomly chosen subsets of variables that could be possible input sets for node x_i. On the other hand, Yu *et al.* (2004) presented an algorithm for inferring a Boolean network with perturbations from steady-state data. Based on certain assumptions about the size of the basin of attraction for each observed state and lengths of transients, a matrix describing the transition between different attractors is computed.

Some of the algorithms mentioned above have been implemented as either C++ code, such as the algorithm of Akutsu *et al.*, or within other software packages, such as the algorithms of Shmulevich *et al.* (2002), which require the commercial software package Matlab. Furthermore, experimental data need to be Booleanized ahead of time before applying these algorithms. In Section 4.4, we describe the software package *Polynome* (Dimitrova *et al.*, 2009) that incorporates several different algorithms using tools from computational algebra and algebraic geometry. The software is capable of inferring wiring diagrams as well as deterministic and PBNs. Furthermore, the software can be used to simulate and explore the dynamics of the inferred network.

4.4. *Polynome*: Parameter estimation for Boolean models of biological networks

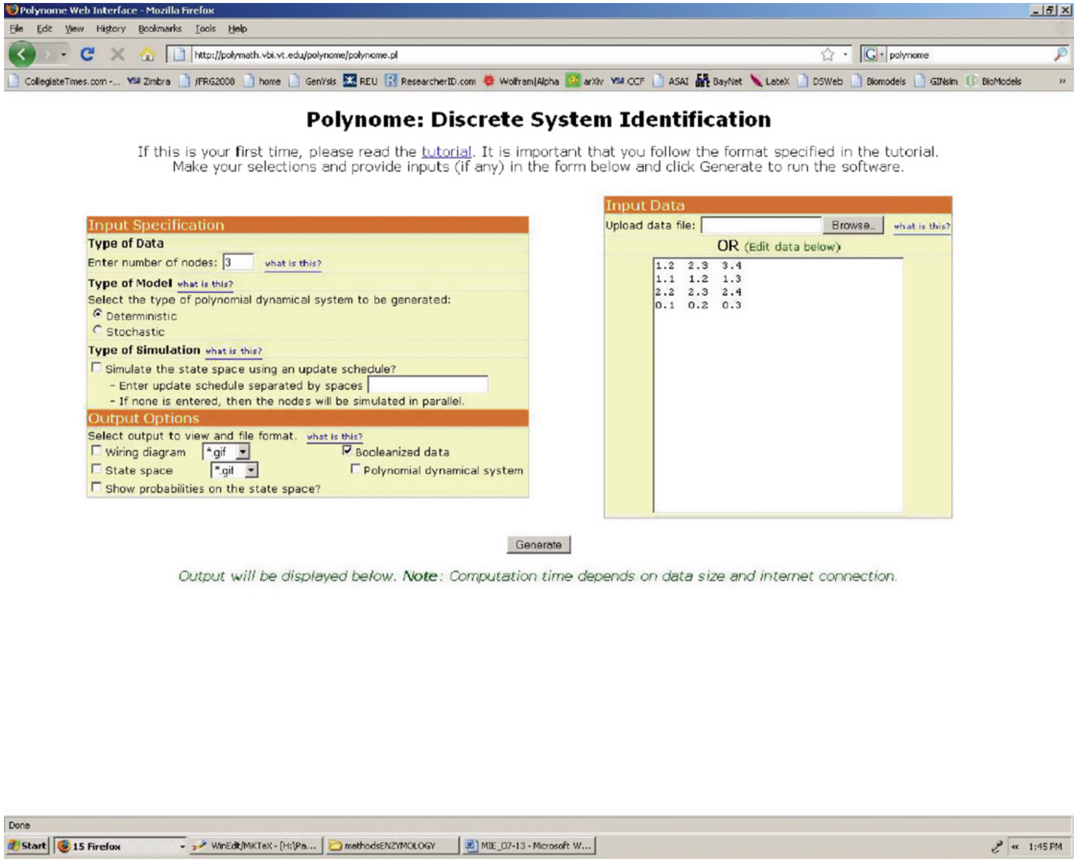
As described earlier, the goal of parameter estimation is to use experimental time course data to determine missing information in the description of a Boolean network model for the biological system from which the data were generated. This can be done with either partial or no prior information about the wiring diagram and dynamics of the system. *Polynome* will infer either a static wiring diagram alone or a dynamical model, with both deterministic and stochastic model choices. The software is available via a Web interface at http://polymath.vbi.vt.edu/polynome. Figure 7.9 shows a screenshot of the interface of *Polynome*.

The main idea behind the algebraic approach underlying *Polynome* is that any Boolean function can be written uniquely as a polynomial where the exponent of any variable is either 0 or 1 (hence the name). The dictionary is constructed from the basic correspondence:

$$x \text{ AND } y = xy, \quad x \text{ OR } y = x + y + xy, \quad \text{NOT} x = x + 1.$$

Therefore, any Boolean network f can be written as a polynomial dynamical system

$$f(x_1, \ldots, x_n) = f(x) = (f_1(x), \ldots, f_n(x)) : \{0,1\}^n \to \{0,1\}^n,$$



Polynome: Discrete System Identification

If this is your first time, please read the tutorial. It is important that you follow the format specified in the tutorial.
Make your selections and provide inputs (if any) in the form below and click Generate to run the software.

Input Specification

Type of Data

Enter number of nodes: 3

Type of Model

Select the type of polynomial dynamical system to be generated:
[✓ Deterministic]
Stochastic

Type of Simulation

Simulate the state space using an update schedule?
- Enter update schedule separated by spaces
- If none is entered, then the nodes will be simulated in parallel.

Output Options

Select output to view and file format.
Wiring diagram *gif
State space *gif
Show probabilities on the state space?
[✓ Booleanized data]
Polynomial dynamical system

Input Data

Upload data file:

OR (Edit data below)

```
1.2  2.3  3.4
1.1  1.2  1.3
2.2  2.3  2.4
0.1  0.2  0.3
```

Output will be displayed below. ***Note***: *Computation time depends on data size and internet connection.*

Polynome Web Interface - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://polymath.vbi.vt.edu/polynome/polynome.pl
what is this?
Browse...
Generate
Done
Start
1:45 PM

Figure 7.9 A screenshot of POLYNOME at http://polymath.vbi.vt.edu/polynome.

where the polynomial function f_i is used to compute the next state of node i in the network. For example, the Boolean network in Eq. (7.1) has the following polynomial form:

$$f(x_1, \ldots, x_4) = (x_1x_3, 1 + x_4 + x_2x_4, x_1, x_1 + x_2 + x_1x_2).$$

The Boolean network from Eq. (7.2) has the polynomial form:

$$\begin{aligned} f(x_1, \ldots, x_9) = ((1 + x_4)x_5, x_1, x_1, 1, (1 + x_6)(1 + x_7), \\ x_3x_8, x_6 + x_8 + x_9 + x_6x_8 + x_6x_9 + x_8x_9, x_2(x_8 + 1)), \end{aligned}$$

where $g = 1$ and $a = 1$, and $(x_1, \ldots, x_9) = (M, P, B, C, R, A, A_{\text{low}}, L, L_{\text{low}})$.

Studying Boolean networks as polynomial dynamical systems has many advantages, primarily that within the polynomial framework, a wide variety of algorithmic and theoretical tools from computer algebra and algebraic geometry can be applied. The remainder of the section will describe the parameter estimation algorithms implemented in *Polynome* and illustrate the software using the *lac* operon example described in detail above. This example is also used in Dimitrova *et al.* (2009) to validate the software.

Input. The user can input two kinds of information. The first kind consists of one or more time courses of experimental data. While several different data types are possible, we will focus here on DNA microarray data, for the sake of simplicity. If the network model to be estimated has nodes $x_1, \ldots, x_n$, then a data point consists of a vector $(a_1, \ldots, a_n)$ of measurements, one for each gene in the network. Since the model is Boolean, the first step is to discretize the input data into two states, 0 and 1. The user can provide a threshold to discriminate between the two states. As default algorithm, *Polynome* uses the algorithm in Dimitrova *et al.* (2008), which incorporates an information-theoretic criterion and is designed to preserve dynamic features of the continuous time series and to be robust to noise in the data.

The second type of input consists of biological information. This can take the form of known edges in the wiring diagram or known Boolean functions for some of the network nodes. Recall that an edge from node x_i to node x_j in the wiring diagram indicates that node x_i exerts causal influence on the regulation of node x_j. In other words, the variable x_i appears in the local update function f_j for x_j. In the absence of this type of information, the problem is equal to what is often called *reverse-engineering* of the network, that is, network inference using exclusively system-level data for the network.

To understand the algorithms, it is necessary to clarify the relationship between the input data and the networks produced by the software. The software produces networks that fit the given experimental data in the following sense. Suppose that the input consists of a time course $\mathbf{s}_1, \ldots, \mathbf{s}_t$, which each $\mathbf{s}_i \in \{0, 1\}^n$. Then, we say that a Boolean network f fits the given time course if $f(\mathbf{s}_i) = \mathbf{s}_{i+1}$ for all i.

Software output. There are five types of output the user can request. We briefly describe these and the algorithms used to obtain them.

A static wiring diagram of the network, that is, a directed graph with vertices the nodes of the network and edges indicating causal regulatory relationships. Since there is generally more than one such diagram for the given information (unless a complete wiring diagram is already provided as input), the user can request either a diagram with weights on the edges, indicating the probability of a particular edge being present, or a collection of top-scoring diagrams. The algorithm used for this purpose has been published in Jarrah *et al.* (2009). It computes all possible wiring diagrams of Boolean networks that fit the given data. The algorithm outputs only *minimal wiring diagrams*. Here, a wiring diagram is minimal if it is not possible to remove an edge and still obtain a wiring diagram of a model that fits the given data. In this sense, the output is similar to that in Ideker *et al.* (2002). However, the approach in Jarrah *et al.* (2009) is to encode the family of all wiring diagrams as an algebraic object, a certain monomial ideal, which has the advantage that ALL minimal wiring diagrams can be calculated, in contrast to a diagram produced by a heuristic search.

A deterministic dynamic model in the form of a Boolean network that fits the given data exactly and satisfies the constraints imposed by the input on the wiring diagram and the Boolean functions, using the algorithm described in Laubenbacher and Stigler (2004). This is done by first computing the set of all Boolean networks that fit the given data and the constraints. Using tools from computational algebra, this can be done by describing the entire set of models, that is, the entire parameter space, in a way similar to the description of the set of all solutions to a system of nonhomogeneous linear equations. As in the case of nonhomogeneous linear equations, if f and g are two Boolean networks that fit the given data set, that is, $f(\mathbf{s}_t) = \mathbf{s}_{t+1} = g(\mathbf{s}_t)$, then $(f–g)(\mathbf{s}_t) = 0$ for all t. Hence, all networks that fit the data can be found by finding one particular model f and adding to it any Boolean network g such $g(\mathbf{s}_t) = 0$ for all t. The space of all such g can be described by a type of basis that is similar to a vector space basis for the null space of a homogeneous system of linear equations.

A PBN that fits the given data. That is, the network has a family of update functions for each node, together with a probability distribution on the functions, as described earlier. This network has the property that for any choice of function at any update the network fits exactly the given data. The network is constructed using an algorithm that builds on the one described in Dimitrova *et al.* (2007).

A Boolean network that optimizes data fit and model complexity. In contrast to the previous two choices of output, this network does not necessarily fit the given data exactly but gives a network that is optimized with respect to both data fit and model complexity. This is a good model choice if the data are assumed to contain significant noise, since it reduces the tendency to overfit the data with a complex model. This option uses an evolutionary algorithm (Vera-Licona *et al.*, 2009) that is computationally intensive and is only feasible for small networks at this time.

A deterministic model that is simulated stochastically. This model is constructed by estimating Boolean functions that fit the data exactly, when simulated with synchronous update. But the network is then simulated using a stochastic update order. That is, the simulated network may not fit the given data exactly, but will have the same steady states as the synchronous model. The stochastic update order is obtained by representing the deterministic system as a PBN by adding the identify function to each node. At a given update, if the identity function is chosen, this represents a delay of the corresponding variable. Choosing an appropriate probability distribution, one can in this way simulate a stochastic sequential update order. The resulting phase space is a complete graph, with transition probabilities on the edges. This approach is also computationally very intensive, so this option is only feasible for small networks.

4.5. Example: Inferring the *lac* operon

In this section, we demonstrate some of the features of *Polynome* by applying it to data generated from the Boolean *lac* operon model in Eq. (7.1) above. That is, we take the approach that this model represents the biological system we want to construct a model of, based on "experimental" data generated directly from the model. This approach has the advantage that it is straightforward to estimate the performance of the estimation algorithm in this case.

The data in Table 7.1 include four time courses: all molecules are *high*, only R is *high*, only M is *high*, and only L and L_{low} are *high*.

Table 7.1 A set of time courses from the *lac* operon model, generated using Eq. (7.2)

All are *high*	R is *high*	M is *high*	L and L_{low} are *high*
1 1 1 1 1 1 1 1 1	0 0 0 0 1 0 0 0 0	1 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 1 1
0 1 1 1 0 1 1 1 1	0 0 0 1 1 0 0 0 1	0 1 1 1 1 0 0 0 1	0 0 0 1 1 0 1 0 1
1 0 0 1 0 1 1 1 1	0 0 0 1 1 0 1 0 1	0 0 0 1 1 0 1 1 1	0 0 0 1 0 0 1 0 1
1 1 1 1 0 0 1 0 1	0 0 0 1 0 0 1 0 1	0 0 0 1 0 0 1 0 1	1 0 0 1 0 0 1 0 1
1 1 1 1 0 0 1 1 1	1 0 0 1 0 0 1 0 1	1 0 0 1 0 0 1 0 1	1 1 1 1 0 0 1 0 1
1 1 1 1 0 1 1 1 1	1 1 1 1 0 0 1 0 1	1 1 1 1 0 0 1 0 1	1 1 1 1 0 0 1 1 1
1 1 1 1 0 1 1 1 1	1 1 1 1 0 0 1 1 1	1 1 1 1 0 0 1 1 1	1 1 1 1 0 1 1 1 1
	1 1 1 1 0 1 1 1 1	1 1 1 1 0 1 1 1 1	1 1 1 1 0 1 1 1 1
	1 1 1 1 0 1 1 1 1	1 1 1 1 0 1 1 1 1	

Table 7.2 shows a PBN (in polynomial form) inferred from the data in Table 7.1, using *Polynome*. Here, for each node, a list of update functions and their probabilities is given. The bold functions are the ones with probability higher than 0.1. (This threshold is provided by the user.) Notice that the true function $(1 + x_4)x_5$ for x_1 is in the list of inferred functions for x_1 with the second highest probability, the same as for the true function x_3x_8 for x_6. The inferred functions with highest probability for nodes 2,3,4 are the correct ones. In the case of node 7, the only inferred polynomial x_9 is clearly not the "true" function, which is $x_6 + x_8 + x_9$. However, it is important to remember that we are using four time courses involving only 26 states from the phase space of 512 states. Parameter estimation methods cannot recover information about the network that is missing from the data.

The phase space of this system has 512 states and many edges connecting them and so a visual inspection of the phase space graph is not possible. *Polynome* in this case provides a summary of the dynamics that includes the number of components, the number of limit cycles of each possible length as well as the stability of these cycles. Here, the stability of a cycle is the probability of remaining in that cycle. Table 7.3 shows that our inferred system in Table 7.2 has only one component which has the steady state (111101111), and its stability is 0.33. Note that the original Boolean *lac* operon model in Eq. (7.2) has only one component and the same steady state as the inferred model! The wiring diagram of the inferred network is shown in Fig. 7.10.

5. Discussion

Mathematical models have become an important tool in the repertoire of systems biologists wanting to understand the structure and dynamics of complex biological networks. Our focus has been on algebraic models and methods for constructing them from experimental time course data. For differential equations-based models, the standard approach to dealing with unknown model parameters is to estimate them by fitting the model to experimental data. The same approach is taken here to the estimate of unknown model parameters in an algebraic model. We have described several approaches to this problem in the literature. For one of these approaches, implemented in the software package *Polynome*, we have provided a detailed guide to how the software can be used with experimental data via a Web interface.

The extreme case of parameter estimation is the lack of any prior biological information, so that the network is to be inferred from experimental data alone. This is typically referred to as *reverse-engineering* or *network inference*. Many different approaches have been published to this "top-down"

Table 7.2 A probabilistic Boolean model inferred from the data in Table 7.1 using *Polynome*

f1 = {
x5⋆x8+x1⋆x5+x5+x2⋆x6+x2⋆x8+x6+x1⋆x7+x8+x1+1 #.0222222
x5+x7⋆x8+x1⋆x9+x8+x1+1 #.0222222
x5+x4+x1⋆x9+x9+x1+1 #.133333
x5+x1⋆x4+x4+x9+x1+1 #.0666667
x4⋆x5+x4 #.2
x5+x7⋆x8+x1⋆x7+x8+x1+1 #.0666667
x5⋆x7+x7 #.244444
x5⋆x9+x4 #.0444444
x2⋆x5+x5⋆x8+x5+x1⋆x4+x1⋆x2+x2+x1⋆x8+x8+x1+1 #.0222222
x5+x4⋆x8+x1⋆x4+x8+x1+1 #.0666667
x5⋆x9+x7⋆x8+x8+x9 #.0222222
x5+x4+x1⋆x7+x9+x1+1 #.0444444
x5⋆x6+x5⋆x8+x5⋆x9+x2⋆x6+x2⋆x8+x6+x8+x9 #.0222222
x5⋆x6+x5⋆x8+x5+x1⋆x4+x1⋆x6+x6+x1⋆x8+x8+x1+1 #.0222222
}
f2 = x1
f3 = x1
f4 = 1
f5 = x7+1
f6 = {
x2⋆x5+x1⋆x5+x2⋆x6+x1⋆x2+x6+x2+x1⋆x8 #.0222222
x5⋆x6+x5⋆x8+x3⋆x6+x4+x6+x8+x9 #.0222222
x3⋆x6+x1⋆x6+x1⋆x8 #.0444444
x5⋆x8+x5⋆x7+x1⋆x5+x5+x3⋆x6+x6+x1⋆x7+x8+x7+x1+1 #.0444444
x2⋆x8 #.377778
x3⋆x8 #.355556
x2⋆x6+x1⋆x6+x1⋆x8 #.0888889
x3⋆x6+x6+x1⋆x8+x3⋆x7+x1⋆x3 #.0222222
x5⋆x6+x5⋆x8+x5⋆x7+x5+x2⋆x6+x6+x1⋆x7+x8+x7+x1+1 #.0222222
}
f7 = {
x5⋆x8+x1⋆x5+x2⋆x6+x2⋆x8+x4+x6+x8 #.0222222
x5⋆x6+x5⋆x8+x4+x1⋆x6+x6+x1⋆x8+x8 #.0222222
x4⋆x8+x4+x8 #.111111
x5⋆x7+x5+x4+x1⋆x7+x7+x1+1 #.0444444
x9 #.666667
x4⋆x5+x5+x1⋆x4+x1+1 #.0666667
x4+x7⋆x8+x8 #.0444444
x2⋆x5+x5⋆x8+x4+x1⋆x2+x2+x1⋆x8+x8 #.0222222
}
f8 = {
x2 #.511111
x3 #.488889
}
f9 = 1

Table 7.3 The analysis of the phase space of the probabilistic Boolean network using local functions with probability more than 0.1 (the bold functions in Table 7.2). This is provided by *Polynome* (and the simulation software package DVD (http://dvd.vbi.vt.edu) in the case the phase space is too large to visualize

Analysis of the phase space [$m = 2$, $n = 9$]
Number of components 1
Number of fixed points 1
Fixed point, component size, stability
(1 1 1 1 0 1 1 1 1), 512, 0.33

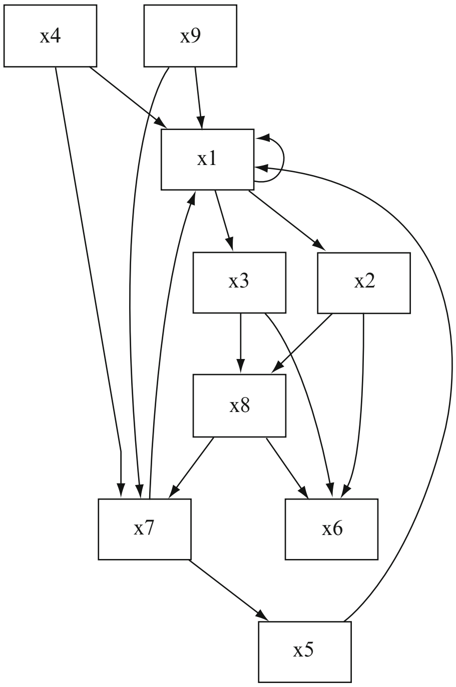


Figure 7.10 The wiring diagram of the inferred network in Table 7.2.

approach to modeling. There are still significant challenges ahead, arising primarily due to the lack of sufficiently large, appropriately collected time course data sets. Nonetheless, the field has advanced to the point where there are some first successes. It is our hope that this chapter has encourage the reader to try this approach to data modeling, whether using algebraic models, or others based on differential equations or statistics.

REFERENCES

Akutsu, T., Miyano, S., *et al.* (1999). Identification of genetic networks from a small number of gene expression patterns under the Boolean network model. *Pac. Symp. Biocomput.* 17–28.

Akutsu, T., Miyano, S., *et al.* (2000a). Algorithms for inferring qualitative models of biological networks. *Pac. Symp. Biocomput.* 293–304.

Akutsu, T., Miyano, S., *et al.* (2000b). Inferring qualitative relations in genetic networks and metabolic pathways. *Bioinformatics* **16**(8), 727–734.

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

Alon, U. (2006). An Introduction to Systems Biology: Design Principles of Biological Circuits. CRC Press, Boca Raton, FL.

Andrec, M., Kholodenko, B. N., *et al.* (2005). Inference of signaling and gene regulatory networks by steady-state perturbation experiments: Structure and accuracy. *J. Theor. Biol.* **232**(3), 427.

Balleza, E., Alvarez-Buylla, E. R., *et al.* (2008). Critical dynamics in gene regulatory networks: Examples from four kingdoms. *PLoS One* **3**(6), e2456.

Bansal, M., Gatta, G. D., *et al.* (2006). Inference of gene regulatory networks and compound mode of action from time course gene expression profiles. *Bioinformatics* **22**(7), 815–822.

Bansal, M., Belcastro, V., *et al.* (2007). How to infer gene networks from expression profiles. *Mol. Syst. Biol.* **3,** 78. doi:10.1038/msb4100120.

Barrett, C. B., Herring, C. D., *et al.* (2005). The global transcriptional regulatory network for metabolism in Escherichia coli exhibits few dominant functional states. *Proc. Natl. Acad. Sci. USA* **102**(52), 19103–19108.

Beal, M. J., Falciani, F., *et al.* (2005). A Bayesian approach to reconstructing genetic regulatory networks with hidden factors. *Bioinformatics* **21**(3), 349–356.

Bentele, M., Lavrik, I., *et al.* (2004). Mathematical modeling reveals threshold mechanism in CD95-induced apoptosis. *J. Cell Biol.* **166**(6), 839–851.

Bernard, A., and Hartemink, A. (2005). Informative structure priors: Joint learning of dynamic regulatory networks from multiple types of data. *Pac. Symp. Biocompu. conf. proceedings* pp. 459–470.

Brenner, S. (1997). Loose ends. *Curr. Biol.* 73.

Bruggemann, F. J., and Westerhoff, H. (2006). The nature of systems biology. *Trends Microbiol.* **15**(1), 45–50.

Butler, J. T., Tsutomu, S., *et al.* (2005). Average path length of binary decision diagrams. *IEEE Trans. Comput.* **54**(9), 1041–1053.

Butte, A. J., Tamayo, P., *et al.* (2000). Discovering functional relationships between RNA expression and chemotherapeutic susceptibility using relevance networks. *PNAS* **97**(22), 12182–12186.

Camacho, D., vera-Licona, P., *et al.* (2007). Comparison of reverse-engineering method using an in silico network. *Ann. NY Acad. Sci.* **1115,** 73–89.

Chang, W.-C., Li, C.-W., *et al.* (2005). Quantitative inference of dynamic regulatory pathways via microarray data. *BMC Bioinform.* **6**(1), 44.

Chaves, M., Albert, R., *et al.* (2005). Robustness and fragility of Boolean models for genetic regulatory networks. *J. Theor. Biol.* **235,** 431–449.

Ching, W. K., Ng, M. M., Fung, E. S., and Akutsu, T. (2005). On construction of stochastic genetic networks based on gene expression sequences. *Int. J. Neural Syst.* **15**(4), 297–310.

Davidich, M. I., and Bornholdt, S. (2007). Boolean network model predicts cell cycle sequence of fission yeast. *PLoS One* **3**(2), e1672.

deBoer, R. J. (2008). Theoretical biology. Undergraduate Course at Utrecht University, available at http://theory.bio.uu.nl/rdb/books/.

de la Fuente, A., and Mendes, P. (2002). Quantifying gene networks with regulatory strengths. *Mol. Biol. Rep.* **29**(1–2), 73–77.

de la Fuente, A., Bing, N., *et al.* (2004). Discovery of meaningful associations in genomic data using partial correlation coefficients. *Bioinformatics* **20**(18), 3565–3574.

Deng, X., Geng, H., *et al.* (2005). EXAMINE: A computational approach to reconstructing gene regulatory networks. *Biosystems* **81**(2), 125.

Dimitrova, E., Jarrah, A., *et al.* (2007). A Groebner-fan-based method for biochemical network modeling. Proceedings of the International Symposium on Symbolic and Algebraic Computation, Assoc Comp Mach, Waterloo, CA.

Dimitrova, E., Vera-Licona, P., *et al.* (2008). *Data discretization for reverse-engineering: A comparative study* (under review).

Dimitrova, E., Garcia-Puente, L., *et al.* (2009). Parameter estimation for Boolean models of biological networks. *Theor. Comp. Sci.* (in press).

Dojer, N., Gambin, A., *et al.* (2006). Applying dynamic Bayesian networks to perturbed gene expression data. *BMC Bioinform.* **7**(1), 249.

Ernst, J., Vainas, O., *et al.* (2007). Reconstructing dynamic regulatory maps. *Mol. Syst. Biol.* **3,** 74.

Espinosa-Soto, C., Padilla-Longoria, P., *et al.* (2004). A gene regulatory network model for cell-fate determination during Arabidopsis thaliana flower development that is robust and recovers experimental gene expression profiles. *Plant Cell* **16**(11), 1923–1939.

Faure, A., Naldi, A., *et al.* (2006). Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle. *Bioinformatics* **22**(14), 124–131.

Friedman, N. (2004). Inferring cellular networks using probabilistic graphical models. *Science* **303**(5659), 799–805.

Friedman, N., Linial, M., *et al.* (2000). Using Bayesian networks to analyze expression data. *J. Comput. Biol.* **7**(3–4), 601–620.

Gadkar, K., Gunawan, R., *et al.* (2005). Iterative approach to model identification of biological networks. *BMC Bioinform.* **6**(1), 155.

Gardner, T. S., di Bernardo, D., *et al.* (2003). Inferring genetic networks and identifying compound mode of action via expression profiling. *Science* **301**(5629), 102–105.

Gat-Viks, I., and Shamir, R. (2003). Chain functions and scoring functions in genetic networks. *Bioinformatics* **19,** 108–117.

Gonzalez, A., Chaouiya, C., *et al.* (2008). Logical modelling of the role of the Hh pathway in the patterning of the Drosophila wing disc. *Bioinformatics* **24**(234–240), 16.

Gupta, S., Bisht, S. S., *et al.* (2007). Boolean network analysis of a neurotransmitter signaling pathway. *J. Theor. Biol.* **244**(3), 463–469.

Harris, S. E., Sawhill, B. K., *et al.* (2002). A model of transcriptional regulatory networks based on biases in the observed regulation rules. *Complex Syst.* **7**(4), 23–40.

Hartemink, A., Gifford, D., *et al.* (2002). Bayesian methods for elucidating genetic regulatory networks. *IEEE Intel. Syst.* **17,** 37–43.

Herrgard, M. J., Lee, B. S., *et al.* (2006). Integrated analysis of regulatory and metabolic networks reveals novel regulatory mechanisms in Saccharomyces cerevisiae. *Genome Res.* **16,** 627–635.

Ideker, T. E., and Lauffenburger, D. (2003). Building with a scaffold: Emerging strategies for high- to low-level cellular modeling. *Trends Biotechnol.* **21**(6), 256–262.

Ideker, T. E., Thorsson, V., *et al.* (2000). Discovery of regulatory interactions through perturbation: Inference and experimental design. *Pac. Symp. Biocomput.* **5,** 305–316.

Ideker, T. E., Ozier, O., *et al.* (2002). Discovering regulatory and signalling circuits in molecular interaction networks. *Bioinformatics* **18**(Suppl. 1), S233–S240.

Jarrah, A., Raposa, B., *et al.* (2007). Nested canalyzing, unate cascade, and polynomial functions. *Physica D* **233**(2), 167–174.

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

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

Kauffman, S. A., Peterson, C., *et al.* (2003). Random Boolean network models and the yeast transcriptional network. *Proc. Natl. Acad. Sci. USA* **100**(25), 14796–14799.

Kauffman, S. A., Peterson, C., *et al.* (2004). Genetic networks with canalyzing Boolean rules are always stable. *Proc. Natl. Acad. Sci. USA* **101**(49), 17102–17107.

Kell, D. B. (2004). Metabolomics and systems biology: Making sense of the soup. *Curr. Opin. Microbiol.* **7**(3), 296–307.

Kim, J., Bates, D., *et al.* (2007). Least-squares methods for identifying biochemical regulatory networks from noisy measurements. *BMC Bioinform.* **8**(1), 8.

Kimura, S., Ide, K., *et al.* (2005). Inference of S-system models of genetic networks using a cooperative coevolutionary algorithm. *Bioinformatics* **21**(7), 1154–1163.

Kremling, A., Fischer, S., *et al.* (2004). A benchmark for methods in reverse engineering and model discrimination: Problem formulation and solutions. *Genome Res.* **14**(9), 1773–1785.

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

Laubenbacher, R., and Sturmfels, B. (2009). Computer algebra in systems biology. *Am. Math. Mon.* (in press).

Li, F., Long, T., *et al.* (2004). The yeast cell-cycle network is robustly designed. *Proc. Natl. Acad. Sci. USA* **101**(14), 4781–4786.

Li, S., Assman, S. M., *et al.* (2006a). Predicting essential components of signal transduction networks: A dynamic model of guard cell abscisic acid signaling. *PLoS Biol.* **4**(10), e312.

Li, X., Rao, S., *et al.* (2006b). Discovery of time-delayed gene regulatory networks based on temporal gene expression profiling. *BMC Bioinform.* **7**(1), 26.

Liang, S., Fuhrman, S., *et al.* (1998). REVEAL, a general reverse engineering algorithm for inference of genetic network architectures. *Pac. Symp Biocomput.* **3,** 18–29.

Loomis, W. F., and Sternberg, P. W. (1995). Genetic networks. *Science* **269**(5224), 649.

Margolin, A. A., Nemenman, I., *et al.* (2006). ARACNE: An algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinform.* **7**(Suppl. 1), S7.

Marino, S., and Voit, E. (2006). An automated procedure for the extraction of metabolic network information from time series data. *J. Bioinform. Comp. Biol.* **4**(3), 665–691.

Martin, S., Zhang, Z., *et al.* (2007). Boolean dynamics of genetic regulatory networks inferred from microarray time series data. *Bioinformatics* **23**(7), 866–874.

Mehra, S., Hu, W.-S., *et al.* (2004). A Boolean algorithm for reconstructing the structure of regulatory networks. *Metab. Eng.* **6**(4), 326.

Mendoza, L. (2006). A network model for the control of the differentiation process in Th cells. *Biosystems* **84,** 101–114.

Mendoza, L., and Alvarez-Buylla, E. R. (2000). Genetic regulation of root hair development in Arabidopsis thaliana: A network model. *J. Theor. Biol.* **204,** 311–326.

Nariai, N., Tamada, Y., *et al.* (2005). Estimating gene regulatory networks and protein-protein interactions of Saccharomyces cerevisiae from multiple genome-wide data. *Bioinformatics* **21**(Suppl. 2), ii206–ii212.

Nikolayewaa, S., Friedela, M., *et al.* (2007). Boolean networks with biologically relevant rules show ordered behavior. *Biosystems* **90**(1), 40–47.

Pe'er, D., Regev, A., *et al.* (2001). Inferring subnetworks from perturbed expression profiles. *Bioinformatics* **17**(Suppl. 1), S215–S224.

Pournara, I., and Wernisch, L. (2004). Reconstruction of gene networks using Bayesian learning and manipulation experiments. *Bioinformatics* **20**(17), 2934–2942.

Raeymaekers, L. (2002). Dynamics of Boolean networks controlled by biologically meaningful functions. *J. Theor. Biol.* **218**(3), 331–341.

Rice, J. J., Tu, Y., *et al.* (2005). Reconstructing biological networks using conditional correlation analysis. *Bioinformatics* **21**(6), 765–773.

Robeva, R., and Laubenbacher, R. (2009). Mathematical biology education: Beyond calculus. *Science* **325**(5940), 542–543.

Saez-Rodriguez, J., Simeoni, L., *et al.* (2007). A logical model provides insights into T cell receptor signaling. *PLoS Comp. Biol.* **3**(8), e163.

Samal, A., and Jain, S. (2008). The regulatory network of E. coli metabolism as a Boolean dynamical system exhibits both homeostasis and flexibility of response. *BMC Syst. Biol.* **2,** 21.

Sanchez, L., and Thieffry, D. (2001). A logical analysis of the Drosophila gap-gene system. *J. Theor. Biol.* **211,** 115–141.

Savageau, M. A. (1991). Biochemical systems theory: Operational differences among variant representations and their significance. *J. Theor. Biol.* **151**(4), 509.

Shmulevich, I., Dougherty, E. R., *et al.* (2002). Probabilistic Boolean networks: A rule-based uncertainty model for gene regulatory networks. *Bioinformatics* **18**(2), 261–274.

Shmulevich, I., Gluhovsky, I., *et al.* (2003a). Steady-state analysis of genetic regulatory networks modelled by probabilistic Boolean networks. *Comp. Funct. Genomics* **4**(6), 601–608.

Shmulevich, I., Lahdesmaki, H., *et al.* (2003b). The role of certain Post classes of Boolean network models of genetic networks. *Proc. Natl. Acad. Sci. USA* **100**(19), 10734–10739.

Stigler, B., and Veliz-Cuba, A. (2009). *Network topology as a driver of bistability in the lac operon* http://arxiv.org/abs/0807.3995.

Thomas, R., and D'Ari, R. (1989). Biological Feedback. CRC Press.

Thomas, R., Mehrotra, S., *et al.* (2004). A model-based optimization framework for the inference on gene regulatory networks from DNA array data. *Bioinformatics* **20**(17), 3221–3235.

Tringe, S., Wagner, A., *et al.* (2004). Enriching for direct regulatory targets in perturbed gene-expression profiles. *Genome Biol.* **5**(4), R29.

Vera-Licona, P., Jarrah, A., *et al.* (2009). *An optimization algorithm for the inference of biological networks* (in preparation).

Waddington, C. H. (1942). Canalisation of development and the inheritance of acquired characters. *Nature* **150,** 563–564.

Wagner, A. (2001). How to reconstruct a large genetic network from n gene perturbations in fewer than n(2) easy steps. *Bioinformatics* **17**(12), 1183–1197.

Wagner, A. (2004). Reconstructing pathways in large genetic networks from genetic perturbations. *J. Comput. Biol.* **11**(1), 53–60.

Werhli, A. V., Grzegorczyk, M., *et al.* (2006). Comparative evaluation of reverse engineering gene regulatory networks with relevance networks, graphical Gaussian models and Bayesian networks. *Bioinformatics* **22**(20), 2523–2531.

Yeung, M. K., Tegner, J., *et al.* (2002). Reverse engineering gene networks using singular value decomposition and robust regression. *Proc. Natl. Acad. Sci. USA* **99**(9), 6163–6168.

Yu, J., Smith, V. A., *et al.* (2004). Advances to Bayesian network inference for generating causal networks from observational biological data. *Bioinformatics* **20**(18), 3594–3603.

Zhang, R., Shah, M. V., *et al.* (2008). Network model of survival signaling in large granular lymphocyte leukemia. *Proc. Natl. Acad. Sci. USA* **105**(42), 16308–16313.

Zou, M., and Conzen, S. D. (2005). A new dynamic Bayesian network (DBN) approach for identifying gene regulatory networks from time course microarray data. *Bioinformatics* **21**(1), 71–79.