CHAPTER

1 Mechanisms of Gene Regulation: Boolean Network Models of the Lactose Operon in *Escherichia coli*

Raina Robeva*, Bessie Kirkwood* and Robin Davies†

*Department of Mathematical Sciences, Sweet Briar College, Sweet Briar, VA, USA
†Department of Biology, Sweet Briar College, Sweet Briar, VA, USA

1.1 INTRODUCTION

Understanding the mechanisms of gene expression is critically important for understanding the regulation of cellular behavior. Transcription of genes (messenger RNA (mRNA) synthesis), translation of mRNA (protein synthesis), degradation of mRNA and proteins, and protein–protein interactions are all involved in the control of gene expression where proteins may bind with DNA, with mRNA, and with other proteins, leading to complex networks of interactions. Cells have many more genes than they need to express under any given set of environmental conditions. Transcription and translation are energetically expensive processes, so cells should only express the genes required for the environmental circumstances in which they find themselves.

Certain genes, often termed *housekeeping genes*, are required to support basic life processes and are expressed continuously. The expression of many other genes, though, is contingent upon environmental or physiological factors. The expression of these *regulated genes* is controlled by the cell to ensure efficient use of its energy and materials. In this chapter we will focus on a set of regulated genes in *Escherichia coli* (*E. coli*), which are expressed only when lactose is the sole sugar available.

The most efficient point for controlling gene expression is at the level of transcription where the cell can control whether or not the gene is transcribed, at what rate it is transcribed, and under what conditions transcription occurs. Bacteria control transcription through the binding of specific proteins to their DNA. Some DNA binding proteins block transcription, while others cause the DNA to bend in a manner that facilitates the action of RNA polymerase. Still other proteins, the polymerase-associated sigma factors, confer sequence-specific binding ability on the RNA polymerase, allowing it to transcribe genes accurately. In the more complex eukaryotic cells of higher organisms, multiple proteins may bind to multiple sites in the DNA, and protein–protein interactions are also involved in the control of transcription.

Mathematical Concepts and Methods in Modern Biology. http://dx.doi.org/10.1016/B978-0-12-415780-4.00001-6

DNA sequences controlling transcription may be found at considerable distances from the gene to be controlled and may be brought to the vicinity of the RNA polymerase binding site (the promoter) by the bending of the DNA. This highly complex structure presents significant experimental challenges in the process of understanding and describing cellular behavior.

Mathematics provides a formal framework for organizing the overwhelming amounts of disparate experimental data and for developing models that reflect the dependencies between the system's components. Different types of mathematical models have been developed in an attempt to capture gene regulatory mechanisms and dynamics.

Various broad classifications are used in reference to such models. *Deterministic* models generate the exact same outcomes under a given set of initial conditions while in *stochastic* models the outcomes will differ due to inherent randomness. *Dynamic* models focus on the time-evolution of a system while *static* models do not consider time as part of the modeling framework. Among the dynamic models, *time-continuous* models utilize time as a continuous variable, while in *time-discrete* models time can only assume integer values. *Space-continuous* models refer to situations where the model variables can assume a continuum of values while in *space-discrete* models those variables can only assume values from a finite set. Space-continuous models of gene regulation are often constructed in the form of differential equations (in the case of continuous time) or difference equations (in the case of discrete time) and focus on the fine kinetics of biochemical reactions. We will refer to such models as *DE models*. Discrete-time models built from functions of finite-state variables are referred to as *algebraic models.*

In a DE model, all variables assume values from within biologically feasible ranges. Modelers usually need comprehensive knowledge of the interactions between variables, which may include detailed information of recognized control mechanisms, rates of production and degradation, minimal and maximal biologically relevant concentrations, and so on. In an algebraic model only values from a finite set are allowed. The special case of a *Boolean network* allows only two states, e.g., 0 and 1, generally representing the absence or presence of gene products in a model of gene regulation. In contrast to DE models, the information necessary to construct a Boolean model requires only a conceptual understanding of the causal links of dependency. Thus, in general, DE models are quantitative while Boolean models are qualitative in nature.

Historically, DE models have been the preferred type of mathematical models used in biology. This type of dynamical modeling has proved to be essential for problems in ecology, epidemiology, physiology, and endocrinology, among many others. Boolean models were first introduced to biology in 1969 to study the dynamic properties of gene regulatory networks [1]. They are appropriate in cases where network dynamics are determined by the logic of interactions rather than finely tuned kinetics, which may often be unknown.

In this chapter we present some of the fundamentals of creating Boolean network models for one of the simplest and best understood mechanisms of gene regulation: the *lactose* (*lac*) *operon* that controls the transport and metabolism of lactose

in *E. coli.* Since the seminal work by Jacob and Monod [2,3], the *lac* operon has become one of the most widely studied and best understood mechanisms of gene regulation. It has also been used as a test system for virtually every mathematical method of modeling gene regulation (see, e.g., [4–10]).

The rest of the chapter is organized as follows: In Section 1.2 we outline the basic structure of the *lac* operon. This section is meant only as a quick introduction and is not comprehensive in any way (see [11] for a more thorough introduction). Section 1.3 focuses on the construction and initial testing of a mathematical model with an emphasis on Boolean networks. A primer on Boolean algebra is included to make the chapter self-contained. We consider several Boolean models of the *lac* operon, then introduce and utilize the web-based suite of applications *Discrete Visualizer of Dynamics* [12] to perform initial testing and validation of the models. In Section 1.4 we turn to the question of determining the steady states (fixed points) of Boolean networks, casting the question in the broader context of polynomial dynamical systems and the use of Groebner bases for solving systems of polynomial equations. In Section 1.5 we point out directions for extending and generalizing the models and provide some concluding remarks regarding the possible use of this material in the undergraduate mathematics and biology curricula.

1.2 *E. COLI* AND THE *LAC* OPERON

E. coli is a short rod-shaped bacterium which is a common intestinal resident of mammals and birds. It has been the object of extensive study for decades. DNA replication, transcription, and translation were all elucidated in *E. coli* before they were studied in eukaryotic cells. Its physiology is well-understood and its entire gene sequence is known. For an overview of its importance to the study of genetics, see, for example, [13].

Since it lives in the intestines, any given *E. coli* bacterium's nutrition depends upon the diet of the animal whose digestive tract it inhabits. Digestion of the complex biomolecules in the foods consumed by the animal generally provides the bacterium with all of the simple biomolecules it needs. Digestion of starches provides the monosaccharide (simple sugar) glucose, digestion of proteins provides all of the amino acids, and, whenever milk is consumed by the host, *E. coli* is also exposed to lactose (milk sugar). Lactose is a disaccharide consisting of one glucose sugar linked to one galactose sugar. Galactose is a six carbon simple sugar which is an isomer of glucose. It has the same chemical formula as glucose but differs in the position of one hydroxyl group. Like glucose, galactose can be used as an energy source, although some additional enzymatic manipulation will be required.

In order to import sugars across their plasma membranes, cells must produce specific sugar *transport proteins* (e.g., glucose requires a glucose transporter; lactose requires a lactose transporter, and so on). Once the sugar has been imported into the cell, specific enzymes will act on the sugar, either to use it to make a required cellular molecule (in the constructive processes collectively called *anabolism*) or to break it

down in order to harvest its chemical energy in the form of ATP (in the destructive processes collectively called *catabolism*). Glucose is the preferred energy source for all cells, as it is used in *glycolysis*, the first and most fundamental energy-providing pathway of both anaerobic and aerobic cellular respiration.

The interactions between sugars and their transport proteins and the enzymes involved in their utilization within the cell are very specialized and cells must have the capacity to make specific proteins for every sugar they take up and use. However, it would not make sense for the *E. coli* to make the lactose transport protein if no lactose is present in its environment. It would also be inefficient to make the enzyme necessary to break lactose into glucose and galactose if there were no lactose inside the cell. If a cell had to make all of its proteins all of the time, it would be expending a lot of cellular energy in the making of proteins for which it has no use. Instead, cells have the ability to make certain of their proteins only when the environmental conditions warrant. Such proteins are called *inducible proteins*, because their synthesis is induced as a consequence of a certain cellular condition. The proteins needed for lactose uptake and utilization are inducible proteins, as they are only made if lactose is present and glucose, the preferred energy source, is not. Their concentrations increase 1000-fold under these conditions. Inducible genes belong to the group of regulated genes, in that they are only transcribed under certain specific conditions.

In *E. coli*, when lactose is the only sugar present, the transporter protein *lactose* (*lac*) *permease* and the enzyme *β-galactosidase* are both required in order to utilize it. *Lac* permease is a transmembrane protein which binds the disaccharide and brings it across the plasma membrane into the cytoplasm of the cell. The *lac* permease mediated transport of lactose through the cellular membrane is reversible, meaning that high levels of lactose inside the cell result in reverse transport of lactose to the outside of the cell. β-galactosidase catalyzes the hydrolysis of lactose into glucose and galactose. It also catalyzes the rearrangement of lactose into allolactose. These two proteins, lactose permease and β-galactosidase are produced by a tightly coordinated mechanism described by Francois Jacob and Jacques Monod and termed the *lac operon*. In the *lac* operon, due to the arrangement of its genes *LacZ*, *LacY*, and *LacA*, a single messenger RNA encodes both β-galactosidase and lactose permease, as well as a third enzyme, *transacetylase*, not involved in the metabolism of lactose (see Figure 1.1). Thus, when the *lac* promoter is active, all three proteins are produced.

Adjacent to the *lac* genes *lacZ*, *lacY*, and *lacA*, is the gene *LacI* that encodes a regulatory protein, the *lac repressor* (see Figure 1.2A). When there is no lactose present in the cell's environment, the *lac* repressor protein will bind to the operator, preventing the RNA polymerase from producing the *lac* mRNA. When lactose is present in the medium outside of the cell, a small amount of it will be transported into the cell by the few lactose permease molecules found in the plasma membrane. Once the lactose is inside the cell, it is converted to allolactose by the few β-galactosidase molecules present. Allolactose binds the *lac* repressor and causes it to undergo a conformational change, such that it can no longer bind to the operator (see Figure 1.2B). As a result, the RNA polymerase is able to read right through, produce the mRNA, and the three proteins are produced. This causes more lactose to be brought

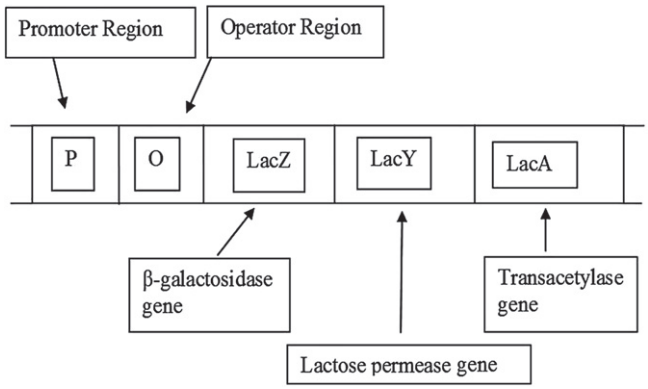

FIGURE 1.1

The structure of the *lac* operon. The RNA polymerase binds at the promoter and then proceeds along the *LacZ*, *LacY*, and *LacA* genes, copying them into a single mRNA which encodes the three proteins β-galactosidase, lactose permease, and transacetylase. The operator is a controlling region. See the text for more details.

into the cell by lactose permease and then broken down by β-galactosidase. We say that the operon is on. When all of the lactose is used up, there will be no allolactose, so the *lac* repressor protein will bind the operator and the synthesis of mRNA will stop. Levels of lactose permease and β-galactosidase will fall. The operon has turned itself off.

If both glucose and lactose are available to *E. coli*, glucose, the preferred energy source, is always used up before the bacterium begins to utilize the lactose. This is controlled by the mechanism of *catabolite repression*. Without it, the presence of lactose would cause allolactose to be made, preventing the *lac* repressor protein from binding the operator and allowing the transcription of the *lac* operon. In reality, the availability of glucose represses this process, and when both glucose and lactose are available, RNA polymerase is not able to initiate transcription sufficiently to ensure the levels of production of mRNA, β-galactosidase, and lactose permease reached under the condition of exclusive lactose availability. This failure of transcription initiation is called catabolite repression.

The mechanism of catabolite repression requires a DNA-binding protein, called CAP (for *catabolite activator protein*), which is the product of the *crp* (for cAMP receptor protein) gene. The CAP protein is capable of binding DNA only in the presence of cAMP. This small molecule is a common cellular signal or messenger and is produced by the enzyme adenylate cyclase. When glucose is present, adenylate cyclase does not produce cAMP. When there is no glucose, cAMP is produced. When cAMP binds CAP, the CAP-cAMP complex attaches to the DNA at the *lac* promoter and facilitates the attachment of RNA polymerase at the *lac* promoter. If there is no cAMP present, the CAP protein can't bind to the DNA, the RNA polymerase

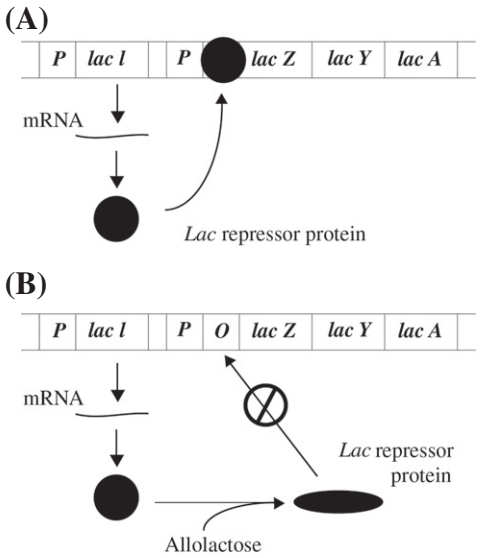

FIGURE 1.2

Panel A. The *lac* repressor protein in action. The *lac* repressor protein binds the *lac* operon at the operator, preventing transcription of the *lac* operon messenger RNA. The operon is Off. **Panel B.** Binding of allolactose to the *lac* repressor causes a conformational change in the repressor, preventing it from binding at the operator. Transcription of the *lac* operon messenger RNA can proceed. The Operon is On. Figure reproduced from *CBE-Life Sciences Education*, Vol. 9, Fall 2010, p. 233.

won't bind at the *lac* promoter, and the *lac* mRNA is not made. No mRNA means no β-galactosidase or lactose permease, and the operon will be off.[1] The key to the entire system is cAMP. If glucose is present, *E. coli* uses the glucose and the catabolism of glucose keeps cAMP levels low. After the glucose is used up, cAMP levels rise. The CAP protein binds cAMP, and the CAP-cAMP complex binds the DNA and facilitates the attachment of RNA polymerase. Figure 1.3 presents a schematic of the whole lac operon regulatory mechanism.

1.3 BOOLEAN NETWORK MODELS OF THE *LAC* OPERON

In this section we design some basic dynamical models of the *lac* operon mechanism, capable of reflecting its biological behavior. At the very minimum, a model should

[1]To be more exact, when both lactose and glucose are present, some small amounts of mRNA, β-galactosidase, and lactose permease will always be made (since the repressor protein will be blocked from binding to the operator site). However, those levels will be thousands of times lower than they would be in the absence of glucose.

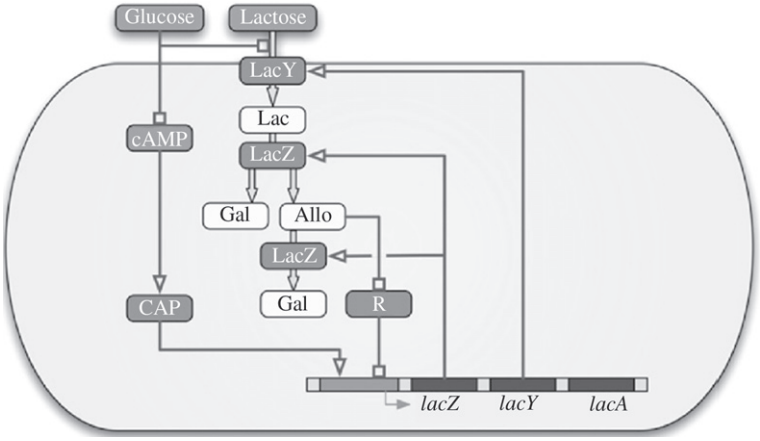

FIGURE 1.3

Schematic of the *lac* operon regulatory mechanism (from Santillan at al. [14]) Protein LacY is a permease that transports external lactose into the cell. Protein LacZ polymerizes into a homotetramer named β-galactosidase. This enzyme transforms internal lactose (Lac) to allolactose (Allo) or to glucose and galactose (Gal). It also converts allolactose to glucose and galactose. Allolactose can bind to the repressor (R) inhibiting it. When not bound by allolactose, R can bind to a specific site upstream of the operon structural genes and thus prevent transcription initiation. External glucose inhibits the production of cAMP that, when bound to the CRP protein (also known as CAP) to form the CAP-cAMP complex, acts as an activator of the *lac* operon. External glucose also inhibits lactose uptake by permease proteins. Reprinted from *Biophysics Journal*, Vol. 92, M. Santillan, M. C. Mackey, and E. S. Zeron, Origin of bistability in the lac operon 3830 - 3842, Copyright (2007), with permission from Elsevier.

imply that when lactose is absent from the medium, the operon is off, and that when lactose is present but glucose is not, the operon is on. We consider Boolean models with various levels of complexity and compare the results. The material can be used as a gentle entry into mathematical modeling, especially for students without Calculus background (see Section 1.5 for more details).

1.3.1 Identifying the Model Variables and Parameters

The modeling process begins with careful examination of the major interactions and components of the system, as depicted in Figure 1.3, and with selecting the model variables and parameters. The model variables are generally chosen to represent the major dynamic elements of the system (quantities that change with time) while the parameters correspond to static descriptors. Different decisions regarding the exclusion or inclusion of any given component or part of the system will lead to different models. The next step is to define a *wiring diagram* (sometimes also called a

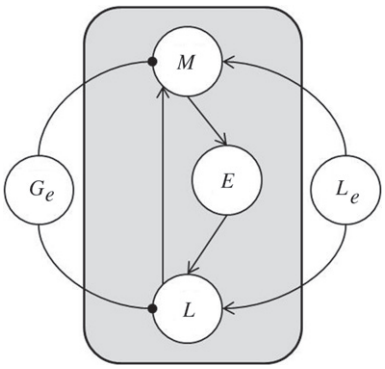

FIGURE 1.4

Wiring diagram for the minimal model. *E* denotes the *LacZ* polypeptide, *M* – the mRNA, *L*- internal lactose. L_e and G_e denote external lactose and glucose, respectively. The nodes in the shaded rectangle represent the model variables while the outside nodes represent the model parameters. Directed links represent influences between the variables: A positive influence is indicated by an arrow; a negative influence is depicted by a circle. Self-regulating links (for nodes that may influence themselves) are not pictured.

dependency graph) for the model that reflects the network topology; that is, it depicts the dependencies between variables and parameters. A wiring diagram is a directed graph in which each node represents a variable or a parameter for the model and the directed links depict influential interactions: if x and y are two nodes of the graph, a directed link from x to y indicates that the quantity x affects the quantity y. Some wiring diagrams contain additional information about the types of interactions between the model components: a positive influence is usually indicated by an arrow, while a negative influence may be depicted by a square (as in Figure 1.3) or by a circle (see Figure 1.4). In many ways, the diagram in Figure 1.3 is similar to a wiring diagram except that it also includes a cartoon of the structural arrangement of the *lac* genes with the promoter and operator regions, and some of the system's elements appear more than once (e.g., the LacZ protein).

It is always best to begin with a simple starter model with a small number of variables and parameters, from which to expand, if necessary. The models of the *lac* operon we will discuss initially follow a "minimal model" approach for choosing variables and parameters after Santillan et al. [14]. The model does not consider the CAP-cAMP positive control mechanism, which is essentially an amplifier for the transcription process. We focus on the following remaining elements (the notations in the parentheses are the mathematical names we will use from now on to denote their concentrations): mRNA (M), β-galactosidase (B), *lac* permease (P), intracellular lactose (L), allolactose (A), external lactose (L_e), and external glucose (G_e). Due to the fact that external conditions for the cell change slowly compared to the lifespan of *E. coli*, the concentrations L_e and G_e remain relatively unchanged with time. We will assume that they are constants and include them in the set of model parameters.

The other quantities (*M, B, P, L*, and *A*) will be assumed to vary with time. Some of these variables, however, exhibit related dynamics due to similarities in their underlying biochemical structures and mechanisms of production. Thus, as described next, we can further reduce the number of model variables based on such known dependences.

β-galactosidase is a homo-tetramer made up of four identical LacZ polypeptides. If we denote the LacZ polypeptide by E, the following holds for the concentrations of β-galactosidase and LacZ: $B = E/4$. Further, since the translation rate of the LacY transcript can be assumed to be the same as the rate for the LacZ transcript, the following holds for the concentration of permease: $P = E$. Finally, we can assume that the concentrations of internal lactose (L) and allolactose (A) are proportional, that is $A = pL$, where p indicates the fraction of lactose converted into allolactose and can be determined experimentally. Thus, in our first "minimal" model, we will consider only three variables—*M, E, L*, and two parameters—L_e and G_e. Knowing the variables *M, E,* and *L* would allow us to determine the values of *P, B,* and *A* from the equations $B = E/4$, $P = E$, and $A = pL$. The corresponding dependency graph is depicted in Figure 1.4.

The choice for variables and parameters discussed here is just one possibility among many others. The model by Yildrim and Mackey for instance [15] is based on assumptions leading to a wiring diagram including five nodes corresponding (in our notation) to the variables *M, B, P, L,* and *A*, and a node for external lactose as a parameter. In [16], the authors consider a reduction of this five-variable model to a network of three nodes: *M, B,* and *L*. These models, together with their Boolean approximations, will be introduced and discussed in Chapter 2 of this volume. Later in this chapter we consider a Boolean model of the *lac* operon with nine variables and two parameters that includes the mechanism of catabolite repression.

Once the model variables have been identified, the decision on the type of mathematical model should be made. As mentioned earlier, various types of mathematical models can be developed (including DE, algebraic, stochastic, and simulation models among others) from the same wiring diagram. In this section, we will focus on developing a Boolean network model.

1.3.2 Boolean Network Models

Boolean variables and Boolean expressions. Boolean models allow only two states, e.g., 0 and 1, indicating the absence or presence of the components represented by the nodes of the wiring diagram. Since in biology trace amounts of various substances may be present at all times, "absence" usually stands for concentrations lower than a certain threshold value separating higher concentrations from the baseline, and "presence" is interpreted as concentrations higher than this threshold. It may appear that because chemical concentrations span a continuous range of values, choosing a single threshold cutoff may not be appropriate—two concentration values may be very close numerically with one of them falling above the threshold and the other one falling below it. Although this may be a legitimate concern in general, it would rarely apply to a model of gene regulation. When the gene is expressed, the concentration

Table 1.1 Tables of values for the basic Boolean operations: Logical AND ($\wedge$), logical OR ($\vee$) and NOT (an overbar identifies the negation of the Boolean variable underneath).

Logical AND: $z = x \wedge y$			**Logical OR: $z = x \vee y$**			**Logical NOT: $z = \overline{y}$**	
Input		**Output**	**Input**		**Output**	**Input**	**Output**
x	y	z	x	y	z	y	z
0	0	0	0	0	0	0	1
0	1	0	0	1	1	1	0
1	0	0	1	0	1		
1	1	1	1	1	1		

of the protein it makes would be thousands of times higher than the trace amounts that are present when the gene is silent, with a clear distinction between present and absent (1 or 0). Throughout this chapter, we will assume without further mention that a Boolean value of 0 indicates a concentration near the basal level while a Boolean value of 1 signifies a markedly higher concentration.

In Boolean models, the dynamical evolution of the system is described using Boolean functions defined in terms of the model variables and the logical operators AND (denoted by the symbol $\wedge$), OR (denoted by the symbol $\vee$), and NOT (denoted by an overbar on the variable it negates). The tables of values for these Boolean operators are presented in Table 1.1.

In the context of modeling network dynamics, it is often useful to consider the following intuitive interpretation for the operations AND and OR: if the components x and y of the system influence (control) a third component z, then $z = x \wedge y$ means that x and y need to be simultaneously present (have values 1) to affect z; $z = x \vee y$ means that x and y influence z independently and z is affected when x OR y (or both) are present. In the absence of any parentheses, the order of precedence is this: logical NOT has highest precedence, followed by the logical AND, followed by the logical OR.

Sometimes, if we know the value for one of the operands in the AND / OR operations, we may not need to evaluate the other operand to be able to determine the value for the operation. For instance, if $\mathrm{A} = 0, \mathrm{C} = \mathrm{A} \wedge \mathrm{B} = 0$ regardless of the value of B. In general, if at least one of the operands of the AND operation is zero, the resulting value for the operation is zero no matter what the value of the other operand is. In a similar way, if one of the operands of the OR operation has value 1, the value for the operation is 1 regardless of the value of the other operand. This is known as *short-circuit evaluation.*

Example 1.1. Assume the Boolean variables A, B, C, and D have values $A = 0$, $B = 1, C = 1, D = 1$. Determine the value of the Boolean expression $(\overline{D} \vee B) \wedge \overline{A} \wedge C \vee B$.

The expression in the parentheses will be evaluated first and in order to do this, $\overline{D}$ must be computed because NOT has higher precedence than OR. Since $D = 1$, the value of $\overline{D} = 0$. Since $B = 1$, the value of $(\overline{D} \vee B) = 1$. Next, following the rules

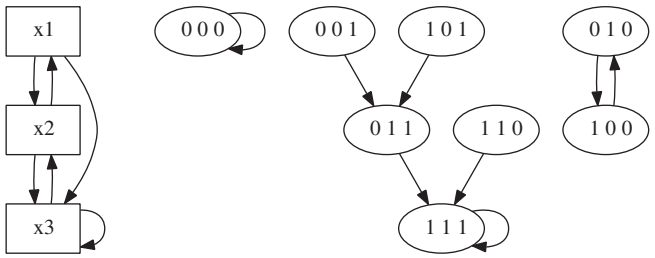

FIGURE 1.5

Wiring diagram and the state space transition diagram for the Boolean dynamical system in Example 1.2 given by Eqs. (1.2). Graphs produced with DVD [12].

of precedence, we compute $\overline{A} = 1$; $(\overline{D} \vee B) \wedge \overline{A} = 1$; $(\overline{D} \vee B) \wedge \overline{A} \wedge C = 1$; and finally $(\overline{D} \vee B) \wedge \overline{A} \wedge C \vee B = 1$.

Exercise 1.1. Consider the Boolean variables *w, x, y, z* and the Boolean function *f* of those variables defined by the expression $f(w, x, y, z) = (x \vee y) \wedge (\overline{w \wedge z})$. Determine the values of: (1) $f(1, 1, 0, 1)$; (2) $f(0, 1, 0, 1)$; (3) $f(1, 1, 1, 1)$; (4) $f(0, 0, 0, 0)$. $\triangledown$

Modeling the dynamics of a Boolean network: Transition functions. Assume the wiring diagram contains n Boolean model variables (also called *nodes*) denoted by $x_1, x_2, \ldots, x_n$. Each of these n variables can take a value 0 or 1, resulting in a set of n-tuples $V = \{0, 1\}^n = \{(x_1, x_2, \ldots, x_n) | x_i \in \{0, 1\}, i = 1, 2, \ldots, n\}$, containing 2^n elements, representing all possible states for the model variables. Time is discrete and can only take values $t = 0, 1, 2, \ldots$ The values of the nodes $x_1, x_2, \ldots, x_n$ change with time and we write $x_i = x_i(t)$ for the value of the variable x_i at time t. Thus, at each time step t, the system is represented by a binary n-tuple from the set V, where each component stands for the value of the respective Boolean variable at time t. The rules for transitioning between the states at each time step are described by n functions $f_{x_i}, i = 1, 2, \ldots, n$, one function for each model variable. The Boolean expression defining the function f_{x_i}, written in terms of the Boolean operations AND, OR, and NOT, describes in what way the values of the variables $x_1, x_2, \ldots, x_n$ at time t affect the value of the variable x_i at time $t+1$. Thus, for any value of $t = 0, 1, 2, \ldots$, the system "update" for variable x_i from time t to time $t + 1$ is determined by $x_i(t + 1) = f_{x_i}(x_1(t), x_2(t), \ldots x_n(t)), i = 1, 2, \ldots, n$. The functions $f_{x_i}, i = 1, 2, \ldots, n$, are the *transition functions* (also called *update rules* or *rules*) for the model. The updates we will be using here are synchronous, meaning that all variables x_i are computed first for time t and then used to evaluate the functions f_{xi}. If we write $x = (x_1, x_2, \ldots, x_n)$ and $f(x) = (f_{x_1}(x), \ldots, f_{x_n}(x))$, the *state space* of the model is defined by the directed graph $\{V, T\}$, where the set $T = \{(x, f(x)) | x \in V\}$ represents the set of edges.

Example 1.2. Assume we have a Boolean network containing three Boolean variables x_1, x_2 and x_3, assume that the wiring diagram for the network is depicted in

Figure 1.5a, and that the transition functions are given by

$$\begin{aligned} x_1(t+1) &= f_{x_1}(x_1(t), x_2(t), x_3(t)) = x_2(t) \\ x_2(t+1) &= f_{x_2}(x_1(t), x_2(t), x_3(t)) = x_1(t) \vee x_3(t) \\ x_3(t+1) &= f_{x_3}(x_1(t), x_2(t), x_3(t)) = x_1(t) \wedge x_2(t) \vee x_3(t). \end{aligned} \tag{1.1}$$

With the understanding that the variables on the right-hand side are always evaluated at time t and that the variables on the left-hand side stand for the values at time $t+1$ we can simplify the notation by removing t and $t+1$ from the equations:

$$\begin{aligned} x_1 &= f_{x_1}(x_1, x_2, x_3) = x_2 \\ x_2 &= f_{x_2}(x_1, x_2, x_3) = x_1 \vee x_3 \\ x_3 &= f_{x_3}(x_1, x_2, x_3) = x_1 \wedge x_2 \vee x_3. \end{aligned} \tag{1.2}$$

Assume next that at time $t = 0$, the values of the variables x_1, x_2, and x_3 are $x_1 = 0, x_2 = 0, x_3 = 1$. The values of the model variables at time $t = 0$ are often referred to as *initial conditions*. Using these values to evaluate the transition functions above, we obtain the values of the variables at time $t = 1$:

$$\begin{aligned} x_1 &= f_{x_1}(x_1, x_2, x_3) = f_{x_1}(0, 0, 1) = 0 \\ x_2 &= f_{x_2}(x_1, x_2, x_3) = f_{x_2}(0, 0, 1) = 0 \vee 1 = 1 \\ x_3 &= f_{x_3}(x_1, x_2, x_3) = f_{x_3}(0, 0, 1) = 0 \wedge 0 \vee 1 = 1. \end{aligned}$$

Now take the new values $x_1 = 0, x_2 = 1, x_3 = 1$. These values are used to evaluate the transition functions f_{x_i} again, producing, for time $t = 2$, the values $x_1 = 1, x_2 = 1, x_3 = 1$. Plugging these values into the functions again, returns the same values $x_1 = 1, x_2 = 1, x_3 = 1$ which correspond to the values of the variables at time $t = 3$. Thus, for any future values of t, the values of the model variables will remain $x_1 = 1, x_2 = 1, x_3 = 1$. We say that we have computed the *trajectory* of the state $(0, 0, 1) : (0, 0, 1) \to (0, 1, 1) \to (1, 1, 1)$. We say that (1,1,1) is a *fixed point* for the Boolean network. Similar considerations show that (0,0,0) is also a fixed point (see Exercise 1.4). Using different starting values for the Boolean variables will lead to different trajectories. For instance, the initial state (0,1,0) generates the following repeating pattern: $(0, 1, 0) \to (1, 0, 0) \to (0, 1, 0) \to (1, 0, 0) \to (0, 1, 0) \to (1, 0, 0) \ldots$ (see Exercise 1.5). We say that the states $(0, 1, 0)$ and $(1, 0, 0)$ form a *cycle of length two*. Fixed points can be considered to be cycles of length one.

Since the system is composed of three variables each of which can take values 0 and 1, there are $2^3 = 8$ possible states for the system (see Exercise 1.2). Computing the trajectories initiating from each of these eight initial states and plotting them as in Figure 1.5b visualizes all possible trajectories for the Boolean network, containing all possible transitions between the eight states and, thus, forming the *state space transition diagram* of the Boolean network. Clearly, for a much larger number of variables, computing the trajectories by hand would be impossible and the use of appropriate software is essential. The graphs in Figure 1.5 were created with the

freely-available web application *Discrete Visualizer of Dynamics* (*DVD*) (available at http://dvd.vbi.vt.edu). More details about DVD are provided in Section 1.3.5 below.

Mathematically, the diagram in Figure 1.5b depicts another directed graph. This time, however, the directed graph represents the state-to-state transitions of the system. The edges of the directed graph correspond to transitions between the states as time evolves. Fixed points are recognized as the states that transition to themselves (with edges pointing back to the same state in the diagram). The state space diagram in Figure 1.5b indicates that the three-variable system with wiring diagram given in Figure 1.5a and transition functions described by Eqs. (1.2) has two fixed points: (0, 0, 0) and (1, 1, 1) and a limit cycle of length two. It also shows that the state space graph has three *disconnected components*: The first is {(0, 0, 0)} containing only the fixed point (0, 0, 0), the second is {(0, 0, 1), (1, 0, 1), (0, 1, 1), (1, 1, 0), (1, 1, 1)}, and the third one is {(0, 1, 0), (1, 0, 0)}. Any trajectories originating from points in the second component will terminate at the fixed point (1, 1, 1). Any trajectory originating from a point in the third component will oscillate, alternating between the two states of the cycle.

Example 1.3. Consider a Boolean network defined by the following set of transition functions describing the interaction dynamics of the three variables x_1, x_2, and x_3:

$$\begin{aligned} x_1(t+1) &= f_{x_1}(x_1(t), x_2(t), x_3(t)) = x_3(t) \\ x_2(t+1) &= f_{x_2}(x_1(t), x_2(t), x_3(t)) = x_1(t) \\ x_3(t+1) &= f_{x_3}(x_1(t), x_2(t), x_3(t)) = x_2(t). \end{aligned} \tag{1.3}$$

The wiring diagram for this system and its state space transition diagram are presented in Figure 1.6. The verification is left as an exercise (Exercise 1.7). In addition to the fixed points (0, 0, 0) and (1, 1, 1), the state space diagram contains two cycles of length three: (0, 0, 1) → (1, 0, 0) → (0, 1, 0) → (0, 0, 1) and (0, 1, 1) → (1, 0, 1) → (1, 1, 0) → (0, 1, 1). We say that the state space diagram contains four components, two fixed points, and two cycles of length three.

When the number of variables is small, it is common to use capital letters to denote them instead of using $x_1, x_2, \ldots, x_n$. With such notation, for the example in

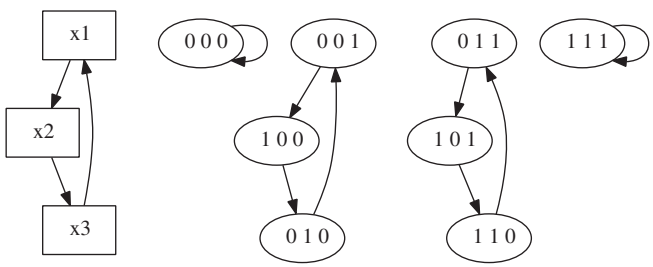

FIGURE 1.6

Wiring diagram and the state space transition diagram for the Boolean dynamical system in Example 1.3 given by Eqs. (1.3). Graphs produced with DVD [12].

Figure 1.4 the model variables are $x_1 = M, x_2 = E$, and $x_3 = L$. The functions defining the systems transition will then be denoted as $f_{x_1} = f_M, f_{x_2} = f_E, f_{x_3} = f_L$.

Deciding on the actual expressions for these Boolean functions is a critical step in the definition of a model. It should take into account all relevant parts of the known biological mechanism of the system being modeled. In the next section we will consider ways of doing this for the *lac* operon.

Exercise 1.2. How many possible states does a system of n Boolean variables have if:

a. $n = 2$?
b. $n = 4$?
c. $n = 5$?
d. $n = 10$?
e. $n = 100$? ▽

Exercise 1.3. For each of the Boolean expressions below, compute the value if $x_1 = 1, x_2 = 1, x_3 = 0, x_4 = 0, x_5 = 1$. In the cases where the expressions are not fully parenthesized, be mindful of the operation precedence.

a. $\overline{x_3} \wedge x_5$
b. $(x_3 \vee x_5) \wedge (\overline{x_2} \vee x_1) \vee x_4$
c. $(x_2 \wedge \overline{x_1}) \vee (\overline{x_2} \wedge \overline{x_3}) \wedge x_5$
d. $(x_1 \vee x_2) \wedge \overline{(x_2 \wedge x_3)}$. ▽

Exercise 1.4. Show that $(0, 0, 0)$ is a fixed point for the system described by Eqs. (1.1) and (1.2). ▽

Exercise 1.5. Show that $(0, 1, 0)$ and $(1, 0, 0)$ form a cycle of length two. That is, show that Eqs. (1.1) and (1.2) lead to the following trajectories that alternate between these two states: $(0, 1, 0) \to (1, 0, 0) \to (0, 1, 0) \to \ldots$ and $(1, 0, 0) \to (0, 1, 0) \to (1, 0, 0) \to \ldots$ ▽

Exercise 1.6. For the system whose transition functions are given by Eqs. (1.1) and (1.2), compute the trajectory of $(1, 0, 1)$. ▽

Exercise 1.7. Verify that the wiring diagram and the state space diagram for the Boolean network defined by Eqs. (1.3) are as depicted in Figure 1.6. ▽

1.3.3 Creating a Boolean Model of the *Lac* Operon

Once a choice for model variables has been made and the wiring diagram has been constructed, the Boolean transition functions for the model are determined from the wiring diagram and additional available information or assumptions regarding the variable interactions. When multiple quantities impact a third, we need to know if the simultaneous presence of each of these quantities is necessary to exert the effect or if the presence of just one of them would be enough. In the first case, the transition

function will use the operator AND, and in the second, it will use the operator OR. We illustrate this by defining a set of transition functions consistent with the wiring diagram in Figure 1.4 for the *lac* operon.

We make the following additional model assumptions. Although some of them may appear too restrictive, keep in mind that they can always be modified or removed after the initial testing and analysis of the model:

- Transcription and translation require one unit of time. This means that when all necessary conditions for the activation of the molecular mechanism are present at time t, the protein production will occur and the product will be available at time $t + 1$.
- Degradation of all mRNA and proteins occurs in one time step. When conditions for making new mRNA or proteins are not met at time t, all previously available amounts of mRNA and proteins have already fallen below the threshold levels and will be absent at time $t + 1$.
- In the presence of β-galactosidase, lactose metabolism occurs in one time step. If lactose and β-galactosidase are available at time t but the conditions for bringing new lactose through the cell membrane from the external medium are not met at time t, by time $t + 1$ all of the lactose is converted to glucose and galactose, and lactose is not available at time $t + 1$.

With this, we define the transition functions (update rules) for the "minimal" *lac* operon model based on the wiring diagram presented in Figure 1.4 as:

$$\begin{aligned} x_M(t+1) &= f_M(t+1) = \overline{G_e} \wedge (L(t) \vee L_e(t)) \\ x_E(t+1) &= f_E(t+1) = M(t) \\ x_L(t+1) &= f_L(t+1) = \overline{G_e} \wedge ((E(t) \wedge L_e(t)) \vee (L(t) \wedge \overline{E}(t))). \end{aligned} \tag{1.4}$$

We explain these equations next.

Boolean functions for M: The update rule states that for mRNA to be present at time $t+1$, there should be no external glucose at time t, and either internal or external lactose should be present. When external glucose is present at time t ($G_e(t) = 1$), no mRNA will be available at time $t+1$ ($M(t+1) = 0$). Also, when there is no external glucose at time t ($G_e(t) = 0$) and there are high concentrations of lactose inside the cell at time t ($L(t) = 1$) or outside the cell ($L_e(t) = 1$), lactose concentrations in the cell will be sufficiently high (above threshold levels) to cause mRNA production and make mRNA available at time $t + 1$ ($M(t + 1) = 1$).

Boolean function for E: With mRNA available at time t ($M(t) = 1$), the LacZ polypeptide will be produced and available at time $t + 1$ ($E(t) = 1$).

Boolean function for L: When external glucose is available at time t ($G_e(t) = 1$), no lactose will be brought into the cell from the medium and, thus, no internal lactose will be available at time $t+1$ ($L(t+1) = 0$). When external glucose is absent from the medium at time t ($G_e(t) = 0$), external lactose will be available at time $t + 1$ when at least one of the following conditions is satisfied: (i) External lactose and *lac* permease (as represented by the polypeptide E) are both present at time t ($E(t) \wedge L_e(t) = 1$).

The *lac* permease will then bring the extracellular lactose inside the cell, ensuring the presence of intracellular lactose at time $t + 1$; or (ii) Internal lactose is available at time t but no β-galactosidase (as represented again by the polypeptide E) is available at time t to metabolize it ($L(t) \wedge \overline{E}(t) = 1$). Thus, the internal lactose will still be present at time $t + 1$.

1.3.4 Initial Testing of the Boolean Model of the *Lac* Operon from Eqs. (1.4)

Now that we have defined a model of the *lac* operon, it must be analyzed and validated. Since a model can never be shown to be correct in an absolute sense and is always just an approximation of the actual system, its validation is only appropriate within the context of the questions that the model is developed to help answer. In our case, the simple model we have created should be able to describe the basic qualitative dynamic properties of the *lac* operon. Thus, at a minimum, our model should show that the operon has two steady states, On and Off. When extracellular glucose is available, the operon should be Off. When extracellular glucose is not present and extracellular lactose is, the operon must be On. We next demonstrate that our model satisfies these conditions.

Recall that the operon is On when mRNA is being produced ($M = 1$). When mRNA is present, the production of *lac* permease, and β-galactosidase is also turned on. This corresponds to the fixed-point state $(M, E, L) = (1, 1, 1)$. On the other hand, when mRNA is not made, the operon is Off. This also means no production of *lac* permease, and β-galactosidase. This corresponds to the fixed-point state $(M, E, L) = (0, 0, 0)$.

For the Boolean model of the *lac* operon from Eqs. (1.4), there are four possible combinations for the values L_e and G_e of the model parameters: $L_e = 0, G_e = 0$; $L_e = 0, G_e = 1$; $L_e = 1, G_e = 0$; and $L_e = 1, G_e = 1$. For each one of these pairs of values we can determine the state space transition diagram of the model from the update functions in Eqs. (1.4). The results are shown in Figure 1.7. Notice that according to the model, the operon is On only when external glucose is unavailable and external lactose is present. In all other cases, the operon is Off. These model predictions reflect exactly the expected behavior of the *lac* operon based on the underlying regulatory mechanisms described earlier. This means that our initial model is capable of describing the most fundamental behavior of the *lac* operon system and captures the main qualitative properties of *lac* operon regulation.

Exercise 1.8. Verify that the space state diagram for the Boolean model described by Eqs. (1.4) is as presented in Figure 1.7b. Notice that for some values of the parameters, the transition functions simplify significantly when we apply short-circuit evaluation for the appropriate Boolean expressions. For instance, when $G_e = 1$, the equations for the transition functions will be: $x_M(t+1) = f_M(t+1) = \overline{1} \wedge (L(t) \vee L_e(t)) = 0$, regardless of the values of L and L_e and $x_L(t + 1) = f_L(t + 1) = \overline{1} \wedge ((E(t) \wedge L_e(t)) \vee (L(t) \wedge \overline{E}(t))) = 0$, regardless of the values of L, E, and L_e. ▽

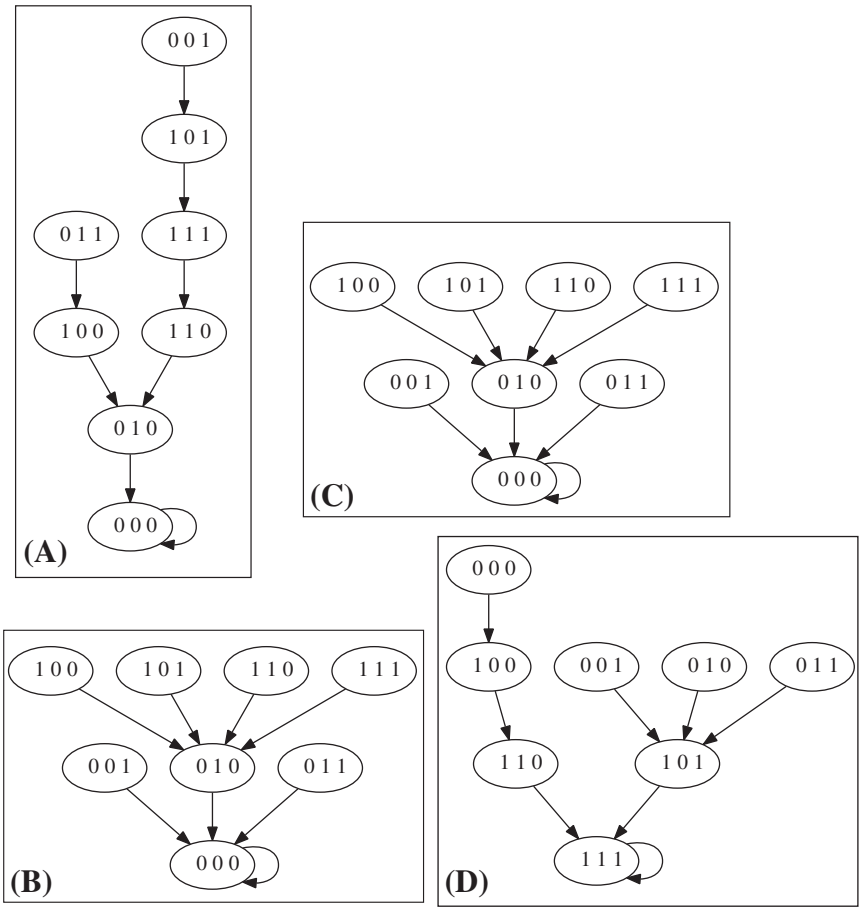

FIGURE 1.7

The state space transition diagram of triples (M, E, L) for the Boolean model of the *lac* operon defined in Eqs. (1.4) for the four possible combinations of parameter values. In each of the cases, the operon has a single fixed point and no limit cycles. Panel (A) $L_e = 0; G_e = 0$; The operon is Off. Panel (B) $L_e = 0; G_e = 1$; The operon is Off. Panel (C) $L_e = 1; G_e = 1$; The operon is Off. Panel (D) $L_e = 1; G_e = 0$; The operon is On. Graphs obtained using DVD [12].

1.3.5 Using Discrete Visualizer of Dynamics (DVD) to Test a Boolean Model

The *Discrete Visualizer of Dynamics* (DVD) is a suite of freely available web applications (http://dvd.vbi.vt.edu) that takes the transition functions of the Boolean model

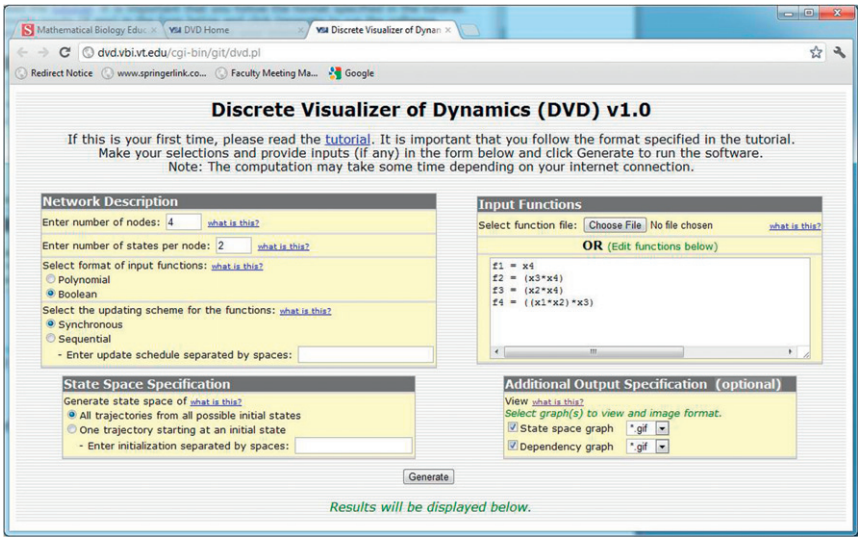

FIGURE 1.8

DVD v1.0 main page screenshot.

as input and returns the wiring diagram and information about the state space of the network including fixed points, number of components, cycles, and cycle lengths. For systems with a relatively small number of variables, DVD can also compute and display the entire space transition diagram. For larger networks, given any initial state, DVD can compute and display the trajectory of this initial state. DVD is fairly intuitive, well documented, and easy to use. In this chapter we will use DVD v1.0 available under the link with the same name from DVDs main page (Figure 1.8). A link to an online tutorial is available on the same page (see [12] for more details). Our next example illustrates the use of DVD.

Example 1.4. Assume we want to analyze the behavior of a Boolean network with four nodes with the following set of transition functions:

$$\begin{aligned}
x_1(t+1) &= f_{x1}(x_1(t), x_2(t), x_3(t), x_4(t)) = x_2(t)\\
x_2(t+1) &= f_{x2}(x_1(t), x_2(t), x_3(t), x_4(t)) = x_3(t) \wedge x_4(t)\\
x_3(t+1) &= f_{x3}(x_1(t), x_2(t), x_3(t), x_4(t)) = x_2(t) \wedge x_4(t)\\
x_4(t+1) &= f_{x4}(x_1(t), x_2(t), x_3(t), x_4(t)) = x_1(t) \wedge x_2(t) \wedge x_3(t).
\end{aligned}$$

In DVD, the number of nodes under "Network Description" should be set to four; the number of states for each node should be two (since only the Boolean values 0 and 1 are allowed); the format for the input functions should be set to Boolean, and the updating schedule should be set to synchronous (see Figure 1.8).

To analyze the network, we enter the transition functions in the Input Functions text area as follows:

$$\begin{aligned} f1 &= x4 \\ f2 &= (x3 * x4) \\ f3 &= (x2 * x4) \\ f4 &= ((x1 * x2) * x3). \end{aligned} \tag{1.5}$$

DVD uses the following symbols for the basic logical operations: $*$ for AND, + for OR, and $\sim$ for NOT. Notice the multiple sets of parentheses. DVD requires that all Boolean expressions are *fully parenthesized*, meaning that every single operation should be enclosed in parentheses. Multiple sets of parentheses for the same operations *should not be used*, as this may lead to incorrect results. In DVD the nodes are always labeled x1, x2, and so on; their respective update rules are denoted f1, f2, and so on. If your model uses different variable names, they need to be changed before using DVD to conform with this requirement.

Exercise 1.9. Open DVD v1.0 and enter and run the example from Eqs. (1.5). Check the appropriate boxes in the bottom right panel "Additional Output Specification" to generate the state space transition graph and the dependency graph. Clicking on the "Generate" button displays the characteristics of the Boolean network. Follow the links to the space graph and dependency graph to examine them. Answer the following questions: (1) How many fixed points does the model have? (2) How many components does the state space graph have? (3) Are there any limit cycles? ▽

Exercise 1.10. For the example from Eqs. (1.5), use DVD to find the trajectory of the state (0, 1, 1, 1). To do so, select the radio button "One Trajectory…" in the "State Space Specification" panel and enter the components of the space separated by spaces: 0 1 1 1. Clicking on the "Generate" button will display the path. ▽

DVD does not provide a specific option for designating selected nodes as parameters. Thus, parameter values should either be entered explicitly as 0s or 1s in the model equations or they should be treated as "variables" that retain their constant values for all time steps. To do the latter, for each parameter P, we add an update rule of the form $f_P = P$ (usually at the end of the model, after the update rules for all of the variables). This ensures that P retains its initial value for all time steps t. Of course, with this approach, the DVD output should be interpreted carefully and with the understanding that different sets of initial values for the parameters should be considered as separate outputs from the model.

Example 1.5. We will show how the *lac* operon model from Eqs. (1.4) can be analyzed using DVD. First, the variable names will need to change to names accepted by DVD. We will use x1 for M, x2 for E, and x3 for L. Two different approaches are possible for the model parameters:

1. Enter and run the model four different times for each of the four combinations of the parameters L_e and G_e (see Exercise 1.11). As an example, for $L_e = 1$ and

Table 1.2 DVD output for the update functions in Eqs. (1.6) See the text for details.

ANALYSIS OF THE STATE SPACE [m = 2, n = 5]

There are 4 components and 4 fixed point(s)

Components	Size	Cycle Length
1	8	1
2	8	1
3	8	1
4	8	1

TOTAL: $32 = 2^5$ nodes
Printing fixed point(s).
[0 0 0 0 0] lies in a component of size 8.
[0 0 0 0 1] lies in a component of size 8.
[0 0 0 1 1] lies in a component of size 8.
[1 1 1 1 0] lies in a component of size 8.

$G_e = 0$, we should enter the model into DVD as

$$f1 = ((\sim 0) * (x3 + 1))$$
$$f2 = x1$$
$$f3 = ((\sim 0) * ((x2 * 1) + (x3 * (\sim x2)))).$$

2. Introduce two new variables, $x4$ for L_e and $x5$ for G_e. The update rules for these variables are $f4 = x4$ and $f5 = x5$ and the entire model is

$$\begin{aligned} f1 &= ((\sim x5) * (x3 + x4)) \\ f2 &= x1 \\ f3 &= ((\sim x5) * ((x2 * x4) + (x3 * (\sim x2)))) \\ f4 &= x4 \\ f5 &= x5. \end{aligned} \tag{1.6}$$

Taking the second approach and running the model in DVD produces the output in Table 1.2 and the state space diagram in Figure 1.9. There are four fixed points, each one of which corresponds to a different combination of the parameter values for L_e and G_e. Each such combination corresponds to a component in the state space transition graph in Figure 1.9. The last two values of each state encode the values of the parameters L_e and G_e, respectively. Viewed this way, Figure 1.9 is identical to the analysis of the *lac* operon model presented in Figure 1.7.

Exercise 1.11. Use DVD to analyze the model of the *lac* operon from Eqs. (1.4), running it four times for the four different values of the parameters. Compare the results with those in Table 1.2, Figure 1.7 and Figure 1.9. ▽

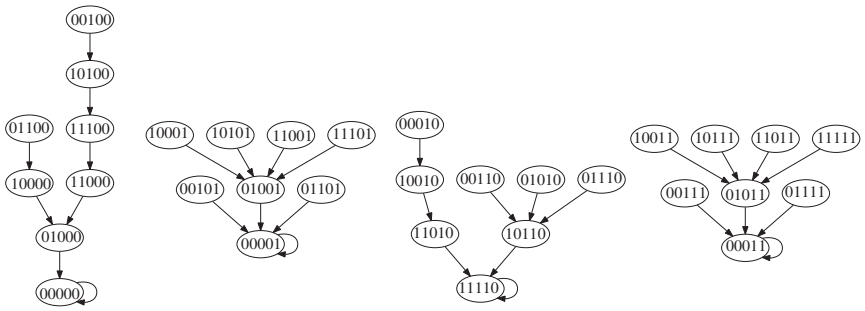

FIGURE 1.9

State space diagram for the model defined by the update functions in Eqs. (1.6). The order of components for each state is (M,E,L,L_e,G_e). See the text for further details.

1.3.6 How to Recognize a Deficient Model

In the minimal model introduced and discussed above, the testing and analysis showed that the model can adequately represent the behavior of the operon to be On or Off. One should remember, however, that this model is just one of many possible models that can be created and that every model relies on a set of assumptions made during the modeling process. For the minimal Boolean model above, we made several assumptions both during the process of selecting the model variables and at the stage of writing down the transition functions that determine the dynamical behavior of the system. Different choices and assumptions would lead to different models, some of which may work and some of which may not. In this section we present a "model" (and the quotation marks here are meant to indicate that we will ultimately show that it is not a good model) that may look legitimate at first. The initial testing, however, will show that the model does not adequately capture the regulatory behavior of the *lac* operon.

The model is based on the following five variables: mRNA (M), β-galactosidase (B), lac permease (P), intracellular lactose (L), and allolactose (A). This certainly appears to be a reasonable choice, as it includes the primary components of the *lac* operon regulatory mechanism. As in the previous model, and for the same reasons as before, the CAP-cAMP positive control mechanism is excluded from the modeling effort. The following transition functions are proposed for describing the underlying biology:

$$\begin{aligned} f_M &= A \\ f_B &= M \\ f_A &= A \vee (L \wedge B) \\ f_L &= P \vee (L \wedge \overline{B}) \\ f_P &= M. \end{aligned} \tag{1.7}$$

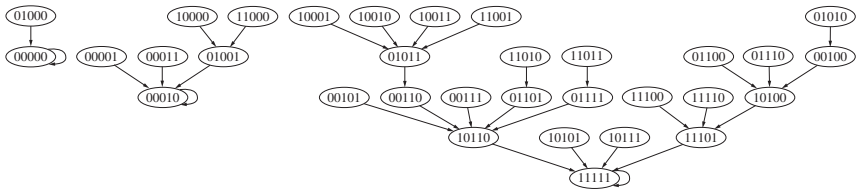

FIGURE 1.10

The state space transition diagram of the points (M, B, A, L,P) for the Boolean model defined by Eqs. (1.7).

Here, as for the minimal model from Eqs. (1.4), it is assumed that translation and transcription require one unit of time, protein and mRNA degradation require one unit of time, and lactose metabolism requires one unit of time. This model does not involve any parameters: instead, it assumes that extracellular lactose is always available, and extracellular glucose is always unavailable. The rationale and interpretation for the functions proposed by Eqs. (1.12) comes from the biological interactions between the main regulatory elements. For instance, the transition function f_L indicates that at time $t + 1$ lactose (L) will be available if permease (P) is available at time t to bring the extracellular lactose into the cell or, in case lactose (L) is already available at time t, there is no β-galactosidase at time t ($\overline{B}$) to metabolize the lactose into glucose and galactose.

Exercise 1.12. Sketch the wiring diagram (the dependency graph) for the model described by Eqs. (1.7). ▽

As for our earlier model, one way to initially test the model from Eqs. (1.7) is to examine its fixed points and determine if they correspond to states that are biologically feasible. The state space graph for the model defined by Eqs. (1.7) is presented in Figure 1.10. It has three fixed points (M, B, A, L, P): (0, 0, 0, 0, 0), (1, 1, 1, 1, 1), and (0, 0, 0, 1, 0). The first two correspond to the operon being Off and On, respectively. The third one, however, corresponds to a biological scenario under which the bacterium does not metabolize the intracellular lactose, which is unrealistic (recall that the model assumed that extracellular lactose is always available and that there is no extracellular glucose).

The fact that the state (M, B, A, L, P) = (0, 0, 0, 1, 0) is a fixed point for the model indicates that the model does not represent accurately the most important qualitative behavior of the *lac* operon regulation. Thus, the model fails the initial testing and is in need of modification.

Exercise 1.13. Consider the transition functions in Eqs. (1.7) and criticize the model. Can you find reasons to question the definitions of the transition functions? The way to approach this is by examining each function and asking if it accurately reflects the underlying biology and/or the model assumptions. ▽

Exercise 1.14. Consider possible modifications of the transition functions in Eqs. (1.7), aimed at eliminating the biologically infeasible fixed point. Give the

rationale for your modification and specify the biological mechanism or model assumptions that justify the change. Use DVD to analyze the modified model. For each of your modifications, use the number of fixed points to decide if they correspond to biologically realistic situations. Note that there should be no limit cycles. ▽

1.3.7 A More Refined Boolean Model of the *Lac* Operon

The models we have considered so far attempted to reproduce features of the *lac* operon mechanisms by including a relatively small number of variables. The catabolite repression mechanism (the CAP-cAMP positive control loop) was excluded, as was the explicit modeling of the presence of low levels of lactose and allolactose. For proteins and enzymes, since basal concentrations are always nonzero, we used the Boolean value 0 to refer to basal levels and 1 for concentrations that are much higher. This does not apply to lactose and allolactose, the concentrations of which may be truly zero. Thus, in the case of lactose and allolactose, it is justified to look for ways to model low but nonzero concentrations separately.

There are several possible options to do so. One of them is to consider discrete models in which the variables take values from a set S of three or more values, corresponding to ranges in the respective concentrations. If $S = \{0, 1, 2\}$, the value 0 may be used to represent a concentration near zero, 1 to represent a low concentration that is not near zero, and 2 to represent a high concentration. This approach can no longer be implemented using Boolean networks and leads to discrete models with multiple states. When the set S has a prime number of elements, it can be shown that the transition functions of the model have a representation as polynomials of the model variables. The theory of polynomial dynamical systems is then applicable to the analysis of such networks and there are many interesting mathematical questions arising in this context (see, e.g., [17–21]).

Another possible approach, which we will examine here, is to stay within the framework of Boolean networks and introduce additional Boolean variables to allow for a separation into three concentration ranges instead of two. The model that follows comes from Stigler and Veliz-Cuba [22] and utilizes this approach. It also includes the CAP-cAMP positive control mechanism of the *lac* operon that was not considered in the models discussed so far.

We begin by identifying the model variables and parameters. As in the minimal model, L_e and G_e are model parameters, denoting the extracellular lactose and the extracellular glucose. The variables L_l and A_l (the index stands for low concentration) are introduced to facilitate the ability of the model to distinguish between "no lactose" and "some lactose" and similarly for allolactose. This is an improvement over our initial model because, unlike for proteins and enzymes, it would not be warranted to assume that baseline levels of lactose or allolactose are always present. When L_l and A_l have value 1, this means that at least low concentrations of lactose and allolactose, respectively, are available in the cell. As before, $L = 1$ and $A = 1$ stand for high levels of lactose and allolactose. High levels of lactose or allolactose at any given

time t imply at least low levels for the next time step $t + 1$. The complete list of model variables is:

M = *lac* mRNA	L = high concentration of intracellular lactose
P = *lac* permease	A = high concentration of allolacose (inducer)
$B = \beta$-galactosidase	L_l = (at least) low concentration of intracellular lactose
C = catabolite activator protein CAP	
R = repressor protein *lac*I	A_l = (at least) low concentration of allolactose

The model assumptions are:

- Transcription and translation require one unit of time. This means that if all necessary conditions for the activation of the molecular mechanism are present at time t, the protein production will be happening in time $t + 1$.
- Degradation of all mRNA and proteins occurs in one time step.
- High levels of lactose or allolactose at any given time t imply at least low levels for the next time step $t + 1$.

The Boolean transition functions for the model reflect the dependencies between variables according to the regulatory mechanisms of the *lac* operon from Section 2. We provide justification for two of the functions; the rest are left as exercises. The corresponding wiring diagram is depicted in Figure 1.11.

$$\begin{array}{ll} f_M = \overline{R} \wedge C & \\ f_P = M & f_B = M \\ f_C = \overline{G_e} & f_R = \overline{A} \wedge \overline{A_l} \\ f_A = L \wedge B & f_{A_l} = A \vee L \vee L_l \\ f_L = \overline{G_e} \wedge P \wedge L_e & f_{L_l} = \overline{G_e} \wedge (L \vee L_e). \end{array} \tag{1.8}$$

Boolean function for R : For the concentration of the repressor protein to be high ($R = 1$), there should be no allolactose present; that is $A = A_l = 0$.

Boolean function for M: In order for production of mRNA to be high, there should be no repressor protein ($R = 0$) and the concentration of CAP should be high ($C = 1$).

Exercise 1.15. Consider the rest of the transition functions from Eqs. (1.8). Give justification for the definition of each function and provide any additional assumptions that the definition may imply. ▽

Exercise 1.16. There is no variable to represent cAMP in the Boolean model with the wiring diagram depicted in Figure 1.11 and defined by Eqs. (1.8). Could you justify this decision? How would the model change if a cAMP variable is introduced? Do you think this change will impact the qualitative behavior of the model? ▽

As with the earlier Boolean models we have examined, the next logical step in the process is to find the fixed points of the model and determine if it accurately reflects the ability of the *lac* operon to be On or Off. The model now has nine variables,

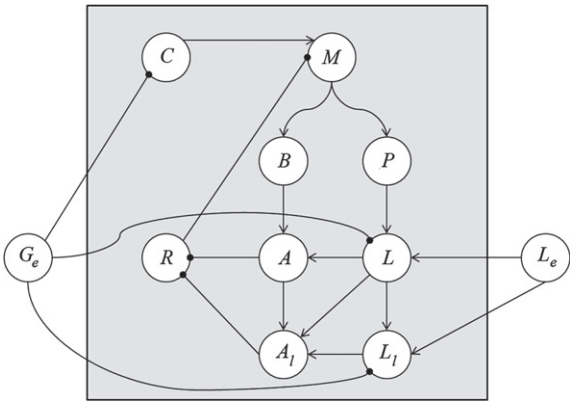

FIGURE 1.11

The wiring diagram for the Boolean model defined by Eqs. (1.8). The nodes inside the square region represent model variables, while the outside nodes correspond to model parameters. Arrows indicate positive interactions, circles indicate negative interactions. (Figure adapted from Stigler and Veliz-Cuba [22]).

leading to a state space for the Boolean network model of size $2^9 = 512$ for each of the four combinations of the parameter values. It would be nearly impossible to compute the individual trajectories of each of these points by hand and determine the fixed points this way but we may think that a computer would be able to easily do that. However, for networks with large numbers of nodes, such a "brute force" approach based on considering all possible trajectories becomes impossible even for the fastest computers. As an example, a Boolean model of T cell receptor signaling developed in [23] contains 94 nodes. This means that the state space contains 2^{94} states (approximately equal to $2 * 10^{28}$, a number that is about 3,000,000 times larger than the estimated number of stars in the observable universe [24]). This means that years of computing time would be needed even on the fastest computers. Clearly, this is not practical, implying that DVD and other similar software applications must use a different, computationally efficient, way to determine the fixed points of a Boolean network. This justifies the following question: Given a Boolean network model, how can we determine its fixed points without having to compute the entire state space transition diagram?

1.4 DETERMINING THE FIXED POINTS OF BOOLEAN NETWORKS

The dynamical behavior of a Boolean network model with nodes $x_1, x_2, \ldots, x_n$ is described by the transition functions $f_{x_1}, f_{x_2}, \ldots, f_{x_n}$, namely

$$
\begin{aligned}
x_1(t+1) &= f_{x_1}(x_1(t), x_2(t), \ldots, x_n(t)) \\
x_2(t+1) &= f_{x_2}(x_1(t), x_2(t), \ldots, x_n(t)) \\
&\ldots \\
x_n(t+1) &= f_{x_n}(x_1(t), x_2(t), \ldots, x_n(t)).
\end{aligned} \tag{1.9}
$$

A point $(p_1, p_2, \ldots, p_n)$ is *a fixed point* for the set of Eqs. (1.9), when plugging the values $(p_1, p_2, \ldots, p_n)$ for $x_1(t), x_2(t), \ldots, x_n(t)$ into the functions $f_{x_1}, f_{x_2}, \ldots f_{x_n}$, from Eqs. (1.9) returns the same values $(p_1, p_2, \ldots, p_n)$ for $x_1(t+1), x_2(t+1), \ldots, x_n(t+1)$. In other words, $(p_1, p_2, \ldots, p_n)$ is a fixed point for the set of Eqs. (1.9) when

$$
\begin{aligned}
p_1 &= f_{x_1}(p_1, p_2, \ldots, p_n) \\
p_2 &= f_{x_2}(p_1, p_2, \ldots, p_n) \\
&\ldots \\
p_n &= f_{x_n}(p_1, p_2, \ldots, p_n).
\end{aligned} \tag{1.10}
$$

We seek a method for determining all points $(p_1, p_2, \ldots, p_n)$ that satisfies the condition in Eqs. (1.10).

The brute force approach in this context would be to test all possible states for the fixed-point property defined by Eqs. (1.10). However, since the size of the state space grows exponentially with the number of nodes in the network, we already decided that going over the entire state space to test all of its elements is not practical (see Exercise 1.2). The approach we describe next makes it possible to rephrase the problem of finding all fixed points $(p_1, p_2, \ldots, p_n)$ of a Boolean network as a problem of solving systems of polynomial equations and determining a computationally efficient way of finding the solutions. This method uses a computational algebra approach and is based on some fundamental results in abstract algebra and algebraic geometry and on the use of Groebner bases for solving systems of polynomial equations [25].

Equations (1.10) state that the fixed points of the model are the solutions $(x_1, x_2, \ldots, x_n) = (p_1, p_2, \ldots, p_n)$ of the system of equations

$$
\begin{aligned}
x_1 &= f_{x_1}(x_1, x_2, \ldots, x_n) \\
x_2 &= f_{x_2}(x_1, x_2, \ldots, x_n) \\
&\ldots \\
x_n &= f_{x_n}(x_1, x_2, \ldots, x_n)
\end{aligned}
\quad \text{or, equivalently,} \quad
\begin{aligned}
f_{x_1}(x_1, x_2, \ldots, x_n) - x_1 &= 0 \\
f_{x_2}(x_1, x_2, \ldots, x_n) - x_2 &= 0 \\
&\ldots \\
f_{x_n}(x_1, x_2, \ldots, x_n) - x_n &= 0,
\end{aligned} \tag{1.11}
$$

where, the functions $f_{x_1}, f_{x_2}, \ldots, f_{x_n}$, are Boolean functions.

Table 1.3 The table of values for Boolean functions $x_1 \wedge x_2$; $x_1 \vee x_2$, and $\overline{x_1}$ and their equivalents in polynomial form.

x_1	x_2	$x_1 \wedge x_2$	$x_1 x_2$	$x_1 \vee x_2$	$x_1 + x_2 + x_1 x_2$	$\overline{x_1}$	$x_1 + 1$
0	0	0	0				
0	1	0	0				
1	0	0	0				
1	1	1	1				

The general theory we will be applying here describes a method for solving such systems of equations in the special case where the functions $f_{x_1}, f_{x_2}, \ldots, f_{x_n}$, are polynomials of the variables $x_1, x_2, \ldots, x_n$. This may appear to be a serious restriction since in the case of Boolean networks the transition functions do not seem to satisfy this condition. However, as we will see next, each Boolean function can be rewritten as a *Boolean polynomial*, where a Boolean polynomial of the (Boolean) variables $x_1, x_2, \ldots, x_n$ is a sum of terms of the form $x_1{}^{b_1} x_2{}^{b_2} \ldots x_n{}^{b_n}$ with $b_i \in \{1, 0\}, i = 1, 2, \ldots, n$.

Example 1.6. $f(x_1, x_2, x_3, x_4) = x_1x_3x_4 + x_2x_3 + x_1x_3$ is a Boolean polynomial of the variables x_1, x_2, x_3, x_4. For the first term, we have $b_1 = 1, b_2 = 0, b_3 = 1$, and $b_4 = 1$. For the second and third terms, $b_1 = 0, b_2 = 1, b_3 = 1$, and $b_4 = 0$, and $b_1 = 1, b_2 = 0, b_3 = 1$, and $b_4 = 0$, respectively.

To convert any Boolean function into a Boolean polynomial, the Boolean operations AND, OR, and NOT need to be converted to polynomials as follows:

1. $x_1 \wedge x_2 = x_1x_2$
2. $x_1 \vee x_2 = x_1 + x_2 + x_1x_2$
3. $\overline{x_1} = x_1 + 1$.

To verify that the Boolean and polynomial functions (1)–(3) are identical, we need to compare their values and verify that the two functions return the same values for the same inputs. Table 1.3 contains the proof that $x_1 \wedge x_2 = x_1x_2$.

Since addition and multiplication are performed over the field {0,1}, the following addition and multiplication rules apply for any value $a \in \{0, 1\} : a + a = 0$ and $a * a = a$.

Exercise 1.17. Fill in the remaining columns of Table 1.3 to show that the equalities (2) and (3) above hold. ▽

Exercise 1.18. Show that for any $a, b \in \{0, 1\}, a - b = a + b$. As a special case, over the field $\{0, 1\}, -1 = 1$. ▽

Example 1.7. We next translate the Boolean model of the *lac* operon defined by Eqs. (1.8) into polynomial form. We will give the translation for the function $f_{A_l} = A \vee L \vee L_l$ and leave the rest of the functions as an exercise:

$$\begin{aligned} f_{A_l} &= A \vee L \vee L_l = ((A \vee L) \vee L_l) = ((A + L + AL) \vee L_l) \\ &= (A + L + AL) + L_l + (A + L + AL)L_l \\ &= A + L + AL + L_l + AL_l + LL_l + ALL_l. \end{aligned}$$

Thus, in polynomial form, $f_{A_l} = ALL_l + AL_l + LL_l + AL + A + L + L_l$.

Exercise 1.19. Translate the remaining eight functions in Eqs. (1.8) into polynomials and simplify as much as possible to obtain the polynomial form of the Boolean model defined by the set of Eqs. (1.8). Write down the whole model (it has nine variables and two parameters!) and save it. You will need this polynomial form of the model again in Example 1.10 and in Exercise 1.20 below. ▽

Once the system of Boolean Eqs. (1.10) is translated into a system of polynomial equations, finding the fixed points becomes a problem of solving a polynomial system of equations. This can be done by employing a technique from computational algebra involving Groebner bases. With this technique, it is generally possible to rewrite a system of polynomial equations in a much simpler form while preserving the set of solutions. Under some additional conditions, the reduced form of the system of equations is guaranteed to have a form that makes finding the solutions possible by back-substitution. Readers familiar with the method of Gaussian elimination for systems of linear equations will notice that finding a Groebner basis that diagonalizes the system of equations can be viewed as a generalization of the Gaussian elimination method for polynomial functions.

For readers with appropriate mathematical background in abstract algebra, including the basic theory of rings, fields, ideals, and varieties, an outline of the theory of Groebner bases, as it relates to solving systems of polynomial equations, is presented in the online Appendix 1. For what follows, however, we do not assume familiarity with this theory. Instead, we illustrate how knowing the diagonalized forms of the polynomial systems (obtained by determining the Groebner basis with the use of computational software) can help to solve the system of polynomial equations and determine the fixed points.

Various computer systems including Macaulay 2, MAGMA, CoCoA, SINGULAR, and others, have the capability of computing Groebner bases. For our needs, using the web-based *SAGE* interface for Macaulay 2 for determining the Groebner basis for a diagonalized reduction is easy and convenient (http://www.sagemath.org/). We suggest that you use the online option first. Select "Try *SAGE* Online" and register for a *SAGE* notebook account. We consider a few examples below. The first two illustrate how the methods can be used to solve systems of polynomial equations over the real numbers, a problem that the reader is likely to be more familiar with. We then use the method to determine the fixed points of the Boolean model of the *lac* operon defined by Eqs. (1.8).

Example 1.8. Consider the system of polynomial equations below where x, y, and z are real numbers. We want to find the solutions of the system of equations.

$$\begin{aligned} x^2 + y^2 + z^2 &= 1 \\ x^2 + z^2 &= y \\ x - z &= 0. \end{aligned}$$

To use Groebner bases to diagonalize the system, we first need to rewrite the equations to ensure that the right-hand side is zero:

$$\begin{aligned} x^2 + y^2 + z^2 - 1 &= 0 \\ x^2 + z^2 - y &= 0 \\ x - z &= 0. \end{aligned} \tag{1.12}$$

Next, we need to find the Groebner basis for the functions that form the left-hand sides of the equations: $h_1 = x^2 + y^2 + z^2 - 1$; $h_2 = x^2 + z^2 - y$; $h_3 = x - z$. To compute the Groebner basis for these functions in *SAGE*, we use the following commands (click on "evaluate" after entering each line)[2]:

```
P.<x,y,z> = PolynomialRing(RR, 3, order='lex')
I = ideal(x^2+y^2+z^2-1, x^2+z^2-y, x-z)
B = I.groebner_basis()
```

In the first command, the variables in the system are listed on the left. On the right, RR denotes the real numbers (meaning we want to work over the reals) and three is the number of variables. For our purposes, we will always need to use `order='lex'`.

After evaluating the last command, *SAGE* returns the Groebner basis:

```
[x-z, y - 2*z^2, z^4 + 1/2*z^2 - 1/4].
```

It follows from the general theory outlined in the online Appendix 1 that the systems of Eqs. (1.12) has a solution set equivalent to the solution set of the system

$$\begin{aligned} z^4 + \frac{1}{2}z^2 - \frac{1}{4} &= 0 \\ y - 2z^2 &= 0 \\ x - z &= 0. \end{aligned}$$

Notice that this system is "diagonal" in the sense that if we solve the first equation for z, we can use the values in the second equation to solve for y and then in the third equation for x. After doing this, we obtain the solutions $z = \pm\sqrt{\frac{-1+\sqrt{5}}{4}}$; $y = 2z^2$; $x = z$ for the system, which leads to the solutions $z = \sqrt{\frac{-1+\sqrt{5}}{4}}$; $y = \frac{-1+\sqrt{5}}{2}$; $x = \sqrt{\frac{-1+\sqrt{5}}{4}}$ and $z = -\sqrt{\frac{-1+\sqrt{5}}{4}}$; $y = \frac{-1+\sqrt{5}}{2}$; $x = -\sqrt{\frac{-1+\sqrt{5}}{4}}$.

[2]Note that exponentiation is indicated with ^ and multiplication must be indicated with *.

Example 1.9. Repeat Example 1.8 with the following system of polynomial equations.

$$\begin{aligned} x^2y - z^3 &= 0 \\ 2xy - 4z - 1 &= 0 \\ z - y^2 &= 0 \\ x^3 - 4zy &= 0. \end{aligned}$$

After entering the appropriate commands as in Example 1.8 for this system of equations, *SAGE* returns the Groebner basis [1] for the functions $f_1 = x^2y - z^3$; $f_2 = 2xy - 4z - 1$; $f_3 = z - y^2$; $f_4 = x^3 - 4zy$.

This means that the solution set of the system of equations above is equivalent to the solution set of the equation $1 = 0$, indicating that the solution set is empty. The system does not have a real-valued solution.

Example 1.10. Consider again Exercise 1.19 where you wrote the polynomial form of the Boolean model of the *lac* operon given by Eqs. (1.7). If in those equations we rename the variables M, P, B, C, R, A, A_l, L, L_l to $x1$, $x2$, $x3$, $x4$, $x5$, $x6$, $x7$, $x8$, $x9$ (in this order) and the parameters L_e and G_e to a and g, respectively, and rewrite the equations in a form where the right-hand side is zero, we will obtain[3]

$$\begin{aligned} &x1 + x4 * x5 + x4 = 0 \\ &x1 + x2 = 0, \\ &x1 + x3 = 0 \\ &x4 + (g + 1) = 0 \\ &x5 + x6 * x7 + x6 + x7 + 1 = 0 \\ &x6 + x3 * x8 = 0 \\ &x6 + x7 + x8 + x9 + x8 * x9 + x6 * x8 + x6 * x9 + x6 * x8 * x9 = 0 \\ &x8 + (g + 1) * a * x2 = 0 \\ &x9 + (g + 1) * (x8 + a * x8 + a) = 0. \end{aligned} \tag{1.13}$$

We now need to solve this system four times, using all four combinations of $a = 0,1$ and $g = 0,1$ for the parameter values. Using *SAGE* we need to "evaluate" the following command lines:

```
P.<x1,x2,x3,x4,x5,x6,x7,x8,x9>=PolynomialRing(GF(2),
9, order = 'lex').
```

We use GF(2) here since we want to find the solutions of the system of equations from Eqs. (1.13) over the field $\{0, 1\}$, often referred to as the Galois field of two elements (hence the notation). We also use nine to indicate the number of variables. After setting $a = g = 0$ in the polynomials from Eqs. (1.13), we enter the set of functions:

[3]To get the system in this form we used the facts that, over the field {0,1}, 1 + 1 = 0, −1 = 1, and a − b = a + b.

```
I=ideal(x1+x4*x5+x4,x1+x2,x1+x3,x4+1,x5+x6*x7+x6+x7
+1,x6+x3*x8,
x6+x7+x8+x9+x8*x9+x6*x8+x6*x9+x6*x8*x9, x8, x9+x8)
I.groebner_basis().
```

Evaluating, *SAGE* returns the following Groebner basis for the set of functions in the left-hand sides of Eqs. (1.13):

```
[x1, x2, x3, x4 + 1, x5 + 1, x6, x7, x8, x9].
```

Thus, the system of equations from Eqs.(1.13) has the same solution set as the system of equations

$$\begin{array}{lll} x1 = 0 & x4 + 1 = 0 & x7 = 0 \\ x2 = 0 & x5 + 1 = 0 & x8 = 0 \\ x3 = 0 & x6 = 0 & x9 = 0. \end{array}$$

This gives the steady state $(M, P, B, C, R, A, A_l, L, L_l) = (x1, x2, x3, x4, x5, x6, x7, x8, x9) = (0, 0, 0, 1, 1, 0, 0, 0, 0)$ for $L_e = a = 0, G_e = g = 0$.

Exercise 1.20. Use *SAGE* to continue the computations from Example 1.10 and show that the Boolean model of the *lac* operon from Eqs. (1.8) has the following fixed points for the remaining combinations of parameter values: (1) For : $a = 0, g = 1$: $(M, P, B, C, R, A, A_l, L, L_l) = (0, 0, 0, 0, 1, 0, 0, 0, 0)$; (2) For : $a = 1, g = 1$: $(M, P, B, C, R, A, A_l, L, L_l) = (0, 0, 0, 0, 1, 0, 0, 0, 0)$; (3) For : $(a = 1, g = 0$: $(M, P, B, C, R, A, A_l, L, L_l) = (1, 1, 1, 1, 0, 1, 1, 1, 1)$. ▽

The results from Example 1.10 and Exercise 1.20 show that the Boolean model of the *lac* operon from Eqs. (1.8) has the right qualitative behavior, predicting that the operon is On only when external lactose is available and external glucose is not. In this case all variables of the model, except for the repressor protein, are present. When glucose is available, the operon is Off. All fixed points are biologically feasible.

Now that we understand how fixed points may be found as solutions of systems of polynomial equations, we can use DVD again to analyze the model and compare the results to those from Example 1.10 and from Exercise 1.20. In DVD you can either enter and run the model four times for the four different combinations of the parameter values or use the approach we took in Example 1.5 (2) and include an additional "variable" for each of the parameters with an update rule that keeps that variable constant for all time steps. The analysis in DVD confirms the fixed points from Example 1.10 and from Exercise 1.20 and affirms that the state space does not contain limit cycles.

1.5 CONCLUSIONS AND DISCUSSION

Utilization of lactose by the bacterium *E.coli* requires the production of two proteins, the transporter lactose permease and the enzyme β-galactosidase. Efficient use of cellular energy and materials requires that the bacterium only make these two proteins when needed. The regulatory mechanism of the *lac* operon ensures the repression of these two proteins when glucose, the preferred energy source, is present and the

coordinated production of these two proteins when lactose alone is present. In this chapter we show how Boolean networks can be used to model the control mechanisms of the *lac* operon system. The chapter can be used as an introduction to mathematical modeling without calculus. A brief primer of Boolean algebra is included, so virtually no mathematical prerequisites are required. We introduce specialized web-based software for testing and analyzing the models, which is a convenient way to separate theory from applications. For readers interested in the theoretical underpinnings, the chapter provides an online appendix (Appendix 1) that outlines the use of Groebner bases as it pertains to solving systems of polynomial equations.

The models considered in this chapter are relatively simple and capture only the most significant qualitative behavior of the *lac* operon – its ability to turn on its lactose utilization mechanism when glucose is absent from the external medium and lactose is present. It is clear even from these simple models, that threshold values separating concentrations with Boolean value 0 from those with Boolean value 1 need to be selected carefully based on the biology. We generally assume that a value of 0 and 1 indicate, respectively, "low" and "high" concentrations, where "low" does not always mean a concentration of zero. For concentrations of mRNA, *lac* permease, and β-galactosidase, as well as for other proteins and enzymes involved in the *lac* operon control, which increase by a factor of thousands when the operon is turned on in comparison with their baseline concentrations, such threshold values can easily be selected. The concentrations of lactose, allolactose, and glucose, however, depend on the external environment and change more gradually. For those, "low" could truly mean "absent" and medium-level concentrations are biologically completely feasible. In the model from Eqs. (1.8), the authors take care to introduce designated Boolean variables for low lactose and allolactose to ensure that "low" means "some but not zero" [22]. Other models (see, e.g., [26,27] and Chapter 2 of this volume) consider Boolean frameworks within which it is possible to distinguish between low, medium, and high lactose concentrations.

Focusing on the medium range of lactose concentration is of particular interest since the lactose operon has been shown to exhibit *bistability* for medium levels of lactose. Bistability is the ability of a system to settle in one of two different fixed points under the same set of external conditions. Which of these fixed points the system will reach depends on its history. It has been known since the 1950s [28] that for medium lactose concentrations, both induced and uninduced cells can be observed in a population of *E.coli*. Cells grown in an environment poor in extracellular lactose will likely remain uninduced for medium lactose levels while cells grown in a lactose-rich environment will likely retain their induced state. Chapter 2 of this volume examines Boolean network models of the *lac* operon that can capture the bistability property of the system.

Boolean models are important from an educational perspective since they require only a minimal mathematics background. At the introductory level, the construction of a simple model may essentially amount to translating the system's interactions represented by a biology "cartoon" into a directed graph (wiring diagram), followed by a subsequent translation into logical expressions (the update rules). This makes

Boolean models ideal for an early (below-calculus level) introduction to mathematical models, removing the need for calculus or other mathematical prerequisites. For mathematics students, such models can be introduced in low-level finite mathematics or discrete mathematics courses and used to provide an early demonstration of the important link between mathematics and biology.

At the more advanced mathematics level, Boolean models can be generalized to finite dynamical systems (FDS) and used to provide an introduction to some serious theoretical mathematical questions or as a path to questions appropriate for student research projects. In this chapter we examined the questions of determining the fixed points of FDS. This leads to a question of solving systems of polynomial equations over a finite field, for which the theory of Groebner bases provides a practical solution. The algorithm is essentially a generalization of the well-known process of Gaussian elimination for solving systems of linear equations.

The actual implementation of the method requires the use of specialized software (as even the verification that a given set of polynomials is a Groebner basis for an ideal is labor intensive and virtually impossible to do by hand). Although there are several open-source computational algebra systems that compute Groebner bases (e.g., Macaulay 2, MAGMA, CoCoA, SINGULAR, and others), most such systems require download and installation. For the purposes of this chapter the web-based SAGE interface to Macaulay 2 is appropriate, as it requires only a few straightforward commands. The students can then focus on the output and its interpretation with regard to the question of solving polynomial systems of equations.

For mathematics students, we see the use of the chapter material to be threefold. On one hand, it introduces them to a new modeling approach that is currently not taught in any of the mainstream undergraduate mathematics courses. On the other hand, FDS models provide links to important mathematical theory and results in abstract algebra and algebraic geometry that can be further pursued in advanced-level courses or as independent student research projects. Finally, the topic provides evidence for the important connections between modern biology and modern mathematics, and can be used to highlight mathematical and systems biology as career paths for mathematics majors.

For biology students, the chapter can be used as an introduction to mathematical modeling without calculus prerequisites. Our experience indicates that the "just-in-time" approach for developing the necessary mathematical concepts as a way to formalize specific aspects of the biology works well for Boolean models. It allows students to focus on the logical links that determine the variable interactions instead of on the detailed kinetics needed for calculus-based models. Concurrent or subsequent introduction to such models in calculus or differential equations courses will allow students to reinforce the conceptual framework, further improve their mathematical sophistication, and solidify the retention of basic ideas.

Acknowledgments

The authors gratefully acknowledge the support of the National Science Foundation under the Division of Undergraduate Education award 0737467.

1.6 SUPPLEMENTARY MATERIALS

The online appendix, additional files, and computer code associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/B978-0-12-415780-4.00021-1 and from the volume's website http://booksite.elsevier.com/9780124157804

References

[1] Kauffman S. Metabolic stability and epigenetics in randomly constructed gene nets. J Theor Biol 1969;22:437–467.
[2] Jacob F, Perrin D, Sanchez C, Monod J. L'Operon: groupe de gène à expression par un operatour. C.R. Seances Acad Sci 1960;250:1727–1729.
[3] Jacob F, Monod J. Genetic regulatory mechanisms in the synthesis of proteins. J Mol Biol 1961;3:318–356.
[4] Cheng B, Fournier RL, Relue PA. The inhibition of Escherichia coli lac operon gene expression by antigene oligonucleotidesmathematical modeling. Biotechnol Bioeng 2000;70:467–472.
[5] Doi A, Fujita S, Matsuno H, Nagasaki M, Miyano S. Constructing biological pathway models with hybrid functional Petri nets. In Silico Biology 2004;4:271–91.
[6] Farina M, Prandini M. A mathematical model for genetic regulation of the lactose operon. In: Bemorad, A. et al., editors. Hybrid systems: computation and control, Lecture Notes in Computer Science, Vol. 4416; 2007 p. 693–7.
[7] Romero-Campero FJ, Pérez-Jiménez MJ. Modelling gene expression control using P systems: the lac operon, a case study. Biosystems 2008;91:438–457.
[8] Setty Y, Mayo AE, Surette MG, Alon U. Detailed map of a cis-regulatory input function. Proc Natl Acad Sci USA 2003;100:7702–07.
[9] Tian, T., Burrage, K. A mathematical model for genetic regulation of the lactose operon. In: Gervasi, O. et al. editors. Computational science and its applications—ICCSA 2005, Lecture Notes in Computer Science, vol. 3481; 2005, p. 1245–53.
[10] van Hoek M, Hogeweg P. The effect of stochasticity on the lac operon: an evolutionary perspective. PLoS Comput Biol 2007;3:e111.
[11] Muller-Hill B. The lac operon: a short history of a genetic paradigm. Berlin: Walter De Gruyter, Inc.; 1996.
[12] Vastani H, Jarrah AS, Laubenbacher R, Visualization of dynamics for biological networks. http://dvd.vbi.vt.edu/dvd.pdf.

[13] Russell PJ. iGenetics. Benjamin Cummings, San Francisco: A Molecular Approach; 2006.
[14] Santillán M, Mackey MC, Zeron E. Origin of bistability in the lac operon. Biophys J 2007;92:3830–3842.
[15] Yildrim N, Mackey MC. Feedback regulation in the lactose operon: a mathematical modeling study and Comparison with Experimental Data. Biophys J 2003;84:2841–51.
[16] Yildirim N, Santillán M, Horike D, Mackey MC. Dynamics and bistability in a reduced model of the lac operon. Chaos 2004;14:279–292.
[17] Laubenbacher, R. Algebraic models in systems biology. In: Anai H, Horimoto K, editors. Algebraic Biology, Universal Academy Press: Tokyo; 2005, p. 33–40.
[18] Allen E, Fetrow J, Daniel L, Thomas S, John D. Algebraic dependency models of protein signal transduction networks from time-series data. J Theor Biol 2006;238:317–330.
[19] Delgado-Eckert E. An algebraic and graph theoretic framework to study monomial dynamical systems over a finite field. Complex Systems 2009;19:307–328.
[20] Jarrah AS, Laubenbacher R, Veliz-Cuba A. The dynamics of conjunctive and disjunctive Boolean network models. Bull Math Biol 2010;72:1425–47.
[21] Veliz-Cuba A, Jarrah AS, Laubenbacher R. Polynomial algebra of discrete models in systems biology. Bioinformatics 2010;26:1637–1643.
[22] Stigler B, Veliz-Cuba A, Network topology as a driver of bistability in the lac operon. 2008, arXiv:0807.3995. http://arxiv.org/abs/0807.3995
[23] Saez-Rodriguez J, Simeoni L, Lindquist JA, Hemenway R, Bommhardt U, Arndt B, Haus U, Weismantel R, Gilles ED, Klamt S, Schraven B. A logical model provides insights into T cell receptor signaling. PLOS Comp Biol 2007;3:e163.
[24] Astronomers count the stars. BBC News, July 22, 2003. http://news.bbc.co.uk/2/hi/science/nature/3085885.stm. Retrieved June 25, 2012.
[25] Laubenbacher R, Sturmfels B. Computer algebra in systems biology. Am Math Monthly 2009;116:882–891.
[26] Hinkelmann F, Laubenbacher R. Boolean models of bistable biological systems. Discrete and continuous dynamical systems 2011;4:1442–56.
[27] Veliz-Cuba A, Stigler B. Boolean models can explain bistability in the lac operon. J Comput Biol 2011;18:783–94.
[28] Novick A, Weiner M. Enzyme induction as an all-or-none phenomenon. Proc Natl Acad Sci U S A 1957;43:553–566.