Visual Programming for Modeling and Simulation of Biomolecular Regulatory Networks*

Rajeev Alur[1], Calin Belta[1], Franjo Ivančić[1], Vijay Kumar[1], Harvey Rubin[1], Jonathan Schug[1], Oleg Sokolsky[1], and Jonathan Webb[2]

[1] Hybrid Systems Group
University of Pennsylvania
Philadelphia, PA 19104, USA
[2] BBN Technologies
Cambridge, MA 02138, USA

Abstract. In this paper we introduce our new tool BioSketchPad that allows visual progamming and modeling of biological regulatory networks. The tool allows biologists to create dynamic models of networks using a menu of icons, arrows, and pop-up menus, and translates the input model into Charon, a modeling language for modular design of interacting multi-agent hybrid systems. Hybrid systems are systems that are characterized by continuous as well as discrete dynamics. Once a Charon model of the underlying system is generated, we are able to exploit the various analysis capabilities of the Charon toolkit, including simulation and reachability analysis. We illustrate the advantages of this approach using a case study concerning the regulation of bioluminescence in a marine bacterium.

1 Introduction

We now know that approximately 30,000 to 40,000 genes control and regulate the human body. The recent completion of a rough draft of the human genome and the complete sequence of *Drosophila melanogaster*, *Caenorhabditis elegans*, *Mycobacterium tuberculosis*, and numerous other sequencing projects provide a vast amount of genomic data for further refinement and analysis [17]. These landmarks in human scientific achievement promise remarkable advances in our understanding of fundamental biological processes. To achieve this goal, we must develop the ability to model, analyze, and predict the effect of the products of specific genes and genetic networks on cell and tissue function [12].

Traditional models and simulations of metabolic and cellular control pathways are based on either continuous or discrete dynamics [6, 15]. However, many important systems in biology are *hybrid* – they involve both discrete and continuous dynamics. At the molecular level, the fundamental process of inhibitor proteins turning off the transcription of genes by RNA polymerase reflects a switch

* This research was supported in part by NSF grants CDS-97-03220 and and EIA-01-30797, and DARPA grant F30602-01-2-0563.

between two continuous processes. This is perhaps most clearly manifested in the classic genetic switch observed in the λ-phage [16], where we see two distinct behaviors, *lysis* and *lysogeny*, each with different mathematical models. At the cellular level, we can best describe cell growth and division in a eukaryotic cell as a sequence of four processes, each being a continuous process being triggered by a set of conditions or events [14]. At the inter-cellular level, we can even view cell differentiation as a hybrid system [9].

In all of these examples, a hybrid approach that combines elements of discrete and continuous dynamics is necessary to model, analyze, and simulate the system. This paper addresses the modeling of networks of biochemical reactions, and the control of molecular and cellular functions with genetic regulatory apparatus, and a set of theories, algorithms, and methodologies that allows biologists to analyze and characterize such systems.

2 Background

Our approach to modeling the different elements (biomolecules, cells, proteins) and their interactions is based on modern concepts in software engineering and control theory derived from the literature on hybrid systems [1].

A hybrid system consists of a set of *hybrid automata*. A hybrid automaton is characterized by a *continuous state* $x \in \mathbb{R}^n$ and a collection of *discrete modes*. Each mode consists of a set of *ordinary differential equations* (ODEs) that govern the evolution of the continuous state x and a set of *invariants* that describe the conditions under which the ODEs are valid. We can write the ODEs of a particular mode as $\dot{x} = f(x, z)$, where $z \in \mathbb{R}^p$ is the information from other hybrid automata running concurrently as part of the overall hybrid system. The definition of a hybrid automaton includes *transitions* among its modes. A transition specifies source and destination modes, an enabling condition called *guard*, and the associated discrete *update* of variables.

As an illustrative example, consider the description of a simple hybrid system in Figure 1. It consists of one hybrid automaton that contains two discrete modes q_1 and q_2, and the continuous variable x which evolves under the differential equation $\dot{x} = f_1(x)$ in discrete mode q_1, and $\dot{x} = f_2(x)$ in mode q_2. The invariant sets associated with the locations q_1 and q_2 are $g_1(x) \leq 0$ and

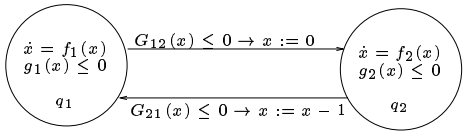


Fig. 1. A simple hybrid system.

$g_2(x) \leq 0$, respectively. The hybrid system continuously evolves in discrete mode q_1 according to the differential equation $\dot{x} = f_1(x)$, as long as x remains inside

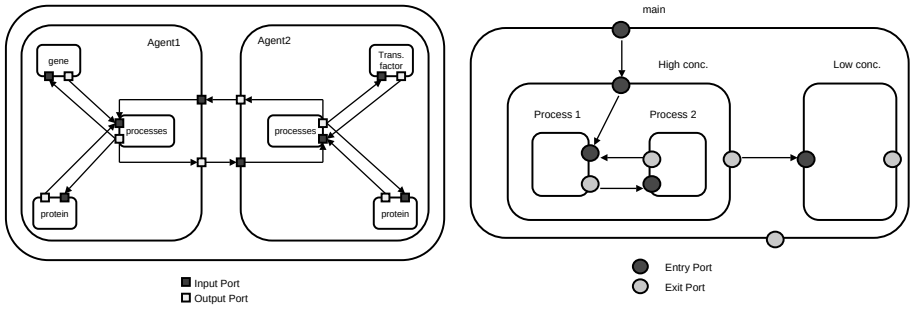


Fig. 2. The architectural hierarchy in CHARON is based on agents. New agents can be formed by parallel composition of two or more agents (left). The behavioral hierarchy is described by modes. Sequential composition of modes models the switching between behaviors (right).

the invariant set $g_1(x) \leq 0$. If during the continuous flow, it happens that x belongs in the guard set $G_{12}(x) \leq 0$, then the transition from q_1 to q_2 is enabled. If the transition is enabled, the system can switch from the mode q_1 to q_2. The variable x is reset to zero and then evolves according to the differential equation $f_2(x)$.

Hybrid system models are increasingly being used for modeling embedded systems, in particular for automotive control systems, avionics and robotics. Modern object-oriented design paradigms such as *Unified Modeling Language* (UML) allow specifications of the architecture and control at high levels of abstraction in a modular fashion [5]. Emerging tools such as RationalRose (see `www.rational.com`) support modeling, simulation, and code generation. Tools such as MATLAB and SIMULINK (see `www.mathworks.com`) allow the modeling and simulation of systems with models of continuous dynamics combined with state-machine-based models of discrete states.

All these paradigms and tools are directly applicable to modeling and analysis of biological regulatory networks. In the next two sections, we'll describe CHARON, a programming language for modeling such systems, and BIOSKETCHPAD, an easy-to-use, interactive tool that allows users to create CHARON models of regulatory networks.

3 Modeling Hybrid Systems in CHARON

We have developed the programming language CHARON [1], for modeling and analyzing hybrid systems (see Figure 2). The language incorporates ideas from concurrency theory (languages such as CSP [13]), object-oriented software design notations (such as Statecharts [11] and UML [5]), and formal models for hybrid systems. The key features of CHARON are:

Architectural hierarchy. The building block for describing the system architecture is an *agent* that communicates with its environment via shared variables. The language supports the operations of *composition* of agents to model concurrency, *hiding* of variables to restrict sharing of information, and *instantiation* of agents to support reuse.

Behavior hierarchy. The building block for describing flow of control inside an atomic agent is a *mode*. A mode is basically a hierarchical state machine, that is, a mode can have submodes and transitions connecting them. Variables can be declared locally inside any mode with standard scoping rules for visibility. Modes can be connected to each other only via well-defined entry and exit points. We allow *sharing* of modes so that the same mode definition can be instantiated in multiple contexts. Finally, to support *exceptions*, the language allows group transitions from default exit points that are applicable to all enclosing modes.

Discrete updates. Discrete updates are specified by *guarded actions* labeling transitions connecting the modes. Actions can have calls to externally defined Java functions which can be used to write complex data manipulations. It also allows us to mimic stochastic aspects through randomization.

Continuous updates. Some of the variables in CHARON can be declared *analog*, and they flow continuously during continuous updates that model passage of time. The evolution of analog variables can be constrained in three ways: *differential* constraints (e.g. by equations such as $\dot{x} = f(x, u)$), *algebraic* constraints (e.g. by equations such as $y = g(x, u)$), and *invariants* (e.g. $|x - y| \leq \varepsilon$) which limit the allowed durations of flows. Such constraints can be declared at different levels of the mode hierarchy.

4 BIOSKETCHPAD

The BIOSKETCHPAD (BSP) is an interactive tool for modeling and designing biomolecular and cellular networks with a simple, easy to use, graphical front end leveraging powerful tools from control theory, hybrid systems, and software engineering. Models developed using BSP can be automatically converted to CHARON input files.

4.1 Nodes in BIOSKETCHPAD

The elements of a BSP model diagram are nodes and connecting edges. The nodes are either species or processes. Processes include chemical reactions, regulation, and cell growth. The edges describe relations between species and processes.

Species Nodes. A species node is identified by a tuple of parameters which are listed below. The different parameters support the kinds of common biological distinctions between similar species. Thus Ca and $Ca^{+}2$ will have the same name value, but have different values for charge. A particular tuple of parameters can be represented by one or more graphical instance of a species node.

The *name* of the species, i.e., "Ca", "alcohol dehydrogenase", or "notch" is one parameter of a species node. Other parameters are the *type* of the species such as "gene" or "protein", the *physical location* of the species, which may be "cell membrane" or "cell nucleus" reflecting different physical areas of interest where concentrations of a particular compound may be different. In addition, the *n-mer* polymerization of the species, the *state*, and the *electrical charge* of a species are parameters. Finally, when appropriate, the user can specify an *initial concentration* for a species which will be used in simulation runs. The initial concentration can be specified in a consistent unit system for the model as a whole or the particular species.

Reaction Nodes. A reaction node represents some kind of interaction which may alter the concentration of one or more species. Each reaction node must have at least one node attached to it. Further constraints are placed on attached nodes for specific types of reactions. Species nodes attached to a reaction are considered either input or output nodes.

The *type* of the reaction represented by the node may, for example, be "transformation" or "transcription." Some types of reactions may be restricted in the types of species they may operate on, specific species properties may be required, or there may be restrictions on reaction geometry. Sources and sinks are special types of reaction nodes.

Regulation Nodes. A reaction node is used to modulate the rate of a reaction by the concentration of one or more species. It must have at least one output node which must be a reaction and at least one input species node.

There are currently three functional forms for regulation nodes, "weighted sum," "product," and "tabular" and they represent the mathematical operation performed for the node. Species inputs reference the concentration of the species. Regulation node inputs reference the computed value of the regulation function.

4.2 Models for Transformation, Transcription and Translation

Rate laws for transformation, transport, transcription and translation reaction types are the basic reaction formulations in the BSP. Rate laws can be reversible or irreversible, and can be regulated by specified species.

The BSP allows the specification of a variety of rate laws. For the sake of simplicity, we will consider the case of a *reversible mass action* (see Figure 3). For a reaction of the form

$$aA + bB \leftrightarrow cC + dD \tag{1}$$

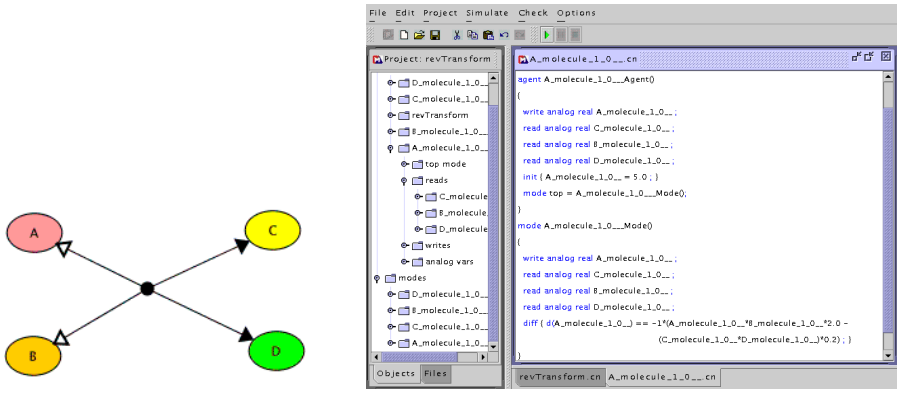
the reaction rate is given by

$$\nu = k_f * [A]^{\alpha} * [B]^{\beta} - k_r * [C]^{\gamma} * [D]^{\delta}, \tag{2}$$

where k_f and k_r stand for the forward and reverse rate constant, and $[A]$ and $[B]$ are the concentrations of species A and B. The constants a, b, c, and d are the stoichiometric coefficients, whereas α, β, γ and δ are called order of reaction

and are determined experimentally. Terms in the ODE for the concentrations of reaction participants are constructed as follows:

$$\begin{aligned} d[A]/dt &= -a * \nu, \\ d[B]/dt &= -b * \nu, \\ d[C]/dt &= c * \nu, \text{ and} \\ d[D]/dt &= d * \nu. \end{aligned} \tag{3}$$



(a) (b)

Fig. 3. A simple reversible mass action model with four species. (a) The model as a snapshot of BSP. (b) The automatically generated CHARON model representing the BSP input model in (a). The CHARON toolkit editor on the right shows the model of species A. The left panel of the CHARON toolkit is the project window showing the internal tree representation of the CHARON model.

The BSP also allows the definition of *tabulated functions*. Tabulated functions allow the specification of different behaviors in various parts of the state space. These functions introduce *hybrid* behavior into the underlying CHARON models. The general form of a tabulated functional form mass action is shown below for a reversible reaction of the form shown previously.

$$\nu = k_f * f_1([A],[B],[C],[D]) * [A]^{\alpha} * [B]^{\beta} - k_r * f_2([A],[B],[C],[D]) * [C]^{\gamma} * [D]^{\delta}. \tag{4}$$

The tabulated function may also include the concentration of any regulating species as an independent variable.

In addition to the reactions described above, BSP also models *irreversible mass action*, *regulated mass action*, *Michelis-Menton reactions*, and *cell growth*.

The BSP translates such an input model automatically into an equivalent CHARON model, which is shown in Figure 3 (b).

We next introduce a case study in which we show how a fairly complex biological phenomenom can be modeled using BSP and CHARON.

5 Case Study: Luminescence Regulation in V. Fischeri

Vibrio fischeri is a marine bacterium which can be found both as free-living organism and as a symbiont of some marine fish and squid. As a free-living organism, *V. fisheri* exists at low densities and appears to be non-luminescent. As a symbiont, the bacteria live at high densities and are, usually, luminescent.

5.1 Description of the Regulatory System

The luminescence in *V. fischeri* is controlled by the transcriptional activation of the *lux* genes. The *lux* regulon is organized in two transcriptional units (see Figure 4). The leftward operon contains the *luxR* gene encoding the protein[1] LuxR, a transcriptional regulator of the system. The rightward operon contains seven genes *luxICDABEG*. The transcription of the *luxI* gene results in the production of protein LuxI. This protein is required for endogenous production of *autoinducer*, Ai, a small membrane-permeant signal molecule (acyl-homoserine lactone). The genes *luxA* and *luxB* code for the luciferase subunits, which in turn are responsible for luminescence. *luxC*, *luxD*, and *luxE* code for polypeptides of the fatty-acid reductase, which generates aldehyde substrate for luciferase. Along with LuxR and LuxI, cAMP receptor protein (CRP) plays an important role in controlling luminescence. The network of biochemical reactions in the cell is as follows: The autoinducer *Ai* binds to protein LuxR to form a complex *Co* which binds to the *lux box*. The *lux box* is in the middle of a regulatory region between the two transcriptional units (operons). This region also contains a binding site for CRP. The transcription from the *luxR* promoter is activated by the binding of CRP to its binding site, and the transcription of the *luxICDABEG* by the binding of *Co* to the *lux box*. However, growth in the levels of *Co* and cAMP/CRP inhibit *luxR* and *luxICDABEG* transcription, respectively.

5.2 BIOSKETCHPAD Model and Simulation in CHARON

The regulatory network described above can be easily drawn in the BSP as shown in Figure 4. Notice the figure shows different icons for the gene, the transcribed mRNA, and the translated protein, all of which play an important role in the network. The transcriptional regulation of the two operons by CRP and *Co* is modeled using the tabular function approach, as described before.

The tool then translates the graphical model into equivalent CHARON code. Each species (mRNA, protein, complex) is modeled by an agent. The complete

[1] We use italics (*e.g.*, *luxR*) to indicate the genes and plain font to denote the protein expressed by the gene (*e.g.*, LuxR)

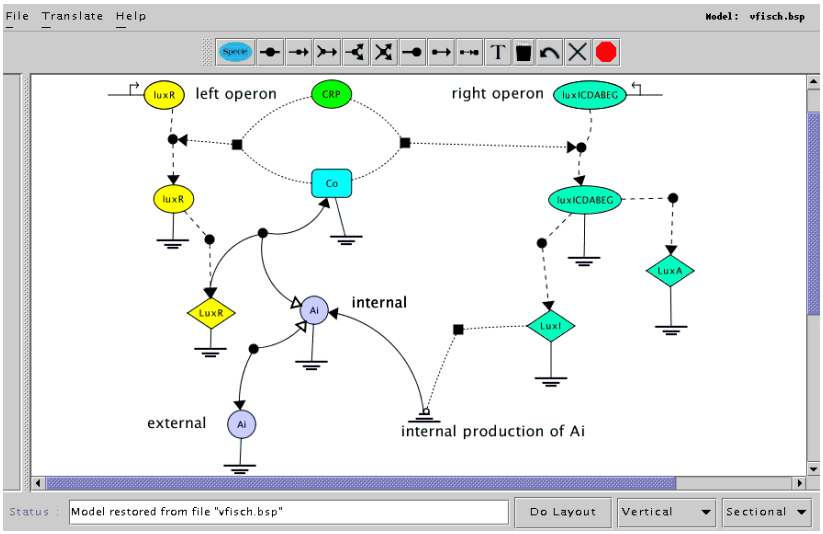


Fig. 4. Snapshot of the BioSketchPad model of the luminescence regulatory system in *V. fischeri*. The top line represents buttons used to perform standard operations, such as "create a new species" or "add a two-to-one edge". The bottom line holds a status message and buttons to perform automatic layout operation.

model is the parallel composition of all the agents. Each agent has only one mode (biological behavior) except for mRNAs luxR and luxICDABEG, which are regulated by signals given in tabular form. The differential equations describing the dynamics of the system are computed according to the edges in the model and the parameters specified by the user and translated into Charon.

The above model was simulated starting from zero initial conditions for all the species except for external autoinducer Ai_e, which was set at 100nM. As expected, the non-zero initial value of external autoinducer determines an increase in the concentration of internal autoinducer (see Figure 5). The positive feedback loop LuxR - Ai_i turns on the luminescence gene. For the sake of brevity, we omit the parameters used during this simulation run and instead refer the reader to details on the website (`www.cis.upenn.edu/biocomp`).

6 Discussion

In this paper we introduced a new tool BioSketchPad that allows visual programming and modeling of biological regulatory networks through an easy-to-use interactive, icon-based interface. BioSketchPad allows the automatic generation of Charon models. The underlying formal semantics of Charon models allow symbolic in addition to more traditional numerical analysis. We will give a brief overview of our ongoing research directed toward the formal analysis of hybrid systems.

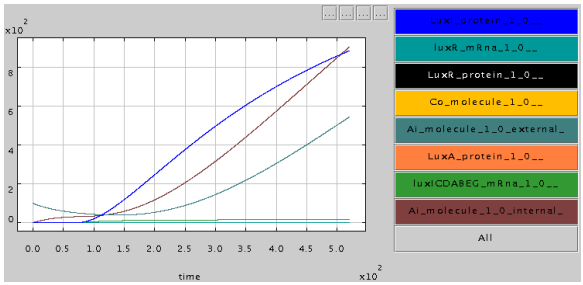


Fig. 5. Snapshot of CHARON showing the time-evolution of the concentrations in nanomolars. More details are available in online papers at `www.cis.upenn.edu/biocomp`.

Simulation with accurate event detection. Simulation of hybrid systems is challenging because it is difficult to develop an efficient integration algorithm that also guarantees the detection of transitions between modes. The commonly used technique of just checking the value of the guards at integration points can cause the simulator to miss critical events. It has been shown that such inaccuracies can lead to grossly inaccurate simulations due to the discontinuous nature of hybrid systems. We have developed a method [8] which is guaranteed to detect enabling of all transitions, and is at least as efficient as conventional, adaptive integration algorithms.

Multirate simulation. Many systems naturally evolve on different time scales. Traditional numerical integration methods force all coupled differential equations to be integrated using the same step size. The idea behind multirate integration method is to use larger step sizes for the slow changing sets of differential equations and smaller step sizes for the differential equations evolving on the fast time scale. To implement such a scheme we need to show how to accommodate coupling between the sets of fast and slow equations when they are integrated asynchronously and how to schedule the order of integration. In [7] we introduce a multi-rate algorithm suited to hybrid system simulation.

Predicate Abstraction. Abstraction is emerging as the key to formal verification as a means of reducing the size of the system to be analyzed.The main obstacle towards an effective application of model checking to hybrid systems is the complexity of the reachability procedures which require expensive computations over sets of states. For analysis purposes, it is often useful to abstract a system in a way that preserves the properties being analyzed while hiding the details that are of no interest [3]. We build upon the notion of predicate abstraction [10] for formal analysis of hybrid systems. Using a set of boolean predicates, that are crucial with respect to the property to be verified, we construct a finite partition of the state space of the hybrid system. By using conservative reachability approximations we guarantee that if the property holds in the abstracted system, then it also holds in the concrete system represented by the hybrid system [2].

Reachability Analysis. While reachability analysis for hybrid systems is, in general, intractable, we are working on methods that efficiently partition the state space for *multi-affine*, hybrid systems into discrete states. For example, equation (2) is multi-affine when the order of reaction are unity. In other words, they are affine in each variable. Our preliminary work in this direction is described in a forthcoming paper [4].

We envision reporting on a more complete toolbox with the software tools described above, tightly integrated with BioSketchPad in forthcoming publications. We believe such a toolbox can have a significant impact on post-genomics biology research.

References

1. R. Alur, T. Dang, J. Esposito, R. Fierro, Y. Hur, F. Ivančić, V. Kumar, I. Lee, P. Mishra, G. Pappas, and O. Sokolsky. Hierarchical hybrid modeling of embedded systems. In *Embedded Software, First Intern. Workshop*, LNCS 2211. 2001.
2. R. Alur, T. Dang, and F. Ivančić. Reachability analysis of hybrid systems via predicate abstraction. In *Hybrid Systems: Computation and Control, Fifth International Workshop*, LNCS 2289, pages 35–48. Springer-Verlag, March 2002.
3. R. Alur, T. Henzinger, G. Lafferriere, and G. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88(7):971–984, July 2000.
4. C. Belta, L. Habets, and V. Kumar. Control of multi-affine systems on rectangles with applications to hybrid biomolecular networks. *CDC 2002*, Dec. 2002.
5. G. Booch, I. Jacobson, and J. Rumbaugh. *Unified Modeling Language User Guide.* Addison Wesley, 1997.
6. M. Elowitz and S. Leibler. Asynthetic oscillatory network of transciptional regulators. *Nature*, 403:335–338, January 2000.
7. J. Esposito and V. Kumar. Efficient dynamic simulation of robotic systems with hierarchy. In *Intl. Conf. on Robotics and Automation*, pages 2818–2823, 2001.
8. J. Esposito, V. Kumar, and G. Pappas. Accurate event detection for simulating hybrid systems. In *Hybrid Systems : Computation and Control*, LNCS 2034, 2001.
9. R. Ghosh and C. J. Tomlin. Lateral inhibition through delta-notch signaling: A piecewise affine hybrid model. In *HSCC*, Rome, Italy, Mar 28-30 2001.
10. S. Graf and H. Saidi. Construction of abstract state graphs with PVS. In *Proc. 9th Intl. Conf. on Computer Aided Verification*, LNCS 1254, 1997.
11. D. Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8:231–274, 1987.
12. L. H. Hartwell, J. J. Hopfield, S. Leibler, and A. W. Murray. From molecular to modular cell biology. *Nature*, 402((6761 Suppl)):C47–52, December 1999.
13. C.A.R. Hoare. *Communicating Sequential Processes.* Prentice-Hall, 1985.
14. B. Lewin. *Genes VII.* Oxford University Press, 2000.
15. P. Mendes and D. B. Kell. Non-linear optimization of biochemical pathways: applications to metabolic engineering and parameter estimation. *Bioinformatics*, 10:869–883, 1998.
16. M. Ptashne. *A Genetic Switch: Phage λ and Higher Organisms.* Cell Press and Blackwell Science, 1992.
17. J. C. Venter et al. The sequence of the human genome. *Science*, 291(5507):1304–51., 2001.