Modeling and Analyzing Biomolecular Networks

The authors argue for the need to model and analyze biological networks at molecular and cellular levels. They propose a computational toolbox for biologists. Central to their approach is the paradigm of hybrid models in which discrete events are combined with continuous differential equations to capture switching behavior.

We now know that approximately 30,000 to 40,000 genes effectively 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.[1] 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.[2]

Traditional models and simulations of metabolic and cellular control pathways are based on either continuous or discrete dynamics.[3–5] However, many important biological systems 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 between two continuous processes.[6] This is perhaps most clearly manifested in the classic lambda switch system, where we see two biological processes. At the cellular level, we can best describe cell growth and division in a eukaryotic cell as a sequence of four processes, each one a continuous process triggered by a set of conditions.[7] At the intercellular level, we can even view cell differentiation as a hybrid system.[8]

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's richness. To understand how a network of biochemical reactions implements and controls cellular functions and the genetic regulatory apparatus, we must develop a new set of theories, algorithms, and methodologies that combine the two fundamentally different ways of characterizing such systems.

We advocate modeling biological systems as stochastic, networked hybrid systems that consist of discrete and continuous components with complex interactions.[9–11] Even in many continuous biological systems characterized by differential equations, a hybrid model offers a computationally tractable approach to modeling, analysis, and synthesis. Networks model the in-

1521-9615/02/$17.00 © 2002 IEEE

Rajeev Alur, Calin Belta, Vijay Kumar, Max Mintz, George J. Pappas, Harvey Rubin, and Jonathan Schug
University of Pennsylvania

teraction between hybrid subsystems. Significant nondeterministic fluctuations in the exogenous variables and structural components within the dynamic system that we cannot model deterministically underscore the need to consider stochastic models for such systems. We should adopt a similar perspective for complex, computer-controlled electromechanical systems. Robotics, avionics, and embedded systems are intrinsically hybrid.[12,13] The control software's complexity coupled with the communication networks and their interaction with the physical environment make designing and analyzing such embedded systems a great challenge, particularly in safety-critical applications.[14] No systematic approach to designing and developing such hybrid systems exists today.

In this article, we describe the enabling technologies needed to understand and predict the integrated functions of cellular regulatory networks. We describe the models and abstract principles of organization, design, control, and coordination. We also outline a research agenda that emphasizes hybrid modeling of biological systems, the use of formal methods and algorithms for simulation and analysis, and a software environment for analysis and design.

Modeling biomolecular networks

The genetic circuits and biomolecular networks considered here and elsewhere are remarkably similar to the hybrid systems encountered in engineering. Our approach to modeling the different elements and their interactions is based on modern concepts in software engineering and control theory.

Species and processes

Central to our approach are the concepts of *agents* and *modes*. An agent is a dynamic system that interacts with other agents through well-defined input and output ports. Agents operate concurrently. All species—proteins, cells, and DNA—are dynamic systems and therefore modeled as agents (see Figure 1). We call them *S-agents* to distinguish them from process agents or *P-agents*, which capture the dynamics involved in transcription, translation, protein binding, protein–protein interactions, and cell growth. The inputs of P-agents are the quantities (concentrations or numbers) of different species relevant to the process; outputs are rates. S-agents describe the accumulation or degradation of species in terms of concentration or simply numbers. An S-agent's description necessarily involves differential equations or update equations. These equations take as inputs the rates of different processes and yield species' quantity. As Figure 1 shows, concurrent processes govern the species. The processes communicate with each other and influence each other's behavior. A similar picture occurs at the intercellular level, where we can view cells as S-agents interacting with each other through different processes that capture intercellular signaling or simple growth based on nutrient availability.

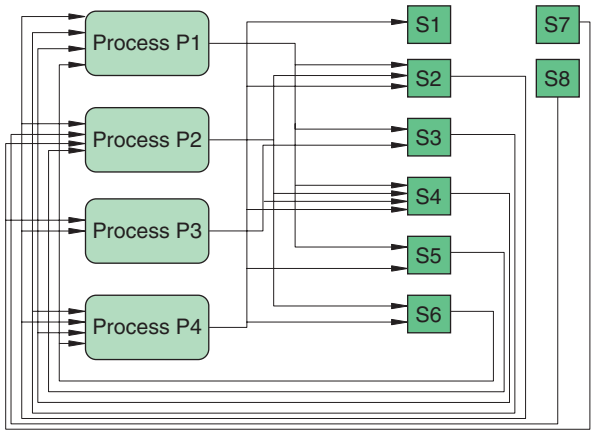

Figure 1. A biomolecular network represented by P-agents and S-agents. The outputs of P-agents are inputs to S-agents; the inputs are outputs of S-agents. External signals can be inputs to P-agents, and outputs of S-agents can signal agents external to this network.

Agents and modes

Each agent is characterized by a continuous state $x \in \Re^n$ and a collection of discrete modes denoted by Q. Each mode is characterized by a set of *ordinary differential equations* that govern the evolution of the continuous state x and a set of invariants that describe the conditions (typically algebraic constraints on the continous state) under which the ODEs are valid. We can write the ODEs as

$$\dot{x} = f_{q_i}(x, z), \quad (1)$$

where $q_i \in Q \subset Z$ is the agent's mode, and $z \in \Re^p$ is the information from other agents available through the input/output ports. A mode's definition includes *transitions* among its submodes. A transition specifies source and destination modes, the enabling condition, and the associated discrete update of variables. Each

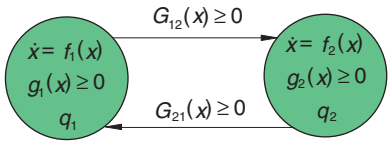

Figure 2. A simple hybrid system.

mode can have submodes; there is generally a hierarchy of modes that typifies most systems.

As an illustrative example, consider the description of the agent in Figure 2. It consists of two discrete modes q_1 and q_2; 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 locations q_1 and q_2 are $g_1(x) \leq 0$ and $g_2(x) \leq 0$, 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 the invariant set $g_1(x) \leq 0$. Transitions between modes are governed by a set of guards called a guard set. Each guard is typically an algebraic constraint on the state. 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. A state jumps from q_1 to q_2, and the system evolves according to the differential equation in mode q_2 as long as the invariant $g_2(x) \leq 0$ is satisfied.

Typed variables—discrete or analog—characterize an agent. Analog variables are updated continuously, whereas discrete variables are updated only on initialization and mode switches. An agent's variables are partitioned into `read`, `write`, and `private` to allow modular specifications. Private variables represent those that are internal to the agent and depict its internal state.

At the lowest level, differential equations such as in Equation 1 can describe the evolution of entities such as proteins. A generic formula for any molecular species (messenger RNA, protein, protein complex, or small molecule) reflects this:[15]

$$dX/dt = synthesis - decay \pm transformation \pm transport. \quad (2)$$

The *synthesis* term represents transcription for mRNA and translation for proteins; the *decay* category represents a first-order degradation process.

Many molecules undergo transformations such as cleavage or ligand-binding reactions—many participate in transport processes such as diffusion through a membrane. However, at low concentrations, we cannot justify continuous rate equations based on concentrations. We must use stochastic process models that predict the numbers of molecules of different species.[16,17]

The hybrid structure in the models arises in two ways. First, a certain activity might be trig-

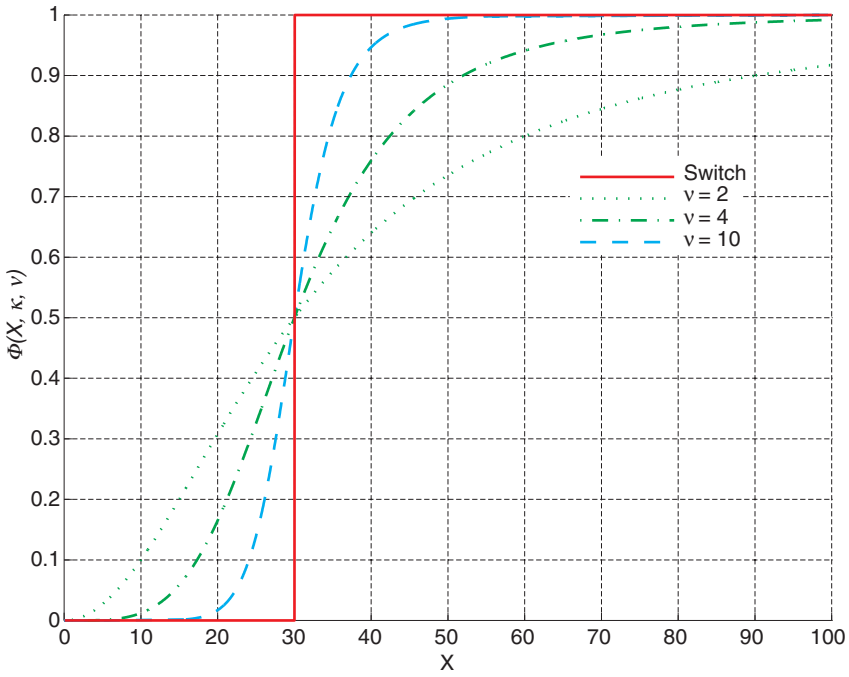

Figure 3. The function $\Phi(X, \kappa, \nu)$ for $\kappa = 30$ and a piecewise continuous abstraction.

gered by an increase in concentration of a protein or other species above a specified threshold. This leads to switching between active and dormant states. Second, different models might be more appropriate at different levels of concentration. The process models must incorporate this ability to switch between differential equations and the discrete update equations used in stochastic models.

Continuous versus hybrid models

A promoter, the region of DNA that is needed for the initiation of gene transcription, typically has multiple regulatory sites, both positive and negative, distributed throughout its regulatory region. We generally model regulation with two functions:[18]

$$\Phi(X, \kappa_{Xm}, \nu_{Xm}) = \frac{X^{\nu_{Xm}}}{\kappa_{Xm}^{\nu_{Xm}} + X^{\nu_{Xm}}}, \quad (3)$$

and

$$\Psi(X, \kappa_{Xm}, \nu_{Xm}) = 1 - \Phi(X, \kappa_{Xm}, \nu_{Xm}), \quad (4)$$

where X is the concentration of some species with a regulatory effect on the transcription of mRNA m, ν_{Xm} is called the cooperativity coefficient, and κ_{Xm} is the concentration of X at which transcription of m is "half-maximally" activated. Figure 3 shows the graph of function Φ, the so-called *sigmoid function*.

The curve describing the regulation of transcription in Figure 3 is merely a convenient way of modeling the turning off of gene expression at low concentrations and the turning on at high concentrations. Little experimental data confirms the sigmoid curve's exact shape in the figure and we lack the data needed to estimate parameters such as κ and ν (as in Equation 3). In the absence of such data, it is simpler to pursue piecewise constant approximations that map the degree of gene transcription activation (inhibition) to intervals of activator (inhibitor) concentration. Figure 3 shows a two-step model that effectively turns off gene expression completely below a certain concentration of the regulator and turns it on completely above that concentration. We can also imagine a more sophisticated n-step approximation based on n experimental data points. From a system-analysis viewpoint, dealing with such piecewise constant functions to describe transcription is definitely advantageous. They let us abstract a nonlinear system as a switched, lower dimensional system that could be easier to analyze because of the sigmoid nonlinearity's absence.

We can use approximations similar to Figure 3 at different levels in the system hierarchy. These approximations lead to abstractions that are conceptually similar and computationally more tractable, but the ability to validate the fidelity of these models through analysis and experimentation is also necessary.

Charon: A programming language for hybrid systems

We developed the programming language Charon[19] for modeling and analyzing hybrid systems (see Figure 4). The language incorporates ideas from concurrency theory (languages such as CSP[20]), object-oriented software design notations (such as Statecharts[21] and UML[22]), and formal models for hybrid systems (such as hybrid automata and hybrid I/O automata[23]).

Charon's key features are

- *Architectural hierarchy*. The building block for describing the system architecture is an agent that communicates with its environment by sharing variables. The language supports composing agents to model concurrency, hiding variables to restrict information sharing, and instantiating agents to support reuse.
- *Behavior hierarchy*. The building block for describing control flow inside an atomic agent is a mode, which is basically a hierarchical state machine. A mode can have submodes and transitions connecting them. Variables are declared locally inside any mode with standard scoping rules for visibility. Modes connect to each other only through well-defined entry and exit points. We allow sharing of modes so that the same mode definition can be instantiated in multiple contexts. To support exceptions, the language allows group transitions from default exit points that apply to all enclosing modes.
- *Discrete updates*. Guarded actions label the transitions that connect modes to specify discrete updates. Actions can have calls to externally defined Java functions, which we can use to write complex data manipulations. They also lets us mimic stochastic aspects through randomization.
- *Continuous updates*. Some of the variables in Charon are analog, and they flow continuously during updates that model the passage of time. We can constrain the evolution of analog variables in three ways: differential constraints (equations such as $\dot{x} = f(x, u)$), algebraic con-

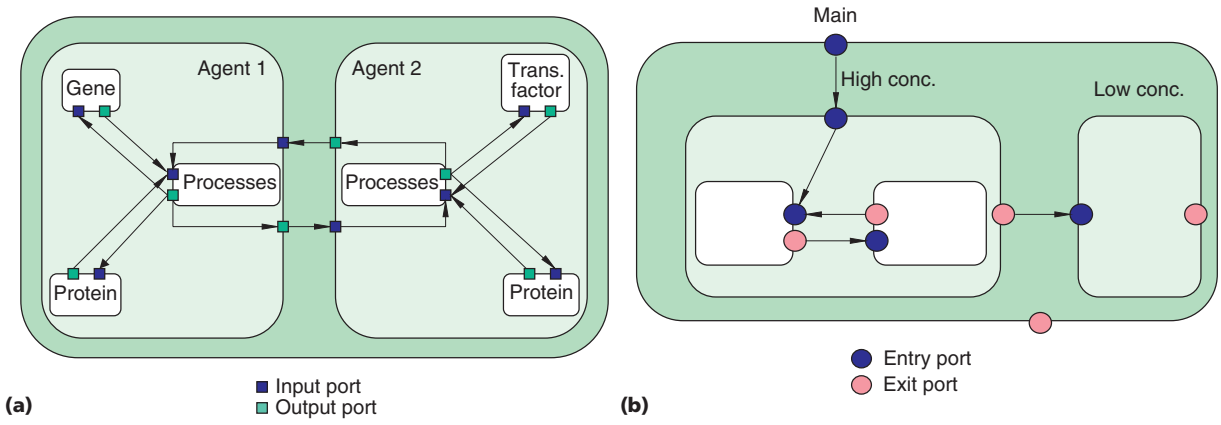

Figure 4. The architectural hierarchy in Charon is based on agents. (a) New agents are formed by the parallel composition of two or more agents. (b) Modes describe the behavioral hierarchy. The sequential composition of modes models the switching between behaviors.

straints (such as $y = g(x, u)$), and invariants (such as $|x - y| \leq \varepsilon$) that limit the allowed flow duration. Such constraints are declared at different levels of the mode hierarchy.

Charon's modular features allow succinct and structured description of complex systems. Languages such as Shift[24] and Stateflow (see www.mathworks.com) support similar features, but in Charon, modularity is not solely apparent in syntax. We are developing analysis tools (such as simulation) to exploit this modularity. Furthermore, Charon has formal foundations supporting compositional refinement calculus, which allows relating different models of the system in a mathematically precise manner. A formal mathematical description lets us develop tools for computing equilibria and for analyzing properties such as stability and reachability.

Software tools for system biology

Recent efforts have addressed the complexity of developing continuous system models and simulators for genetic networks.[25] Thousands of differential equations with hundreds of modes and inequality constraints will characterize hybrid system models for biology and robotics. Accuracy is extremely important in such systems. Unlike continuous or discrete systems, small errors in detecting the violation of unilateral constraints can completely change the state's ensuing time history.

Many important issues arise in the context of hybrid systems simulation. First, accurately simulating systems of differential equations with inequalities of state is important. Failure to detect events can have disastrous effects on the global solution due to the problem's discontinuous nature. Second, complex multiagent systems have multiple spatial and temporal timescales. Multirate numerical integration methods[26] that let us simulate the individual components of a set of coupled ODEs at different rates are extremely relevant. Models properly specified and programmed in Charon will explicitly describe spatiotemporal hierarchy and concurrency in multiagent systems, thus allowing efficient simulation. Finally, we must consider simulating systems of stochastic differential equations.

In our previous work, we developed a control-theoretic approach that correctly detects an event for certain classes of constraints and guarantees that the ODE is never evaluated on the opposite side of the switching surface ($g(x) = 0$).

To exploit spatiotemporal hierarchy, we have considered systems that naturally exhibit a hierarchical structure and a model in a language such as Charon that preserves this hierarchical structure. Commercial packages such as Matlab don't do this. Elsewhere[27] we present a method that exploits hierarchy to improve efficiency without degrading the system's accuracy.

Our ultimate goal is to develop a simulation technique that can simulate each agent asynchronously when the agent is far from the nearest constraint surface, allowing each agent to be integrated with its own largest acceptable step size and increasing the simulation's overall efficiency. As the system of agents approaches constraints

that affect one or more of them, the relevant local time clocks automatically synchronize to properly detect and localize the violation of constraints in state space and in time.[27]

Formal verification

Model checking is emerging as a powerful technique for detecting logical errors, thus ensuring higher safety and reliability for embedded software. In model checking, a high-level model is compared to a correctness requirement to reveal inconsistencies. Model checking is largely automated and relies on an exhaustive state-space analysis of the model. Researchers originally developed reachability and verification tools for finite-state discrete systems; they have recently extended them to special classes of hybrid systems such as timed automata, linear hybrid automata,[28] and piecewise linear systems.

We are interested in the organism's survival in the biological realm where we consider a metabolomechanical system—the cell and the embedded computer—to be the genetically encoded program. This approach to formal analysis lets researchers establish useful properties of models of biomolecular networks and provide useful feedback for designing biological experiments. For example, in a regulatory network, we might hypothesize if the concentration of protein A in the environment stays between c_l and c_h, then the concentration of regulatory protein B stays below an activation threshold δ. Affirmation or violation of such hypotheses by model checking can help us understand the behavioral interdependencies of complex pathways. Similarly, reachability analysis can establish biological properties—for example, we can establish the existence of cellular rhythms or the stability of a mode of operation.

There are many recent and significant advances on computing reachable sets for hybrid systems. You can use HyTech to verify linear hybrid automata. CheckMate and d/dt can handle more complex systems.[29] In particular, CheckMate can help compute reachable sets for nonlinear systems of low dimension. The package d/dt can handle linear differential inclusions and lends itself to the analysis of our switched linear system.

Control

Nonlinear control theory offers analytical tools for establishing properties such as small-time local controllability, global controllability, and stability.[30] These tools can be valuable in establishing the relationship between inputs or signaling mechanisms and regulated variables in biology.

Traditional control theory mostly enables the design of controllers in a single mode of operation in which the task and the system model are fixed.[30] When operating in unstructured or dynamic environments with many different sources of uncertainty, designing controllers that guarantee performance, even in a local sense, is difficult if not impossible.[31] In contrast, we also know that designing reactive controllers or behaviors that react to simple stimuli or commands from the environment is relatively easy. We can see successful applications of this idea in subsumption architectures and their derivatives in robotics.[32,33] Similarly, biology is full of examples of such simple controllers—in some cases they are well understood.[6,34] Although control and estimation theory let us model each behavior as a dynamical system and provide us with design and analysis tools, we currently do not have tools for more complex systems that involve

- switches in behavior or sequential composition of modes;
- hierarchical composition of modes; or
- parallel composition of agents (or concurrent operation of behaviors).

There is potentially a lot to gain by specializing existing tools and facilitating analysis of biological systems. Applications of geometric control theory could yield insight into the regulation of proteins along the hybrid system's flow. Tools from optimal control theory could synthesize open loop controls that might point experimental biologists to new paradigms for experimentation.

Issues of robustness and fault tolerance are of special importance. Biological systems have built-in mechanisms that provide robustness and fault tolerance. It is particularly important to understand how random fluctuations affect regulation in hybrid systems where precision and reliability are required.[35]

Case study: Quorum sensing in *V. fischeri*

A good illustration of a multicellular network is the cell-density-dependent gene expression seen in prokaryotes. In this process, a single cell can sense when a quorum of bacteria—a minimum population unit—is achieved. Under these conditions, the quorum efficiently performs certain behavior such as bioluminescence, the best-known model for understanding the mechanism of cell-density-dependent gene expression.

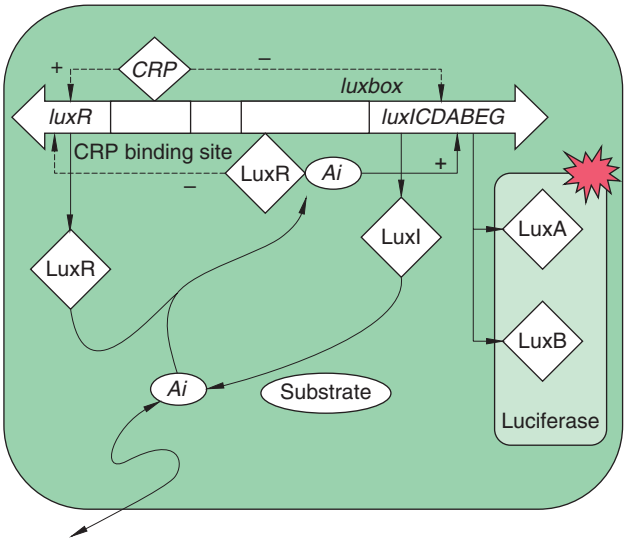

Figure 5. A portion of DNA showing the *luxR* (transcribed from the leftward operon O_L) and *luxICDABEG* (transcribed from the rightward operon O_R) genes and the binding sites for complex *Co* and CRP.

Vibrio fischeri is a marine bacterium found both as a 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 nonluminescent. As a symbiont, the bacteria live at high densities and are usually luminescent. In a liquid culture, the bacterium's level of luminescence is low until the culture reaches mid to late exponential phase. Later, in stationary phase, the bioluminescence level decreases. The transcriptional activation of the *lux* genes in the bacterium controls this luminescence.

The lux regulon[36] is organized in two transcriptional units: O_L and O_R (see Figure 5). O_L contains the *luxR* gene that encodes the protein LuxR, a transcriptional regulator of the system. (We use italics to indicate the genes and roman text to denote the protein the gene expresses.) O_R contains seven genes *luxICDABEG*. The transcription of the *luxI* gene produces the protein LuxI, which is required for endogenous production of the autoinducer *Ai*, a small membrane-permeable signal molecule (acyl-homoserine lactone). The genes *luxA* and *luxB* code for the luciferase subunits, and *luxC*, *luxD*, and *luxE* code for proteins of the fatty acid reductase, which generates aldehyde substrate for luciferase. The *luxG* gene is thought to encode a flavin reductase. 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* in Figure 5. 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—the transcription of the *luxICDABEG* by the binding of *Co* to the lux box.[37] However, growth in the levels of *Co* and cAMP/CRP inhibit *luxR* and *luxICDABEG* transcription, respectively. These feedback loops are shown through dotted arrows with + and – signs in Figure 5.

Model

Our model of bioluminescence in *V. fischeri* consists of nine state variables, each representing the evolution of an S-agent. x_0 represents the cell population, whereas the x_1 through x_8 represent nanomolar concentrations of mRNAs and proteins. This model is described by[15]

$$\begin{aligned}
\dot{x}_0 &= k_G x_0 \\
\dot{x}_1 &= T_c\left(\Psi(x_8, \kappa_{Co-icdabeg}, \nu_{Co-icdabeg})\Phi(c_{CRP}, \kappa_{CRPr}, \nu_{CRPr}) + b\right) - \frac{x_1}{H_{RNA}} - k_G x_1 \\
\dot{x}_2 &= T_c\left(\Phi(x_8, \kappa_{Co-icdabeg}, \nu_{Co-icdabeg})\Psi(c_{CRP}, \kappa_{CRPr}, \nu_{CRPr}) + b\right) - \frac{x_2}{H_{RNA}} - k_G x_2 \\
\dot{x}_3 &= T_l x_1 - x_3 / H_{sp} - r_{AIR} x_7 x_3 + r_{Co} x_8 - k_G x_3 \\
\dot{x}_4 &= T_l x_2 - x_4 / H_{sp} - k_G x_4 \\
\dot{x}_5 &= T_l x_2 - x_5 / H_{sp} - k_G x_5 \\
\dot{x}_6 &= T_l x_2 - x_6 / H_{up} - k_G x_6 \\
\dot{x}_7 &= x_0\left(r_{AII} x_4 - r_{AIR} x_7 x_3 + r_{Co} x_8\right) - x_7 / H_{AI} \\
\dot{x}_8 &= r_{AIR} x_7 x_3 - x_8 / H_{sp} - r_{Co} x_8 - k_G x_8
\end{aligned} \tag{5}$$

where

$$k_G = k_g(1 - x_0/x_{0\max}) \tag{6}$$

and x_i are nonnegative real numbers:

$$\begin{aligned}
x_0 &= \text{scaled population (population} \times v_b/V), \\
x_1 &= \text{mRNA transcribed from } O_L, \\
x_2 &= \text{mRNA transcribed from } O_R, \\
x_3 &= \text{protein LuxR}, \\
x_4 &= \text{protein LuxI}, \\
x_5 &= \text{protein LuxA/B}, \\
x_6 &= \text{protein LuxC/D/E}, \\
x_7 &= \text{autoinducer } Ai, \\
x_8 &= \text{complex } Co.
\end{aligned} \tag{7}$$

Table 1 gives the parameters in Equation 5.[38] c_{CRP} denotes the CRP concentration in the bacterium.

Our hybrid model is obtained by approximating the functions Φ and Ψ by piecewise constant functions. Assume m levels of activation of the *luxICDABEG* gene at $c_0, c_1, \ldots, c_{m-1}$ with $c_0 = 0$

and $c_{m-1} = 1$. The *luxICDABEG* gene's level of activation is c_i when

$$x_{8sw}^{i} < x_8 < x_{8sw}^{i+1}, \text{ where}$$

$$x_{8sw}^{0} = 0 \text{ and } x_{8sw}^{m} = \infty.$$

Similarly, for the *luxR* gene, we assume n steps at $d_0, d_1, \ldots, d_{n-1}$, $d_0 = 0$, $d_{n-1} = 1$, and the activation is at level d_j when

$$c_{CRPsw}^{j} < c_{CRP} < c_{CRPsw}^{j+1}.$$

The continuous, nine-dimensional system is now replaced by a hybrid system with mn distinct modes. (More specifically, this is an example of a *switched* system, a special case of a hybrid system.) The modes are indexed by $i = 1, \ldots, m - 1$ and $j = 1, \ldots, n - 1$, and a seven-dimensional system describes dynamics in each mode.

Figure 6a shows log-linear plots of numerical simulations of the continuous system, for a constant c_{CRP} of 10nM. The population (bottom) grows from 10^5/ml to 10^9/ml over approximately 15 hours then remains constant for an additional 20 hours. For all the species, there is a short initial adjustment period when protein, complex, and *Ai* concentrations reach a nearly constant state at low population density, which lasts for roughly eight hours. LuxR is abundant, ready to sense increases in *Ai*. On the other hand, *Co*, LuxI, *Ai*, and total luminescence are low. As the population grows, *Ai* production begins to rise above the background, which triggers the formation of LuxR: *Ai* complex *Co*. Increasing *Co* activates the positive feedback loop of O_R, LuxI, and *Ai*. The concentrations of these components increase rapidly as does the total luminescence, and this is what we expect to get through experiments.[37]

Figure 6b shows the simulation results of the switched model obtained from Equation 5 by replacing the sigmoids with piecewise constant functions with three steps ($m = n = 3$). The plots closely resemble the continuous simulations. The final equilibrium values are also in good agreement between the two models, thus arguing for the hybrid approximation's validity.

Stability analysis

For the full continuous system in Equation 9, deriving the equilibria and studying their stability for arbitrary parameters are difficult due to the essentially nonlinear behavior of the activation functions. For the simplified hybrid system, the nonlinearities involve products of the continuous state variables. We considered a fully grown population (x_0 is constant). Carefully analyzing the timescales governing the system shows that the mRNA dynamics are fast compared to other dynamics,[38] and we can assume the corresponding mRNA concentrations are constant in each mode (i,j):

$$x_1^{ij} = H_{RNA}T_c\left[(1-c_i)d_j + b\right], x_2^{ij} = H_{RNA}T_c\left[c_i(1-d_j) + b\right]. \tag{8}$$

The system's dynamic properties in mode (i,j) are completely determined by the time evolution of the reduced system:

$$\dot{x} = Ax + g(x) + b_{ij}, x = \left[x_3 x_4 x_7 x_8\right]^T, \tag{9}$$

where A is a constant 4×4 matrix, b_{ij} is a 4×1 vector dependent on the mode (i,j), and g is a 4×1 vector, which is quadratic in x_3 and x_7. The invariant sets as described earlier are given by

$$x_{8sw}^{i} < x_8 < x_{8sw}^{i+1} \wedge c_{CRPsw}^{j} < c_{CRP} < c_{CRPsw}^{j+1}. \tag{10}$$

As shown elsewhere,[37] each mode (i,j) has a unique equilibrium. Studying each equilibrium point's stability reduces to examining the eigenvalues of the Jacobian matrix for each mode. We

Table 1. Model parameters: notation and description.

Parameter	Description
T_c	Maximum transcription rate
T_l	Maximum translation rate
H_{RNA}	RNA half-life
H_{sp}	Stable protein half-life
H_{up}	Unstable protein half-life
H_{AI}	*Ai* half-life
r_{AII}	Rate constant at which LuxI makes *Ai*
r_{AIR}	Rate constant *Ai* binding to LuxR
r_{Co}	Rate constant of *Co* dissociation
v_{CRPr}	Cooperativity coefficient for CRP
κ_{CRPr}	Half maximum conc. for CRP
$v_{Co\text{-}icdabeg}$	Cooperativity coefficient for *Co*
$\kappa_{Co\text{-}icdabeg}$	Half maximum conc. for *Co*
b	Basal transcription rate
v_b	Volume of a bacterium
V	Volume of solution
k_g	Growth rate
x_{0max}	Maximum population

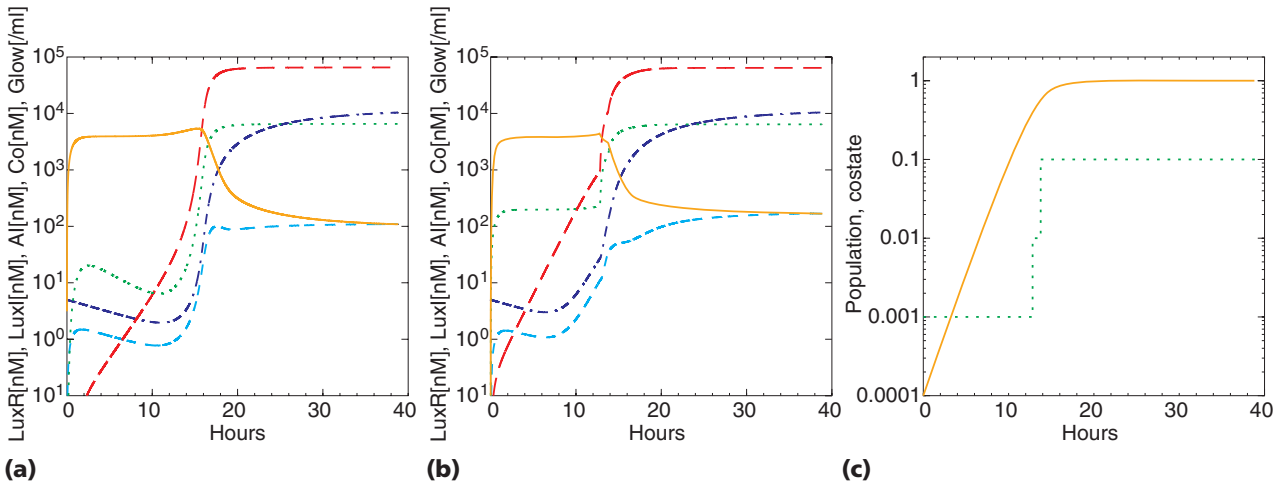

Figure 6. Time evolution of concentrations (in nM) during population growth: proteins LuxR (solid), LuxI (dotted), *Ai* (dot-dashed), *Co* (dashed), and total luminescence (long dashes): (a) continuous model, (b) switched model. (c) Population growth curve. The switch history for the hybrid model is dotted.

implemented this set of calculations in a Mathematica notebook and showed that all equilibrium points are stable.

This stability result is a local one. In other words, this computation simply shows that under small perturbations from an equilibrium, the system will return to the equilibrium point. However, we can do better. We used the vanishing perturbation technique in our previous work[39] to construct an ellipsoidal region of attraction around the equilibrium point's origin.

Besides stability, another interesting question for both biologists and control engineers would be what makes the system switch among modes. Or, from a synthesis viewpoint, what parameters or variables can we artificially modify to promote the production of luciferase, the proteins that eventually result in luminescence. Analyzing the system in Equation 9 gives us some insight into this. For example, a careful inspection of the directional derivatives of the output x_5 along the flow of the system shows that this variable is directly affected by the transcription of *luxICDABEG* (x_2), indirectly by the production of CRP, and even more indirectly by the production of the *Ai*(x_8). From a synthetic biology standpoint, we now have three avenues to increase the production of luciferase.

Reachability analysis

Another interesting question from a biologist's standpoint relates to reachability. For example, can a cell population from a specified initial condition reach an equilibrium that corresponds to luminescence? The equilibrium's stability does not answer this question because it only indicates that if the system reaches a region around the stable point, then it will move to the equilibrium point.

This is a so-called reachability problem. Consider the mutant of *V. fischeri* described elsewhere[37] that disrupts the operon structure and thus the regulatory system described earlier. In the mutant, the *luxR* gene is deleted from its normal location, and a copy is placed in a plasmid (a separate piece of DNA) and incorporated back into *V. fischeri*. The plasmid copy of *luxR* is under a different promoter's control, which yields a constant high rate of transcription unaffected by any of the molecules described earlier. We assume a constant concentration of protein LuxR (justified in this model), and the CRP concentration is low (regulation kept at $d_0 = 0$). The model becomes linear with the nontrivial dynamics in mode (i, j) being described by

$$\dot{\bar{x}} = \bar{A}\bar{x} + \bar{b}_{ij}, \tag{11}$$

where $\bar{x} = [x_4 x_7 x_8]^T$, $\bar{A}$ is a constant 3×3 matrix, and b_{ij} is a 3×1 vector that depends on the mode. Although the associated Jacobian's eigenvalues show that the equilibrium points are asymptotically stable,[37] the stability result is not a global one. The eigenvalue analysis tells us that if the system starts from a point within a region of attraction around the equilibrium of the mode (i,j), it converges to the equilibrium point. However,

even if the dynamics are linear, it does not tell us anything about initial conditions that could lie elsewhere (possibly in a different mode). This is the main motivation for reachability analysis.

We are interested in computing the set of states that converge to the equilibrium point x_e^{lum} in the luminescent mode. Figure 7 shows the results of a reachability analysis performed with d/dt, a software package that uses polyhedra to over- or underapproximate reachable sets of hybrid automata. For our problem, we need an underapproximation because we must make sure that all the states in the computed set go to the equilibrium x_e^{lum}.

We envision the link between hybrid systems of technology and biology to strengthen. The scalable nature of computational tools such as Charon will enable the unified and improved modeling of biological cellular networks, leading to better understanding and providing us with the opportunity to determine how local biological changes can affect global behavior. Conversely, a good understanding of the robustness of noisy biological networks will lead to new approaches to designing networked embedded systems. Efforts such as ours can only succeed if they are closely tied to research in experimental biology. Our goal is to provide the analytical and computational tools that biologists can use in a fashion that complements experimental research.

References

1. J.C. Venter et al., "The Sequence of the Human Genome," *Science*, vol. 291, no. 5507, 2001, pp. 1304–1351.
2. L.H. Hartwell et al., "From Molecular to Modular Cell Biology," *Nature*, vol. 402, no. 6761, Dec. 1999, pp. C47–52.
3. J.P. Klein, T.H. Leete, and H. Rubin, "A Biomolecular Implementation of Logically Reversible Computation with Minimal Energy Dissipation," *BioSystems*, vol. 52, 1999, pp. 15–23.
4. M.B. Elowitz and S. Leibler, "A Synthetic Oscillatory Network of Transcriptional Regulators," *Nature*, vol. 43, no. 20, Jan. 2000, pp. 335–338.
5. P. Mendes and D.B. Kell, "Nonlinear Optimization of Biochemical Pathways: Applications to Metabolic Engineering and Parameter Estimation," *Bioinformatics*, vol. 14, no. 10, 1998, pp. 869–883.
6. M. Ptashne, *A Genetic Switch: Phage and Higher Organisms*, Cell Press, Cambridge, Mass., 1992.
7. B. Lewin, *Genes VII*, Oxford Univ. Press, Cambridge, UK, 2000.
8. R. Ghosh and C.J. Tomlin, "Lateral Inhibition through Delta-Notch Signaling: A Piecewise Affine Hybrid Model," *Hybrid Systems: Computation and Control*, Lecture Notes in Computer Science, vol. 2034, Springer-Verlag, Berlin, 2001.
9. R. Alur, T.A. Henzinger, and E.D. Sontag, eds., *Hybrid Systems III: Verification and Control*, Lecture Notes in Computer Science 1066, Springer-Verlag, Berlin, 1996.
10. T. Henzinger and S. Sastry, eds., *Hybrid Systems: Computation and Control*, Lecture Notes in Computer Science 1386, Springer-Verlag, Berlin, 1998.

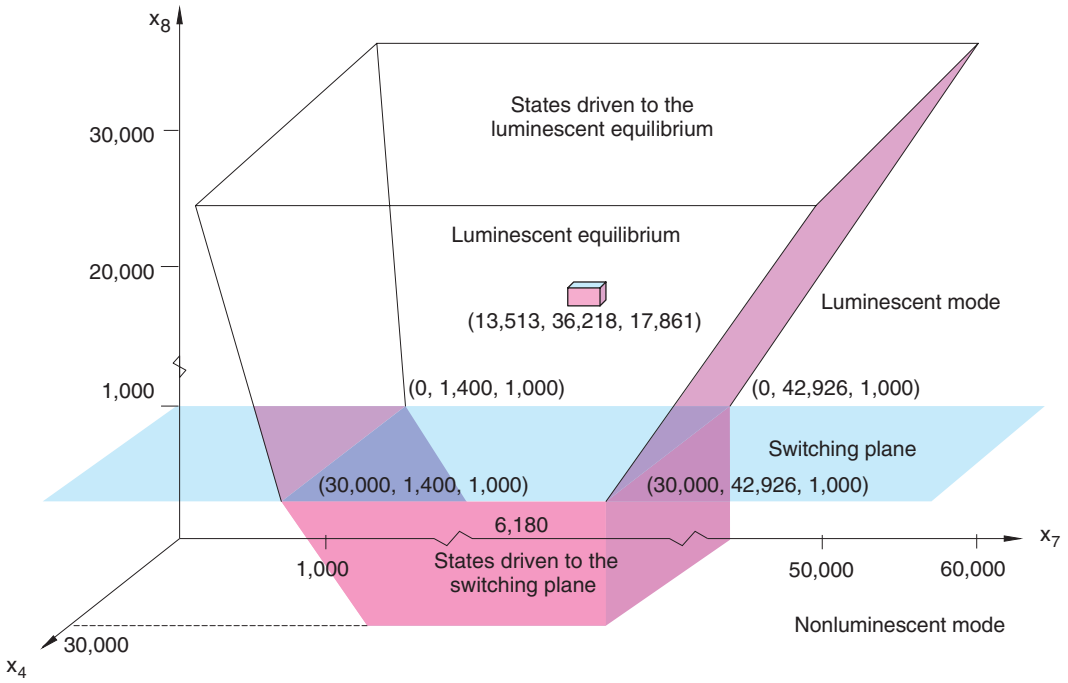

Figure 7. The set of states that are driven to the luminescent equilibrium by switching on the *lux* gene.

11. N. Lynch and B.H. Krogh, eds., *Hybrid Systems: Computation and Control*, Lecture Notes in Computer Science 1790, Springer-Verlag, Berlin, 2000.
12. R.W. Brockett, "Hybrid Models for Motion Control Systems," *Essays in Control: Perspectives in the Theory and its Applications*, Birkhauser, Boston, 1993, pp. 29–53.
13. V. Kumar, M. Zefran, and J. Ostrowski, "Intelligent Motion Planning and Control," *Handbook of Industrial Robotics*, John Wiley & Sons, New York, 1999.
14. C. Tomlin, G. J. Pappas, and S. Sastry, "Conflict Resolution for Air Traffic Management: A Study in Multi-Agent Hybrid Systems," *IEEE Trans. Automatic Control*, vol. 43, no. 4, 1998, pp. 509–521.
15. G. von Dassow et al., "The Segment Polarity Network Is a Robust Development Module," *Nature*, vol. 406, no. 6792, 13 July 2000, pp. 188–192.
16. D.T. Gillespie, "Exact Stochastic Simulation of Coupled Chemical Reactions," *J. Physical Chemistry*, vol. 81, 1977, pp. 2340–2361.
17. D.T. Gillespie, "Approximating the Master Equation by Fokker-Planck-Type Equations for Single-Variable Chemical Systems," *J. Chemical Physics*, vol. 72, 1980, pp. 5363–5370.
18. R. Heinrich and S. Schuster, *The Regulation of Cellular Systems*, Int'l Thomson Publishing, New York, 1996.
19. R. Alur et al., "Modular Specifications of Hybrid Systems in CHARON," *Hybrid Systems: Computation and Control,* Lecture Notes in Computer Science, vol. 1790, Springer-Verlag, Berlin, 2000, pp. 6–19.
20. C.A.R. Hoare, *Communicating Sequential Processes*, Prentice-Hall, Upper Saddle River, N.J., 1985.
21. D. Harel, "Statecharts: A Visual Formalism for Complex Systems," *Science of Computer Programming*, vol. 8, no. 3, June 1987, pp. 231–274.
22. G. Booch, I. Jacobson, and J. Rumbaugh, *Unified Modeling Language User Guide*, Addison Wesley, Reading, Mass., 1997.
23. N. Lynch et al., "Hybrid I/O Automata," *Hybrid Systems III: Verification and Control*, Lecture Notes in Computer Science 1066, Springer-Verlag, Berlin, 1996, pp. 496–510.
24. A. Deshpande, A. Gollu, and L. Semenzato, *SHIFT Programming Language and Runtime Systems for Dynamic Networks of Hybrid Automata*, Tech. Report, Univ. California, Berkeley, 1997.
25. M. Tomita et al., "E-Cell: Software Environment for Whole-Cell Simulation," *Bioinformatics*, vol. 15, no. 1, Jan. 1999, pp. 72–84.
26. C.W. Gear and D.R. Wells, "Multirate Linear Multistep Methods," *BIT*, vol. 24, 1984, pp. 484–502.
27. J. Esposito, V. Kumar, and G. Pappas, "Accurate Event Detection for Simulating Hybrid Systems," to be published in *Hybrid Systems: Computation and Control*, Lecture Notes in Computer Science, Springer-Verlag, Berlin, 2001.
28. R. Alur, T.A. Henzinger, and P.-H. Ho, "Automatic Symbolic Verification of Embedded Systems," *IEEE Trans. Software Eng.*, vol. 22, no. 3, 1996, pp. 181–201.
29. A. Chutinam and B. Krogh, "Verification of Polyhedral-Invariant Hybrid Automata Using Polygonal Flow Pipe Approximations," *Hybrid Systems: Computation and Control*, Lecture Notes in Computer Science 1569, Springer-Verlag, Berlin, 1999.
30. S. Sastry, *Nonlinear Systems: Analysis, Stability and Control*, Springer Verlag, New York, 1999.
31. D. Liberzon and A.S. Morse, "Basic Problems in Stability and Design of Switched Systems," *IEEE Control Systems*, vol. 19, no. 5, Oct. 1999, pp. 59–70.
32. R. Brooks, "A Robust Layered Control System for a Mobile Robot," *IEEE J. Robotics and Automation*, vol. 2, no. 1, 1986, pp. 14–23.
33. R. Arkin and T. Balch, *Artificial Intelligence and Mobile Robots*, MIT Press, Cambridge, Mass., 1998.
34. A.J. Arkin, J. Ross, and H.H. McAdams, "Stochastic Kinetic Analysis of Developmental Pathway Bifurcation in Phage Lambda-Infected *escherichia coli* Cells," *Genetics*, vol. 149, no. 4, 1998, pp. 1633–1648.
35. O.G. Berg, J. Paulsson, and M. Ehrenberg, "Fluctuations and Quality of Control in Biological Cells: Zero-Order Ultrasensitivity Reinvestigated," *Biophysical J.*, vol. 79, no. 9, Sept. 2000, pp. 1228–1236.
36. D.M. Sitnikov, J.B. Schineller, and T.O. Baldwin, "Transcriptional Regulation of Bioluminescence Genes from Vibrio Fischeri," *Molecular Microbiology*, vol. 17, no. 5, 1995, pp. 801–812.
37. P.V. Dunlap, "Quorum Regulation of Luminescence in *Vibrio fischeri*," *Molecular Marine Microbiology*, Horizon Press, Norfolk, UK, 2000, pp. 3–21.
38. C. Belta et al., "Stability and Reachability Analysis of a Hybrid Model of Luminescence in the Marine Bacterium *V. fischeri*," Grasp Lab., Univ. Pennsylvania, Philadelphia, 2001.
39. H. Khalil, *Nonlinear Systems*, Prentice Hall, 2nd ed., Upper Saddle River, N.J., 1996.

Rajeev Alur is a professor of Computer and Information Science at the University of Pennsylvania. His areas of research include embedded software, formal specification and verification, hybrid systems, concurrency theory, and programming languages. He received his BS in computer science from Indian Institute of Technology at Kanpur and his PhD in computer science from Stanford University. Contact him at the Dept. of Computer & Information Science, 200 South 33rd St., Univ. of Pennsylvania, Philadelphia, PA 19104-6389; www.cis.upenn.edu/~alur.

Calin Belta received his BS and MS in control and computer science from the Technical University of Iasi, Romania and an MS in electrical engineering from Louisiana State University. He is currently pursuing his PhD in mechanical engineering at the University of Pennsylvania. His research interests include motion planning for rigid bodies and formations of robots, hybrid systems, and modeling of biomolecular networks. Contact him at GRASP Laboratory, Univ. of Pennsylvania, 3401 Walnut St., Room 301C, Philadelphia, PA 19104-6228; calin@grasp.cis.upenn.edu; www.seas.upenn.edu/~calin.

Vijay Kumar is a professor and the deputy dean for research in the School of Engineering and Applied Science at the University of Pennsylvania. His research interests include robotics, dynamics, control, design, and biomechanics. He received his MSc and PhD in mechanical engineering from Ohio State University. He is a member of the American Society of Mechanical Engineers, Institute of Electrical and Electronic Engineers, Robotics International, and the Society of Manufactur-

ing Engineers. Contact him at the GRASP Laboratory, Univ. of Pennsylvania, 3401 Walnut St., Philadelphia, PA 19104-6228; kumar@cis.upenn.edu; www.cis.upenn.edu/~kumar.

Max Mintz is a professor of computer and information science and chair of the undergraduate Computer Science and Engineering Program at the University of Pennsylvania. His research interests include modeling, identification, and control of dynamic systems; statistical decision theory with applications to machine perception and robotics; and the development of algorithms that achieve guaranteed performance in decentralized estimation and control of uncertain dynamic systems. He received his BEE, MS, and PhD from Cornell University. Contact him at the GRASP Laboratory, Room 301, 3401 Walnut St., Univ. of Pennsylvania, Philadelphia, PA 19104-6228; mintz@central.cis.upenn.edu.

George J. Pappas is an assistant professor of electrical engineering at the University of Pennsylvania. His research interests include embedded hybrid systems, hierarchical control systems, nonlinear control systems, geometric control theory, flight and air traffic management systems, robotics, and unmanned aerial vehicles. He received a BS and MS in computer and systems engineering, both from Rensselaer Polytechnic Institute, and a PhD in electrical engineering and computer sciences from the University of California at Berkeley. Contact him at the Dept. of Electrical Engineering, 200 South 33rd St., Univ. of Pennsylvania, Philadelphia, PA 19104; pappasg@ee.upenn.edu.

Harvey Rubin is a professor in the School of Medicine at the University of Pennsylvania. His research interests include elucidating the genetic and metabolic regulatory networks that allow tuberculosis to persist in the human host for years, determination of the molecular basis of serine protease inhibition, and the mathematical modeling of complex biomolecular systems. He received a PhD in molecular biology from the University of Pennsylvania and a MD from Columbia University. Contact him at the School of Medicine, Univ. of Pennsylvania, 522 Johnson Pavilion, Philadelphia PA 19104; http://mail.med.upenn.edu/~rubinh.

Jonathan Schug received his BA in mathematics and MSE in computer and information science from the University of Pennsylvania. He is currently pursuing his PhD in computer and information science. His research interests include the regulation of gene expression and the analysis of promoter and other DNA sequences. Contact him at the Center for Bioinformatics, Univ. of Pennsylvania, 1315 Blockley Hall, Philadelphia, PA 19014-6021; jschug@pcbi.upenn.edu.

How to Reach *CiSE*

Writers

For detailed information on submitting articles, write to cise@computer.org or visit http://computer.org/cise/edguide.htm.

Letters to the Editors

Send letters to

Jenny Ferrero, Contact Editor
jferrero@computer.org

Please provide an email address or daytime phone number with your letter.

On the Web

Access http://computer.org/cise or http://ojps.aip.org/cise for information about *CiSE*.

Subscription Change of Address (IEEE/CS)

Send change-of-address requests for magazine subscriptions to address.change@ieee.org. Be sure to specify *CiSE*.

Subscription Change of Address (AIP)

Send general subscription and refund inquiries to subs@aip.org.

Subscribe

Visit http://ojps.aip.org/cise/subscrib.html or http://computer.org/subscribe.

Missing or Damaged Copies

If you are missing an issue or you received a damaged copy, contact membership@computer.org.

Reprints of Articles

For price information or to order reprints, send email to cise@computer.org or fax +1 714 821 4010.

Reprint Permission

To obtain permission to reprint an article, contact William Hagen, IEEE Copyrights and Trademarks Manager, at whagen@ieee.org.