ACS Synthetic Biology
Research Article
pubs.acs.org/synthbio

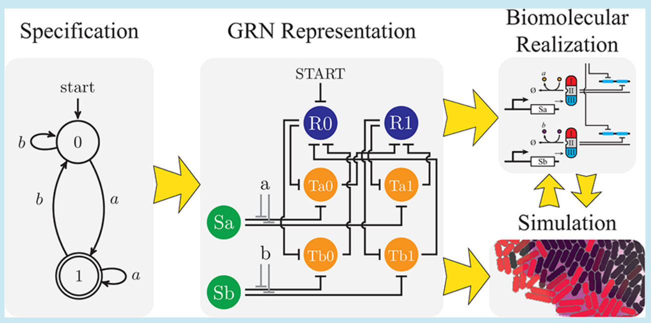
Framework for Engineering Finite State Machines in Gene Regulatory Networks

Kevin Oishi* and Eric Klavins

Department of Electrical Engineering, University of Washington, Seattle 98195, United States

S *Supporting Information*

ABSTRACT: Finite state machines are fundamental computing devices at the core of many models of computation. In biology, finite state machines are commonly used as models of development in multicellular organisms. However, it remains unclear to what extent cells can remember state, how they can transition from one state to another reliably, and whether the existing parts available to the synthetic biologist are sufficient to implement specified finite state machines in living cells. Furthermore, how complex multicellular behaviors can be realized by multiple cells coordinating their states with signaling, growth, and division is not well understood. Here, we describe a method by which *any* finite state machine can be built using nothing more than a suitably engineered network of readily available repressing transcription factors. In particular, we show the mathematical equivalence of finite state machines with a Boolean model of gene regulatory networks. We describe how such networks can be realized with a small class of promoters and transcription factors. To demonstrate the effectiveness of our approach, we show that the behavior of the coarse grained ideal Boolean network model approximates a fine grained delay differential equation model of gene expression. Finally, we explore a framework for the design of more complex systems via an example, synthetic bacterial microcolony edge detection, that illustrates how finite state machines could be used together with cell signaling to construct novel multicellular behaviors.



KEYWORDS: *finite state machines, gene regulatory networks, multicellular behavior, framework, specification*

An engineering framework for *finite state machines* (FSMs) in living cells is crucial scientifically and practically. Naturally occurring finite state machines control how cells switch states from stem cells to tissue specific cells according to chemical, mechanical, and logical cues from their local environments.[1−4] However, many questions remain unanswered: How is state encoded by patterns of gene expression? How are states stabilized to avoid spontaneous switching? How do state-specific signals reliably cause transitions between states? Most importantly, can we build novel finite state machines in synthetic cells that control the process of differentiation and development? Here, we present a framework and design methodology for engineering state control in cells. The design method is rooted in the rich theory of finite state machines and sequential logic from computer engineering.[5,6] We apply the method to a variety of examples modeled at different levels of detail and abstraction: as a Boolean network representation of a *gene regulatory network* (GRN), as *delay differential equations* (DDEs) modeling a biomolecular implementation of the GRN, and finally as a 2D multicellular simulation illustrating a complex microcolony edge detection behavior implemented from a high-level finite state machine specification.

Finite state machines are a fundamental model of computation and can be used to represent and reason about many useful machines including counters, adders, and any other kind of sequential logic. An FSM can be in any one of a finite number of states at a given time. An FSM takes as input a string of symbols that belong to a finite set. Upon the arrival of an input symbol, the FSM updates its state according to a *transition function* that depends on the current state of the machine and the current input symbol. As a consequence, an FSM has a memory and may respond differently to the same input depending on its current state. Although the restriction to finiteness makes FSMs theoretically less capable than other computing machines (for example, pushdown automata or Turing machines), FSMs nevertheless form the basis of most modern models of computation.[7] Additionally, FSMs have been richly explored and applied to engineering problems in electronics, computer architecture, and computer science. In computer science, the FSM formalization is fundamental in categorizing and understanding the limitations of theoretical and physical machines that compute.[6,8,9] FSMs were fundamental in shaping the design of modern computers where state is typically stored in latches or flip-flops.[10,11]

One of the first forms of finite automata explored in the literature was a model of nerve cell networks investigated by theoretical biologists and mathematicians[12,13] that was later

Received: November 1, 2013
Published: February 21, 2014

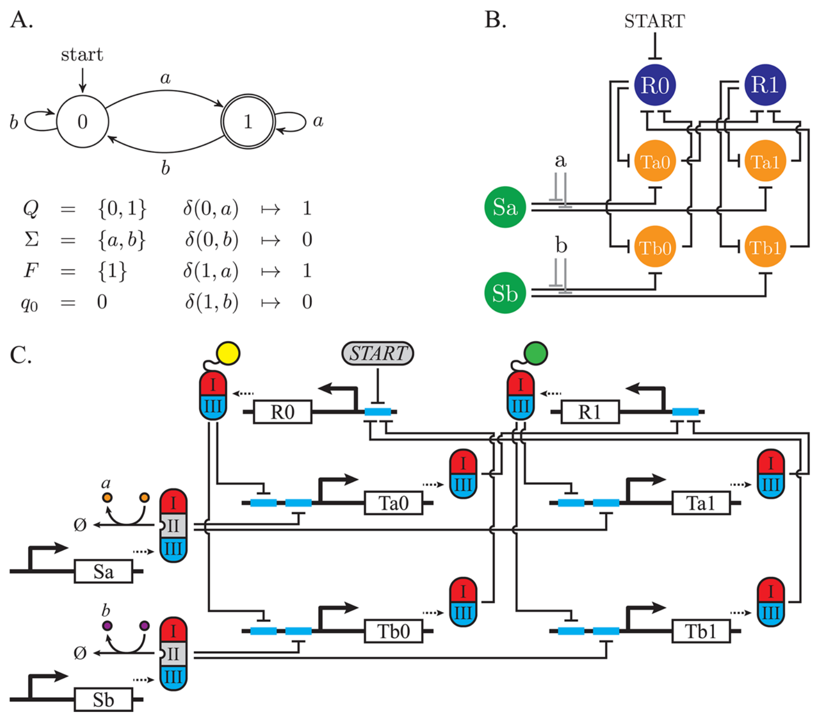


Figure 1. Simple two-state machine described as (A) a directed graph representation of a finite state machine, (B) a gene regulatory network made of repressing transcription factors and inducers, and (C) a biomolecular realization of the same GRN using the parts described in Figure 2. In the GRN representation orange circles denote transition species, purple circles denote state species, and green circles denote sensor species. In the GRN and biomolecular realization, the gene network is in state i when species Ri is at a low level expression, and following transition $\delta(q, \sigma)$ when transition species Tσq is at a high level of expression.

shown to be equivalent to FSMs.[6] Here, we explore to what extent gene regulatory networks are equivalent to FSMs as well. In fact, several synthetic biological mechanisms in the literature have been shown to implement simple finite state machines *in vivo* with just a few states. Broadly speaking, these mechanisms fall into the categories of cells that compute, cells that remember, and cells that communicate. Simple computation using Boolean logic has been demonstrated in multiple organisms with some degree of cascaded modularity.[14−17] Genetic toggle switches have been heavily explored as a mechanism for remembering stimulus events.[18−21] Similar architectures have been used to implement more complex machines, such as a counter[22] and a timer.[23] Recently, elements of simple computation and memory have been combined to implement all two-input Boolean logic functions in *Escherichia coli* and store the result in DNA over the subsequent 90 cell divisions.[24] Synthetic multicellular systems have employed cell−cell communication mechanisms with simple computing or memory circuits to implement robust and complex behaviors such as synchronized oscillation,[25] population density control,[26] photosensitive edge detection,[27] stripe formation,[28] and logic networks.[29,30] Although these examples hint at the potential power of synthetic biology to build arbitrary FSMs in cells, a general framework[31] for the design and of synthesis of any given FSM—a sort of 'FSM compiler'—has yet to emerge.

A framework for engineering finite state machines from biomolecular parts would enable the design, construction, and characterization of complex circuits from simple functional parts. In light of the success of simple synthetic circuits, and the availability of libraries of new parts, such as CRISPR, TALE, and Zinc Finger transcription factors,[32−36] we pose the questions: Are these parts sufficient to realize the multistate behaviors observed in nature? How powerful of a computer can be made using only transcription factors in a GRN? To answer these questions, we demonstrate via explicit mathematical construction that modeled as a Boolean network, GRNs are exactly as powerful as FSMs. Furthermore, a continuous-time DDE model of gene dynamics can closely follow the behavior of the Boolean network model over a range of physically relevant parameters.

We begin with a brief review of FSMs, providing background and context for subsequent discussion of biomolecular parts and models. Next, we introduce the general classes of parts we use in our biomolecular constructions and present the Boolean network and DDE models that we use to analyze and simulate GRNs made from our biomolecular parts. Notably, the biomolecular gene regulatory parts we consider consist of only repressing transcription factors. Mathematically, repressing transcription factors by themselves represent a minimal set of part types (i.e., an activating relation can be built from two repressing relations in series). Practically repressing transcription factors may be easier to engineer than activators. We then present a method for implementing any finite state machine with a network of suitably wired repressing transcription factors. The correctness and effectiveness of the construction is then examined assuming the Boolean network and DDE models for two example systems: a simple two state machine and a four state modulo-two pulse counter. Finally, we show how our framework can be used to design a logical control system for cell fate interfacing various biomolecular sensors and effectors. We illustrate the approach through the design and simulation of a bacterial microcolony edge detection system where cells grow, divide, and dynamically differentiate into red fluorescing "edge" cells and nonfluorescing "center" cells.

RESULTS AND DISCUSSION

Finite State Machines. Finite state machines are a well understood and intuitive model of state control and computation. A finite state machine is specified by the tuple,

$$M = (Q, \Sigma, \delta, q_0, F) \tag{1}$$

where Q and Σ are finite sets, δ: $Q \times \Sigma \to Q$, $q_0 \in Q$, and $F \subseteq Q$. Q is a set of *states*, Σ is a set of *input symbols*, q_0 is the unique *start state*, δ is a *state transition function*, and F is a (possibly empty) set of *accepting states*. Finite state machines are typically represented as a directed graph. For example, Figure 1A depicts a two-state FSM. The set of states $Q = \{0,1\}$ are represented by labeled vertexes. The start state $q_0 = 0$ is denoted by the "start" arrow, and the set of accepting states $F = \{1\}$ are illustrated as a double-circled vertex. Edge labels denote the full set of inputs $\Sigma = \{a, b\}$. The state transition map δ is represented by edges labeled with a list of input symbols that move the FSM between states.

The semantics of a finite state machine is then a set of rules defining its operation specified by M. The FSM takes as input a finite length sequence of symbols $w \in \Sigma^*$ (where Σ^* is the Kleene operator applied to Σ) and determines weather or not the sequence is in a set of accepted sequences through a series of state transitions. The FSM begins in the initial state q_0. Given a sequence of inputs $w = \sigma_1\sigma_2...\sigma_n$ with $\sigma_i \in \Sigma$ for $i = 1,2,...,n$, machine first transitions to the state $q_1 = \delta(q_0, \sigma_1)$. The k^{th} state the FSM transitions to is then $q_k = \delta(q_{k-1}, \sigma_k)$. The w is an *accepted sequence* if $q_n \in F$. If $q_n \notin F$ or if for some k, $\delta(q_{k-1}, \sigma_k)$ is not defined, the FSM does not accept w. In other words, w is accepted if and only if there is a path through the graph induced by M that begins at q_0 and ends at $q_n \in F$ where consecutive edges in the path have labels that contain $\sigma_1,\sigma_2,...,\sigma_n$.

In the sequel, we present a general method for constructing a GRN with a small number of component types to realize an arbitrary FSM specification. We illustrate the method through example, using the two-state machine in Figure 1, and show how the Boolean network model of the GRN implements the two-state FSM. Additionally, we show that a biomolecular realization of the GRN modeled by DDEs closely approximates the Boolean network dynamics for a wide range of physically realistic parameters. Finally, we show how finite state machines can be used to engineer complex multicellular behaviors with a bacterial microcolony edge detection example simulated in Gro.[37]

Modeling Gene Regulatory Networks. Gene regulatory networks are specified by the tuple,

$$G = (G_V, G_U) \tag{2}$$

$$G_V = (V, E_r, E_a) \tag{3}$$

$$G_U = (U, I_r, I_a) \tag{4}$$

where G_V is a internal gene network graph, and G_U is a signal graph. In G_V, V is a finite set of gene products, $E_r \subseteq V \times V$ is a repression relation, and $E_a \subseteq V \times V$ is an activation relation. In G_U, U is a finite set of input signals, $I_r \subseteq U \times (V \cup E_r \cup E_a)$ is a signal repression relation, and $I_a \subseteq U \times (V \cup E_r \cup E_a)$ is a signal activation relation. Gene regulatory networks are typically represented as a directed graph, where V and U are sets of vertexes, E_r and E_a are directed repression and activation edges connecting nodes in V, and I_r and I_a are directed signal repression and signal activation edges connecting nodes in U to nodes or edges in G_V. For example, in Figure 1B, green, orange, and purple circles denote gene products, and START, a, and b denote input signals to the network. Repression edges that connect nodes denote repression relations between those gene products. For example, the edge connecting R0 to Ta0 indicates that R0 represses the production of Ta0. Some signal repression edges connect nodes to other edges, for example, a signal repression edge connects a to the edge between Sa and Ta0. This denotes a biomolecular reaction where signal a prevents Sa from repressing Ta0. These directed graphs provide a high level description of the relationship between genes and input signals in a regulatory network.

Remark For simplicity of interpretation and implementation, we are mainly concerned with gene networks made of only repressing relations, and will specify these networks as

$$G = (V, E_r, U, I_r) \tag{5}$$

where E_a and I_a are empty and therefore not shown.

Biomolecular Parts. Specific biomolecular parts can be used to implement high level GRN specifications. Figure 2 illustrates

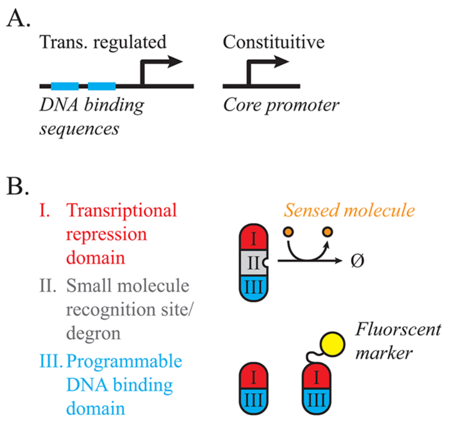

A.
Trans. regulated
Constituitive
DNA binding sequences
Core promoter
B.
I. Transriptional repression domain
II. Small molecule recognition site/ degron
III. Programmable DNA binding domain
Sensed molecule
Ø
Fluorscent marker
I
II
III

Figure 2. Biomolecular parts for realizing finite state machines: (A) Transcriptionally regulated and unregulated promoters. Transcriptionally regulated promoters have specific DNA sequences illustrated as blue bars that may be bound by domain III on a repressing transcription factors. Transcriptionally unregulated promoter is nominally "on". (B) Transcription factors are made of three primary components: a transcriptional repression domain; an optional signaling molecule binding site; and a programmable DNA binding domain. Sensed molecules are small diffusable signaling molecules that bind to recognition sites (i.e., a degron) in programmable transcription factors and catalyze degradation of the transcription factor. Also illustrated is a fluorescently labeled transcription factor, which we use to distinguish "state" proteins from "transition" proteins.

a class of parts for transcription regulation and small molecule sensing that we use throughout this paper. Our goal here is simply to show what a minimal set of parts can do. It will become apparent that the parts used could be replaced with similar parts, or that the designs we propose could be made more robust or efficient.

We consider gene networks made of promoters, small diffusable signal molecules, and repressing transcription factors with programmable DNA binding domains. Figure 2a shows two varieties of promoters: *constitutive* and *transcriptionally regulated*. Constitutive promoters are always "on", expressing

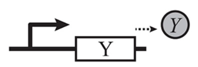
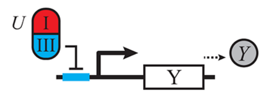
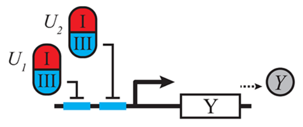
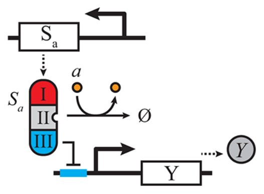
Component Type	Biomolecular Realization	GRN	Boolean Network Equations	Delay Differential Equations
Transcriptionally Unregulated Gene		Y	$Y^t = on$	$\frac{d}{dt}Y(t) = V_{max} - \beta Y(t)$
Singly Regulated Gene		$U \dashv Y$	$Y^{t+1} = \neg U^t$	$\frac{d}{dt}Y(t) = \frac{V_{max}}{1+\left(\frac{U(t-\tau)}{k_{1/2}}\right)^n} - \beta Y(t)$
Doubly Regulated Gene		$U_1 \dashv Y$, $U_2 \dashv Y$	$Y^{t+1} = \neg(U_1^t \vee U_2^t)$	$\frac{d}{dt}Y(t) = \frac{V_{max}}{1+\left(\frac{U_1(t-\tau)+U_2(t-\tau)}{k_{1/2}}\right)^n} - \beta Y(t)$
Small Molecule Sensor		$a \dashv S_a \dashv Y$	$S_a^t = \neg a^t$; $Y^{t+1} = \neg S_a^t$	$\frac{d}{dt}S_a(t) = V_{max} - (\beta + k_p a(t))S_a(t)$; $\frac{d}{dt}Y(t) = \frac{V_{max}}{1+\left(\frac{S_a(t-\tau)}{k_{1/2}}\right)^n} - \beta Y(t)$

Figure 3. Components of a biomolecular realization of finite state machines, represented as a GRN, Boolean network equations, and delay differential equations. In the delay differential equation representation, τ denotes the delay associated with transcription and translation, β is the rate of dilution associated with cell growth, and V_{max} is the maximum rate of production of a gene product. The delay differential equations follow a Michaeles–Menten form where $k_{1/2}$ and n are the Michaeles constant and Hill coefficient, respectively.

their associated gene product at some nominal level. Transcriptionally regulated promoters contain one or more specific DNA binding sequences upstream of a core promoter. We consider transcriptionally regulated promoters that are nominally "on" unless bound by one or more repressing transcription factors.

Figure 2b shows several transcription factors made by fusing two or three functional domains: a transcriptional repression domain (I); a degron domain (II); a DNA binding domain (III). The construction of such transcription factors is now routine.[16,36,38−40] All transcription factors we consider contain domains I and III, and special signal sensing transcription factors contain domain II. Transcription factors might be fluorescently labeled, and for clarity, we use this labeling to distinguish "state" transcription factors from "transition" transcription factors. Small signal molecules (represented as small circles) bind to and catalyze the degradation of specific "sensor" transcription factors. Signals may be inducers or cell−cell signaling molecules that can be enzymatically produced and exported by the cell and other cells or supplied exogenously. In this way, signals may be applied exogenously as input to a GRN or produced internally as an intercellular communication medium in multicellular systems.

In Figure 3, we depict biomolecular realizations and corresponding gene regulatory networks for the four types of components that we interconnect to construct finite state machines: the *transcriptionally unregulated gene*, the *singly regulated gene*, the *doubly regulated gene*, and the *small molecule sensor*. The first three components represent motifs for gene regulation via repressing transcription factors, while the fourth component will be used to sense a small input signal molecule. The biomolecular parts shown in Figure 2 and networks made from the composition of these parts can be modeled as Boolean networks, as illustrated in Figure 3. This is just one possible way to implement gene regulatory networks, and there are certainly others. For example, a similar implementation can be imagined with CRISPR transcription factors.[32] Now that we have defined the syntax of gene regulatory networks and a set of biomolecular parts to model, we will use Boolean network dynamics and delay differential equations to provide semantic interpretation as dynamical systems.

Boolean Network Model. The *Boolean network* model of gene regulation is a discrete time dynamical system used to model course-grained dynamics of gene expression.[41,42] Boolean networks were first introduced in this application to investigate the dynamics of randomly generated gene regulatory networks and have been used successfully as a descriptive tool, and more recently (and perhaps surprisingly) as a predictive model for gene regulation dynamics in natural and synthetic systems.[43−47] In a Boolean network, the expression level of each gene product or input to the network is in one of two states: *on* and *off*, (or *true* and *false*) denoting high or low expression level respectively. For brevity, we define the domain $\mathbb{B} \triangleq \{on, off\}$. Boolean states are updated at discrete times, and the expression level of a gene product at time $t + 1$ is a Boolean function of the expression level of gene products that affect it at time t. In general, for gene products $Y_1, Y_2,...,Y_n$ and inputs $U_1, U_2,...,U_m$, we denote the state of the gene products and inputs at time t as $Y_1^t, Y_2^t,...,Y_n^t \in \mathbb{B}$ and $U_1^t, U_2^t,...,U_m^t \in \mathbb{B}$ respectively. The dynamics of this network can then be written as follows:

$$Y_1^{t+1} = f_1(Y_1^t, Y_2^t, ..., Y_n^t, U_1^t, U_2^t, ..., U_m^t) \tag{6}$$

$$Y_2^{t+1} = f_2(Y_1^t, Y_2^t, ..., Y_n^t, U_1^t, U_2^t, ..., U_m^t) \tag{7}$$

$$\vdots \tag{8}$$

$$Y_n^{t+1} = f_n(Y_1^t, Y_2^t, ..., Y_n^t, U_1^t, U_2^t, ..., U_m^t) \tag{9}$$

where f_i: $\mathbb{B}^{n+m} \to \mathbb{B}$ is some Boolean update function for i = 1,2,...,n.

Remark For simplicity, we use the same symbol for the name of the gene product and the time-varying state of the gene product. Additionally, we name genes without subscripts, and gene products and time-varying state with subscripts. In general, the meaning of these symbols should be clear from context.

In the Boolean network model, the update function f_i typically corresponds to a mechanistic model of transcriptional regulation. In the networks we consider, all transcription factors are repressing, and all promoters are nominally "on" unless bound by some repressing transcription factor. This means the transcriptionally unregulated gene motif in Figure 3 always expresses its gene product. The Boolean network dynamics for the transcriptionally unregulated gene encodes the sequential logic for *true*. In the singly regulated gene shown in Figure 3, the input U is a repressing transcription factor with a DNA binding domain that binds to and represses transcription of gene Y, preventing expression of gene product Y. The Boolean network dynamics for the singly regulated gene encodes the sequential logic for *NOT*. In other words, gene product Y is *on* unless the input U was present at the previous time step. Similarly, the doubly regulated gene in Figure 3, has two inputs U_1 and U_2 that may each bind to and repress transcription of gene Y. The Boolean network dynamics for the doubly regulated gene encodes the sequential logic for *NOR*, meaning the output Y is *on* unless input U_1 or input U_2 was present at the previous time step. Finally, the small molecule sensor in Figure 3 is a two-gene component that makes use of two additional parts from Figure 2: a signal molecule a, and a transcription factor S_a that contains the degron domain (II) that is sensitive to a. The transcription factor S_a is transcriptionally unregulated, and in the absence of signal a, S_a binds to the upstream binding sequence of gene Y and prevents expression of gene product Y (same as the singly regulated gene). However, in the presence of a, the signal molecule quickly binds to S_a and catalyzes its degradation through fast protein−protein interactions. The resulting absence of S_a allows gene product Y to be expressed. This interaction is denoted in the GRN in Figure 3 by a repression arrow pointing from a to the repression arrow connecting $\mathrm{S_a}$ to Y. Here, a^t is an input to the network. Since the interaction between S_a and a is much faster than gene expression, we consider the state S_a^t to be a function of a^t,

$$S_a^t = \neg\, a^t \tag{10}$$

The Boolean network dynamics of the small molecule sensor component are then,

$$Y^{t+1} = \neg\, S^t \tag{11}$$

$$= a^t \tag{12}$$

The advantage of a Boolean network model is that it allows for the complete enumeration of the state space, which we use later to show that for any FSM a Boolean network can be constructed from the components in Figure 3 such that the dynamics of the Boolean network are isomorphic to the transition function of the FSM. The disadvantage of such a model is that it may make unrealistic simplifying assumptions and lacks the fidelity of a continuous system such as a network modeled by Hill functions or chemical reaction networks. For this reason, we also consider a delay differential equation model of gene expression.

Delay Differential Equation Model. There are some questions that cannot be addressed assuming the Boolean network model. Gene expression levels are in general not binary and depend on factors such as binding affinities of DNA binding domains and dilution as a result of cell growth and division. Ultimately, any model of a gene regulatory network depends on the specific biomolecular components used to implement the system. However, it is useful to consider a simple continuous model to examine how well a motif for implementing finite state machines approximates the discrete Boolean network model under various kinetic parameters. To this end, we use DDEs and Hill equations[44,48] to model the dynamics of gene product concentrations in the components outlined in Figure 3. Notably, we will make the following simplifying assumptions about the gene network: all regulated and unregulated genes are built around the same core promoter, all gene products have the same nominal rates of translation and transcription, all proteins have approximately the same rate of nominal dilution and degradation, and all transcription factors have similar binding affinities and cooperativity. These are reasonable design assumptions that greatly reduce the number of parameters in the model.

The DDE for the transcriptionally unregulated gene in Figure 3 is

$$\frac{\mathrm{d}}{\mathrm{d}t}Y(t) = V_{\max} - \beta Y(t) \tag{13}$$

where $Y(t) \in \mathbb{R}$ is the time-varying concentration of gene product Y, and $V_{\max} \in \mathbb{R}$ and $\beta \in \mathbb{R}$ are tunable parameters to the model. $V_{\max}$ is the rate of transcription and translation of gene Y, and β denotes the cumulative rate dilution due to cell growth and nominal protein degradation. In steady state,

$$\lim_{t\to\infty} Y(t) = \frac{V_{\max}}{\beta} \tag{14}$$

The DDE for describing the dynamics of the singly regulated gene component in Figure 3 is

$$\frac{\mathrm{d}}{\mathrm{d}t}Y(t) = \frac{V_{\max}}{1 + \left(\frac{U(t-\tau)}{k_{1/2}}\right)^n} - \beta Y(t) \tag{15}$$

The first term in this DDE takes the form of a Hill function. As before, $V_{\max}$ is the maximum unregulated rate of transcription, and β is a cumulative diffusion and nominal protein degradation rate. In addition, $U(t) \in \mathbb{R}$ is the time-varying concentration of repressing transcription factor U, and $k_{1/2} \in \mathbb{R}$, $n \in \mathbb{R}$, and $\tau \in \mathbb{R}$ are three new tunable parameters. $k_{1/2}$ is the concentration of $U(t)$ required to half the rate of transcription of gene Y, and n is the Hill coefficient for repression. τ denotes the time delay from transcription of a gene to translation of the gene product. Similarly, the DDE for the doubly regulated gene component is

$$\frac{\mathrm{d}}{\mathrm{d}t}Y(t) = \frac{V_{\max}}{1 + \left(\frac{U_1(t-\tau) + U_2(t-\tau)}{k_{1/2}}\right)^n} - \beta Y(t) \tag{16}$$

This is the same as eq 15 for the singly regulated gene, except that here the time-delayed concentration of both repressing transcription factors U_1 and U_2 are summed in the Hill function. This assumes that both U_1 and U_2 bind to the same DNA binding sequence and have identical effects on the transcription of gene Y. Finally, two equations model the dynamics of the small molecule sensor in Figure 3,

$$\frac{\mathrm{d}}{\mathrm{d}t}S_a(t) = V_{\max} - (\beta + k_p a(t))S_a(t) \tag{17}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}Y(t) = \frac{V_{\max}}{1 + \left(\frac{S_a(t-\tau)}{k_{1/2}}\right)^n} - \beta Y(t) \tag{18}$$

Equations 13 and 17 captures the dynamics of production and degradation of sensor molecule S_a via signal molecule a. As before, $V_{\max}$ is the maximum rate of production of S_a and β is the cumulative rate of diffusion and nominal degradation. Additionally, S_a is degraded through fast protein−protein interactions catalyzed by signal a. The kinetics of these reactions are lumped into a single rate parameter k_p. Equation 18 then models the production of output molecule Y regulated by repressing transcription factor S_a just as with the singly regulated gene in eq 15.

General Construction of a GRN from an FSM Specification. To implement an FSM with a GRN, we define states and input symbols in terms of the parts and components illustrated in Figures 2 and 3. A natural choice is to represent a state in the FSM with a particular gene expression level or set of gene expression levels and represent input symbols with the presence or absence of signal molecules that alter the gene expression level. A sequence of input symbols could then be encoded as a trajectory over signal concentrations. The GRN implementation of an FSM machine would then act as a temporal sequence recognizer over signal concentrations, and the notion of a final accepting state would encode the logic for recognizing a particular set of trajectories over input signals. Specifically, beginning with the FSM,

$$M = (Q, \Sigma, \delta, q_0, F) \tag{19}$$

we generate the gene regulatory network

$$(V, E_r, U, I_r) = g(M) \tag{20}$$

where V is the set of gene products in the network, E_r describes how the proteins in V are "wired" together as a GRN, U is the set of input signals to the network, and I_r describes how the signal molecules affect transcription of the genes in V.

Inputs to the network are a set of signal molecules Σ and an inducible repressing transcription factor START,

$$U = \Sigma \cup \{\mathrm{START}\} \tag{21}$$

Every gene in the network encodes a repressing transcription factor, and we break these into three categories: *state genes*, *transition genes*, and *sensor genes*. For clarity, we will name all the state genes "Rq" where q is the name of state in the FSM. The state genes are defined as a set of singly repressed genes,

$$V_R = \{\mathrm{R}q | \ \forall\ q \in Q\} \tag{22}$$

For $p \in Q$, we will consider the GRN to be in state p when Rp is at a *low* level of expression, and for all $q \in Q$ where $q \neq p$, Rq is at a *high* level of expression.

Similarly, we will prefix the names of all transitions genes with a "T". The transition genes are a set of doubly repressed genes,

$$V_T = \{\mathrm{T}\sigma q | \ \forall\ q \in Q,\ \sigma \in \Sigma \text{s.t.}\ \exists\ \delta(q, \sigma)\} \tag{23}$$

For $q \in Q$ and $\sigma \in \Sigma$, we consider the GRN to be following transition $\delta(q, \sigma)$ when Tσq is at a *high* level of expression, and all other transition genes are at a *low* level of expression.

Finally, all the sensor gene names are prefixed with a "S". Sensor genes are a set of transcriptionally unregulated genes,

$$V_S = \{\mathrm{S}\sigma | \ \forall\ \sigma \in \Sigma\} \tag{24}$$

where each signal molecule σ binds uniquely to the sensor transcription factor Sσ. For $\sigma \in \Sigma$, we consider the symbol σ to be applied to the machine when the signal molecule σ is present.

The repression edges E_r are then made up of edges between genes. Specifically, let $E_{R,T}$ be the set of repression edges from state genes to transition genes. For each state $q \in Q$, state gene Rq represses transition genes Tσq for all $\sigma \in \Sigma$.

$$E_{R,T} = \{(\mathrm{R}q, \mathrm{T}\sigma q) | \ \forall\ \mathrm{R}q \in V_R,\ \mathrm{T}\sigma q \in V_T\} \tag{25}$$

Let $E_{T,R}$ be the set of repression edges from transition genes to state genes. To encode the transition function, for all $q,q' \in Q$ and $\sigma \in \Sigma$ where $\delta(q,\sigma) \rightarrow q'$, transition gene T$\sigma q$ should repress state gene Rq'.

$$E_{T,R} = \{(\mathrm{T}\sigma q, \mathrm{R}q) | \ \forall\ \mathrm{T}\sigma q \in V_T,\ \mathrm{R}q \in V_R\} \tag{26}$$

Let $E_{S,T}$ be the set of repression edges from sensor genes to transition genes. For each input symbol $\sigma \in \Sigma$, sensor gene Sσ represses transition genes Tσq for all $q \in Q$.

$$E_{S,T} = \{(\mathrm{S}\sigma, \mathrm{T}\sigma q) | \ \forall\ \mathrm{S}\sigma \in V_S,\ \mathrm{T}\sigma q \in V_T\} \tag{27}$$

Let $I_{\Sigma,S}$ be the set of repression edges from signal molecules in Σ to the edges in $E_{S,T}$,

$$I_{\Sigma,S} = \{(\sigma, (\mathrm{S}\sigma, \mathrm{T}\sigma q)) | \ \forall\ \sigma \in \Sigma,\ (\mathrm{S}\sigma, \mathrm{T}\sigma q) \in E_{S,T}\} \tag{28}$$

Finally, encoding the start state q_0, START should repress start gene Rq_0. Then, for $(V,E_r,U,I_r) = g(M)$,

$$V = V_R \cup V_T \cup V_S \tag{29}$$

$$E_r = E_{R,T} \cup E_{T,R} \cup E_{S,T} \tag{30}$$

$$U = \Sigma \cup \{\mathrm{START}\} \tag{31}$$

$$I_r = I_{\Sigma,S} \cup \{(\mathrm{START}, \mathrm{R}q_0)\} \tag{32}$$

Boolean Network Model of the General Construction. In general, the Boolean network equations for a sensor gene Sσ and a transition gene Tσq are

$$S_\sigma^t = \neg\ \sigma^t \tag{33}$$

$$T_{\sigma,q}^{t+1} = \neg\ (R_q^t \vee S_\sigma^t) \tag{34}$$

where S_σ^t, $T_{\sigma,q}^t$, and R_q^t denote the Boolean value of the sensor gene, transition gene, and state gene, respectively, at time t. There are two forms of the equation describing the dynamics of each state gene Rq. When $q \neq q_0$,

$$R_q^{t+1} = \neg \bigvee_{\{(q',\sigma) | \delta(q',\sigma) \rightarrow q\}} T_{\sigma,q'}^t \tag{35}$$

and when $q = q_0$,

$$R_q^{t+1} = \neg\ \Big(\Big(\bigvee_{\{(q',\sigma) | \delta(q',\sigma) \rightarrow q\}} T_{\sigma,q'}^t\Big) \vee \mathrm{START}^t\Big) \tag{36}$$

In the absence of any inputs $\sigma \in \Sigma$ and a low expression of START, the network reaches steady-state in two time steps. The expression level at steady state is

$$S_\sigma^t = \text{on} \tag{37}$$

$$T_{\sigma,q}^t = \text{off} \tag{38}$$

$$R_q^t = \text{on} \tag{39}$$

for all $\sigma \in \Sigma$ and $q \in Q$. At steady state, the network is not in any state of the FSM, nor is it following any transitions.

In the Boolean network framework, an input sequence to the FSM is encoded as a trajectory over signal molecules and the START gene activity. Let the set of input symbols $\Sigma = \{\sigma_1, \sigma_2, ..., \sigma_n\}$, and let $w = \sigma_{c1}\sigma_{c2}...\sigma_{cm} \in \Sigma^*$ be an input string to the FSM M where the index $c_i \in \{1,...,n\}$ for $i = 1...m$. In the Boolean network model of the GRN $g(M)$, the sequence w specifies an input trajectory $u^t = h_{BN}(w,t)$ for $t \in \mathbb{N}$, defined as

$$h_{BN}(w, t) = \begin{bmatrix} \text{START}^t \\ \sigma_1^t \\ \sigma_2^t \\ \vdots \\ \sigma_n^t \end{bmatrix} \tag{40}$$

where

$$\text{START}^t = \begin{cases} \text{on}, & t \in \{0, 1\} \\ \text{off}, & \text{otherwise} \end{cases} \tag{41}$$

and for each σ_j^t with $j \in \{1,2,...,n\}$,

$$\sigma_j^t = \begin{cases} \text{on}, & \exists\ c_i \text{ s.t. } j = c_i \text{ and } t \in \{2i, 2i + 1\} \\ \text{off}, & \text{otherwise} \end{cases} \tag{42}$$

Exactly one signal or the START gene is active at any particular time. The START gene is *on* for two time steps (i.e., from $t = 0$ through $t = 1$). Immediately following the START pulse, each signal σ_{ci} (for $i = 1,2,...,m$) is *on* in sequence for two time steps. An example trajectory is shown in Figure 4 and discussed later in detail.

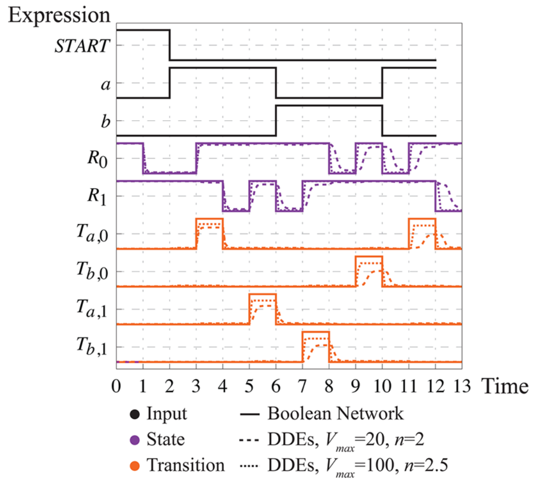


Figure 4. Boolean network and DDE trajectories for the two-state FSM from Figure 1. In this simulation, $\tau = 1$, $n = 2$, $V_{\max} = \beta = 20$, $k_{1/2} = 0.2$, $k_p = 200$. Solid lines denote Boolean network trajectories and dotted lines denote DDE trajectories. The top three plots show the input trajectories of a, b, and START that encode the sequence "aabba". The control input for the Boolean network and DDE model are identical. The middle two plots show to the expression level of state genes R0 and R1 where low expression of Ri corresponds to being in state i of the FSM. The bottom four plots illustrate the expression level of transition genes Ta0, Tb0, Ta1, and Tb1, where high expression of Tσq denotes the transition $\delta(q, \sigma)$ in the FSM.

DDE Model of the General Construction. Similarly, the dynamics of a biomolecular realization of sensor and transition genes can be writen as a system of DDEs. Where $\sigma(t)$ is the concentration of signal molecule σ at time t, and $S_\sigma(t)$, $T_{\sigma,q}(t)$, $R_q(t)$, and START(t) are the concentrations of sensor, transition, state, and "start" transcription factors respectively,

$$\frac{d}{dt}S_\sigma(t) = V_{\max} - (\beta + k_p\sigma(t))S_\sigma(t) \tag{43}$$

$$\frac{d}{dt}T_{\sigma,q}(t) = \frac{V_{\max}}{1 + \left(\frac{S_\sigma(t-\tau) + R_q(t-\tau)}{k_{1/2}}\right)^n} - \beta T_{\sigma,q}(t) \tag{44}$$

for each $\sigma \in \Sigma$ and $q \in Q$. As with the Boolean network representation, there are two forms of the DDE describing the dynamics of a state gene. When $q \neq q_0$,

$$\frac{d}{dt}R_q(t) = \frac{V_{\max}}{1 + \left(\frac{\text{START}(t-\tau) + \sum_{(q',\sigma)|\delta(q',\sigma)\to q} T_{\sigma,q'}(t-\tau)}{k_{1/2}}\right)^n} - \beta R_q(t) \tag{45}$$

When $q = q_0$,

$$\frac{d}{dt}R_q(t) = \frac{V_{\max}}{1 + \left(\frac{\sum_{(q',\sigma)|\delta(q',\sigma)\to q} T_{\sigma,q'}(t-\tau)}{k_{1/2}}\right)^n} - \beta R_q(t) \tag{46}$$

In general, the steady-state of this system of DDEs can be solved for numerically given the control inputs $S_\sigma(t) = 0\ \forall \sigma \in \Sigma$ and START$(t) = 0$. The steady state of the DDEs qualitatively reflect the steady state of the Boolean network for a large range of parameters.

As with the Boolean network model, an input sequence to the FSM is encoded as an input trajectory $u(t) = h_{DDE}(w,\Delta t,t)$ for $t \in \mathbb{R}$ and *pulse width* $\Delta t \in \mathbb{R}$, defined as

$$h_{DDE}(w, t) = \begin{bmatrix} \text{START}(t/\Delta t) \\ \sigma_1(t/\Delta t) \\ \sigma_2(t/\Delta t) \\ \vdots \\ \sigma_n(t/\Delta t) \end{bmatrix} \tag{47}$$

where

$$\text{START}(t) = \begin{cases} 1, & t \in [0, 1) \\ 0, & \text{otherwise} \end{cases} \tag{49}$$

and for each σ_j with $j \in \{1,2,...,n\}$

$$\sigma_j(t) = \begin{cases} 1, & \exists\ c_i \text{ s.t. } j = c_i \text{ and } t \in [2i, 2i + 1) \\ 0, & \text{otherwise} \end{cases} \tag{51}$$

As with the Boolean network model, exactly one signal or the START gene is active at any given time. Here, Δt controls for the input signal pulse width, typically taken to be $\Delta t \geq \tau$. In 4, the pulse width $\Delta t = \tau$, making the sample input trajectory for the DDE model match the input to the Boolean network.

Example: Two-State Machine as a Boolean Network. As an example, Figure 1A depicts a simple two-state FSM represented as a directed graph, along with a GRN implementation and a biomolecular realization of the same machine. This machine has two states, $Q = \{0,1\}$. The FSM begins in state $q_0 = 0$, and from here, two transitions are possible. If the next input symbol is an "a", then the machine stays in state 0. Conversely, from state 1, and "a" leaves the machine in state 1, while a "b" moves the machine back to state 0. Since the set of accepting states F consists only of state 1, this fSM accepts all sequences of "a" and "b" that end in "a".

In the GRN implementation in Figure 1B, the state 0 is represented by the low expression of gene R0, and the state 1 is represented by the low expression of gene R1. Modeled as a Boolean network, the equations describing the dynamics of this GRN are

$$R_0^{t+1} = \neg\,(T_{b,0}^t \vee T_{b,1}^t \vee \mathrm{START}^t) \tag{52}$$

$$R_1^{t+1} = \neg\,(T_{a,0}^t \vee T_{a,1}^t) \tag{53}$$

$$T_{a,0}^{t+1} = a^t \wedge \neg\, R_0^t \tag{54}$$

$$T_{b,0}^{t+1} = b^t \wedge \neg\, R_0^t \tag{55}$$

$$T_{a,1}^{t+1} = a^t \wedge \neg\, R_1^t \tag{56}$$

$$T_{b,1}^{t+1} = b^t \wedge \neg\, R_1^t \tag{57}$$

Figure 4 shows the Boolean network and DDE trajectories for this GRN given the input sequence "aabba". In the Boolean network, the initial high expression of START at time $t = 0$ through $t = 1$ results in a low expression of R0 during time $t = 1$ through $t = 2$, denoting the state $q_0 = 0$ in the FSM. Since every transition gene Tσq is repressed by both state gene Rq and sensor gene Sσ, low expression of Rq at time t means that the expression level of Tσq at time $t + 1$ is sensitive to signal σ at time t. Specifically, from eqs 52–57, when R0 is *off*,

$$T_{a,0}^{t+1} = a^t \tag{58}$$

$$T_{b,0}^{t+1} = b^t \tag{59}$$

When the signal a is present at time $t = 2$, repression is temporarily relieved on transition gene Ta0 at time $t = 3$. The high expression of Ta0 indicates transition $\delta(0,a)$ in the FSM. In general, the presence of an input signal coinciding with the repression of a state gene determines which transition gene will be active at the following time step. Transition gene Ta0 represses state gene R1 (state 1 in the FSM) at time $t = 4$, leaving transition genes Ta1 and Tb1 sensitive to signal molecule a. Since signal a is still active at time $t = 4$, Ta1 is expressed at time $t = 5$ and R1 is again repressed at time $t = 6$ (in the FSM, $\delta(1,a)\rightarrow 1$). This process is repeated for the subsequent long pulse of signal b and short pulse of signal a (encode two "b" input symbols followed by an "a"). After the final pulse of signal a is supplied, the GRN arrives in the final accepting state, and R1 is repressed at time $t = 12$. In this example the transition function δ was defined for all combinations of input symbols and states; however, as shown in the Supporting Information, if the simulation were to continue absent of input signals, or if an input symbol was provided that did not correspond to a valid transition, the Boolean network model of this GRN would return to its initial steady state in two time steps.

Here, assuming a Boolean network model, the set of input sequences to the two-state FSM are exactly the set of input sequences that end in an accepting state in the GRN realization of the two-state FSM. In computer science, given two models of computation, if behavior of any machine constructed in one model can be recapitulated with a machine constructed in the other model, we say that one model is *simulated* by the other. In fact, given any FSM the general construction can be used to design a GRN that recapitulates the behavior of the FSM. In other words, assuming a Boolean network model, GRNs simulate FSMs.

Theorem 1. *Given a finite state machine* $M = (Q, \Sigma, \delta, q_0, F)$, *the gene regulatory network* $(V, E_r, U, I_r) = g(M)$ *simulates* M *when modeled as a Boolean network.*

Additionally, because any Boolean network can be simulated by an FSM, *any* gene regulatory network modeled as a Boolean network is no more or less capable than a finite state machine.

Corollary 1. *Modeled as a Boolean network, gene regulatory networks are computationally equivalent to finite state machines.*

A precise definition of simulation, and proofs for Theorem 1 and Corollary 1 are provided in the Supporting Information.

Example: Two-state Machine as a System of DDEs. As mentioned previously, gene expression levels are, in general, not binary. In order to assess continuous effects such as production, dilution, and degradation rates, and binding affinities, we model the GRN as a system of delay differential equations. The equations describing the continuous dynamics of the biomolecular realization in Figure 1C are

$$\frac{\mathrm{d}}{\mathrm{d}t}S_a(t) = V_{\max} - (\beta + k_p a(t))S_a(t) \tag{60}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}S_b(t) = V_{\max} - (\beta + k_p b(t))S_b(t) \tag{61}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}T_{a,0}(t) = \frac{V_{\max}}{1 + \left(\frac{S_a(t-\tau) + R_0(t-\tau)}{k_{1/2}}\right)} - \beta T_{a,0}(t) \tag{62}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}T_{b,0}(t) = \frac{V_{\max}}{1 + \left(\frac{S_b(t-\tau) + R_0(t-\tau)}{k_{1/2}}\right)} - \beta T_{b,0}(t) \tag{63}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}T_{a,1}(t) = \frac{V_{\max}}{1 + \left(\frac{S_a(t-\tau) + R_1(t-\tau)}{k_{1/2}}\right)} - \beta T_{a,1}(t) \tag{64}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}T_{b,1}(t) = \frac{V_{\max}}{1 + \left(\frac{S_b(t-\tau) + R_1(t-\tau)}{k_{1/2}}\right)} - \beta T_{b,1}(t) \tag{65}$$

Here, we use the same time delay τ, rate parameters $V_{\max}$, β, and k_p, and Hill parameters $k_{1/2}$ and n throughout the system. This is a sensible choice assuming all genes use the same core promoter, and all transcription factors use the same repression domain and carefully tuned DNA binding domains. A sample trajectory of this system is shown in Figure 4. Choosing $\tau = 1$ allows direct comparison to the Boolean network trajectory with the same control input. In an experimental system, the stepwise constant input could be approximated by washing media across cells in a microfluidic device. For the other parameters, $k_{1/2} = 0.2$, $k_p = 200$, and $V_{\max}$, β, and n were varied.

In Figure 4 the DDE simulations qualitatively track the trajectories of the Boolean network model. Initially both DDE simulations track quite well. However, by time $t = 6$ the behavior of the DDE model for $n = 2$ and $V_{\max} = \beta = 20$ appears to lag compared to the ideal Boolean network model, and by time $t = 12$, the DDE model appears to be lagging by nearly a

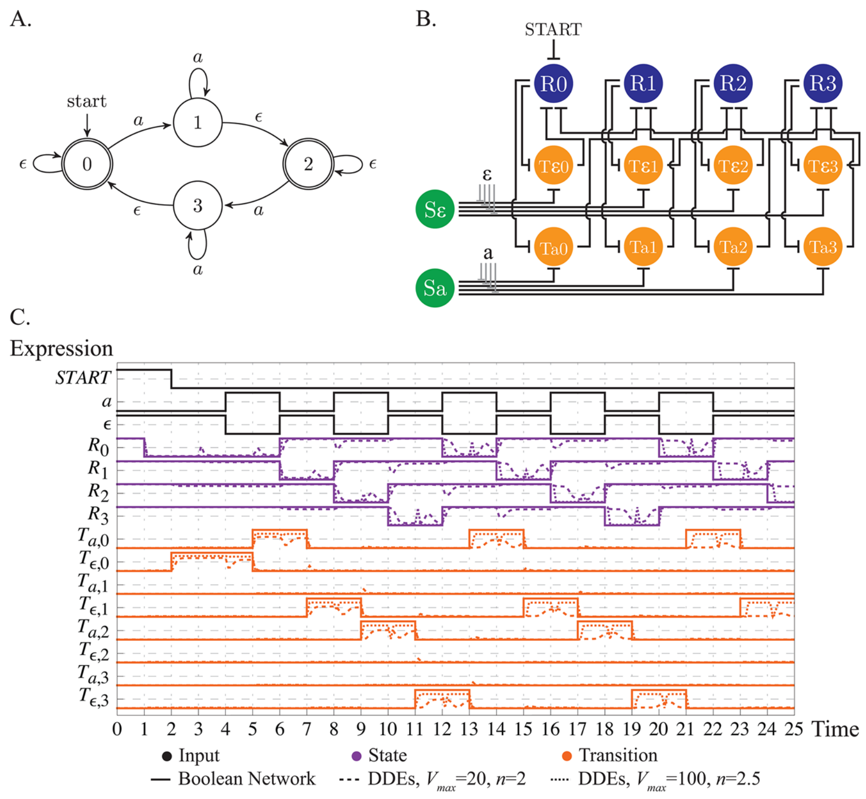


Figure 5. Modulo-two pulse counter machine described as (A) a directed graph representa-tion of a finite state machine, (B) a gene regulatory network. In the GRN representation, an unspecified mechanism makes the ε signal available except in the presence of the a signal. (C) Boolean network and DDE trajectories for the modulo-two pulse counter machine specified by the GRN. In this simulation, $\tau = 1$, $k_{1/2} = 0.2$, and $\beta = V_{\max}$. $V_{\max}$ and n are varied, with larger values of $V_{\max}$ and n resulting in trajectories that more closely follow the ideal Boolean network trajectories.

half time unit. Additionally, the maximum amplitude of the gene expression levels decreases over time, and this is particularly evident with the transition genes. By comparison, the DDE model for $V_{\max} = \beta = 100$ and $n = 2.5$ tracks the Boolean network model without significant lag or change in maximum amplitude of expression. There are many methods for improving the behavior of the DDE model. One possible solution is to re-engineer the finite state machine. Other FSMs may accept the same language, and result in a different GRN that is more robust to the same input. In general, we would like to know, given an FSM and GRN implementation of that FSM, how robust is the GRN implementation to changes in the parameters $V_{\max}$, β, n, $k_{1/2}$, and τ.

DDEs Approximate the Behavior of the Boolean Network. One method for examining how well the behavior of the DDE model approximates the ideal Boolean network model is to simply compare the state of each model after applying a prescribed input. To illustrate this method, we introduce a new example, the modulo-two pulse counter. The finite state machine encoding the modulo-two pulse counter is shown in Figure 5A. There are two input symbols to this machine, $\Sigma = \{a, \varepsilon\}$, and the machine accepts all sequences of input symbols that end in an ε. Viewed another way, any accepted input sequence can be split into a sequence of contiguous a symbols interrupted by sequences of contiguous "ε" symbols. In this way, an accepted input sequence consisting of an even number of contiguous a sequences should end in state 0, while an accepted input sequence consisting of an odd number of continuous a sequences should end in state 2. The GRN implementation of this machine is shown in Figure 5B. If we imagine a biomolecular implementation where signal molecule ε is available except in the presence of the input signal a, this machine counts discrete pulses of a modulo two.

The Boolean network model of the GRN in Figure 5B simulates the FSM shown in Figure 5A. Initially, input signal a is absent and signal ε is present, all state genes are *on*, and all transition genes are *off*. At time $t = 0$, the START gene is *on*, leading to the repression of R_0 at time $t = 1$. The machine then transitions between states 0 through 3 according to the prescribed pulses of input signal a. Figure 5C illustrates a few sample trajectories of the DDE model against the trajectory produced by the ideal Boolean network model for an input of five pulses of equal duration of signal molecule a followed by signal molecule ε. In these trajectories, $\beta = V_{\max}$, $\tau = 1$, $k_{1/2} = 0.2$, and $V_{\max}$ and n were varied.

Intuitively, for a fixed τ, increasing $V_{\max}$ and β will improve the dynamic response of the DDE model, and increasing the Hill coefficient n results in a sharper sigmoidal response. This is reflected in Figure 5C where the DDE model for $V_{\max} = \beta = 100$ and $n = 2.5$ track the ideal Boolean network trajectory more closely than the model where $V_{\max} = \beta = 20$ and $n = 2$. Additionally, varying $k_{1/2}$ affects the sensitivity of expression to upstream transcription factors. The duration of the input pulse may also be increased relative to τ. For example, the control input shown in Figure 5C applies *START* followed by five pulses of the input signal a according to a square wave with pulse width of $\Delta t = 2$. A precise value for τ may not be known, and in networks like the modulo-two pulse counter, more reliable performance may be achieved by allowing the network to settle into a periodic orbit before changing inputs.

To quantitatively measure the effects of these parameters, the following two metrics compare the state of gene expression at a fixed time after applying a control input in a DDE model to the ideal Boolean network. Where $\hat{R}_q(t)$ and $R_q(t)$ are the values of expression for gene Rq in the DDE model and Boolean network model respectively,

$$e_{\text{avg}} = \max_{q \in Q} \frac{2}{\Delta t} \int_{12\Delta t}^{12.5\Delta t} |R_q(t) - \hat{R}_q(t)| \mathrm{d}t \tag{66}$$

$$e_{\text{thresh}} = \begin{cases} 0, & e_{\text{avg}} < 1/2 \\ 1, & e_{\text{avg}} \geq 1/2 \end{cases} \tag{67}$$

Equation 66 describes error e_{avg}, which is the L_∞ norm on the average error eq 66 takes the maximum average error e_{avg}, between the DDE model and Boolean network model for the gene products R_q for all $q \in Q$ for half a pulse width following the five pulses of the input signal a. In Figure 5, e_{avg} the maximum average difference between the DDE model and Boolean network model for gene products R_0, R_1, R_2, and R_3 on the time interval t = 24 to t = 25. Equation 67 digitizes this error with a threshold of 1/2 as e_{thresh}. Figure 6 shows the average error and thresholded error over a range of parameters for the pulse counter machine given $V_{\max} = \beta$ and τ = 1, and nominal values of Δt = 20, $V_{\max} = \beta$ = 10, and $k_{1/2}$ = 0.3. The figure shows that for this machine, long pulse widths and larger values of $V_{\max}$ and β result in less error for all values of n, and that for smaller values of n, there is an optimal value of $k_{1/2}$ near 0.3.

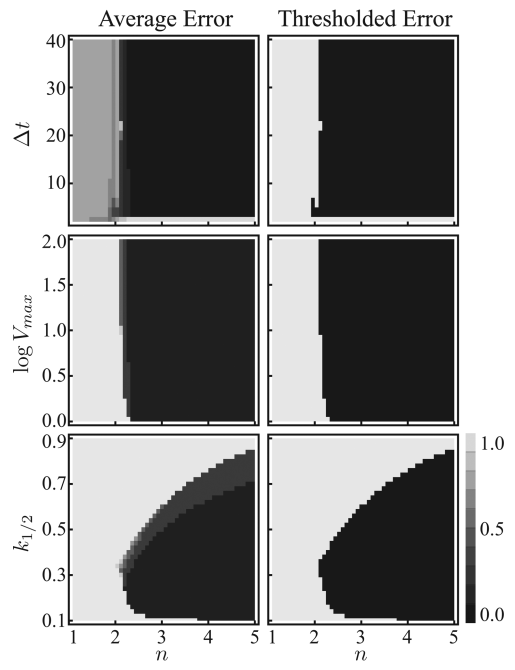


Figure 6. Average error and threshold error for the modulo-two pulse counter illustrated in Figure 5. Error metrics (eqs 66−67) compare the state of the DDE model to the state of the ideal Boolean network model. Error is computed following a control input of five pulses of signaling molecule a. The heat map shows the error for varying Hill coefficient n over a range of values for $k_{1/2}$, log $V_{\max}$, and input pulse width Δt, given τ = 1 and $\beta = V_{\max}$. Nominally τ = 1, Δt = 20, $V_{\max} = \beta$ = 10, and $k_{1/2}$ = 0.3. Zero error means that the DDE model and Boolean network model end in the same state, while an error of one means there is maximal error between models. In this example, increasing pulse width Δt, the rate of production and degradation dynamics $V_{\max}$ and β, or the Hill coefficient n, improves performance.

FSM Framework in a Cellular Information Processing Context. In the previous two examples, finite state machines were shown as complete circuits. Many synthetic and naturally occurring biomolecular circuits have been developed and characterized, and in the larger context of cellular information processing, finite state machines may be used to specify the logical control that wires together a broad array of biomolecular sensor and effector circuits. This strategy may be an intuitive way to design multicellular behaviors. To illustrate this concept, we present a specification for a microcolony edge detection circuit built around finite state machine control logic, and implemented as a GRN.

Figure 7A depicts a specification for a microcolony edge detection circuit. The figure shows the information flow between the control logic modules (labeled "wave generator", "edge detection", and "toggle switch"), and biomolecular sensors and effectors (labeled "stochastic pulse generator", "band pass filter", and "timer"). Each control module is specified by a finite state machine. Here we amend the notion of an FSM $A = (Q, \Sigma, \delta, q_0, F)$ to include a finite set of output symbols Λ and a multivalued output function,

$$\gamma: Q \times \Sigma \to \Lambda \tag{68}$$

On the arrival of each input symbol σ, when the FSM is in state $q \in Q$, in addition to following transition $\delta(q, \sigma) \to q'$ the machine emits output symbol $\lambda = \gamma(\sigma,q)$ with $q' \in Q$ and $\lambda \in \Lambda$. This is similar to the operation of a type of a finite state machine called a *Mealy machine*.[49] As with input symbols, the biomolecular realization of an output symbol is the upregulation of a transcription factor, or enzymatic production of an inducer or signaling molecule that may be exported from the cell. In this way, output symbols are a medium for intercellular or intracellular communication.

For each control module in Figure 7, the state transition map $\delta(q,\sigma) \to q'$ and output map $\gamma(q,\sigma) \to \lambda$ are represented as a set of directed edges labeled σ/λ connecting node q to node q'. If there is no output symbol $\lambda \in \Lambda$ associated with a state transition $\delta(q,\sigma) \to q'$, then the edge is simply labeled σ. For brevity, if multiple input symbols, say σ_1 and $\sigma_2 \in \Sigma$ result in the same transition from state q to q', the input symbols are enumerated as $\sigma_1|\sigma_2$ in the edge label. Similarly, if the output function $\gamma(q,\sigma)$ is multivalued, say $\gamma(q,\sigma) \to \lambda_1$ and $\gamma(q,\sigma) \to \lambda_2$, the output symbols are enumerated as λ_1 and λ_2. For example, in the wave generation module of Figure 7A the edge labeled $\varepsilon^0|k/(\text{emit})$ specifies the transition function,

$$\delta(Q_0^0, \varepsilon^0) \to Q_0^0 \tag{69}$$

$$\delta(Q_0^0, k) \to Q_0^0 \tag{70}$$

and output function γ is defined as,

$$\gamma(Q_0^0, k) \to \text{emit} \tag{71}$$

The biomolecular realization of this specification is that signals ε^0 and k both leave the system in state Q_0^0, however a transition on signal k results in the upregulation of the *emit* signaling molecule.

In operation, the wave generator, edge detection, and toggle switch control logic modules in Figure 7A act in parallel with a stochastic pulse generator, a concentration band-pass filter, and a timer through named communication channels. We consider the *emit* output symbol to be an intercellular communication channel, and all other output symbols are intracellular. In a

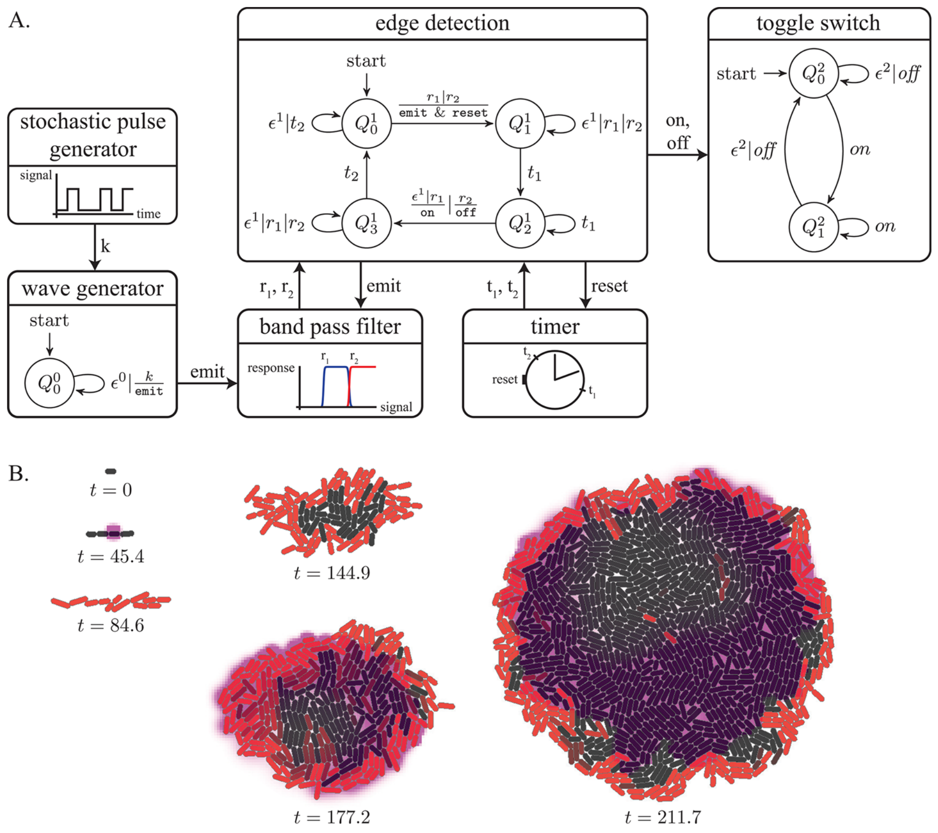


Figure 7. Finite state machine specification and snapshots from a Gro simulation of the bacterial microcolony edge detection circuit. (A) The edge detection circuit consists of three asynchronous and parallel finite state machines and three sensor/effector modules based on existing biomolecular circuits. The stochastic pulse generator is a source of genetic noise, the band-pass filter responds to middle and high concentrations of a diffusable *emit* signal, and the timer emits an intercellular signal at times t_1 and t_2 after receiving a pulse of the *reset* signal. (B) A homogeneous microcolony grows from a single cell. As the microcolony grows, cells stochastically begin a "wave". Cells relay the wave and measure the local concentration of the diffusable *emit* signal after a short refractory period, to determine whether a cell is in the middle or on the edge of a microcolony. Cells on the edge move to a RFP producing state, while cells in the middle relax to a non-RFP producing state.

biomolecular realization, the *emit* symbol can be implemented as a diffusable signaling molecule that is exported from the cell. The intuition behind this design is that at a stochastic rate, any cell in the microcolony can emit a pulse of a diffusable signaling molecule that is sensed and relayed by neighboring cells. By sensing the local concentration of signaling molecules a short time after relaying a wave, a cell can determine whether or not it is on the edge of a microcolony. Cells that detect a high local concentration of signaling molecules have many close neighbors that relayed a wave recently, and thus not on the edge of the microcolony. Alternatively, cells that detect a relatively low concentration of signaling molecules have fewer neighbors, indicating that the cell is on the edge of the microcolony. When a cell detects that is it on the edge of a microcolony, it goes into a RFP producing state. When the cell no longer senses that it is on the edge of a microcolony, it stops production of RFP. This high-level specification was implemented in the 2D simulation environment Gro, robustly producing the red ring colony phenotype illustrated in Figure 7. More detailed discussion of the specification and *Gro* implementation are available in the Supporting Information.

The high level specification in Figure 7 can be refined by replacing finite state machines with gene regulatory networks, and specifying sensors and effectors as input/output modules. Figure 8 illustrates how the GRN implementations of the logical control modules are interconnected with sensors and effectors. The wave generator, edge detection, and toggle switch FSMs are modular components that realized as three distinct GRNs. The stochastic pulse generator, band-pass filter, and timer sensors and effectors are biomolecular modules that take a signaling molecule or transcription factor as input, and produce a signaling molecule or transcription factor as output. Transition genes in the FSMs are used as input to sensors and effectors, and the output of sensors and effectors are then wired to the production of signals in the GRN realizations of the FSMs.

Discussion. We showed that any finite state machine M can be implemented as a gene regulatory network $g(M)$ made entirely of nominally "on" repressing transcription factors. We presented this result in the context of a Boolean network model of gene regulatory networks, where finite state machines are computationally equivalent to gene regulatory networks. Furthermore, we described a construction for arbitrary finite state machines with a set of simple biomolecular parts. We presented a delay differential equation model for the biomolecular construction, and showed through example how the DDE model can be compared to the ideal Boolean network model using a metric induced by the L_∞ norm. Additionally, we showed that the behavior of the DDE model is close to the behavior of the Boolean network model over a range of physically relevant parameters. Finally, we applied our frame-work to a bacterial microcolony edge detection example, using a Gro simulation to show that our approach can be used in a more general cellular information processing context to

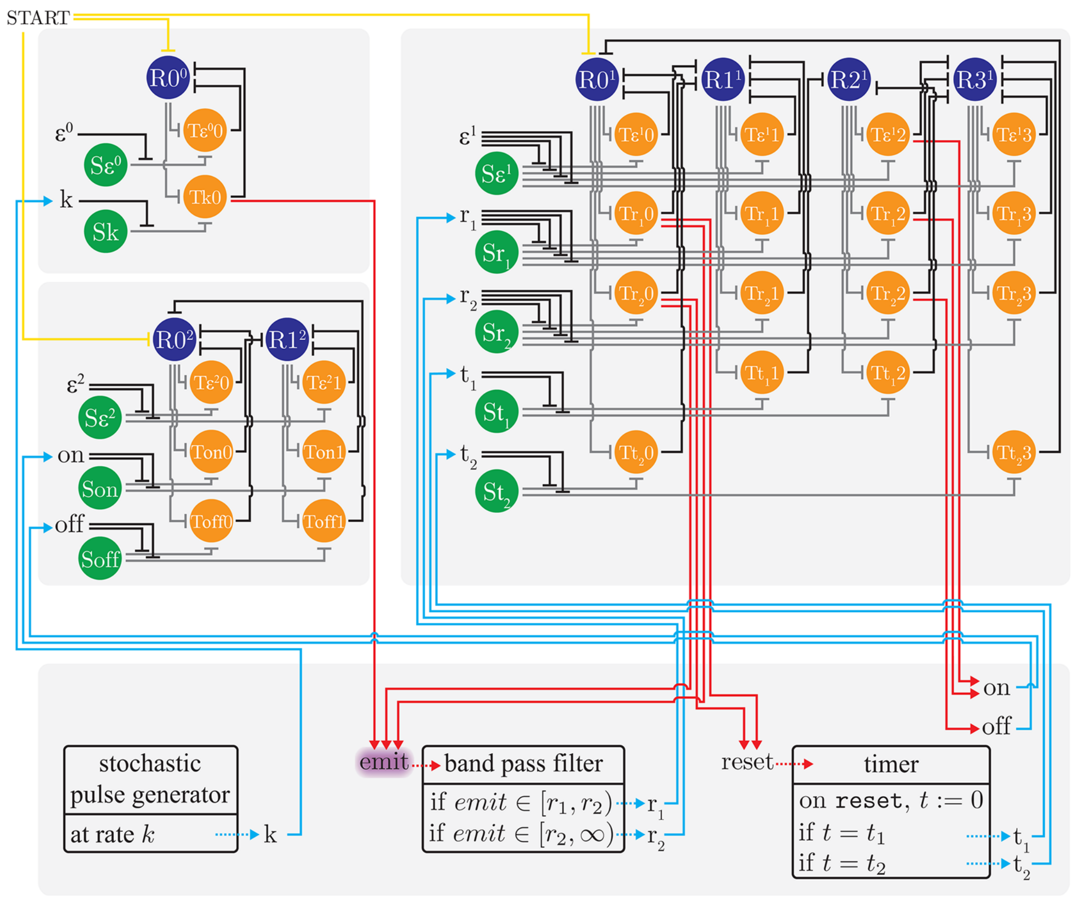


Figure 8. Gene regulatory network realization of the bacterial microcolony edge detection circuit depicted in Figure 7. Finite state machine modules are separated by gray modules. The upper left module encodes the wave generator, the module below it encodes the toggle switch, and the module in the upper right encodes the edge detection FSM. The bottom module contains specifications for the unrealized sensors and effectors. Sensor and effector specifications consist of an optional input signal, output signal, and a description of the behavior of the module. Red lines depict how genes in the finite state machines are wired to sensors, and blue lines depict how sensors are wired to signals. The purple cloud around *emit* indicates that *emit* is an external diffusable cell–cell signaling molecule; all other signals are considered internal.

implement asynchronous logical control interfacing with a set of biomolecular sensors and actuators.

Our results may have implications to understanding the biology of cells, possibly suggesting a new way to relate cell state to observed patterns of gene expression, and more generally, hinting at the computational power and limitations of cells. Assuming gene expression dynamics are approximately binary, a single cell can recognize any sequence of molecular inputs that can be represented as a regular expression; however, no arrangement of a gene regulatory network can enable a single cell to recognize more complex sets of sequences such as those specified by context-free or recursively enumerable languages. This means, for example, that a single cell can be programmed to compute the sum of two integers represented by signal pulses; however, a single cell could not, without some extra internal memory of variable size, compute the product of two arbitrary integers, or recognize when an arbitrary sequence consists of an equal number of *a* and *b* pulses. This observation suggests that multicellular organisms may have evolved precisely because doing so enables a more powerful form of computation. Indeed, given single cells that implement finite state machines, it is a small theoretical step to arrive at multicellular systems that are capable of more complex computation. L-systems, for example, are a cell-based model for plant development in which cells have finite state that when taken together is computationally equivalent to pushdown automata.[50] Furthermore, a growing, dividing, linear arrangement of cells such as cyanobacteria could hypothetically implement a Turing tape machine. In future work we will more carefully explore the computational power and limitations of multicellular systems in more detail.

Our gene regulatory network and biomolecular constructions were optimized for clarity and simplicity of parts and not for performance. Other gene regulatory network topologies and biomolecular implementations may perform more robustly, consist of fewer biological parts, or be otherwise better suited for a particular task. For example, the CRISPR system might be used to implement state and transition transcription factors,[32,40] or a system utilizing recombinase and genome editing may result in greater state stability.[51] Incorporating new parts, such as activators, analog sensors, or molecular insulation devices,[52,53] could also improve performance. Additionally, many different finite state machines may encode the same control logic or recognize the same language over input symbols. It is certainly the case that given a desired high level behavior, some finite state machines result in a more robust biomolecular implementation that others. Our metric for comparing the behavior of a continuous time and continuous space system to an ideal Boolean network model can be used to explore and optimize over the space of possible machines. More detailed analysis of a differential equation model could be done, but the issues addressed in such an analysis really depend on the actual implementation.

METHODS

All Boolean network and DDE simulations are performed in Mathematica Version 8.0,[54] with numerical solver NDSolve. The microcolony edge detection example was simulated in Gro, Version beta.4.[37] Mathematica and Gro files are available upon request.

ASSOCIATED CONTENT

Supporting Information

Proofs for Theorem 1 and Corollary 1 and a detailed description of the operation of the microcolony edge detection circuit. This material is available free of charge *via* the Internet at http://pubs.acs.org/.

AUTHOR INFORMATION

Corresponding Author

*E-mail: koishi@uw.edu.

Notes

The authors declare no competing financial interest.

ACKNOWLEDGMENTS

This work was supported in part by National Science Foundation (NSF) grant 1317653. The authors thank David Soloveichik for useful discussion.

REFERENCES

(1) Lawrence, P. A. (1992) *The Making of a Fly: The Genetics of Animal Design*, Blackwell Scientific Publications Ltd., Hoboken, NJ.

(2) Bonasio, R., Tu, S., and Reinberg, D. (2010) Molecular signals of epigenetic states. *Science 330*, 612−6.

(3) Feng, S., Jacobsen, S. E., and Reik, W. (2010) Epigenetic reprogramming in plant and animal development. *Science 330*, 622−7.

(4) Inbar-Feigenberg, M., Choufani, S., Butcher, D. T., Roifman, M., and Weksberg, R. (2013) Basic concepts of epigenetics. *Fertil. Steril. 99*, 607−15.

(5) Hopcroft, J. E.; Motwani, R., and Ullman, J. D. (2007) *Introduction to Automata Theory, Languages, and Computation*, Pearson/Addison Wesley, Boston.

(6) Minsky, M. L. (1967) *Computation: Finite and Infinite Machines*, Prentice Hall, Upper Saddle River, NJ.

(7) Chomsky, N. (1956) Three models for the description of language. *IEEE Trans. Inf. Theory 2*, 113−124.

(8) Moore, E. F. (1956) Gedanken-experiments on sequential machines. *Automata Studies 34*, 129−153.

(9) Rabin, M. O., and Scott, D. (1959) Finite Automata and their decision problems. *IBM J. Res. Dev. 3*, 114−125.

(10) Burks, A. W., and Wang, H. (1957) The logic of Automata−Part I. *J. ACM 4*, 193−218.

(11) Minsky, M. L. (1961) Recursive unsolvability of Post's problem of "tag" and other topics in theory of turing machines. *Ann. Math. 74*, 437−455.

(12) McCulloch, W. S., and Pitts, W. (1943) A logical calculus of the ideas immanent in nervous activity. *Bull.f Math. Biol. 52*, 99−115.

(13) Kleene, S. C. (1951) *Representation of Events in Nerve Nets and Finite Automata*, Rand Corporation, Santa Monica, CA.

(14) Wieland, M., and Fussenegger, M. (2012) Engineering molecular circuits using synthetic biology in mammalian cells. *Annu. Rev. Chem. Biomol. Eng. 3*, 209−34.

(15) Miyamoto, T., Razavi, S., DeRose, R., and Inoue, T. (2012) Synthesizing biomolecule-based Boolean logic gates. *ACS Synth. Biol. 2*, 72−82.

(16) Lohmueller, J. J., Armel, T. Z., and Silver, P. A. (2012) A tunable zinc finger-based framework for Boolean logic computation in mammalian cells. *Nucleic Acids Res. 40*, 5180−5187.

(17) Wang, B., Kitney, R. I., Joly, N., and Buck, M. (2011) Engineering modular and orthogonal genetic logic gates for robust digital-like synthetic biology. *Nat. Commun. 2*, 508.

(18) Gardner, T. S., Cantor, C. R., and Collins, J. J. (2000) Construction of a genetic toggle switch in *Escherichia coli*. *Nature 403*, 339−42.

(19) Becskei, A., Séraphin, B., and Serrano, L. (2001) Positive feedback in eukaryotic gene networks: Cell differentiation by graded to binary response conversion. *EMBO J. 20*, 2528−35.

(20) Hasty, J., McMillen, D., and Collins, J. J. (2002) Engineered gene circuits. *Nature 420*, 224−230.

(21) Bonnet, J., Subsoontorn, P., and Endy, D. (2012) Rewritable digital data storage in live cells *via* engineered control of recombination directionality. *Proc. Natl. Acad. Sci. U.S.A. 109*, 8884−9.

(22) Basu, S., Gerchman, Y., Collins, C. H., Arnold, F. H., and Weiss, R. (2005) A synthetic multicellular system for programmed pattern formation. *Nature 434*, 1130−4.

(23) Ellis, T., Wang, X., and Collins, J. J. (2009) Diversity-based, model-guided construction of synthetic gene networks with predicted functions. *Nat. Biotechnol. 27*, 465−71.

(24) Siuti, P., Yazbek, J., and Lu, T. K. (2013) Synthetic circuits integrating logic and memory in living cells. *Nat. Biotechnol. 31*, 448−52.

(25) Danino, T., Mondragón-Palomino, O., Tsimring, L., and Hasty, J. (2010) A synchronized quorum of genetic clocks. *Nature 463*, 326−30.

(26) You, L., Cox, R. S., Weiss, R., and Arnold, F. H. (2004) Programmed population control by cell−cell communication and regulated killing. *Nature 428*, 868−71.

(27) Tabor, J. J., Salis, H. M., Simpson, Z. B., Chevalier, A. A., Levskaya, A., Marcotte, E. M., Voigt, C. A., and Ellington, A. D. (2009) A synthetic genetic edge detection program. *Cell 137*, 1272−81.

(28) Liu, C., Fu, X., Liu, L., Ren, X., Chau, C. K. L., Li, S., Xiang, L., Zeng, H., Chen, G., Tang, L.-H. H., Lenz, P., Cui, X., Huang, W., Hwa, T., and Huang, J.-D. D. (2011) Sequential establishment of stripe patterns in an expanding cell population. *Science 334*, 238−41.

(29) Tamsir, A., Tabor, J. J., and Voigt, C. A. (2011) Robust multicellular computing using genetically encoded NOR gates and chemical 'wires'. *Nature 469*, 212−5.

(30) Regot, S., Macia, J., Conde, N., Furukawa, K., Kjellén, J., Peeters, T., Hohmann, S., de Nadal, E., Posas, F., and Solé, R. (2011) Distributed biological computation with multicellular engineered networks. *Nature 469*, 207−11.

(31) Lu, T. K., Khalil, A. S., and Collins, J. J. (2009) Next-generation synthetic gene networks. *Nat. Biotechnol. 27*, 1139−50.

(32) Qi, L. S., Larson, M. H., Gilbert, L. A., Doudna, J. A., Weissman, J. S., Arkin, A. P., and Lim, W. A. (2013) Repurposing CRISPR as an RNA-guided platform for sequence-specific control of gene expression. *Cell 152*, 1173−83.

(33) Beerli, R. R., and Barbas, C. F. (2002) Engineering polydactyl zinc-finger transcription factors. *Nat. Biotechnol. 20*, 135−41.

(34) Moscou, M. J., and Bogdanove, A. J. (2009) A simple cipher governs DNA recognition by TAL effectors. *Science 326*, 1501.

(35) Li, Y., Moore, R., Guinn, M., and Bleris, L. (2012) Transcription activator-like effector hybrids for conditional control and rewiring of chromosomal transgene expression. *Sci. Rep. 2*, 897.

(36) Khalil, A. S., Lu, T. K., Bashor, C. J., Ramirez, C. L., Pyenson, N. C., Joung, J. K., and Collins, J. J. (2012) A synthetic biology framework for programming eukaryotic transcription functions. *Cell 150*, 647−58.

(37) Jang, S. S., Oishi, K. T., Egbert, R. G., and Klavins, E. (2012) Specification and simulation of synthetic multicelled behaviors. *ACS Synth. Biol. 1*, 365−374.

(38) Garg, A., Lohmueller, J. J., Silver, P. A., and Armel, T. Z. (2012) Engineering synthetic TAL effectors with orthogonal target sites. *Nucleic Acids Res. 40*, 7584−7595.

(39) Havens, K. A., Guseman, J. M., Jang, S. S., Pierre-Jerome, E., Bolten, N., Klavins, E., and Nemhauser, J. L. (2012) A synthetic approach reveals extensive tunability of auxin signaling. *Plant Physiol. 160*, 135−42.
(40) Gilbert, L. A., Larson, M. H., Morsut, L., Liu, Z., Brar, G. A., Torres, S. E., Stern-Ginossar, N., Brandman, O., Whitehead, E. H., Doudna, J. A., Lim, W. A., Weissman, J. S., and Qi, L. S. (2013) CRISPR-mediated modular RNA-guided regulation of transcription in eukaryotes. *Cell 154*, 442−51.
(41) Kauffman, S. A. (1969) Metabolic stability and epigenesis in randomly constructed genetic nets. *J. Theor. Biol. 22*, 437−67.
(42) Thomas, R. (1973) Boolean formalization of genetic control circuits. *J. Theor. Biol. 42*, 563−85.
(43) Strogatz, S. H. (2001) Exploring complex networks. *Nature 410*, 268−276.
(44) de Jong, H. (2002) Modeling and simulation of genetic regulatory systems: A literature review. *J. Comput. Biol. 9*, 67−103.
(45) Shen-Orr, S. S., Milo, R., Mangan, S., and Alon, U. (2002) Network motifs in the transcriptional regulation network of *Escherichia coli*. *Nat. Genet. 31*, 64−8.
(46) Bornholdt, S. (2008) Boolean network models of cellular regulation: Prospects and limitations. *J. R. Soc. Interface 5* (Suppl 1), S85−94.
(47) Sadot, A., Sarbu, S., Kesseli, J., Amir-Kroll, H., Zhang, W., Nykter, M., and Shmulevich, I. (2013) Information-theoretic analysis of the dynamics of an executable biological model. *PLoS One 8*, e59303.
(48) Weiss, J. N. (1997) The Hill equation revisited: Uses and misuses. *FASEB J. 11*, 835−41.
(49) Mealy, G. H. (1955) Method for synthesizing sequential circuits. *Bell Syst. Tech. J. 34*, 1045−1079.
(50) Prusinkiewicz, P., and Lindenmayer, A. (1996) *The Algorithmic Beauty of Plants*, Springer, New York.
(51) Friedland, A. E., Lu, T. K., Wang, X., Shi, D., Church, G., and Collins, J. J. (2009) Synthetic gene networks that count. *Science 324*, 1199−202.
(52) Daniel, R., Rubens, J. R., Sarpeshkar, R., and Lu, T. K. (2013) Synthetic analog computation in living cells. *Nature 497*, 619−23.
(53) Del Vecchio, D., Ninfa, A. J., and Sontag, E. D. (2008) Modular cell biology: Retroactivity and insulation. *Mol. Syst. Biol. 4*, 161.
(54) Wolfram Research, Inc. (2010) *Mathematica*, Version 8.0, Wolfram Research, Inc., Champaign, IL.