Executable cell biology

Jasmin Fisher[1,2] & Thomas A Henzinger[2,3]

Computational modeling of biological systems is becoming increasingly important in efforts to better understand complex biological behaviors. In this review, we distinguish between two types of biological models—mathematical and computational—which differ in their representations of biological phenomena. We call the approach of constructing computational models of biological systems 'executable biology', as it focuses on the design of executable computer algorithms that mimic biological phenomena. We survey the main modeling efforts in this direction, emphasize the applicability and benefits of executable models in biological research and highlight some of the challenges that executable biology poses for biology and computer science. We claim that for executable biology to reach its full potential as a mainstream biological technique, formal and algorithmic approaches must be integrated into biological research. This will drive biology toward a more precise engineering discipline.

Over the past decade, biological research has reached a point where the accumulated data exceed the human capacity to analyze it. The vast information generated by DNA microarrays, genome sequencers and other large-scale technologies requires computer power for storage, searching and integration into a coherent picture. Systems biology, which combines biology, chemistry, physics, mathematics, electrical engineering and computer science, among other disciplines, aims to integrate the data concerning individual genes and proteins and to investigate the behavior and relationships of various elements in a biological system to explain how it functions[1–3].

At the core of systems biology lies the construction of models describing biological systems. Over the years, biologists have used diagrammatic models to summarize a mechanistic understanding of a set of observations. Despite the many benefits of such models, as well as their simplicity, they give a rather static picture of cellular processes. The growing need to translate these models into more dynamic forms that can capture time-dependent processes, together with increases in the models' scale and complexity, has prompted biologists to harness computers to build and analyze ever-larger models. The long-term vision is that large-scale models should revolutionize biology and medicine and enable design of new therapies.

We distinguish between two types of models: (i) those that use computer power to analyze mathematical relationships between quantities and (ii) a new variety, resembling a computer program, which is central to an emerging field that we call executable biology. Here, we explain the differences between these two approaches, explore some recent executable biology models and emphasize some challenges facing this new field.

[1]Microsoft Research, Cambridge CB3 0FB, UK. [2]School of Computer and Communication Sciences, EPFL, Lausanne CH-1015, Switzerland. [3]Electrical Engineering & Computer Sciences, University of California at Berkeley, California 94720-1770, USA. Correspondence should be addressed to J.F. (jasmin.fisher@microsoft.com) or T.A.H (tah@epfl.ch).

Published online 7 November 2007; doi:10.1038/nbt1356

Mathematical versus computational models

Mathematical models, such as those based on differential equations, can represent many situations in the natural sciences and engineering. Although they were developed before computation became feasible on a grand scale, they are now profiting from our increasing computational ability.

In contrast, computational models present a recipe—an algorithm—for an abstract execution engine to mimic a design or natural phenomenon. Such models are ideally suited to representing complicated chains of events. They have been used recently to model biochemical processes[4–7], thymocyte development and cell fate determination during *Caenorhabditis elegans* development[8–14].

Mathematical and computational models (**Box 1**) differ in the languages in which they are specified. Whereas the former are specified in mathematics, typically equations, the latter are specified by computer programs, often very high-level code written in a modeling language such as Statecharts[15] or Reactive Modules[16]. Consequently, the two types of models yield different kinds of insights. The differences are exemplified by comparing different modeling approaches to cell fate determination during *C. elegans* vulval development[11,13,17], which concentrate on different aspects and consequently provide different kinds of insights into the same system. In contrast with a mathematical model[17] that predicts rates of intercellular reactions and suggests a time frame in which cell fate determination is established, the computational models[11,13] predict the timing and order of signaling events as well as new modes of interaction between the epidermal growth factor receptor and LIN-12/Notch signaling pathways.

Mathematical models can be simulated and possibly solved. The basic entity of a mathematical model is the transfer function, which relates different numerical quantities to each other. A transfer function may be specified, for example, by a differential equation that relates an input to an output quantity. Complex mathematical models are constructed through the composition of transfer functions, yielding a network of interdependent quantities. If the constraints for individual transfer functions are relatively simple (e.g., linear differential equations), then mathematical models are amenable to mathematical

Box 1 Mathematical versus computational models

A computational model is a formal model whose primary semantics is operational; that is, the model prescribes a sequence of steps or instructions that can be executed by an abstract machine, which can be implemented on a real computer. A mathematical model is a formal model whose primary semantics is denotational; that is, the model describes by equations a relationship between quantities and how they change over time. The equations do not determine an algorithm for solving them; in general, there may be many different solution algorithms and often such algorithms compute only approximate solutions.

There is an entire sub-field of computer science that studies the relationships and differences between computational (operational) and mathematical (denotational) views of a system. Whereas for computational models, the computer implementation is by definition a faithful representation of the model, for mathematical models, there is a gap between the meaning of the model and its implementation on a computer. This gap needs to be bridged, for example, by proving that a certain algorithm solves a certain equation with a certain precision. This is not to say that for computational models, the representation gap magically disappears; rather it is shifted and reappears between the biological system and the model. Bridging that gap requires the adequacy of abstractions, not the faithfulness of implementations. Although computational models are further from the biological system and closer to the computer, a good computational model–if one can be found–may explain the mechanisms behind a biological system in more intuitive and more easily analyzable terms than a mathematical model.

analysis. In more complicated cases (e.g., nonlinear or stochastic differential equations) and in very high-dimensional cases (where the number of variables is large), mathematical models require computational simulation to plot changes in quantities of substances over time.

Computational models can be executed. By contrast, the basic entity of computational models is the state machine, which relates different qualitative configurations ('states') to each other. A state machine may be specified by simple computer programs that define how, given certain events, one state is transformed into another. Complex computational models are constructed through the composition of state machines, yielding a reactive system (**Box 2**). The components of such a system represent biological entities, such as cells, which react to events involving neighboring components by state transformations. This is often useful in cell biology, because it requires the modeler to think in terms of 'cause and effect' rather than rates of change.

Such computational models can have a very large number of states, are often highly nonlinear and nondeterministic (**Box 2**) and are generally not amenable to mathematical analysis. Whereas an algorithm must be devised to simulate a mathematical model, a computational model prescribes the steps taken by an abstract machine and is therefore inherently and immediately executable. As the primary semantics of computational models are operational, we use the term execution instead of simulation—hence executable biology. The efficiency with which computers can execute instructions, which exceeds their ability to solve or simulate mathematical equations, makes them ideally suited to the execution of very large computational models.

Quantitative versus qualitative modeling of biology. In biology, mathematical models for many quantitative relationships between variables, such as molecule concentrations and gene activity levels, have been devised to represent cell signaling pathways in a physically and biologically realistic manner and have been shown repeatedly to generate novel and useful hypotheses[18–27]. Such models, however, are difficult to obtain and analyze if the number of interdependent variables grows and if the relationships depend on qualitative events, such as a concentration reaching a threshold value. Computational models offer an effective alternative if precise quantitative relationships are unknown, if they involve many different variables or if they change over time, depending on certain events. Because computational models are qualitative, they do not presuppose a precision absent from the experimental data; because they are nondeterministic or stochastic, they allow many possible outcomes of a chain of events, which is often observed in biological systems.

A significant advantage of qualitative models is that different models can be used to describe the same system at different levels of detail and that the various levels can be related formally. There are several natural levels of abstraction for describing biological systems using computational models. For example, the individual components may represent molecules or, at a less detailed level, they may represent cells. In such models, it may not be necessary to know exactly how a certain process (e.g., protein synthesis) achieves a certain output, provided that the behavior of the process can be defined qualitatively in a robust manner. Hence, computational models can be useful even when not every detail about a system is known.

Computational models can be analyzed by model checking. Computational models can be used for testing and comparing hypotheses. Suppose that we have collected experimental data. A computational model represents a hypothesis about the mechanism that results in the data. An execution of the model can be used to check whether a possible outcome of the mechanism conforms to the data (**Fig. 1**). Owing to nondeterminism or stochastic choices, each repeated execution may yield a different outcome. Therefore it is impossible to check by executing the model whether all possible outcomes conform to the data, or whether the distribution of outcomes conforms to the data. This, however, can be done by a technique called model checking[28], which systematically analyzes all of the infinitely many possible outcomes of a computational model without executing them one by one. Intuitively, this is done by exploring the states and possible state changes of a model, rather than by exploring all possible executions of the model. Model checking is effective, because there are usually many more executions than states. A state that may repeat can give rise to infinitely many possible executions.

If model checking tells us that all possible outcomes of the computational model agree with the experimental data and that all experimental outcomes can be reproduced by the model, then the model represents a mechanism that satisfactorily explains the experimental data. If, on one hand, some of the experimental data cannot be reproduced, then the hypothesis is wrong. In this case, either the model must be improved to produce the additional outcomes that are present in the data, or completely revised. If, on the other hand, some outcomes of the computational model disagree with the experimental data, then the situation is more interesting. In this case, the mechanistic hypothesis represented by the model may be wrong and one may attempt to restrict the model so it does not produce outcomes that are not supported by the data, as recently illustrated by a model of crosstalk between Notch and Wnt signaling[29] and a model of *C. elegans* vulval development[13]. Alternatively, the experimental data may be incom-

Box 2 Glossary of terms

Reactive system. A system that consists of parallel processes, where each process may change state in reaction to another process changing state. Biological systems are highly reactive (e.g., cells constantly send and receive signals and operate under various conditions simultaneously).

Nondeterministic system. A system that may have several possible reactions to the same stimulus. In biological systems, for example, we can observe various patterns of cell fate under the same genotype. Hence, nondeterministic models capture the diverse behavior often observed in biological systems by allowing different choices of execution, without assigning priorities or probabilities to each choice.

Distributed system. A system that consists of a collection of autonomous computers, connected through a network that enables the computers to coordinate their activities and to share the resources of the system, so that users perceive the system as a single, integrated computing facility.

Concurrency. In computer science, a property of systems that consist of many processes running in parallel and sharing common resources.

Tokens in Petri nets. These describe the presence or absence of a condition, a signal, or a resource. In the case of metabolic networks, the number of tokens in a place stands for the number of molecules of that metabolite existing at a given moment. Alternatively, tokens may correspond to a predefined unit measuring the amount of a substance, such as mole and millimole.

Visual languages. Languages that allow programming with visual expressions (such as diagrams, drawings, animations and icons) as opposed to conventional textual languages that use only textual code. Visual programming environments provide graphical interfaces, which can be manipulated by the user in an interactive way.

Reactive animation. A visual front-end that can be set up to yield interactive animation movies that follow in real time the events taking place during model execution and which can be manipulated and changed during run-time.

Compositional analysis. Analysis through which the properties of a system can be derived from properties of its parts.

The Delta-Notch decision. A signaling process where two equipotent cells that initially express equal amounts of Notch and its ligand Delta gradually express either Notch or Delta. The Notch-expressing cell receives activation signals from the Delta-presenting neighboring cell, resulting in these two cells adopting very different cell fates.

Very-large-scale integration (VLSI). The process of creating integrated circuits by combining many thousands of transistor-based circuits into a single chip.

plete and not exhibit some possible observations that would become evident if more data were collected. In this case, model checking can offer suggestions for additional, targeted experiments that would either confirm or invalidate the mechanistic hypothesis represented by the computational model (**Fig. 1**).

Models for executable biology

Here we summarize several research efforts aimed at realizing the executable biology framework. These are explained further in **Box 3**.

Boolean networks for analyzing systems robustness and stability. Boolean networks were first introduced by Kauffman in the early 1970s[30,31] and are the oldest form of executable biology models. Boolean networks approximate the dynamics of biological networks by considering each molecule (e.g., gene or protein) in the network as either active (1) or inactive (0); intermediate expression levels are neglected. Thus, the state of the system corresponds to the activation state of each of the molecules that make up the network. A molecule is considered to become active if the sum of its activations is larger than the sum of its inhibitions and inactive if the sum of its activations is smaller than the sum of its inhibitions (**Fig. 2a**). Hence, we obtain a system whose state evolves according to the postulated connections between its molecules (**Fig. 2b**). Despite this clearly simplified view of biological networks, several examples from models of genetic regulatory networks show that Boolean approaches give meaningful biological information[32–34].

Boolean networks have proved useful in analyzing system dynamics and reasoning about the stability and robustness of biological systems[34,35]. The possible states of a Boolean network are drawn as nodes of a graph and possible state changes are drawn as edges. Loops in the graph are used to deduce which are the stable states of the system. The number of loops can be used to reason about system robustness (**Fig. 2c**). The strong simplifying assumptions on the structure and dynamics of a genetic regulatory system enable the efficient analysis of large regulatory networks. Boolean networks were also among the first formalisms for which algorithms were devised to infer genetic interactions from gene expression data[36–41].

From a computational point of view, it is difficult to compose larger models from smaller building blocks using Boolean networks. Hierarchical structuring, which makes the design and analysis of models simpler, is not possible in Boolean networks. Recently, Schaub *et al.* introduced an extension to Boolean networks, called 'qualitative networks', in an attempt to support hierarchical structuring[29].

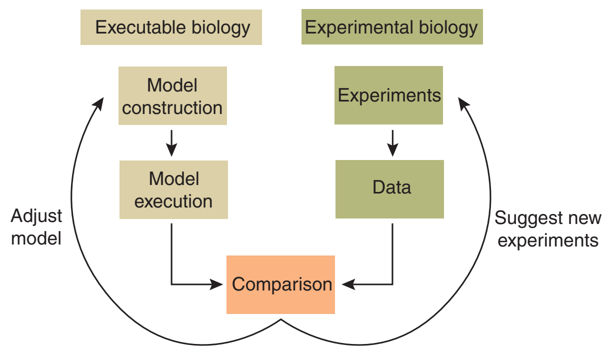


Figure 1 The methodology of executable biology. Our view of executable biology is an interplay between collecting data in experiments (experimental biology) and constructing executable models that capture some mechanistic understanding of how the systems under study work. By executing the models under various conditions that correspond to the experiments and by comparing the outcomes to the experimental data, one can identify discrepancies between hypothetical mechanisms and the experimental observations. These differences can be used to suggest new hypotheses, which serve to adjust the model and need to be validated experimentally, or new experiments, which can confirm or falsify modeling hypotheses.

Box 3 Computational models

Boolean networks. These models are *computational*, because from a given activation state of all molecules, they prescribe which molecules become active in the next step. The execution of a Boolean network thus illuminates the causal and temporal relationships between the activation of different molecules. The main drawback of Boolean networks is that they do not support the composition of larger models from smaller ones. To allow integration of several interacting mechanisms, a model needs to offer a so-called composition operation. Interacting state machines and process calculi support such a composition operation.

Petri nets. These models are computational, because from a given assignment of tokens to places, they prescribe which tokens can change place in the next step. Petri nets are more general than Boolean networks, because their execution semantics allows for true concurrency: several tokens may change place independently in the same step. Also, whereas Boolean networks are deterministic (that is, the outcome of execution is unique), Petri nets may be nondeterministic (execution may have many different outcomes; e.g. when there are multiple options to move tokens), or stochastic (that is, there is a probability distribution of possible outcomes), or both (when there are several different probability distributions of possible outcomes). Like Boolean networks, Petri nets do not support the composition of several networks.

Interacting state machines. Several languages are available to specify these models, for example, the language of Reactive Modules. They are computational because from given states of all interacting machines, the model prescribes the next states of the machines. The interaction may be synchronous, when some machines change state simultaneously because of causal dependencies, or asynchronous, when some machines change state independently, in any order. Asynchronous interaction gives rise to nondeterminism, because different orders may give different results. Like Petri nets, interacting state machines may be nondeterministic, stochastic, or both. Unlike Boolean networks and Petri nets, interacting state machines are compositional, because several machines can be put together and will interact with each other. State machines can also be equipped with a hierarchical structure, as in Statecharts. A hierarchical machine may change state at a microlevel and several microsteps together make up a macro-step, which is a single state change of a higher-level machine providing a more abstract (that is, less detailed) view of the system.

Process calculi. Like interacting state machines, these models are computational and compositional. They may be nondeterministic, stochastic, or both. The main difference between interacting state machines and process calculi is that in the former case, the most basic notion is that of a state and the model prescribes how the state changes, whereas in the latter case, the most basic notion is that of an event and the model prescribes how events either cause or are independent of other events. Although an event can be represented by a state change and a state by a history of events, the two views give rise to different styles of modeling.

Hybrid models. The discrete part of these models is computational and the continuous part, mathematical. Executing the continuous change of variables, as described by differential equations, requires an algorithm that is independent of the model and often gives only approximate results. For example, the possibility of a discrete state transition may be missed if that possibility depends on the exact value of a continuous variable.

Petri nets for analyzing biological networks. Petri nets represent a well-established technique in computer science for modeling distributed systems (**Box 2**). The model stresses concurrency (**Box 2**), which is important when modeling biological systems. A Petri net is a graph with two types of nodes: places, which represent the resources of the system, and transitions, which correspond to events that can change the state of the resources. The edges of the graph connect places to transitions and transitions to places (**Fig. 3a**). The state of the system is represented by places holding so-called tokens (**Box 2**); one place may hold multiple tokens. Thus, different assignments of tokens to places induce different states of the system. Transitions change the state of the system by moving tokens along edges. In a given state of the system, there may be more than one transition that can move a token, leading to nondeterminism.

Petri nets are well-suited for modeling the concurrent behavior of biochemical networks[42,43] and have been used to represent metabolic pathways[44] and protein synthesis[45,46]. **Figure 3** shows a Petri net model of the biosynthesis of tryptophan in *Escherichia coli*.

Some of the main advantages of Petri nets are that they are visual, have different flavors and can be designed and analyzed by a range of tools. The simple type of Petri nets described above subsume Boolean networks: a place represents a molecule and a token at that place represents the active state of the molecule. Choosing to model Boolean networks using Petri nets has the added advantage of ready-made visual design and analysis tools. Recently an automatic translation of Boolean networks to Petri nets has been suggested[47]. However, much like Boolean networks, Petri nets do no support hierarchical structuring, which makes them difficult to use for large-scale models.

More complex flavors of Petri nets provide additional possibilities in modeling. For example, in colored Petri nets, different-colored tokens induce multiple possible values for each place, allowing different activation levels to be assigned to resources. Colored Petri nets have been used to analyze metabolic pathways[48]. Stochastic Petri nets add probabilities to the different choices of the transitions and have been used to analyze signaling pathways[49–51], where the number of molecules of a given type is represented by the color of a place and probabilities represent reaction rates. The Pathalyzer is a software tool that builds on the availability of Petri nets analysis and design tools to standardize and collect information about signal transduction pathways[52]. It uses analysis techniques for Petri nets to answer queries such as "what could cause the activation of a certain substance?", or "is it possible that a certain substance will reach activation in the absence of a different substance?"[52].

Interacting state machine models for biological mechanisms. State machine models define the behavior of objects over time, based on the various states that an object can be in over its lifetime. In other words, states are abstract situations in an object's life cycle. Interacting state machines can specify causal relationships between state changes in different machines. These models describe both how objects communicate and collaborate as well as how they behave under different circumstances. Usually, the state of an object is determined by the states of its parts. For example, the state of a cell is determined by the states of various genes and proteins, each having its own reaction to the presence or absence of some other molecules. Changes in the state of the cell are determined by the interdependent state changes of all parts.

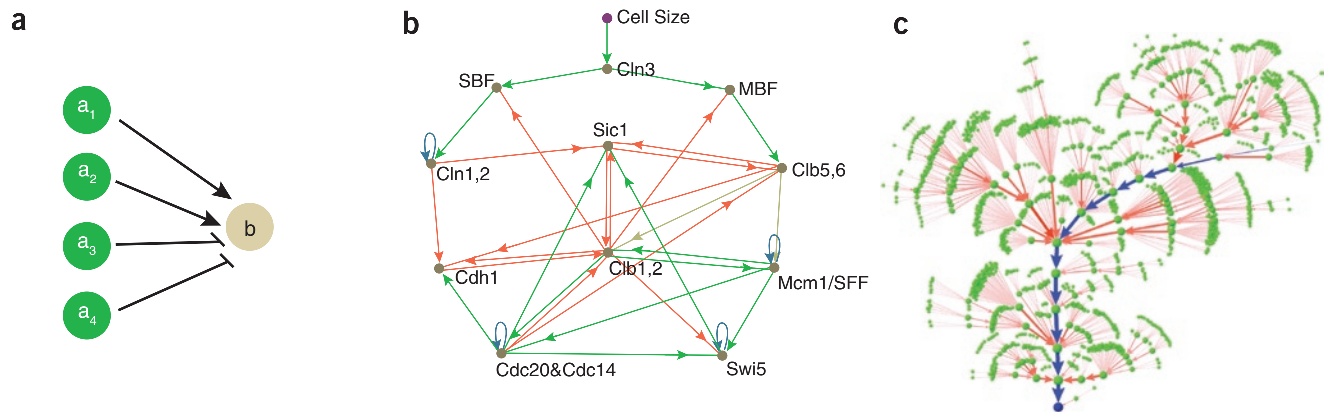


Figure 2 Boolean networks. (**a**) An isolated part of a Boolean network representing the behavior of one substance. Arrows indicates activation and bars denote inhibition. The next value of the substance is determined by the sum of activations minus the sum of inhibitions. In this example, if we denote the values of a_1, a_2, a_3 and a_4 at time *t* by a_1, a_2, a_3 and a_4, then the value of substance b at time *t* + 1 will be 1 if $a_1 + a_2 - (a_3 + a_4)$ is positive and 0 otherwise. Sometimes arrows are given strengths and then we take the sum of strengths of activation arrows whose source is active (that is, set to 1) minus the sum of strengths of inhibition arrows whose source is active. (**b**) Simplified cell-cycle network of the budding yeast. (**c**) Analysis of the yeast cell-cycle network using Boolean networks. Each dot represents a state of the proteins in the system, where each of the proteins is either active or inactive. Each arrow represents a transition from one state to another. The blue transitions correspond to the cell-cycle sequence. Starting from any point in the graph, in order to avoid reaching the stable state at the bottom of the diagram, one would have to continuously perturb the system. Hence, the normal behavior converges fast to the stable state at the bottom of the diagram, corresponding to the G1 stationary state in which the cell awaits a signal that will start another round of division. This demonstrates that the yeast cell-cycle regulatory network is stable and robust for its function. Figures reproduced with permission from ref. 34.

A hierarchical structure allows one to view a system at different levels of detail (e.g., whole organism, tissues, cells; **Fig. 4a**). Models of this kind have been used to model T-cell activation and differentiation[8,9], as well as *C. elegans* development[10,11,13,14].

Interacting state machine models are particularly suitable for describing mechanistic models of biological systems that are well understood qualitatively. Such models do not require quantitative data relating to the number of molecules and reaction rates. They allow the creation of abstract high-level models and the application of strong analysis tools such as model checking. The possibility of hierarchical structuring is extremely useful in cases where the behavior is distributed over many cells and where multiple copies of the same process are executed in parallel.

There are many different languages to express interacting state machine models. Using the visual language (**Box 2**) of Statecharts[15], Kam *et al.* developed a model that described the various stages in the life span of a T-cell and the transitions between these stages[8]. The initial T-cell model was followed by a more extensive animated model of T-cell differentiation in the thymus[9]. A major advantage of Statecharts compared to other state-based formalisms, such as Reactive Modules[16], is the fact that this language is visual. The user can draw states and state changes and the tool automatically creates an executable model, enabling relatively easy and intuitive programming even for nonspecialists. Efroni *et al.* used reactive animation (**Box 2**)[9,53], where a reactive system drives the display of animation software to visualize the model. These studies were followed by ongoing efforts to model *C. elegans* development[10,11,13,14], which used Statecharts and a visual language called Live Sequence Charts[54] and more recently a language called Reactive Modules[16] that supports compositional analysis techniques (**Box 2**).

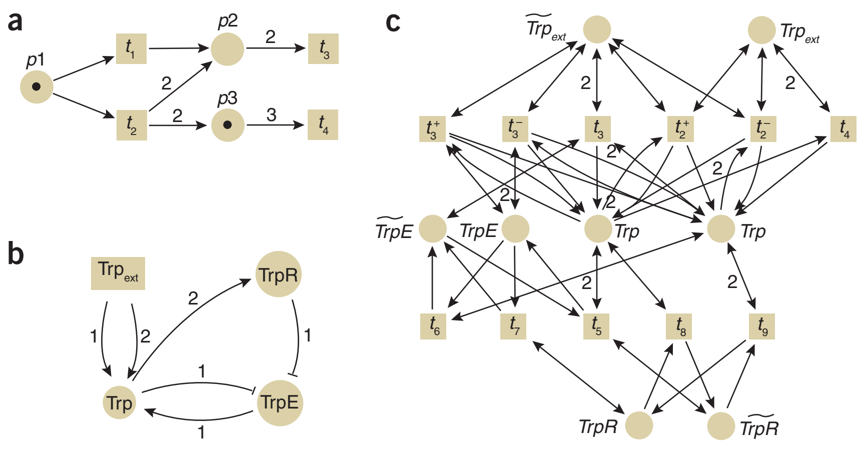


Figure 3 Petri nets. (**a**) A simple, standard Petri net. The circles denote places, whereas the boxes denote transitions. The distribution of tokens (black dots) in the places at a given time defines a marking. Transitions change the marking by removing a token from each incoming arrow and adding a token to each outgoing arrow. (**b**) Simplified logical regulatory graph for the biosynthesis of tryptophan in *E. coli*. Each node of the regulatory graph represents an active component: tryptophan (Trp), the active enzyme (TrpE) and the active repressor (TrpR). The node marked by a rectangle accounts for the import of Trp from external medium. All nodes are binary (that is, can take the value 0 or 1), except Trp, which is represented by a ternary variable (taking the values 0, 1, 2). Arrows represent activation and bars denote inhibition. (**c**) Petri net of the Trp regulatory network. Each of the four components of **b** is represented by two complementary places and all the different situations that lead to a change of the state of the system are modeled by one of the nine transitions (t_1–t_9). Figures reproduced with permission from ref. 46.

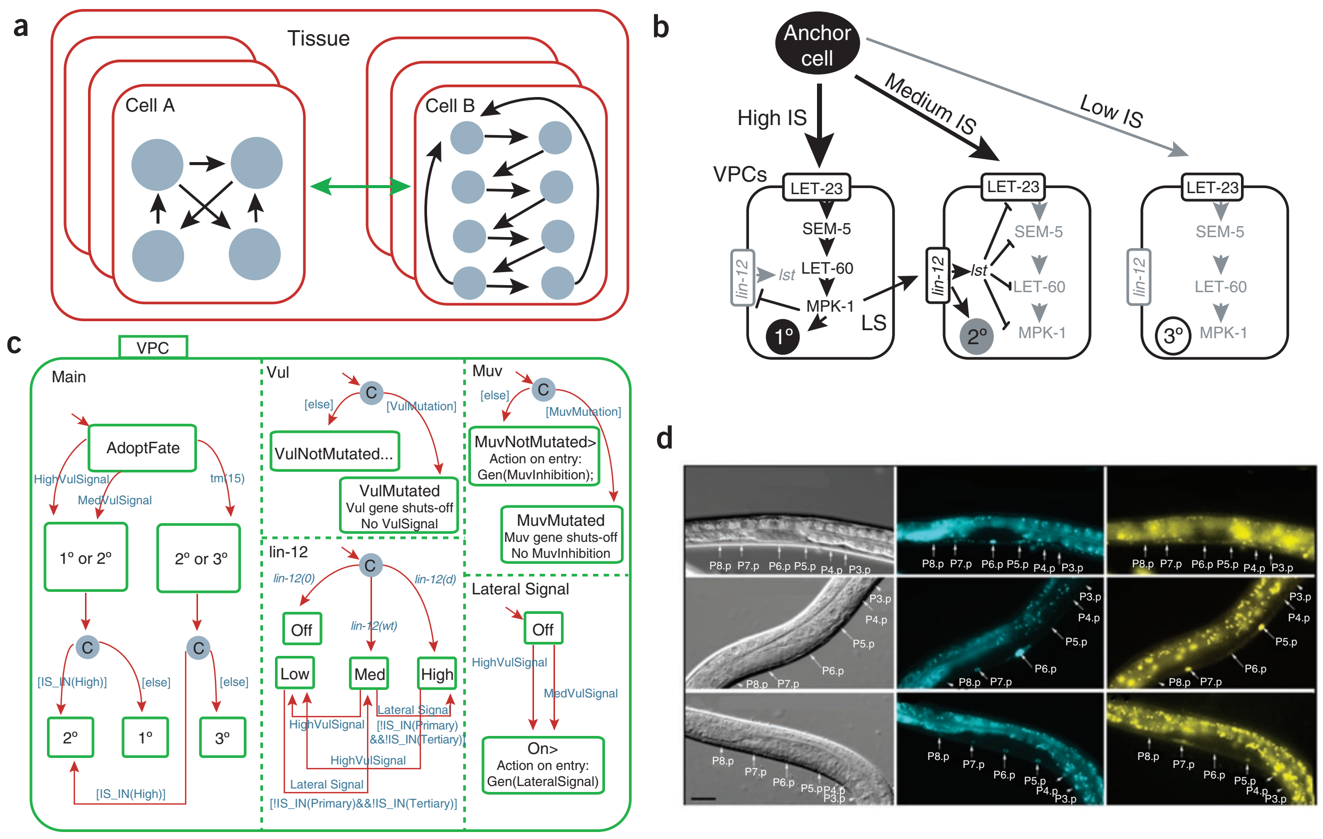


Figure 4 Interacting state machine models. (**a**) Hierarchical diagram representing a tissue comprising three cells of type A and three cells of type B. All cells of the same type work according to the same program. (**b**) Diagrammatic mechanistic model for the signaling events underlying vulval precursor cell (VPC) fate specification. IS, inductive signal; LS, lateral signal; cell fates: 1°, primary; 2°, secondary; 3°, tertiary. (**c**) Statecharts model of a vulval precursor cell. Rectangles represent states and arrows represent transitions. A short arrow exiting a small circle marks the initial state of the object. The circled C denotes a condition between two states. A transition from a condition is taken if the guard is true. Areas separated by dashed lines are concurrent components; that is, the cell is present in all these components simultaneously. Figure reproduced with permission from ref. 11. (**d**) Experimental validation of the loss of sequential activation in *lin-15* mutants, as predicted by the computational model. The pictures visualize cell fate specification in *C. elegans* using blue and yellow fluorescent proteins (EGL-17::CFP and LIP-1::YFP) expressed during activation of the inductive and lateral signaling pathways, respectively. The upper and middle rows show examples of wild-type animals at mid and late L2 stage, expressing the EGL-17 marker in P6.p and the LIP-1 marker in P5.p and P7.p, respectively. The lower row shows examples of a *lin-15* mutant at the late L2 stage showing simultaneous expression of both markers in P5.p and P7.p. Figure reproduced from ref. 13.

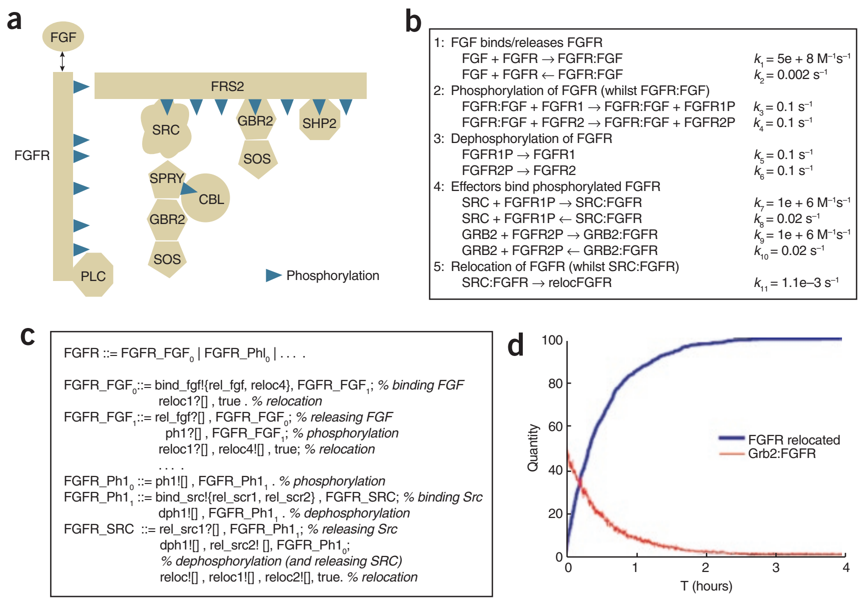
Fisher *et al.*[11] created a formal dynamic model of vulval fate specification based on the proposed mechanistic model of Sternberg and Horvitz[55]. This work revealed that state-based mechanistic modeling is well-suited to developmental genetics and can provide new insights into the temporal aspects of cell fate specification during *C. elegans* vulval development. Subsequent work[13] was based on the more sophisticated current understanding of vulval fate specification (**Fig. 4b**). Model checking allowed us to test the consistency of the current conceptual model for vulval precursor cell fate specification with an extensive set of observed behaviors and experimental perturbations of the vulval system. The analysis of this model predicted new genetic interactions between the signaling pathways involved in the patterning process, together with temporal constraints that may further elucidate the mechanisms underlying precise pattern formation during animal development. These predictions were validated experimentally (**Fig. 4**).

Process calculi for executing molecular processes. A different approach stresses the importance of interactions between molecules as the driving force for biological processes. As opposed to previous approaches where execution results in a sequence of states, here execution is defined through a sequence of events and their causal dependencies. This approach uses process calculi—languages that have been developed to model networks of communicating processes[56]. In this context, a process is a state machine for which some state changes can be observed as events. Events provide communication between processes.

To model biological behaviors, a process is associated with a molecule and many copies of the same process run in parallel to simulate the existence of many molecules. Communication between processes is used to model interactions between molecules. For example, the activation of a certain molecule by the energy released from ATP hydrolysis can be modeled by two processes and a communication event between them as follows: a process associated with ATP proceeds from the 'ATP-state' to the 'ADP-state', a process associated with the specific molecule proceeds from the 'inactive-state' to the 'active-state', and the two state changes are simultaneous because of a communication event. The inactivation of the same molecule can then be modeled by an independent state change.

This modeling approach is applicable to molecular interactions that occur stochastically. It can be used for the detailed analysis of the stochastic behavior of a molecular interaction network using model

Figure 5 Pi calculus. (**a**) Possible molecular interactions in the fibroblast growth factor (FGF) pathway. Figure reproduced from reference 63. (**b**) Partial summary of reactions between the components presented in the diagram, including reaction rates obtained from the literature. (**c**) A fragment of the stochastic pi-calculus code (in the textual format of BioSPI) relating to FGF receptor (FGFR) and its interactions with FGF and Src. (**d**) The BioSPI system inputs the pi-calculus code and performs simulations using the Gillespie algorithm. The curves show the amount of relocated FGFR and Grb2 bound to FGFR over time, for an average of ten simulations. Figures reproduced from ref. 72.



1: FGF binds/releases FGFR	
FGF + FGFR → FGFR:FGF	$k_1 = 5e + 8\ M^{-1}s^{-1}$
FGF + FGFR ← FGFR:FGF	$k_2 = 0.002\ s^{-1}$
2: Phosphorylation of FGFR (whilst FGFR:FGF)	
FGFR:FGF + FGFR1 → FGFR:FGF + FGFR1P	$k_3 = 0.1\ s^{-1}$
FGFR:FGF + FGFR2 → FGFR:FGF + FGFR2P	$k_4 = 0.1\ s^{-1}$
3: Dephosphorylation of FGFR	
FGFR1P → FGFR1	$k_5 = 0.1\ s^{-1}$
FGFR2P → FGFR2	$k_6 = 0.1\ s^{-1}$
4: Effectors bind phosphorylated FGFR	
SRC + FGFR1P → SRC:FGFR	$k_7 = 1e + 6\ M^{-1}s^{-1}$
SRC + FGFR1P ← SRC:FGFR	$k_8 = 0.02\ s^{-1}$
GRB2 + FGFR2P → GRB2:FGFR	$k_9 = 1e + 6\ M^{-1}s^{-1}$
GRB2 + FGFR2P ← GRB2:FGFR	$k_{10} = 0.02\ s^{-1}$
5: Relocation of FGFR (whilst SRC:FGFR)	
SRC:FGFR → relocFGFR	$k_{11} = 1.1e–3\ s^{-1}$

```
FGFR ::= FGFR_FGF0 | FGFR_PhI0 | . . . .

FGFR_FGF0::= bind_fgf!{rel_fgf, reloc4}, FGFR_FGF1; % binding FGF
             reloc1?[] , true . % relocation
FGFR_FGF1::= rel_fgf?[] , FGFR_FGF0; % releasing FGF
              ph1?[] , FGFR_FGF1; % phosphorylation
             reloc1?[] , reloc4![] , true; % relocation
             . . . .
FGFR_Ph10 ::= ph1![] , FGFR_Ph11 . % phosphorylation
FGFR_Ph11 ::= bind_src!{rel_scr1, rel_scr2} , FGFR_SRC; % binding Src
             dph1![] , FGFR_Ph11 . % dephosphorylation
FGFR_SRC  ::= rel_src1?[] , FGFR_Ph11; % releasing Src
              dph1![] , rel_src2! [], FGFR_Ph10;
              % dephosphorylation (and releasing SRC)
              reloc![] , reloc1![] , reloc2![], true. % relocation
```

checking. Currently, owing to scalability issues, such an analysis can be applied only to relatively small models. However, the information and insights provided by this kind of analysis suggest that it is beneficial to create oversimplified models of large and complex networks.

Initial work along this line used a process calculus called pi-calculus[56] as a modeling language for molecular interactions[5]. These studies included the modeling of the receptor tyrosine kinase and the mitogen-activated protein kinase signal-transduction pathway and the construction of a simulation environment called BioSPI. The stochastic pi-calculus[57] was later used to model a gene regulatory positive-feedback loop[4]. Many other studies have followed this direction, including experiments using the ambient calculus[58] and the brane calculus[59]. The methodology has also been used to model transcription factor activation and glycolysis[60], Raf kinase inhibitory protein inhibition of extracellular signal-regulated kinase[61] and more recently, the mitogen-activated protein kinase cascade (including a comparison with a similar model using differential equations)[62] and the fibroblast growth factor pathway (**Fig. 5**) (and its extensive analysis using stochastic simulation and probabilistic model checking)[63]. A recent review[7] discusses the process calculus approach in depth.

Hybrid models combining mathematical and computational models. Hybrid systems combine in a single framework variables that span discrete and continuous domains[64]. The discrete variables are controlled by discrete state changes that may depend on the values of continuous variables. The changes in continuous variables are governed by differential equations (preferably linear), which depend on discrete states (that is, the combined value of all discrete variables determines the discrete state and the discrete state determines which differential equations are to be used to govern the rates of change

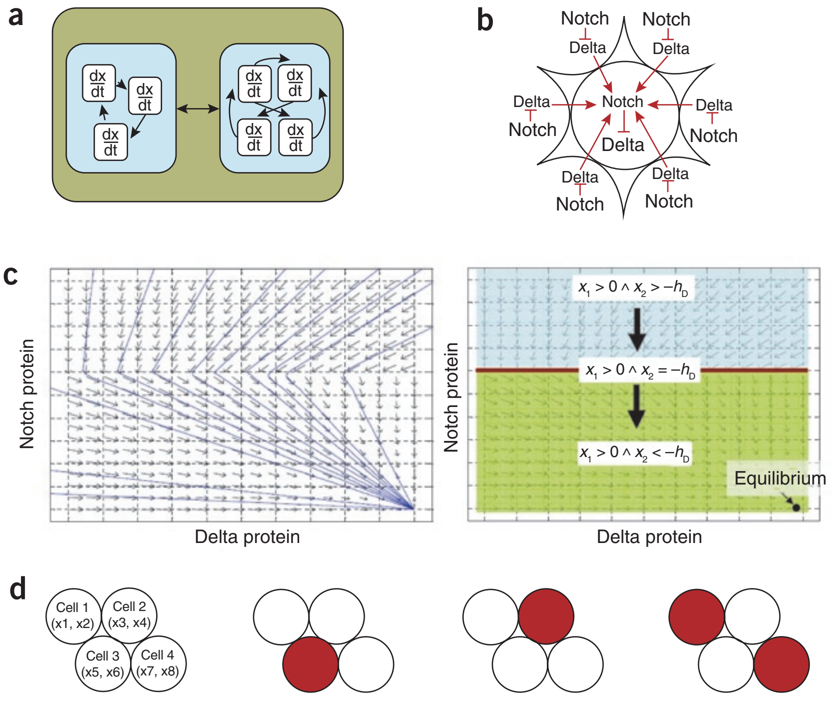


Figure 6 Hybrid systems. (**a**) In hybrid system models, discrete state changes modify the way continuous variables change. The discrete changes are governed by the values of the continuous variables. Each discrete state has its own differential equations, which govern the dynamics of continuous variables. (**b**) Influence diagram for Delta-Notch protein signaling network in a hexagonal close-packed lattice. (**c**) Plots of the continuous changes of the levels of Delta (x_1) and Notch (x_2) proteins as governed by changing differential equations that match three different discrete states, blue, brown and green. An isolated single cell will converge to a steady state where it has a high level of Delta protein and a low level of Notch protein. (**d**) From left to right, layout of four cell Delta-Notch network showing the variables associated with each cell; biologically consistent steady states of the four-cell network, a shaded cell represents a high steady-state concentration of Delta protein and low level of Notch protein; an unshaded cell has low Delta protein and high Notch protein at steady state. Figures reproduced from ref. 66.

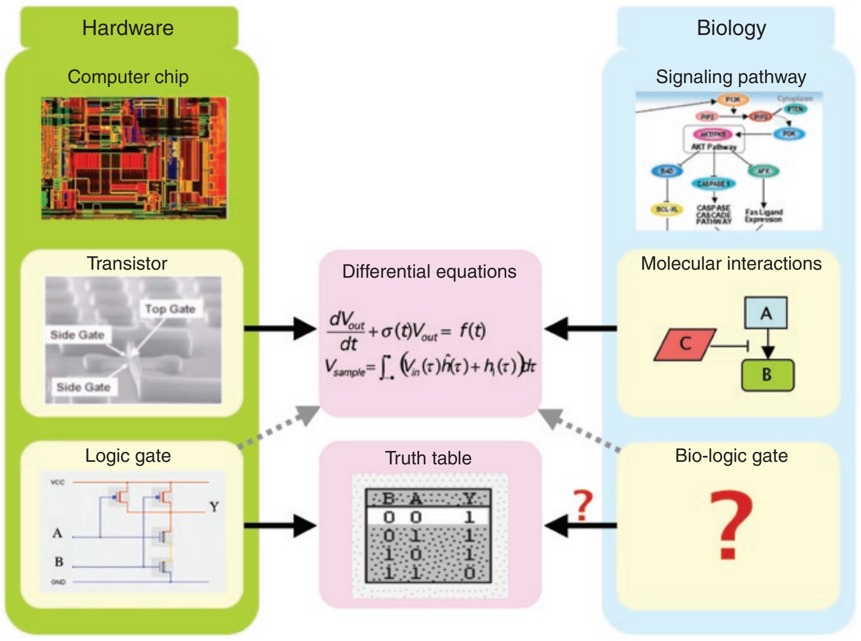


Figure 7 The analogy between hardware design and biological models. Computer chips are built from transistors. The most accurate way to describe the behavior of transistors is by differential equations. However, hardware designers work neither with transistors nor with differential equations. The basic units used for hardware design are logic gates, each of which consists of a few transistors. The behavior of a gate is not described by the sum of the differential equations for the transistors comprising it, but rather by a simple truth table of inputs and outputs, each with a Boolean value of 0 or 1. Conventional models of biological pathways use differential equations to describe molecular interactions. A key challenge is to identify a more abstract level, perhaps similar to the gate level in hardware design, based on functional units each of which involves many molecules. Such 'bio-logic gates' need not correspond to traditional biological entities, but could simply be useful building blocks of a language for defining abstractions of certain biological systems.

for the continuous variables). Hybrid systems aim to bridge the gap between mathematical models and computational models by combining the two. The discrete part of such models is the executable control mechanism that drives a hybrid system. A major portion of the work on hybrid systems has focused on constructing algorithms that perform the required simulation and analysis of the continuous part. Hybrid systems are particularly suitable to model biological systems where the relationships between substances change over time (that is, different equations are used for the same variables in different discrete states). Yet, they require exact quantitative data in order to fine-tune the equations that produce the continuous part of the model.

A hybrid model of the Delta-Notch decision process[65,66] (**Box 2** and **Fig. 6**) distinguishes between the control structure regulating the production of the Notch and Delta proteins (represented by discrete control variables) and the levels of these proteins (continuous values controlled by differential equations). The model reproduces the Delta-Notch decision: cells that express Delta are surrounded by cells that express Notch. In addition, a detailed analysis of a two-cell model shows that the model's behavior matches a classical model using nonlinear differential equations[65]. In a separate analysis of the same model, Ghosh and Tomlin checked which initial states of the hybrid system lead to the various Delta-Notch patterns[66].

Challenges

Executable biology poses new challenges both for computer science and biology. One challenge facing computer science is to adjust formalisms that were originally developed for modeling hardware and software systems to the modeling of biological systems. We must also develop techniques that handle the complexity and magnitude of biological systems and modeling tools that are more accessible to biologists. For biology, some of the key challenges are to develop quantitative techniques to experimentally test dynamic scenarios proposed by executable models, to identify useful building blocks of complex biological networks and perhaps most importantly, to shift biology toward an engineering science, where students learn to use formal approaches.

Developing quantitative measures to test system dynamics. Dynamic executable models can represent biologically important phenomena such as time and system dynamics that cannot be represented by static diagrammatic models. The influence of time on system behavior must thus be examined experimentally. The executable models of cell fate specification described above[11,13] predict such dynamic scenarios, suggesting a race between the epidermal growth factor recetor and LIN-12/Notch signaling pathways that determines cell fates during *C. elegans* vulval development.

Unfortunately, in practice, time is experimentally daunting. Using techniques that enable the direct quantitative measurement of pathway activities or protein levels, at a single-cell resolution, is not yet common practice. Experimental data, such as fluorescence levels of tagged proteins or immunoblots, are usually limited to unit-less ratios of expression levels that are only proportional to the actual protein concentrations. Not having direct measures can significantly hinder or complicate finding parameter values. In many studies, the linear relation between the concentrations of fluorescent proteins and their measured fluorescence intensities has been used to measure protein levels in living cells, albeit only in relative terms and not in absolute numbers[67,68]. Hence, improvements in the existing experimental methodologies to enable direct quantitative measurements are essential. One of the recent efforts to follow this path is the development of a technique that converts observed fluorescence intensities into numbers of molecules[69]. The challenge is twofold: first, to develop techniques that allow such quantitative measurements; and second, to allow these measurements to be taken continuously from the same element (e.g., cell, tissue), providing a dynamic view of the involved processes. From the perspective of executable biology, quantitative parts can be then incorporated into computational executable models to establish probabilistic or hybrid models[64], adding another dimension of accuracy to computational models.

Identifying 'logic gates' in biology. A complex system offers different characteristics when viewed from different distances. In a close-up view of an electronic circuit, for example, the circuit appears as a continuous-time system where individual transistors are visible and these building

Table 1 Comparison of *Executable Biology* modeling approaches

Modeling approach	Boolean networks	Petri nets	Process calculi	Interacting state machines	Hybrid models
References[a]	32–36	45–47, 52	56–63	8–16	65–66, 73
Referenced applications	Gene regulatory networks	Metabolic and signal transduction pathways	Metabolic and signal transduction pathways	Intracellular signaling, cell-cell interactions	Cell-cell interactions
Examples of modeled systems	Yeast cell-cycle regulation	EGFR signaling pathway, tryptophan regulatory network, glycolysis pathway	RTK-MAPK and FGF signaling pathways	T-cell activation, thymocytes differentiation, *C. elegans* vulval development	Delta-Notch decision, bacteria quorum sensing
Examples of description languages	–	–	Pi calculus, Ambient calculus, Brane calculus	Statecharts, Reactive Modules	Hybrid automata
Time	Discrete	Discrete	Continuous	Discrete	Continuous
Concurrency[b]	–	Synchronous and asynch.	Synchronous and asynch. and stochastic	Synchronous and asynch.	Synchronous
Structuring[c]	–	–	Compositional	Hierarchical and compositional	Compositional
Referenced analyses	Reasoning about stability and robustness	Static analysis of system dynamics	Dynamic analysis (simulation) of molecule quantities	Static analysis of system dynamics (model checking)	Reasoning about stability and system dynamics
Examples of software tools	Matlab	Pathalyzer	BioSPI, SPiM, PEPA	Rhapsody, Mocha	Matlab, Charon

[a]The table refers only to the work cited in the references row. [b]Concurrency refers to the way in which different parts of a system interact and change. A synchronous state change is a state change where the individual parts of the system change their contributions to the state simultaneously. An asynchronous state change is a state change where different parts proceed independently of each other. [c]A language is compositional if the behavior of a system can be specified by modeling a set of interacting sub-systems. A language is hierarchical if models of sub-systems can serve as indivisible, reusable building blocks within a larger system model.

blocks are best modeled using differential equations. Viewed from further away, however, the same circuit appears as a discrete-time system where the identifiable building blocks are Boolean gates, which are naturally modeled in the mathematics of truth values (so-called propositional logic). The key enabler of very-large-scale integration (VLSI) (**Box 2**) design has been the insight that we can build circuits with a desired functionality, such as the processors of modern computers, by working solely at the Boolean level. In other words, the circuit designer uses logic gates as black boxes and does not need to know that each black box comprises transistors. Although of course, to design an individual gate, one has to look at transistors and to build a transistor, one has to look at the molecular level, the circuit designer is completely insulated from these details. Similarly, operating at a higher level, a computer architect does not even see individual gates, but works with more coarse-grained building blocks that implement arithmetic operations and storage registers. If a computer architect had to deal with gates or a circuit designer with transistors, the task at hand would be too complicated.

Biological systems are even more complex than electronic circuits. Yet the only universally accepted models are either very low level (that is, detailed), or very high level (that is, coarse grained): one kind of models deals with individual molecules and their interaction, whereas the other kind views entire cells as indivisible units. This situation is akin to one where the gate level is absent from circuit design. Better understanding, analysis and synthesis of biological systems may need a standardized intermediate layer, each of whose building blocks may consist of many molecules but plays a single functional role within a cell or pathway. Such building blocks could be dubbed 'bio-logic gates' (**Fig. 7**). We do not suggest that such gates correspond to biological entities. Rather, the question is whether useful biological models can be assembled from a small number of different, possibly parameterized pieces (that is, building blocks). Identifying these pieces can be seen as one of the key challenges in biological modeling.

Biology as an engineering science. The modeling of biological systems is intended to improve our understanding of biological phenomena in their full complexity. For executable biology to realize its potential as a mainstream technique, the method must be used extensively by biologists. Thus, we have to suggest ways to represent models and data that are natural and can become standard. Stressing user friendliness, flexibility and visuality are critical to integrating computational tools into experimental biology research. Biologists are not expected to become engineers and therefore, one of the challenges facing computer scientists today is to design modeling languages and build software tools that are more accessible to nonexperts. We hope this will enable integration of executable biology into everyday biological methodology.

At the same time, a major challenge for biologists is to apply more formal approaches in biology and to develop precise, unambiguous and standardized representations of biological knowledge and data[70,71]. Even with the most user-friendly tools, the construction of executable models is still similar to the engineering of hardware and software systems, which require the user to think about state machines and recipes of computation. Such algorithmic notions, to which computer scientists are introduced at an early stage, are crucial for the successful, large-scale application of these techniques by biologists. Part of the formal education of future biologists should therefore include instruction in engineering disciplines that teach an algorithmic and formal way of thinking and foster an appreciation of notions such as state changes, concurrency and abstraction. **[AU:OK?]** A main goal is to make computer science tools accessible to biologists. This requires, on one hand, an understanding by computer scientists of what kind and style of tools might prove useful to biologists and on the other hand, an understanding by biologists of what kind of assistance could be provided by computer science tools.

Although several areas of computer science have already been successful in assisting biologists, such as algorithmics in the decoding of the genome and geometric modeling in the visualization of proteins, also formal modeling and analysis of reactive systems, which has originated in computer science, may play a useful role in assisting biology.

Concluding remarks

Modeling offers great advantages for integrating and evaluating information, generating predictions and focusing experimental directions. The long-term vision is that system-level models will revolutionize our understanding of biology and eventually enable design of new therapies. Although we are a long way from achieving this goal, formal models promise to change the face of biology and medicine. Moreover, the magnitude of the systems to be handled will challenge computer science to develop techniques that handle ever more complex models. Abstraction and high-level reasoning will likely play important roles in this endeavor.

Although executable biology is a pioneering and powerful approach in this direction, research into the design and use of computational models is still in its infancy and requires closer collaborations between biologists and computer scientists. The inherent differences between mathematical and computational models, along with the difficulty of obtaining precise biological data, make both approaches indispensable. We therefore believe that executable biology is poised to take its place in mainstream biological research.

ACKNOWLEDGMENTS
We apologize to colleagues whose work was not reviewed due to lack of space. We thank John K. Heath, Alex Hajnal, Freddy Radtke and Nir Piterman for helpful discussions and critical readings of the manuscript and the anonymous referees for valuable comments. J.F. is particularly grateful to David Harel for introducing her to this line of research and for many fruitful discussions over the years. Our research is supported in part by the Swiss National Science Foundation under grant 205321-111840.

Published online at http://www.nature.com/naturebiotechnology/
Reprints and permissions information is available online at http://npg.nature.com/reprintsandpermissions/

1. Ideker, T., Galitski, T. & Hood, L. A new approach to decoding life: systems biology. *Annu. Rev. Genomics Hum. Genet.* **2**, 343–372 (2001).
2. Kitano, H. Systems biology: a brief overview. *Science* **295**, 1662–1664 (2002).
3. Aebersold, R. & Mann, M. Mass spectrometry-based proteomics. *Nature* **422**, 198–207 (2003).
4. Priami, C., Regev, A., Shapiro, E.Y. & Silverman, W. Application of a stochastic name-passing calculus to representation and simulation of molecular processes. *Inf. Process. Lett.* **80**, 25–31 (2001).
5. Regev, A., Silverman, W. & Shapiro, E. Representation and simulation of biochemical processes using the pi-calculus process algebra. *Pac. Symp. Biocomput.* 459–470 (2001).
6. Errampalli, D.D., Priami, C. & Quaglia, P. A formal language for computational systems biology. *OMICS* **8**, 370–380 (2004).
7. Cardelli, L. Abstract machines of systems biology. Transactions on Computational Systems Biology III. *LNCS* **3737**, 145–168, (2005).
8. Kam, N., Harel, D. & Cohen, I.R. in *Visual Languages and Formal Methods* Stressa, Italy, September 5-7, 2001 (IEEE, 2001).
9. Efroni, S., Harel, D. & Cohen, I.R. Toward rigorous comprehension of biological complexity: modeling, execution and visualization of thymic T-cell maturation. *Genome Res.* **13**, 2485–2497 (2003).
10. Kam, N. *et al.* in *First International Workshop on Computational Methods in Systems Biology (CMSB)*, Roverto, Italy, February 24–26, 2003 (ed. Priami, C.) *LNCS* **2602**, 4–20 (2003).
11. Fisher, J., Piterman, N., Hubbard, E.J., Stern, M.J. & Harel, D. Computational insights into *Caenorhabditis elegans* vulval development. *Proc. Natl. Acad. Sci. USA* **102**, 1951–1956 (2005).
12. Efroni, S., Harel, D. & Cohen, I.R. Emergent dynamics of thymocyte development and lineage determination. *PLoS Comput. Biol.* **3**, e13 (2007).
13. Fisher, J., Piterman, N., Hajnal, A. & Henzinger, T.A. Predictive modeling of signaling croostalk during *C. elegans* vulval development. *PLoS Comput. Biol.* **3**, e92 (2007).
14. Sadot, A. *et al.* Towards verified biological models. in *Transactions on Computational Biology and Bioinformatics* (in the press).
15. Harel, D. Statecharts: a visual formalism for complex systems. *Sci. Comput. Program.* **8**, 231–274 (1987).
16. Alur, R. & Henzinger, T.A. Reactive Modules. *Form. Methods Syst. Des.* **15**, 7–48 (1999).
17. Giurumescu, C.A., Sternberg, P.W. & Asthagiri, A.R. Intercellular coupling amplifies fate segregation during *Caenorhabditis elegans* vulval development. *Proc. Natl. Acad. Sci. USA* **103**, 1331–1336 (2006).
18. Janes, K.A. & Yaffe, M.B. Data-driven modelling of signal-transduction networks. *Nat. Rev. Mol. Cell Biol.* **7**, 820–828 (2006).
19. Aldridge, B.B., Burke, J.M., Lauffenburger, D.A. & Sorger, P.K. Physicochemical modelling of cell signalling pathways. *Nat. Cell Biol.* **8**, 1195–1203 (2006).
20. Janes, K.A. & Lauffenburger, D.A. A biological approach to computational models of proteomic networks. *Curr. Opin. Chem. Biol.* **10**, 73–80 (2006).
21. Hua, F., Hautaniemi, S., Yokoo, R. & Lauffenburger, D.A. Integrated mechanistic and data-driven modelling for multivariate analysis of signalling pathways. *J. R. Soc. Interface* **3**, 515–526 (2006).
22. Stelling, J., Sauer, U., Szallasi, Z., Doyle, F.J., III & Doyle, J. Robustness of cellular functions. *Cell* **118**, 675–685 (2004).
23. Stelling, J. Mathematical models in microbial systems biology. *Curr. Opin. Microbiol.* **7**, 513–518 (2004).
24. Alon, U. Network motifs: theory and experimental approaches. *Nat. Rev. Genet.* **8**, 450–461 (2007).
25. Davidson, E.H. *et al.* A genomic regulatory network for development. *Science* **295**, 1669–1678 (2002).
26. Bolouri, H. & Davidson, E.H. Modeling transcriptional regulatory networks. *Bioessays* **24**, 1118–1129 (2002).
27. Bolouri, H. & Davidson, E.H. Transcriptional regulatory cascades in development: initial rates, not steady state, determine network kinetics. *Proc. Natl. Acad. Sci. USA* **100**, 9371–9376 (2003).
28. Clarke, E.M., Grumberg, O. & Peled, D. *Model Checking* (MIT Press, Cambridge, Massachusetts, 1999).
29. Schaub, M.A., Henzinger, T.A. & Fisher, J. Qualitative networks: a symbolic approach to analyze biological signaling networks. *BMC Syst. Biol.* **1** (2007).
30. Kauffman, S.A. Metabolic stability and epigenesis in randomly constructed genetic nets. *J. Theor. Biol.* **22**, 437–467 (1969).
31. Glass, L. & Kauffman, S.A. The logical analysis of continuous, non-linear biochemical control networks. *J. Theor. Biol.* **39**, 103–129 (1973).
32. Shmulevich, I. & Zhang, W. Binary analysis and optimization-based normalization of gene expression data. *Bioinformatics* **18**, 555–565 (2002).
33. Shmulevich, I., Lahdesmaki, H., Dougherty, E.R., Astola, J. & Zhang, W. The role of certain Post classes in Boolean network models of genetic networks. *Proc. Natl. Acad. Sci. USA* **100**, 10734–10739 (2003).
34. Li, F., Long, T., Lu, Y., Ouyang, Q. & Tang, C. The yeast cell-cycle network is robustly designed. *Proc. Natl. Acad. Sci. USA* **101**, 4781–4786 (2004).
35. Albert, R. & Othmer, H.G. The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in *Drosophila melanogaster. J. Theor. Biol.* **223**, 1–18 (2003).
36. Akutsu, T., Miyano, S. & Kuhara, S. Identification of genetic networks from a small number of gene expression patterns under the Boolean network model. *Pac. Symp. Biocomput.*, 17–28 (1999).
37. Friedman, N., Linial, M., Nachman, I. & Pe'er, D. Using Bayesian networks to analyze expression data. *J. Comput. Biol.* **7**, 601–620 (2000).
38. Ideker, T.E., Thorsson, V. & Karp, R.M. Discovery of regulatory interactions through perturbation: inference and experimental design. *Pac. Symp. Biocomput.*, 305–316 (2000). [AU: Please provide the missing volume number in this journal reference. (in reference 38 "Ideker, Thorsson, Karp, 2000").]
39. Sachs, K., Perez, O., Pe'er, D., Lauffenburger, D.A. & Nolan, G.P. Causal protein-signaling networks derived from multiparameter single-cell data. *Science* **308**, 523–529 (2005).
40. D'Haeseleer, P., Liang, S. & Somogyi, R. Genetic network inference: from co-expression clustering to reverse engineering. *Bioinformatics* **16**, 707–726 (2000).
41. de Jong, H. Modeling and simulation of genetic regulatory systems: a literature review. *J. Comput. Biol.* **9**, 67–103 (2002).
42. Chaouiya, C. Petri net modelling of biological networks. *Brief. Bioinform.* **8**, 210–219 (2007).
43. Li, C., Ge, Q.W., Nakata, M., Matsuno, H. & Miyano, S. Modelling and simulation of signal transductions in an apoptosis pathway by using timed Petri nets. *J. Biosci.* **32**, 113–127 (2007).
44. Reddy, V.N., Mavrovouniotis, M.L. & Liebman, M.N. in *1st ISMB* , Bethesda, Maryland, July 1993 (eds. Hunter, L., Searls, D. & Shavlik, J.) 328–336 (AAAI, 1993).
45. Barjis, J. & Barjis, I. in *Conference on Information Intelligence and Systems (ICIIS)*, Bethesda, Maryland, October 31–November 3, 1999, 4–9 (IEEE, 1999).
46. Simao, E., Remy, E., Thieffry, D. & Chaouiya, C. Qualitative modelling of regulated metabolic pathways: application to the tryptophan biosynthesis in *E. coli. Bioinformatics* **21** Suppl 2, ii190–ii196 (2005).
47. Steggles, L.J., Banks, R. & Wipat, A. in *4th International Conference on Computational Methods in Systems Biology (CMSB)*, Trento, Italy, October 18–19, 2006 (ed. Priami, C.) *LNCS* **4210**, 127–142 (2006).
48. Genrich, H., Küffner, R. & Voss, K. Executable Petri net models for the analysis of metabolic pathways. *Int. J. Softw. Tools Tech. Transf.* **3**, 394–404 (2001).
49. Goss, P.J. & Peccoud, J. Quantitative modeling of stochastic systems in molecular biology by using stochastic Petri nets. *Proc. Natl. Acad. Sci. USA* **95**, 6750–6755 (1998).
50. Srivastava, R., Peterson, M.S. & Bentley, W.E. Stochastic kinetic analysis of the *Escherichia coli* stress circuit using sigma(32)-targeted antisense. *Biotechnol. Bioeng.* **75**, 120–129 (2001).
51. Srivastava, R., You, L., Summers, J. & Yin, J. Stochastic vs. deterministic modeling of intracellular viral kinetics. *J. Theor. Biol.* **218**, 309–321 (2002).
52. Dill, D. *et al.* The Pathalyzer: a tool for analysis of signal transduction pathways. [in Biology], San Diego, December 2–4, 2005, *LNCS* **4023** (2005).
53. Efroni, S., Harel, D. & Cohen, I.R. Reactive animation: Realistic modeling of complex dynamic systems. *Computer* **38**, 38–47 (2005).
54. Damm, W. & Harel, D. LSCs: Breathing life into message sequence charts. *Form. Methods Syst. Des.* **19**, 45–80 (2001).
55. Sternberg, P.W. & Horvitz, H.R. The combined action of two intercellular signaling

pathways specifies three cell fates during vulval induction in *C. elegans. Cell* **58**, 679–693 (1989).
56. Milner, R. *Communicating and Mobile Systems: The pi-Calculus* (Cambridge University Press, Cambridge, UK, 1999).
57. Priami, C. The stochastic pi-calculus. *Comp. J.* **38** 578–589 (1995).
58. Regev, A., Panina, E.M., Silverman, W., Cardelli, L. & Shapiro, E.Y. Bioambients: An abstraction for biological compartments. *Theor. Comput. Sci.* **325**, 141–167 (2004).
59. Cardelli, L. Brane caluli. in *Computational Methods in Systems Biology (CMSB)* Paris, May 26, 2004 (eds. Danos, V. & Schächter) *LNCS* **3082**, 257 (2004).
60. Curti, M., Degano, P., Priami, C. & Baldari, C. Modeling biochemical pathways through enhanced pi-calculus. *Theor. Comput. Sci.* **325**, 111–140 (2004).
61. Calder, M., Vyshemirsky, V., Gilbert, D. & Orton, R. Analysis of signaling pathways using the prism model checker in *3rd International Conference on Computational Methods in Systems Biology (CMSB) (ed. Plotkin, G.) 179–190,* (ed. G. Plotkin) Edinburgh, Scotland (2005).
62. Calder, M., Duguid, A., Gilmore, S. & Hillston, J. Stronger computational modelling of signalling pathways using both continuous and discrete-state methods in *4th International Conference on Computational Methods in Systems Biology (CMSB)*, Trento, Italy, October 18–19 (ed. C. Priami) *LNCS* **4210**, 63–78 (2006).
63. Heath, J., Kwiatkowska, M., Norman, G., Parker, D. & Tymchyshyn, O. in *4th International Conference on Computational Methods in Systems Biology (CMSB)*, Trento, Italy, October 18–19 (ed. C. Priami) *LNCS* **4210**, 32–48 (2006).
64. Henzinger, T.A. The theory of Hybrid Automata in *Proceedings 11th IEEE Symposium on Logic in Computer Science* 278–292 (1996).
65. Ghosh, R. & Tomlin, C. Lateral inhibition through delta-notch signalling: A piecewise affine hybrid model in *4th International Workshop on Hybrid Systems Computation and Control*, Rome, Italy, *LNCS* **2034**, 232–246 (2001).
66. Ghosh, R. & Tomlin, C. Symbolic reachable set computation of piecewise affine hybrid automata and its application to biological modeling: Delta-Notch protein signaling in I*EE Transactions on Systems Biology*, volume 1, 170–183, June 2004.
67. Kaern, M., Elston, T.C., Blake, W.J. & Collins, J.J. Stochasticity in gene expression: from theories to phenotypes. *Nat. Rev. Genet.* **6**, 451–464 (2005).
68. Sprinzak, D. & Elowitz, M.B. Reconstruction of genetic circuits. *Nature* **438**, 443–448 (2005).
69. Rosenfeld, N., Perkins, T.J., Alon, U., Elowitz, M.B. & Swain, P.S. A fluctuation method to quantify in vivo fluorescence data. *Biophys. J.* **91**, 759–766 (2006).
70. Gilman, A. & Arkin, A.P. Genetic "code": representations and dynamical models of genetic components and networks. *Annu. Rev. Genomics Hum. Genet.* **3**, 341–369 (2002).
71. Lazebnik, Y. Can a biologist fix a radio?–Or, what I learned while studying apoptosis. *Cancer Cell* **2**, 179–182 (2002).
72. Kwiatkowska, M. *et al.* Simulation and verification for computational modeling of signaling pathways. in *Proceedings* of *Winter Simulation Conference*, Monterey, California, December 2–6, 2006, 1666–1674 (IEEE, 2006).
73. Alur, R. *et al.* Hybrid modeling and simulation of biomolecular networks. in *Fourth Internations Workshop on Hybrid Systems: Computation and Control*, Rome, Italy, March 28–30, 2001 (eds. Di Benedetto, M.D. & Sangiovanni-Vincentelli, A.L.) *LNCS* **2034**, 19–32 (2001).