DOI:10.1145/2001269.2001289

Exploring the connection of biology with reactive systems to better understand living systems.

BY JASMIN FISHER, DAVID HAREL, AND THOMAS A. HENZINGER

Biology as Reactivity

BIOLOGY IS NOT an exact science. Biological systems are messy and noisy, and our understanding of many biological scenarios remains extremely vague and incomplete. We cannot assign precise rules to the way cells behave and interact with one another, and we often cannot quantify the exact amounts of molecules, such as genes and proteins, in the resolution of a single cell. To make matters worse (so to speak), the combinatorial complexity observed in biological networks (for example, metabolic and signaling pathways) is staggering, which renders the comprehension and analysis of such systems a major challenge.

One way to explain a certain class of complex dynamical systems is to view them as highly concurrent *reactive systems*.[42] We argue that this perspective is a natural fit for many biological systems.[40] A reactive system is characterized by the way it responds to its inputs, as they arrive over time, sequentially, or concurrently. The system's behavior and outputs are not just a function of the input values but also of the order in which the inputs arrive, their arrival times, speeds, and locations, and so forth.

A living cell, we claim, is not only reactive in nature, but is the ultimate example of a reactive system, and so are collections thereof. As explained in Cohen and Harel 2007,[16] a cell succeeds by being robust and resilient. It reacts to inputs and perturbations and continues to survive thanks to its reactive dynamics. The cell is a reactive system that expresses a dynamic narrative, in which the DNA code is one of many formative inputs. Structural proteins, enzymes, carbohydrates, lipids, hormones, and other molecules also play key roles in forming and informing the system.

Biological systems are also highly adaptive, to both internal and external changes; they use signals coming in from receptors and sensors, as well as emergent properties to fine-tune their functioning. This adaptivity is another facet of the reactivity of such systems.

We are well aware of the fact that there are many aspects of biology that are not reactive, or that reactivity is not the best way to view them. These include, for example, structural aspects of chemicals. While we point to the importance of combining other modeling methods with reactive models, the main thrust of this article is to explain the connection of biology with reactive systems, and the benefits that can be gained from adopting such a view.

Viewing biology as reactivity is not just an illustrative analogy, but has

» key insights

- **Living cells, and collections thereof, can be viewed as reactive systems.**
- **Reactive models emphasize important aspects of biological systems, such as executability, concurrency and interaction, multiple scales, and combinatorial complexity.**
- **Concepts, languages, and tools for the description and analysis of reactive systems can help in the process of biological discovery, ultimately by providing biologists with virtual experimentation environments.**
- **Biological experimentation needs to obtain quantitative data across different levels, and reactive modeling needs to focus on incorporating and linking such data.**

far-reaching consequences. Complex reactive behavior is difficult to comprehend because it is neither predominantly computation-rich nor primarily data-intensive, and does not yield well to techniques based on algorithmics, mathematics, and data management. Rather, reactive systems "live" (if we may use a pun in an article on biology) in order to react.

Within computer science, several approaches have been offered for dealing with reactivity. One of the richest and most fruitful is that of *finite automata*, or *finite state machines*, which are used widely in many stages of the design of hardware and software systems. It suffices to observe the central role of state machines in standards for the engineering of embedded and real-time software and systems, such as the UML.

The original theory of finite automata, which was conceived of in order to model hardware circuits, has brought in its wake many extensions and variants, such as automata on infinite words and trees; communicating, hierarchical, and timed state machines; Statecharts, and many others.[1,75] In addition, and related to these, special logics, algebras, and calculi have been devised for reactivity, first and foremost among which are the varieties of *temporal logic*[63] and *process algebra*.[57] Scenario-based languages have been used to model biology too, such as *live sequence charts* (LSC).[19] All of this, in turn, has resulted not only in numerous languages and tools for specifying and programming reactive behavior, but has also given rise to powerful means for analysis, such as model checking.[15]

However, when reviewing the great efforts made in understanding and building complex reactive systems, it is not only the final results—languages and tools—that stand out. An even more important contribution can be found in the fundamental insights, concepts, and ways of thinking that have resulted from this research. Hand-in-hand with the central notion of reactivity go the discrete event-based *execution and simulation* of dynamical systems, which

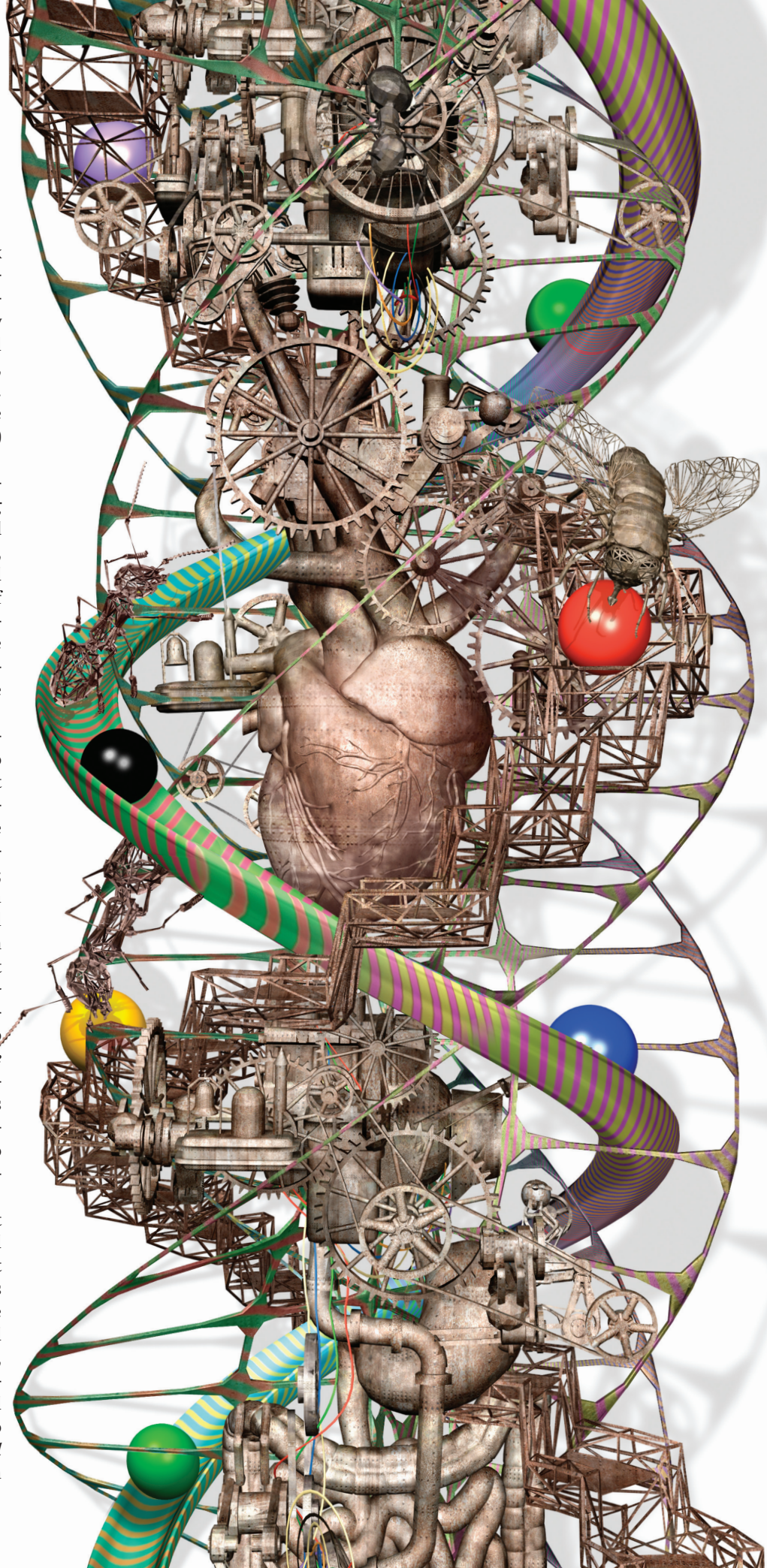

ILLUSTRATION BY J.F. PODEVIN

requires a fundamental understanding of *parallelism*, *interaction*, and *causality*; the design of complex systems from building blocks, requiring means for *composition* and *encapsulation*; and the description of systems at different levels of granularity, requiring methods for *abstraction* and *refinement*.

Of course, many of these concepts are not exclusive to reactive systems, but they are critical for understanding them, and they are among the main ideas that render possible the reliable development of truly complex reactivity. The claim made in this article is that these ideas can be of great benefit in the modeling and analysis of biology.

Reactivity and Biology

The process of modeling a piece of biology is very different from that of modeling a human-made system. The motivation is different, the goals are very different, the people involved are also different, the scales are different, and so on. Still the underlying maxim of this article is that a fighter jet and a fruit fly, for example, have many things in common, and that there is much to be gained from using ideas from engineering the former to reverse-engineering the latter, even though the fly is in a way far more complex than the airplane.

Biological systems are ultimately molecular and cellular machines. Using a fixed set of building blocks (molecules) with some fixed functionalities (for example, cleavage, connection, ionization, phosphorylation), cells emerge, and these multilevel machines then utilize various combination techniques to give rise to the life we see around us. The "execution" of a biological system is distributed to a level that is still beyond comprehension. In a sense, a biological system can be viewed, almost on a philosophical level, as a collection of cells (each containing a multitude of molecules) that work in concert and independently to achieve a mutual goal without a central controller that coordinates them all. The cells in such a collection communicate and pass information to each other in order to help achieve their mutual goal. In doing this, they can also affect limited control on their environment.

Let us now concentrate on the programs (to put it simply, the DNA molecules) that drive these enormously distributed processes. In the case of the fruit fly *Drosophila Melanogaster*, for example, a 165 million base-pair program defines approximately 13K functions (proteins). Such programs deal with two entangled and inseparable tasks: how to evolve the machine (cell collection) from a single cell, and how to run the machine so that it eventually creates other such machines. At all stages, both the development and maintenance are determined by which genes are expressed and which are not. Effectively, in each cell, the expressed genes constitute a *local control state* of the program, which tells the cell which functions it should execute and which are irrelevant to its (current) role. The state is local because it may be different in different cells. At all points, communication with other cells and reaction to the environment are the keys to determine what happens next.

More precisely, the expressed genes tell the cell and its machinery which proteins to produce and how many of them, and which proteins not to produce. Gene expression and the corresponding protein production are the basic mechanisms that govern all actions of a cell. Proteins serve various purposes in the cell itself and also act as mediators of communication between cells. The protein concentration levels within a cell can be considered to be the *local data state* of the machine. As in computer programming, the data state helps prescribe the control flow, in that the proteins determine which genes are expressed and which are not.

Consider the first part of the program, the one that tells the system how to evolve. All multicellular organisms evolve from a single cell. The main engine driving this part is called *differentiation*, which is the process by which a cell that divides can give rise to what will ultimately become very different types of cells. There are mainly two mechanisms for cell differentiation. One, which is mostly active in the early stages of development of the machine, is by *asymmetric segregation*—asymmetric accumulation of substances inside a single cell that divides into two different kinds of cells. The second, which becomes active after a few initial divisions have taken place, is governed by communication between cells, by *morphogen gradient* or *inductive signaling*. The former breaks the symmetry between a group of equivalent cells, by one cell, external to the group, secreting a gradient of molecules (for example, according to distance from the secreting cell). The latter involves direct communication between cells, leading them to choose different fates through the passage of information. All communication is carried out by transferring and manipulating molecules. Such a process may start, for example, by one cell sending a molecule to a neighboring cell, which will then trigger the creation of another molecule in that neighboring cell (function activation).

The differentiation phase of cells leads to a global system state where different cells have specialized to perform certain roles that are necessary for the mutual fulfillment of their common goal (reproduction). The second part of the program guides the system's behavior after having evolved from an embryo, although major pieces are already in use throughout development because the cells are alive and need to maintain themselves during the process. Initially, this part of the program includes little coordination and communication between cells. However, as the cells specialize, they start to communicate and to manipulate their environment in order to perform basic actions such as transferring nutrients from one place to the other, transferring oxygen, and so on.

Perhaps the most magical part of this program controls movement (actuation). This process is slightly different, as it involves a new form of communication through electrical signals that are transmitted via nerve cells, and which are then translated into local molecular mechanisms. But notice that when a muscle contracts, the nerve reaches only some of the muscle cells, and responsibility for the remainder of the signal's effect must be taken over by other mechanisms. In the reverse direction, sensors such as eyes, ears, and taste buds pass information from chemical to electrical signals. Moreover, as mentioned earlier, the ability of biological systems to adapt, as a result of incoming signals, is also a highly relevant feature of their reactivity.

These are the kinds of processes that we aim to understand better by creating executable reactive models. The ul-

timate challenge is to understand how the behavior of an organism, or a part thereof, emerges from the individual behaviors of the cells in the collection. We believe this is best done by focusing on the reactivity of cells and the communication between them.

Concurrency and interaction. The essence of a reactive system is that different parts of the system are simultaneously active ("alive") and interact with each other. In the case of computerized systems, these can be software processes, hardware components, intelligent agents, or the environment, and in the case of biology they can be molecules and molecular processes, cells, tissues, bacteria, or again, the environment.

The operation that combines several simultaneously active parts of a system is called *parallel composition* because the individual activities happen concurrently, in "parallel," rather than sequentially (one after the other). However, in computer science there are several different interpretations of what "in parallel" actually means. At one extreme is the *clock synchronous* interpretation, where execution proceeds in lockstep with a global clock, as a sequence of discrete steps, with each building block (component) of a system contributing one action to each time unit. This happens, for example, in a clocked hardware circuit built from individual logic gates without combinational loops, where each gate performs one operation per clock tick. At the other extreme is the *interleaved asynchronous* interpretation, where each action represents the contribution of a single component, but different actions, possibly carried out at different time rates, may represent different components. This happens, for example, in distributed software systems where the individual processes proceed at independent speeds. Implicit to the asynchronous interpretation is a scheduler, which chooses for each step the component that will contribute to that step. The scheduler is usually assumed to be "fair," the simplest interpretation of which is that it cannot neglect to choose any component forever.

Both kinds of concurrency occur in biology, for example, in molecular processes "running" in parallel, and techniques for capturing both have found their way into biological models. Still, sticking religiously to these two interpretations may be inadequate for representing the parallelism that arises in a large collection of individual cells. Cells, for example, often proceed "roughly" in lockstep, but not completely so: some reactions may be a bit faster here but a bit slower there, but biological reactions do not proceed at completely independent rates either. An example of how to adapt useful computer science techniques to biology can be found in the recent compromise between these two called *bounded asynchrony*.[33] It allows the components of a system (for example, cells) to proceed independently while bounding the differences in their rates. Bounded asynchrony has been used to accurately model some of the experimental observations made about certain cell-cell interactions: it captures the variability observed in cells that, although equally potent, assume distinct fates, and thus can be used to provide mechanistic explanations for some phenomena that are observed during cell-fate determination.

The process of modeling a piece of biology is very different from that of modeling a human-made system. The motivation is different, the goals are very different, the people involved are also different, and the scales are different.

Bounded asynchrony is but one example of how the modeling of biological systems may give rise to new variations of reactive concepts. Once it is agreed upon what the most suitable interpretation of "parallel progress" means for a collection of cells, the communication between these cells needs to be modeled. Again, there are several standard alternatives, designed for modeling hardware and software, which may not be readily adequate for our purposes. Traditionally, the interaction between concurrent processes can happen through sharing state or through sending messages. This distinction has led, for example, to thread-based programming languages on one hand, and actor-based languages on the other. The transfer of molecules between cells seems more akin to message passing, but within that paradigm itself there exists a wide spectrum of design choices, ranging from rendezvous, which stops the execution of two processes until simultaneously the sending process is ready to send and the receiving process is ready to receive a message, to unbounded message buffers, which retain sent messages until they are received. Explicitly spatial models[11,69] are natural candidates for capturing morphogen gradients.

While a biological scenario may suggest a particular execution and interaction model, often it is a matter of style and the availability of analysis tools that determines the choice of model. If there is any lesson to be learned from several decades of research on concurrency theory in computer science, it is that we should not preach the use of any particular modeling languages or methods, but the very notion of reactive modeling itself, and the freedom of choice it offers through a wide variety of different execution and interaction mechanisms.

Towers of abstraction. Every model is necessarily a simplification of a physical situation. The modeler is interested in some, but not all, aspects of the physical system. The key to modeling a complex system is to design a simplification, such that (1) the behavior of the model is correlated to behavior of the system and (2) the model is amenable to mathematical or computational analysis. Point (1) measures models according to their *precision*; point (2), according to their *performance*. Precision must be sacrificed in order to gain performance, but this must be done in a quantifiable way.

Computer science offers a very rich and useful theory, called *abstraction*, for trading off precision against performance in a principled manner.[17] The theory of abstraction is used in computer science very beneficially in work on verification and testing, and is aimed at relating models of different precision. It thus differs greatly from theories of approximation, which measure the precision of a model in terms of an error bound on how much a model may deviate from the system. A theory of abstraction, by contrast, measures the precision of a model in terms of which properties of the system are preserved in the model—hence the relevance to verification.

Consider, for example, the property that event *A* is never followed by event *B*. A model may preserve such a property, despite possibly introducing a very large error. On the other hand, even a model that introduces only a very small error may violate the property when the actual system does not. The preservation of properties can ensure that it suffices to check a property on the model (simple) in order to conclude that it holds for the real system (complex). This principle lies at the heart of all hardware and software design. Modern computer systems could not be built without several layers of abstraction—the gate-level abstraction of the underlying physics (transistors); the register level; the machine instruction level; the programming language; among others.

Verification can become very expensive, or impossible, for human-made systems, and biological systems are often even more complex. The good news is program and system verification has made enormous strides in recent years.

In analogy, a biological system may be viewed at many different levels, ranging from the molecular level to the cell level to the level of organs and entire organisms. The power of an abstraction layer derives from the fact that all lower-level details may be omitted from consideration. For example, when designing a logical circuit from Boolean gates, the designer does not need to know about voltages and capacities. Different layers may exhibit different scales in space and time, even switching between continuous and discrete scales (for example, continuous voltages representing Boolean values). By contrast, in biology, we have not (yet) been able to identify building blocks from which we can explain metabolic pathways and cell behavior without referring to the underlying biochemical (molecular) mechanisms.

In multiscale reactive systems, an additional characteristic phenomenon is the *emergence* of new high-level properties. An emergent property is a behavior of the system that is not easily expressed at a lower scale. Life, for example, is an emergent property; none of the component molecules of a cell are alive, only a whole cell lives. The cell as a whole emerges only when we zoom out, so to speak, reaching the scale at which it functions as an object with its own interactions with other cells and molecules. Thus, interactions at one scale create new objects at a higher scale, which is the essence of emergence. Quoting from Cohen and Harel,[16] "A major goal of systems biology is to learn how the concurrent reactions and interactions of the lower-scale components of a cell, organism, or society generate emergent properties visible at higher scales and higher layers of reality."

All this leads to the need to be able to observe and manipulate biological systems on multiple scales. Abstraction can work wonders here, and if carried out multiple times we get a *tower of abstractions*. A unique feature of software that helps in handling multiple scales

is *inheritance*, where an existing behavior is taken and augmented with a few modifications. This encapsulates the previous behavior and overrides or extends parts of it. One of the major challenges in biological modeling is to find similar means to encapsulate biological and biochemical complexity that will allow us to use abstraction beneficially to bridge and relate different scales in order to manage the immense complexity observed in living systems. Once we have such multiscale models we will need to search for the right computational frameworks that will allow us to zoom back and forth between lower-scale data and higher-scale behavior, while experimenting *in-silico*. This, we feel, is an ideal way to study emergence computationally. A modest attempt in this direction can be found in the *Biocharts* approach of Kugler et al.[52]

Noise and choice. Stochasticity has received much attention in systems biology,[4,55,56] as numerous experimental studies have reported the presence of probabilistic mechanisms in cellular processes.[26,29,61] The investigation of stochastic properties of biological systems requires that computational models take into consideration the inherent randomness of biochemical reactions. Stochastic kinetic approaches give rise to dynamics that differ significantly from those predicted by deterministic models because a system may follow very different scenarios with non-zero but varying likelihoods.

The dogma for this kind of modeling assumes that a molecular mixture is well stirred and has fixed volume and temperature (though PDEs can be used to model variations in these too). The state of a network of biochemical reactions at any point in time is then given by the population vector of the involved chemical species (such as, molecules). The temporal evolution of the system can be described by a continuous-time Markov process,[37] which is usually represented as a system of ordinary differential equations (ODEs) called the *chemical master equation* (CME). While individual system parameters, such as the mean of the state distribution changing in time, can be studied using deterministic differential equations, this is inadequate for uncovering branching, switching, or oscillatory behavior, such as cell fate determination (the mean between two alternative cell fates is hardly meaningful). Such phenomena require a fully stochastic analysis.

However, building a stochastic model that would mimic sufficiently accurately the stochastic behavior of the actual biological system is extremely difficult when sufficiently accurate rates of change are not known, as is usually the case. In addition, we have no satisfactory theory for abstracting stochastic models. This becomes a central issue when we wish to analyze the higher-level tiers of a biological model—say, those at the intercellular level—by hiding the underlying molecular reactions. In such situations, a nondeterministic modeling of the possible behavioral alternatives of a system may be justified. For example, for determining possible cell fates, it has proved fruitful to quantize concentration levels of molecules into a few discrete ranges (for example, low, medium, high) with nondeterministic transitions between the possible ranges.[32,34] A nondeterministic model can only provide potential outcomes, without the corresponding probabilities, but it does provide hypotheses that can be confirmed or refuted experimentally.

In computer science, many formalisms have been designed—or existing ones have been extended—to support nondeterministic transitions for modeling alternative choices; for example, Petri nets, various kinds of interacting state machines, live sequence charts (for example, Peterson,[62] Harel,[38] Harel and Gery,[41] Damm and Harel[19]). Many of these formalisms have been used to model biology as well.[6,23,25,32,49,73] The question of how such discrete, nondeterministic models relate to the underlying continuous, stochastic mechanisms, and which properties they preserve, remains an interesting topic of investigation. Hybrid (mixed discrete-continuous) abstractions can also play a central role to bridge the gap.[35,45,74] To find an optimal trade-off between precision and performance, it may be best to treat some system parameters as variables that change continuously in time, while others can be safely represented as Boolean switches.

One main advantage of nondeterministic over stochastic models, and of discrete over continuous models, is the former more efficiently support a broad class of techniques, generally subsumed under the title of *verification*, which we discuss here.

From simulation to verification. Computer science offers a rich spectrum of means for assessing the dynamic properties of reactive models, ranging from simulation to verification. While simulation generates one behavior of a model at a time, verification looks systematically at the set of all possible behaviors.

Simulation has a long tradition in computational science, based mostly on numerical methods for solving equational models of a system. For example, from the CME for a system of (bio) chemical reactions, stochastic simulation can be used to generate trajectories of the underlying Markov process.[36] Simulation methods are in widespread use because they are easy to implement, and each simulation run can be viewed as a single "in silico'' experiment, thus fitting well into the methodology of experimental science. However, if we are interested in properties of the set of all runs, such as estimates for probability distributions on the system state at a given point in time, then the number of trajectories required for statistical accuracy is very large.[21] This is because in order to halve the confidence interval of an estimate, four times more trajectories have to be generated. Consequently, even when computationally feasible, stochastic simulation may often result in a very low level of confidence in the accuracy of the results.

More information can be obtained, for example, by a reachability analysis of the model, which explores the state space from an initial state or state distribution in a breadth-first, rather than depth-first, manner. The distinction between simulation and reachability analysis is akin to the distinction between program testing and program verification. A reachability analysis can provide insights into biochemical models,[22] but many techniques that have been developed for coping with large and unbounded state spaces in the reachability analysis of nondeterministic models—such as model abstraction, model decomposition, symbolic data structures, symmetry, and partial-order reduction—have yet to be adapted satisfactorily to stochastic models.

It goes without saying that one must

not forget the main difference between simulation and verification: the worst-case computational intractability of the latter. In general, as is so well-known, verification can become very expensive, or impossible, for human-made systems, and biological systems are often even more complex. The good news is that program and system verification has made enormous strides in recent years: the worst-case performance can sometimes be alleviated by clever and powerful means, though obviously this is outside the scope of the present article. It is these advances that we hope to be able to utilize in modeling and verifying biology, too.

As discussed earlier, higher precision in the model generally means lower performance in the analysis. The modeler therefore aims at the lowest possible precision that preserves the property of interest. Reachability analyses offer the possibility of dynamically changing the level of abstraction during the analysis.[14] In this way, the precision of a model can be refined on demand, precisely in those areas of the state space where more detail is required for determining the truth or falsehood of a given property. Reachability analysis can also be equipped with rich languages for defining and checking temporal properties of systems, such as temporal logics—a research direction known as *model checking*; for example, see Calzone et al.,[10] which describes the BIOCHEM tool used in the arena of systems biology.

Reactive models come with an operational semantics; that is, every model defines a virtual machine with instructions for executing the model step by step. This is true not only of textual modeling and programming languages, but also of visual formalisms, such as Petri nets[62] and Statecharts.[38] This is in contrast with most equational and denotational models, where generating trajectories is a mathematical task that requires algorithmic and numeric insights. This benefit of reactive models has led to the term "executable biology."[32] An operational semantics is not only enormously helpful in simulation and reachability analysis, but also offers the possibility of interactive execution—or "playing *in silico*."[39]

A second important attribute of models of reactivity is that they offer a compact syntactic description of a dynamical system. The actual number of states and transitions of a biological system—its semantics—is usually very large or unbounded. Scientists and mathematicians often make little distinction between the semantics of a system, say, as a Markov process, and its description, say, as a transition probability matrix. By not differentiating sufficiently between the two, a potential critical advantage of syntax is lost; namely, that the description of a system can be much smaller than the system itself. For example, the rule-based description of a (bio)chemical system can be exponentially smaller than the matrix description of the same system; in fact, even many infinite state systems have finite descriptions. The syntax of a language can offer scaling operations, such as parallel composition and encapsulation, which greatly magnify this effect.

Syntax matters also in other ways. For one, the right choice of syntax can substantially improve the performance of analysis methods. Certain crucial optimizations of reachability analysis, such as on-the-fly state-space generation and partial-order reduction (things that can be extremely helpful when analyzing a piece of biology), are only available when the individual transitions of the underlying system are described compactly, by a syntactic expression (a rule, a process algebraic term, or a state machine) rather than a matrix equation. Furthermore, an inductively defined syntax, which features operators on basic expressions for constructing more complex expressions, offers the possibility of defining a structured operational semantics. In such semantics, the execution engine is defined compositionally; that is, it is put together from small primitives by using the syntactic operators of the language. Finally, a visual syntax makes modeling appeal to a larger group of people, such as biologists, and reactive models offer natural opportunities for visual representations.

State of the Art and Challenges

A large number of efforts to construct and analyze complete reactive models of various biological systems are under way. For lack of space, we can point only to a few of these efforts. We arrange them in a way that proceeds from more detailed, molecular-level models to more abstract, cell-level models. Alas, establishing formal relationships between the various levels, that would enable seamless combinations of executable models, still remains one of the key challenges in this area (see Kugler[52]).

Individual molecules can be modeled and combined as processes within a process algebra. Such *process-calculus models* stress the importance of concurrency and interaction between molecules as the main driver behind the dynamics of biological systems. Initial work suggested the use of the pi-calculus[59] as a modeling language for molecular interactions,[70] using it to study the cancer-related signal transduction pathway RTK-MAPK and to build the BioSPI simulation environment. The language was later extended to the stochastic pi-calculus[66] in order to model a gene-regulatory positive feedback loop.[68] These initial successes have led to the design of several bio-inspired and location-aware process calculi, such as BIO-PEPA, the ambient calculus,[69] and the brane calculus.[11] The methodology was applied, among others, to transcription factor activation and the glycolysis pathway,[18] RKIP inhibition of ERK,[9] the FGF pathway,[51] and EGFR signaling.[76] These approaches are very beneficial on the level of pathways and molecular interactions, but lack natural power of expression when dealing with larger biological systems, say, on the intercellular level.

On a higher level, instead of representing individual molecules as computational objects, we may refer to quantities of molecules through variables in a programming language. A single reaction may increment or decrement such variables. Inspired by guarded commands and reactive modules,[2] languages in this style were used for building qualitative models[7,32] and discrete-time Markov processes.[50] They were later extended to continuous time, for describing biochemical reaction networks. Such *transition-class models* may use general arithmetic expressions for specifying reaction rates.[45] These models can be executed (interpreted) directly, without the need for constructing a separate simulation engine. They can compactly represent unbounded quan-

tities of molecules. It is their executability and compactness that allows the stochastic analysis and model checking of complex molecular systems, often involving many different molecule types and very large molecule quantities, such as a genetic toggle switch. Further extensions lead to hybrid systems, handling particularly large quantities in a continuous domain.[36,45]

Another class of languages for modeling biology is based on term-rewrite systems.[24] *Rule-based models* can offer an even more compact syntax than guarded-command definitions of molecular reactions, by defining the reactive behavior of molecules in terms of what happens at individual binding sites within molecules. By formulating rewrite rules whose patterns are matched against fragments of molecules, one can avoid referring explicitly to the state of an entire molecule, and instead specify only the state of the affected sites before and after the application of the rule. Such rules, which may simultaneously apply to many different sites within a single complex molecule, can lead to a further reduction in the size of the description of a model. Rule-based modeling has been applied to an increasing number of systems such as signal transduction in the immune system,[29,54,55] bacterial migration,[6,61] cancer-related signaling,[9,21] mechanisms by which various proteins regulate cell signaling through their association with membrane proteins,[44] and more.[48] Ideas from programming languages, such as abstract interpretation, have influenced the design of more recent rule-based languages for biological applications.[31]

At the highest, non-molecular level, *state-machine based models* provide a visual approach for defining the behavior of complex objects, such as collections of cells, over time. Interaction mechanisms between state machines can specify causal relationships between events and state changes in different objects. A hierarchical structure allows one to view a system at different levels of detail (for example, whole organism, tissues, cells). For example, the language of Statecharts supports interacting and hierarchical state-based modeling,[39] based on a visual syntax, and has been used to model immune-cell activation and differentiation;[25,49] cellular decision-making processes during animal development;[32,35,50,72] as well as organ formation.[74] State-machine models are particularly suitable for describing mechanistic models of multicellular systems that are well-understood qualitatively. Such models do not require detailed quantitative data relating to the number of molecules and reaction rates, and indeed are inadequate when it comes to modeling pathways and molecular interactions. The possibility of hierarchical structuring is particularly useful in cases where behavior is distributed over many cells and where multiple copies of the same process are executed in parallel. These models also allow the application of strong analysis tools such as model checking. Combined methods seem very promising when it comes to systems for which one wants to model intercellular as well as intermolecular behavior, such as the Biocharts approach that is sketched in Kugler, Larjo and Harel.[53]

The research directions described in this article are intended, first and foremost, to yield beneficial results in biology and medicine, thus enhancing our ability to improve our lives.

There are many additional efforts in modeling biological systems that have not found their way into this article, mainly because they are less along the reactive system lines presented here. Some of these are particularly exciting and insightful. They include abstract chemical machines,[13] work on the brane and ambient calculi mentioned earlier, and other efforts to integrate behavior directly with space and movement considerations. In addition, the reader is referred to Priami's recent article in *Communications*[68] for a different perspective, more algorithmic, and thus somewhat complementary to ours.

Are we doing science or engineering? It is worth briefly addressing the connections between biological modeling and both science and engineering. Our discussion follows recent insights voiced by Luca Cardelli.

In science and engineering we find notions of "systems," which are the objects of study, and "models," which are formal or semiformal descriptions of those systems. The *scientific method* starts from a given natural system of interest, and through a discovery process we gain knowledge about the system, until we are in a position to formulate a model that aims to characterize its important features. Scientists then attempt to falsify that model, showing that it is inaccurate or incorrect, usu-

ally by experimentation based on the model's predictions. This process of falsification is the defining characteristic of scientific models.[65] If it is successful, we have discovered a property of the system that is not captured correctly by the model, which may lead to an improved model and to a new falsification cycle. If it is unsuccessful, the model stands, which does not necessarily mean that it is correct: it simply means that it has not yet been proven wrong (and we should keep trying). A main feature of the scientific method is that the "truth" is in the system, while the model is in principle never fully correct.

This Popperian approach has been adopted as part of the recent idea of a Turing test aimed at biological modelling,[40] where the model is deemed valid if it cannot be told apart from the actual biology. Here, of course, we do not advocate comparison of the actual material, but just of the behavior (as is the case for Turing's original test for machine-generated intelligence). However, in contrast to Turing's original test, falsifying the model here is something that we actually strive for, since it is a wonderful way to encourage further research.

The *engineering method* starts by producing a model (for example, a blueprint, or a specification) of what we want to build, and proceeds by building it. We then aim to show that what we built is in fact an implementation of the model. Such a verification process compares the outcomes of the system to the predicted outcomes of the model, by testing and model checking, which is in many ways similar to scientific experimentation. If this is unsuccessful, it means that we have discovered a property of the model that is not correctly implemented by the system, which may lead to an improved system and a new verification cycle. If it is successful, the system stands, which does not mean that it is correct: it simply means that we have not yet found the next bug (and we should keep trying, by making the model/specification more complete). A main feature of the engineering method is that the "truth" is in the model, while the system is in principle never fully correct. Thus, science and engineering work in opposite directions.

Incidentally, *reverse engineering*, the process of deriving an unknown model from an existing system, follows very much the scientific method. Conversely, (direct) engineering could be also called reverse science. As an example of the difference within a single discipline, consider systems biology, which is largely a scientific enterprise, as opposed to synthetic biology, which is largely an engineering enterprise. Of course, there are strong interactions between science and engineering, with one inspiring the other. Many engineered systems are inspired by biological systems that have been scientifically investigated (for example, genetic algorithms, neural networks), and conversely, as we have argued, modeling biological systems can be inspired by modeling techniques in engineering.

Indeed, when considering complex systems, both in science and in engineering one is usually in a position where the model is so complex that it is in constant flux, and where new knowledge about the system is expanding so fast that it is difficult to tell what "the system" actually is. In these situations, what emerges is a joint iterative method, in which our understanding of the system continuously improves, as a result of the modeling being continuously refined, which is turn is done by discovering the discrepancies between system and model; and all this in an endless cycle. This situation is actually quite common in software engineering, possibly more than in any other branch of engineering, where the model (the specification) typically evolves while the system (the code) is being built. And this situation is also quite common in modern biology, where scientific discovery is closely coupled with the construction of artificial systems, for example, by genetic engineering, so that not even nature is taken as a given. In such complex situations we must combine the conflicting views of systems and models into a wider *scientific-engineering method*, which still works by two opposite cycles. Discovery can still be coupled with falsification (when starting from the system) and construction can still be coupled with verification (when starting from the model), but there is no longer a privileged starting point for the process. In this sense, computing and biology are already remarkably close to each other, in the kind of general methods they use to expand knowledge.

Modeling a complete organism. We feel that it might be beneficial to use the ideas and methods discussed here to model a complete biological system. In fact, a "grand challenge" of modeling a full multicellular organism has been proposed,[39] motivated by the belief that unprecedented depth of understanding life and its mechanisms will result from such a model. The dream is to model the organism as a reactive system, the backbone of which would be its multitude of cells and their interactions, but to include the relevant inner behavioral aspects of the cell on the molecular and biochemical level as well. The 1000-cell *Caenorhabditis elegans* nematode worm, better known simply as *C. elegans*, was suggested in Harel[39] as a possible system to model.

Obviously, this is less ambitious than modeling, say, the entire population of a species, and more ambitious than modeling a mere cell. The choice of which system to address is a matter of taste, but our feeling is that an organism would be a good compromise that would yield enormous benefits, if it can indeed be done satisfactorily. The question of when to stop, that is, when is the model deemed valid or complete, is a very interesting one, and we have proposed that the Turing test mentioned previously could be a good first approximation: We are done when the model's behavior cannot be distinguished from that of the real thing, in which case the model can be said to be a theory of the organism; see Harel.[40]

This *whole organism project* (WOP) would take many years of work, and would entail using a variety of methods and to interconnect them all smoothly into a full, true-to-all-known-facts, 4-dimensional model of the creature. We would want the model to be easily modifiable and extendable as new facts are discovered, to have an animated, anatomically correct front-end, which would have to be tightly linked to a reactive system model of the organism. The front-to-back linking could be done using the idea of *reactive animation*.[24] Most importantly, the model would enable realistic simulation of the organism's development and behavior (this is the fourth dimension), and would lend itself to the kinds of analysis techniques discussed earlier. All of this could help uncover gaps, correct errors, suggest

new experiments, and help predict unobserved phenomena. More generally, the expectation is that it would allow researchers to see and understand the organism, its development, and its behavior in ways not otherwise possible.

Of course, this idea might be far too vast to be practical, but it seems worthy of consideration, if only as a very distant holy grail of sorts, toward which it would be beneficial to aspire.

Challenges for computer science. The research directions described in this article are intended, first and foremost, to yield beneficial results in biology and medicine, thus enhancing our ability to improve our lives. The central challenges they raise are also biological in nature, involving the need for biology to become a more formal, precise, and quantitative science, and the need for acquiring and consolidating sufficient information about the biology of interest to model it as a reactive system. This is especially true of the WOP and the work that is necessary to lead up to it. However, most readers of this article are computer scientists, who will be primarily interested in the new challenges this area of work raises for computer science, and in the benefits it can yield "at home." The two are linked, of course: once our field rises to the relevant challenges, the new ideas that are found to work well in the modeling of complex biological systems will benefit the development of human-made computerized software and systems as well. So, what are the main challenges for computer science? What new ideas are needed, and what kinds of extensions should be sought for the methods used in the modeling efforts mentioned earlier?

Our feeling is that we need ways to build models that seamlessly combine qualitative and quantitative data, and which come with appropriately powerful analysis methods. And we need to find ways to make our models more robust and less sensitive to faults and gaps in the available data. In other words, not only biology needs to become a more quantitative science, also computer science needs to become more quantitative. Formal methods have excelled in structuring and handling large, complex discrete systems, but we have neglected the incorporation of quantitative data. Similarly, we need to move our focus away from Boolean properties of systems, such as correctness (which really has no meaning in biology), toward quantitative properties such as fitness, robustness, and resilience. We believe the study of such quantitative properties will greatly benefit computer science itself which, as an engineering discipline, ought to have ways of expressing and measuring quantitative preferences between different implementations of a system, and estimating their reliability, cost, and performance. Preliminary ideas in this direction can be found in Cerny et al.[13] Needless to say, by studying biological systems in this way, we may also learn a thing or two about building more adaptive and robust software and hardware systems.

Two major deficiencies of current reactive models that need to be researched thoroughly are *genericity* and *linkage*. Genericity is related to inheritance, but for temporal, reactive behavior. We would like to have a generic model of, say, a cell or a central intracell substance, and be able to specialize it to specific types of cells or substances in a relatively painless way; see Amir-Kroll et al.[3] for a preliminary attempt at this. As far as linkage is concerned, we attach great importance to developing means for linking heterogeneous parts of biological models both horizontally and vertically, to yield compound models that can be seamlessly visualized, executed, and analyzed. *Horizontal linkage* refers to compositionality—the ability to compose side-by-side parts of the desired model into a whole, which is a particular challenge when the individual parts have different execution semantics,[72] an issue that is central also to embedded-systems design.[46] *Vertical linkage* is related to abstraction, and is the ability to link higher levels of the model with lower levels, for example, models of the intracellular pathway and network information with models of the reactive intercellular effects.[52] Ideally, we hope to provide biologists with computational "experimentation environments," where they can effortlessly play in a cycle of changing the model and looking at the resulting behaviors, all the time zooming in and out between different levels. We believe that such experimentation environments will go a long way toward efficiently identifying new, interesting hypotheses, testing them first "in-silico," and ultimately comparing them with nature.

We hope this article will help increase interest, within the computer science community, in the process of modeling and analyzing biological systems, viewing them as reactive systems of the most complex and challenging kind. The potential benefits of this, we feel, are difficult to overestimate, and we believe that concepts and ideas from software and systems engineering can form the basis of such work. Computer science is thus poised to play a role in the science of the 21st century, which will be dominated by the life sciences, similar to the role played by mathematics in the science of the 20th century, much of which was dominated by the physical sciences.

Acknowledgments

We would like to thank our past collaborators on these topics, for the wisdom and ideas they have contributed to us over the years. They include Luca Cardelli, Yaron Cohen, Sol Efroni, Walter Fontana, Alex Hajnal, Jane Hubbard, Na'aman Kam, Maria Mateescu, Nir Piterman, Yaki Setty, Michael Stern, Verena Wolf, and the late Amir Pnueli.

This research was supported in part by the ERC Advanced Grant LIBPR (Liberating Programming) awarded to DH, by the John von Neumann Minerva Center for the Development of Reactive Systems at the Weizmann Institute of Science, and by the ERC Advanced Grant QUAREM (Quantitative Reactive Modeling), awarded to TAH.

References

1. Alur, R. and Dill, D. Automata for modeling real-time systems. In *Proc. Int. Conf. Automata, Languages, and Programming 17* (1990), 322–335.
2. Alur, R. and Henzinger, T.A. Reactive modules. *Formal Methods in System Design 15*, 7 (1999), 48.
3. Amir-Kroll, H., Sadot, A., Cohen, I.R. and Harel, D. GemCell: A generic platform for modeling multi-cellular biological systems. *Theoret. Comput. Sci. 391*, 3 (2008), 276–290.
4. Arkin, A., Ross, J. and McAdams, H.H. Stochastic kinetic analysis of developmental pathway bifurcation in phage lambda-infected escherichia coli cells. *Genetics 149* (1998), 1633–1648.
5. Baker, M.D., Wolanin, P.M. and Stock, J.B. Signal transduction in bacterial chemotaxis. *Bioessays 28* (2006), 9–22.
6. Barjis, J. and Barjis, I. Formalization of the protein production by means of petri nets. *Proc. Int. Conf. on Information Intelligence Systems* (1999), IEEE.
7. Batt, G., Ropers, D., de Jong, H., Geiselman, J., Mateescu, R., Page, M., and Schneider, D. Validation

of qualitative models of genetic regulatory networks by model checking: Analysis of the nutritional stress response in Escherichia coli. *Bioinformatics 21* (2005), 19–28.
8. Blinov, M.L., Faeder, J.R., Goldstein, B. and Hlavacek, W.S. BioNetGen: Software for rule-based modeling of signal transduction based on the interactions of molecular domains. *Bioinformatics 20* (2001), 3289–3291.
9. Calder, M., Vyshemirsky, V., Gilbert, D. and Orton, R. Analysis of signaling pathways using the prism model checker. In *Proc. 3rd International Conference on Computational Methods in Systems Biology*. G. Plotkin, ed. (Edinburgh, Scotland, 2005), 179–190.
10. Calzone, L., Fages, F. and Soliman, S. BIOCHAM: An environment for modeling biological systems and formalizing experimental knowledge. *Bioinformatics 22*, 14 (2006), 1805–1807.
11. Cardelli, L. Brane caluli. In *Proc. Computational Methods in Systems Biology* (Paris, May 26, 2004) V. Danos and V. Schächter, eds. LNCS 3082, 257 (2004).
12. Cardelli, L. Abstract machines of systems biology. In *Transactions on Computational Systems Biology III*. C. Priami et al. eds. LNCS 3737, (2005), 145–168.
13. Cerny, P., Henzinger, T.A. and Radhakrishna, A. Simulation distances. In *Proc. Concurrency Theory 2010*. LNCS 2011, 253–268.
14. Clarke, E., Grumberg, O., Jha, S., Lu, Y. and Veith, H. Counterexample-guided abstraction refinement for symbolic model checking. *J. ACM 50*, (2003), 752–794.
15. Clarke, E., Grumberg, O. and Peled, D. *Model Checking*, MIT Press, 2000.
16. Cohen, I.R. and Harel, D. Explaining a complex living system: Dynamics, multi-scaling and emergence. *J. Royal Society Interface 4*, (2007), 175–182.
17. Cousot, P. and Cousot, R. Abstract interpretation. In *Proc. Symp. Principles of Programming Languages 4*, (1977), 238–252.
18. Curti, M., Degano, P., Priami, C. and Baldari, C. Modeling biochemical pathways through enhanced pi-calculus. *Theor. Comput. Sci. 325*, (2004), 111–140.
19. Damm, W. and Harel, D. LSCs: Breathing life into message sequence charts. *Formal Methods in System Design 19*, 1, (2001), 45–80.
20. Danos, V., Feret, J., Fontana, W., Harmer, R. and Krivine, J. Rule-based modelling of cellular signalling. *Concur* 2007, LNCS 4703, 17–41.
21. Didier, F., Henzinger, T.A., Mateescu, M. and Wolf, V. Approximation of event probabilities in noisy cellular processes. *Computational Methods in Systems Biology 7*, (2009) 173–188.
22. Didier, F., Henzinger, T.A., Mateescu, M. and Wolf, V. Fast adaptive uniformization of the chemical master equation. *High-Performance Computational Systems Biology 1*, (2009).
23. Dill, D.L., Knapp, M.A., Gage, P., Talcott, C., Laderoute, K. and Lincoln, P. The pathalyzer: A tool for analysis of signal transduction pathways. In *Proc. 1st Annual Recomb Satellite Workshop on Systems Biology*, 2005.
24. Efroni, S., Harel, D. and Cohen, I.R. Reactive animation: Realistic modeling of complex dynamic systems. *Computer 38*, (2005), 38–47.
25. Efroni, S., Harel, D., and Cohen, I.R. Emergent dynamics of thymocyte development and lineage determination. *PLoS Computational Biology 3*, 1 (2007), 127–136.
26. Elowitz, M.B., Levine, A.J., Siggia, E.D. and Swain, P.S. Stochastic gene expression in a single cell. *Science 297*, (2002), 1183–1186.
27. Emerson, E.A. Temporal and modal logic. In *Handbook of Theoretical Computer Science*, (1990), 995–1072.
28. Faeder, J.R., Hlavacek, J.S., Reischl, I., Blinov, M.L., Metzger, H., Redondo, A., Wofsy, C. and Goldstein, B. Investigation of early events in FcεRI-mediated signaling using a detailed mathematical model. *J. Immunol. 170*, (2003), 3769–3781.
29. Fedoroff, N. and Fontana, W. Small numbers of big molecules. *Science 297*, (2002), 1129–1131.
30. Feret, J., Danos, V., Krivine, J., Harmer, R. and Fontana, W. Internal coarse-graining of molecular systems. *Proc. Natl. Acad. Sci. 106*, 16, (2009), 6453–6458.
31. Fisher, J., Piterman, N., Hajnal, A., and Henzinger, T.A. Predictive modeling of signaling crosstalk during C. elegans vulval development. *PLoS Computational Biology 3*, 5, (2007) 92.
32. Fisher, J. and Henzinger, T.A. Executable cell biology. *Nat. Biotechnol. 25*, 11, (2007), 1239–49.
33. Fisher, J., Henzinger, T.A., Mateescu, M. and Piterman, N. Bounded asynchrony: A biologically-inspired notion of concurrency. In *Proc. of FMSB '08 Cambridge, UK*. Springer, 2008.
34. Fisher, J., Piterman, N., Hubbard, E.J., Stern, M.J. and Harel, D. Computational insights into Caenorhabditis elegans vulval development. In *Proc. Natl. Acad. Sci. USA 102*, (2005), 1951–1956.
35. Ghosh, R. and Tomlin, C.J. Symbolic reachable set computation of piecewise affine hybrid automata and its application to biological modeling: Delta-Notch protein signaling. *IEE Transactions on Systems Biology 1*, 1, (2004), 170–183.
36. Gillespie, D.T. Exact stochastic simulation of coupled chemical reactions. *J. of Physical Chemistry 81*, (1977), 2340–2361.
37. Gillespie, D.T. Markov Processes. 1992: Academic Press.
38. Harel, D. Statecharts: A visual formalism for complex systems. *Sci. Comput. Programming 8*, (1987), 231–274.
39. Harel, D. A grand challenge for computing: Full reactive modeling of a multi-cellular animal. Bulletin of the EATCS 81, (2003), 226-235. (Reprinted in *Current Trends in Theoretical Computer Science: The Challenge of the New Century, Algorithms and Complexity*, Vol I. G. Paun et al. eds. World Scientific, (2004), 559–568.
40. Harel, D. A Turing-like test for biological modelling. *Nature Biotechnology 23*, (2005), 495–496.
41. Harel, D. and Gery, E. Executable object modeling with statecharts. *Computer 30*, 7, (July 1997). IEEE Press, 31–42.
42. Harel, D. and Pnueli, A. On the development of reactive systems. In *Logics and Models of Concurrent Systems*. K.R. Apt, ed. NATO ASI Series, Vol. F-13, Springer-Verlag, NY, (1985), 477–498.
43. Haugh, J.M., Schneider, I.C. and Lewis, J.M. On the cross-regulation of protein tyrosine phosphatases and receptor tyrosine kinases in intracellular signaling. *J. Theor. Biol. 230*, (2004), 119–132.
44. Henzinger, T.A., Jobstmann, B. and Wolf, V. Formalisms for specifying Markovian population models. In *Proc. 3rd Int. Workshop on Reachability Problems*. LNCS 5797, Springer, 2009.
45. Henzinger, T.A., Mateescu, M., Mikeev, L. and Wolf, V. Hybrid numerical solution of the chemical master equation. In *Proc. 8th Int. Conf. on Computational Methods in Systems Biology*. Lecture Notes in Bioinformatics, Springer, 2010.
46. Henzinger, T.A., and Sifakis, J. The discipline of embedded systems design. *IEEE Computer 40*, 10, (2007), 36–44.
47. Hlavacek, W.S., Faeder, J.R., Blinov, M.L., Posner, R.G., Hucka, M., and Fontana, W. Rules for modelling signal transduction systems. *Science STKE* 2006/344/re6.
48. Kam, N., Harel, D. and Cohen, I.R. *Visual Languages and Formal Methods*. (Stressa, Italy, Sept. 5–7, 2001). IEEE, 2001.
49. Kam, N., Kugler, H., Marelly, R., Appleby, L., Fisher, J., Pnueli, A., Harel, D., Stern, M.J. and Hubbard, E.J.A. Scenario-based approach to modeling development: A prototype model of *C. Elegans* vulval cell fate specification. *Developmental Biology 323* (2008), 1–5.
50. Kwiatkowska, M., Norman, G. and Parker, D. PRISM: Probabilistic symbolic model checker. In *Proc. TOOLS 2002*. T. Field et al. eds. LNCS 2324, (2002), 200–204.
51. Kwiatkowska, M., Norman, G., Parker, D., Tymchyshyn, O., Heath, J., and Gaffney, E. Simulation and verification for computational modeling of signalling pathways. In *Proc. Winter Simulation Conference* (Monterey, CA, Dec. 2–6, 2006). IEEE, 2006, 1666–1674 .
52. Kugler, H., Larjo, A. and Harel, D. Biocharts: A visual formalism for complex biological systems. *J. Royal Society Interface*, 2010.
53. Lee, K.H., Dinner, A.R., Tu, C., Campi, G., Raychaudhuri, S., Varma, R., Sims, T.N., Burack, W.R., Wu, H., Wang, J., Kanagawa, O., Markiewicz, M., Allen, P.M., Dustin, M.L., Chakraborty, A.K. and Shaw, A.S. The immunological synapse balances T cell receptor signaling and degradation. *Science 302*, (2003), 1218–1222.
54. Li, Q.J., Dinner, A.R., Qi, S., Irvine, D.J., Huppa, J.B., Davis, M.M., and Chakraborty, A.K. CD4 enhances T cell sensitivity to antigen by coordinating Lck accumulation at the immunological synapse. *Nat. Immunol. 5*, (2004)791–799.
55. McAdams, H.H. and Arkin, A. Stochastic mechanisms in gene expression. *Proceedings of the National Academy of Science 94*, (1997), 814–819.
56. McAdams, H.H. and Arkin, A. It's a noisy business! *Trends in Genetics 15*, 2 (1999), 65–69.
57. Milner, R. *A Calculus of Communicating Systems*. Springer Verlag, 1980
58. Milner, R. Operational and algebraic semantics of concurrent processes. *Handbook of Theoretical Computer Science B*, (1990), 1201–1242.
59. Milner, R. *Communicating and Mobile Systems: The pi-Calculus*. Cambridge University Press, Cambridge, UK, 1999).
60. Parkinson, J.S., Ames, P. and Studdert, C.A. Collaborative signaling by bacterial chemoreceptors. *Curr. Opin. Microbiol. 8*, (2005), 116–121.
61. Paulsson, J. Summing up the noise in gene networks. *Nature 427*, (2004), 415–418.
62. Peterson, J.L. *Petri Net Theory and the Modeling of Systems*. Prentice Hall 1981
63. Pnueli, A. The temporal logic of programs. In *Proc. Symp. Found. Computer Science*, (1977) 46–57.
64. Pnueli, A. and Rosner, R. On the synthesis of a reactive module. In *Proc. Symp. Principles of Programming Languages 16*, (1989), 179–190.
65. Popper, K. *The logic of scientific discovery*. Hutchinson, London, 1959.
66. Priami, C. The stochastic pi-calculus. *Comp. J. 38*, (1995), 578–589.
67. Priami, C. Algorithmic systems biology. *Comm. ACM 52*, 5, (May 2009) 80–88.
68. Priami, C., Regev, A., Shapiro, E.Y., and Silverman, W. Application of a stochastic name passing calculus to representation and simulation of molecular processes. *Inf. Process. Lett. 80*, (2001) 25–31.
69. Regev, A., Panina, E.M., Silverman, W., Cardelli, L., and Shapiro, E.Y. Bioambients: An abstraction for biological compartments. *Theor. Comput. Sci. 325*, (2004), 141–167.
70. Regev, A., Silverman, W., and Shapiro, E. Representation and simulation of biochemical processes using the pi-calculus process algebra. *Pac. Symp. Biocomput.* (2001) 459–470.
71. Sadot, A., Fisher, J., Barak, D., Admanit, Y., Stern, M.J., Hubbard, E.J.A. and Harel, D. Towards verified biological models, *Transactions on Computational Biology and Bioinformatics 5*,(2008), 1–12.
72. Schaub, M.A., Henzinger, T.A., and Fisher, J. Qualitative networks: A symbolic approach to analyze biological signaling networks. *BMC Systems Biology 1*, (2007).
73. Setty, Y., Cohen, I.R., Dor, Y., and Harel, D. Four-dimensional realistic modeling of pancreatic organogenesis. In *Proc. Natl. Acad. Sci. 105*, 51 (2008), 20374–20379.
74. Shen, X., Collier, J., Dill, D., Shapiro, L., Horowitz, M. and McAdams, H.H. Architecture and inherent robustness of a bacterial cell-cycle control system. *PNAS. 105*, 32 (2008), 11340–11345.
75. Thomas, W. Automata on infinite objects. In *Handbook of Theoretical Computer Science B*, (1990), 133–192.
76. Wang, D., Cardelli L., Phillips A., Piterman N. and Fisher J. Computational modeling of the EGFR network elucidates control mechanisms regulating signal dynamics. In *BMC Systems Biology 3*, 118, (2009), 22.
77. Wolf, V., Goel, R., Mateescu, M. and Henzinger, T.A. Solving the chemical master equation using sliding windows. *BMC Systems Biology 4*, 42, (2010).

Jasmin Fisher (Jasmin.Fisher@microsoft.com) is a researcher at Microsoft Research Cambridge in the Programming Principles and Tools Group, Cambridge, UK.

David Harel (dharel@weizmann.ac.il) is the William Sussman Professorial Chair of the Dept. of Computer Science and Applied Mathematics at the Weizmann Institute of Science, Rehovot, Israel.

Thomas A. Henzinger (tah@ ist.ac.at) is president of the Institute of Science and Technology Austria (IST Austria) and an adjunct professor of electrical engineering and computer sciences at the University of California, Berkeley.