Symbolic Systems Biology: Theory and Methods, Jones and Bartlett Publishers 2010

On Statecharts for Biology

Jasmin Fisher and David Harel

Biology as Reactivity

One of the central issues in software and system engineering over the last couple of decades has been to develop languages, methods and tools for the reliable construction of *reactive systems*, those whose complexity stems not necessarily from complicated computation but from complicated reactivity over time (Harel and Pnueli, 1985). Such systems are usually highly concurrent and time-intensive, and can exhibit hybrid behavior that is predominantly discrete in nature but has continuous and stochastic aspects too. The structure of these systems consists of many interacting, often distributed, components. Very often the structure itself is dynamic, with its components being repeatedly created and destroyed during the system's life span. The heart of the problem is the need for good approaches to modeling and analyzing the dynamic behavior of such systems.

In the last decade or so, we and others have carried out work on viewing biological systems as reactive systems, and on modeling and analyzing a variety of sample pieces of biology. This approach is based on three premises (Harel, 2003): (i) that satisfactory frameworks exist in software and systems engineering for reactive system modeling and design; (ii) that biological research is ready for an extremely significant transition from analysis (reducing experimental observations to elementary building blocks) to synthesis (integrating the parts into a comprehensive whole), a transition that requires mathematics and computation; and (iii) that the true complexity of biological systems --- specifically multi-cellular living organisms --- stems from their reactivity.

As far as the third of these goes, biological systems exhibit the characteristics of reactive systems remarkably, and on many levels; from the molecular, via the cellular, and all the way up to organs, full organisms, and even entire populations. It doesn't take much to observe within such systems the heavy concurrency, the event-driven discrete nature of the behavior, the chain-reactions and cause-effect phenomena, the time-dependent patterns, etc. From these three premises there follows a thesis, prompting much of our own work in this area: that biological systems can be modeled beneficially as reactive systems, using languages and tools developed in computer science for the construction of man-made systems. One of the main languages we have used to do the modeling is that of *Statecharts* (Harel, 1987; Harel and Gery 1997), and the present paper is aimed at discussing some of this work and its ramifications.

Our claim is that biological systems, just like many engineered systems, are complex reactive systems – interacting and responding to the environment

and to other components of the system (Harel, 2003). The main efforts on modeling reactive biological systems with the aid of Statecharts include the following: Kam *et al.* (Kam et al., 2001) have modeled T-cell activation, and Efroni *et al.* (Efroni et al., 2003) have modeled T-cell development in the thymus. Following work on a model of cell fate determination during *C. elegans* vulval development in Kam *et al.* (Kam et al., 2003), which used a somewhat different language (live sequence charts), statecharts were used to model certain advanced aspects of that system in Fisher *et al.* (Fisher et al., 2005; Fisher et al., 2007) and Sadot *et al.* (Sadot et al., 2008). Swerdlin *et al.* have modeled the dynamics of the lymph node (Swerdlin *et al.* , 2008), and Setty *et al.* have modeled the morphogenisis of the pancreas (Setty *et al.*, 2008).

To help see how reactivity is indeed relevant, here is how the reactivity of biology has been described recently:

A reactive system, in contrast to a transformational system, does not behave according to a pre-programmed chain of linked instructions. Rather, such a system reacts in parallel to many concurrent inputs, and its behaviors, outputs and effects, are not just a function of the values of its inputs but also of their variety, of the order in which they arrive, of their timing, of their arrival speeds and so forth. A biological system expresses a dynamic narrative in which the DNA code is one of the many formative inputs. Structural proteins, enzymes, carbohydrates, lipids, intracellular signals, hormones and other molecules play key roles in forming and informing the system. The environment of the living system is a most critical source of information (Cohen 2006). True, DNA serves as a special repository of information because it is replicated and transmitted across generations, but DNA is meaningless without the proteins and other molecules that selectively activate segments of the DNA sequence in variable and alternative ways to create genes. The activation of specific genes emerges from the dynamic state of the cell. One could argue that DNA is just as much a servant of the cell's state as it is the cell's master; there is no hierarchical master plan (Cohen & Atlan 2006).

Note that, unlike a transformational system, a reactive system does not seek equilibrium, has no set point and no state of rest. A reactive system holds itself together as a system just by reacting. A reactive system succeeds not by reaching homeostasis; a brain in homeostasis is clinically dead. A reactive system succeeds by being both robust and resilient. The reactive system responds to simultaneous perturbations and continues to survive, thanks to its reactive dynamics. It is true that organisms feature sub-systems that can be described nicely by homeostatic principles; some examples are the thermoregulatory system that maintains our internal temperature at 37°C, our water balance system and blood glucose regulation. But these systems act only like the transformational systems when viewed as a whole; internally, each of these systems is fashioned by collectives of concurrently reactive, never-resting cells (Cohen 2000).

Visual Languages for Reactivity

We should emphasis from the start that dynamic reactive behavior is the main issue of concern in this paper, and we will have nothing much to say about how the structural non-dynamics aspects of a biological system are to be captured.

One of the most widely used ideas that have proven relevant to the process of specifying the dynamic behavior of reactive systems, is that of *visual formalisms* (Harel 1988). These are languages that are both graphically intuitive and mathematically rigorous, and many are supported by powerful tools that enable full model executability, and code generation. The models they give rise to are linkable to GUIs and other structural descriptions of the system under development. This enables realistic simulation prior to actual implementation. At present, such languages and tools --- recently many of these are based on the *object-oriented* paradigm --- are being strengthened by verification modules, making it possible not only to execute and simulate the system models (test and observe) but also to verify dynamic properties thereof (prove). They are also being linked to tools for dealing with the system's continuous aspects in a full hybrid fashion. Most of the tools are state-based, encouraging intra-object style specification, and within these the language of choice for reactive behavior is most often *Statecharts* (Harel 1987). One of the most powerful and versatile of these tools is *Rhapsody* (Harel & Gery 1987).

We should add that the fact that we concentrate in visual formalisms does not mean that there are no other approaches. There has been, for example, a tremendous amount of extremely interesting work done on using non-visual approaches to modeling reactive systems, among which are temporal logic, process calculi and process algebras, etc. In many cases these have also been used successfully in modeling biology; see, e.g., (Cardelli, 2004; Kwaitkowska et al., 2006; Priami and Quaglia, 2004; Regev, Silverman and Shapiro, 2001).

Another important point when we consider computational tools for modeling biology is that they must support ways to represent models and data that are natural and can become standard. User friendliness, flexibility and visuality are critical to integrating computational tools into experimental biology research. Biologists are not expected to become engineers, so that one of the challenges facing computer scientists is to design modeling languages and software tools that are more accessible to non-experts.

On Statecharts

There are actually two dual approaches to modeling reactivity, intra-object and inter-object, and in the realm of visual formalisms they manifest themselves in the form of two families of languages, state-based and scenario-based (Damm and Harel, 2001; Harel and Marelly 2003). We will use *Statecharts* and *Live Sequence Charts* (LSCs) as representatives of these. They are both visual languages with clear and rigorously defined syntax and semantics, and which

enable the construction and execution of formal models. Statecharts specify the full state-based behaviours of each object in the system (e.g., a cell), whereas LSCs specify the multimodal (e.g., possible, necessary and forbidden) scenarios that link together the objects by their possible sequences of behaviour.

In this paper we concentrate on statecharts, and on the intra-object approach. The main reason is in the fact that the use of states, or modes, appears to be a very natural way to view biological dynamics. In a state, part of the system (e.g., a cell) stays put, remains, dwells for some non-zero amount of time, to move on only as a result of the occurrence of some event (arrival of a signal, change in some relevant quantity or value, etc.). Using scenarios or pathways does not necessarily have an explicit notion of staying put, being, dwelling, residing, etc.

Statecharts define behavior using a hierarchy of states with transitions, events, and conditions (Harel, 1987). Classical finite-state machines (FSMs) and their state transition diagrams are extended by a semantically meaningful hierarchical sub-stating mechanism and by a notion of orthogonal simultaneity. Both of these are reflected in the graphics themselves, the hierarchy by encapsulation and the orthogonality by adjacent portions separated by a dashed line. Orthogonal components can cooperate and know about each other by several means, including direct sensing of the state status in another component or by actions. The cooperation mechanism – within a single statechart – has a broadcasting flavor. It helps to note that capsulated sub-states represent OR (actually this is XOR; exclusive or), and orthogonality is AND.

Transitions become far more elaborate and rich than in conventional FSMs. They can start or stop at any level of the hierarchy, can cross levels, and in general can be hyperedges, since both sources and targets of transitions can contain sets of states. At any given point in time a statechart will be in a vector, or combination, of states, whose length is not fixed. Exiting and entering orthogonal components on the various levels of the hierarchy continuously changes the size of the state vector. Default states generalize start states, and they too can be level-crossing and of hyperedge nature. And the language has history connectors, conditions, selection connectors, and more. A transition can be labeled with an event and optionally also with a parenthesized condition, as well as with Mealy-like outputs, or actions. (Actions can also occur within states, as in the Moore style.)

Using Rhapsody (or other similar tools) a Statechart model can be compiled into executable reactive code (for example, in Java or C++). At run-time, one state in each orthogonal component is active – which is the current state of this component. Thus, the state of an object is identified by the set of active states of all of its components. As the simulation advances, various events move the active state in each orthogonal component from one state to another. Concurrent execution is naturally captured by the running of the model, or alternatively in the executable code that is generated from it.

When using the object-oriented version of Statecharts (e.g., as supported by Rhapsody) the underlying structure of the objects gives them too a measure of modularity and hierarchy. One can add or remove objects, connections, states and transitions at any level. Statecharts also have ample means to specify rich

kinds of behavior, such as concurrency, stochasticity, chain reactions and time-dependent or time-constrained actions.

Statecharts for Biology

Starting in the mid-1980s the second-listed author has often claimed that biological systems should be viewed as systems the way we know them in the world of computing and software engineering, and that biological modeling should be attempted using languages and tools constructed for developing reactive systems. As mentioned in the previous section, one obvious approach is to use states. State-based models define the behavior of objects (or other elements of the system under description) 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, and how they behave under different circumstances. Interacting state machine models are particularly suitable for describing mechanistic models of biological systems that are well understood qualitatively. Such models do not require much 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.

Usually, the state of an object is determined in part by the states of its sub-objects. For example, significant portions of the state of a cell would be determined by the states of the various genes and proteins comprising it. Each gene or protein would then have its own reaction to the presence or absence of some other molecules, and the change in the state of the cell would be influences greatly by the inter-dependent state changes of all parts. A hierarchical structure allows one to view a system at different levels of detail, and hierarchy in the behavioral description itself is no exception. Now, while there are many different languages to express interacting state machine models, Statecharts stand out, among other things, because of their support of hierarchy. In fact, a large part of the appeal of the language has been its hierarchical description capabilities. 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.

Another advantage of Statecharts compared to other state-based formalisms, such as Reactive Modules (Alur & Henzinger 1996), is the fact that it is visual. The user can draw states and state changes, and good supporting tools will be able to execute the resulting models, enabling relatively easy and intuitive programming even for non-specialists. Using the visual language of Statecharts, 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 (Kam et al., 2001). The initial T-cell model was followed by a more extensive animated model of T-cell differentiation in the thymus, carried out by Efroni et al. (Efroni et al., 2003,

Efroni et al., 2007). That project introduced an idea called *reactive animation* (RA) (Efroni et al., 2005), where a reactive system (e.g., built using Statecharts) drives the display of animation software to visualize the model. At run-time, the front-end displays the simulation continuously and provides the means to interact with it. Initially, RA was implemented in an ad-hoc fashion, that is, one reactive system engine to one animation tool. Recently, the technique has been upgraded, resulting in a generic platform that enables interaction between various tools such as multiple reactive engines, 3D animation, real-time analysis and more (Setty et al., 2008).

Reactive systems, especially when used in modeling biology, call our attention to their *emergent properties*. These are properties of the behavior of the system, taken as whole, which are not expressed by any one of the lower-scale components that comprise it. Life, for example, is an emergent property; none of the component molecules of a cell are alive, only a whole cell lives.

Following an argument made in (Cohen and Harel, 2007), one notices that emergence is difficult to define in biological terms. It is a matter of scale. A cell emerges from the structured interactions of the molecules that comprise the cell. But the cell is orders of magnitude larger than its component molecular interactions. Obviously, you cannot see a cell from the inside at the molecular scale. The cell emerges only when you step back—zoom out—and look at the cellular system at a scale appropriate to seeing an entire cell. The cell emerges from its component interactions at the scale at which the cell functions as an object, with its own capabilities, and its own interactions with other cells and molecules.

Actual Modeling Efforts

In some modeling projects carried out over the last few years, mainly in the Harel group at the Weizmann Institute of Science, we have illustrated the usefulness of statecharts to reason about various biological processes. We briefly describe these models, not necessarily in their chronological order.

Pancreatic organogenesis in the embryonic mouse

The Statechart model of pancreatic organogenesis in the embryonic mouse (Setty et al., 2008), consists of a concurrent execution of pancreatic cells, which leads to the formation of the unique 3D structure of the organ (Figure 1). The model was studied by comparing simulations against relevant experimental data, and it reproduced some genetic ablation experiments in silico. As an example, Figure 2 shows a histological cut of the pancreas (left) and the emerging structure in the model at approximately the same day (middle). On the right, the figure shows the result of an in silico experiment, in which the aorta was disabled, leading to a complete lose of structure. During the analysis of the pancreas model, concurrent execution of pancreatic cells was shown to give rise to a property that corresponds well with endocrinic clusters that appear early in the developing organ in vivo (Setty et al., 2008).

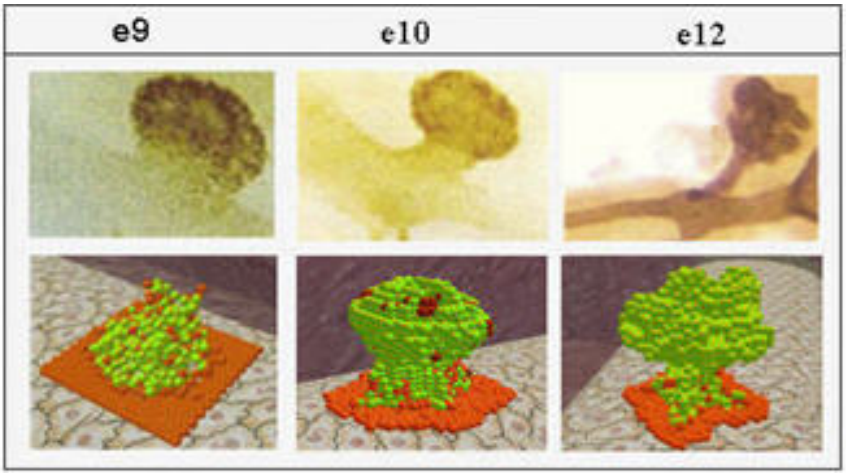

Figure 1. (From Setty et al., 2008) Pancreatic organogenesis in the embryonic mouse consists of a concurrent execution of pancreatic cells, which leads to the formation of the unique 3D structure of the organ.

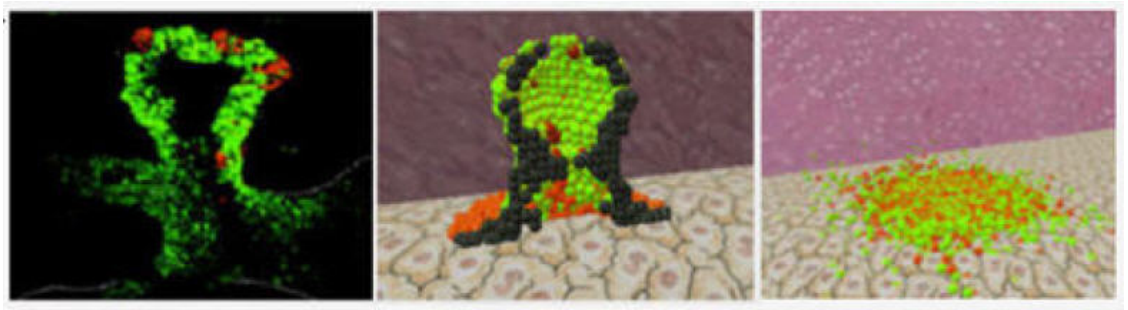

Figure 2. (From Setty et al., 2008) A histological cut of the pancreas (left) and the emerging structure in the model at approximately the same day (middle), and the result of an in silico experiment, in which the aorta was disabled, leading to a complete lose of structure (left).

Cell fate specification during *C. elegans* development

Following the LSC model of vulval fate specification in Kam et al. (Kam et al., 2003), Fisher et al. (Fisher et al., 2005) created a Statecharts model of vulval fate specification based on the proposed mechanistic model of Sternberg and Horvitz from 1989 (Sternberg & Horvitz 1989) 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. More recent work (Fisher et al., 2007) was based on the more sophisticated understanding of vulval fate specification that we have today. 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 also validated experimentally.

T cell activation and differentiation in the thymus

The other models relate to the immune system and simulate the development and function of T cells in the thymus (Efroni et al., 2003; Efroni et al., 2007). Figure 3 shows analysis of the thymus model; microscopy images of knockout (A) and wild type (B) thymus (left column), compared with the same area in the model (right column). On the right, Figure 3A shows an in silico experiment, in which the knockout of a single gene leads to a major change in the behavior of the cell population.

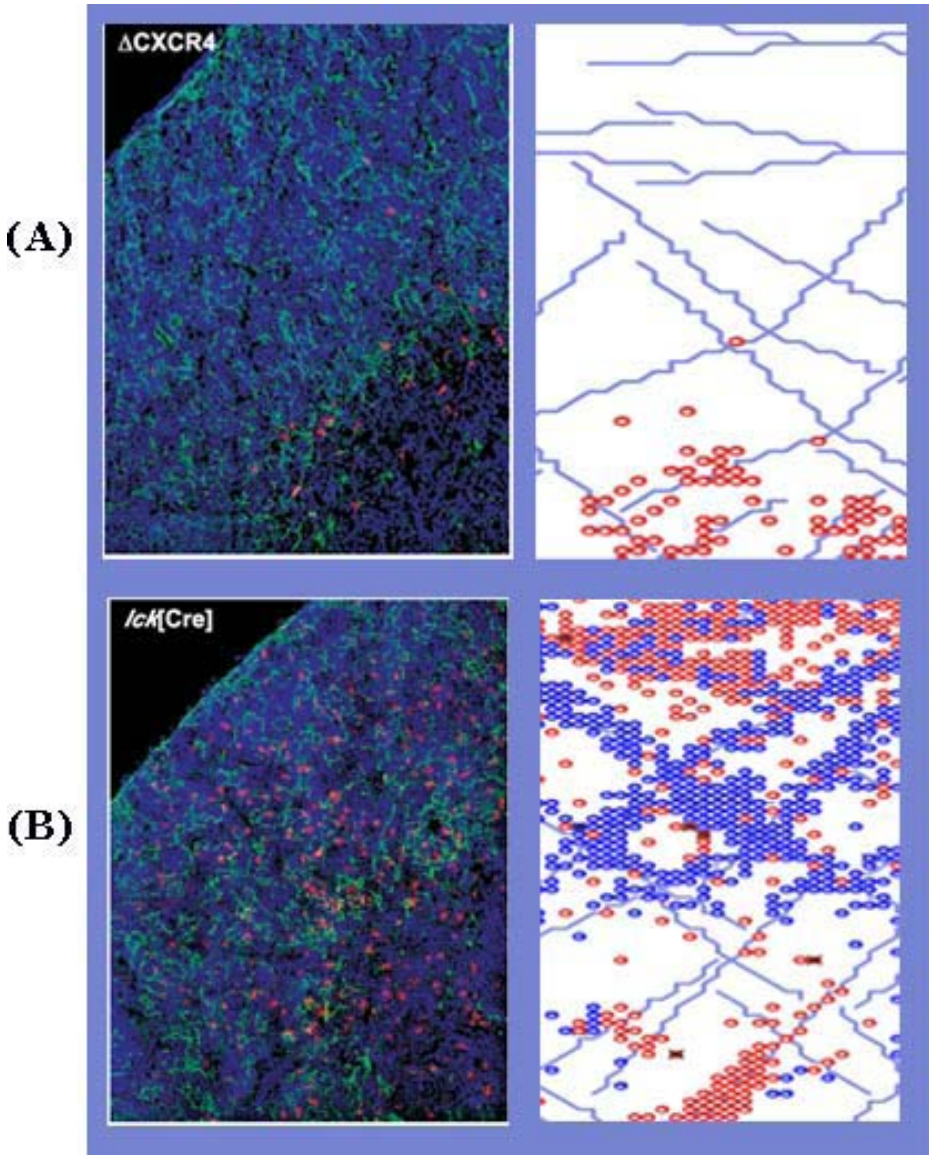

Figure 3. (From Efroni et al., 2007) The thymus model: microscopy images of knockout (A) and wild type (B) thymus (left column; from Plotkin et al., 2003), compared with the same area in the model (right column).

Analysis of the models also revealed several interesting emergent properties, which correspond well with biological phenomena. For example, the concurrent execution of T-cell development in the thymus led to the emergence of competitive behavior between the cells (Efroni et al., 2007). These properties were analyzed and studied, and were used to suggest some insights into the phenomena.

Development and function of B cells in the lymph node
This work consists of a Statechart and RA model of the lymph node; see Swerdlin et al. (Swerdlin et al., 2008). The effects of the amount of antigen, as well as the actual size of the lymph nod, on the emergent properties of lymphocyte dynamics were studied. The model emphasized differentiation and anatomic localization. The dynamic organization of the lymph node visualized by RA sheds new light on how the immune system transforms antigen stimulation into a highly sensitive, yet buffered response.

Generic Cell model
The most recent modeling work in the Harel group has resulted in a system call ed *GemCell*. (Amir-Kroll et al., 2008). It contains a generic statechart model of cell behavior, which captures the five main aspects of cell behavior (proliferation, death, movement, import and export). This generic model is coupled with a database of biological specifics (DBS), which holds the information about the specific cellular system, and which is expected to be filled in by biologists with the relevant expertise. Modeling a particular segment of biology involves setting up the DBS to contain data about the specific behaviors and responses of the kinds of cells in the system under description. During execution, statecharts read in the specific data and the combination runs just as described in the particular models above.

Exercise

Construct a statechart model for the Delta-Notch decision, as follows.

Consider a cell that is about to take a decision to adopt one of two fates – A or B. The decision is made according to the competition between two signalling pathways operating inside the cell and between neighbouring cells. The following proteins are of interest: *Notch* receptor and its ligand *Delta*, which signify the level of the two signalling pathways.

Here is how the cell behaves. The Notch pathway encourages the cell to adopt fate A while the Delta pathway encourages the cell to adopt fate B. In a normal cell, Notch starts in a low level which gradually increases. When Notch reaches a certain level (threshold) it forces the cell to adopt fate A. At the same time, Delta encourages the Notch in the neighbouring cells (via the Notch receptor on

neighbouring cells) and inhibits the Notch in the same cell. Delta starts on a low level and if not encouraged it decreases until it disappears. Delta may be encouraged by the level of Notch in the neighbouring cell (sensed by Delta). If Delta reaches a certain threshold it inhibits the Notch and causes the cell to adopt fate B.

The following behaviours are observed:
1. When a cell is run in isolation, the Notch should prevail and the cell should assume fate A.
2. When two cells are run in parallel either of them can assume fate A, in which case the other assumes fate B. There are rare cases where both cells assume fate A.
3. When one of the cells gets an external boost to the pathway, it is always the case that this cell adopts fate A and the other fate B.

Test your model on varying numbers of cells and see what configurations of fate (A and B) your model yields.

Acks

We would like to thank the many colleagues with whom we have worked on biological modeling, and especially those with whom the work was dominated by the use of Statecharts. Central among these are Irun R. Cohen, Sol Efroni and Yaki Setty.

Bibliography

R. Alur and T.A. Henzinger (1996), “Reactive Modules”, *Proc. 11th Ann. IEEE Symp. On Logic in Computer Science*, 207-218.

H. Amir-Kroll, A. Sadot, I.R. Cohen and D. Harel (2008), "GemCell: A Generic Platform for Modeling Multi-Cellular Biological Systems", *Theoret. Comput. Sci.* 391:3, 276-290.

L. Cardelli (2004), “Brane Calculi”, Proc. Computational Methods in Systems Biology (CMSB), 257-278.

I.R. Cohen (2000), *Tending Adam’s Garden: Evolving the Cognitive Immune Self*, Academic Press, London.

I.R. Cohen (2006), “Informational landscapes in art, science and evolution”, *Bull. Math. Biol.* 68, 1213–1229.

I.R. Cohen and H. Atlan (2006), “Genetics as explanation: limits to the human genome project”, In *Encyclopedia of Life Sciences*, pp. 1–7. Wiley, New York.

I.R. Cohen and D. Harel (2007), "Explaining a Complex Living System: Dynamics, Multi-scaling and Emergence", *J. Royal Society Interface* 4,175-182.

W. Damm and D. Harel (2001), "LSCs: Breathing Life into Message Sequence Charts", *Formal Methods in System Design* 19:1, 45-80.

S. Efroni, D. Harel and I R. Cohen (2003), "Towards Rigorous Comprehension of Biological Complexity: Modeling, Execution and Visualization of Thymic T Cell Maturation", *Genome Research* 13, 2485-2497.

S. Efroni, D. Harel and I.R. Cohen (2005), "Reactive Animation: Realistic Modeling of Complex Dynamic Systems", *IEEE Computer* 38:1, 38-47.

S. Efroni, D. Harel and I.R. Cohen (2007), "Emergent Dynamics of Thymocyte Development and Lineage Determination", *PLoS Computational Biology* 3:1, 127-136.

J. Fisher, N. Piterman, E.J.A. Hubbard, M.J. Stern and D. Harel (2005), "Computational Insights into *C. elegans* Vulval Development", *Proc. Natl. Acad. Sci.* 102:6, 1951-1956.

J. Fisher, N. Piterman, A. Hajnal and T.A. Henzinger (2007), "Predictive Modeling of Signaling Crosstalk during *C. elegans* Vulval Development", *PLoS Computational Biology*, 3(5):e92.

D. Harel (1987), "Statecharts: A Visual Formalism for Complex Systems", *Sci. Comput. Programming* 8, 231-274.

D. Harel (1988), "On Visual Formalisms", *Comm. Assoc. Comput. Mach.* 31:5, 514-530.

D. Harel (2003), "A Grand Challenge for Computing: Towards 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, Paun, Rozenberg and Salomaa, eds., World Scientific, pp. 559-568, 2004.)

D. Harel and E. Gery (1997), "Executable Object Modeling with Statecharts", Computer 30:7, IEEE Press, 31-42.

D. Harel and R. Marelly (2003), *Come, Let's Play: Scenario-Based Programming Using LSCs and the Play-Engine*, Springer-Verlag, 2003.

D. Harel and A. Pnueli (1985), "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, New York, pp. 477-498.

N. Kam, I.R. Cohen and D. Harel (2001), "The Immune System as a Reactive System: Modeling T Cell Activation with Statecharts", *Visual Languages and Formal Methods* (VLFM'01), part of *IEEE Symp. on Human-Centric Computing* (HCC'01), pp. 15-22.

N. Kam, D. Harel, H. Kugler, R. Marelly, A. Pnueli, E.J.A. Hubbard and M.J. Stern (2003), "Formal Modeling of C. elegans Development: A Scenario-Based Approach", *Proc. Int. Workshop on Computational Methods in Systems Biology* (ICMSB), Lecture Notes in Computer Science, Vol. 2602, Springer-Verlag, pp. 4-20, 2003. (Revised version in *Modeling in Molecular Biology* (G. Ciobanu and G. Rozenberg, eds.), Springer, Berlin, 2004, pp. 151-173.)

M.Z. Kwiatkowska, G. Norman, D. Parker, O. Tymchyshyn, J. Heath and E. Gaffney (2006), “Simulation and verification for computational modelling of signalling pathways”, *Proc. Winter Simulation Conference* , 1666-1674.

J. Plotkin, S.E. Prockop, A. Lepique and H.T. Petrie (2003), “ Critical role for CXCR4 signaling in progenitor localization and T cell differentiation in the postnatal thymus”, *J. Immunol*. 171, 4521-4527.

C. Priami and P. Quaglia (2004), “Modelling the dynamics of biosystems”, *Briefings in Bioinformatics* 5(3), 259-269.

A. Regev, W. Silverman and E.Y. Shapiro (2001), “Representation and Simulation of Biochemical Processes Using the pi-Calculus Process Algebra”, *Proc. Pacific Symposium on Biocomputing*, 459-470.

A. Sadot, J. Fisher, D. Barak, Y. Admanit, M.J. Stern, E.J.A Hubbard and D. Harel (2008), "Towards Verified Biological Models", *IEEE/ACM Trans. Comput. Biology and Bioinformatics*, 5(2):1-12, 2008.

Y. Setty, I. R. Cohen, Y, Dor and D. Harel (2008), "Four-Dimensional Realistic Modeling of Pancreatic Organogenesis", *Proc. Natl. Acad. Sci.* 105:51 (2008), 20374-20379.

P.W. Sternberg and H. R. Horvitz (1989), “The combined action of two intercellular signaling pathways specifies three cell fates during vulval induction in *C. elegans*, *Cell* 58, 679-93.

N. Swerdlin, I.R. Cohen and D. Harel (2008), "The Lymph Node B Cell Immune Response: Dynamic Analysis in-silico", *Proceedings of the IEEE*, special issue on Computational System Biology, in press.